ИНФОРМАТИКА

2004 **июль-сентябрь** **№3**

УДК 519.71

Д.И. Черемисинов

ФОРМАЛЬНЫЕ МЕТОДЫ АНАЛИЗА ПОВЕДЕНИЯ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Анализируются понятия параллелизма и асинхронности – ключевые понятия, лежащие в основе современных представлений о проблемах проектирования вычислительных устройств. При разработке вычислительных устройств широко применяются теоретические модели архитектуры и технологии, использующие концепцию распределенных мультиагентных систем, состоящих из агентов, поведение которых задается посредством протоколов. Рассматривается задача разработки описания поведения агентов как моделирование системы в рамках некоторого формализма, особое внимание уделяется трактовке понятия времени в формализмах, моделирующих проектируемую систему. Показано, что, несмотря на отсутствие явных ссылок, во всех формализмах время учитывается как порядок смены событий.

Введение

Любой распределенный алгоритм, задающий согласованное решение некоторой проблемы, принято называть протоколом [1]. Сложные задачи, для решения которых создаются современные информационно-вычислительные системы, требуют высокой производительности и надежности работы аппаратуры. Магистральный путь значительного улучшения этих характеристик состоит в использовании вычислительных сетей, основой функционирования которых являются протоколы обмена информацией между компонентами. Теория протоколов – это интенсивно развивающаяся отрасль прикладной математики. Библиография по этой проблеме уже насчитывает тысячи работ [1, 2].

В настоящее время не существует всеобъемлющего понятия «распределенная система», которое одинаково понимается всеми. Первоначально это понятие обозначало тип архитектуры вычислительных систем [3], затем оно стало широко применяться в теории дискретных устройств для обозначения систем, поведение которых характеризуется асинхронностью и параллелизмом [4]. Очевидно, различие трактовок отражает разный уровень абстракции при рассмотрении систем определенного типа. Далее это понятие будет использоваться в более абстрактном смысле.

Термин «протокол» используется в теории распределенных систем для обозначения как процедур, необходимых для осуществления связи в системе, так и используемых при этом форматов данных. Протокол – это программа для распределенной вычислительной системы, управляющая передачей сообщений [5]. В широкой трактовке протокол – это описание поведения одного исполнителя, участвующего в решении задачи, которая может быть решена только совместно несколькими исполнителями [6]. Еще более широко данный термин понимается, когда протокол рассматривается как последовательная запись поведения любого процесса [4]. В этой работе протокол обозначает описание поведения.

Вычислительная система на уровне протоколов рассматривается как композиция взаимодействующих компонентов, связанных каналами передачи данных, которые представляют собой среду взаимодействия компонентов. Каналы могут быть как физическими, так и логическими. Среда взаимодействия в виде совокупности логических каналов обычно рассматривается как структура, аналогичная исходной вычислительной сети (т. е. также состоит из объектов и среды взаимодействия), но более низкого уровня [7]. Двигаясь с уровня на уровень, в конце концов можно достичь описания системы, в котором роль каналов выполняет некоторая физическая среда передачи данных. Горизонтальное разбиение на уровни позволяет в обозримой форме иерархически описать функции управления в распределенной системе. Уровневая организация протоколов часто является следствием распределения функций управления по физическим компонентам распределенной системы.

1. Моделирование систем

Специфика теории протоколов как прикладной науки, обслуживающей нужды проектирования распределенных систем, состоит в перенесении внимания с внутренних процессов, происходящих в процессоре и памяти отдельного элемента, на внешнее поведение, описывающее непрерывное взаимодействие элементов со своим окружением. Основная особенность подхода, характерного для теории протоколов (то, что выделяет ее из других наук информатики), состоит в том, что распределенная система рассматривается как совокупность параллельно и асинхронно работающих компонентов, обменивающихся информацией друг с другом. Системы управления могут рассматриваться как частный вид распределенной системы, компонентами которой являются система управления и внешняя среда.

1.1. Трансформации моделей при проектировании дискретных устройств

При проектировании сложных систем используются два типа методов. В методах первого типа основой разработки служит набор физических компонентов, использовавшихся ранее для аналогичных проектов. Реальное изделие и описание его работы разрабатываются независимо друг от друга, и второе создается обычно после первого. В другой группе методов проектирование начинается с создания полного и непротиворечивого описания условий работы устройства. Проект реального изделия получается в результате работы с этим описанием. Тип проектирования, для которого характерен подобный стиль, называется проектированием сверху вниз в отличие от методов первой группы, когда проектирование ведется снизу вверх. В технологии проектирования снизу вверх упор делается на опыт и профессиональные знания проектировщика, а сама процедура проектирования в целом плохо поддается автоматизации. При проектировании в противоположном направлении, когда для описания условий работы использован формальный язык, перспективы автоматизации значительно лучше [8].

При выполнении разработки сверху вниз описание системы – ее спецификация – декомпозируется на компоненты, имеющие более подробное описание функционирования. Эти компоненты, в свою очередь, декомпозируются до тех пор, пока не будет достигнут такой уровень, в котором функционирование компонентов задано (уровень базиса реализации). В данном случае возникает проблема обеспечения условий, при которых спецификации нижнего уровня описывают функционирование системы, заданное в исходных спецификациях.

Иногда имеется возможность разработать алгоритм синтеза, который осуществляет декомпозицию на основе формальных спецификаций. В других случаях можно выполнить верификацию операции декомпозиции, выполненной вручную, т. е. формальными методами установить, удовлетворяет ли исходным спецификациям произведенная декомпозиция. Если в процессе проектирования системы исходили из недостаточно полных спецификаций или если формальную верификацию декомпозиции слишком трудно осуществить, то при оценке корректности процесса проектирования приходится отталкиваться от других предпосылок. Можно исходить не из исходных спецификаций, а из фундаментальных законов, выполнение которых совершенно необходимо для надежного и безошибочного функционирования системы. В литературе круг проблем, связанных с проверкой логической непротиворечивости описания системы или демонстрацией того, что описание обладает некоторыми желательными свойствами, называется «приемкой» системы, а термин «верификация» оставлен для проблемы соответствия реализации протокола исходной спецификации после выполнения декомпозиции [9].

Описание системы представляет собой ту или иную структуризацию, отражающую ее членение на физические (при анализе) или функциональные (при синтезе) компоненты. Характерным для описания систем на уровне протоколов является выделение таких компонентов, как протокольный объект и среда взаимодействия [10].

1.2. Проблема семантической эквивалентности описания поведения

В современном состоянии теорию протоколов составляют формализмы для описания поведения распределенных систем, на которых ставятся задачи преобразования формальных выражений и анализа их свойств. На методологию этой отрасли знания заметное влияние оказывает теоретическое программирование.

Для технологии проектирования сверху вниз характерно представление о процессе проектирования как о системе преобразований на наборе согласованных и оптимизированных моделей. При проектировании протоколов цель каждого преобразования состоит в том, чтобы перевести описание протокола на более высокий уровень детальности фиксации поведения. В ходе преобразований описание претерпевает качественный скачок: описание «что представляет собой поведение системы» преобразуется в описание «каким способом это поведение реализуется».

Качественное различие исходной и результирующей моделей естественно отражается в использовании для их представления отдельных языков. На первом этапе составления проекта протокола поведение системы описывается наиболее простыми и общими средствами. Основное требование к используемому языку состоит в наглядности и обозримости. Затем описание протокола преобразуется в форму, облегчающую анализ на отсутствие в поведении системы опасных ситуаций. Проектирование протокола заканчивается, когда он представлен на входном языке одной из САПР дискретных устройств.

На всех этапах проектирования исходная и результирующая модели должны быть поведенчески эквивалентными. Существующие представления о способах установления семантической эквивалентности описаний параллельных систем укладываются в рамки двух подходов – бисимуляционного и тестового [2]. Две системы считаются бисимуляционно-эквивалентными, если внешний наблюдатель не может обнаружить различие в их поведении. При тестовом подходе поведение исследуется посредством набора тестов. Две системы считаются тестово-эквивалентными, если они обе могут без ошибок выполнить один и тот же набор тестов.

С другой стороны, для полной формализации процесса преобразования представлений протокола требуется, чтобы исходное и построенное описания можно было рассматривать в рамках единой теории. Современный подход к этой проблеме состоит в использовании в качестве средств описания формализмов, обладающих большим разнообразием интерпретаций. В этом случае перевод описания на другой уровень детальности связан со сменой интерпретации используемого формального языка.

Существует противоречие между изобразительными возможностями языка и его способностью эффективно отражать средства базиса реализации. В процессе проектирования желательно иметь ряд методов описания протоколов вместе со средствами преобразования одного формализма в другой. Такой подход особенно полезен в том случае, когда официальные описания протокола используют один формализм, а практические реализации – другой. Официальные описания могут быть трансформированы в практические реализации в ходе автоматического или ручного преобразования. Ясно, что такой подход увеличивает надежность протоколов, упрощает проблемы совместимости и уменьшает стоимость проектирования.

Для обеспечения возможности преобразования выражений одного формализма в другой нужно иметь представление о совместимости структурных компонентов различных формализмов. Для правильного применения преобразований важно знать, как они сохраняют желательные свойства протоколов. Указанные проблемы составляют специфику синтеза реализаций протоколов и обычно называются этапом логического проектирования протоколов. Задачи этапа логического проектирования протоколов являются до сих пор сравнительно мало изученными, для их решения особое значение имеет анализ причинно-следственных и временных зависимостей в протоколах.

1.3. Структурированное описание поведения распределенных систем

Для преодоления проблемы сложности описания поведения вычислительной сети целесообразно иерархически структурировать модели, лежащие в основе описания. Структурирование распределенной системы по уровням иерархии позволяет упростить описание, используя согласованные модели различного уровня абстрактности. Согласование моделей означает, что в моделях заданного уровня используются функциональные возможности, реализация которых раскрывается на модели более низкого уровня. Говорят, что каждый уровень обеспечивает услуги для смежного с ним верхнего уровня. Структурирование по уровням позволяет отразить разбиение системы по задачам, для решения которых эта система предназначена. На каждом уровне решается своя задача, например модель системы на уровне сеанса описывает организа-

цию диалога между компонентами сети, на транспортном уровне рассматривается обнаружение и устранение ошибок передачи данных и т. д. [7].

Иерархическое структурирование по уровням в настоящее время принято повсеместно при проектировании вычислительных сетей. Уровневая организация проектирования удобна с точки зрения практики проектирования, так как она позволяет рассматривать задачи и задавать критерии эффективности реализации отдельно для каждого уровня.

Протоколы самого нижнего, «физического», уровня характеризуются относительно несложными алгоритмами поведения. Видов физических сред немного – обычно это витая пара проводов, или коаксиальный кабель, или оптическое волокно. Имеется много причин, из-за которых различные распределенные системы, использующие определенную физическую среду, должны быть совместимы между собой по протоколам. Это приводит к разработке стандартов, в которых фиксируются рекомендуемые протоколы доступа к физической среде.

Определяющим требованием при разработке протоколов физического уровня является критерий производительности. Он диктует базис реализации, в качестве которого обычно используются логические элементы и схемы из них. При реализации протоколов физического уровня методы теории протоколов применяются редко: проверку на корректность делать не надо, так как протокол зафиксирован в стандарте, и его корректность продемонстрирована рядом реализаций; из-за несложности поведения преобразование его описания до уровня языка регистровых передач или до уровня логических уравнений легко осуществляется вручную и т. д. Из всего инструментария теории протоколов трудно обойтись только без методов формального описания протоколов. Целесообразность применения формальных методов описания объясняется не перспективами автоматизации проектирования, а необходимостью обеспечения единообразного понимания стандартов различными группами проектировщиков. Специфика реализации протоколов среди других типов дискретных устройств здесь почти полностью пропадает.

Чем выше уровень протокола, тем меньше стандартизируется его поведение. С некоторого уровня протоколы реализуются полностью программным путем. В связи с этим становятся все более актуальными задачи проверки корректности протокола, анализа его производительности, автоматического синтеза реализации.

1.4. Проблема корректности протоколов

Понятие «корректность протокола» связано с установлением факта наличия у протокола некоторого набора желательных свойств. Все утверждения о желательном поведении протоколов можно разбить на два класса: свойства безопасности и свойства живости [11]. Свойства безопасности имеют форму «нежелательные вещи при функционировании протокола не происходят». Свойства живости имеют форму «желательные вещи происходят всегда» (например, после приема сообщения всегда посылается подтверждение). Таким образом, свойство безопасности – это утверждение, что если что-то делается, то не возникает опасных ситуаций, а свойство живости – это утверждение о возможности всех предусмотренных действий.

Набор свойств, необходимых и достаточных для того, чтобы протокол считался корректным, в общем специфичен для конкретной распределенной системы. В этот набор обычно входят свойства из числа перечисленных ниже [9]:

ограниченность – при функционировании протокола число сообщений в каждом канале не превышает емкости канала;

однозначность определения состояния – отсутствие протокольного объекта, в котором одно состояние сосуществует одновременно с несколькими состояниями другого объекта;

отсутствие избыточности в описании протокола, например, отсутствие описания обработки непоступивших сообщений;

полнота – в описании протокола предусмотрен прием всех возможных сообщений;

отсутствие статических тупиков – отсутствие ситуаций, когда протокол попадает в состояние, из которого нет выхода;

отсутствие динамических тупиков – протокол не попадает в неэффективные циклы, в которых происходит бесполезный обмен одними и теми же сообщениями;

завершаемость – протокол обеспечивает предоставление требуемых услуг за конечное время (т. е. протокол достигает конечного состояния).

Считается, что протокол функционирует корректно, если поведение агентов отвечает ряду априори сформулированных требований, причем существуют как общие (живость, безопасность), так и частные требования (например, приведенные в предыдущем абзаце). Формальной моделью требований является описание поведения целого класса систем. Таким образом, проверка наличия желательного свойства состоит в установлении эквивалентности поведения заданной системы и системы, поведение которой задается проверяемым требованием. Метод установления эквивалентности зависит от формализма для задания свойств. Этот формализм очевидно также должен относится к классу формализмов для описания поведения распределенных систем.

1.5. Анализ производительности протоколов

Оценка производительности протокола (времени отклика или пропускной способности) требует исследования вероятностно-временных характеристик процесса его функционирования. Чтобы оценивать производительность протокола, нужно иметь представление о времени протекания операций и разработать формальное описание поведения, содержащее эти оценки. После того как спецификация протокола, задающая длительности протекания операций, будет получена, возможны несколько путей ее использования для автоматического анализа производительности. Наиболее важные из этих путей следующие:

- анализ протокола для проверки логической непротиворечивости спецификаций с учетом временных ограничений;
- моделирование с целью оценки производительности;
- прямой расчет производительности по спецификации.

Методы проверки логической корректности оперируют исключительно с последовательностями возникновения событий при функционировании протокола без учета длительности событий.

Если учесть длительность протекания операций, то окажется, что некоторые из последовательностей не могут реализоваться ни при каких обстоятельствах. Если в результате логического анализа протокол забракован как содержащий ошибки, то может оказаться, что опасные события происходят только в этих нереализуемых последовательностях. Таким образом, при логическом анализе без учета времени преувеличивается опасность поведения протокола. В то же время успешное прохождение такой проверки означает, что протокол гарантированно свободен от ошибок. В реальных протоколах возникновение многих ошибочных ситуаций зависит от длительности выполнения операций протокола. Опасные события не происходят только при вполне определенных соотношениях длительностей, и задачей проектирования становится вычисление требуемых значений.

Спецификация с заданными временами протекания операций может быть использована для прямого расчета производительности протокола. Наиболее исследованы методы такого расчета в случае, когда спецификация представляет собой сетевую модель, интерпретируемую как сеть массового обслуживания [12]. Основными элементами сети массового обслуживания (СМО) являются приборы, выполняющие обслуживание заявок на некоторые работы, и пути перемещения заявок от одного прибора к другому. В терминах протокола приборами являются протокольные объекты, а обслуживание заявок состоит в передаче или приеме сообщений. Показатели эффективности протокола выражаются через основные характеристики СМО: время нахождения заявки в СМО и среднюю производительность прибора.

Для вычисления производительности нужно находить все пути в сети и назначать относительную вероятность реализации этих путей. Зная времена протекания операций в протоколе, можно подсчитать среднюю производительность протокола и время отклика. Принципиальный порок этого метода связан с назначением относительной вероятности различных путей (или назначением вероятности переходов между приборами, если вероятности реализации путей вычисляются). На этапе проектирования, не имея статистики временных параметров в реальных вычислительных устройствах, нельзя назначить эти вероятности без значительных ошибок.

Эффективным методом анализа протоколов является моделирование. Использование этого метода возможно, если формальное описание протокола может интерпретироваться как программа для ЭВМ. Метод анализа производительности путем моделирования состоит в разработке модели протокола, которая может быть реализована в виде программы для ЭВМ, и проведении вычислительного эксперимента с целью сбора статистики о производительности модели. Оценку производительности протокола вычисляют, интерпретируя полученную статистику.

2. Причинно-следственные зависимости в описаниях поведения

Спецификация является просто точной формулировкой требований, предъявляемых к системе, и представляет собой результат кодирования реальных понятий формальным языком. В этом смысле спецификации как результат кодирования могут быть оценены по обычным критериям, используемым для оценки изделий программного обеспечения.

Основные критерии качества спецификации состоят в том, что спецификация должна быть ясной, однозначной и понятной. Эти критерии очевидны и выполняются автоматически при использовании подходящего формализма. Для спецификации распределенных мультиагентных систем предложено большое количество формальных моделей. Из-за возрастающей сложности таких систем для их описания с требуемым уровнем понятности приходится использовать различные формальные модели. Спецификации системы в целом в этом случае гетерогенны и содержат части, закодированные с использованием выражений различных формальных языков. Гетерогенные описания проектируемой системы приходится использовать и в процессе проектирования для моделирования и верификации промежуточных результатов.

Параллелизм означает одновременность. Параллельные процессы могут взаимодействовать между собой или выполняться совершенно независимо. В первом случае процессы для гарантии правильности работы системы могут быть синхронизированы всегда (синхронные системы) или только в определенные моменты времени (асинхронные системы). Приведенные объяснения явно используют понятие времени. С другой стороны, в ряде формализмов для описания параллельных систем время не присутствует.

2.1. Конечный автомат как описание причинно-следственных зависимостей

Пусть имеется некоторое дискретное устройство. Внешнее поведение этого устройства проявляется в изменении значений входных X и выходных Y переменных. Эти переменные являются функциями времени. Однако для описания поведения ни конкретные значения этих переменных, ни конкретные моменты времени, в которые они принимают эти значения, не важны. Важно знать, сохраняется ли значение переменной постоянным в течение некоторого периода времени.

Предположим, что возможно находить периоды стабильности значений (состояния) входных и выходных переменных. Кроме того, предположим, что других способов повлиять на поведение устройства, кроме как изменяя значения входных переменных, нет. В этом случае поведение переменных X и Y как функций времени может быть представлено в форме упорядоченности во времени их состояний, т. е. последовательности состояний.

Обозначим множество возможных состояний входных переменных через **X**, а множество состояний выходных переменных – через **Y**. При переходе от физических значений к состояниям переменных неважно, сколько входных и выходных переменных имеет устройство, так как состояния можно выделять по совокупности значений переменных. В этом случае **X** называется входным алфавитом устройства, а **Y** – выходным.

Поведение устройства в целом описывается функцией, переводящей входные последовательности состояний в выходные. Таким образом, введение состояния в качестве первичного понятия дает возможность исключить понятие времени при описании поведения устройств.

Описание поведения естественно представлять в виде причинно-следственной зависимости. При использовании состояний как первичных понятий требуется описать поведение так, чтобы в явном виде были указаны причины возникновения каждого выходного состояния. Очевидно, после подачи на вход любой допустимой последовательности выход устройства оказывается в определенном состоянии, т. е. для описания поведения любого устройства достаточно

указать функцию, переводящую различные входные последовательности в элементы алфавита Y.

Обозначим множества всех входных последовательностей состояний из X через X^*. Тогда описание поведения – это функция $f : X^* \cup Y$. Такое описание поведения было предложено Кроном и Роудзом [13]. Если f трактовать как причинно-следственную зависимость, то f символизирует утверждение, что причиной возникновения определенного выходного состояния является класс последовательностей входных состояний.

Чтобы построить функцию f, можно подать на вход устройства всевозможные конечные последовательности входных состояний. При подаче каждой последовательности фиксируется выходное состояние. Для примера будем считать, что выходной алфавит – булев. В этом случае все множество входных последовательностей можно разбить на два класса, дающих на выходе «0» и «1» соответственно. Для задания поведения достаточно выделить в X^* множество последовательностей, устанавливающих устройство в выходное состояние «1». Нетрудно заметить, что это множество может интерпретироваться как язык в алфавите X и для его задания можно использовать аппарат грамматик. При большем чем два числе выходных состояний нужно для каждого выходного состояния указать язык, содержащий слова, переводящие выход устройства в заданное состояние.

Возможно описание поведения и так, что причинно-следственная связь между входными и выходными состояниями будет указана в явном виде. Последовательности α_1, $\alpha_2 \in X^*$ эквивалентны, если существует $\alpha_3 \in X^*$, такая, что $\alpha_1\alpha_3$ и $\alpha_2\alpha_3$ переводят выход устройства в одно и то же состояние. Найдем языки $Q_i \in Q$, содержащие только эквивалентные последовательности. Здесь Q – множество всех возможных классов эквивалентности над алфавитом X. Тогда поведение устройства можно описать формулами [13]:

$$\phi : Q \times X \cup Y;$$
$$\psi : Q \times X \cup Q.$$

Эти формулы можно интерпретировать как утверждения, что выходное состояние зависит от последнего входного состояния и класса эквивалентности входных последовательностей, поданных до данного состояния. Вторая формула может пониматься как утверждение о том, что отношение эквивалентности замкнуто относительно операции добавления символа к последовательности.

Введенную в этих формулах дополнительную причину $q \in Q$ перехода к новому состоянию выхода называют внутренним состоянием устройства, хотя «узнавание» состояния происходит только на основе внешнего поведения.

Несмотря на то, что по определению понятие внутреннего состояния не имеет физического смысла, оно может быть сопоставлено реальным переменным в устройствах, внутренняя структура которых состоит из комбинационной схемы и элементов памяти (рис. 1).

Рис. 1. Схема устройства со структурной реализацией внутренних состояний

В таких устройствах переменные, определяющие внутренние состояния, управляют работой элементов памяти. Если элементы памяти имеют два состояния, то для описания поведения в виде причинно-следственных зависимостей удобно использовать секвенции.

Секвенциальный автомат [14] – это неупорядоченная система выражений типа $f\tau k$, называемых секвенциями. Здесь f – это булева функция, не тождественно равная нулю, а k – конъюнкция, ранг которой не меньше 1. Секвенция символизирует следующее утверждение. пере-

менные, являющиеся аргументами *f*, находятся в состояниях, обращающих *f* в 1, то переменные, указанные в конъюнкции, находятся в состояниях, обращающих *k* также в 1.

Выделим среди переменных, упоминаемых в секвенциях, описывающих поведение некоторого устройства, те переменные, которые фигурируют и в правых, и в левых частях секвенций. Эти переменные называются внутренними. Если внутренние переменные в секвенциальном автомате отсутствуют, то система секвенций описывает комбинационную схему – автомат без памяти.

Очевидно, в качестве описаний поведения имеет смысл рассматривать лишь такие системы секвенций, в которых отсутствуют противоречивые предписания. Система секвенций противоречива, если согласно одной секвенции требуется установка некоторой переменной в 0, а согласно другой, описывающей совместимые условия, требуется установка ее же в 1. Отсутствие противоречий гарантируется при выполнении условия

$$(k_1 \wedge k_2 = 0) \Rightarrow (f_1 \wedge f_2 = 0)$$

для любых пар секвенций $f_1 \tau k_1$ и $f_2 \tau k_2$ [14]. Здесь => – символ импликации. Таким образом, символ секвенции τ может интерпретироваться как обозначение причинной связи.

Секвенциальный автомат называется инерциальным, если предполагается, что переходы из состояния в состояние выполняются следующим образом. Пусть в текущем состоянии обратились в 1 левые части секвенций так, что в соответствующих правых частях оказался неупомянутым символ некоторой переменной. В этом случае считается, что внутренняя переменная сохранила свое значение в новом состоянии, а выходная приняла значение 0.

2.2. Время как логическое понятие

Отсутствие в приведенных рассуждениях ссылок на время не означает, что оно при описании поведения игнорируется. «Физическое понятие "время" выражает порядок смены событий» [15]. Аспект времени, связанный с порядком смены событий, присутствует в любом описании поведения.

Временные отношения могут быть описаны двумя способами: «каждое событие происходит раньше, чем некоторое другое и позднее, чем некоторое третье»; или «каждое событие расположено или в прошлом, или в настоящем, или в будущем» [16]. При рассмотрении времени как логического понятия отказываются от учета его свойств, связанных с измерениями, т.е. характеризуют поведение устройства системой временных отношений между событиями, происходящими в экспериментах с устройством. Здесь «логическое» имеет тот же оттенок смысла, что и в следующем определении: «формальную систему называют логической теорией некоторой предметной области, если все рассуждения в ее рамках выполняются аксиоматическим методом» [17]. Подход к понятию времени, при котором интересуются только порядком смены событий, характерен для большинства формализмов описания поведения, поэтому имеет смысл рассмотреть его более подробно.

При аксиоматическом подходе время обычно определяют через понятия события и процесса, которые считаются первичными. Процессом считается некоторое отношение между событиями [21]. Логические теории времени [16,18-20] представляют собой аксиоматические теории процессов. Эти теории формулируются или как прикладные исчисления предикатов с предикатными константами, имеющими смысл временных отношений, или как модальные логики, в которых модальности трактуются как операторы времени.

Исследования, в рамках которых первоначально разрабатывались логические теории времени, были связаны с анализом выражений естественных языков, имеющих временной аспект. Логическим анализом предложений типа «завтра будет то-то» занимался еще Аристотель и особенно Диодор Кронос. Толчок к современным исследованиям в этой области дала работа А. Прайора [16], в которой временная логика излагалась на языке современной формальной теории модальностей. В технических науках временная логика первоначально предлагалась для спецификации программных систем реального времени [20].

Логические теории времени представляют собой ветвь модальной логики. Предметом модальной логики является изучение модальностей – прежде всего необходимости и возможности. В логической теории времени этим модальностям придается временной оттенок: необ-

ходимость понимается как «всегда», а возможность – как «иногда». Модальная логика изучает не только утверждения и отрицания, но так называемые сильные и слабые утверждения и отрицания. К сильным относятся утверждения: это необходимо (всегда) истинно, к слабым: это возможно (иногда) истинно. Предложения модальной логики не являются категориями истинными или ложными, а являются истинными и ложными в определенном случае, в некоторых случаях или во всех случаях (всегда).

Усиление логического анализа по сравнению с классической логикой происходит в результате учета возможности изменения истинностных значений логических формул с течением времени. Формализация более сложных логических отношений требует усложнения символики логического языка и принятия дополнительных аксиом.

Важным результатом логической теории времени является установление факта, что существует набор аксиом, без которых не может быть построено ни одно логическое исчисление времени. Этот набор совместно с аксиомами классического пропозиционального исчисления образует «минимальную» временную логику. Аксиомы «минимального» исчисления отражают самые фундаментальные свойства понятия процесса.

Исследования по логической теории времени показали существование неоднозначности очевидных понятий. Возможны два взгляда на время: в одном из них время линейно (в каждый момент существует только одно возможное время), в другом – имеет ветвящийся, деревообразный характер (в каждый момент время может разделиться на альтернативные потоки, представляющие различное возможное будущее). Оба времени, и линейное и ветвящееся, могут быть описаны одной и той же системой аксиом, и, следовательно, временные утверждения описываются одними и теми же формулами. Был предложен логический язык, базирующийся на временных операторах «всегда» и «иногда». Этот язык может интерпретироваться как утверждения и линейной и ветвящейся логики, т. е. понятия «всегда» и «иногда» не однозначны во временном аспекте.

Альтернативным подходом к изучению логической природы времени является алгебраический, развиваемый в виде формальных систем «исчисления взаимодействующих процессов» CSP [4] и «исчисления взаимодействующих систем» CCS [10].

3. Взаимодействие процессов

Для описания поведения устройств требуются предположения об источнике входных воздействий. Последовательность входных состояний определяется средой, в которой работает устройство. Поведение среды может быть описано таким же способом, как и поведение устройства. Поведение замкнутой системы состоит во взаимодействии устройства и его окружения. В такой системе имеются два канала взаимодействия, связывающие входы/выходы устройства с выходами/входами среды. Поведение каждого канала описывается последовательностью соответствующих состояний.

Если в системе имеется больше двух взаимодействующих устройств, то каналы взаимодействия могут работать в значительной степени независимо друг от друга. Определение полного входного состояния устройства по всем его входным каналам становится трудной задачей из-за того, что установившиеся значения переменных в одном канале могут наступать в далеко отстоящие моменты времени от установки значений переменных в другом канале.

Одним из путей преодоления этих трудностей является переход к другой системе понятий. Основой данного подхода служит предложение описывать поведение взаимодействующих устройств в терминах событий, происходящих в каналах (точках) взаимодействия. Считается, что конкретное событие происходит мгновенно, т. е. является элементарным актом, не имеющим протяженности по времени, к тому же событие не имеет точной координаты на временной оси. Таким образом происходит абстрагирование от физического времени и переход в область логики, когда интересуются только упорядочением событий во времени. Так как события происходят мгновенно, то одновременность двух событий невозможна.

Если одновременность двух моментов существенна для поведения, то они сводятся в одно событие. Протяженные во времени действия следует рассматривать как пару событий, первое из которых отмечает начало действия, а второе – его завершение. Из-за того, что не делает-

ся различий между событиями, инициированными устройством или его окружением, причинно-следственные отношения между событиями не представимы, если они не проявляются во временной упорядоченности событий. Этот подход развивается в формализмах CCS [10] и CSP [4], предназначенных для описания поведения распределенных систем.

Процессы конструируются из других процессов с помощью специальных операций. Константами в этой алгебре процессов служат последовательные процессы. Последовательный процесс – это некоторая последовательность событий. В CCS операциями алгебры процессов являются суммирование, префиксация, параллельная композиция, упрятывание и переименование портов.

Суммирование объединяет два процесса P и Q в процесс $P+Q$, который ведет себя как P или как Q. Если P и Q – последовательные процессы (их история функционирования описывается единственным образом в виде некоторой последовательности событий), то процесс $P+Q$ может иметь в качестве истории функционирования как первую последовательность, так и вторую. Операция суммирования в качестве нуля на множестве процессов имеет процесс *STOP*.

Операция префиксации, действующая на событие e и процесс P, дает в результате процесс $e.P$, в котором вначале происходит событие e, а затем процесс ведет себя подобно P. Таким образом, любая история функционирования $e.P$ начинается с e. Процесс $e.STOP$ имеет единственную историю функционирования, и эта история содержит одно событие e.

Событие e может быть или передачей $v!$, или приемом $v?$, или внутренним событием τ.

Параллельная композиция объединяет два процесса P и Q в процесс $P\psi Q$, который ведет себя подобно двум компонентам, обменивающимся информацией через каналы взаимодействия, называемые портами v этих компонентов. Взаимодействие рассматривается как событие, происходящее одновременно в обоих процессах (рис. 2). Если P и Q – последовательные процессы, то история функционирования $P\psi Q$ получается из первой и второй последовательности произвольным чередованием событий. При построении таких историй, если оказываются соседними события $e!$ и $e?$, то они заменяются символом τ, что трактуется как взаимодействие процессов P и Q с участием порта e.

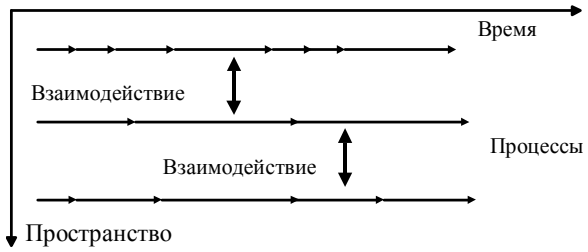

Рис. 2. Взаимодействие параллельных процессов

Цель разработки формализмов CCS и CSP состояла в предоставлении основы для сравнения моделей поведения распределенных систем разной степени абстрактности. Значение этих разработок в изучении параллелизма и асинхронности можно сравнить с ролью исследований по основаниям математики. Как практический инструмент анализа или синтеза распределенных систем CCS и CSP имеют невысокую ценность, так как игнорирование средств задания причинно-следственных зависимостей приводит к большим трудностям в описаниях поведения реальных устройств.

Заключение

Чтобы выполнить общую работу в мультиагентной системе, нужно управлять индивидуальными агентами путем задания протокола взаимодействия, описывающего параллельное и асинхронное поведение агентов. При разработке системы, работа которой удовлетворяет поставленным целям по назначению, стоимости и надежности, процесс проектирования должен быть основан на ясном представлении о сущности проектных операций, трансформирующих формальные описания. Для представления дискретных систем предложено много формальных

моделей. Часто эти модели не включают явное понятие времени. Показано, что несмотря на отсутствие явных ссылок, во всех формальных моделях время принимается во внимание в форме причинно-следственных зависимостей. В данной концепции течение времени моделируется упорядочением событий.

Выполненный в этой части работы анализ базовых понятий формальных моделей: параллелизма, асинхронности и времени – создает базу для разработки описаний поведения параллельных систем. Во второй части работы рассматривается иерархия моделей для описания поведения параллельных систем с асинхронным взаимодействием.

Список литературы

1. Зандере Л.Я., Мартинсонс К.Я., Фрицнович Г.Ф. Автоматизация разработки протоколов сетей ЭВМ и их реализаций (библиография с комментариями) // Вычислительные сети: Логическое проектирование протоколов. – Рига: Зинатне, 1988. – Вып. 2. – С. 200-273.

2. Manna Z., Pnueli A. The temporal logic of reactive systems. – Springer-Verlag, 1991.

3. Ritchie D. M., Thompson K. The UNIX time sharing system // Comm. ACM. – V.1. – №7, 1974. – P. 365-375.

4. Хоар Ч. Взаимодействующие последовательные процессы. – М.: Мир, 1989. – 264 с.

5. Серф В. Передача пакетов // Протоколы и методы управления в сетях передачи данных. – М.: Радио и связь, 1985. – С. 80.

6. Черемисинов Д.И. Формальные методы описания поведения распределенных систем. – Минск, 1991. – 44 с. (Препринт/ Ин-т техн. кибернетики АН БССР; № 38).

7. Дей Дж. Д., Зиммерман Ю. Эталонная модель взаимосвязи открытых систем (ВОС) // ТИИЭР. – 1983. – Т.71. – № 12. – С. 8-17.

8. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. – СПб.: Невский диалект, 1998.

9. Пранявичус Г.И., Хмеляускас А.В. Спецификация и верификация протоколов // Зарубежная радиоэлектроника. – №6. – 1986. – С. 33-49.

10. Milner R. A calculus of concurrent systems. LCNS v. 92 – New-Jork: Springer-Verlag, 1980.

11. Genrich H.J., Lautenbach K. System Modelling with High-level Petri Nets // Theoretical Computer Science. – V. 13. – 1981. – P. 109-136.

12. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. – М.: Техносфера, 2003. – 512 с.

13. Krohn K.B., Rhodes J.L. Algebraic theory of mashines // Trans. Am. Math. Soc. – V.116. – 1965. – P. 450-464.

14. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981.

15. Физический энциклопедический словарь, 1983. – С. 592.

16. Prior A. Past, Present and Future. – Oxford: Claredon, 1967. – 217 p.

17. Ершов Ю.Л., Палютин Е.А. Математическая логика. – М.: Наука, 1979. – 320 с.

18. Караваев Э.Ф. Основания временной логики. – Л.: ЛГУ, 1983. – 176 с.

19. Ишмуратов А.Т. Логические теории временных контекстов (временная логика). – Киев: Наукова думка, 1981. – 150 с.

20. Pnuely A. The temporal semantics of concurrent programs. LCNS v. 70. – Berlin: Springer-Verlag, 1979. – P. 1-20.

21. Авакян В.В., Юдицкий С.А. Описание и анализ параллельных алгоритмов логического управления с применением графов операций // АиТ. – №1. – 1990. – С. 100-108.

Поступила 21.06.04

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail:cher@newman.bas-net.by

D.I. Cheremisinov

FORMAL METHODS OF REACTIVE SYSTEMS BEHAVIOUR ANALYSIS

Concurrency is a core issue in computer science, and it is vital to have theoretical foundations to understand it. The control of multi-agent systems focuses on the control of individual agents to accomplish a mission collectively, while satisfying their dynamic equations and inter-agent formation constraints, for an underlying communication protocol being deployed. To develop the systems that meet the performance, cost and reliability goals, the design process should be founded upon a clear representation that allows to accomplish the design cycle, based on formal notation. Many formal models have been proposed to represent the digital systems. Generally those models do not include an explicit notion of time. It is shown that, despite obvious references lack, in all formal models the time is taken into account as the concept of causal time. With this concept the progression of time is modeled in the system by the ordering of events.