I. ОБЩАЯ ХАРАКТЕРИСТИКА МЕТОДОВ ФОРМАЛИЗОВАННОГО СИНТЕЗА И АНАЛИЗА ПРОТОКОЛОВ ИНФОРМАЦИОННОГО ОБМЕНА

I.I. Понятие протокола

Интенсивное развитие методов проектирования процедур обмена нашло свое отражение во многих работах, посвященных формализации описания, анализа и реализации протоколов взаимодействия структурных объектов вычислительных систем.

Однако, прежде чем говорить об описании протоколов, следует обратить внимание на то, что широкое использование термина "протокол" позволяет с единых позиций рассматривать аспект взаимодействия (общения) порой разнородных по своей внутренней структуре объектов. С другой стороны, широта этого понятия приводит к весьма разнообразной трактовке этого понятия в самых различных областях вычислительной техники. Например, термин "протокол" используется в сфере разработки вычислительных сетей, сетей телефонной и телеграфной связи [I7,44,59] и в то же время при проектировании программных операционных систем [75,III], внутримашинных интерфейсов [II,95]. Подавляющее большинство моделей отличается друг от друга не только тем, какой формализм в них принят за основу, но и тем, что подразумевают авторы моделей под протоколом обмена. Учитывая возможность неформального определения этого понятия, можно представить те трудности, с которыми сталкиваются разработчики конкретных структур взаимодействия.

Для выявления понятия протокола используются обобщенные концепции, в контексте которых определяется данное понятие. Примеры таких концепций: теория обменов [96], асинхронные процессы [3]. Теория обменов определяет основные термины, связанные со взаимодействием "собеседников", представляет каждого из них в виде структуры из обобщенных логических процедурных блоков. Основное

ее достоинство - структуризация взаимодействия объектов. Вторая концепция исходит из необходимости обобщения описания асинхронных параллельных систем и процессов, а также условий их согласованного взаимодействия. Основное достоинство - идея асинхронности и параллелизма в поведении объектов.

Указанные концепции отражают два возможных подхода к определению протокола. Первый подход отличает следующее:

- содержательное определение участников взаимодействия;
- определение протокола обмена - формально-лингвистическое;
- структурирование участников обмена (выделение процедурных блоков, осуществляющих собственно протокол данного уровня, и блоков, реализующих межуровневое взаимодействие внутри объектов);
- выбор какого-либо формального изобразительного средства задания функционирования указанных блоков.

Второй подход отличает следующее:

- разработка обобщенного формального языка описания объектов и систем (асинхронного процесса), обладающих параллелизмом и органической асинхронностью в поведении;
- покомпонентная структуризация ситуаций процессов;
- выявление способов композиции структурированных процессов;
- определение протокола путем задания системы взаимодействующих объектов на языке композиции асинхронных процессов;
- возможность перехода к другим частным формальным изобразительным средствам путем интерпретации асинхронных процессов.

Как правило, при описании структуры вычислительных сетей в определении протокола старается отразить его функциональное назначение в многоуровневой, физически распределенной системе. Под протоколом понимается совокупность правил взаимодействия объектов при выполнении ими определенных сервисных функций, заданных для некоторого уровня структуры распределенной системы [51,59].

Сервисными функциями могут служить, например, для транспортного уровня – примитивы СОЕДИНИТЬ, РАЗЪЕДИНИТЬ, ПОСЛАТЬ, ПРИНЯТЬ. В условиях описания межмодульного взаимодействия внутри локальной системы [66], например, ЭВМ с магистральной структурой, такими примитивами могут быть команды ввода-вывода – ЧИТАТЬ ДАННЫЕ ИЗ ОЗУ, ВЫВЕСТИ ПАКЕТ ДАННЫХ НА ВНЕШНЕЕ УСТРОЙСТВО.

Поскольку область использования процедур обмена не ограничивается физически распределенными системами, то понятие протокола вправе применить для более широкого класса взаимодействий, в частности для информационного обмена по межмодульной интерфейсной магистрали. Таким образом, протокол – это совокупность правил, которым следуют рассматриваемые на некотором уровне абстракции объекты, взаимодействующие путем обмена друг с другом сигналами (сообщениями, пакетами, кадрами или любыми другими единицами информации) [99].

Что же должно включать в себя описание протокола?

В [67] показано, что протокол не может быть вполне задан без описания всего архитектурного уровня, на котором он реализуется. Описание же уровня должно включать следующие элементы:

1. Общее описание цели уровня и сервиса, который он обеспечивает для верхнего уровня.

2. Точная спецификация функций сервиса.

3. Точная спецификация сервиса, обеспечиваемого нижним по отношению к данному уровнем и требуемого для правильного и эффективного функционирования протокола. (Это в некотором смысле избыточно, хотя делает описание протокола замкнутым).

4. Внутренняя структура уровня в терминах объектов и отношений между ними.

5. Описание протокола обмена между объектами, включающее:

а) общее неформальное описание работы объектов;

б) спецификация протокола, включающая:

- список типов и форматов сообщений, которыми обмениваются объекты,

- правила реакции каждого объекта на команды пользователя (верхнего уровня), сообщения других объектов и внутренние события;

в) некоторые дополнительные детали (не указанные в п.б.), такие как характеристики по повышению эффективности, предложения по выбору средств реализации или даже детальное описание, которое может быть максимально приближено к реализации.

I.2. Основные способы описания протоколов

Описания сервисных функций и протоколов должны быть как легко доступными для понимания, так и точными. Эти цели часто конфликтны. Использование естественного языка дает иллюзию быстрого понимания, но ведет к длинным и неформальным спецификациям, которым свойственны, во-первых, неоднозначность трактовки и трудности при проверке полноты и корректности с точки зрения семантики и синтаксиса протокола, а, во-вторых, неизбежность формализации при переходе к аппаратным или программным средствам реализации.

В большинстве работ по формализации взаимодействий затронута проблема описания самих протоколов, а не сервиса, ими представляемого. Изобразительные средства, используемые при этом, можно разбить на три основные группы [67]:

I. "Модели переходов", отражающие тот факт, что протоколы состоят из относительно простых по вычислению преобразований информации, однако, осуществляемых при большом потоке событий и условий, таких как команды верхнего уровня, сигнализация с нижним уровнем при приеме и передаче сообщений, внутренние тайм-

ауты, встроенные задержки. К таким моделям относятся диаграммы состояний [8,64,83,92,II9], логические матрицы и матричные схемы алгоритмов [33,60,76], граф-схемы алгоритмов [33,36,77], автоматы с "переменной структурой" [6I,I0I], формальные грамматики [86,II8], сети Петри и их расширения [I00,I08,II7], графы UCLA[II2].

2. Алгоритмы на языках программирования [27,7I,8I,85], мотивируемые тем фактом, что протоколы представляют один тип алгоритма, а языки высокого уровня обеспечивают ясные и относительно сжатые средства описания алгоритмов.

3. Гибридные модели, составляющие комбинацию элементов первых двух групп. Как правило, в них строится базовая модель из формализмов первой группы – для описания управляющих потоков событий, а затем она расширяется дополнительными "контекстными" переменными и обрабатывающими процедурами для всех или некоторых состояний базовой модели. Примеры гибридных описаний – обобщенная модель переходов Бохманна [69], комбинированные сети Петри [77], Е-сети [I04].

Основными достоинствами первой группы моделей является наглядность описания, возможность динамической смены уровня абстракции, высокая степень формализации анализа описаний. Однако применение формализмов этой группы часто ограничено проблемой размерности ("взрыв пространства состояний" [67]). Пример явного усложнения описания – представление процедуры обработки порядковых номеров пакетов для фазы обмена данными в протоколе X.25 (Уровень 3) [II4]. Чтобы задать эту процедуру в терминах, например, сетей Петри, приходится для каждого порядкового номера пакета вводить свой набор условий и событий.

Описание протоколов при помощи языков программирования разрешает в известной степени проблему размерности описания. Однако,

при этом затрудняется верификация протокола ввиду сложности построения и доказательства определяющих утверждений для программ протокола, с помощью которых проверяется его корректность. Кроме того при задании протокола на языке программирования становится невозможным отделить существенные черты функций, возложенных на программу, от конкретного пути, принятого для реализации этих функций, что значительно снижает уровень абстракции описания. Это в большей степени относится к описаниям протоколов, заведомо реализуемых программно, поскольку описание, например, с помощью "неисполняемого" языка высокого уровня может быть удобным [27,81] и при аппаратной реализации.

Как отмечается в [67] перспективными для сетевых протоколов являются "гибридные" модели, в которых управляющие функции протокола (синхронизация, инициация, установление связи, разъединение и т.д.) представлены в виде "модели переходов", а функции обработки данных (средства задания номеров пакетов, переменных счетчиков и т.п.) изложены на языке высокого уровня.

Для исследуемого класса межмодульных взаимодействий аппаратурного уровня наиболее удобными являются модели первой группы, так как они позволяют с любой степенью абстракции, вплоть до функциональных схем, описывать и анализировать полученные структуры на достижимость заданных ситуаций, независимость от скорости и т.п. Для этого существуют удобные средства анализа асинхронных структур [31,55].

Как правило, методология проектирования модели состоит в следующем. Разработчик сперва строит основной "сценарий" взаимодействия, который включает определение перечня основных сигналов или сообщений, а также базовой последовательности этих сигналов. Затем он исследует альтернативные ситуации и может ввести новые сигналы и их последовательности. Разумеется, он учитывает в такой

сценарии возможность возникновения ошибок и восстановления "ритма" взаимодействия. В процессе такого синтеза он использует "подручные" изобразительные средства и инструменты анализа своего сценария.

В процессе создания модели взаимодействия могут быть применены различные структурные подходы, которые отражают как уровень абстракции описания, так и условия самого взаимодействия (например, наличие или отсутствие помехозащищенного канала связи между объектами):

I. Модель непосредственного взаимодействия пары объектов (рис.I.I,а). В ней подразумевается отсутствие какого-либо описания согласующего объекта (процесса) - адаптера, канала или другого средства передачи информации на нижнем уровне. Кроме того не выделяется верхний уровень, т.е. "запросы" и "ответы" сервиса такого взаимодействия, - все это относится к внутренним событиям объектов O_1 и O_2. Примером такой модели является взаимодействие программ, ориентированных на обмен сообщениями [75], с блокировкой обоих видов примитивов "send" и "receive". Другим примером может служить модель взаимодействия апериодического автомата и среды [2].

2. Модель опосредованного взаимодействия пары объектов (рис.I.I,б). Она предполагает наличие некоторого согласующего звена - средства передачи на нижнем уровне. Большинство асинхронных интерфейсов может быть описано с помощью такой структуры. Учет реального канала связи с "перекосом задержек" осуществляется, например, в "общей шине" компенсирующей задержкой - 75 нс. Другой пример - взаимодействие программ - обмен сообщениями с отсутствием блокировки какого-либо из указанных примитивов ввиду наличия буфера.

3. Модель управляемого взаимодействия пары объектов (рис.I,I,в). Предполагается структурирование каждого из взаимодействующих

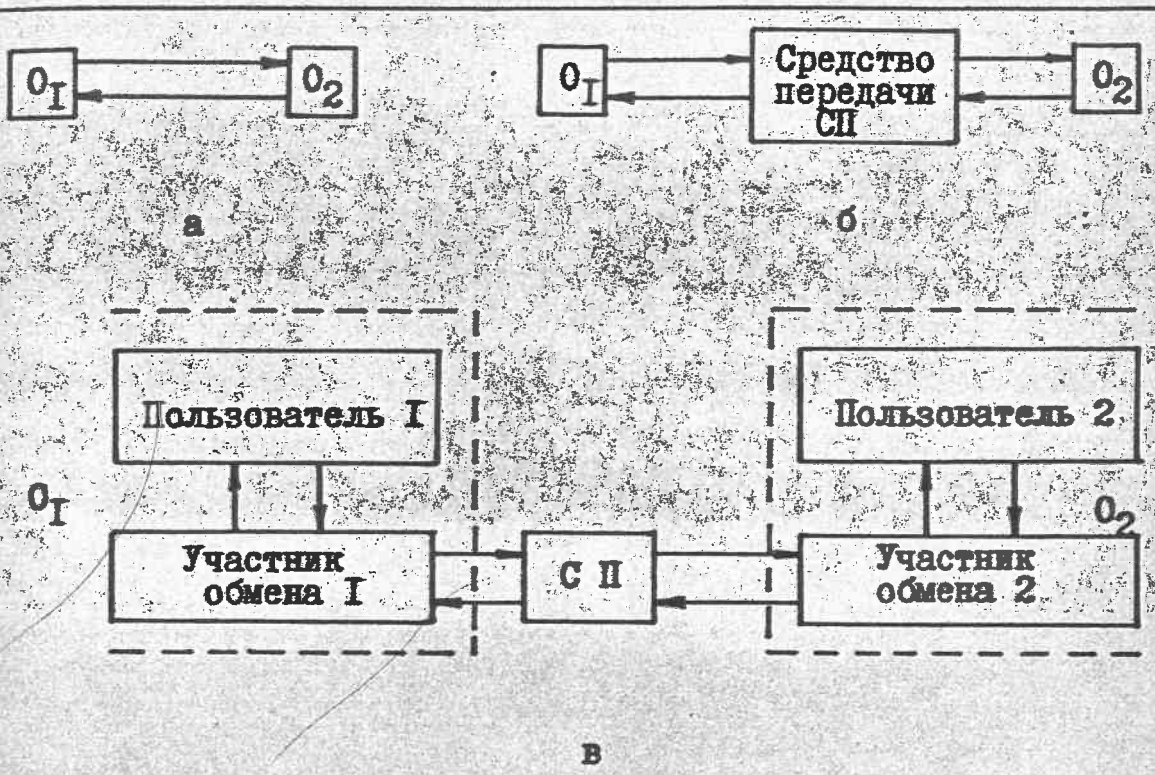
O_1
O_2
а
O_1
Средство передачи СП
O_2
б
Пользователь I
Пользователь 2
O_1
O_2
Участник обмена I
С П
Участник обмена 2
в

Рис. I.I.

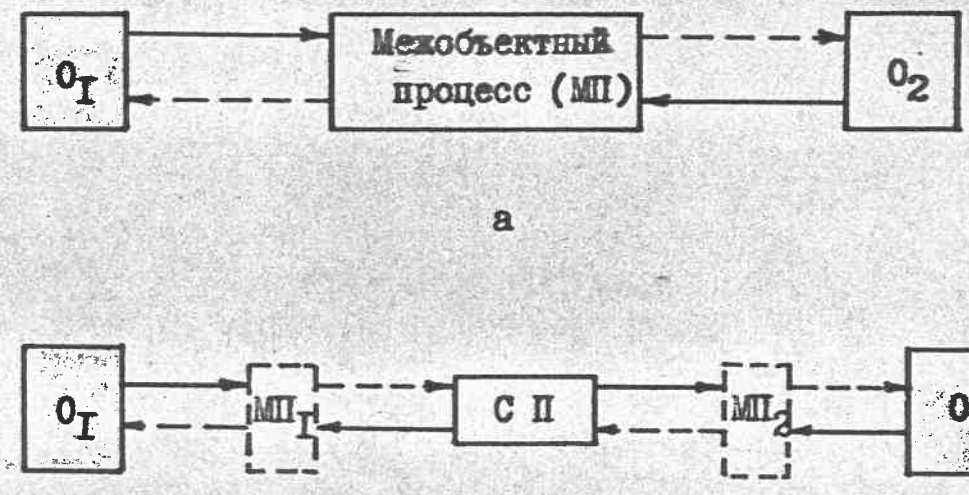
O_1
Межобъектный процесс (МП)
O_2
а
O_1
$МП_1$
С П
$МП_2$
O_2
б

Рис. I.2.

объектов – выделение логического участника взаимодействия и управляющего им звена. Такая модель обычно используется для описания протоколов типа "Конец - Конец" [59,78], протоколов, описывающих приборные и интерфейсные функции для интерфейса МЭК [88].

4. Модели взаимодействия в условиях сложной топологии. Легко заметить, что методы описания протоколов не должны ограничиваться моделями парных взаимодействий. Действительно, структуры взаимодействия подчас должны охватывать несколько процессов [72]. Это касается не только сетевых протоколов маршрутизации [9,99]. При организации межмодульного взаимодействия возникает проблема описания функции захвата общего ресурса – шины обмена – путем арбитража запросов [106] или проблема организации обмена одного источника информации и многих приемников [27,120]. Оказывается не всегда такая модель может быть сведена к суперпозиции моделей парного взаимодействия [49], в частности проверка работоспособности сложной по топологии системы объектов не всегда обеспечивается анализом всех "парных" протоколов.

Рассмотрим два простейших примера описания протокола. Первый пример иллюстрирует использование диаграмм состояний в модели непосредственного взаимодействия. Удобным средством для представления такой модели является гипотетический межобъектный процесс ("интерфейсная машина" [77]), для которого все сигналы, поступающие от объектов – участников обмена, являются входными (рис.I,2,а). Состояния такого процесса однозначно отображают состояние глобальной системы взаимодействия пары объектов через идеальное средство связи. Основное достоинство использования модели межобъектного процесса – наличие минимальной информации в описании взаимодействий: не учитываются внутренние события, присущие объектам. Рассмотрим межобъектный процесс для протокола X.25/3, заданный диаграммой состояний. Диаграмма состояний представляет конечный

орграф, вершины которого отображают состояния межобъектного процесса, а ребра - имена пакетов, посылаемых DTE (терминальной аппаратурой) и DCE (аппаратурой сети передачи данных). На рис.I.3,а,б изображен фрагмент описания X.25/3 для фаз установки и разъединения. В табл.I.I. перечислены используемые типы пакетов.

Таблица I.I.

Пакеты от DTE	Пакеты от DCE
a_1 - запрос вызова	$в_1$ - прошедший вызов
a_2 - вызов принят	$в_2$ - вызов установлен
a_3 - запрос разъединения	$в_3$ - индикация разъединения
a_4 - подтверждение разъединения DTE	$в_4$ - подтверждение разъединения DCE

Состояния протокола обозначены следующим образом:

p_1 - готовность, p_2 - DTE ожидает, p_3 - DCE ожидает, p_4 - обмен данными, p_5 - столкновение вызовов, p_6 - DTE запрашивает разъединение, p_7 - DCE запрашивает разъединение,

$p_i = p_1 \vee p_2 \vee p_3 \vee p_4 \vee p_5$

Отметим следующее:

I. В состояниях p_6 и p_7 переходы по $в_2$ и a_2 соответственно возможны лишь в том случае, если предыдущее состояние $p_i = p_2 \vee p_3$.

(2. В состояниях p_6 и p_7 переходы по a_3 и $в_3$ соответственно осуществляются после таймаута.

3. Если в некотором состоянии на вход межобъектного процесса поступает пакет, не заданный в диаграмме состояний, то этот пакет является ошибочным. (Соответствующий объект - DTE или DCE - при приеме ошибочного пакета отвечают пакетами a_3 или $в_3$).

В модели непосредственного взаимодействия состояние межобъектного процесса соответствует состоянию каждого из участников. При описании модели непосредственного взаимодействия информация,

задающая поведение межобъектного процесса, отражает и глобальное состояние системы в целом. Однако, если применить такой подход в модели опосредованного взаимодействия, то задать глобальное поведение с помощью одного межобъектного процесса нельзя. Можно применить модель с двумя такими процессами (рис.I.2,б), каждый из которых отражает ту информацию, которая имеется у соответствующего объекта о глобальном состоянии. При этом возможно большое различие между состоянием каждого процесса и реальным глобальным состоянием, "благодаря" влиянию неидеального канала связи. Таким образом, модель опосредованного взаимодействия удобно задавать в виде пары описаний участников взаимодействия и модели канала согласования. Описание одного из участников будем называть локальной моделью.

Вторым примером описания протокола является локальная модель участника взаимодействия на основе простейшего "гибридного" описания. Рассмотрим процедуру обработки сообщения с параметрами, например, пакета данных с порядковым номером. Ко множеству базовых состояний протокола X.25/3: $p_j (j=1,\dots,7)$, $d_k (k=1,2,3)$, $r_l (l=1,2)$ добавляется множество "контекстных" переменных и "контекстные" процедуры [7

Пусть $I \subset I_1 \times I_2$ – множество входных пакетов, I_1 – множество типов пакетов, I_2 – множество значений вектора параметров.

$X \subset X_1 \times X_2$ – полное внутреннее состояние локальной модели, X_1 – конечное множество базовых состояний локальной модели (с относительно невысоким кардинальным числом), X_2 – множество значений вектора "контекстных" переменных, причем: U – множество "недвоичных" переменных, X_U – множество значений вектора из переменных $u \in U$, V – множество двоичных переменных, значение которых соответствует предикатам над $u \in U$, X_V – множество значений вектора из переменных $v \in V$. $O \subset O_1 \times O_2$ – множество выходных пакетов, O_1 – множество типов входных пакетов, O_2 – множество

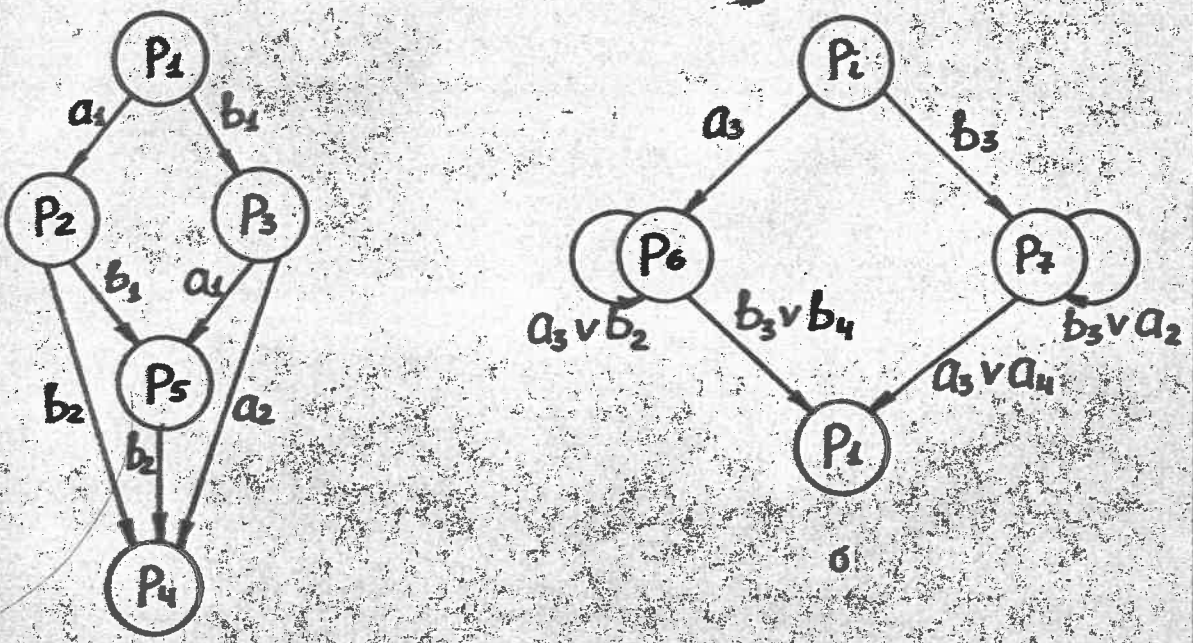
P1
a1
b1
P2
P3
b1
a1
P5
b2
a2
b2
P4
а
Pi
a3
b3
P6
P7
a3 v b2
b3 v b4
b3 v a2
a3 v a4
Pl
б

Рис. I.3.

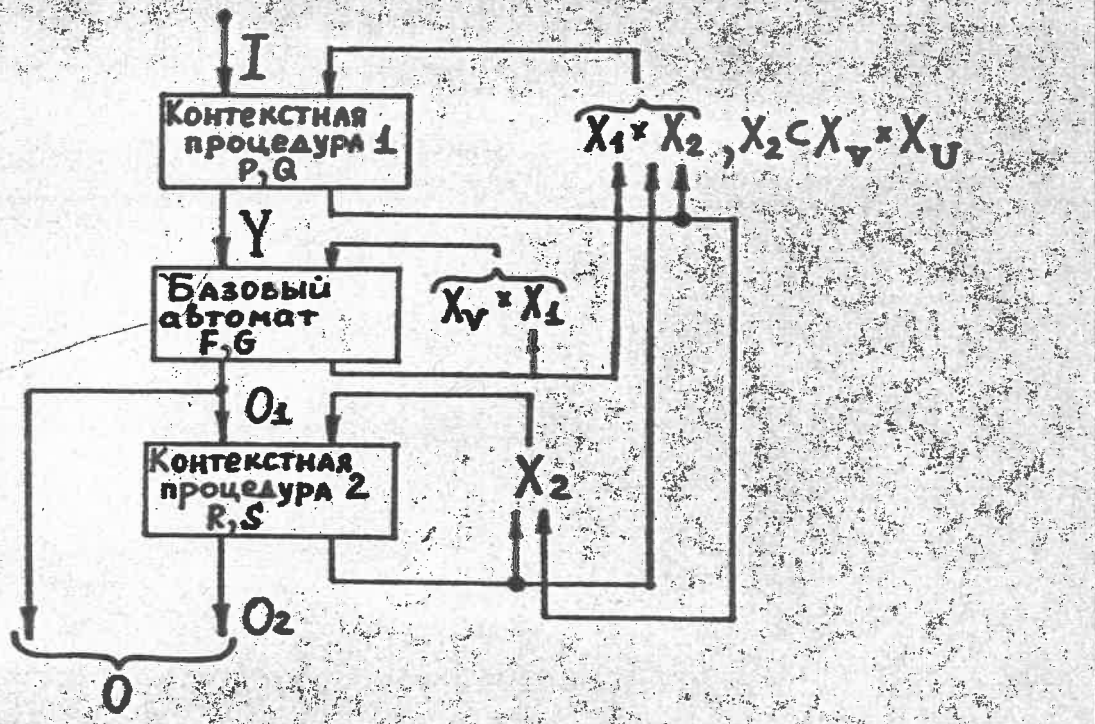
I
Контекстная процедура 1
P, Q
X1 * X2 , X2 ⊂ XV * XU
Y
Базовый автомат
F, G
XV * X1
O1
Контекстная процедура 2
R, S
X2
O2
O

Рис. I.4.

значений вектора параметров.

Шаги работы локальной модели (в скобках указана основная функция, выполняемая на данном шаге) (рис.I.4).

I шаг. Работает контекстная процедура I (анализ параметров входного пакета). Контекстная процедура описывается следующим конечным автоматом: $\langle X_1 \times X_2, I, Y, P, Q \rangle$;

$P: I \times X_1 \times X_2 \to X_2$; $Q: I \times X_1 \times X_2 \to Y$,

где Y - множество выходных символов процедуры, т.е. множество типов пакетов, поступающих на вход базового автомата.

2 шаг. Работает базовый автомат (определение типа выходного пакета и нового базового состояния): $\langle X_V \times X_1, Y, O_1, F, G \rangle$;

$F: Y \times X_V \times X_1 \to X_1$; $G: Y \times X_V \times X_1 \to O_1$.

3 шаг. Работает контекстная процедура 2 (вычисление параметров выходного пакета): $\langle X_2, O_1, O_2, R, S \rangle$;

$R: O_1 \times X_2 \to X_2$; $S: O_1 \times X_2 \to O_2$.

Учитывая, что кардинальное число X_U значительно выше, чем для X_V, базовый автомат может быть представлен "моделью переходов", а контекстные процедуры - программой на языке высокого уровня. Пример построения контекстной процедуры I для DCE в фазе обмена данными приведен в приложении I.

I.3. Общая характеристика методов анализа протоколов

Сущность анализа (называемого иногда верификацией) протокола состоит в выяснении: удовлетворяет ли поведение системы взаимодействующих объектов своему описанию. Можно выделить две основные задачи анализа. Во-первых, проект протокола должен быть проверен путем исследования всех возможных вариантов взаимодействия объектов и необходимых ситуаций с учетом семантически выбранных свойств, во-вторых, реализация каждого логического участника взаимодействия должна соответствовать заданному абстрактному описанию

протокола. Работы по верификации протоколов в основном касаются решения первой проблемы. Решение же второй задачи производится в условиях выбранного способа реализации, и таким образом отделено от первой задачи. Специфика апериодической реализации протоколов предполагает наличие единых формальных средств для решения обеих проблем.

Необходимо учесть, что при анализе протокола некоторого уровня несомненно приходится учитывать влияние нижних уровней, поскольку нижний уровень обеспечивает правильную "доставку почты" для данного уровня взаимодействия. Из этого следует, что при нарушении в описании необходимых свойств протокола, его перестройки может выполняться как на уровне самого протокола, так и на нижнем уровне.

Бохманн и Саншайн выделяют [67] две категории свойств протоколов, которые подвергаются анализу. По одной оси различают общие свойства и частные свойства. По другой - частичную корректность и завершенность, либо прогрессирование.

Общие свойства - это те свойства, которые присущи требованиям ко всем протоколам. Наиболее примечательными являются: отсутствие дедлоков (эффективность или живость), те. состояний или классов, не имеющих "выхода", полноты, т.е. обеспечение описания объекта при всех возможных входных воздействиях. Прогрессирование или завершенность можно рассматривать в этой категории с общих позиций, т.е. минимум указания того, что составляет "полезную работу" или каково желательное терминальное состояние.

Специфические (частные) свойства требуют указания обеспечения определенного сервиса протоколом. Например, для протокола координации программных процессов необходима ограниченность числа сообщений в межпроцессных буферах. Для протокола с самосинхронизацией, необходимо, чтобы дисциплина согласования функциональных модулей не зависела от скоростей работы их элементов и канала связи.

Под частичной корректностью обычно понимается, что если протокол вообще выполняет какое-либо действие, то он удовлетворяет заданию своего сервиса. Например, если апериодический протокол обеспечивает доставку приемнику символа в самосинхронизирующемся коде, то этот символ будет доставлен в требуемом коде без однократных ошибок в каком-либо из разрядов кода.

Завершенность или прогрессирование означают, что заданный сервис будет действительно осуществлен протоколом за конечное время. Протокол апериодического взаимодействия обеспечивает неограниченную априорно, но конечную задержку при передаче данных от источника к приемнику.

Мерлин [99] приводит конкретный список основных свойств желательного поведения структуры взаимодействия:

I. Отсутствие дедлоков: "Никаких безвыходных положений".

2. Живость или связность: "Из каждого достижимого состояния достижимо любое другое" или "Для каждого достижимого состояния и любого события существует достижимое состояние, при котором это событие произойдет, т.е. нет "мертвых" событий".

3. Отсутствие блокировки темпа: "Не существует неэффективных бесконечных циклов".

4. Отсутствие "голодной смерти": "Если несколько процессов пытаются захватить ресурс, который может становиться доступным бесконечно много раз, то нет такого процесса, который бы никогда не смог получить этот ресурс ". (Иногда интерпретируется свойством равномерной диспетчеризации).

5. Восстановление от ошибок: "После ошибки (сбоя) протокол возвращается к нормальному циклу за конечное число шагов".

6. Самосинхронизация: "Из любого ненормального состояния протокол "возвращается" в нормальное состояние за конечное число шагов" - тесно связано с п.5.

7. Правильное исполнение цели протокола: весьма специфичное свойство. Например, "правильно доставлять данные" - для протокола передачи данных или "одновременно шиной владеет только один ведущий" - для протокола арбитража запросов.

Основные методы, применяемые для анализа протоколов, естественно связаны с тем теоретическим аппаратом формулировки свойств, который имеется в рамках указанных изобразительных средств. Исходя из этого, имеется два основных направления анализа - исследование достижимости и верификация - доказательство правильности - программ.

Анализ достижимости базируется на переборе всех возможных взаимодействий двух (диалоги) или более объектов данного уровня. Полное или глобальное состояние системы определяется как комбинация (например, кортеж) состояний взаимодействующих объектов и нижнего связующего уровня. Задается некоторое исходное состояние и генерируется все пространство достижимости путем выполнения всех допустимых в каждом из состояний действий. Такая процедура повторяется для каждого вновь образованного состояния до тех пор, пока не перестанут встречаться новые состояния. Пример эффективной реализации такой процедуры описан в [119]. Основной метод анализа формализмов первой группы - анализ достижимости. Он хорошо приспособлен для проверки свойств общей корректности, так как эти свойства являются прямым следствием структуры графа достижимости. Глобальные состояния, имеющие только входные инцидентные дуги, интерпретируются либо как дедлоки, либо как желаемые терминальные ситуации.

Генерация глобальных состояний для "моделей переходов" может быть достаточно легко автоматизирована путем выбора подходящей формы представления описания в ЭВМ и подбора некоторого алгоритма перебора [78,112,115,119]. В ряде моделей второй группы анализ

сводится к проверке достижимости путем установления определенных узловых точек в программе протокола, так что этим точкам ставится в соответствие некоторое состояние управления [75,81].

Доказательство правильности программы состоит в формулировке утверждений, отражающих желательные свойства протокола [93]. Было бы идеальным, если бы такие утверждения формулировались исходя из сервиса, однако, ввиду общей неформализованности сервиса в настоящее время, пока таких примеров выявить не удалось [67]. Основная задача - показать, что программы протокола для каждого участника - локальные модели - удовлетворяют утверждениям, включающим обоих участников взаимодействия [69,71]. Основное достоинство данного подхода - возможность оперировать с полным набором свойств протокола, а не только с общими свойствами. Однако, автоматизация такого анализа еще весьма проблематична ввиду сложности формализации процедуры синтеза определяющих утверждений. Связующим мостиком между двумя выделенными подходами является метод анализа путем символического исполнения программ, составляющих описание протокола [71].

Особая форма доказательства - "индукция по топологии" [99] - является полезной для протоколов с большим числом взаимодействующих объектов (например, протоколы маршрутизации). Желаемые свойства сперва доказываются для минимального подмножества объектов, а затем доказывается индукционный переход, что если свойство сохраняется для системы из n объектов, то оно сохранится для системы из $n+1$ объекта.

В случае обнаружения ошибки в задании протокола необходимо выяснить ее причину. Возможно некоторые действия или операторы могут "отодвинуть" причину от той точки, в которой возникла ошибка, например, прием дубликата сообщения в приемнике может быть вызван ошибкой в нумерации источника. Важно, таким образом, еще

в описании учитывать возможность локализации (бестестовой) места проявления ошибки. В некоторых случаях протокол может быть смоделирован некорректно, либо условие корректности, в свою очередь, формулируется некорректно. В других случаях, нежелательное поведение может быть вызвано неучтенными свойствами канала связи (искажение кода) или наличием непредусмотренных арбитражных ситуаций (запуск механизма ретрансмиссии после отработки таймера, полагая что сообщение утеряно, и одновременно прибытие данного сообщения к абоненту в правильном виде, за которым следует посылка подтверждения о приеме).

Оригинальные подходы [II9,8] к построению корректных протоколов основаны на использовании конструктивных правил, которые автоматически позволяют синтезировать правильное описание. В первом [II9] случае формулируются правила синтеза, которые гарантируют, что описание, полученное для множества взаимодействующих объектов, будут полными. В другом случае [8] описание одного из участников определяется исходя из заданного описания другого участника и полного глобального описания системы.

В заключение приведем ряд фактов практического использования формальных методов описания и анализа реальных протоколов и стандартов. В основном это касается сферы создания сетевых протоколов и совсем в малой степени внутрисистемных. В одних случаях формальное описание было сделано уже после создания проекта и служило средством дополнительного анализа корректности или как средство перехода к реализации. В других случаях, формальное описание использовалось как справочный документ на этапе системного проектирования.

Наибольший интерес привлек анализ Рекомендаций CCITT X.21, X.25 и протокола связи HDLC . Фаза установки вызова протокола X.21 была смоделирована при помощи диаграмм состояний и пропа-

лизирована методом анализа достижимости [123]. Проверены общие свойства – полнота задания и дедлоки и выявлен ряд "скрытых" ошибок по отсутствию полноты: участник обмена мог получить сообщение, для которого не была предусмотрена обработка. Ряд некорректностей был также обнаружен в описании X.21 при помощи автоматизированной системы SARA [115].

Протокол X.25 (установка виртуального соединения) был промоделирован диаграммами состояний и проанализирован вручную на достижимость [65,68]. Была показана возможность существования устойчивых неэффективных циклов после того, как протокол попадал в некоторые "несинхронизированные" состояния.

Связной протокол HDLC был описан регулярной грамматикой [86] с использованием индексирования для описания нумерации последовательности кадров. Этот же протокол был описан "гибридно" диаграммами состояний в сочетании с контекстными переменными и операторами языка высокого уровня. Такое описание с сильной декомпозицией логического участия на отдельные модули было использовано для перехода к программной реализации X.25 (Уровень 2), соответствующего по уровню HDLC , в [70].

Практически единственный пример формального описания логического уровня межмодульного интерфейса – это описание с помощью диаграмм состояний [11,95] стандарта МЭК [88].

В Табл.I.2. приводится сравнительная характеристика основных методов формализации протоколов, позволяющая судить о степени охвата каждым из методов тех или иных аспектов моделирования и анализа взаимодействий.

Поскольку аспекты реализации протоколов весьма сложно расклассифицировать, и кроме того, учитывая специфику апериодического подхода, принятого в работе за основу, вряд ли уместно подробно останавливаться на вариантах программной или микропрограммной

Таблица I.2. Сравнительная характеристика основных методов описания и анализа протоколов

Авторы /работы/	Вид формализма	Метод анализа /возможность синтеза/	Исследуемые аспекты	Практика применения	Исследуемые свойства	Проблемы описания и дальнейшая разработка
Дэнтайн, Бремер [76-78]	Конечные автоматы +алгоритмический язык	Выделение совместимых переходов и достижимость ситуаций	Управление	Сетевые стандарты	Дедлоки	Синтез конечного автомата;обработка данных,выявление циклов
Бранд, Джойнер [71]	Алгоритмический язык	Символическое исполнение	Обработка данных Управление	Локальные микропроцессорные системы	Дедлоки Циклы	Построение утверждений;описание сложного канала связи, сервиса
Рудин; Вест,Зафиропуло [119,124, 127]	Диаграммы состояний локальной модели	Достижимость глобальных состояний /правила синтеза локальной модели/	Управление	Сетевые стандарты	Дедлоки, полнота	Утверждения,сложный канал связи,циклы, обработка данных
Вохманн, Мерлин [8,68,69]	Диаграммы состояний + алгоритмический язык	Глобальная достижимость,утверждения/Синтез локальной модели по глобальному поведению/	Управление + обработка данных	Сетевые стандарты	Дедлоки, Циклы, Живость	Синтез диаграмм состояний,доказательство утверждений;автоматизация анализа и синтеза,описание сложного канала связи
Харангозо [86]	Формальные грамматики	-	Управление, Данные,структуры пакетов	Сетевые стандарты	-	Синтез грамматики Управление потоком, анализ протокола
Симонс [117]	Диаграммы состояний Сети Петри	Глобальная достижимость	Управление	Сетевые стандарты	Дедлоки, циклы	Автоматизация, формулировка утверждений

реализации протоколов. Как показано во введении, способы построения апериодических структур, поведение которых не зависит от скорости, в области проектирования интерфейсов практически не имеют аналогов в настоящее время. Исключением разве что можно считать проект интерфейса ТРИМОС-БАС [120], анализ которого приведен в разделе 4. Следует, однако, отметить подход к аппаратурной реализации, изложенный в [81], при котором возможна эффективная декомпозиция логического участника обмена на подструктуры - операционную и управляющую.

Краткие выводы

1. Анализ работ по формальным методам описания протоколов показывает принципиальную возможность использования понятия протокола и структурных моделей для формализации описания взаимодействия функциональных модулей ЭВМ при условии апериодической схемной реализации средства сопряжения.

2. До сих пор нельзя выделить общепринятого способа формального описания протоколов логического уровня внутрисистемных межмодульных интерфейсов. Единственное исключение составляет использование диаграмм состояний для описания стандарта МЭК [11,88,95].

3. Для интерфейсных протоколов отсутствуют примеры анализа описаний логики взаимодействий на наличие как общих свойств - дедлоки, полнота, так и специфических свойств - например, независимость от скорости, полумодулярность, дистрибутивность. Последние свойства, присущие апериодической реализации, обеспечивают попутно наличие таких общих свойств, как самосинхронизация и самодиагностирование.

4. В работах [27,35] изложен весьма нестрогий подход к решению проблемы перехода от диаграмм состояний протокола к аппа-

ратурной реализации отдельных интерфейсных функций. Причем полученные таким образом схемы не являются независимыми от скорости [39], а потому не обладают требуемыми свойствами.

5. Из работ [77,81,96] можно сделать полезный вывод об эффективности декомпозиции структуры логического участника выделения управляющей подструктуры и интерпретации (операционной части) [19,29], реализация и анализ которых могут выполняться достаточно независимо друг от друга.