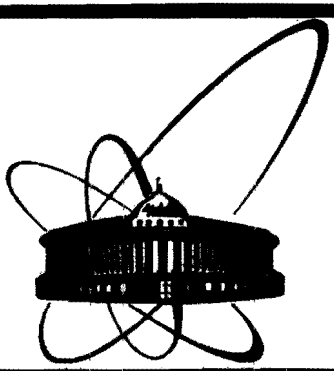

сообщения
объединенного
института
ядерных
исследований
дубна

P11-86-542

Х.Юнгклауссен

МЕТОД МНОГОУРОВНЕВОГО ПОТОКА СОБЫТИЙ (МПС) ДЛЯ ОПИСАНИЯ ПРОЦЕССОВ В СЕТЯХ И ИЕРАРХИЯХ ОПЕРАТОРОВ

1986

I. Постановка задачи

С развитием науки и техники и специализацией их направлений идут параллельные процессы развития и специализации профессиональных средств общения - систем понятий и терминологии. Высокие темпы развития науки и техники приводят к отставанию средств профессионального обучения (предметных языков), что может привести к трудностям во взаимопонимании специалистов, в преподавании соответствующих дисциплин и пропаганде новейших достижений.

Указанные тенденции весьма характерны для нынешнего состояния вычислительной техники и особенно - для ее программного обеспечения, что объясняется как сложностью объектов (программ), так и разнообразием существующих средств их описания (языков программирования).

Известен целый ряд попыток установления согласованной терминологии в области вычислительной техники и программирования - от выпуска терминологических толковых словарей /5/ до создания банка терминов программирования и его приложений в рамках деятельности научно-технической комиссии по технологии программирования Государственного комитета СССР по науке и технике /1,3,4/. Цель последней работы заключается в том, чтобы из широкого фонда терминов, предлагаемых разными авторами, без навязывания частных точек зрения, а путем сравнения выявить самые удачные термины как по содержанию, так и по способности к словообразованию. Это, пожалуй, лучший путь ускорения образования обоснованного предметного языка, претендующего на общее признание и долговечность.

Наряду с таким чисто эмпирическим путем возможен и другой путь, состоящий в попытке надстроить новую систему понятий над некоторой апробированной базовой системой понятий, не вызывающей сомнений и не содержащей неоднозначностей. В качестве базовой системы понятий как основы производной системы понятий информатики и вычислительной техники напрашиваются в первую очередь понятия математики, особенно таких ее разделов, как математическая логика, теория алгоритмов, теория вычислимости, теория доказательств и т.п. С точки зрения такого подхода вычислительная техника в известном смысле превращается в раздел математики. Возможен, однако, другой подход, который в большей

мере соответствует именно технической направленности вычислительной техники. Он заключается в попытке определить новые понятия, исходя не из математических, а из физических основ информатики, более точно, из тех свойств и характеристик, которые по физическим соображениям являются обязательными и всеобщими для всех реальных систем переработки информации. Отысканию таких общих характеристик посвящен ряд работ автора /7-12/. Настоящая статья является кратким обзором этих работ и одновременно их продолжением, однако уже с несколько другой, а именно терминологической точки зрения.

В основе всех дальнейших рассуждений лежит следующий факт. Любой процесс переработки информации описывается "событийно", т.е. через конечное множество событий, а система переработки информации представляет собой "иерархически-событийную" систему. Дадим определение этих понятий.

Событием в некоторой системе C называется наступление или изменение некоторого определенного состояния системы C или составной ее части. Совокупность событий в C, между которыми существует причинно-следственная связь, называется процессом в C. Точнее, процесс – это множество событий, полуупорядоченное отношением причинно-следственной связи. Временную последовательность событий процесса будем называть потоком событий и будем говорить об описании процесса потоком событий. Если процесс описывается конечным числом событий, то описание процесса и сам процесс называется событийным. Система называется событийной, если процессы, протекающие в ней, описываются событийно. Обратим внимание на то, что событие может произойти в любой момент времени, т.е. ось времени является непрерывным, а не дискретным множеством точек. По этой причине мы предпочитаем говорить о событийном, а не о дискретном процессе. Слово "событийно" указывает на то, что ось времени сегментируется последовательностью событий. Если процесс, протекающий в многоуровневой иерархической системе, описывается на всех уровнях событийно, то будем говорить, что процесс описывается многоуровневым потоком событий (МПС). Термин "Иерархия" используется здесь в общепринятом смысле. Его точное определение будет дано в главе 4.

Мы не будем останавливаться на доводах, проведенных в /9/, из которых вытекает необходимость именно событийного описания информационных процессов. Укажем только на то, что элементарные события технической системы переработки информации – это переходы бистабильных физических элементов из одного состояния в другое, а события самого высокого уровня – это начало и конец выполнения программ пользователей. Забегая вперед, укажем на то, что главную роль при событийном описании играет событие окончания выполнения операции оператором и

начало выполнения следующей операции тем же самым или другим оператором. Если пренебречь временным расстоянием между окончанием предыдущего и началом следующей операции, то событием является передача операнда или передача управления между операторами (имея в виду оператор языка программирования).

Событийное описание характерно не только для процессов переработки информации, но и для многих других процессов. В технологических процессах, например, событием является завершение технологической операции, т.е. появление готовой детали (готового полуфабриката). В измерительных процессах событием является завершение процесса измерения некоторой величины, т.е. появление значения этой величины в виде числа. Это сходство технологических и измерительных процессов с информационными процессами существенно с точки зрения совместимости терминологии программирования и его приложений, поскольку важными областями применения вычислительной техники как раз и являются управление технологическими и измерительными процессами.

Поставленную выше задачу выработки системы понятий информатики и вычислительной техники сформулируем следующим образом: разработать систему понятий для описания любых событийных процессов. Выполнение этой задачи можно рассматривать как первый шаг в направлении создания четкого и одновременно достаточно общего предметного языка, который был понятен и удобен для специалистов разных областей, таких, как информатика, производство, измерительная техника, а также математика, физика и другие области естественных наук. Поставленная цель означает, в частности, что такие центральные понятия, как оператор, операция, операнд, процесс и т.п. должны быть общими в области вычислительной техники и в областях ее применения. Созданию системы понятий и терминов, удовлетворяющих указанным требованиям, посвящена настоящая статья.

2. Реальные и формальные операторы

Введем следующее обобщенное понятие оператора: оператором называется реальный объект (например, человек или прибор) или предписание, который (которое) сопоставляет входному операнду $x \in X$ выходной операнд (результат) $y \in Y$. В случае реального объекта оператор называется реальным, в случае предписаний – формальным. Операнд называется составным, если он состоит из нескольких компонент, например из компонент вектора. Между операторами составной операнд может передаваться в виде временной последовательности его компонент. Если x – временная последовательность и если перерабатывающий оператор после поступления каждой компоненты x_i вырабатывает очередную выходную компоненту y_i, зависящую от поступивших до этого момента входных

компонент, но не зависящую от будущих компонент (т.е. если оператор работает как конечный автомат), то он называется оператором с памятью или автоматом. В противном случае, если x -простой (несоставной) или если y (и тем самым все компоненты y_i) зависит от всех x_i в целом, т.е. если y вырабатывается только после поступления всех x_i, то оператор называется оператором без памяти.

Оператор, который может выполнять разные операции (осуществлять разные отображения $X \to Y$), называется переменным или управляемым. Конкретная операция, выполняемая переменным оператором, задается управляющей информацией u, так что оператор выполняет сопоставление $(u,x) \mapsto y$ и осуществляет отображение $U \times X \to Y$, где $u \in U$. Пару (u,x) можно рассматривать как обобщенный входной операнд и соответственно x называть рабочим операндом, а u управляющим операндом. Начальное состояние оператора с памятью (т.е. содержание памяти в момент поступления первой компоненты составного операнда) можно рассматривать как управляющую информацию.

Оператор называется однозначным, если отображение $f: U \times X \to Y$ однозначно, т.е. если функция $y=f(u,x)$ однозначна. В случае оператора с памятью однозначность означает, что все компоненты выходного операнда вырабатываются под действием одного и того же управляющего операнда. В дальнейшем под оператором всегда будет пониматься однозначный оператор и, если не сказано другое, оператор без памяти. Как выясняется в дальнейшем, оператор может "приобретать память" при декомпонировании.

Для выполнения операции, предписанной формальным оператором, необходим исполнитель, т.е. реальный оператор, выполняющий предписание. Мы называем его интерпретатором (int). Если к формальному оператору (fop) присоединить его интерпретатор, то получится реальный оператор (rop). Выразим это обстоятельство следующей записью:

$$(int;fop) - : rop \quad . \qquad (I)$$

Знак -:(или :-) называется знаком компоновки. Выражение (I) - частный случай применения этого знака. Запись (b;a) -:c или c:- (b;a) может читаться по-разному, например, a компонует (конструирует, создает, формирует, организует) c из b. Если из контекста ясно, из каких компонентов c компонуется, то применяется запись a -:c и читается как : a компонует c. В (I) int можно рассматривать как управляемый оператор, fop как управляющую информацию, фиксирующую операцию, выполняемую интерпретатором, так что интерпретатор выполняет сопоставление $(fop,x) \mapsto y$. Если интерпретатор представляет собой ЭВМ, то fop - программа, x -исходные данные, y -результат.

Забегая вперед, отметим, что запись (I) можно истолковать и так: согласно предписанию fop компонуется реальный оператор rop из операторов-компонентов, которыми располагает интерпретатор int.

3. Сети операторов

Совокупность операторов, связанных между собой и, может быть, связанных с окружением каналами передачи операндов, называется сетью операторов. Каналом передачи может быть, например, транспортное средство, конвейер, электрический проводник, общая шина и т.п. В случае сети формальных операторов передача операндов осуществляется интерпретатором. Сеть операторов с внешними каналами передачи (с внешними входом и выходом) представляет собой оператор. Он называется составным оператором в отличие от операторов сети, называемых операторами-компонентами. В качестве примеров сетей реальных операторов можно назвать совокупность станков и транспортных роботов автоматического металлообрабатывающего цеха, комбинационную схему, многопроцессорную ЭВМ, а примерами сетей формальных операторов являются сложные математические выражения, а также программы, состоящие из отдельных модулей, и пакеты прикладных программ. Составной оператор, т.е. сеть операторов с входом и выходом, обозначим через $\tilde{O}_P$. Тот факт, что $\tilde{O}_P$ компонуется из элементов множества O_P операторов-компонентов, запишем так:

$$\tilde{O}_P :- (O_P; R_O) \quad , \quad \text{где} \qquad (2)$$

$$R_O = \{ (op_i, op_j) \mid \text{передача операнда от } op_i \text{ к } op_j \text{ возможна} \}$$
$$op_i, op_j \in O_P .$$

Отношение R_O называется отношением связи.

Для более детального описания структуры реального составного оператора и процесса выполнения составной операции (т.е. операции составного оператора) необходимо учитывать два обстоятельства. Во-первых, в случае асинхронной работы операторов-компонентов, а также в присутствии обратных связей (контуров) в сети, может возникать необходимость буферизации потоков операндов. Для этого существуют различные устройства. Для формального описания вводим понятие места. Место операнда - это устройство для временного хранения одного операнда. Совокупность мест может быть сосредоточена в специальном устройстве (склад, запоминающее устройство). В ЭВМ местом является регистр или совокупность регистров (ячеек памяти).

Во-вторых, система каналов передачи должна обеспечить необходимое распределение потока операндов через сеть операторов. Поток может быть разветвленным и может быть управляемым. Для описания конфигурации и управляемости потока вводим понятие узла потока операндов. Узлы

потока классифицируются по двум признакам, по топологии и по управляемости. По топологии различаем два класса: узлы слияния - это узлы с двумя или несколькими входами и одним выходом, и узлы ветвления - это узлы с одним входом и двумя или несколькими выходами. По управляемости также различаем два класса: неуправляемые узлы - это узлы, входы и выходы которых постоянно открыты, и управляемые узлы - это узлы, у которых в любой момент времени только один вход и один выход открыты. По обоим признакам можно различать 4 вида узлов потока: I) неуправляемые узлы слияния или узлы с синхронными входами; 2) неуправляемые узлы ветвления или развилки; 3) управляемые узлы слияния ; 4) управляемые узлы ветвления.

Класс развилок распадается на два подкласса: I) разделяющие развилки, они разлагают входной операнд на части и выдают на разных выходах разные части (например, разные компоненты входного вектора); 2) копирующие развилки, они размножают входной операнд и выдают его на всех выходах. Используя только что введенные понятия, дадим определение графа потока операндов (ГПО):

$$\text{ГПО} :- (\text{Op} \cup P \cup F;\ R_O) \quad , \tag{3}$$

где Op - множество операторов (точнее - множество названий операторов), P - множество мест операндов и F - множество узлов потока операндов. ГПО-это граф с тремя классами вершин. Если места и узлы потока рассматривать как специальные виды операторов, то мы вернемся к старой записи (2) составного оператора, подразумевая под $\tilde{o}_P$ сеть обобщенных операторов. Можно еще дальше идти, и каналы передачи (т.е. элементы отношения R_O) также рассматривать как операторы. Тогда составной оператор описывается обобщенным графом потока операндов

$$\text{ГПО}^* : - (\tilde{o}_P^* ; R_O^*) \quad , \quad \tilde{o}_P^* = \text{Op} \cup P \cup \text{Op}_K \quad ,$$

где множество Op_K коммуникационных операторов включает в себя все средства передачи операндов между "обрабатывающими" операторами и местами хранения операндов. Обобщенное отношение связи R_O^* определяется так же, как и R_O. Однако в случае реальных сетей - пока только о них идет речь - за элементами R_O скрываются технические устройства и процессы передачи, в то время как элементы R_O^* обозначают только абстрактную возможность передачи. Обобщенный граф ГПО^* является наиболее адэкватным средством описания, когда передача операндов осуществляется роботами, а также в случае коммуникационных сетей, включая вычислительные комплексы. Обратим внимание на то, что ГПО (а также ГПО^*), содержит все элементы, необходимые для реализации событийных систем с любой структурой (если не включить в рассмотрение реализацию управления). Иными словами, структуру любой событийной системы

(кроме управления) можно описать графом потока операндов. Накладывая те или другие ограничения на ГПО, например, на допустимые каналы связи или типы узлов потока, можно определить разные классы составных операторов (событийных систем). Добавляя те или другие ограничения относительно моментов передачи операндов, например, требования синхронизации, можно определить разные классы событийных процессов, в том числе и классы вычислительных процессов. На этих проблемах мы не будем останавливаться, они рассмотрены в /IO/. Укажем только на одно особенно важное структурное свойство, а именно присутствие или отсутствие обратных связей. Сети операторов, составные операторы и графы потока операндов называются бесконтурными, если они не содержат обратных связей (контуров). В противном случае они называются контурными.

В этой главе до сих пор шла речь о сетях реальных операторов. Все рассуждения можно повторять относительно сетей формальных операторов, имея в виду при этом, что операцию, предписанную формальным оператором, выполняет не сам оператор, а интерпретатор. Очевидно, что все введенные классы узлов потока операндов сохраняют свой смысл и в случае сетей формальных операторов. Составной оператор, ГПО которого содержит управляемые узлы потока или управляемые операторы-компоненты, является управляемым. Для его управления, т.е. для фиксации выполняемой составной операции, необходима соответствующая управляющая информация u . Ее можно представить в виде пары (u_{Op}, u_F), где u_{Op} - информация, фиксирующая операции, выполнимые управляемыми операторами - компонентами, а u_F - информация, управляющая потоком операндов. Тогда мы можем написать

$$(\tilde{o}_P ; u) -: o_P \quad \text{или} \quad (\text{ГПО} ; u) -: o_P \quad , \tag{4}$$

где $u = (u_{Op}, u_F)$,

словами: управляющая информация u формирует из управляемого составного оператора $\tilde{o}_P$ (или из ГПО) фиксированный оператор o_P . Во второй записи (4) ГПО выступает в качестве управляемого формального составного оператора. Это оправдано, поскольку ГПО может рассматриваться не только как описание составного оператора, но и как предписание для некоторого интерпретатора, например, для человека. ГПО вместе с управляющей информацией будем называть планом потока операндов (ППО), поскольку им полностью задается план выполнения составной операции, т.е.

$$(\text{ГПО} ; (u_{Op}, u_F)) -: \text{ППО} \quad . \tag{5}$$

Если ППО выступает как предписание (как программа), то указание мест операндов может быть опущено. ППО, не содержащий места операндов, называется редуцированным ППО и обозначается через $\overline{\text{ППО}}$. Из (3) и (5) следует

$$\text{ППО}' :- ((Op \cup F ; R_O); (u_{Op}, u_F)) \quad . \qquad (6)$$

ППО' содержит все необходимые сведения и никаких лишних сведений для того, чтобы интерпретатор мог выполнить описанную составную операцию (при условии, конечно, что он может выполнить все операции -компоненты). Это свойство ППО' лежит в основе как идеи так называемой ЭВМ поточного типа, так и идеи функционального или аппликативного программирования, осуществимого с помощью таких языков, как LISP или APL. Отсутствие мест в ППО' означает, что в функциональных (аппликативных) программах отсутствуют адреса и названия операндов. Большое значение плана потока операндов для анализа и синтеза событийных систем и процессов следует из того факта, что любой реальный составной оператор (например, аппаратное обеспечение ЭВМ) можно представить в виде ППО, а любой формальный составной оператор (например, программу для ЭВМ)-в виде ППО'.

Обращает на себя внимание тот факт, что программы, написанные на таких языках, как FORTRAN или ALGOL, а также их блок-схемы, не представляют собой ППО' в явной форме. Дело в том, что они не основываются на отношении связи R_O, а на отношении активации R_A:

$$R_A = \{(op_i, op_j) \mid \text{передача активности от } op_i \text{ к } op_j \text{ возможна}\}.$$

Активность - это состояние оператора, сохраняющееся, пока его операция выполняется. В программировании передачу активности принято называть передачей управления. В аналогии с ГПО можно определить граф потока активности ГПА:

$$\text{ГПА}' :- (Op \cup F ; R_A) \quad , \qquad (7)$$

где Op - множество операторов -компонентов некоторой сети операторов (некоторого составного оператора). Очевидно, что понятие места здесь не имеет смысла, а понятие узла потока можно перенять от ГПО в смысле узла потока активности. Только неуправляемые узлы слияния и разделяющие развилки теряют свой смысл. При попытке определить план потока активности ППА в аналогии с ППО возникают трудности. В то время как из ППО однозначно следует, какие операторы обрабатывают какие операнды, из аналогично построенного плана потока активности это не следует, если в ППО вместо R_O поставить R_A. Для того, чтобы ППА полностью задал план выполнения составной операции, необходимо указать операнды каждой операции. Это достигается тем, что операторы заменяются командами. Команда - это совокупность названий данного оператора и входного и выходного операндов. Вследствие такой замены план потока активностей запишется как

$$\text{ППА}: - ((K \cup F ; R_A); (u_{Op}, u_F)) \quad , \qquad (8)$$

где K - множество команд. Каждый формальный составной оператор (например, каждую программу) можно представить в виде ППА, причем обычно во многих вариантах. Причиной этой неоднозначности является тот факт, что ППА накладывает лишние ограничения на временную последовательность выполнения отдельных команд. Поэтому ППО' является более фундаментальным представлением формальных операторов. Тем не менее, самые ходовые языки, например, все алголоподобные языки, предназначены для представления ППА (а не ППО) в подходящей для ЭВМ форме. Это - следствие фон-неймановских принципов архитектуры ЭВМ.

ППА и соответственно программы и языки можно классифицировать в зависимости от того, какие узлы потока активности они содержат или могут описывать. Например, программа, написанная на языке FORTRAN, представляет собой ППА без неуправляемых узлов потока. Отсюда следует, что в любой момент только один единственный оператор находится в состоянии активности, откуда дальше следует, что управляемые узлы слияния излишни. Действительно, в FORTRANe имеется только одно языковое средство для описания узлов потока, а именно (обобщенный) оператор условного перехода, описывающий управляемый узел ветвления вместе с информацией управления u_F (т.е. предикатом перехода). Существуют языки, обладающие средствами для выражения неуправляемых узлов потока активности, например, ALGOL68 и Concurrent Pascal.

Заканчивая эту главу, подчеркнем, что описанный здесь подход к описанию структуры составного оператора отличается от широко принятого подхода тем, что элементарными средствами компоновки не служат последовательная, параллельная и обратная связи, а более элементарные и одновременно более мощные средства - узлы потоков операндов и активности.

4. Иерархия операторов

Составной оператор может в качестве оператора-компонента входить в сеть операторов более сложного типа, т.е. он может служить оператором-компонентом при компоновке составного оператора более высокого уровня. Таким образом, может быть построена многоуровневая иерархия операторов. Перепишем (4) для перехода от уровня c к уровню c+1:

$$(\widetilde{Op}^{(c)} ; u^{(c+1)}) -: op^{(c+1)} \quad , \qquad (9)$$

где c - номер ступени композиции или номер уровня или слоя иерархии. Говоря о слое, считаем, что все операторы одной и той же ступени композиции относятся к одному слою. (Более точное определение понятий слоя и ступени композиции приводится в 5 главе). Самый "низкий" слой (c=0) и операторы $op^{(0)}$ этого слоя называются элементарными.

Оператор любого слоя строится в конечном счете из элементарных операторов.

Для описания структуры иерархии введем отношение принадлежности $R_{при}$:

$$R_{при} = \{ (op_i, op_j) \mid op_i \dot{\in} op_j \} , \qquad (10)$$

где $op_i \dot{\in} op_j$ означает, что op_i является непосредственным оператором-компонентом оператора op_j. Множество Op операторов, полуупорядоченное отношением принадлежности, называется иерархией принадлежности операторов и обозначается через $\hat{O}p_{при}$, т.е.

$$\hat{O}p_{при} :- (Op; R_{при}) . \qquad (11)$$

Существует определенная аналогия между множеством Op, полуупорядоченным отношением принадлежности, и множеством подмножеств P(M) множества M, полуупорядоченным отношением "быть подмножеством". Иерархия может состоять из реальных или из формальных операторов. Она называется соответственно реальной или формальной иерархией.

При выполнении составной операции, т.е. операции неэлементарного оператора, происходит процесс пошаговой компоновки снизу вверх. В случае формальной иерархии компоновку осуществляет интерпретатор, например, человек, который вычисляет значение сложного арифметического выражения. В случае реальной иерархии процесс компоновки управляется соответствующей управляющей информацией u, представляющей собой совокупность управляющих сообщений, (в самом простом случае - сигналов), генерируемых самой иерархией и передаваемых "сверху вниз". Проанализируем подробнее, что представляет собой эта информация и как она генерируется. Рассмотрим событие запуска оператора $op^{(c)}$ слоя c. Для простоты будем считать, что оператор запускается сообщением $u_{op}^{(c+1)}$, фиксирующим операцию, выполняемую оператором $op^{(c)}$. Верхний индекс сообщения обозначает передающий слой. Поскольку оператор $op^{(c)}$ составной, вслед за его запуском должна генерироваться последовательность $u^{*(c)}$ сообщений, управляющих узлами потока операндов и фиксирующих и запускающих операторы слоя $c-1$. Генератор, генерирующий последовательность $u^{*(c)}$, представляет собой реальный оператор, обладающий в общем случае памятью. Мы его называем управляющим автоматом (этот термин перенят из книги /2/. Автомат управления выполняет сопоставление $u_{op}^{(c+1)} \longmapsto u^{*(c)}$. Последовательность u^{*} представляет собой декомпозицию глобальной информации управления u в (4) и (9) или (u_{Op}, u_F) в (6). Временное расстояние между отдельными сообщениями последовательности u^{*} зависит от времени выполнения составных операций. Если необходимые временные затраты заранее не известны, то управляющие сообщения должны генерироваться в зависимости не только от u_{op}, но и от извещений v, через которые операторы-компоненты осведомляют управляющий автомат об окончании текущей операции. Генератор управляющих сообщений осуществляет сопоставление

$$u_{op}^{(c+1)} \circ v^{*(c-1)} \longmapsto u^{*(c)} , \qquad (12)$$

где $\circ$ - знак конкатенации. Из (12) следует, что генератор управляющих сообщений действительно является оператором с памятью, т.е. автоматом. Заметим, что в случае формальной иерархии управляющая информация не выступает явно в виде последовательности сообщений. Она закодирована в формальном операторе средствами используемого языка, например, скобками, правилами приоритета и т.п.

Если управление узлами потока зависит от самого управляемого процесса, например, от значений тех или других операндов, то среди сообщений v должны находиться и такие, которые осведомляют управляющий автомат о значениях этих операндов.

Перепишем (9) для реальной иерархии, подставляя вместо управляющей информации u ее генератор, т.е. автомат a:

$$(\tilde{R}op^{(c)}; a^{(c+1)}) \; -: \; rop^{(c+1)} , \qquad (13)$$

где $a^{(c+1)}$ - автомат слоя $c+1$, генерирующий управляющие сообщения для слоя c. Запись (13) можно читать так: автомат $a^{(c+1)}$ формирует реальный оператор $rop^{(c+1)}$ из операторов сети $\tilde{R}op^{(c)}$ (или из управляемого составного оператора $\tilde{R}op^{(c)}$).

Обозначая через $A^{(c)}$ множество управляющих автоматов слоя c, можно написать

$$(\tilde{R}op^{(c)}; A^{(c+1)}) \; -: \; \tilde{R}op^{(c+1)} .$$

Поскольку элементарные операторы не компонуются, имеет место $A^{(o)} = \emptyset$. Однако иногда удобно использовать обозначения $a^{(o)}$ и $A^{(o)}$, отождествляя

$$a^{(o)} \equiv rop^{(o)} \; ; \; A^{(o)} \equiv Rop^{(o)} . \qquad (14)$$

Для более точного описания структуры реальной иерархии введем кроме $R_{при}$ (см.(10)) еще два отношения, отношение подчинения $R_{под}$ и отношение компоновки $R_{ком}$:

$$R_{под} = \{ (a_i, a_j) \mid a_i \dagger a_j \} , \qquad (15)$$

где $a_i \dagger a_j$ означает, что a_i подчиняется a_j (или что a_j управляет a_i);

$$R_{ком} = \{ (a, op) \mid a \; -: op \},$$

где $a \; -: op$ означает, что a компонует op. Множество операторов-компонентов, из которого op компонуется, не указано; оно задается

отношением $R_{при}$. (Примечание. Здесь и в дальнейшем будем писать op вместо ^{r}op, если это не вызывает недоразумений). Множество A автоматов, полуупорядоченное отношением подчинения, называется иерархией подчинения и обозначается через $\hat{A}$:

$$\hat{A} :- (A; R_{под}) .$$

Используя отношение $R_{ком}$, можно объединить иерархии $\hat{op}_{прц}$ (см. (II)) и $\hat{A}$ в одну иерархию операторов $\hat{op}$:

$$\hat{op} :- (op \cup A; R_{\wedge}) , \quad \text{где}$$

$$R_{\wedge} = \{((a_i, op_i), (a_j, op_j)) \mid a_i -: op_i \wedge a_j -: op_j \wedge a_i \neq a_j \wedge op_i \in op_j\} . \qquad (I6)$$

С учетом (I4) выражения (I5) и (I6) остаются в силе и для элементарного слоя. Иерархия, построенная согласно (I3) и (I6), негибкая, поскольку она может выполнять только ограниченное число операций, а именно столько, сколько имеется управляющих автоматов, включая элементарные операторы. Выполняемая операция фиксируется тем, что запускается соответствующий автомат. Чтобы обеспечить бо́льшую гибкость, надо систему сделать интерпретирующей с тем, чтобы выполняемую операцию можно было задавать предписанием (формальным оператором). Этого можно достичь заменой автоматов так называемыми интерпретирующими автоматами a_{int}, которые генерируют необходимые управляющие сообщения согласно соответствующим предписаниям. По аналогии с (I) напишем

$$a :- (a_{int}; fop) , \qquad (I7)$$

имея, однако, в виду, что a_{int} на самом деле не интерпретирует (выполняет) fop, а только генерирует необходимые для его выполнения управляющие сообщения. Если автоматы иерархии $\hat{A}$ декомпонировать согласно (I7), то совокупность всех fop, входящих в эту иерархию, образует иерархию формальных операторов (например, программных модулей). Ее можно представить в виде (9). При этом управляющая информация содержится в соответствующем fop, сформулированы на языке соответствующего a_{int}.

Иерархию операторов $\hat{op}$ (I6), содержащую по меньшей мере один интерпретирующий управляющий автомат, будем называть интерпретирующей иерархией операторов (ИИО). Большое значение ИИО для анализа и синтеза систем со сложным управлением следует из того факта, что любую многоуровневую событийную систему (например, вычислительную систему, включая аппаратное и программное обеспечение, можно представить в виде ИИО. Накладывая те или другие ограничения на отношения принадлежности и подчинения и на число управляющих автоматов, можно определить разные классы иерархий (см.следующую главу). Управляющие

функции автоматов могут быть возложены на самих операторов. Это можно осуществить двумя путями: переходом к локальному управлению или периодическому переключению того или другого оператора из режима выполнения "рабочей" операции в режим управления и назад. Однопроцессорные ЭВМ работают по этому принципу.

Заметим в заключение, что любой процесс, протекающий в иерархической системе, можно описать как многоуровневый поток событий, где каждому событию соответствуют осведомляющее сообщение снизу вверх и управляющее сообщение сверху вниз.

5. Характеристики иерархий

Ряд свойств иерархии удобно определить через свойства соответствующих графов, т.е. графа подчинения $(A ; R_{под})$ и графа принадлежности $(op ; R_{при})$. Оба графа ориентированы и бесконтурны. Установим: дуги обоих графов направлены вниз в направлении уменьшения номера слоя.

Заметим, что в определение иерархии согласно (9) понятие слоя было введено при предположении строго послойной структуры иерархии, т.е. каждый составной оператор компонуется из операторов ближайшего нижележащего слоя. Можно дать более общее определение терминов ступени композиции и слоя: наибольшее расстояние оператора от элементарного слоя в графе принадлежности (т.е. наибольшее число дуг, соединяющих данный оператор с элементарным слоем) называется ступенью композиции оператора. Все операторы с одинаковой ступенью композиции относятся к одному слою.

Перечислим некоторые характеристики иерархии в целом и ее операторов в отдельности, которые играют важную роль при описании, классификации и оценке иерархии под разными аспектами. Предварительно напомним: подграф ориентированного бесконтурного графа, вершины которого достигаемы из заданной вершины, называется подграфом этой вершины.

5.I. Структурные характеристики иерархии

Два оператора или два автомата иерархии называются несвязными, если их подграфы не имеют общих вершин. Они называются несвязными относительно некоторого слоя, если их подграфы не имеют общих вершин в этом слое. Число входных дуг оператора (в графе принадлежности) называется числом разделения оператора. Он равняется числу вышележащих операторов, в которые данный оператор входит в качестве оператора-компонента. Иерархия называется иерархией без разделения операторов, если все операторы неразделимы, т.е. если граф принадлежности является деревом или лесом.

Число входных дуг автомата управления (в графе подчинения) называется числом подчинения автомата. Иерархия называется иерархией с однозначным подчинением, если числа подчинения всех автоматов равны единице, т.е. если граф подчинения является деревом или лесом.

Иерархия называется иерархией со строго послойным компонированием, если каждый составной оператор компонируется из операторов только ближайшего нижележащего слоя. Иерархия называется иерархией со строго послойным подчинением, если каждый автомат подчиняется автоматам только ближайшего вышележащего слоя.

Иерархия подчинения называется линейной, если ее граф является линейным, т.е. если каждая вершина (каждый автомат) имеет не более одной входной и одной выходной дуги. В противном случае иерархия называется разветвленной.

5.2. Режимы работы оператора

Составной оператор работает в параллельном режиме, если два или больше его операторов-компонентов выполняют свои операции одновременно (или с временным перекрытием). В противном случае оператор работает в последовательном режиме.

Составной оператор работает в режиме внешнего параллелизма, если он выполняет свою (составную) операцию над двумя или несколькими входными операндами одновременно, причем над каждым операндом выполняется операция другого оператора-компонента. Режим внешнего параллелизма называется конвейерным, если составной оператор является цепочкой операторов-компонентов, через которую течет последовательность операндов, так что когда k-ый оператор обрабатывает i-ый операнд, $(k+1)$-ый оператор обрабатывает $(i-1)$-ый операнд.

Оператор работает в режиме внутреннего параллелизма, если при обработке одного единственного входного операнда составного оператора два или несколько его операторов-компонентов выполняют свои операции одновременно. Для этого необходимо, чтобы ГПО составного оператора содержал по крайней мере одну развилку.

5.3. Методы управления

Управление составного оператора (или ГПО) называется центральным, если сигналы управления одной его операции генерируются одним единственным автоматом. В противном случае оно называется распределенным.

Управление называется управлением с обратной связью, если оно учитывает события (сообщения) управляемого процесса (см.(I2)). В противном случае оно называется без обратной связи или жестко запланированным.

Управление называется пассивным, если управляющий автомат запускается событиями (сообщениями) управляемого процесса. В противном случае оно называется активным. Активное управление называется тактовым, если управляющий автомат запускается сигналами тактового генератора (часов).

6. Заключительное замечание

В предыдущих главах по ходу решения поставленной в первой главе задачи была введена система понятий, включающая около 80 понятий и терминов. Эти понятия были введены на достаточно высоком уровне абстракции, так что они применимы в самых различных областях, если только речь идет о событийных процессах. Многие понятия применимы и для описания непрерывных процессов, описываемых дифференциальными уравнениями. Это касается в первую очередь понятий, связанных структурой систем и процессов.

Абстрактность и обобщенность данных понятий влечет за собой необходимость разного рода уточнений, т.е. введения более конкретных понятий и обозначающих их терминов. Приведем один пример.

Термином "операнд" обозначаются объекты, которые в зависимости от конкретной области существенно отличаются друг от друга. Так, операнд токарного станка (т.е. деталь) является носителем некоторого измеряемого значения (например, диаметра), подлежащего изменению при обработке. Операнд системы переработки информации (т.е. информация) является носителем некоторого интерпретируемого значения (смысла, например, числа), которое нельзя определить простым измерением. Исходя из этого, можно ввести термины измеряемый операнд и интерпретируемый операнд. Производственно-технологический процесс переводит измеряемые операнды в измеряемые, информационный процесс переводит интерпретируемые операнды в интерпретируемые, а измерительный процесс переводит измеряемые операнды в интерпретируемые. (Эти вопросы подробно рассмотрены в /9/). Можно ввести соответствующие конкретизирующие термины для термина "Оператор", например, обрабатывающий, измеряющий и интерпретирующий операторы.

Укажем на некоторые возможности, вытекающие из высокого уровня абстракции основных понятий метода МПС: I) описание систем и процессов самой различной природы на одном и том же языке; 2) создание единой базы терминов вычислительной техники, программирования и ее применений; 3) описание, классификация и оценка аппаратного и программного обеспечения, а также языков программирования с помощью единой системы признаков; 4) повышение эффективности обучения путем введения единой системы понятий, используемой преподавателями различных предметов и специальностей информатики.

За поддержку, дискуссию и ценные советы автор выражает благодарность А.А.Корнейчуку и В.Шуберту.

ЛИТЕРАТУРА

I. Вельбицкий И.В., Зыкин Г.П., Корнейчук А.А. О формировании банка терминов по программированию. Международный семинар по машинному переводу (Москва, 1983 г.). Тезисы докладов. ВЦП, М., 1983,с.58-59.

2. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра, языки, программирование. Киев, „Наукова думка", 1974.

3. Горбенко В.Н., Зыкин Г.П., Корнейчук А.А. Деятельность терминологической целевой подгруппы РГТП. Тезисы докладов всесоюзного семинара "Промышленная технология создания и применения программных средств в организационном управлении и НИОКР". Институт математики УНЦ АН СССР, Свердловск, 1984.

4. Корнейчук А.А. О некоторых проблемах технологии программирования и ее информационного обеспечения. ОИЯИ, 11-84-319, Дубна, 1984.

5. Шишмарев А.И., Заморин А.П. Англо-русско-немецко-французский толковый словарь по вычислительной технике и обработке данных. Под ред.акад. А.А.Дородницына. Москва, "Русский язык", 1981.

6. Engelien M.,Stahn H. Software-Engineering,ARS-Technologie Berlin, Akad.-Verl.,1984.

7. Jungclaussen H.:Grundlagen der Kybernetik III. Asynchrone Operatorennetze. Teil 1. Zentralinstitut für Kernforschung Rossendorf bei Dresden. ZfK-420,1980.

8. Jungclaussen H.: как /7/. Teil 2. ZfK-475, 1982.

9. Jungclaussen H.: как /7/. Teil 3. ZfK-501, 1983.

10. Jungclaussen H.: как /7/. Teil 4. ZfK-536, 1984.

11. Jungclaussen H.: как /7/, Teil 5 в печати.

12. Jungclaussen H.: Zur Einheit von Hardware und Software. Wiss. Z. Techn. Univers. Dresden 34(1985)H.4, s.95-102.

13. Schumann J.,Gerisch M.: Softwareentwurf. Berlin: Verl. Technik, 1984.

Рукопись поступила в издательский отдел
5 августа 1986 года.

Юнгклауссен Х. P11-86-542

Метод многоуровневого потока событий /МПС/
для описания процессов в сетях и иерархиях операторов

Излагается новый метод описания процессов в сетях и иерархиях операторов. Предлагается соответствующая система понятий.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1986

Перевод О.С.Виноградовой

Jungklaussen H. P11-86-542

Multilevel Event Stream /MES/ Method
for Description of Processes in Operator Nets
and Operator Hierarchies

A new method for description of processes into operator nets and operator hierarchies is stated. A corresponding concept system is proposed.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1986