Сайт: Параллельные алгоритмы и логика: **paralg.ucoz.com**

g4144_s106	23.04.16

Статья. Библиографические данные:

Мунасыпов Р.А., Житников А.П.
Структурный фактор робототехнической параллельной алгоритмики.
// Сб. матер. междунар. научно-практ. форума: Современные технологии преподавания естественно-научных дисциплин в системе общего и профессионального образования.
– Борисоглебск: ООО "Кристина и К", 2016.
– С. 52-72.

СОДЕРЖАНИЕ

СТРУКТУРНЫЙ ФАКТОР РОБОТОТЕХНИЧЕСКОЙ ПАРАЛЛЕЛЬНОЙ АЛГОРИТМИКИ

Мунасыпов Р.А., Житников А.П.
Уфимский государственный авиационный технический университет

Данная статья является продолжением тематики, подробно изложенной в статьях [1, 2] относительно наличия управляющей структуры и управляемой информационной структуры в составе алгоритмов вычислительного и, вообще, математического назначения. Но в данном случае на этой основе кратко излагается проблемная специфика управляющей и управляемой (материально-информационной) структуры в составе параллельных (и, в частности, последовательных) алгоритмов технического, технологического и робототехнического назначения.

1 ОБЩАЯ СТРУКТУРА ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ

Последовательно-параллельный структурный аспект

Понятие "параллельные алгоритмы" предполагает понятие "последовательные алгоритмы", и все это содержит ***структурный классификационный аспект*** – деление алгоритмов (по их структуре, то есть строению) на два структурных класса:

последовательные и параллельные алгоритмы управления последовательными и параллельными (во времени) дискретными процессами – с последующим их классификационным делением на составляющие их подклассы (по разным основаниям классификации процессов и алгоритмов).

При этом последовательные алгоритмы:

- составляют, как правило, внешнюю и внутреннюю опорную среду формирования параллельных структур алгоритмов (то есть алгоритмов с наличием параллелизма в составе их разных фрагментов);
- удобно условно интерпретировать как предельный, вырожденный, но базисный частный случай параллельных алгоритмов – с вырожденной единичной степенью параллелизма (p = 1).

Ведущий структурный фактор – организация управления

В основе указанного выше последовательно-параллельного структурного аспекта лежит более глубокий структурный фактор, впервые обнаруженный в составе вычислительных программ и алгоритмов – структурный фактор ***организации управления*** (в той или иной форме его отражения). Он заключается в выделении двух базовых компонент в составе общей структуры любых вычислительных (и вообще математических) алгоритмов, что отражает состав реализующих их алгоритмических процессов и алгоритмических исполнительных систем (Рис. 1.1, Рис. 2.1– Рис. 2.3):

1) ***Управляющая структура*** – система операторов команд алгоритма, взаимосвязанных между собой простыми и сложными отношениями сле-

дования во времени (при их исполнении). Такие отношения определяют связи и пучки связей ***передачи управления*** между операторами. При этом:

- управляющая структура именуется также как ***поток исполнения*** или ***поток управления*** алгоритма (поток команд управления);
- это символьное отражение некоторого реального потока:

управляющая структура в составе текста (или схемы) алгоритма отражает реальный поток (команд) управления в составе управляющей структуры исполнительной системы (в некотором ***канале управления***).

2) Управляемая ***информационная структура*** – система операторов команд алгоритма, реализующих некоторые преобразования данных и взаимосвязанных между собой информационными связями. Такие связи представлены общими данными (одноименными общими переменными) разных операторов. При этом:

- информационная структура именуется также как ***поток данных*** (поток обработки данных) алгоритма;
- это символьное отражение некоторого реального потока:

(управляемая) информационная структура в составе текста (или схемы) алгоритма отражает реальный поток данных в составе (управляемой) информационной структуры исполнительной системы (в некотором ***канале данных*** – канале продвижения, хранения и обработки данных).

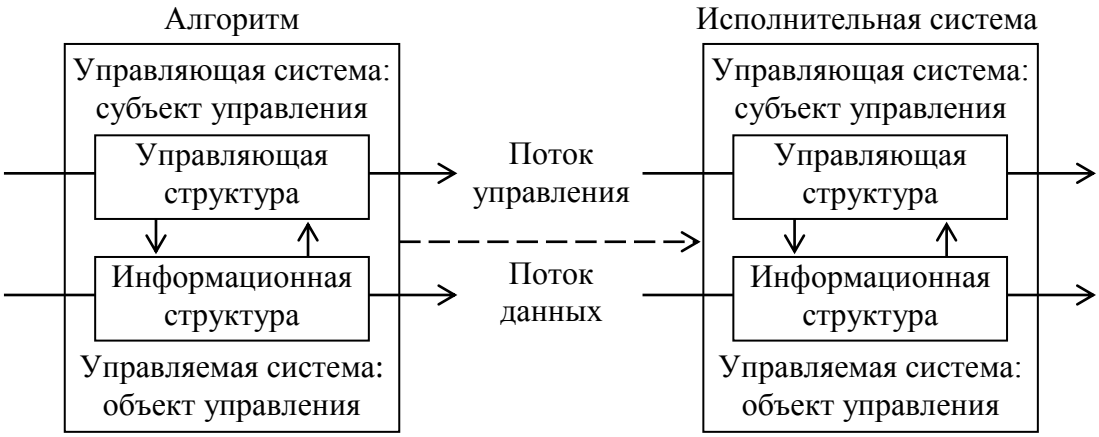


Рис. 1.1. Управление в вычислительном алгоритме и его исполнительной системе

Соотношение двух базовых структур алгоритмов

Информационная структура в ее представлении некоторой схемой (графом) связи данных является "информационным ядром" общего алгоритма решения задачи [3]. При наличии потенциального параллелизма обработки данных она определяет множество всевозможных вариантов допустимых управляющих структур (потока управления) – с разными допустимыми порядками вычислений и с разной степенью параллелизма: от вариантов максимального и частичного параллелизма до разных последовательных вариантов (вырожденный единичный параллелизм).

Этот аспект относительно более просто, четко и ясно отражается в составе достаточно простых образцов вычислительных алгоритмов. Он подробно анализируется в статьях [1, 2] (на примере простой вычислитель-

ной задачи опорного концептуального типа, кратко представленной далее). Но в параллельных вычислениях существует такой ***парадокс***:
для простых вычислительных задач ***нет практического смысла их распараллеливать*** в программировании, поскольку накладные ходы расходы времени на формирование и обслуживание параллельных программных структур и режимов перекрывают выигрыш времени параллельного счета.

Для реальных вычислительных задач выявление "информационного ядра" алгоритмов составляет большую проблему ([4], с.197), особенно с большим числом ветвлений и вложенных циклов. Возможны влияние значений исходных данных на логику вычислений и само "информационное ядро" алгоритма, проблемы точности параллельных вычислений и т.п.

Описание допустимых структур потока управления – это относительно более простая проблема (особенно для первичных базовых классов задач). Но для каждой задачи в общем случае таких структур может быть очень много. При этом появляются вторичные проблемы:

- учета общего множества допустимых частных вариантов управляющих структур (разной степени параллелизма);
- их (явного или неявного) объединения в общий вариативный алгоритм;
- управления выбором вариантов порядка вычислений до начала или по ходу счета (в зависимости от наличных ресурсов);
- конкуренции процессов по доступу к общим ресурсам (с понижением степени параллелизма счета), исключения их взаимных блокировок и т.д.;
- оптимизация (выравнивание) загрузки процессоров и т.п.

Общая оценка параллельной проблематики

Основное наименование пособия [3] "Вычислительная математика и структура алгоритмов" имеет очень точное и выразительное дополнение:
"10 лекций о том, ***почему трудно решать задачи*** на вычислительных системах параллельной архитектуры, и ***что надо знать дополнительно***, чтобы успешно преодолевать эти трудности".

"Обсуждаются особенности математического образования по отношению к требованиям параллельных вычислений" и отмечается ([3], с. 33):

- "до сих пор специалистов в области вычислительной математики учили, как ***решать задачи математически правильно***";
- "теперь надо, к тому же, учить, как ***решать задачи эффективно*** на современной вычислительной технике";
- "это совсем ***другая наука***, математическая по своей сути, но которую пока ***почти не изучают в вузах***" (это по состоянию на 2006 г.).

В технической алгоритмике с (относительно быстрым) управлением (относительно более медленными) механическими дискретными процессами обычно нет такого парадокса нецелесообразности распараллеливания простых алгоритмических задач. При этом большое значение имеет первичная ориентировка на вычислительные алгоритмы, простой пример которых приводится далее (в краткой сводке).

2 ПРОСТОЙ ПРИМЕР ВЫЧИСЛИТЕЛЬНОГО АЛГОРИТМА

Далее кратко отражается простой пример из статей [1, 2] для демонстрации понятий управляющей и информационной структуры алгоритма – для вычислительных алгоритмов в данном случае.

Дано: Простая вычислительная формула:

y = Sin(x1 + x2) * Cos(x1 +x2).

Надо: 1) Представить алгоритм пошаговых вычислений по формуле.

2) Обеспечить гибкую (вариативную) алгоритмизацию задачи с учетом допустимых вариантов параллельного и последовательного выполнения (вариативного) ***общего алгоритма***:
выявление допустимых ***частных алгоритмов*** решения задачи.

Простой анализ заданной вычислительной формулы позволяет выявить наличие потенциального параллелизма задачи (степени p = 2). При этом возможны следующие ***частные алгоритмы*** решения задачи, представленные блок-схемами (Рис. 2.1) и псевдокодами (Табл. 2.1, Табл. 2.2).

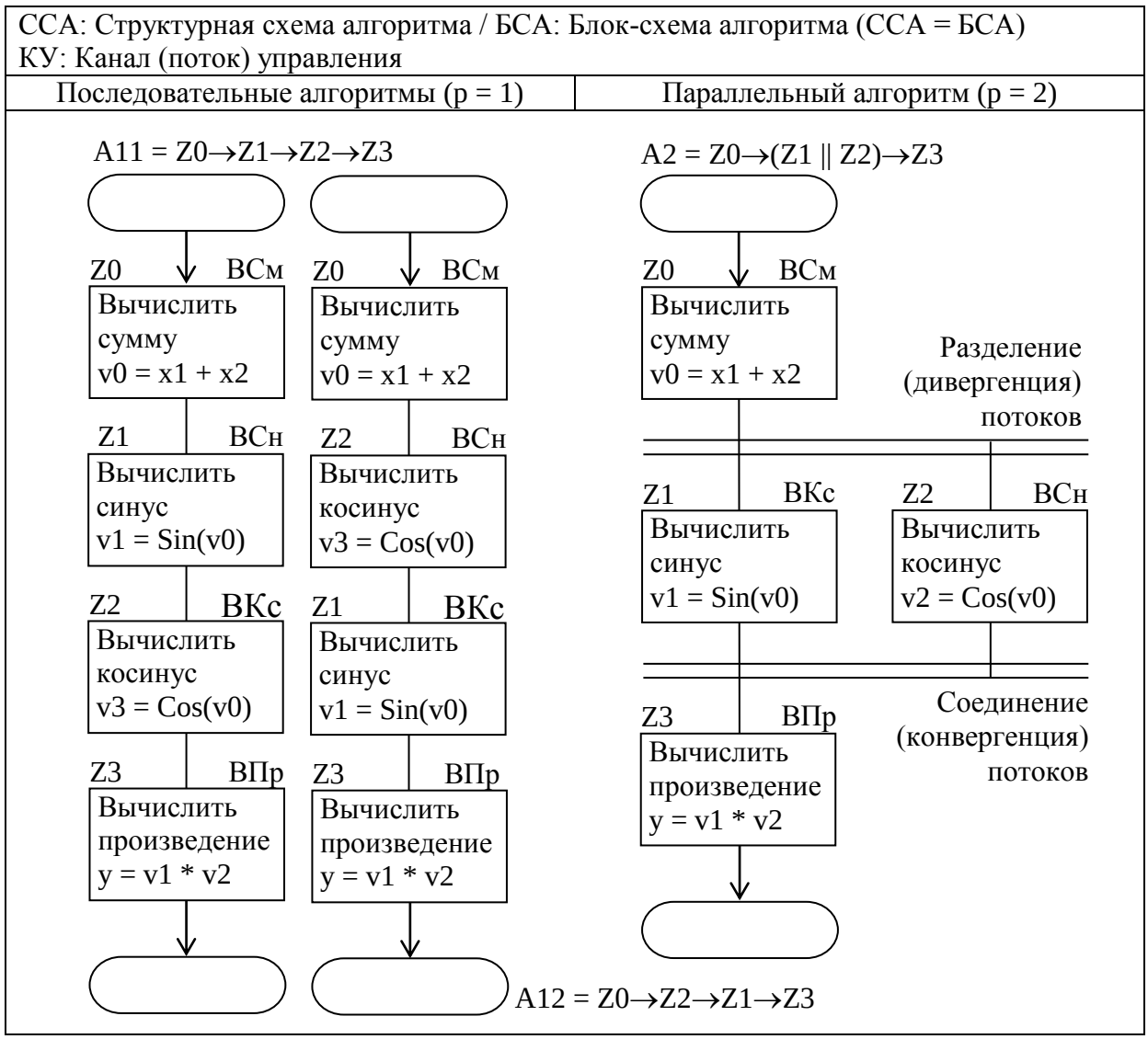


Рис. 2.1. Структурная схема алгоритма. Управляющая структура (поток управления)

Представлены также структурные формулы алгоритмов (рис. 2.1), где:

→ = –> = –	секвенция – операция последовательной связи операторов
||	параллель – операция параллельной связи операторов

Табл. 2.1. Псевдокоды частных алгоритмов. Стиль Алгол-68

Последовательный частный алгоритм A11	Последовательный частный алгоритм A12	Параллельный частный алгоритм A2
Латиница – английский язык		

alg A11
begin
v0 := (x1 + x2);
v1 := Sin(v0);
v2 := Cos(v0);
y := v1 * v2
end
	alg A12
	begin
	v0 := (x1 + x2);
	v2 := Cos(v0);
	v1 := Sin(v0);
	y := v1 * v2
	end
		alg A2
		begin
		v0 := (x1 + x2);
		par
		begin
		v1 := Sin(v0)**,**
		v2 := Cos(v0)
		end;
		y := v1 * v2
		end

Кириллица – русский язык		

алг A11
нач // начало
v0 := (x1 + x2);
v1 := Sin(v0);
v2 := Cos(v0);
y := v1 * v2
кон // конец
	алг A12
	нач
	v0 := (x1 + x2);
	v2 := Cos(v0);
	v1 := Sin(v0);
	y := v1 * v2
	кон
		алг A2
		нач
		v0 := (x1 + x2);
		пар
		нач
		v1 := Sin(v0)**,**
		v2 := Cos(v0)
		кон;
		y := v1 * v2
		кон

Для последовательных алгоритмов A11, A12 (Табл. 2.1) используются псевдокоды алгоритмов типа ***учебного (школьного) алгоритмического языка***. Он представляет собой простое подмножество русской версии Алгол-68 языка программирования Algol-68 [5], где предусматривались и средства описания параллелизма алгоритмов и программ. Это используется для псевдокода алгоритма A2. В данном случае это ***простое расширение*** учебного алгоритмического языка в стиле языка Algol-68 (Алгол-68).

Далее (Табл. 2.2) используются псевдокоды алгоритмов в стиле языка программирования Algol-60 (и Алгол-60). Для последовательных алгоритмов ничего не меняется, для параллельного алгоритма также получается простое расширение учебного алгоритмического языка.

Это алгоритмическая классика. Возможны другие виды псевдокодов.

Табл. 2.2. Псевдокоды частных алгоритмов. Стиль Алгол-60

Последовательные алгоритмы		Параллельный алгоритм
Частный алгоритм А11	Частный алгоритм А12	Частный алгоритм А2
Латиница – английский язык		

alg A11
begin
v0 := (x1 + x2);
v1 := Sin(v0);
v2 := Cos(v0);
y := v1 * v2
end
	alg A12
	begin
	v0 := (x1 + x2);
	v2 := Cos(v0);
	v1 := Sin(v0);
	y := v1 * v2
	end
		alg A2
		begin
		v0 := (x1 + x2);
		parbegin
		v1 := Sin(v0)**,**
		v2 := Cos(v0)
		parend;
		y := v1 * v2
		end

Кириллица – русский язык		

алг A11
нач // начало
v0 := (x1 + x2);
v1 := Sin(v0);
v2 := Cos(v0);
y := v1 * v2
кон // конец
	алг A12
	нач
	v0 := (x1 + x2);
	v2 := Cos(v0);
	v1 := Sin(v0);
	y := v1 * v2
	кон
		алг A2
		нач
		v0 := (x1 + x2);
		парнач
		v1 := Sin(v0)**,**
		v2 := Cos(v0)
		паркон;
		y := v1 * v2
		кон

В схемах и псевдокодах, а также в структурных формулах алгоритмов явно отражается структура потока управления. Структура потока данных явно не отражается, и косвенно она представлена далее (Табл. 2.3) системой элементарных функций пошагового вычисления заданной функции – в традиционной (слева) и в ярусно-параллельной (справа) форме записи. Во втором случае (ярусно-параллельная форма) выявляется наличие потенциального параллелизма задачи. При этом взаимосвязь составляющих функций отражается косвенно – их одноименными общими переменными.

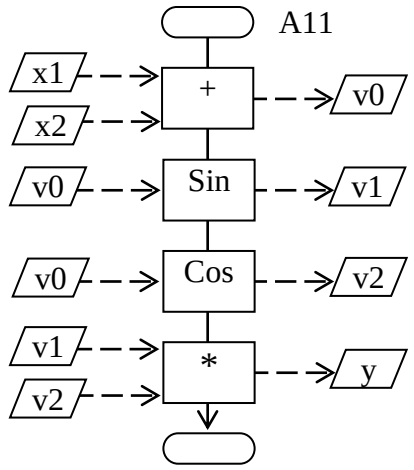
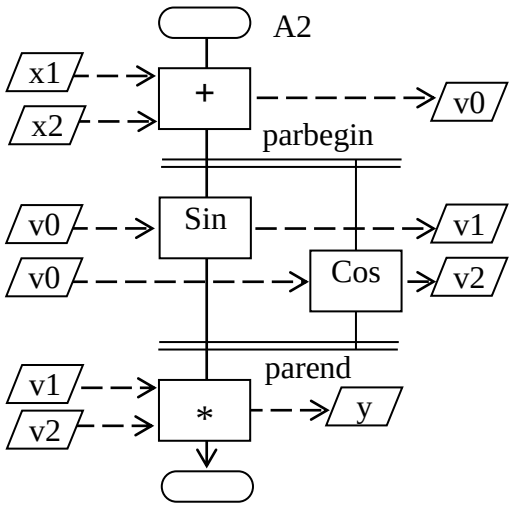
Табл. 2.3. Системы элементарных функций алгоритмической задачи

Традиционная форма записи	Ярусно-параллельная форма записи
⎧ **v0 = x1 + x2** ⎨ **v1 = Sin(v0)** ⎪ **v2 = Cos(v0)** ⎩ **y = v1 * v2**	**v0 = x1 + x2** **v1 = Sin(v0) v2 = Cos(v0)** **y = v1 * v2**

Для явного и наглядного визуального отображения связей переменных в системе функций и в целом структуры потока данных алгоритма

необходимо применение графических схем [1, 2], в частности – (Рис. 2.2, Рис. 2.3). В последнем случае (Рис. 2.3) наглядно отражается общий факт: одна структура потока данных и несколько (много, в общем случае) разных допустимых структур потока управления алгоритма.

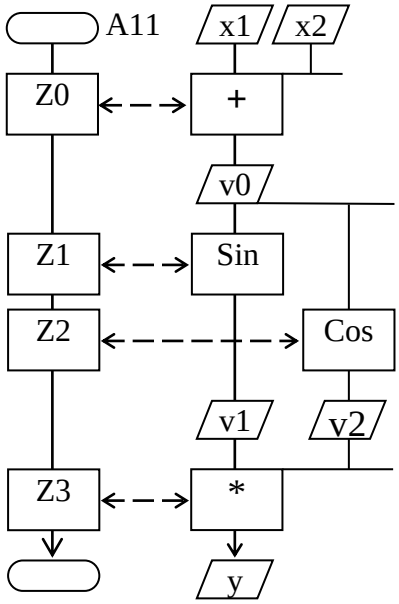
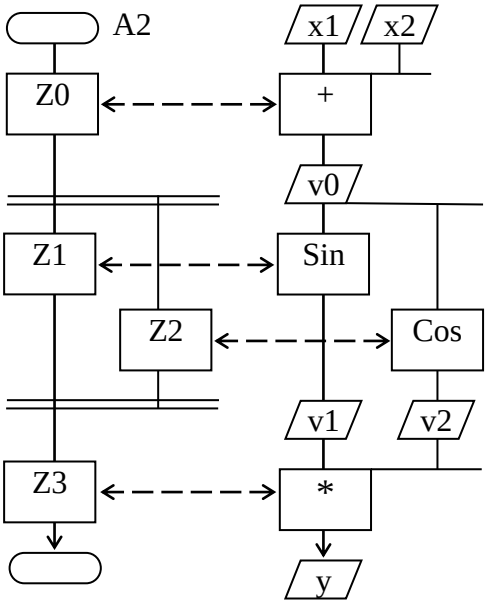
КУ: Канал (поток) управления + КД: Канал (поток) данных:
совмещение (пересечение: +) двух структур

Последовательные алгоритм	Параллельный алгоритм
	

Связная схема потока управления. Фрагментация схемы потока данных

Рис. 2.2. Комбинированная схема алгоритма. Пересечение потоков

КУ: Канал (поток) управления – КД: Канал (поток) данных:
разделение (–) двух структур

Последовательный алгоритм		Параллельный алгоритм	
КУ: Канал (поток) управления	КД: Канал (поток) данных	КУ: Канал (поток) управления	КД: Канал (поток) данных
			

Связные схемы потока управления и потока данных

Рис. 2.3. Комбинированная схема алгоритма. Разделение потоков

3 ОБЩАЯ СТРУКТУРА ТЕХНИЧЕСКИХ АЛГОРИТМОВ

Технические обобщения ведущего структурного фактора

Спецификой вычислительных (и вообще математических) алгоритмов является то, что математика является целью и средством выполнения алгоритмов. Для технических, технологических и, в частности, робототехнических алгоритмов возможный математический результат не является основной конечной целью выполнения алгоритмов. Здесь ситуация сложнее. Появляются некоторые существенные качественные отличия (Рис. 3.1):

- в составе исполнительной системы появляется ***дополнительная управляемая структура*** с целевыми (управляемыми) ***материальными объектами и потоками***, в частности (для автоматизации производства):
технологические каналы продвижения и обработки дискретных потоков продукции (деталей, узлов, изделий), инструмента, оснастки, тары и т.п.;
- необходимо ***обобщение*** понятия данных на материальные объекты и потоки, которые существуют в составе исполнительной системы и отражаются данными информационной структуры в текстах алгоритмов;
- при этом ***промежуточная информационная структура*** алгоритма ***может отсутствовать*** (в явно выраженной форме):

такие вопросы соотношения технических алгоритмов с наличием и без наличия (явно выраженного) информационного ядра и вычислительной математики мало изучены (и представляют интерес для анализа).

Кроме того, возможны алгоритмы так называемых реагирующих систем, взаимодействующих со средой:
в отличие от обычных алгоритмов обычных изолированных систем обрабатывающего типа (только на основе исходных данных) и т.п.

Такие аспекты пока не имеют в целом хорошо наработанного и систематического концептуального и теоретического обеспечения (за исключением отдельных частных видов систем и задач, направлений и т.п.). Существуют неясности по соотношению этих двух управляемых структур данных материально-информационного типа. Это обстоятельство представляет интерес для систематического анализа применительно к принятым областям приложений алгоритмов, что отражается далее – в кратком обзорном ознакомительно-постановочном порядке.

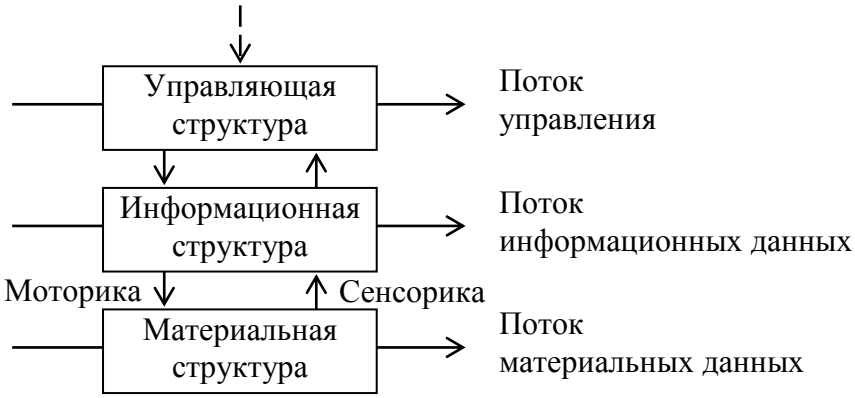


Рис. 3.1. Управление в технической исполнительной системе

Двойной состав управляющих и управляемых процессов

Существует еще один ключевой отличительный аспект.

Для конкретики далее за основу принимается алгоритмизация управления дискретными процессами в составе автоматизированных технологических систем:

(одноопераційных) гибких производственных модулей и (многоопераційных) гибких производственных систем и, в том числе, робототехнологических комплексов и систем.

Функционирование таких систем представляет собой иерархическую систему дискретных процессов двух двойственных типов (Рис. 3.2):

1) ***Движение материальных объектов*** и систем – в широком понимании терминов "***движение (процессия)***", "***процессы***", "***действия***" и т.п., включая двойственную систему составляющих процессов:

1.1) ***Комплексы действий*** (дискретные процессы функционирования) автоматизированного технологического ***оборудования*** и техоснастки.

1.2) Причинно связанное с этим ***движение материальных потоков*** (в технологических каналах) – процессы продвижения, хранения и обработки, применения дискретных потоков продукции и прочих объектов – деталей, узлов, инструмента, тары и т.д.:
поштучно, комплектами, партиями и т.п.

2) ***Управление движением*** материальных систем, включая соответствующую двойственную систему составляющих процессов:

2.1) Прямое, но вспомогательное (инструментальное) ***управление*** (автоматизированным) ***оборудованием*** и техоснасткой:
для их (причинного) воздействия на конечные материальные потоки.

2.2) Основное (целевое), но косвенное ***управление материальными потоками*** продукции:
посредством воздействий на них оборудования, инструмента, оснастки.

Рис. 3.2. Двойственные типы дискретных процессов автоматизированных систем

Такой ***двойной состав*** управляющих и управляемых ***процессов*** прямо или косвенно связан с ***двойным составом*** управляющей и управляемой ***структуры алгоритмов*** управления процессами. Однако этот аспект является мало разработанным в алгоритмике технических процессов.

Оборудование и, в частности, промышленные роботы могут быть стационарными и мобильными, в частности, транспортными. При этом возможно групповое управление движением множества единиц такого оборудования, включая управление движением их потоков.

4 ПРОСТОЙ ПРИМЕР ТЕХНИЧЕСКОГО АЛГОРИТМА

Кратко рассматривается пример простой технической задачи (опорного концептуального типа). Она аналогична по структуре потока управления алгоритмов в примере простой вычислительной задачи (также опорного типа), рассмотренной ранее и подробно анализируемого в статьях [1, 2].

Робот автоматизированного складского комплекса

Рассматривается автоматизированный складской комплекс (АСК, Рис. 4.1) с программируемым роботом-штабелером для обслуживания позиций ячеистого стеллажа – в автоматизации складского хозяйства на производстве, в магазине, в системе логистики товаров и т.п. Это особая (стеллажная) разновидность транспортного робота (с транспортными перемещениями грузов между ячейками стеллажа). В упрощенной постановке задачи (Рис. 4.1) надо переместить тару с грузом из некоторой исходной ячейки стеллажа в некоторую заданную ячейку:
это модельная задача второго типа – внутрискладское целевое (но косвенное) управление потоками продукции технологических систем.

Явное прямое (инструментальное) управление оборудованием на данном этапе алгоритмом не отражается (но позднее кратко оговаривается).

Более подробно краткая постановка задачи изложена в статье [6].

Используется программная модель в учебно-игровой среде Scratch.

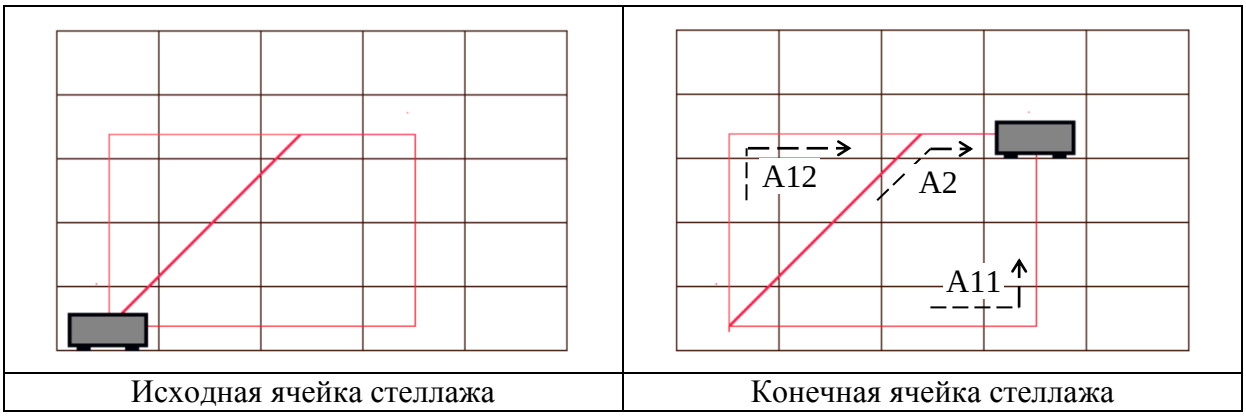


Рис. 4.1. АСК: Автоматизированный складской комплекс. Видеосхема стеллажа АСК

Краткий анализ условий задачи

Возможны (Рис. 4.1) 3 траектории движения груза между ячейками с последовательным и параллельным выполнением транспортных ходов.

В ручном режиме управления допустимо только последовательное выполнение ходов по горизонтали или вертикали – частные варианты A11 и A12 общего алгоритма A: параллелизм отсутствует (вырожденная единичная степень параллелизма: p = 1).

В автоматическом режиме управления возможны эти же варианты и, дополнительно, параллельное выполнение ходов – частный вариант A2 общего алгоритма A (2-я степень реального параллелизма: p = 2). Таким образом, выявляется факт наличия ***потенциального параллелизма*** задачи.

Краткое алгоритмическое описание задачи

Далее (Табл. 4.1) представлены условные литерные обозначения (УЛО) алгоритмов и команд. Возможно уточнение алгоритмов и команд с указанием параметров адресации перемещений – по вертикальным рядам и по ярусам ячеек стеллажа (не приводятся для упрощения записи).

Табл. 4.1. УЛО: Условные литерные обозначения алгоритмов и команд

ООА: Общее обозначение алгоритма
A = A(A11, A12, A2) – общий (сводный) алгоритм задачи
A11, A12, A2 частные алгоритмы (варианты исполнения общего алгоритма)
СКА: Система команд общего алгоритма
Z0: ЗТ: Загрузка (выборка) тары: вызов подпрограммы загрузки транспортной позиции робота (с выборкой тары с грузом или без груза из текущей ячейки)
Z1: ГХ: Горизонтальный ход: перемещение до заданного вертикального ряда стеллажа
Z2: ВХ: Вертикальный ход: перемещение до заданного яруса стеллажа
Z4: РТ: Разгрузка (установка) тары: вызов подпрограммы разгрузки транспортной позиции робота (с установкой тары с грузом или без груза в текущую ячейку)

Далее представлены структурные формулы алгоритмов (Табл. 4.2), структурные схемы (блок-схемы) и псевдокоды алгоритмов (Табл. 4.3). Приводятся псевдокоды алгоритмов типа учебного (школьного) алгоритмического языка и его расширения. В блок-схемах и псевдокодах алгоритмов для плотности их записи в общей таблице использованы аббревиатуры

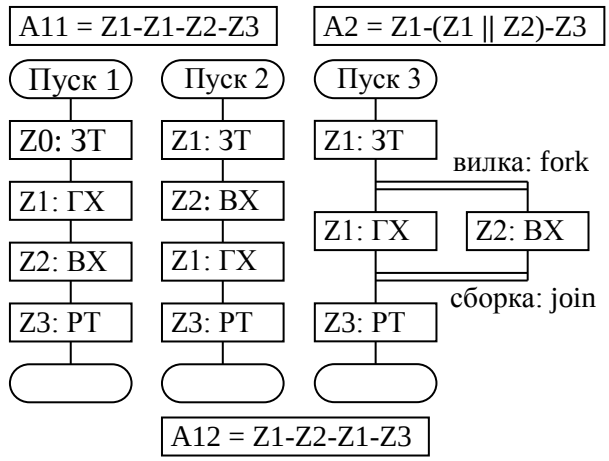
имен команд. Но практически лучше подставлять полные имена команд типа: загрузка тары (загрузить тару), горизонтальный ход (двигаться горизонтально) и т.п. Аббревиатуры можно подставлять в структурные формулы как мнемонику, например:

A11 = ЗТ–ГХ–ВХ–РТ	A12 = ЗТ–ВХ–ГХ–РТ	A2 = ЗТ–(ГХ || ВХ)–РТ

Табл. 4.2. Частные алгоритмы A11, A12, A2 общего алгоритма A

СФА: Структурная формула алгоритма – частные варианты
A11 = (Z0→(Z1→Z2)→Z3) = Z0–Z1–Z2–Z3 = $Z_0Z_1Z_2Z_3$ прямой порядок выполнения команд Z1 и Z2
A12 = (Z1→(Z2→Z1)→Z3) = Z1–Z1–Z2–Z3 = $Z_1Z_2Z_1Z_3$ обратный порядок выполнения команд Z1 и Z2
A2 = (Z1→(Z2 || Z1)→Z3) = Z1–(Z2 || Z1)–Z3 = $Z_1(Z_2 \| Z_1)Z_3$ параллельное выполнение команд Z1 и Z2

Табл. 4.3. Схемы и псевдокоды алгоритмов

ССА: Структурная схема алгоритма	ПКА: Псевдокод алгоритма К: Кириллица: Ru	
БСА: Блок-схема алгоритма (ССА = БСА)	АлгПТ: Алгол-подобный текст	
ВИ: Вертикальное исполнение	Стиль: Алгол-68	Алгол-60'
	алг A11: нач ЗТ; ГХ; ВХ; РТ кон	алг A2: нач ЗТ; пар нач ГХ, ВХ кон; РТ кон Соответствие обозначений (Algol-68, Algol-60'): нач кон пар парнач паркон begin end par parbegin parend	алг A2: нач ЗТ; парнач ГХ, ВХ паркон; РТ кон

Далее (Рис. 4.2) представлена временная диаграмма и расчет длительности исполнения (mai) алгоритма (Ai) для некоторых принятых данных по длительности исполнения (mzi) команд (Zi). Автоматизировано построение только линейных диаграмм, но это очень трудоемкая работа (особенно для сложных алгоритмов). Сетевые диаграммы строятся путем ручной доработки линейных диаграмм, но это менее трудоемкая работа, и это оказалось хорошим учебным средством структурных построений.

Узел сборки контролирует окончание всех (двух в данном случае) параллельных процессов, одновременно начатых после узла вилки.

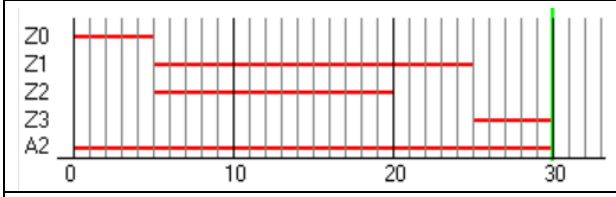
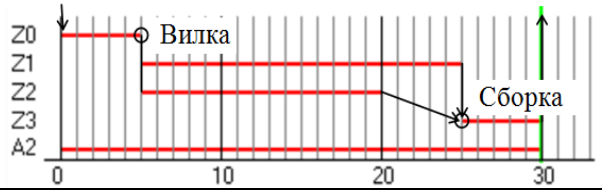
ДИА: Диаграмма исполнения алгоритма	
ЛД: Линейная диаграмма	СД: Сетевая диаграмма
	
ФРД: Формула расчета длительности (исполнения алгоритма)	
ma2 = mz0 + max(mz1, mz2) + mz3 = 5 + max(20, 15) + 5 = 5 + 20 +5 = 30	

Рис. 4.2. Временная диаграмма параллельного алгоритма A2

5 УПРАВЛЕНИЕ ОБОРУДОВАНИЕМ

Алгоритмическая система управления

АСК представляет собой автоматизированную систему обслуживания стеллажа и включает в себя следующие две основные компоненты:

1) Механический объект управления, представленный роботом-штабелером (с его механизмами) для обслуживания складской стеллажа.

2) Управляющая система в некоторой ее реализации:

- релейно-контактная или бесконтактная схема логического управления;
- возможно применение микропроцессорной техники, промышленных логических контроллеров или управляющей ЭВМ общего или специального назначения.

Не рассматриваются задачи складского учета, регулирования заделов производства и хода производства (по уровням заделов) и т.п.

Система управления выполняет два вида управления:

- непосредственное (прямое) управление механизмами складского робота-штабелера;
- сопутствующее (косвенное) управление потоком материальных данных: в данном случае – это объекты типа тары с грузом или пустой тары.

Нормальное управление потоком материальных данных выполняется, если эти материальные данные будут находиться в соответствующих исходных позициях разных циклов. При этом:

- если необходимые исходные материальные данные отсутствуют, то механизмы будут отрабатывать так называемые холостые циклы:
это может использоваться, например, в наладочном режиме работы АСК;
- если данные будут располагаться неверно, то возможна неправильная и аварийная работа АСК (попытка загрузить уже занятую ячейку и т.п.).

Для повышения адекватности управления могут использоваться датчики (необходимого или недопустимого) наличия или отсутствия материальных данных – тары, деталей в таре и т.п. Однако этот аспект не учиты-

вается в излагаемой алгоритмической задаче – в предположении идеального обеспечения необходимых условий.

Циклы выполнения команд алгоритма

Для вычислительных алгоритмов вычислительные команды могут иметь обозначение типа:

Zi :: yi = Fi(xi),

где Fi – общее обозначение некоторой численной функции;

xi = (xi1, xi2, …, xim), yi = (yi1, yi2, …, yin) – составные в общем случае переменные (аргументы и результаты) многоместной в общем случае функции.

Отдельные команды излагаемого алгоритма управления АСК выполняют некоторые технические функции (действия) манипулирования материальными данными (объектами) в пространстве и во времени:

- непосредственно это действия (с манипуляциями) разных целевых узлов механизмов робота-штабелера двух типов:

рабочие действия с тарой и холостые хода механизмов – без наличия тары;

- косвенно рабочие хода механизмов связны с сопутствующими пространственными перемещениями (манипуляциями) тары с грузом.

Эти технические функции (действия) в принципе можно пытаться описать некоторыми математическими функциям (перемещения улов и материальных данных из одной позиции другую, с определенной скоростью и т.п.). Однако адекватное описание таких действий математическими функциями в пространственно-временной интерпретации для обычного складского робота представляет собой определенную проблему. Реально эти данные косвенно присутствую в схемах и расчетах механизмов робота-штабелера. автоматизированного склада и т.п.

Но на данном этапе анализа для алгоритма управления этого и не требуется:

общий алгоритм может быть построен непосредственно на основе указанных выше описаний команд и общих технических представлений (в некотором минимальном контексте).

Можно уточнить описания транспортных команд, например, типа:

Z1 = Z1(x1, z1)	Z2 = Z2(x2, z2)

где xi, zi – конечные адреса перемещения (номера рядов и ярусов):

по горизонтали (до вертикального ряда xi) и по вертикали (до яруса zi).

Соответственно этому уточняются записи частных алгоритмов.

Позиционное управление техническим объектом

Для детализации алгоритма (если это требуется) необходимо пользовать дополнительные сведения по составу действий выполнения указанных команд и порядку их выполнения. При этом команды Z0..Z3 алгоритма A верхнего уровня конкретно интерпретируются как команды вызова более детальных алгоритмов (подалгоритмов) их исполнения.

Далее (Рис. 5.1) представлены (в общем виде) позиционные диаграммы (циклограммы) выполнения команд алгоритма А верхнего уровня управления (исполнения подалгоритмов 2-го уровня). В данном случае они необходимы для конкретизации и понимания общего принципа действия механизмов штабелера (содержательной интерпретации команд Z0..Z3):
для обеспечения некоторого запаса конкретных представлений, для последующего программирования и т.п.

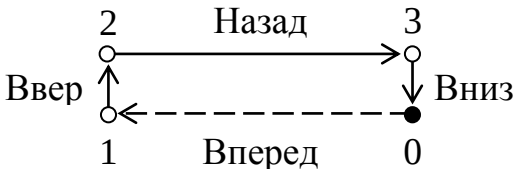
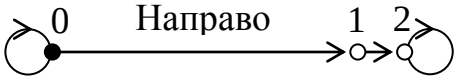
Позиционная диаграмма	Описание составляющих ходов
Команда Z0: Взять: Get – выборка груза (тары с грузом) из текущей ячейки стеллажа	
	0-1: Вперед – вилка захвата тары подает вперед (по оси y) в зону ячейки стеллажа: холостой ход вилки захвата (без тары). 1-2: Вверх – вилка захвата тары подается вверх: тара приподнимается (в ячейке стеллажа). 2-3: Назад – вилка захвата тары подается назад: тара выводится из зоны ячейки стеллажа. 3-0: Вниз – вилка захвата тары подается вниз: тара устанавливается в транспортную позицию (транспортного механизма) штабелера.
Команда Z1 = Z1(x): Гор_ход(x): Hor_goto(x) – движение направо или налево: выбор направления движения выполняется автоматический. Приводится пример выполнения хода направо (ход налево выполняется аналогично)	
	0-0: Разжим – расфиксация механизма движения по горизонтали (по оси x). 0-1: Ход – быстрый ход направо к заданному вертикальному ряду стеллажа. 1-2: Подвод – медленный подвод в рабочую координатную позицию: на "ползучей" скорости с точным остановом. 2-2: Зажим – фиксация механизма движения по горизонтали.
Команда Z2: Вер_ход(z): Ver_goto(z) – движение вверх или вниз: выбор направления автоматический – выполняется аналогично команде Z1.	
Команда Z3: Уставить: Set – установка груза (тары с грузом) в заданную ячейку	
	0-3: Вверх – вилка захвата тары подается вверх: тара приподнимается над транспортной позицией (транспортного механизма) штабелера. 3-2: Вперед – вилка подается вперед (по оси y): тара перемещается в зону ячейки стеллажа. 2-3: Вниз – вилка захвата тары подается вниз: тара опускается на дно ячейки стеллажа. 1-0: Назад – вилка захват тары подается назад (по оси y): холостой ход вилки (без тары).

Рис. 5.1. Позиционные диаграммы (циклограммы) выполнения команд алгоритма

В целом данные позиционные диаграммы выполнения команд Z0..Z3 (Рис. 5.1) представляют алгоритм А управления циклами перемещений тары (с грузом или без груза) в системе ***позиционного управления*** с четко

определенными исходными и конечными позициями ходов управляемых механизмов (при необходимости – с параметрами адресации позиций).

Возможно приведение такого описания команд и алгоритма в целом к некоторым дискретным моделям типа конечных автоматов или сетей Петри. Однако принимаются к сведению следующие замечания:

- во-первых, это должны быть не обычные конечные автоматы или сети Петри, а некоторые их разновидности с учетом длительности выполнения составляющих действий;
- во-вторых, для данной простой алгоритмической задачи подобное модельное описание команд – это будет излишнее теоретизирование: затемняется логика структурных построений алгоритма позиционного управления, хорошо понятного и достаточного для его программной реализации без таких дополнительных моделей.

6 ВАРИАТИВНОСТЬ РЕАЛИЗАЦИИ ПАРАЛЛЕЛИЗМА

Объедение частных алгоритмов в общий алгоритм

Частные алгоритмы решения вычислительных и технических задач в принципе могут быть объединены в общий вариативный (многовариантный в общем случае) алгоритм. Далее кратко представлена общая идея объединения трех частных алгоритмов A11, A12, A2 в один общий составной алгоритм:

A = A(A11, A12, A2).

Принимается структурная формула составного общего алгоритма, объединяющего составляющие частные алгоритмы:

$A = S(A11, A12, A2) = p11A11 \vee p12A2 \vee p2A2 = p_{11}A_{11} \vee p_{12}A_2 \vee p_2A_2,$

где S – условное обозначение структурной операции (альтернативного) объединения частных алгоритмов A11, A12, A2 в общий алгоритм A;

p11, p12, p2 – альтернативные (взаимно исключающие) логические условия (предикаты) выбора вариантов исполнения алгоритма;

$\vee$ – операции объединения частных алгоритмов (с условиями выбора) в общий алгоритм по функции дизъюнкции (логической функции Или, Or).

Это достаточно удовлетворительная модель полной алгоритмизации данной простой задачи. Но она доступна для малого числа частных вариантов исполнения алгоритма (в данном случае всего 3 варианта). С увеличением степени потенциального параллелизма для отдельных участков алгоритмов быстро (лавинообразно) нарастает вариативность алгоритмов по порядку их выполнения сходной и разной степени параллелизма (и, кроме того, с перемножением вариативностей, если есть несколько участков потенциального параллелизма). При этом:

- с одной стороны, повышается гибкость распараллеливания алгоритмов с учетом различных исходных и текущих условий исполнения алгоритма;
- с другой стороны, затрудняется полный перебор вариантов исполнения.

Тем не менее, такая принципиальная возможность может быть реализована некоторыми обобщенными способами, что полезно для исследования алгоритмических задач.

Для вычислений в используемой среде программирования может быть система планирования вычислений, которая может сама выбирать допустимое оптимальное продолжение вычислений по мере выполнения предыдущих этапов вычислений (с учетом наличных ресурсов и т.п.):
в пределах заданной каким-то образом схемы распараллеливания счета.

Обобщенная вариативная запись параллелизма

Можно использовать условную запись следующего типа, как обобщенную запись полной вариативной реализации потенциального параллелизма:

$$A = S(A11, A12, A2) = (Z1 \rightarrow (Z2 \underline{\sqcup} Z1) \rightarrow Z3) = Z1 \rightarrow (Z2 \underline{\|} Z1) \rightarrow Z3,$$

где символ $\underline{\sqcup} = \underline{\|}$ (подчеркнутая параллель) означает возможность произвольного порядка выполнения действий:
возможность любого параллельного порядка (полного или частичный параллелизм) или последовательного порядка (в любой последовательности).

Для вычислительных задач – это основа для автоматизации планирования комплексов действий по текущей ситуации – наличие ресурсов и т.п. (для программных планировщиков).

Эквивалентность вариантов алгоритмов

Все частные алгоритмы A11, A12, A2 функционально взаимно эквивалентны друг другу и общему (вариативному) алгоритму A:

$$A11 = A12 = A2 = A.$$

Это верно, поскольку все они выполняют одну и ту же заданную функцию (в частности вычисляют одну и туже функцию – для вычислительных алгоритмов).

Однако они не эквиваленты друг другу по другим аспектам:

- по требуемым ресурсам – один или два исполнителя (на среднем участке алгоритма);
- по времени исполнения (время, как ресурс) – последовательные варианты выполняются дольше, чем параллельный вариант;
- по организации вычислений – параллельный алгоритм A2 организационно сложнее в исполнении, чем последовательные алгоритмы A11 и A12 и требует более сложных средств программной реализации.

7 ЯВНЫЙ УЧЕТ ИСПОЛНИТЕЛЕЙ АЛГОРИТМА

Исполнители команд и алгоритмов

Для последовательных и параллельных алгоритмов в общем случае все команды могут выполняться разными исполнителями с формированием общего множественного (составного или группового) исполнителя алгоритма в целом. При этом для параллельных алгоритмов для одновременно исполняемых команд необходимы разные исполнители. Для неодновременного исполнения команд могут назначаться одни и те же исполнители. Для последовательных алгоритмов может быть один исполнитель всех команд и всего алгоритма в целом.

Распределение исполнителей

В программной реализации параллельных алгоритмов выбор и распределение исполнителей может выполняться следующими способами:

1) Указанием конкретных исполнителей команд или их разных группировок в рабочем тексте программы и ее алгоритма.

2) Без указания конкретных исполнителей в рабочем тексте программы и ее алгоритма – с их назначением дополнительными средствами:

- до начала (однократного или многократного) исполнения программы: посредством запуска специального начального блока инструкций;
- в начале исполнения программы – в специальном начальном разделе;
- по ходу исполнения программы – автоматически специальным планировщиком задач (в зависимости от разных текущих условий) и при этом: в высокоуровневой записи текстов алгоритмов и программ исполнители не указываются.

В приведенных примерах вычислительного и технического алгоритма в структурных формулах и схемах и в псевдокодах исполнители явно не указаны. При этом в текстовом описании команд технического алгоритма присутствуют упоминания исполнительных механизмов робота-штабелера.

Общий принцип указания исполнителей алгоритмов и команд

В общем случае для параллельного алгоритма A3 возможно уточнение записи структурной формулы алгоритма следующего типа (в ориентации на стиль записи объектно-ориентированного программирования):

E.A3 = E0.Z0→(E1.Z1 || E2.Z2)→E3.Z3,

A3' = Z0'→((Z1' || Z2')→Z3',

где Ei – объект-исполнитель (executor) команды Zi, i = 0..3:
это могут быть разные процессоры в программной реализации алгоритма или разные вычислители в ручных вычислениях, разные механизмы робота технического алгоритма;

E = (E0, E1, E2, E3) – множественный объект-исполнитель алгоритма;

A3' = E.A3	Z0' = E0.Z0	Z1' = E1.Z1	Z2' = E2.Z2	Z3' = E3.Z3

Для вычислительного параллельного алгоритма возможен вариант с минимальным числом исполнителей (процессоров) – в данном случае для двух исполнителей E0' = (E1, E2), например:

E'.A3 = E1.Z0→(E1.Z1 || E2.Z2) →E1.Z3.

Для последовательных алгоритмов с одним исполнителем (E):

A11' = E.A11 = (E.Z0→E.Z1→E.Z2→E.Z3) = E.(Z0→Z1→Z2→Z3),

A11 = (Z0→Z1→Z2→Z3) – указывать исполнителя алгоритма не обязательно (он, обычно, заранее известен).

ЗАКЛЮЧЕНИЕ

Отражена ***ключевая первичная объектная ориентированность*** записи алгоритмов – с привязкой команд и алгоритмов к объектам-исполнителям. Программные классы (специальные вспомогательные объекты) с функциями порождения, тиражирования, ликвидации и обслуживания рабочих объектов (экземпляров классов), наследования классов и т.п. на данном уровне не используются. Но этого уже достаточно для алгоритмического анализа программ в разных модельных средах, например:
в учебном конструкторе мультимедийных игр Scratch с наличием параллелизма работы визуальных объектов (спрайтов).

ЛИТЕРАТУРА

1. Житников А.П. Потоки управления и потоки данных параллельных (и последовательных) алгоритмов. // Сб. матер. междунар. научно-практ. форума: XIV Междунар. научно-практ. конф. «Информационные и коммуникационные технологии в образовании». – Борисоглебск: БГПИ, 2013. – С. 45-61.
http://paralg.ucoz.com/g4110/v5-g4110-s105-ptk_upr_i_ptk_dan.pdf
2. Житников А.П. Структурный алгоритмический анализ простой вычислительной задачи. // Там же. – С. 62-77.
http://paralg.ucoz.com/g4110/v5-g4110-s106-analis_vych_zadachi.pdf
3. Воеводин В.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ, 2006. – 112 с.
4. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб: БХВ-Петербург, 2002. – 608 с.
5. Пересмотренное сообщение об Алголе 68. // Под ред. А.П. Ершова. – М.: Мир, 1979. – 533 с.
6. Житников А.П. Логико-временная интерпретация параллельных алгоритмов робототехнических систем. // Пропедевтика формирования инженерной культуры учащихся в условиях модернизации российского образования [Электронный ресурс]: сборник статей. – М. : БИНОМ. Лаборатория знаний, 2015. – С. 63-83.
http://paralg.ucoz.com/g4140/v5-g4144-s102-log_vrem_interpre.pdf