g4144_s102	06.03.15

Статья. Библиографические данные:

Житников А.П.

Логико-временная интерпретация параллельных алгоритмов робототехнических систем.

// Пропедевтика формирования инженерной культуры учащихся в условиях модернизации российского образования [Электронный ресурс]: сборник статей. – Эл. изд. – Электрон. текстовые дан. (1 файл pdf : 350 с.). – М. : БИНОМ. Лаборатория знаний, 2015.
– Систем. требования: Adobe Reader XI ; экран 10".
– С. 63-83.

ЛОГИКО-ВРЕМЕННАЯ ИНТЕРПРЕТАЦИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

ЛОГИКО-ВРЕМЕННАЯ ИНТЕРПРЕТАЦИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

Кратко рассматриваются первичные принципы интерпретации параллельной алгоритмики как логики (комплексов) действий во времени в приложениях к задачам алгоритмизации практической и образовательной робототехники.

ВВЕДЕНИЕ

Статья [1] начинает серию публикаций автора в области параллельной (и последовательной) алгоритмики ***массовой информатики и робототехники*** (РТ). Эта и другие статьи доступны на сайте http://paralg.ucoz.com.

Ключевой задачей определяется поэтапное решение проблем корректного и общедоступного ***понимания и объяснения логики механизмов управления*** параллельными (и последовательными) дискретными процессами алгоритмических систем во времени. Задача достаточно ***сложная***. Необходимо учитывать сложность и разнообразие координации и взаимодействия параллельных процессов и тот факт, что параллельная алгоритмика – этот качественно более сложное и многообразное научно-техническое направление по сравнению с традиционной последовательной алгоритмикой. Тем не менее, это ***достаточно реальная задача***, если учитывать, что все люди с детства успешно интуитивно осваивают объективную реальность одновременного (параллельного во времени) существования, действия и взаимодействия многих комплексов объектов.

В статье излагаются простые начальные, но практически полезные постановочные подходы решения задачи. Это опыт длительной, хотя и фрагментарной (отдельными вставками) практики преподавания разных аспектов параллельной алгоритмики. Основная аудитория – "младшие" студенты разных технических специальностей, и был (удачный) опыт работы с учениками ПТУ.

1 ИСХОДНЫЕ УСТАНОВКИ

Первичная область робототехнической алгоритмизации

По прежнему опыту автора статьи (еще перестроечных и доперестроечных времен) за основу принимается ***промышленная робототехника***. Это наиболее широкая и успешная (за рубежом) область массовой роботизации в прошлом и на перспективу (до 25-го года [2] и далее). И она должна, так или иначе, отражаться в общей образовательной РТ. При этом на начальных этапах обучения можно использовать простые и общепонятные демонстрационные модели технологических ***робототехнических систем*** (РТС) – для понимания их работы не требуется быть конструкторами, технологами и наладчиками автоматизированного производства (например, Рис. 2.2, Рис. 2.3, Рис. 2.4).

Излагаемые подходы в принципе могут распространять и обобщаться на другие виды РТС, включая применение учебно-игровых робототехнических лего-комплектов и т.п. Однако автор статьи пока такого опыта не имеет.

Исходные объекты параллельной (и последовательной) алгоритмизации

За исходную основу принимаются технологические РТС на уровне выполнения технологических операций – ***робототехнологические комплексы*** (РТК). Технологическая операция – это часть технологического процесса обработки деталей, сборки узлов и т.п., выполняемая на одном рабочем месте.

Имеются в виду РТК следующих типов:

1) ***Обрабатывающие*** РТК: станок и ***обслуживающие его роботы*** – на загрузке-разгрузке деталей, для технического контроля и т.п., причем:

- могут быть металлорежущие станки, штамповочные прессы и т.п.;
- возможны простые режимы поочередного следования смежных итераций в циклах обработки потоков деталей, и режимы конвейерной обработки:

с перекрытием во времени смежных итераций циклов обработки [3];

они могут демонстрироваться и доступно поясняться на первых же обзорно-ознакомительных занятиях с применением модельных программ (Рис. 2.4).

б) РТК с двумя, тремя и более ***технологическими роботами***:

- РТК с окрасочными роботами для окраски сложных изделий, например, кузовов автомобилей;
- РТК со сварочными роботами – для сварки кузовов автомобилей и т.п.;
- сборочные РТК со сборочными роботами и т.д.

Для таких РТК в интернете можно подобрать иллюстрации и деморолики.

На этом уровне могут рассматриваться ***транспортные и складские роботы*** производственного назначения, для систем логистики, магазинов и т.п.

Типовые исходные базовые циклы

За исходную основу принимается следующий обобщенный последовательный алгоритм (итерации) технологического цикла РТК (Табл. 1.1):

Табл. 1.1. Типовой цикл: технологические переходы Ti (technological transitions)

СФА: Структурная формула алгоритма / Инфиксная форма	
ЯИнФ: Явная инфиксная форма	НИнФ: Неявная инфиксная форма
A = (Tb→Tm→Te) = Tb–Tm–Te = $T_bT_mT_e$ // i = b, m, e	
Технологические переходы – основной Tm и вспомогательные Tb, Te	
Tb: Подготовка работы – загрузка детали в рабочую позицию и т.п. // b: begin	
Tm: Выполнение работы – выполнение обработки, сборки и т.п. // m: main	
Te: Завершение работы – разгрузка детали из рабочей позиции и т.п. // e: end	
Структурная операция	
→ = –> = –	Секвенция: последовательная связь операторов алгоритмов

СФА задает последовательное во времени выполнение трех технологических переходов Tb, Tm, Te. Далее рассматриваются РТК с распараллеливанием основного перехода Tm на два, три и более составляющих перехода (Табл. 1.2): типовая первичная структура алгоритма с одним параллельным участком.

Табл. 1.2. Типовой цикл. Распараллеливание основного перехода

СФА: Структурная формула алгоритма	
A = (Tb→(Tm1 || Tm2) →Te) = Tb–(Tm1 || Tm2)–Te = $T_b(T_{m1} \parallel T_{m2})T_e$	
A = (Tb→(Tm1 || Tm2 || Tm3) →Te)	
A = (Tb→(Tm1 || Tm2 || … || Tmn) →Te)	
Вторая структурная операция	
||	Параллель: параллельная связь алгоритмов (операторов алгоритмов)

Данная СФА задает параллельное во времени (одновременное) выполнение составляющих переходов Tm1, Tm2, … Tmn, то есть их выполнение с совмещением во времени разной (реальной) степени параллелизма p = 2, 3, …, n.

Общие символы алгоритмов типа Ai, Ti могут заменяться символами элементарных команд типа Zi – команды вызова подпрограмм (подалгоритмов).

2 ИСХОДНЫЕ ПРИМЕРЫ

Робот автоматизированного складского комплекса

Рассматривается простой автоматизированный складской комплекс (АСК, Рис. 2.1) для автоматизации (машиностроительного, приборостроительного) производства и т.п., в системах логистики потоков товаров и для магазинов.

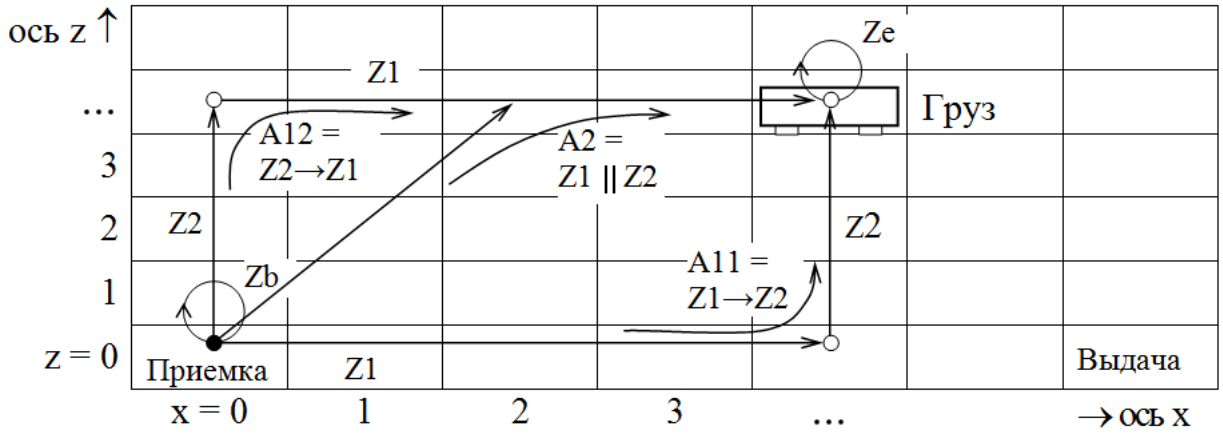


Рис. 2.1. АСК: Автоматизированный складской комплекс. Схема стеллажа

АСК имеет ячеистый вертикальный стеллаж для размещения складских грузов (в таре или на поддонах). Надо представить алгоритм управления типовым циклом работы складского робота-штабелера, обслуживающего стеллаж: перемещение груза (тары с грузом) из одной ячейки склада в другую, например, из позиции приемки в некоторую рабочую позицию стеллажа для хранения груза, или из некоторой рабочей позиции в позицию выдачи.

На фоне схемы стеллажа приводится позиционная схема для типового цикла перемещения груза из одной ячейки стеллажа в другую.

Далее (Табл. 2.1) приводится система команд алгоритма и его альтернативные частные варианты (с возможностью их объединения).

В ручном режиме управления допустимо только последовательное выполнение ходов по горизонтали и вертикали – частные варианты A11 и A12 общего алгоритма: параллелизм отсутствует (вырожденная единичная степень параллелизма p = 1). В автоматическом режиме управления возможно параллельное выполнение ходов – частный вариант A2 общего алгоритма (2-я степень p = 2 реального параллелизма). Возможно уточнение алгоритма с указанием параметров адресации перемещений (не приводятся для упрощения записи).

Табл. 2.1. Команды и частные алгоритмы A11, A12, A2 общего алгоритма A

СКА: Система команд общего алгоритма	
Zb: Загрузка – команда вызова подпрограммы загрузки транспортной позиции робота (с выборкой тары с грузом из текущей ячейки)	
Z1: Горизонтальный ход – перемещение до заданной вертикали стеллажа	
Z2: Вертикальный ход – перемещение до заданного яруса стеллажа	
Ze: Разгрузка – команда вызова подпрограммы разгрузки транспортной позиции робота (с установкой тары грузом в текущую ячейку	
СФА: Структурная формула алгоритма – частные варианты	
A11 = (Zb→(Z1→Z2)→Ze)	// прямой порядок выполнения команд Z1 и Z2
A12 = (Zb→(Z2→Z1)→Ze)	// обратный порядок выполнения ком. Z1 и Z2
A2 = (Zb→(Z2 || Z1)→Ze)	// параллельное выполнение команд Z1 и Z2

Сводный общий алгоритм

Общий алгоритм A можно сформировать альтернативным объединением частных алгоритмов A11, A12, A2 – в данном случае всего трех. В общем случае с большим числом альтернативных вариантов это затруднительно. Но можно использовать условную запись следующего типа:

A = (Zb→(Z2 ⫢ Z1)→Ze) = Zb→(Z2 ⫴ Z1)→Ze,

где символ ⫢ = ⫴ (подчеркнутая параллель) означает возможность произвольного порядка выполнения действий:

возможность любого параллельного порядка (полный или частичный параллелизм) или последовательного порядка (в любой последовательности).

В общем случае – это основа для автоматизации планирования комплексов действий по текущей ситуации (для программных планировщиков).

Агрегатный комплекс механической обработки

Приводится пример с применение программной модели (Рис. 2.2, Рис. 2.3) агрегатного технологического комплекса (АТК) для многосторонней обработки корпусных деталей (сверления отверстий, в данном случае).

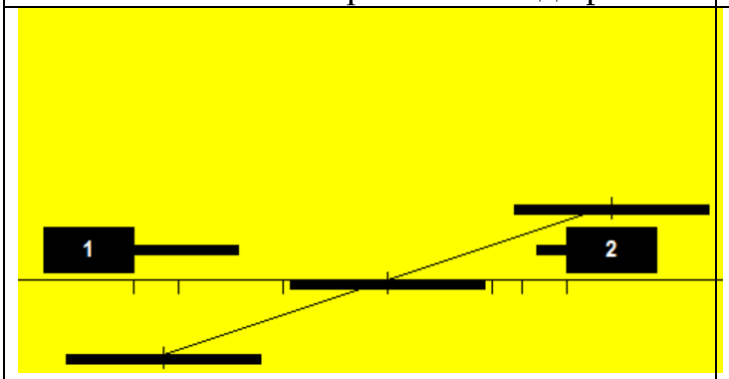
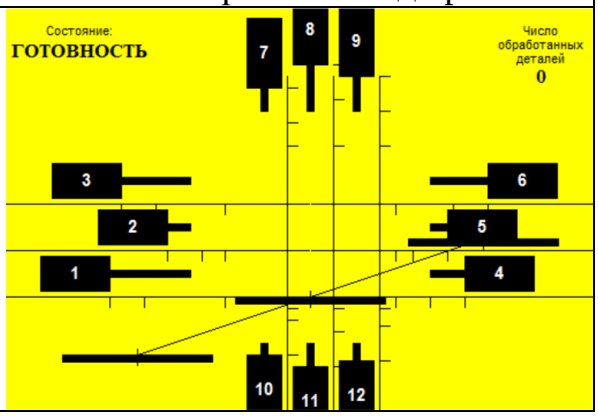
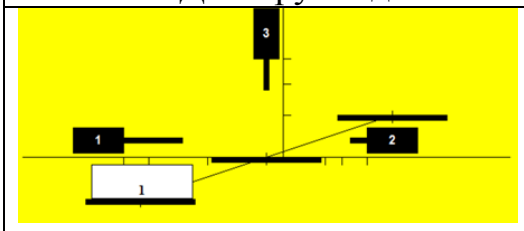
АТК2: Степень параллелизма до p = 2	АТК12: Параллелизм до p = 12
	

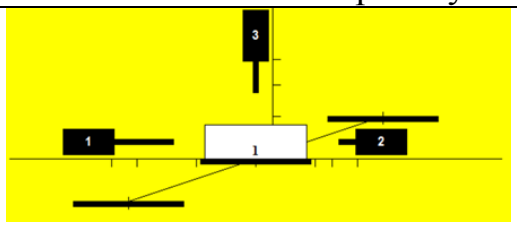
Рис. 2.2. Примеры конфигураций комплекса: АТК2, АТК12

Исходный общий алгоритм операционного цикла:

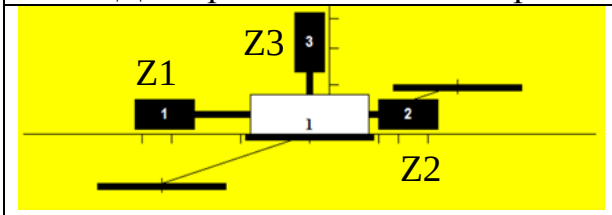
A = (Tb→Tm→Te) = Tb–Tm–Te = $T_bT_mT_e$ = Загрузка – Обработка – Разгрузка

Tb = Zb: ЗД: Загрузка детали: Загрузка // в рабочую позицию





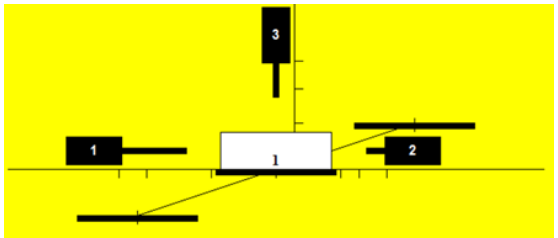
Tm: ОД: Обработка детали: Обработка // в рабочей позиции

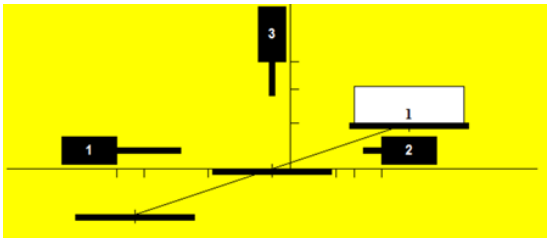


Составной технологический переход.
Параллельная обработка
трех сторон детали (сверление):

Tm = (Z1 || Z2 || Z3) = Z1 || Z2 || Z3

Te = Ze: РД: Разгрузка детали: Разгрузка // из рабочей позиции





Частный алгоритм – максимальный параллелизм: p = 3

A523 = (Tb–Tm–Te) = (Zb→(Z1 || Z2 || Z3)→Ze) = $Z_b(Z_1 \parallel Z_2 \parallel Z_3)Z_e$

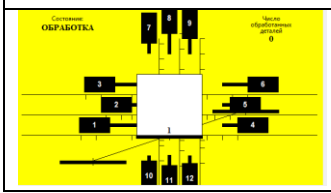
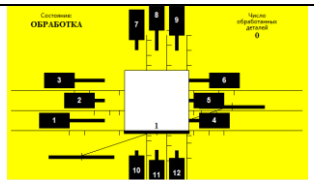
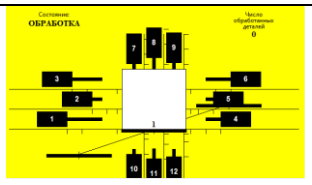
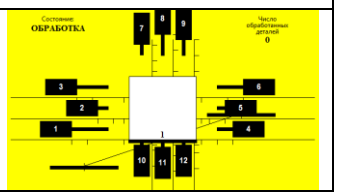
Рис. 2.3. Рабочая мнемосхема АТК3. Операционный цикл

На Рис. 2.3 представлена конфигурация комплекса АТК3. В данном случае задан вариант максимального параллелизма алгоритма 3-й степени (p = 3). Вариативность и сложность решений нарастает. Возможны несколько (простых и особых) вариантов частичного параллелизма (p = 2) [4], и несколько вариантов последовательной обработки (p = 1: вырожденного единичного параллелизма).

Обобщения первичной типовой схемы

Конфигурация АТК12 позволяет организовывать существенно более разнообразные и сложные опорные структуры процессов. В частности далее (Табл. 2.2) представлен алгоритм типовой опорной структуры ***с чередованием параллельных участков*** алгоритма (не обязательно однородных). Возможны более общие типовые опорные структуры ***с чередование последовательных и параллельных участков***. Возможен ***вложенный параллелизм***.

Табл. 2.2. Поочередная обработка сторон – группами отверстий

СФА: Структурная формула алгоритма			
A12.2 = Zb–(Z1 || Z2 || Z3)–(Z4 || Z5 || Z6)–(Z7 || Z8 || Z9)–(Z10 || Z11 || Z12)–Ze			
Параллельные фрагменты (участки) алгоритма			
(Z1 || Z2 || Z3)	(Z4 || Z5 || Z6)	(Z7 || Z8 || Z9)	(Z10 || Z11 || Z12)
			

Это ***последовательно-параллельные*** опорные структуры параллелизма – с последовательным и параллельным соединением двухполюсных структур (с одни входом и одним входом по управлению). Возможны более сложные структуры ***не параллельно-последовательного типа*** – не рекомендуются без особой необходимости. Опорные структуры могут наполняться сложными составляющими (параллельно выполняемыми) процессами с простыми и вложенными ***переключаемыми структурами*** и ***циклами***.

Параллельные процессы могут ***взаимодействовать*** между собой:

это большая и разнообразная область параллельной алгоритмики – с проблемами высокоуровневого преодоления сложности их описания и чтения.

Это как раз случаи особой необходимости сложно связанных параллельных потоков. Например, если требуется (Рис. 2.4) организация внутриоперационного асинхронного конвейерного режима для "скоростного" штамповочного РТК [3] (представлен сокращенный выборочный видеоряд):
очевидно продвижение асинхронного конвейерного потока деталей по внутриоперационному позиционному технологическому каналу.

Рис. 2.4. Мнемосхема штамповочного РТК. Конвейерный режим обработки

Со всеми такими вопросами можно знакомить слушателей с первых обзорно-ознакомительных занятий на основе демонстрационных моделей:
в опережающем порядке (для формирования адекватного целостного кругозора и исключения упрощенческих представлений) – еще до систематического изучения соответствующей проблематики (если будут такие возможности).

3 ЗАДАЧИ ЛОГИКО-ВРЕМЕННОЙ ИНТЕРПРЕТАЦИИ

3.1 Исходные положения

Как уже было отмечено, ключевой задачей определяется поэтапное решение проблем корректного и общедоступного ***понимания и объяснения логики механизмов управления*** параллельными (и, как частный случай, последовательными) дискретными процессами алгоритмических систем во времени.

Общий анализ проблемной задачи позволяет выделить следующие две исходные классификационные проблемы (Рис. 3.1) – в их пересечении:

1) ***Уровни*** логико-временной интерпретации – уровни понимания, объяснения и отражения логико-временной сущности параллельной алгоритмики:

1.1) ***Практическая логика*** здравого (интуитивного) смысла.

1.2) ***Явная структурная логика*** параллельных алгоритмов.

1.3) ***Явная структурно-функциональная логика***.

2) Проблема ***адекватности*** параллельной алгоритмики:

2.1) ***Идеализация*** (структуры) алгоритмов и способов их применения.

2.2) Учет возможности ***ошибок*** разработчика и пользователя алгоритмов. Это порождает задачи их выявления и предотвращения.

Рис. 3.1. Классификационная проблематика ЛВ интерпретации алгоритмов

3.2 Уровни (понимания) логико-временной интерпретации

3.2.1 Практическая логика здравого (интуитивного) смысла

Используется достаточно корректное практическое объяснение состава и действия алгоритмических структур во времени. Например (Рис. 3.2, Рис. 3.3):

- параллельный алгоритм имеет узел разделения (дивергенции) или распараллеливания потоков управления и узел их соединения (конвергенции);

- узлы распараллеливания обуславливают одновременное (в идеале) начало процессов параллельных ветвей, узлы соединения обеспечивают контроль завершения всех процессов параллельных ветвей.

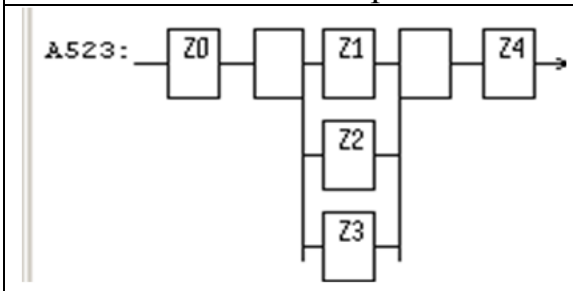
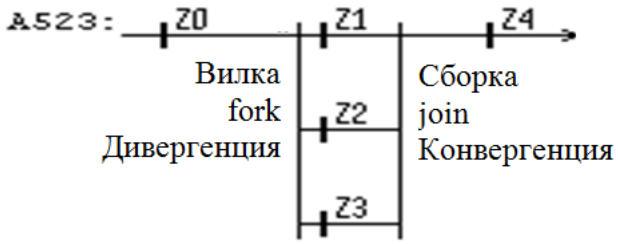
СФА: Структурная формула алгоритма / ИнФ: Инфиксная форма	
A523 = (A1→A2"→A3) = (Загрузка → Обработка →Разгрузка) = = (Z0→ (Z1 || Z2 || Z3) → Z4) = Z0 – (Z1 || Z2 || Z3) – Z4 = $Z_0(Z_1 \| Z_2 \| Z_3)Z_4$	
ССА: Структурная схема алгоритма / ГИ: Горизонтальное исполнение	
БСА: Блок-схема алгоритма	ШСА: Штрих-схема алгоритма
	

Рис. 3.2. Схемное описание структуры алгоритма

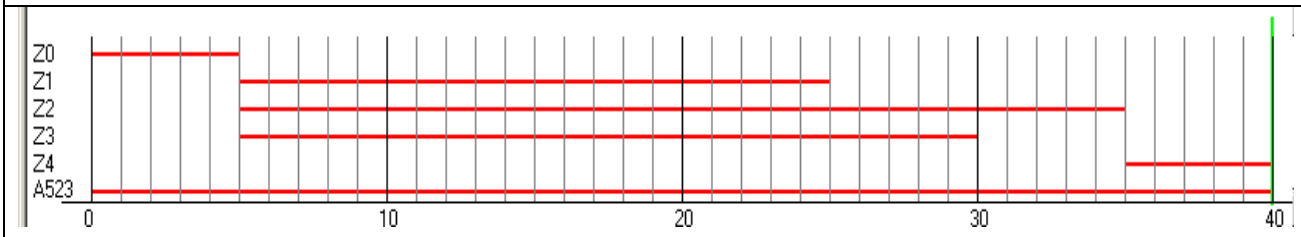
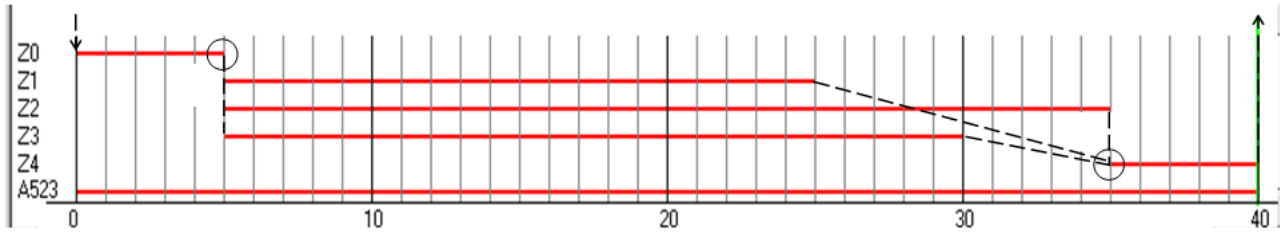
ДИА: Диаграмма исполнения алгоритма / ГИ: Горизонтальное исполнение
ЛД: Линейная диаграмма

СД: Сетевая диаграмма


Рис. 3.3. Временные диаграммы процесса исполнения алгоритма

Характерная особенность: имеют место "немые" (не обозначенные аналитически) узловые структурные элементы и "немые" связи рабочих операторов без указания обозначений для (сигналов) передачи управления между ними. Однако на этом уровне уже можно строить корректные начальные теории:

структурный синтаксис (аналитический и визуальный), алгебры структурных преобразований (при определенных разных допущениях), способы вычислений длительностей и анализа сложных комплексов действий и т.д.

3.2.2 Явная структурная логика

Явные обозначения структурных узлов

Вводятся явные аналитические обозначения для структурных узлов параллельно соединения алгоритмов (Рис. 3.4, Рис. 3.5), а операция параллели интерпретируется как парная операция в обозначениях типа || = RS = #o,

где R = # – узел разделения (дивергенции) потоков (передачи) управления;

S = o – узел соединения (конвергенции) потоков (передачи) управления.

При этом уже возможны подвижки в структурном описании алгоритмов.

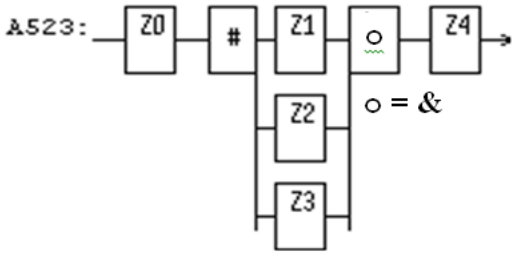
ССА: Структурная схема алгоритма / ГИ: Горизонтальное исполнение	
БСА: Блок-схема алгоритма	ШСА: Штрих-схема алгоритма
	

Рис. 3.4. Уточнение улов схемного описания структуры алгоритма

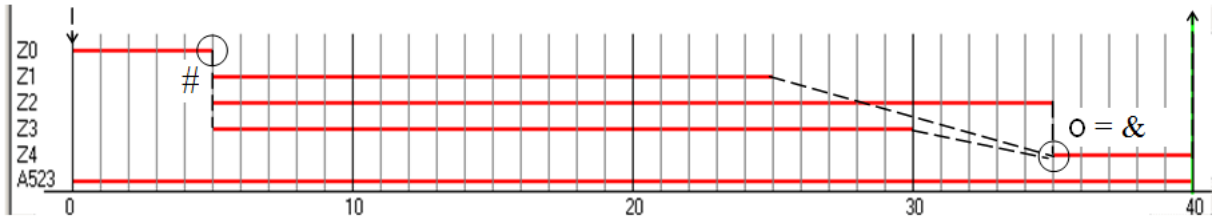
ДИА: Диаграмма исполнения алгоритма / СД: Сетевая диаграмма ГИ: Горизонтальное исполнение


Рис. 3.5. Временная диаграмма. Уточнение узловых событий

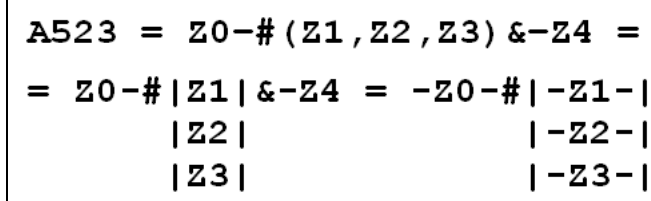
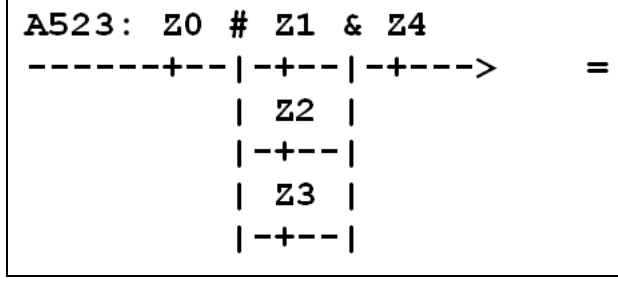
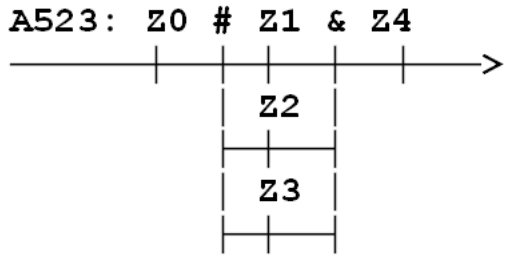
Далее в разных подходах выявляется логика этих структурных узлов:

S = o = & – узел соединения (конвергенции) потоков выполняет функцию ***конъюнкции – логическую операцию И*** (And);

R = # = R' – узел разделения (дивергенции) выполняет тривиальную функцию логического повторителя (репитера: y = R'(x) = x) сигнала передачи управления для каждого потока, а в целом – функцию множественного повторителя.

Соответственно этому уточняются структурная схема, временная диаграмма (Рис. 3.4, Рис. 3.5) и структурная формула (Табл. 3.1) алгоритма.

Табл. 3.1. Символьные структурные построения

СФА: Структурная формула алгоритма	
ИнФ: Инфиксная форма	
A523 = (A1–A2–A3) = (Z0–(Z1 || Z2 || Z3)–Z4) = = (Z0–(Z1 RS Z2 RS Z3)–Z4) = (Z0 – (Z1 #& Z2 #& Z3) – Z4)	
ИнПрПоФ: Инфиксно-префиксно-постфиксная форма	
ОФ: Одномерная форма / TNR: Шрифт типа Times New Roman	
A523 = (Z0–(Z1 #& Z2 #& Z3)–Z4) .=. .=. Z0 – #(Z1, Z2, Z3)& – Z4 = $Z_0 – \#(Z_1, Z_2, Z_3)\& – Z_4$	
ДФ: Двухмерная форма / Courier: штифт с постоянным шагом СПФ: Схемоподобная формула ~ ССА: Структурная схема алгоритма	
	
ССА: Структурная схема алгоритма / ПГ: Псевдографика	
КПГ: Клавиатурная псевдографика	СПГ: Специальная псевдографика
	

Текущие условия

На данном уровне интерпретации логики параллелизма ситуация принципиально улучшается. Это включает следующие обстоятельства:

1) Проясняется, в принципе, характер структурной операции параллели как парной операции. Более понятной, в принципе, становится логика составляю-

щих ее операций (при соответствующих практических пояснениях). Далее такие структуры именуются как ***параллельная конъюнкция*** алгоритмов || = #&.

2) Обнаружение параллельной конъюнкции ставит вопрос о возможности или невозможности параллельной дизъюнкции алгоритмов || = #V. Это относительно новая и мало разработанная сущность в параллельной алгоритмике и пока является проблемной. В статье [4] для представленного выше примера показана специфика, принципиальная возможность и полезность ее использования для вариантов частичной реализации (потенциального) параллелизма.

3) Пока не ясно следующее обстоятельство. Конъюнктивная и дизъюнктивная логика параллельного соединения алгоритмов – это действительно логические операции конъюнкции и дизъюнкции? Или это не очень понятные алгоритмические метафоры (аналогии)? Тем не менее, на интуитивном уровне можно дать подтверждение наличия таких функций. Намечается возможность привлечения булевой алгебры для базовых параллельных структур. При этом:

- с одной стороны, пока это усеченная безинверсная булева алгебра – пока не выявляется структурная операция инверсии, то есть отрицания Не (Not):

это упрощает проблему на текущем уровне интерпретации;

- с другой стороны, есть структурная операция секвенции (последовательной связи операторов), которая не включается (в традиционную) булеву алгебру:

это осложняет проблему, поскольку бесконтрольное (по смыслу) формальное применение булевой алгебры может приводить здесь к ошибкам и нелепостям.

4) Таким образом, в этой области точно выявляется ***фактор неопределенности*** (***неясности***) по существу вопроса логико-временной интерпретации. Причем любые даже самые строгие аксиоматизированные теории параллельной алгоритмики на данном (2-м) уровне логико-временной интерпретации параллельных алгоритмов (без выхода на следующий 3-й уровень) фактически содержат это внутренний фактор неопределенности. Подобные ситуации неопределенности (индефинитики [5]) вообще характерны для научных теорий.

5) Однако, это обстоятельство не является непреодолимым препятствием для применения достаточно приемлемых практических и формализованных

подходов (с корректным учетом областей и условий их применимости). Здесь есть аналогия с физикой – многие физические сущности можно измерять и строить на этом их теории, хотя не понятно, что это такое (например, энергия).

6) Кратко изложенные выше постановочные вопросы 2-го уровня существенно продвигают структурную параллельную алгоритмику опорных базовых структур. В частности:

существенно повышают точность, логичность и степень понимания структурного описания параллельных алгоритмов посредством одномерных и, особенно, двухмерных структурных формул.

Двухмерные структурные формулы алгоритмов

Двухмерные структурные формулы (СФА/ДМ) базовых структур параллельных алгоритмов приведенного выше типа (Табл. 3.1) строго выводятся из исходных одномерных структурных формул. При этом, в частности:

1) ДМ СФА могут быть преобразованы в псевдографику структурных схем (ССА/ПГ), и сами они могут интерпретироваться как особая компактная схемная псевдографика. Такая псевдографика (для достаточно простых схем) может использоваться для схемных построений в текстовых редакторах. В частности – в листингах исходных кодов программ (в комментариях):

это применяется автором в лабораторных работах по многопоточной программной реализации моделей параллельных систем.

2) Все это является основой автоматизации схемных построений базовых параллельных структур (Рис. 3.4) и временных диаграмм (Рис. 3.5).

3) На этом уровне закладывается связная аналитическая (посредством СФА) и визуальная (посредством ССА) простая первичная базовая грамматика алгоритмических языков:

в принципе она понятна (в общих чертах) на приведенных примерах структурных формул и их описания (на лабораторных занятиях изучается ее синтаксис).

3.2.3 Явная структурно-функциональная логика

Задачи структурно-функциональной интерпретации

1) Необходимо введение ***булевых переменных*** передачи управления (сигналов передачи управления) между операторами на схемах алгоритмов (для "немых" до этого связей операторов).

2) Структурная логика алгоритмов работает во времени. Для адекватного описания этого обстоятельства необходимо привлечение ***временных булевых переменных*** и ***временных булевых функций*** (ВБФ):
в общем случае с явными параметрами текущего времени t и параметрами m задержки (сдвига) во времени типа: **y(t) = F(x1(t–m1), x2(t–m2), …, y(t–m0))**.

3) Применение ВБФ непосредственно (в лоб) – это очень громоздкий низкоуровневый подход (особенно в длинных рекуррентных подстановках). Однако существуют возможности построения разных ***операторных форм*** представления временных булевых функций (ОФ ВБФ) с компактной их сверткой посредством методов квантификации (на разных интервалах времени).

4) Существует так называемая ***Темпоральная логика*** (ТЛ) – ***Временная логика*** или ***Логика времени*** (ЛВ) [6-8], которая начинает активно применяться в параллельном программировании и алгоритмике [9]. При этом:

а) Используются разные ***логико-временные операторы*** (ЛВО), которые определяются на интервалах прошлого и будущего времени. Обычно они вводятся непосредственно – "явочным порядком" с последующей аксиоматизацией их свойств. При этом возможна проблема понимания и объяснения их смысла в соотношении с обычной мыслительной практикой. Появляется задача:
точное определение (вывод и обоснование) всех логико-временных операторов на основе ВБФ, что в принципе обеспечивает разрешение этой проблемы.

б) Реально ЛВ (ТЛ) – это множество разных частных теорий разного объема и назначения. Появляется вторая актуальная задача:
их связная системная интеграция – с обоснованием на базе временных булевых функций и адаптацией на задачи параллельной алгоритмики.

При этом цели необходимо ставить по аналогии с информатикой:

- появляются все более сложные системы автоматизации информатики, в частности, для упрощения работы с ними для конечного пользователя (и повышения их дружественности в разных отношениях);
- необходимы все более сложные высокоуровневые логико-алгоритмические теоретические построения, в частности, с целью обеспечения их ясного понимания для конечного пользователя (и повышения их дружественности).

Этот вопрос требует отдельного анализа и здесь не приводится. Но далее приводятся некоторые полезные результаты для прямого применения.

Вербальная логико-временная интерпретация секвенции

Далее поэтапно вводится вербальная (словесная) логико-временная интерпретация структурной операции секвенции:

1) Исходная информация:

- в логике времени выявляется особое использование логического союза "И" (And) в роли логико-временного оператора "и затем" или просто "затем";
- этот факт уже дошел до общеобразовательной формальной логики [10].

2) Далее сопоставляются две разновидности логического союза И (And) и И' (And') с булевой функцией конъюнкции & и &':

а) Обычное применение конъюнкции для высказываний типа:

y = x1 И x2 = x2 И x1 или **x1 & x2 = x2 & x1** – коммутативная конъюнкция.

б) Особое применение конъюнкции для высказываний типа:

y = x1 И' x2 ≠ x2 И' x1 или **x1 &' x2 ≠ x2 &' x1** – некоммутативная кон-ция,

где x1 – высказывание, факты которого относятся к более раннему времени;

x2 – высказывание, факты которого относятся к более позднему времени.

Например, исходное высказывание [10] и уточнения его смысла:

Коллегия рассмотрела дело **и** приняла решение.

Коллегия рассмотрела дело, **и** коллегия приняла решение.

Коллегия рассмотрела дело, **и затем** коллегия приняла решение.

Коллегия рассмотрела дело, **затем** коллегия приняла решение.

3) Вводится логическая интерпретация операции секвенции алгоритмов: → = **–>** = **–** = **&':** **и затем**, **затем** или обобщенно **(и) затем**: **and then**, **then** или обобщенно **(and) then**.

Это, в принципе, выводит базовую алгоритмику на булеву логику.

Вербальная логико-временная интерпретация текстов алгоритмов

1) Для символов команд Zi (и алгоритмов Ai) вводится интерпретация:
выполнить команду Zi (алгоритм Ai), выполнить Zi (Ai);
execute command Zi (algorithm Ai), execute Zi (Ai).

2) Определяется вербальная (словесная) интерпретация (проговаривание) структурных формул и схем алгоритмов со структурной операцией секвенции, а также параллели. Суть этого понятна на примерах (Табл. 3.2,Табл. 3.3):

Табл. 3.2. Интерпретация последовательного (линейного) алгоритма

СФА: Структурная формула алгоритма / ИнФ: Инфиксная форма
Zb→Z1→Z2→Ze = Zb &' Z1 &' Z2 &' Ze
ЛВИ: Логико-временная интерпретация
вып. Zb **и затем** вып. Z1 **и затем** вып. Z2 **и затем** вып. Ze
выполнить Zb **затем** выполнить Z1 **затем** выполнить Z2 **и затем** выполнить Ze

Табл. 3.3. Интерпретация параллельного алгоритма

СФА: Структурная формула алгоритма ИнПрПоФ: Инфиксно-префиксно-постфиксная форма	
ОФ: Одномерная форма	ДФ: Двухмерная форма
Z0–(Z1 #& Z2 #& Z3)–Z4 = **= Z0 &'(Z1 #& Z2 #& Z3) &' Z4**	**–Z0–#|–Z1–|&–Z4–>** **|–Z2–|** **|–Z3–|**
ЛВИ: Логико-временная интерпретация	
выполнить Z0 **и затем** (выполнить Z1 **и параллельно** выполнить Z2 **и параллельно** выполнить Z3) **и затем** выполнить Z4	
выполнить Z0 **затем** (выполнить Z1 **и параллельно** выполнить Z2 **и параллельно** вып Z3) **затем** выполнить Z4	

Вместо общей формы выражения типа "выполнить Zi" можно подставлять конкретные выражения типа "выполнить Загрузку" или "загрузить" и т.п.

Все это можно рассматривать как простое дополнительное средство:

- удобный ***практический прием произношения*** (проговаривания или прочтения) формул и схем (без необходимости теоретической подготовки по ЛВ), что способствует решению проблемы их понимания и объяснения;
- шаблон текста для устного ввода параллельного алгоритма в машину и т.п.

Перевод текста алгоритма из повелительного наклонения в изъявительное

Изначально для алгоритмов их вербальная интерпретация предполагается в повелительном (императивном) наклонении типа: "выполнить команду Zi" и т.д. Но для повелительного наклонения словесных выражений формально не применима булева логика, для которой требуются высказывания (утверждения) в изъявительном наклонении. Однако возможна формальная замена:
рассматривать выражение типа "выполнить команду Zi" как сокращение выражения типа "необходимо выполнить команду Zi" (чтобы получить определенный результат) – его можно интерпретировать как высказывание.

К таким выражением можно применять булеву логику. Можно также вводить разные булевы переменные, характеризующие состояния выполнения операторов алгоритма. Это обширный проблемный вопрос. Но здесь также возможны полезные результаты прямого применения, например (Табл. 3.4):

Табл. 3.4. Дополнительная интерпретация параллельного алгоритма

<table>
<tr><td colspan="2">СФА: Структурная формула алгоритма
ИнПрПоФ: Инфиксно-префиксно-постфиксная форма</td></tr>
<tr><td>ОФ: Одномерная форма</td><td>ДФ: Двухмерная форма</td></tr>
<tr><td>Ze–(Z1 #& Z2)–Ze =
= Ze &'(Z1 #& Z2) &' Ze</td><td>-Zb-#|-Z1-|&-Ze->
|-Z2-|</td></tr>
<tr><td colspan="2">ЛВИ: Логико-временная интерпретация</td></tr>
<tr><td colspan="2">В логике Прошлого (Past) – после окончания выполнения алгоритма</td></tr>
<tr><td colspan="2">была выполнена ком. Zb и затем (была выполнена ком. Z1 и параллельно была выполнена ком. Z2) и затем была выполнена ком. Z4</td></tr>
<tr><td colspan="2">В логике Будущего (Future) – до начала выполнения алгоритма</td></tr>
<tr><td colspan="2">будет выполнена ком. Zb и затем (будет выполнена ком. Z1 и параллельно будет выполнена ком. Z2) и затем будет выполнена ком. Z4</td></tr>
</table>

Это также можно рассматривать как удобный практический прием проговаривания, понимания и объяснения структурных формул и схем такого типа.

3.3 Верификация параллельных программных систем

Для больших параллельных программных систем, взаимодействующих со средой и с многочисленными ветвлениями невозможно обеспечить их стопроцентную отладку для всех возможных сочетаний условий, возникающих по мере развертывания процесса их выполнения и в разных сочетаниях состояний параллельных процессов. Для ответственных программных систем такого типа (для атомной энергетики, космоса, опасного медицинского оборудования и т.п.) разрабатываются: методы верификации (доказательной проверки) моделей таких систем [9] и системы их автоматизации. Для таких (автоматизированных) систем верификации активно применяется Темпоральная логика.

При этом следует учитывать:

- здесь традиционно исходные логико-временные операторы вводятся непосредственно – обозначения и последующая аксиоматизация их свойств;
- существует проблема их четкого и ясного понимания и объяснения по их смыслу и свойствам в соотношении с обычной мыслительной практикой;
- здесь также актуальной является задача точного определения (вывода и обоснования) всех исходных и производных логико-временных операторов на основе ВБФ, что в принципе обеспечивает разрешение этой проблемы.

ЗАКЛЮЧЕНИЕ

Изложен способ алгоритмического описания операционных циклов производственных робототехнических систем с простыми линейными и параллельными базовыми структурами алгоритмов. Отражается общий верхний уровень описания работы систем (без частной детализации). Используются разные строго согласованные аналитические (посредством структурных формул) и схемные формы представления алгоритмов (включая псевдографику). Решается главная задача – строгая логико-временная интерпретация указанных базовых структур алгоритмов, ориентированная на Логику времени (Темпоральную логику), но

без необходимости ее изложения. Представленные данные имеют практическое и учебно-методическое значение сами по себе и как основа для расширения такого подхода на переключаемые и циклические структуры (более простой вопрос) и на взаимодействие параллельных процессов (более сложная проблема).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Житников А.П. Параллельная алгоритмика в массовой информатике и робототехнике. // Современные технологии преподавания естественно-научных дисциплин в системе общего и профессионального образования: Сб. матер. Междунар. науч.-практ. форума. – Борисоглебск: БГПИ, 2012. – С. 52-65.

2. Аналитическое агентство Boston Consulting Group (BCG). Прогноз развития сектора робототехники до 2025 года.

3. Житников А. П. Синтез асинхронного конвейерного алгоритма роботехнологического модуля. / Материалы III Всесоюзной конференции "Роботы и робототехнические системы". Т3. – Уфа: УГАТУ, 2014. С. 265 – 273.

4. Житников А.П. Параллельные алгоритмы технологических мехатронных систем. / 2-я Всероссийская науч.-тех. конф. "Мехатроника, автоматизация, управление: Сб. трудов. Том 2. – Уфа: УГАТУ, 2005. С. 155 – 160.

5. Зверев Г.Н. Теоретическая информатика и ее основания. В 2 т. Т. 1. – М.: Физматлит, 2007. – 592 с.

6. Кандрашина Е.Ю., Литвинцева Л.В., Поспелов Д.А. Представление знаний о времени и пространстве в интеллектуальных системах. – М.: Наука, 1989.

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

8. Ивин А.А. Логика времени. / Сб. "Неклассическая логика". – М.: Наука, 1970. – С. 124 – 190.

9. Карпов Ю.Г. Model Checking, Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010. – 560 с.

10. Грядовой Д.И. Логика. Практический курс основ формальной логики. Учебное пособие. – М.: Щит-М, 2007. – 284 с.