Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий управления

Кафедра интеллектуальных информационных технологий

МОДЕЛИ РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

В двух частях
Часть 1

В. П. Ивашенко

Формальные модели обработки информации и параллельные модели решения задач

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-40 03 01 «Искусственный интеллект»

Минск БГУИР 2020

УДК 004.89:004.832(075.8)
ББК 32.813я73
М74

Р е ц е н з е н т ы:
кафедра интеллектуальных систем
Белорусского государственного университета
(протокол №11 от 12.02.2020);

ведущий научный сотрудник государственного научного учреждения
«Объединённый институт информатики
Национальной академии наук Беларуси»
кандидат технических наук, доцент В. И. Романов

Модели решения задач в интеллектуальных системах. В 2 ч. Ч. 1 :
М74 Формальные модели обработки информации и параллельные модели решения задач : учеб.-метод. пособие / В. П. Ивашенко. – Минск : БГУИР, 2020. – 79 с. : ил.

ISBN 978-985-543-592-2 (ч. 1).

Излагается учебный материал, к которому прилагается список вопросов и задания для лабораторных работ.

УДК 004.89:004.832(075.8)
ББК 32.813я73

ISBN 978-985-543-592-2 (ч. 1)
ISBN 978-985-543-591-5

СОДЕРЖАНИЕ

Перечень обозначений и принятых сокращений

МКМД – множественный поток команд и множественный поток данных (архитектура)

МКОД – множественный поток команд и одиночный поток данных (архитектура)

ОКМД – одиночный поток команд множественный поток данных (архитектура)

ОКОД – одиночный поток команд одиночный поток данных (архитектура)

ЯПФ – ярусно-параллельная форма

A^B – множество однозначных отображений (функций) области B на область прибытия A

A_+^B – множество однозначных соответствий (функций) области B на область прибытия A

$A \times B$ – прямое (декартово) произведение множества A на множество B

A^* – замыкание Клини множества A

R° – транзитивное замыкание бинарного отношения R

R^{-1} – обратное отношение (инверсия) к бинарному отношению R

M^T – транспонированная матрица M

1 Введение в модели решения задач и механизмы их реализации

Чтобы задача существовала, необходим язык, на котором она будет записана.

Языки

Алфавит

Чтобы задать язык, необходимо задать алфавит. *Алфавит* – это множество различных видов **образов** (символов), мощность этого множества не более чем счётна.

Образ – результат проявления объекта. Образ может быть объектом.

Объект – познаваемая часть реальности.

Строки и тексты

Пусть для любого x

$$\{x\}_1 \underset{def}{=} \{x\}$$

и для любого натурального числа k $\left(\{\varnothing\}_k\right)$ верно

$$\{x\}_{k+1} \underset{def}{=} \left\{\{x\}_k\right\},$$

кроме того, пусть для любого множества S выполняется равенство

$$1^S \underset{def}{=} S,$$

примем также, что для любого натурального числа k и множества S выполняется

$$(k+1)^S \underset{def}{=} \left\{k^T \middle| S \supseteq T\right\}.$$

Отметим, что при k, равном единице, в соответствии с последним выражением 2^S есть степень (булеан) множества S.

Тогда *строку* (ориентированное множество) можно представить множеством

$$\bigcup_{i=1}^{k} \left\{(k-i+1)^{\{a_i\}}\right\}_i,$$

где k – длина строки;

a_i – i-я буква в строке.

Строка является *ассоциацией*.

Множества строк и текстов

Пусть задан алфавит A. Формальным языком называется подмножество

$$L \subseteq A^{(*^*)},$$

где $A^{(*^*)} \underset{def}{=} \bigcup_{i\in\mathbb{N}/\{0\}} A^{(*^i)}$; $A^{(*^{i+1})} \underset{def}{=} \left(A^{(*^i)} \cup A\right)^*$; $A^{(*^1)} \underset{def}{=} A^*$; $A^* \underset{def}{=} \bigcup_{i\in\mathbb{N}\cup\{0\}} A^i$;

A^i – декартова степень A, множество всевозможных строк длиной i в алфавите A.

Синтаксис

Синтаксис задаётся множеством синтаксических отношений.

Синтаксические отношения

Синтаксические отношения – отношения (инцидентности) между знаками. Формально синтаксические отношения – языки:

$R \subseteq 2^{A^{(*)}}$.

Синтаксические отношения R могут быть заданы с помощью грамматики.

Грамматика – множество синтаксических правил.

Каждое грамматическое правило выражает предикат логической связи характеризуемых синтаксических отношений.

Семантика

Семантика – связь знака и его значения. Семантика задаётся множеством семантических отношений.

Интерпретации

Семантические отношения – отношения между знаками и их значениями. Семантические отношения называют интерпретациями или трактовками:

$I \subseteq 2^{L \times V}$.

Здесь I – множество интерпретаций, L – язык, V – множество значений.

Операции

Операция – множество (пар) переходов:

$W \subseteq 2^{M \times M}$.

Здесь W – множество операций, M – множество состояний (ситуаций).

Каждая операция над текстами языка – язык (правило).

Если состояния – строки, то изменяющиеся символы и ассоциации в позициях – обрабатываемые (операцией) символы и ассоциации.

Обрабатываемые (операцией) символы и ассоциации могут быть: порождаемыми, дублируемыми, заменяемыми, переставляемыми, удаляемыми, поглощаемыми.

Операции могут быть: порождающими, дублирующими, заменяющими, переставляющими, удаляющими, поглощающими.

Если для какого-то состояния операция задаёт более одного перехода в него, то такое состояние называется состоянием забывания (забвения).

Если для какого-то состояния ни одна операция не задаёт переход в него, то такое состояние называется начальным.

Если для какого-то состояния ни одна операция не задаёт переход из него, то такое состояние называется конечным.

Если операция задаёт переход в то же самое состояние, то состояние называется состоянием ожидания.

Если операция задаёт переход из начального состояния, то операция называется операцией сбывания (запуска).

Если операция задаёт переход в конечное состояние, то операция называется *операцией вывода* (выхода).

Детерминированная операция. Операция, которая задаёт переход не более чем в одно состояние из любого состояния (т. е. является функцией (однозначным соответствием)), называется детерминированной операцией.

Недетерминированная операция. Операция, которая задаёт переход более чем в одно состояние из какого-либо состояния (т. е. не является функцией (однозначным соответствием)), называется недетерминированной операцией или *операцией ввода*. Соответствующие состояния называются состояниями ввода.

Позиции и символы ввода

Обрабатываемые операцией ввода символы и ассоциации (и соответствующие позиции) относительно состояния ввода являются символами и ассоциациями (и позициями) ввода.

Символы и ассоциации ввода могут быть: добавляемыми, дублируемыми, заменяемыми, переставляемыми, удаляемыми, поглощаемыми.

Операция ввода. Операции ввода (относительно состояний ввода) могут быть: добавляющими (порождающими и дублирующими), заменяющими, переставляющими, сокращающими (удаляющими и поглощающими).

Обратимая операция. Операция, задающая не более одного перехода в любое состояние (имеющая обратную функцию), называется обратимой.

Необратимая операция. Операция, задающая более одного перехода в состояние забвения, называется необратимой.

Возвратная операция. Операция, транзитивное замыкание которой является симметричным бинарным отношением, называется возвратной операцией.

Невозвратная операция. Операция, транзитивное замыкание которой не является симметричным бинарным отношением, называется невозвратной операцией.

Данные и признаки

Данные – символьный образ объекта (стандартное определение см. в [4]).

Рассмотрим множество объектов O, множество образов P и отношения между ними.

Тройка $\langle O, P, R \rangle$, где $R \subseteq O \times P$, задаёт *формальный контекст*.

Функции, отображающие объекты на образы (т. е. полностью определённые функции), называют *признаками*:

$f \in P^O$.

Признак, принимающий неотрицательные значения на множестве действительных чисел, называется *мерой*.

Метрика

Под *метрикой* понимается функция, которая для пары объектов возвращает некоторое число. Число равно нулевому элементу, только если объект в паре один и тот же, иначе число больше нулевого элемента. Число для пары объектов равно числу для пары объектов в обратном порядке. Выполняется неравенство треугольника.

Модель

Модель задаётся множеством первичных элементов и сигнатурой – множеством *n*-арных отношений над первичными элементами. В случае когда все отношения бинарные, модель называется *графовой*. Когда бинарное отношение одно, модель является *графом*.

Знания

Знания – данные, которые обладают следующими свойствами:

- связность;
- наличие сложной структуры;
- (внутренняя) интерпретируемость;
- активность;
- наличие семантической метрики.

Связность – знание включается в систему.

Наличие сложной структуры – знание состоит из элементов; оно структурированно и расширяемо, т. е. одно знание может быть включено в другое (метазнание), информационная сложность знания возрастает с возрастанием количества элементов в нём.

Интерпретируемость – элементы знания являются знаками, т. е. имеют семантику, в том числе рефлексивную семантику по отношению к другим элементам знания (метаописание).

Активность – знание обладает операционной семантикой и механизмами её реализации, реализующими переход от текущего состояния (знания) к следующему состоянию (знанию).

Наличие семантической метрики – существует признак (мера), который позволяет задать порядок отличий, оценить, насколько отлично по смыслу одно знание от другого.

Языки представления знаний

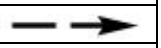
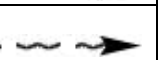
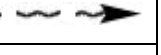
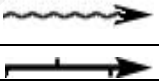
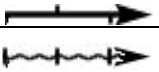
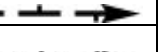
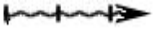
Языки представления знаний соответствуют модели представления знаний. Известными моделями представления знаний являются: фреймовая модель, продукционная модель, логическая модель, семантические сети. Модель унифицированного семантического представления знаний [10] задаёт семейство языков представления знаний, имеющих общий алфавит, синтаксис и семантику в модели ситуативных множеств. Каждый язык L этого семейства удовлетворяет следующим выражениям:

$$L \subseteq \left(\left(A^*\right)^2 \cup \left(A^*\right)\right)^*;\ \left(\left(A^*\right)^2 \cup \left(A^*\right)\right)^* \subseteq A^{(*^*)}.$$

Алфавит языков модели унифицированного семантического представления знаний позволяет выразить следующие элементарные обозначения (sc-элементы) (таблица 1), используемые в текстах этих (симметрических) языков и достаточные для представления любых знаний.

Каждый sc-элемент является знаком (обозначением), т. е. способен значить (обозначать своё значение). Частным случаем знака является sc-знак. Каждый sc-знак имеет единственное собственное значение.

Таблица 1 – Изображения sc-элементов

Название обозначения					Изображение	
					константные	переменные
sc-элемент	—					
	sc-узел					
	sc-коннектор	—			-	-
		sc-ребро				
		sc-дуга	принадлежности	постоянная		
				актуальная временная		
				неактуальная временная		
			нечёткой принадлежности			
				неактуальная временная		
				актуальная временная		
			непринадлежности	постоянная		
			общего вида			

Семантика языков модели унифицированного семантического представления знаний основана на модели ситуативных множеств [7, 10]. Модель ситуативных (событийных) множеств (sc-множеств) может быть задана следующей шестёркой компонентов:

$$\langle U, [0;1], E, r, g, \mu \rangle;\ \mu \subseteq 2^E \times \left(2^{U \times \left(Z \times \left(Z^E\right)\right)}\right);\ Z = [0;1]^{\{\varnothing\}} \cup \bigcup_q^{q \in \left(\mathbb{N}/\{0\}\right)} [0;1]^{\left([0;1]^q\right)},$$

где U – универсальное множество объектов предметной области;

E – множество элементарных событий;

r – отношение доступности (становления) событий;

g – функция, задающая множество событий существования каждого элемента универсального множества;

μ – семейство пар множеств событий существования ситуативного множества и соответствий (нечёткой) ситуативной принадлежности элементов универсального множества ситуативному множеству, отображающих элементы ситуативных множеств, множества событий и соответствующие им наборы степеней нечёткой принадлежности высших порядков на множество степеней нечёткой принадлежности.

Ситуативное множество (sc-множество) из двух компонентов (элементов) – это sc-пара.

Каждый sc-элемент – обозначение sc-множества.

Каждому константному sc-элементу сопоставляется sc-знак sc-множества, обозначаемого этим sc-элементом.

Каждый константный sc-узел не является обозначением ни одной из sc-пар, обозначаемых какой либо константной sc-дугой.

Каждая sc-дуга является обозначением sc-пары, первым компонентом которой является обозначение sc-множества, обозначаемого тем sc-элементом, из которого она выходит, и вторым компонентом которой является обозначение sc-множества, обозначаемого тем sc-элементом, в который она входит.

Каждая константная sc-дуга постоянной (не)принадлежности является обозначением sc-пары, первым компонентом которой является (обозначение) sc-знак sc-множества S, обозначаемого тем sc-элементом, из которого она выходит, и вторым компонентом которой, постоянно (не) являющимся элементом sc-множества S, является обозначение sc-множества, обозначаемого тем sc-элементом, в который она входит.

Каждая константная sc-дуга актуальной временной (не)принадлежности является обозначением sc-пары, первым компонентом которой является (обозначение) sc-знак sc-множества S, обозначаемого тем sc-элементом, из которого она выходит, и вторым компонентом которой, в настоящий момент временно (не) являющимся элементом sc-множества S, является обозначение sc-множества, обозначаемого тем sc-элементом, в который она входит.

Каждая константная sc-дуга неактуальной временной (не)принадлежности является обозначением sc-пары, первым компонентом которой является (обозначение) sc-знак sc-множества S, обозначаемого тем sc-элементом, из которого она выходит, и вторым компонентом которой, временно (не) являющимся элементом sc-множества S, является обозначение sc-множества, обозначаемого тем sc-элементом, в который она входит.

Онтологическая модель событий, явлений и процессов

Явление

Явление рассматривается как становление элементарных событий (ощущений) [7]. Каждое явление может быть задано графом событий.

Граф событий

Граф событий – граф, вершины которого соответствуют (элементарным) событиям, а рёбра – их *становлению* [7].

Каждое явление может быть задано множеством элементарных событий и их становлений.

Рёбра становления, являющиеся петлями, у которых начальная и конечная вершины совпадают, называются рёбрами бытия, и выражают понятие «*быть*».

Рёбра становления, не являющиеся петлями, у которых начальная и конечная вершины не совпадают, называются рёбрами изменения, и выражают понятие «*изменение*».

Ситуация

Ситуация – явление, которое не содержит изменений.

Движение

Движение – явление, которое содержит изменения.

Действие

Будем говорить, что одно событие предшествует другому событию, если на графе событий существует маршрут из первого события во второе. Событие, которому предшествует первое событие, будем называть последующим.

Действие – явление, которое имеет событие, предшествующее всем остальным событиям этого явления.

Взаимодействие

Взаимодействие – явление, которое имеет событие, для всех остальных событий являющееся либо предшествующим, либо последующим.

Система

Система – явление, которое для каждой пары своих различных событий включает их взаимодействие.

Процесс

Процесс – взаимодействие, рассматриваемое по отношению к некоторому явлению.

Состояние процесса – некоторое явление в процессе, не являющееся взаимодействием.

Граф состояний

Граф состояний – граф, вершины которого соответствуют состояниям (ситуациям), а рёбра – возможным переходам между ними (рисунок 1).

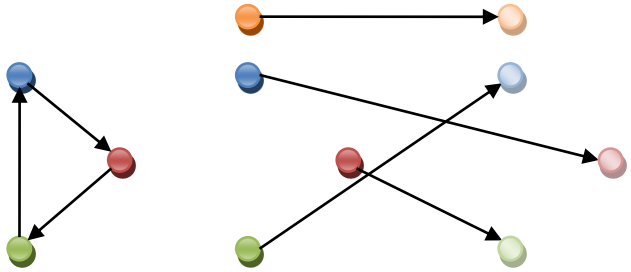


Рисунок 1 – Схема взаимосвязи графа состояний, результата тензорного произведения, с аргументами – графа ситуаций (слева) и графа временной модели (сверху)

Поток данных

Поток данных может быть задан графом событий-явлений данных (см. рисунок 1) относительно некоторого наблюдателя. В отличие от графа состояний произвольная вершина в этом графе может не задавать состояние полностью. Различают одиночный и множественный потоки данных.

В *одиночном потоке данных* одновременно не могут быть более одного экземпляра данных заданного типа.

Во *множественном потоке данных* возможно одновременно более одного экземпляра данных заданного типа.

Именование, индексация и адресация

Рассмотрим множество имён (язык) $I \subseteq A^{(*)}$. Рассмотрим также множество образов или множество данных L.

Схема доступа может быть задана множеством отображений (интерпретаций) имён I на множество данных L:
$\iota_i \in L^{I_i}$, где $I_i \subseteq I$ и $\iota_i \in L_+{}^I$.

Именование соответствует схеме доступа.

Именование (доступ) называется однородным, если и только если $I \subseteq A^n$.

Именование (доступ) называется неоднородным, если и только если $\varnothing \subset I / A^n$.

Именование (доступ) называется абсолютным, если и только если $\bigcup_i \iota_i \in L_+{}^I$.

Именование (доступ) называется относительным, если и только если $\bigcup_i \iota_i \notin L_+{}^I$.

Именование (доступ) называется несинонимическим, если и только если
$\forall i \forall s \forall x \forall z \left(\left(\{\langle x,s \rangle\} \cup \{\langle z,s \rangle\} \subseteq \iota_i \right) \to (x = z) \right)$.

Именование (доступ) называется общим, если и только если
$\forall i \forall j \forall x \forall s \exists z \left(\left(\langle x,s \rangle \in \iota_i \right) \to \left(\langle z,s \rangle \in \iota_j \right) \right)$.

Именование (доступ) называется R-смежным, если и только если $R \subseteq I^2$ и
$\forall i \forall j \forall s \forall d \forall x \forall y \left(\{\langle x, \iota_i(s) \rangle\} \cup \{\langle y, \iota_i(d) \rangle\} \subseteq \iota_j \right) \to$
$\left(\left(\langle s,d \rangle \in R \right) \to \left(\left(\langle x,y \rangle \in R \right) \vee \left(\langle y,x \rangle \in R \right) \right) \right)$.

Именование (доступ) называется R-координированным, если и только если $R \subseteq I^2$, именование (доступ) является R-смежным и
$\forall i \forall j \forall s \forall h \forall d \forall x \forall y \forall z \left(\{\langle x, \iota_i(s) \rangle\} \cup \{\langle y, \iota_i(h) \rangle\} \cup \{\langle z, \iota_i(d) \rangle\} \subseteq \iota_j \right) \to$
$\left(\left(\{\langle s,h \rangle\} \cup \{\langle h,d \rangle\} \cup \{\langle x,y \rangle\} \subseteq R \right) \to \left(\langle y,z \rangle \in R \right) \right)$,

$$\forall i\forall j\forall s\forall h\forall d\forall x\forall y\forall z\left(\left\{\left\langle x,\iota_i(s)\right\rangle\right\}\cup\left\{\left\langle y,\iota_i(h)\right\rangle\right\}\cup\left\{\left\langle z,\iota_i(d)\right\rangle\right\}\subseteq\iota_j\right)\rightarrow$$
$$\left(\left(\left\{\left\langle s,h\right\rangle\right\}\cup\left\{\left\langle h,d\right\rangle\right\}\cup\left\{\left\langle z,y\right\rangle\right\}\subseteq R\right)\rightarrow\left(\left\langle y,x\right\rangle\in R\right)\right).$$

Именование (доступ) называется R-иерархическим, если и только если $R\subseteq I^2$ и

$$\forall i\forall s\forall d\forall x\forall y\forall z\left(\left\{\left\langle x,s\right\rangle\right\}\cup\left\{\left\langle y,d\right\rangle\right\}\cup\left\{\left\langle z,s\right\rangle\right\}\subseteq\iota_i\right)\rightarrow$$
$$\left(\left(\left\{\left\langle x,y\right\rangle\right\}\cup\left\{\left\langle z,y\right\rangle\right\}\subseteq R\right)\rightarrow\left(x=z\right)\right).$$

Если именование (доступ) является ассоциативным, то $I\subseteq L$.

Абсолютное несинонимическое именование (доступ) является каноническим. Абсолютное общее именование (доступ) является централизованным. Именование является индексацией, если на множестве имён задано действие группы.

Задача

Задача может быть задана отношением между описанием исходной ситуации и описанием множества целевых ситуаций. Задачи могут быть индивидными и обобщёнными (массовыми).

Индивидная задача – задача, которая имеет только одну исходную ситуацию.

Обобщённая задача – задача, которая имеет более одной исходной ситуации и может быть задана множеством индивидных задач.

По типу решения задачи делятся на *задачи познания* – задачи, требующие применения недетерминированных операций (операций ввода), при этом множество целей таких задач состоит из не менее чем двух целей, достижение которых зависит от результата недетерминированной операции, и задачи, при решении которых не требуется применять недетерминированные операции, – *задачи исполнения*. К задачам познания относятся поиск, выбор[1], проверка (подбор), а к задачам исполнения – конструкция, реконструкция и деструкция (таблица 2).

Таблица 2 – Общая классификация задач

Задачи	Обобщённые		Индивидные	Обобщённые	
	Обратимые			Необратимые	
	Невозвратные		Возвратные		Невозвратные
Познания	Поиск		–	Выбор	Проверка
Исполнения	Конструкция		Реконструкция	–	Деструкция

Решение задачи

Решение задачи – это процесс, который может быть задан множеством путей на некотором графе состояний.

[1] Термины «выбор» и «проверка» использованы так, а не наоборот, из соображений о том, что выбор предполагает некоторые представления (в том числе из опыта) об его исходе, тогда как проверка – не предполагает.

Поиск на графе состояний

Если при решении задачи ситуации и состояния представляются полностью, то такой подход к решению является поиском (на графе) состояний (ситуаций). Однако если одновременно представлено более одного состояния (ситуации), то задача выражается метазадачей поиска на графе состояний (ситуаций), где переход от исходной или целевой ситуации (состояния) исходной задачи в направлении навстречу друг к другу позволяет свести исходную задачу к ИЛИ-подзадаче (ограниченная редукция метазадачи, дизъюнктивная редукция). Различают разные методы (алгоритмы) поиска на графе: поиск в глубину (рисунок 2), поиск в ширину (рисунок 3) и другие [16]. Если нет никакой дополнительной информации, то поиск осуществляется «вслепую».

$$
\begin{array}{l}
\textit{найтиВглубь} \\
\langle\langle V, E\rangle, s, D\rangle \leftarrow \\
p \leftarrow \langle s\rangle \\
\textit{если}\left(s \notin D\right) \\
\quad p \leftarrow \varepsilon \\
\quad \textit{для}\langle s, x\rangle \in E \\
\quad\quad r \leftarrow \langle s\rangle + \textit{найтиВглубь}\left(\left\langle\left\langle V / \{s\}, E / \left(\left(\{s\} \times V\right) \cup \left(V \times \{s\}\right)\right)\right\rangle, x, D\right\rangle\right) \\
\quad\quad \textit{если}\left(r \neq \langle s\rangle\right) \\
\quad\quad\quad p \leftarrow r \\
\leftarrow p
\end{array}
$$

Рисунок 2 – Алгоритм поиска решения задачи на графе состояний в глубину

Недостатком алгоритма поиска в глубину является то, что он применим только для конечных графов состояний, для бесконечных графов поиск в глубину в общем случае не является результативным, т. е. может не завершиться (когда цель не лежит на выбранной бесконечной ветви). Достоинством алгоритма является то, что с целью отката (возврата) нужно запоминать вершины только текущей выбранной ветви, что удобно делать через стек рекурсивного вызова.

Недостатком алгоритма поиска в ширину является то, что он применим только для графов состояний с конечной полустепенью исхода (на практике крайне маловероятно, что встретится граф состояний с бесконечной полустепенью исхода, как, например, в случае формальной модели с бесконеч-

ным количеством операций или с операциями с бесконечной степенью недетерминизма), иначе утрачивается результативность. Более практически существенным недостатком алгоритма является необходимость хранить все вершины яруса, на котором ведётся поиск, так как количество таких вершин может существенно возрастать, значительно быстрее (пропорционально геометрической прогрессии), чем длина искомого решения.

Алгоритм итеративного поиска в глубину является результатом попытки сочетания достоинств поиска в глубину и поиска в ширину (рисунки 4, 5).

Алгоритм комбинированного поиска решения задачи на графе состояний применим для любых графов состояний (рисунок 6).

найтиВширь

$\langle\langle V, E\rangle, S, D\rangle \leftarrow$

если $(\varnothing = S \cap D)$

$p \leftarrow \varepsilon$

$H \leftarrow \left\{x \middle| \left(\langle s, x\rangle \in E\right) \wedge \left(s \in S\right)\right\} / S$

$r \leftarrow$ *найтиВширь* $\left(\left\langle\left\langle V / S, E / \left(\left(S \times V\right) \cup \left(V \times S\right)\right)\right\rangle, H, D\right\rangle\right)$

если $(r \neq \varepsilon)$

$s \in S \cap \left\{x \middle| \langle x, r_1\rangle \in E\right\}$

$p \leftarrow \langle s\rangle + r$

иначе

$d \in S \cap D$

$p \leftarrow \langle d\rangle$

$\leftarrow p$

Рисунок 3 – Алгоритм поиска решения задачи на графе состояний в ширину

найтиИтеративно

$\langle\langle V, E\rangle, s, D\rangle \leftarrow$

$N \leftarrow 1$

$p \leftarrow \langle s\rangle$

пока $\left(\left(p = \langle s\rangle\right) \wedge \left(s \notin D\right)\right)$

$p \leftarrow$ *найтиВглубьДо* $\left(\langle\langle V, E\rangle, s, D, 1, N\rangle\right)$

$N \leftarrow N + N$

$\leftarrow p$

Рисунок 4 – Алгоритм итеративного поиска решения задачи на графе состояний

При наличии дополнительной информации, например, при возможности вычислить функцию (критерий), (анти)монотонную относительно порядка приближения от исходной ситуации к целевым, используются другие подходы, методы и алгоритмы. Одним из подходов является «метод границ и ветвей».

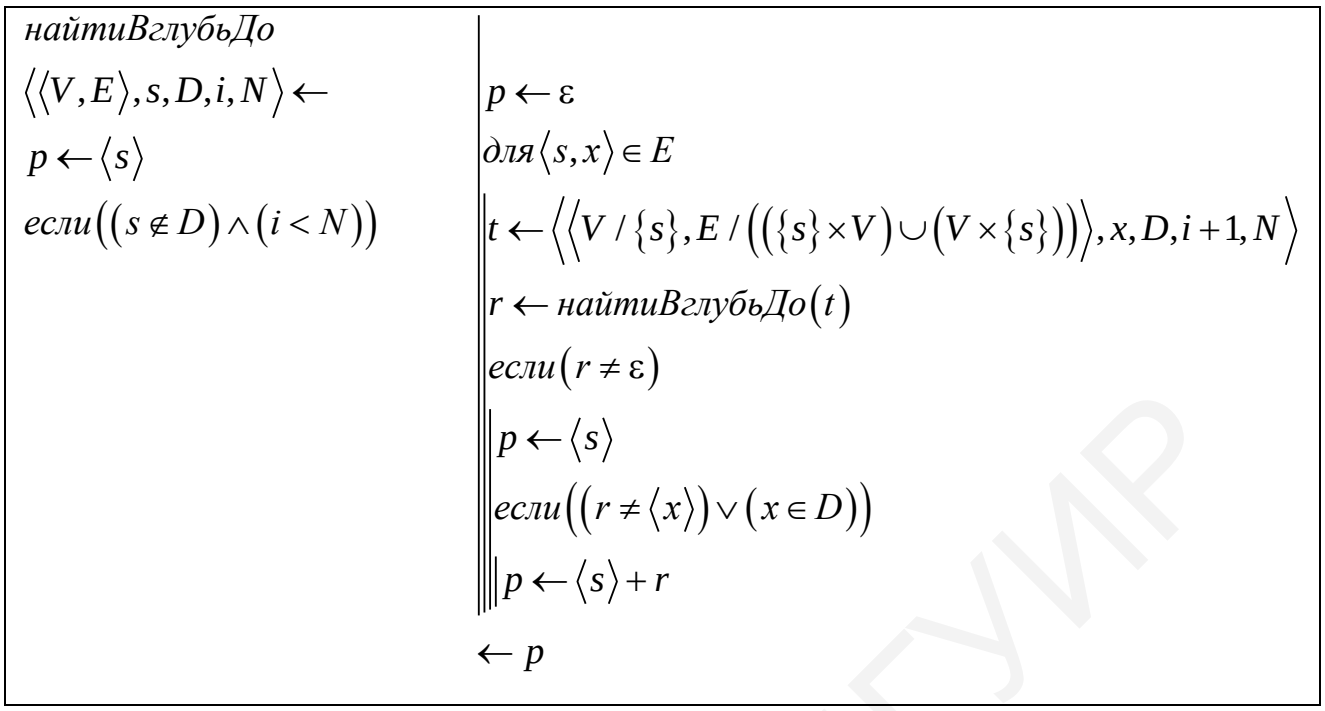


Рисунок 5 – Алгоритм ограниченного поиска решения задачи на графе состояний в глубину

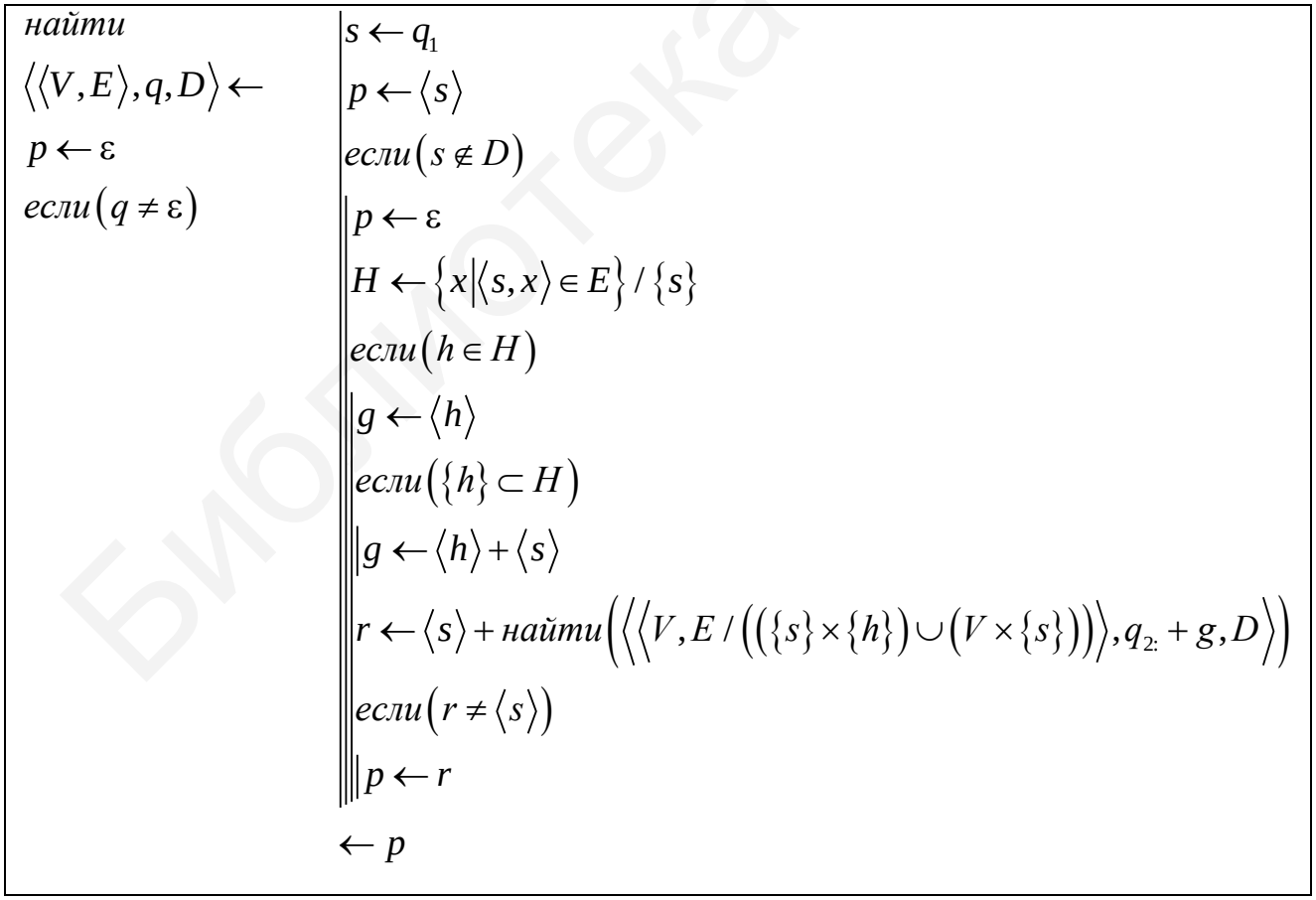


Рисунок 6 – Алгоритм комбинированного поиска решения на графе состояний

Сложность (решения) задачи оценивается типом зависимости числа затрачиваемых ресурсов при её решении от величины сложности формулировки задачи (количества символов и их ассоциаций в тексте) или от её ранга. На графиках, изображённых на рисунках 7 и 8, приведены разные типы зависимостей.

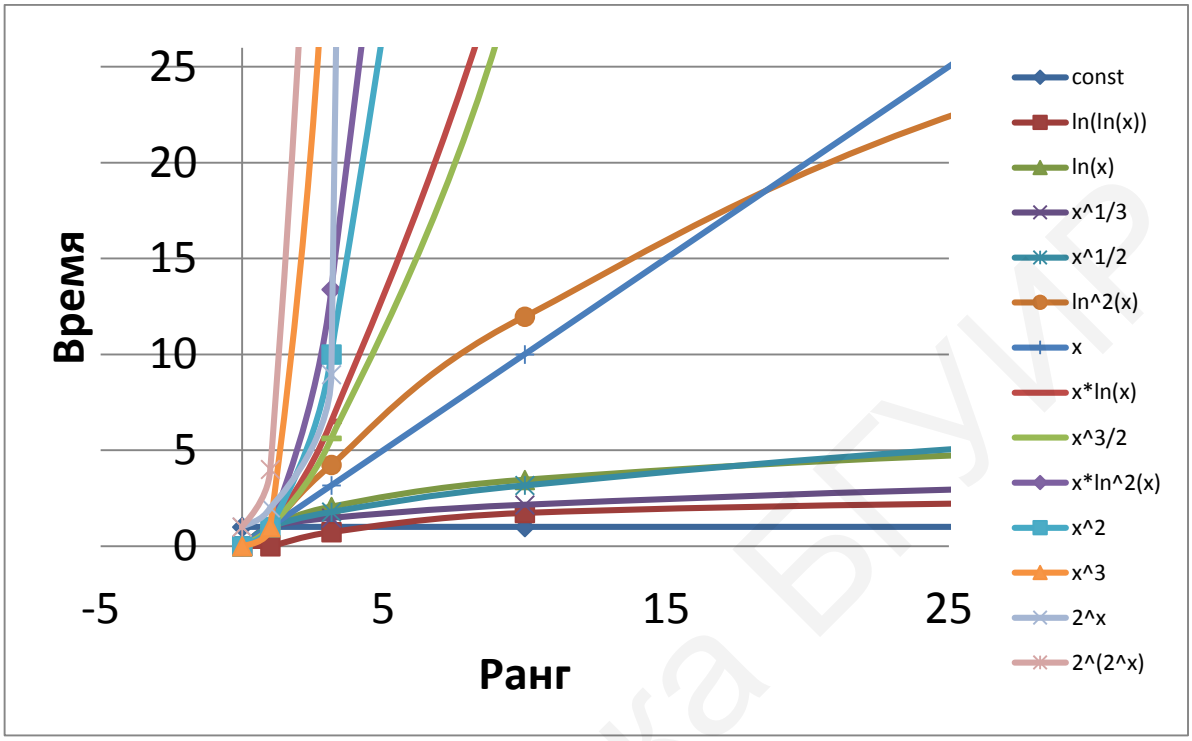


Рисунок 7 – График видов зависимостей величины времени от величины ранга

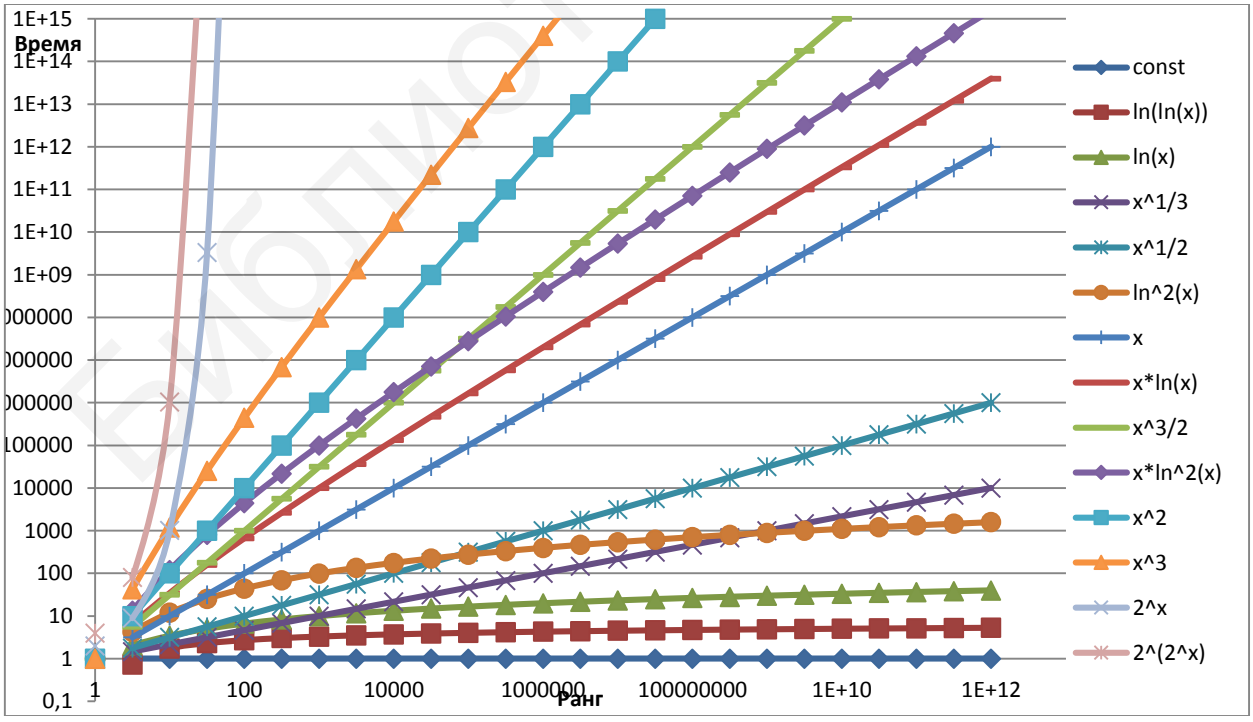


Рисунок 8 – График видов зависимостей величин в логарифмических шкалах

Отношения и операции на графах состояний

Подграфом $\langle V,E\rangle$ графа $\langle W,A\rangle$ является граф $\langle V,E\rangle$, если и только если: $V\subseteq W$, $E\subseteq A$.

Гомоморфизмом графа $\langle V,E\rangle$ в граф $\langle W,A\rangle$ является h, если и только если:

$h\in(W\cup A)^{(V\cup E)}$,

$\left(h\cap(V\times W)\right)\in W^V$,

$\left(h\cap(E\times A)\right)\in A^E$,

$\forall v\forall u\left(\left(\langle v,u\rangle\in E\right)\to\left(h\left(\langle v,u\rangle\right)=\langle h(v),h(u)\rangle\right)\right)$.

Изоморфное вложение графа $\langle V,E\rangle$ в граф $\langle W,A\rangle$ ($\langle V,E\rangle\tilde{\sqsubseteq}\langle W,A\rangle$), если и только если h – гомоморфизм графа $\langle V,E\rangle$ в граф $\langle W,A\rangle$ и

$h^{-1}\in(V\cup E)_{+}^{(W\cup A)}$.

Изоморфизм графов состояний ($\langle V,E\rangle\cong\langle W,A\rangle$), если и только если $\langle V,E\rangle\tilde{\sqsubseteq}\langle W,A\rangle$ и $\langle W,A\rangle\tilde{\sqsubseteq}\langle V,E\rangle$.

Если между графами состояний и соответствующими моделями существует изоморфное вложение, то это значит, что каждое решение задач во вкладывающемся графе состояний присутствует и в графе состояний, в который он вкладывается. Граф состояний, в который вкладывается любой граф состояний заданного множества моделей, будет являться графом универсальной модели обработки информации для этого множества (для детерминированных вычислений универсальной моделью информации является модель универсальной детерминированной машины Тьюринга). Граф Радо [33] является примером графа состояний универсальной модели обработки информации.

Объединением графов $\langle V,E\rangle$ и $\langle W,A\rangle$ является граф $\langle V\cup W,E\cup A\rangle$.

Пересечением графов $\langle V,E\rangle$ и $\langle W,A\rangle$ является граф $\langle V\cap W,E\cap A\rangle$.

Произведением (прямым) графов $\langle V,E\rangle$ и $\langle W,A\rangle$ является граф

$$\left\langle V\times W,\left\{\langle\langle v,u\rangle,\langle v,w\rangle\rangle\middle|\langle v,\langle u,w\rangle\rangle\in V\times A\right\}\cup\left\{\langle\langle v,w\rangle,\langle u,w\rangle\rangle\middle|\langle\langle v,u\rangle,w\rangle\in E\times W\right\}\right\rangle.$$

Произведением (тензорным) графов $\langle V,E\rangle$ и $\langle W,A\rangle$ является граф

$$\left\langle V\times W,\left\{\langle\langle v,u\rangle,\langle y,w\rangle\rangle\middle|\langle\langle v,y\rangle,\langle u,w\rangle\rangle\in E\times A\right\}\right\rangle.$$

Игровая схема

Множество моделей, сигнатура которых состоит из операций, задаёт *игровую схему*. Количество игроков равно мощности этого множества. Если носитель каждой модели является подмножеством формального языка, то он является языком игровой схемы. Операции каждой из моделей являются операциями, реализуемыми соответствующим игроком.

Стратегией игрока является множество операций, реализуемых этим игроком.

Информационные системы и системы обработки информации

Формальная модель обработки информации, знаний

Формальная модель обработки информации [8] рассматривается как формальная система и может быть задана четвёркой:

$\langle A, S, B, W \rangle$,

где A – алфавит;

S – синтаксис (грамматика);

B – множество начальных состояний;

W – множество операций.

Множество начальных состояний является подмножеством множества всех состояний M, которое в свою очередь является подмножеством языка L с грамматикой S в алфавите A:

$B \subseteq M \subseteq L$.

Множество операций является множеством бинарных отношений на множестве M:

$W \subseteq 2^{M \times M} \subseteq 2^{L \times L}$.

Формальная модель обработки знаний задаётся семёркой [24]:

$\langle A, S, B, R, I, W, \mu \rangle$,

где A – алфавит;

S – синтаксис (грамматика);

B – множество начальных состояний;

R – множество синтаксических отношений (выражающих свойства связности и структурированности);

I – множество интерпретаций (свойство интерпретируемости);

W – множество операций (свойство активности);

μ – семантическая метрика.

Множества B, R, I и W удовлетворяют:

$B \subseteq M \subseteq L$; $R \subseteq 2^{A^{(**)}}$; $I \subseteq 2^{L \times V}$; $W \subseteq 2^{M \times M} \subseteq 2^{L \times L}$.

Элементарная модель обработки информации

Модель обработки информации с единственной операцией называется элементарной.

Модель детерминированной обработки информации

Модель детерминированной обработки информации (детерминированных вычислений) имеет только детерминированные операции.

Детерминирование – переход от исходной модели к модели детерминированной обработки информации (может быть интенсивным или экстенсивным). При интенсивном детерминировании каждая детерминированная операция ис-

ходной модели является элементом сигнатуры новой модели, а каждая недетерминированная операция исходной модели соответствует множеству детерминированных операций новой модели, объединение которых является этой недетерминированной операцией. Других операций сигнатура новой модели не имеет.

При экстенсивном детерминировании для исходной модели $\langle M, S \rangle$ найдётся такая биекция g, отображающая сигнатуру исходной модели на сигнатуру новой модели детерминированных вычислений с носителем $Z \subseteq (M \times X) \cup M$, что для любой недетерминированной операции o исходной модели верно

$$\forall m(m \in M) \to \left(\forall y(\langle m, y \rangle \in o) \sim \exists x(x \in X) \wedge \left(\varnothing \subset \left(\{\langle m, x \rangle\} \times \{y, \langle y, x \rangle\}\right) \cap g(o)\right)\right),$$

а для любой детерминированной операции o исходной модели верно

$$\forall m(m \in M) \to \left(\forall y(\langle m, y \rangle \in o) \sim \exists z\left(z \in \left(\{m\} \times X\right) \cup \{m\}\right) \wedge \left(\langle z, y \rangle \in g(o)\right)\right).$$

Модель недетерминированной обработки информации

Модель недетерминированной обработки информации имеет недетерминированные операции.

Модель базовой обработки информации

Модель базовой обработки информации имеет непересекающиеся операции.

Переход от заданной модели к соответствующей модели базовой обработки информации – базификация.

Модель обратимой обработки информации

Модель обратимой обработки информации (обратимых вычислений) имеет только обратимые операции.

Обращение – переход к модели с обратными операциями.

Модель необратимой обработки информации

Модель необратимой обработки информации (необратимых вычислений) имеет только необратимые операции.

Модель возвратной обработки информации

Модель возвратной обработки информации (возвратных вычислений) имеет множество операций, транзитивное замыкание объединения элементов которого является симметричным бинарным отношением.

Возвращение – переход к модели возвратных операций. Обратный переход – невозвращение.

Модель невозвратной обработки информации

Модель невозвратной обработки информации (невозвратных вычислений) имеет множество операций, транзитивное замыкание объединения элементов которого не является симметричным бинарным отношением.

Модель безрезультативной обработки информации

Операции модели безрезультативной обработки информации (безрезультативных вычислений) не имеют конечных состояний.

Модель небезрезультативной обработки информации

Операции модели небезрезультативной обработки информации (небезрезультативных вычислений) имеют конечные состояния.

Модель результативной обработки информации

Операции модели результативной обработки информации (результативных вычислений) не имеют циклов и бесконечных простых цепей, не имеющих конечных состояний.

Модель нерезультативной обработки информации

Операции модели нерезультативной обработки информации (нерезультативных вычислений) имеют циклы или бесконечные простые цепи, не имеющие конечных состояний.

Модель цикличной (небезвозвратной) обработки информации

Операции модели цикличной обработки информации (цикличных вычислений) имеют циклы.

Модель ацикличной (безвозвратной) обработки информации

Операции модели ацикличной обработки информации (ацикличных вычислений) не имеют циклов.

Модель бесконечной обработки информации

Операции модели бесконечной обработки информации (бесконечных вычислений) имеют бесконечные цепи.

Модель конечной обработки информации

Операции модели конечной обработки информации (конечных вычислений) не имеют бесконечных цепей.

Замкнутая модель обработки информации

Замкнутая модель обработки информации содержит все результаты композиций всех операций модели.

Модель одноэтапной обработки информации

Операции модели одноэтапной обработки информации (одноэтапных вычислений) не имеют маршрутов длиной 2.

Команды

Команда – текст языка, строка κ, которая состоит из подстрок – обозначения (кода) операции $\langle\gamma\rangle$ и обозначений (имён, адресов) её операндов (параметров) $\langle\alpha,\beta\rangle$:

$\kappa=\langle\gamma,\alpha,\beta\rangle$.

Параметры в команде могут быть входными и выходными. Множество входов команды κ будем обозначать $IN(\kappa)$ (множество входов команды κ может быть определено как объединение множества входов всех команд, выходы которых являются её входами). Множество выходов команды будем обозначать $OUT(\kappa)$.

Поток команд

Поток команд является частным случаем потока данных, где данными являются команды. Различают одиночный поток команд и множественный поток команд.

Независимые и зависимые команды, виды зависимостей

Пусть β и α – последовательные команды в потоке команд (β предшествут α). Между ними существует *прямая зависимость* [19, 20, 21], если выполняется выражение

$\varnothing\subset IN(\alpha)\cap OUT(\beta)$.

Следует отметить, что если команды β и γ и γ и α обладают прямой зависимостью, то команды β и α также обладают прямой зависимостью, т. е. прямая зависимость транзитивна.

Команды обладают *обратной зависимостью*, если и только если выполняется:

$\varnothing\subset OUT(\alpha)\cap IN(\beta)$.

Команды обладают *конкуренционной зависимостью*, если и только если выполняется:

$\varnothing\subset OUT(\alpha)\cap OUT(\beta)$.

Команды являются *зависимыми командами* [19, 20, 21], если выполняется выражение

$\varnothing\subset\big(OUT(\alpha)\cap OUT(\beta)\big)\cup\big(OUT(\alpha)\cap IN(\beta)\big)\cup\big(IN(\alpha)\cap OUT(\beta)\big)$.

Если *команды независимы* (являются *независимыми командами*), то приведённое выражение не выполняется.

Параллелизм, виды и формы параллелизма и их признаки

Параллелизм – возможность одновременного (допустимого) выполнения команд (операций) [19].

Физический параллелизм – параллелизм, который реализован, т. е. реализованная возможность одновременного выполнения команд (операций).

Логический параллелизм – параллелизм, который не реализован.

Можно выделить другие виды параллелизма, если под *гранулой* (*зерном*) в графе потока данных (графе потока управления) понимать непустой связный подграф, полученный объединением всех простых цепей, начало которых принадлежит одному множеству вершин, а конец – другому множеству вершин, при условии, что любые две вершины в каждом одном из множеств соответствуют независимым операциям. Длина гранулы – максимальное количество из всех количеств вершин каждой простой цепи гранулы. Длина гранулы равна количеству последовательно выполняемых команд в грануле. Гранулы абсолютно независимы, если и только если каждые различные две из них не имеют общих вершин и каждая пара вершин разных гранул соответствует независимым операциям (в частности, отсутствует простая цепь от одной вершины к другой). *Гранулярность* – минимальная длина гранулы из длин всех гранул во множестве абсолютно независимых гранул [19].

Если гранулярность не ниже 100 000, то параллелизм относится к *крупногранулярному (крупнозернистому) параллелизму*.

Если гранулярность не выше 100, то параллелизм относится к *мелкогранулярному (мелкозернистому) параллелизму*.

Если гранулярность не позволяет отнести параллелизм ни к крупногранулярному, ни к мелкогранулярному, то параллелизм относится к *среднегранулярному (среднезернистому) параллелизму* [18].

Под *естественным параллелизмом* понимается возможность одновременного выполнения операции над однотипными данными. Когда данные можно рассматривать как вектор, то используется название «векторный параллелизм».

В литературе можно встретить некорректное по отношению к определению название «параллелизм данных». Под «параллелизмом данных» понимается возможность одновременного выполнения операций над данными во множественном потоке данных, заданных одной командой (операцией). Более корректно использовать названия «векторный параллелизм» или «естественный параллелизм».

Параллелизм независимых ветвей – крупногранулярный параллелизм различных операций (команд), которые соответствуют разным ветвям в графе потока данных и в графе управления программы. Иногда в литературе можно встретить название «параллелизм задач». Ветвь – простая цепь в графе. Две ветви независимы, если и только если они не лежат на одной ветви и их пересечение не имеет их внутренних вершин.

Параллелизм смежных операций (*скалярный параллелизм*) – мелкогранулярный параллелизм различных операций (последовательных команд (смежных в потоке команд)).

Информационный граф, ярусно-параллельная форма

Информационный граф [19, 20, 21] – граф, вершинам которого соответствуют команды (операции), а рёбрам – потоки передаваемых между операциями данных (рисунок 9).

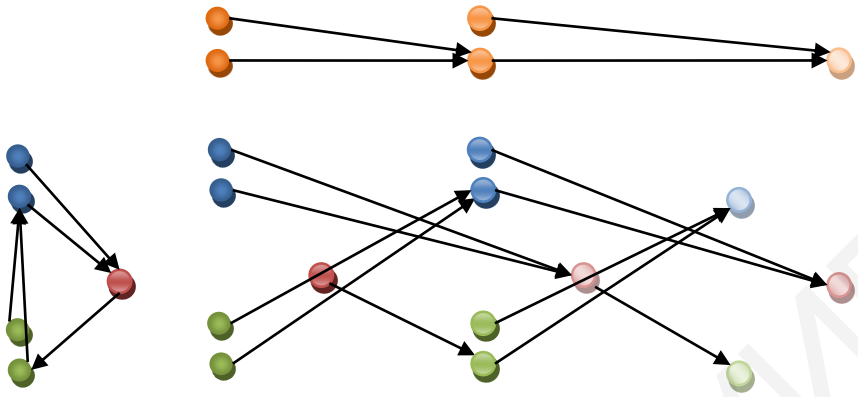


Рисунок 9 – Схема взаимосвязи графа событий и его проекций – графа элементарных ситуаций и информационного графа

Ярусно-параллельная форма (ЯПФ) – разбиение множества вершин информационного графа на упорядоченное (индексированное) множество ярусов, такое, что [19]:

- все вершины команд чтения исходных данных находятся на нулевом ярусе;
- для любых положительных натуральных i ярус i является множеством всех тех и только тех вершин, которые соответствуют командам, значения всех входов которых вычислены на предыдущих ярусах, с яруса 0 по ярус $i-1$, но не раньше.

На рисунке 10 представлены информационный граф и ЯПФ выражения $r = \dfrac{a \cdot b - c \cdot d}{a \cdot c + b \cdot d}$.

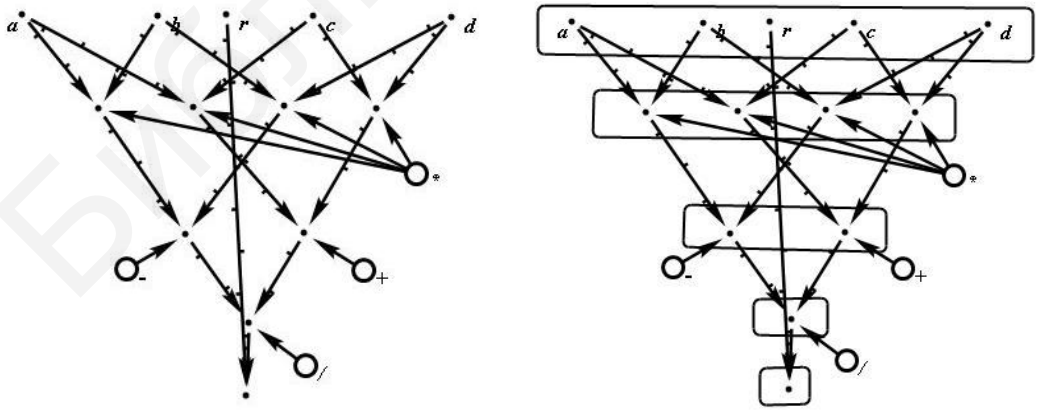


Рисунок 10 – Информационный граф и его ярусно-параллельная форма

Ранг задачи – максимальное количество экземпляров данных (одного типа) с необходимостью подлежащих обработке, которые могут быть обработаны (вне зависимости от варианта реализации) при решении этой задачи одновременно.

Вычислительная система

Вычислительная система [1, 2], как правило, состоит из процессора, памяти и устройств ввода-вывода.

Память

Память – система, которая в своих состояниях хранит тексты некоторого языка из некоторого множества, и поддерживает операции чтения-записи.

Процессор

Вычисление – процесс перехода от исходного состояния данных к другому состоянию данных, отображаемых на числовую шкалу. Для работы с числами используются алгебраические операции.

Алгебраическая операция – операция, область определения которой равна её области прибытия или декартовой степени её области прибытия:
$\bullet \in A^B$, где $B = A$ или $B = A^n$.

Алгебраическая система, которая может быть задана тройкой, элементами которой являются: носитель (множество первичных элементов), множество *n*-арных отношений на элементах носителя, множество алгебраических операций над элементами носителя. Последние два множества называются сигнатурой алгебраической системы.

Примерами алгебраических систем являются:
- полугруппа;
- моноид;
- группа;
- кольцо;
- поле;
- модуль;
- линейное векторное пространство.

Полугруппа – алгебраическая система, элементом сигнатуры которой является единственный элемент, являющийся ассоциативной бинарной алгебраической операцией.

Бинарная ассоциативная операция ($\bullet$):
$\left(a \bullet \left(b \bullet c\right)\right) = \left(\left(a \bullet b\right) \bullet c\right)$.

Примерами бинарных ассоциативных операций являются:
- побитовая операция Исключающее ИЛИ;
- побитовая операция ИЛИ;
- побитовая операция И;
- целочисленное сложение по модулю;
- целочисленное умножение по модулю.

Моноид – полугруппа, элементом носителя которой является нейтральный элемент.

Нейтральный элемент e (e – левый, e' – правый):
$\left(a \bullet e'\right) = a$,

$(\overleftarrow{e} \bullet a) = a$,

$e = \overleftarrow{e} = e'$.

Примерами нейтральных элементов являются:

- битовый вектор нулей;
- нуль;
- единица.

Группа – моноид с обратными элементами.

Обратный элемент $\overline{a}$ ($\overleftarrow{a}$ – левый, $\overline{a}'$ – правый):

$(\overline{a}' \bullet a) = e'$,

$(\overleftarrow{a} \bullet a) = \overleftarrow{e}$,

$(\overline{a} \bullet a) = e$.

Примерами моноидов являются:

- группы перестановок: циклические, диэдральные, антисимметрические, симметрические;
- сложение целых чисел;
- умножение рациональных чисел и др.

Кольцо – алгебра с двумя операциями, вторая операция дистрибутивна к первой, первая коммутативна и образует на носителе коммутативную (абелеву) группу, а вторая образует полугруппу.

Дистрибутивность:

$(a \bullet (b \circ c)) = ((a \bullet b) \circ (a \bullet c))$.

Примерами колец являются:

- побитовые операция Исключающее ИЛИ и операция И;
- целочисленные сложение и умножение по модулю.

Поле – частный случай кольца: кольцо со второй операцией, проекция которой на носителе без нейтрального элемента первой операции образует группу.

Модуль – абелева группа с действием кольца над ней.

Линейное векторное пространство – частный случай модуля, абелева группа с действием поля над ней.

Понятие управления связано с выработкой и реализацией программы (системы команд), что в простейшем случае сводится к переходу от одной команды (операции) к другой (операции), когда последняя команда (её адрес) определяется в результате работы первой («зависимость по управлению»). Это можно выразить в виде следующих классов (управляющих) операций:

$\left(2^{M \times M}\right)^X$; $\left(M^M\right)^X$,

которые представляют классы детерминированных операций, выражающих некоторое управляющее действие по определению операции (детерминированной операции) на множестве M.

Следует отметить, что в силу равномощности существует взаимно однозначное соответствие между парами классов:

$$\left|\left(2^{M\times M}\right)^{X}\right| = \left|2^{(M\times X)\times M}\right|,$$

$$\left|\left(M^{M}\right)^{X}\right| = \left|M^{M\times X}\right|.$$

Это позволяет осуществить переход с помощью преобразования обратного «каррированию», от унарной управляющей операции к бинарной операции:

$\varphi(x)(m) = \psi(\langle m, x\rangle)$, где $\varphi \in \left(M^{M}\right)^{X}$, а $\psi \in M^{M\times X}$.

Таким образом, удобным способом выражения явлений управления является бинарная операция. С помощью бинарных операций можно выражать действие групп и другие преобразования. Бинарная операция также может быть получена в результате экстенсивного детерминирования.

Основными характеристиками вычислительных систем касательно потоков команд и данных являются производительность и пропускная способность. *Производительность* выражается отношением количества (команд) операций ко временному интервалу, за который они выполнены, а *пропускная способность* – количеству элементов данных ко временному интервалу.

$$P(x) = \frac{W(x)}{T(x)}; \qquad B(x) = \frac{Q(x)}{T(x)}.$$

Производительность соответственно измеряется в операциях на единицу времени (операции в секунду (operations per second (OPS))).

Коэффициент ускорения выражается отношением времени выполнения программы на одной архитектуре (реализации) по отношению ко времени выполнения на другой:

$$r = \frac{T(x)}{T(y)}.$$

Закон Амдала задаёт теоретическую оценку коэффициента ускорения:

$$r = \frac{W\cdot\tau}{\left(w_\sigma + \frac{w_\pi}{p}\right)\cdot\tau} = \frac{w_\sigma + w_\pi}{w_\sigma + \frac{w_\pi}{p}} = \frac{1}{\sigma + \frac{1-\sigma}{p}},$$

где w_σ – количество последовательно выполняемых операций (самой длинной ветви информационного графа) одинаковой длительности τ;

w_π – количество параллельно выполняемых операций (параллельно по отношению к последовательно выполняемым операциям) одинаковой длительности τ;

σ – доля последовательно выполняемых операций;

p – количество процессорных элементов (процессоров, параллельных потоков).

Эффективность выражается отношением достигнутого эффекта (коэффициента ускорения) к числу затрат (процессоров или процессорных элементов):

$$e = \frac{r}{p}.$$

Коммутация и коммутаторы

Сеть коммутации [1, 18, 20] представляет собой множество связей (рисунок 11) между двумя наборами узлов, которые называются входами и выходами. Всего между N входами и M выходами существует N^M различных связей, включая парные связи («один – одному»), связи «один – многим», смешанные связи.

Сеть, осуществляющая N^M соединений, называется *обобщенной соединительной сетью* (координатным переключателем) [20].

Если сеть осуществляет только парные связи, она называется *соединительной*, в ней возможно $N!$ соединений. В соединительных сетях любой допустимый набор связей (перестановка) выполняется за один такт и без конфликтов.

Коммутатором [18, 20], противоположным по своим свойствам полному координатному соединителю, является *общая шина* (UNIBUS) – общая магистраль. Этот коммутатор способен за один такт выполнить соединение только между одним входом и одним выходом в отличие от координатного переключателя, который за один такт выполняет все соединения.

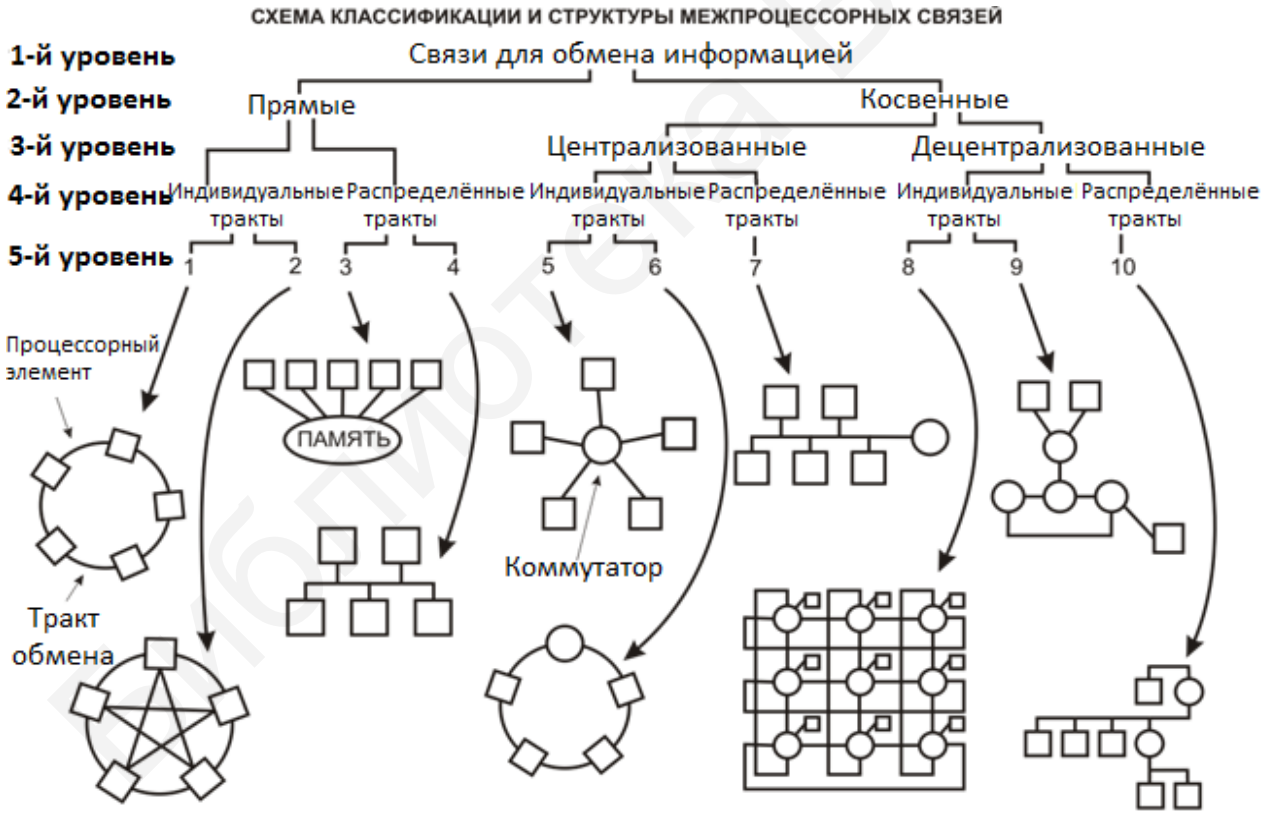


Рисунок 11 – Схема классификации и структуры межпроцессорных связей [3]

Среда – однокаскадный коммутатор, каждый узел которого связан только со своими ближайшими соседями [18, 20].

Систематика архитектур вычислительных систем

ОКОД – архитектура [1, 23] с одиночным потоком команд и одиночным потоком данных. В случае когда единицей данных в потоке данных является один бит, архитектура относится к последовательным архитектурам (архитектура с поразрядной обработкой данных), иначе (архитектура с пословной обработкой данных) возможно наличие параллельных побитовых операций, например, И, ИЛИ, Исключающее ИЛИ и др.

МКОД (MISD) – архитектура со множественным потоком команд и одиночным потоком данных. В соответствии с изложением М. Дж. Флинна [23] (рисунок 12) к МКОД можно отнести конвейерные архитектуры.

Состояния конвейера (конвейерной архитектуры [32]) можно задать в виде:

$\left\langle \sigma_{t-1}^{1}, \sigma_{t-2}^{2}, ..., \sigma_{t-p+1}^{p-1}, \sigma_{t-p}^{p} \right\rangle,$

$\left\langle \sigma_{t}^{1}, \sigma_{t-1}^{2}, ..., \sigma_{t-p+2}^{p-1}, \sigma_{t-p+1}^{p} \right\rangle,$

…

$\left\langle \sigma_{t+p-1}^{1}, \sigma_{t+p-2}^{2}, ..., \sigma_{t+1}^{p-1}, \sigma_{t}^{p} \right\rangle,$

$\left\langle \sigma_{t+p}^{1}, \sigma_{t+p-1}^{2}, ..., \sigma_{t+2}^{p-1}, \sigma_{t+1}^{p} \right\rangle,$

…

Конвейер выполняет операции из набора $\left\langle \rho_1, \rho_2, ..., \rho_p \right\rangle$, где ρ – количество этапов конвейера, такие, что $\sigma_{t-k-1}^{k+1} = \rho_k\left(\sigma_{t-k}^{k}\right)$. Поток данных берётся в сечении состояния, заданным номером этапа q:

$\sigma_{t-q}^{q}, \sigma_{t-q+1}^{q}, \sigma_{t-q+2}^{q}, ...$

Синхронная работа конвейера – данные от каждого этапа следующему этапу передаются синхронно (одновременно). Коэффициент ускорения такого конвейера (k – ранг задачи):

$$r = \frac{k \cdot \sum_{i=1}^{p} \tau_i}{\sum_{j=1}^{k+p-1} \max_{i=\max(\{j-k+1,1\})}^{\min(\{j,p\})} (\tau_i)}.$$

Асинхронная работа конвейера – данные от каждого этапа следующему (свободному) этапу передаются асинхронно, вне зависимости от передачи данных другими этапами. Коэффициент ускорения такого конвейера

$$r = \frac{k \cdot \sum_{i=1}^{p} \tau_i}{\sum_{i=1}^{p} \tau_i + (k-1) \cdot \max_{i=1}^{p} (\tau_i)}.$$

Несбалансированный конвейер – конвейер, имеющий этапы с различным временем работы τ_i.

Сбалансированный конвейер – конвейер [32], все этапы которого имеют одинаковое время работы τ. Коэффициент ускорения и эффективность такого конвейера (k – ранг задачи):

$$r = \frac{k \cdot p \cdot \tau}{(k + p - 1) \cdot \tau}, \quad e = \frac{k \cdot \tau}{(k + p - 1) \cdot \tau}.$$

Скалярные архитектуры могут использовать конвейер.

ОКМД (SIMD) – архитектура с одиночным потоком команд и множественным потоком данных. Команды в таких архитектурах имеют следующий вид:

$$\kappa = \langle \gamma, \alpha, \beta \rangle,$$

$$\alpha = \langle \alpha_1, \alpha_2, ..., \alpha_n \rangle,$$

$$\beta = \langle \beta_1, \beta_2, ..., \beta_n \rangle.$$

К ОКМД относятся векторные и матричные архитектуры.

МКМД (MIMD) – архитектура с множественным потоком команд и множественным потоком данных.

В классе архитектур МКМД выделяют подклассы архитектур МКМД с управлением от потока данных и с управлением от потока команд.

Архитектуры для мелкогранулярного параллелизма [19, 21]:

- последовательностная архитектура;
- зависимостная архитектура;
- независимостная архитектура (с явным параллелизмом (сверхдлинная команда)).

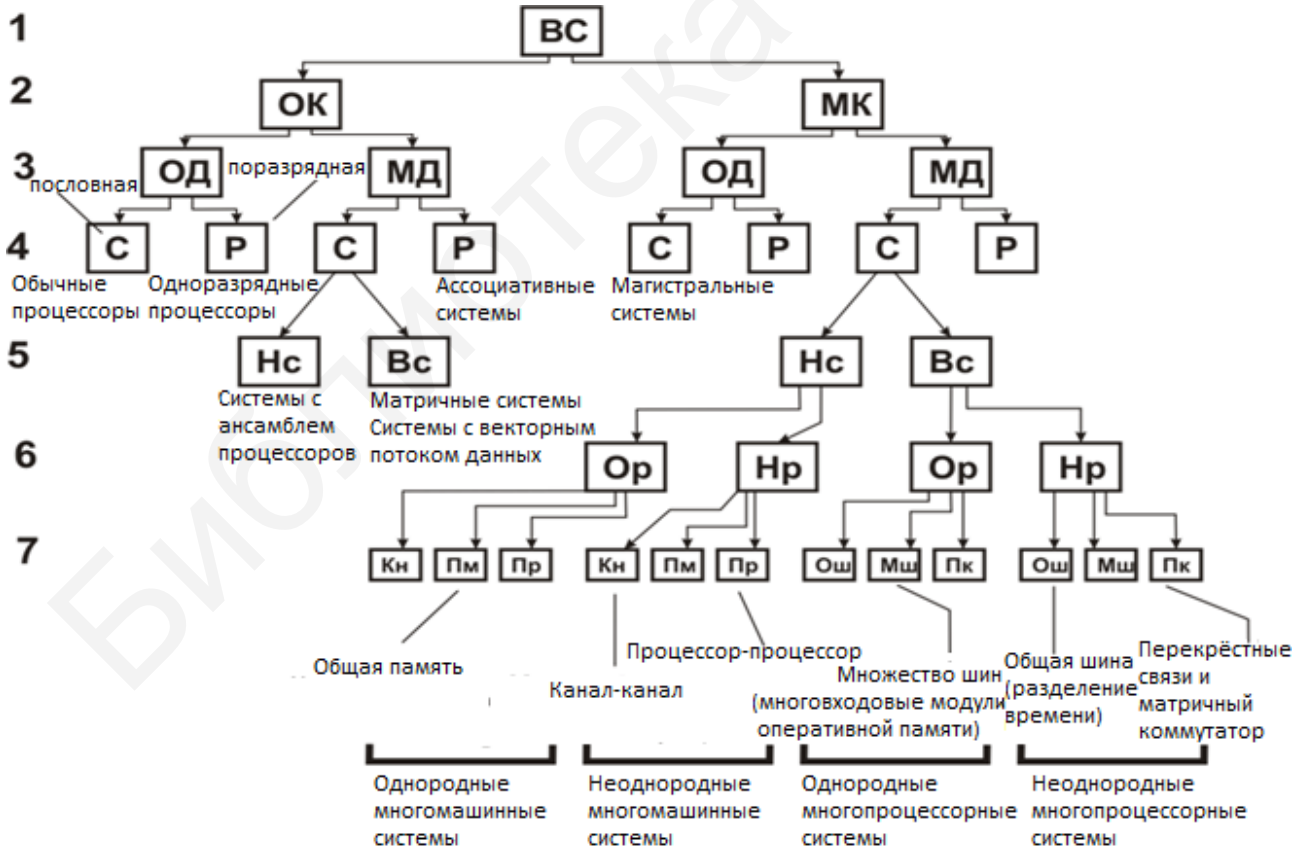


Рисунок 12 – Схема классификации параллельных вычислительных систем [3], расширяющая систематику Флинна

В классе архитектур МКМД выделяют подклассы архитектур МКМД с общей (разделяемой) памятью [1, 18, 19, 20, 21]:

- архитектуры с однородным (равномерным) доступом к памяти (UMA\SMP, UMA\AMP);
- архитектуры с разнородным (унифицированным) доступом к памяти (hUMA);
- архитектуры с неоднородным (неравномерным) доступом к памяти (NUMA)

и МКМД без общей (разделяемой) памяти (с распределённой (индивидуальной) памятью) [1, 18, 19, 20, 21]:

- массивно-параллельная многопроцессорная архитектура (MPP);
- кластерная архитектура – параллельная многомашинная архитектура.

Задача управления

Рассмотрим множество «внутренних» ситуаций (ситуаций модели) M и множество «внешних» ситуаций (проблемной области) Q. Расширим множество ситуаций M множеством входов X и рассмотрим управляющую операцию в виде

$d \in Q_+^{M \times X}$.

Зададим множество состояний множеством расширенных ситуаций $M \times X$ и временной шкалой с носителем T:

$(M \times X) \times T$.

Операцию реакции среды (проблемной области) к заданному моменту времени представим в виде

$r \in 2^{Q \times 2^{T \times M}}$,

в простейшем случае –

$r \in \left(M_+^{\,T}\right)^Q$.

Рассмотрим игровую схему с двумя игроками:

$\left\{\left\langle\left((M \times X) \times T\right) \cup Q, \{d\}\right\rangle, \left\langle Q \cup 2^{T \times M}, \{r\}\right\rangle\right\}$,

тогда можно ввести следующие операции:

$a \in Q_+^{(M \times X) \times T}$; $\quad h \in \left(M_+^{\,T}\right)_+^{M \times X}$; $\quad s \in \left(M_+^{\,T}\right)_+^{(M \times X) \times T}$,

где $h = d \circ r$ и $s = a \circ r$, в простейшем случае такие, что

$a\left(\left\langle\langle m, x\rangle, t\right\rangle\right) = d\left(\langle m, x\rangle\right)$; $\qquad s\left(\left\langle\langle m, x\rangle, t\right\rangle\right) = h\left(\langle m, x\rangle\right)$.

Пусть критерий оптимальности, значения которого характеризуют затраты ресурсов,

$c \in \mathbb{R}_+^{M \times X \times T}$.

Задачу (оптимального) управления [6] можно сформулировать в виде обобщённой задачи следующим образом.

Дано: уравнение состояния

$$g = s\left(\left\langle g(t), u\left(\left\langle g(t), t\right\rangle\right)\right\rangle, t\right)$$

и начальные условия (начальное состояние m_1 и конечное состояние m_0):

$g(t_1) = m_1;\quad g(t_0) = m_0,$

обычно $(t_0 - t_1) \in \mathbb{N}$.

Требуется: определить $u \in X^{M \times T}$, удовлетворяющее уравнению состояния и начальным условиям, а также удовлетворяющее условию

$\arg\min_u \left(\sum_{t=t_1}^{t=t_0} c\left(\left\langle g\left(\left\langle u,t\right\rangle\right), u\left(\left\langle g\left(\left\langle u,t\right\rangle\right),t\right\rangle\right),t\right\rangle\right)\right)$ или

$\arg\min_u \left(\int_{t_1}^{t_0} c\left(\left\langle g(t), u\left(\left\langle g(t),t\right\rangle\right),t\right\rangle\right)dt\right).$

Система, управляемая знаниями. Уровни управления

Системы, управляемые знаниями, (СУЗ) относятся к интеллектуальным системам [12]. В СУЗ можно выделить три уровня управления [11]:

- уровень управления устройствами;
- уровень управления данными;
- уровень управления знаниями.

Задания для лабораторных работ

Задача вычисления попарного произведения (деления) компонентов двух векторов чисел.

Дано: сгенерированные два вектора ***A*** и ***B*** заданной длины ***m*** каждый, элементы которых являются положительными числами заданной разрядности ***p***.

Получить: вектор значений операции [18] целочисленного произведения (деления) для каждой пары чисел, имеющий длину ***m*** и разрядность компонентов 2***p***.

***Варианты*:**

1) алгоритм вычисления произведения пары 4-разрядных чисел умножением с младших разрядов со сдвигом множимого (частичного произведения) влево;
2) алгоритм вычисления произведения пары 4-разрядных чисел умножением с младших разрядов со сдвигом частичной суммы вправо;
3) алгоритм вычисления произведения пары 4-разрядных чисел умножением со старших разрядов со сдвигом множимого (частичного произведения) вправо;
4) алгоритм вычисления произведения пары 4-разрядных чисел умножением со старших разрядов со сдвигом частичной суммы влево;
5) алгоритм вычисления целочисленного частного пары 4-разрядных чисел делением с восстановлением частичного остатка;
6) алгоритм вычисления целочисленного частного пары 4-разрядных чисел делением без восстановления частичного остатка;

7) алгоритм вычисления произведения пары 6-разрядных чисел умножением с младших разрядов со сдвигом множимого (частичного произведения) влево;
8) алгоритм вычисления произведения пары 6-разрядных чисел умножением с младших разрядов со сдвигом частичной суммы вправо;
9) алгоритм вычисления произведения пары 6-разрядных чисел умножением со старших разрядов со сдвигом множимого (частичного произведения) вправо;
10) алгоритм вычисления произведения пары 6-разрядных чисел умножением со старших разрядов со сдвигом частичной суммы влево;
11) алгоритм вычисления целочисленного частного пары 6-разрядных чисел делением с восстановлением частичного остатка;
12) алгоритм вычисления целочисленного частного пары 6-разрядных чисел делением без восстановления частичного остатка;
13) алгоритм вычисления произведения пары 8-разрядных чисел умножением с младших разрядов со сдвигом множимого (частичного произведения) влево;
14) алгоритм вычисления произведения пары 8-разрядных чисел умножением с младших разрядов со сдвигом частичной суммы вправо;
15) алгоритм вычисления произведения пары 8-разрядных чисел умножением со старших разрядов со сдвигом множимого (частичного произведения) вправо;
16) алгоритм вычисления произведения пары 8-разрядных чисел умножением со старших разрядов со сдвигом частичной суммы влево;
17) алгоритм вычисления целочисленного частного пары 8-разрядных чисел делением с восстановлением частичного остатка;
18) алгоритм вычисления целочисленного частного пары 8-разрядных чисел делением без восстановления частичного остатка;
19) алгоритм вычисления целочисленного обращения элемента в конечном поле вычетов (257):
$a2=a*a \bmod 257$
$a4=a2*a2 \bmod 257$
$a8=a4*a4 \bmod 257$
$a9=a8*a \bmod 257$
$a17=a9*a8 \bmod 257$
$a34=a17*a17 \bmod 257$
$a51=a34*a17 \bmod 257$
$a85=a51*a34 \bmod 257$
$a170=a85*a85 \bmod 257$
$a255=a170*a85 \bmod 257$;
20) алгоритм вычисления целочисленного обращения элемента в конечном поле вычетов (17):
21) $a2=a*a \bmod 17$
$a3=a2*a \bmod 17$

$a5=a3*a2 \bmod 17$
$a10=a5*a5 \bmod 17$
$a15=a10*a5 \bmod 17$.

Вопросы

1. Как называется счётное или конечное неориентированное множество символов?
2. Как называется множество информационных конструкций, текстов, представляемых в виде строк, состоящих из символов, принадлежащих некоторому алфавиту?
3. Как называется множество правил, состоящих из условия и следствия, характеризующее синтаксические отношения фрагментов текстов формального языка?
4. Как называется отношение между описанием исходной ситуации и описанием множества целевых ситуаций?
5. Как называется граф, вершины которого соответствуют состояниям, а рёбра – операциям перехода между ними?
6. Как называется бинарное отношение на множестве состояний?
7. Как называется множество состояний, заданное множеством операций, определённых на этом множестве состояний, и множеством начальных состояний, являющихся текстами формального языка с заданными грамматикой и синтаксисом?
8. Чем является формальная модель обработки информации, транзитивное замыкание объединения всех операций которой является симметричным бинарным отношением?
9. Как называется операция, обратная к которой является детерминированной операцией?
10. Как называется операция, область определения которой является декартовой степенью её области прибытия?
11. Чему равно количество недетерминированных операций на множестве состояний мощностью n?
12. Каково количество обратимых операций на множестве состояний мощностью n?
13. Каково количество полностью определённых операций на множестве состояний мощностью n?
14. Каково количество полностью определённых детерминированных операций на множестве состояний мощностью n?
15. Каково количество полностью определённых недетерминированных операций на множестве состояний мощностью n?
16. Каково количество детерминированных обратимых операций на множестве состояний мощностью n?
17. Каково количество задач на множестве состояний мощностью n?
18. Чему равно количество различных индивидных задач на множестве состояний мощностью n?

19. Как называется задача с единственным исходным состоянием?
20. Как называется задача с двумя и более исходными состояниями?
21. Что задаёт множество, единственным элементом которого является маршрут из начального состояния в одно из целевых?
22. Как называется последовательность операций, задающих пути из всех исходных состояний задачи в целевые?
23. Каковы недостатки поиска решения задачи в глубину?
24. Каковы недостатки поиска решения задачи в ширину?
25. Какова мощность результата замыкания Клини конечного множества?
26. Какова мощность результата замыкания Клини счётного множества?
27. Как называется императивная информационная конструкция, задающая или описывающая операцию?
28. Если выходы предыдущей команды являются входами следующей в потоке команд, то как называется зависимость между командами?
29. Если входы предыдущей команды являются выходами следующей в потоке команд, то как называется зависимость между командами?
30. Если выходы одной команды являются выходами следующей в потоке команд, то имеют ли эти команды зависимость? Если имеют, то какого вида?

2 Задачи управления памятью, модели памяти и механизмы синхронизации

К задачам этого уровня относятся: управление памятью, включая её распределение/перераспределение, управление доступом и синхронизация.

Механизмы синхронизации в вычислительных системах

Атомарные операции

Атомарная операция – операция, события процесса выполнения которой наблюдаются всеми процессами системы одинаково и неделимо (однократно).

Общая когерентная память

Память называется когерентной, если и только если для каждой её ячейки для всех процессов в системе существует единая последовательность наблюдений одних и тех же событий записи чтения в эту ячейку, т. е. все операции чтения-записи в эту ячейку атомарны.

Семафоры

Семафор – программный механизм [20], обеспечивающий хранение значений целочисленной переменной и поддерживающий интерфейс операций захвата и высвобождения.

События

События – программный механизм, обеспечивающий хранение значений логической переменной и поддерживающий интерфейс операций ожидания и установки/снятия.

Сигналы и обработчики событий

Обработка сигналов – программный механизм, механизм делегирования, обеспечивающий программную связь между процедурой возбуждения сигнала, процедурой обработки (обратного вызова) и экземпляром типа данных сигнала.

Логическое время

Время задаётся с помощью временной шкалы. *Временная шкала* – шкала χ, отображающая упорядоченное множество событий $\langle M, \geq_M \rangle$ на линейно упорядоченное множество $\langle T, \geq_T \rangle$:

$$\chi \cong \langle \langle M, \geq_M \rangle, \langle T, \geq_T \rangle, \tau \rangle$$

$$\tau \in T^M$$

$$\pi \leq_M \sigma \to \tau(\pi) \leq_T \tau(\sigma).$$

События упорядочены отношением «потенциально причинно-связанны».

1) Два события связаны этим отношением, если:
   а) они выполнены одним процессом, первое выполнено до второго;
   б) одно из них есть событие посылки сообщения одним процессом, а второе – событие получения этого сообщения другим процессом.
2) Отношение рефлексивно.

3) Отношение транзитивно.

Алгоритм часов Лэмпорта $\langle ti, di \rangle$:

1) если событие является событием получения сообщения с меткой tm, то $ti \leftarrow \max\left(\langle tm, ti \rangle\right) + di$;
2) иначе $ti \leftarrow ti + di$;
3) если событие является событием отправки, то отправить сообщение с меткой $tm = ti$;
4) для события вернуть метку ti.

Модели параллельного доступа к памяти

Операции чтения-записи

Операция последовательного чтения-записи – операция изменяет не более одного элемента данных (элемента текста).

Операция параллельного чтения-записи – операция изменяет более одного элемента данных (элемента текста).

Детерминированность операций чтения-записи: по чтению (недетерминированная позиция чтения), по записи (недетерминированная позиция записи).

Модель EREW

Читать-писать в одну ячейку и читать из одной ячейки может только один процесс [36].

Если операция записывает значения n ячеек, то их значения равны значениям, прочитанным из n ячеек. Операции чтения-записи детерминированы (по чтению).

Модель CREW

Писать в одну ячейку может только один процесс [36]. Читать из одной ячейки могут несколько процессов.

Если операция записывает значения n ячеек, то их значения равны значениям, прочитанным из не более чем n ячеек. Операции чтения-записи детерминированы (по чтению).

Модель CRCW

Писать в одну ячейку и читать из одной ячейки могут несколько процессов.

Будем рассматривать вариант модели CRCW [36], который будем называть hard-CRCW. К варианту hard-CRCW относится вариант arbitrary-CRCW. Варианты модели CRCW weak-CRCW, common-CRCW, priority-CRCW, strong-CRCW будем относить к варианту soft-CRCW, который можно задать модификацией модели CREW, дополненной операцией чтения-записи из адреса и операцией параллельного (редукционного) вычисления такого адреса из множества входов.

Вариант модели hard-CRCW отличается операцией, позволяющей писать в одну или несколько ячеек значения, вычисленные операцией, равнозначной (параллельному) применению операции редукции к значениям, прочитанным из (под)множеств (множества) входов. Записываемое значение должно совпадать с одним из значений, прочитанным из множества входов.

Если операция записывает значения n ячеек, то их значения равны значениям, прочитанным из не более чем n ячеек. Операции чтения-записи не детерминированы (по чтению).

Модели согласованности памяти

Модель консистентности (согласованности) [35] определяет порядок событий или состояний памяти вычислительной системы, наблюдаемых процессами внутри и вне системы.

Строгая консистентность

Для всех процессов в системе наблюдение состояний памяти в одинаковые моменты времени одинаково (таблицы 3 и 4).

Таблица 3 – Пример наблюдения поведения системы в модели строгой консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	*m*[0]:=1	*m*[1]:=2
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	0=*m*[0]	0=*m*[2]	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	0=*m*[1]	1=*m*[0]
4	0=*m*[0]	0=*m*[1]	0=*m*[2]	0=*m*[1]	–	*m*[2]:=1

Таблица 4 – Пример наблюдения поведения системы вне модели строгой консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	**0=*m*[0]**	**0=*m*[1]**	0=*m*[2]	–	***m*[0]:=1**	***m*[1]:=2**
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	**1=*m*[0]**	**1=*m*[2]**	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	**2=*m*[1]**	1=*m*[0]
4	0=*m*[0]	0=*m*[1]	**0=*m*[2]**	**2=*m*[1]**	–	***m*[2]:=1**

Последовательная консистентность

Для всех процессов в системе существует последовательность наблюдений одних и тех же состояний памяти каждым из процессов (таблица 5).

Таблица 5 – Пример наблюдения поведения системы в модели последовательной консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	*m*[0]:=1	*m*[1]:=2
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	1=*m*[0]	1=*m*[2]	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	2=*m*[1]	1=*m*[0]
4	0=*m*[0]	0=*m*[1]	0=*m*[2]	2=*m*[1]	–	*m*[2]:=1

Причинная консистентность

Для всех процессов в системе существует последовательность наблюдений одних и тех же потенциально причинно-связанных событий каждым из процессов (таблица 6).

Таблица 6 – Пример наблюдения поведения системы в модели причинной консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	**0=*m*[0]**	0=*m*[1]	0=*m*[2]	–	*m*[1]:=2	***m*[0]:=1**
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	**1=*m*[0]**	**0=*m*[2]**	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	**1=*m*[2]**	**0=*m*[0]**
4	0=*m*[0]	0=*m*[1]	**0=*m*[2]**	2=*m*[1]	–	***m*[2]:=1**

PRAM-консистентность

Для каждого из процессов для всех процессов в системе существует последовательность наблюдений одних и тех же событий чтения-записи этим процессом в память (таблицы 7 и 8).

Таблица 7 – Пример наблюдения поведения системы в модели PRAM-консистентности (конвейеризированного ОЗУ)

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	**0=*m*[0]**	**0=*m*[1]**	0=*m*[2]	–	***m*[0]:=1**	***m*[1]:=2**
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	**1=*m*[1]**	**2=*m*[1]**	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	1=*m*[1]	0=*m*[0]
4	0=*m*[0]	**0=*m*[1]**	0=*m*[2]	**2=*m*[1]**	–	***m*[1]:=1**

Таблица 8 – Пример наблюдения поведения системы вне модели PRAM-консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	**0=*m*[0]**	**0=*m*[1]**	0=*m*[2]	–	***m*[0]:=1**	***m*[1]:=2**
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	1=*m*[1]	2=*m*[1]	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	**2=*m*[1]**	**0=*m*[0]**
4	0=*m*[0]	0=*m*[1]	0=*m*[2]	2=*m*[1]	–	*m*[1]:=1

Процессорная консистентность

Процессорная консистентность требует также, чтобы для каждой ячейки памяти для всех процессов в системе существовала последовательность наблюдений одних и тех же событий чтения-записи в эту ячейку (таблица 9).

Таблица 9 – Пример наблюдения поведения системы в модели процессорной консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	0=*m*[0]	**0=*m*[1]**	0=*m*[2]	–	*m*[0]:=1	***m*[1]:=2**
2	0=*m*[0]	0=*m*[1]	0=*m*[2]	**1=*m*[1]**	**2=*m*[1]**	–
3	0=*m*[0]	0=*m*[1]	0=*m*[2]	–	1=*m*[1]	0=*m*[0]
4	0=*m*[0]	**0=*m*[1]**	0=*m*[2]	**2=*m*[1]**	–	***m*[1]:=1**

Слабая консистентность

Модель слабой консистентности (таблицы 10 и 11) основана на выделении среди переменных специальных синхронизационных переменных и описывается следующими правилами:

а) доступ к синхронизационным переменным определяется моделью последовательной консистентности;

б) доступ к синхронизационным переменным запрещён (задерживается), пока не выполнены все предыдущие операции записи;

в) доступ к данным (запись, чтение) запрещён, пока не выполнены все предыдущие обращения к синхронизационным переменным.

Таблица 10 – Пример наблюдения поведения системы в модели слабой консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	0=*m*[0]	*S*	*m*[0]:=1	*S*	1=*m*[0]	–
2	0=*m*[0]	*S*	–	*S*	1=*m*[0]	–
3	0=*m*[0]	*S*	–	–	0=*m*[0]	*S*
4	0=*m*[0]	*S*	–	–	1=*m*[0]	*S*

Таблица 11 – Пример наблюдения поведения системы вне модели слабой консистентности

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	0=*m*[0]	*S*	*m*[0]:=1	*S*	1=*m*[0]	–
2	0=*m*[0]	*S*	–	*S*	**0=*m*[0]**	–
3	0=*m*[0]	*S*	–	–	0=*m*[0]	*S*
4	0=*m*[0]	*S*	–	–	1=*m*[0]	*S*

Консистентность по выходу

Модель консистентности по выходу (таблицы 12 и 13) описывается следующими правилами:

а) до выполнения обращения к общей переменной должны быть полностью выполнены все предыдущие захваты синхронизационных переменных данным процессором;

б) перед освобождением синхронизационной переменной должны быть закончены все операции чтения-записи, выполнявшиеся процессором прежде;

в) реализация операций захвата и освобождения синхронизационной переменной должна удовлетворять требованиям процессорной консистентности (последовательная консистентность не требуется).

Таблица 12 – Пример наблюдения поведения системы в модели консистентности по выходу

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	Acq(*L*)	*m*[1]:=1	*m*[1]:=2	Rel(*L*)	–	–
2	–	–	–	Acq(*L*)	2=*m*[1]	Rel(*L*)
3	–	–	–	–	–	1=*m*[1]
4	–	–	–	–	2=*m*[1]	–

Таблица 13 – Пример наблюдения поведения системы вне модели консистентности по выходу

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	Acq(*L*)	*m*[1]:=1	*m*[1]:=2	Rel(*L*)	–	–
2	–	–	2=*m*[1]	Acq(*L*)	1=*m*[1]	Rel(*L*)
3	–	–	–	–	–	1=*m*[1]
4	–	–	–	–	2=*m*[1]	–

Консистентность по входу

Модель консистентности по входу (таблицы 14 и 15) описывается следующими правилами:

а) процесс не может захватить синхронизационную переменную до того, пока не обновлены все переменные этого процесса, охраняемые захватываемой синхронизационной переменной;

б) процесс не может захватить синхронизационную переменную в монопольном режиме (для модификации охраняемых данных), пока другой процесс, владеющий этой переменной (даже в немонопольном режиме), не освободит её;

в) если какой-то процесс захватил синхронизационную переменную в монопольном режиме, то ни один процесс не сможет ее захватить даже в немонопольном режиме до тех пор, пока первый процесс не освободит эту переменную и будут обновлены текущие значения охраняемых переменных в процессе, запрашивающем синхронизационную переменную.

Таблица 14 – Пример наблюдения поведения системы в модели консистентности по входу

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	Acq(0)	*m*[0]:=1	Acq(1)	*m*[1]:=1	Rel(0)	Rel(1)
2	–	–	–	Acq(0)	1=*m*[0]	0=*m*[1]
3	–	–	–	–	Acq(1)	1=*m*[1]
4	–	–	–	–	0=*m*[0]	–

Таблица 15 – Пример наблюдения поведения системы вне модели консистентности по входу

	*t*1	*t*2	*t*3	*t*4	*t*5	*t*6
1	Acq(0)	*m*[0]:=1	Acq(1)	*m*[1]:=1	Rel(0)	Rel(1)
2	–	–	–	Acq(0)	0=*m*[0]	1=*m*[1]
3	–	–	–	–	Acq(1)	0=*m*[1]
4	–	–	–	–	1=*m*[0]	–

Распределение памяти

Линейное адресное пространство

Рассмотрим множество (имён) адресов A (не более чем счётной мощности), которые обычно задаются как

$A \subseteq \{0,1\}^n$,

а также рассмотрим метрику d над парами элементов A, которая вместе с A образует метрическое пространство:

$\langle A, \{d\} \rangle;\ d \in D^{A \times A}$,

где D – множество чисел (длин), для которого определена группа:

$\langle D, \{+_D\} \rangle$

$0_D \in D$

$+_D \in D^{D \times D}$

$a +_D 0_D = a$

$a +_D b = b +_D a$

$(a +_D b) +_D c = a +_D (b +_D c)$

и линейное (полное) отношение нестрогого порядка:

$\geq_D \in D \times D$

$a \geq_D a;\quad (a \geq_D b) \vee (b \geq_D a)$

$((a \geq_D b) \wedge (b \geq_D a)) \rightarrow (a = b +_D 0_D)$

$((a \geq_D b) \vee (b \geq_D c)) \rightarrow (a \geq_D c)$,

образующие порядковую метрическую шкалу, в которой метрика обладает следующими свойствами:

$0_D \geq_D d(\langle a,a \rangle)$

$d(\langle a,b \rangle) \geq_D 0_D$

$d(\langle a,b \rangle) \geq_D d(\langle b,a \rangle)$

$d(\langle a,b \rangle) +_D d(\langle b,c \rangle) \geq_D d(\langle a,c \rangle)$.

В рамках образованной шкалы можно индуцировать порядок $\geq_D$ с точностью до обращения в порядок $\geq_A$ на множестве адресов:

$\geq_A \in A \times A$

$a \geq_A a;\ \ (a \geq_A b) \vee (b \geq_A a)$

$$\left(\left(a \geq_A b\right) \wedge \left(b \geq_A a\right)\right) \to \left(0_D \geq_D d\left(\langle a,b \rangle\right)\right)$$
$$\left(\left(a \geq_A b\right) \vee \left(b \geq_A c\right)\right) \to \left(a \geq_A c\right)$$
$$\left(d\left(\langle a,c \rangle\right) \geq_D \left(d\left(\langle a,b \rangle\right) +_D d\left(\langle b,c \rangle\right)\right)\right) \to \left(\left(a \geq_A b\right) \sim \left(b \geq_A c\right)\right),$$

а также ввести адресные операции – операцию увеличения адреса, рассматриваемую как действие группы длин:

$$+_{AD} \in A^{A \times D}$$
$$a +_{AD} 0_D = a$$
$$\left(a +_{AD} c\right) +_{AD} b = a +_{AD} \left(c +_D b\right)$$
$$\left(b \geq_A a\right) \to \left(a +_{AD} d\left(\langle a,b \rangle\right) = b\right)$$

и операцию уменьшения адреса:

$$-_{AD} \in A^{A \times D}$$
$$a -_{AD} 0_D = a$$
$$\left(a -_{AD} c\right) -_{AD} b = a -_{AD} \left(c +_D b\right)$$
$$\left(a -_{AD} c\right) +_{AD} c = a$$
$$\left(a +_{AD} c\right) -_{AD} c = a$$
$$\left(a \geq_A b\right) \to \left(a -_A d\left(\langle a,b \rangle\right) = b\right),$$

через которые задаётся операция сложения адресов:

$$+_A \in A^{A \times A}$$
$$0_A \in A$$
$$\left(b \geq_A a\right) \to \left(a +_A \left(0_A +_{AD} d\left(\langle a,b \rangle\right)\right) = b\right)$$
$$\left(a \geq_A b\right) \to \left(a +_A \left(0_A -_{AD} d\left(\langle a,b \rangle\right)\right) = b\right)$$
$$a +_A 0_A = a.$$

Таким образом, линейное адресное пространство может быть задано как $\left\langle A \cup D, \left\{+_A, +_{AD}, -_{AD}, d, \geq_A\right\}\right\rangle$.

Память содержит множество ячеек мощности $|A|$ (каждой ячейке взаимнооднозначно сопоставлен адрес), множество её состояний можно представить как $\left(\{0,1\}^n\right)^{|A|}$. Каждая ячейка может быть занятой либо свободной. Соседние ячейки, одновременно являющиеся свободными или занятыми, образуют соответственно свободные или занятые участки памяти. Размером участка является количество (образующих) ячеек в нём.

Базовыми задачами распределения памяти [11] являются:

- выделение свободного участка памяти;
- высвобождение занятого участка памяти.

Задача выделения свободного участка памяти

Дано: состояние памяти, размер участка памяти.

Требуется: найти свободный участок памяти требуемого размера и снять пометку его ячеек как ячеек свободного участка памяти, в случае успеха вернуть новое состояние памяти и младший адрес ячейки выделенного участка памяти, иначе вернуть пустой адрес.

Задача высвобождения занятого участка памяти

Дано: состояние памяти, размер участка памяти, адрес младшей ячейки занятого участка.

Требуется: пометить ячейки занятого участка как свободные, начиная с данного адреса, в количестве, равном данному размеру, объединяя их пометку с соседними свободными, если таковые есть, вернуть количество помеченных ячеек и новое состояние памяти.

При этом с целью управления необходимо отслеживать количество требуемых ресурсов при каждом решении этих задач. Одним из ресурсов является время, а вторым ресурсом – объём используемой памяти. Затраты этих ресурсов зависят от машины, соответствующей ей формальной модели обработки информации и выбора метода решения поставленных задач, включая модель представления свободных участков памяти, которая может быть задана представлением этих участков с целью хранения в виде:

- списка;
- раздельных списков;
- деревьев.

Отдельной трудностью при снижении затрат на используемый объём памяти с целью сократить её перерасход является выбор стратегии распределения памяти, обеспечивающей сокращение фрагментации, которая может быть внутренней или внешней. К стратегиям распределения памяти относятся:

- «первый подходящий»;
- «наилучший подходящий»;
- «следующий подходящий»;
- «наихудший подходящий»;
- «системы близнецов» и др.

Конечное линейное адресное пространство

Для регистровой машины с произвольным адресным доступом в конечном линейном адресном пространстве ($|A| \in \mathbb{N}$) алгоритмы, которые используют деревья для представления свободных участков памяти, хранящиеся в них самих, и реализуют стратегию «первый подходящий», обеспечивают следующие гарантии.

Время решения задач выделения и высвобождения не превышает с точностью до константы квадратного логарифма (полилогарифма) от размера адресного пространства, умноженного на время доступа к одной произвольной ячейке. Временная сложность решения этих задач может быть выражена

$O\left(\log^2\left(|A|\right)\cdot f\left(|A|\right)\right)$. При этом (дополнительная) пространственная (ёмкостная) сложность решения этих задач выражена (при измерении в битах необходимой дополнительной памяти) как $O\left(\log\left(|A|\right)\right)$, а пространственная (полная) сложность, которая включает количество (бит) памяти на запись результата, – как $O\left(\log^3\left(|A|\right)\right)$. Ограничения на значения коэффициента перерасхода памяти (из-за внешней фрагментации), значение которого близко к оптимальному, так как используется стратегия «первый подходящий», выражаются:

$$B_i(t) = k \cdot b_i(t)$$

$$m(t) = \sum_i b_i(t)$$

$$M(t) = \sum_i B_i(t) = k \cdot m(t)$$

$$M_{\max} = k \cdot m_{\max} = \max_t\left(M(t)\right)$$

$$b_{\max} = \max_{\langle t,i\rangle}\left(b_i(t)\right) \le m_{\max}$$

$$M_{\max} \cdot k_{OVERHEAD} \le M_{\max} \cdot \log_2\left(C \cdot b_{\max}\right) = k \cdot m_{\max} \cdot \log_2\left(C \cdot b_{\max}\right)$$

$$M_{\max} \cdot k_{OVERHEAD} \le k \cdot m_{\max} \cdot \log_2\left(C \cdot b_{\max}\right) + allocator\_data\_size.$$

Здесь $B_i(t)$ – размер выделяемого i-го блока памяти в момент времени t, k – коэффициент кратности размера блоков, $M(t)$ – суммарный размер выделенных блоков (занятых участков памяти) в момент времени t, $M_{\max}$ – размер максимального выделенного блока, C – константа ($1<C<2$), $allocator\_data\_size$ – размер дополнительных данных подсистемы распределения памяти ($8\cdot n$), $k_{OVERHEAD}$ – коэффициент перерасхода памяти. Для конечного линейного адресного пространства коэффициент перерасхода памяти может быть вычислен как: $k_{OVERHEAD} = \dfrac{|A| - \max_{x\in free}\left(|x|\right)}{\sum_{x\in used}\left(|x|\right)}$, где $free$ – семейство множеств (адресов) ячеек свободных участков памяти в текущий момент времени, а $used$ – семейство множеств ячеек занятых участков памяти в текущий момент времени.

Перерасход памяти из-за внутренней фрагментации не превышает количество разрядов (бит) в одной ячейке.

Бесконечное линейное адресное пространство

Для регистровой машины с произвольным адресным доступом в бесконечном (неограниченном) линейном адресном пространстве ($|A| \in \mathbb{N}$) алгоритмы [9], которые используют деревья систем-близнецов для представления свободных участков памяти, хранящиеся в них самих, и реализуют стратегию «первый подходящий», обеспечивают следующие гарантии.

Время решения задач выделения и высвобождения не превышает с точностью до константы квадратного логарифма (полилогарифма) от размера адрес-

ного пространства, умноженного на время доступа к одной произвольной ячейке. Временная сложность решения этих задач может быть выражена $O\left(\log^4\left(|A|\right)\cdot f\left(|A|\right)\right)$. При этом (дополнительная) пространственная (ёмкостная) сложность решения этих задач выражена (при измерении в битах необходимой дополнительной памяти) как $O\left(\log\left(|A|\right)\right)$, а пространственная (полная) сложность, которая включает количество (бит) памяти на запись результата, – как $O\left(\log^5\left(|A|\right)\right)$. Перерасход памяти не более чем в два раза выше, чем в случае конечного линейного адресного пространства.

Перераспределение памяти

Задача перераспределения памяти [11].
Дано: состояние памяти, размер занятого массивом данных участка памяти, адрес младшей ячейки и требуемый размер занятого участка памяти.
Требуется: занять участок памяти требуемого размера, при этом, если необходимо, найти свободный участок, если младшие адреса ячеек участков отличаются, то скопировать в новый участок соответствующее количество данных массива из прежнего участка, вернуть новое состояние памяти и младший адрес ячейки занятого участка памяти, иначе вернуть пустой адрес.

В случае необходимости копирования данных массива, последовательное копирование требует временной сложности $O\left(n\cdot f\left(n\right)\right)$, где n – количество элементов массива. Время может быть значительно сокращено только в случае параллельного копирования, так как массив может быть преобразован как строка с помощью операций разбиения и конкатенации. В случае параллельного доступа к памяти эти операции сводимы к операциям обеспечения когерентности кэш-памяти, выполняемым за пренебрежимо малое время.

Амортизированное время может быть оценено зависимостью $O\left(\log^2\left(|A|\right)\cdot f\left(|A|\right)\right)$ (для конечного адресного пространства) [11]. Такая гарантия обеспечивается стратегией отложенного перераспределения памяти с постоянным относительным запасом и с эффектом гистерезиса [11]. Однако это увеличивает затраты памяти, как правило, до двух раз. В этом случае пространственная (ёмкостная) сложность $O\left(\log^3\left(|A|\right)+n\right)$ ($n\le|A|$).

Распределённые системы

Задача <u>взаимного исключения</u> [20].
Дано: множество процессов исполнения программ.
Требуется:

- в любой момент времени только один процесс может находиться внутри критического интервала (особого участка кода программы);

- если ни один процесс не находится в критическом интервале, то любой процесс, желающий войти в критический интервал, должен получить разрешение без какой-либо задержки;
- ни один процесс не должен бесконечно долго ждать разрешения на вход в критический интервал (если ни один процесс не будет находиться внутри критического интервала бесконечно долго);
- корректность работы механизмов взаимного исключения не должна полагаться ни на какие предположения о скоростях работы процессоров.

Алгоритмы взаимного исключения в распределённых системах [26, 30]:
- круговой маркерный;
- древовидный маркерный;
- широковещательный на основе временных меток.

Общая распределённая память [26, 30]. Алгоритмы реализации:
- централизованный алгоритм;
- миграционный алгоритм;
- алгоритм «размножение по чтению»;
- алгоритм «размножение по чтению-записи» («полного размножения»).

Вопросы

1. Какой программный объект задаётся целочисленной переменной и процедурами освобождения и захвата?
2. Какой программный объект задаётся логической переменной и процедурами установления, снятия и ожидания?
3. На использовании чего основан алгоритм логического времени Лэмпорта для распределённых систем?
4. Как называется модель консистентности, в которой все процессы получают результаты всех операций чтения-записи в память, как если бы они выполнялись в некоторой последовательности?
5. Как называется модель консистентности, в которой все процессы получают результаты всех операций чтения-записи без какой-либо задержки по времени в одной и той же последовательности, в которой их видят внешние наблюдатели?
6. Как называется модель консистентности, в которой все процессы получают результаты потенциально зависимых операций чтения-записи в одной и той же последовательности?
7. Как называется модель консистентности, в которой все процессы получают результаты операции записи в порядке, не нарушающем последовательность их выполнения по программе каждого процесса, а получение результатов операций записи в одну и ту же ячейку удовлетворяет модели последовательной консистентности?
8. Как называется модель консистентности, в которой присутствуют операции обращения к синхронизационным переменным?
9. Какие модели консистентности строже модели причинной консистентности?
10. Как называется операция, выполняющаяся как единое целое либо вообще не выполняющаяся?

3 Задачи уровня управления данными и операции обработки данных

К задачам этого уровня относятся: обработка данных в структурах данных, включая строки и списки, и управление ими.

Операция редукции

Операция редукции – бинарная ассоциативная операция ($\bullet$). Свойство *ассоциативности* наряду со свойством *степенной ассоциативности* позволяет применять **параллельную редукцию** за счёт преобразований [2] (рисунки 13 и 14):

$$\left(\left((a \bullet b) \bullet c\right) \bullet d\right) = \left((a \bullet b) \bullet (c \bullet d)\right); \left(\left((a \bullet a) \bullet a\right) \bullet a\right) = \left((a \bullet a) \bullet (a \bullet a)\right).$$

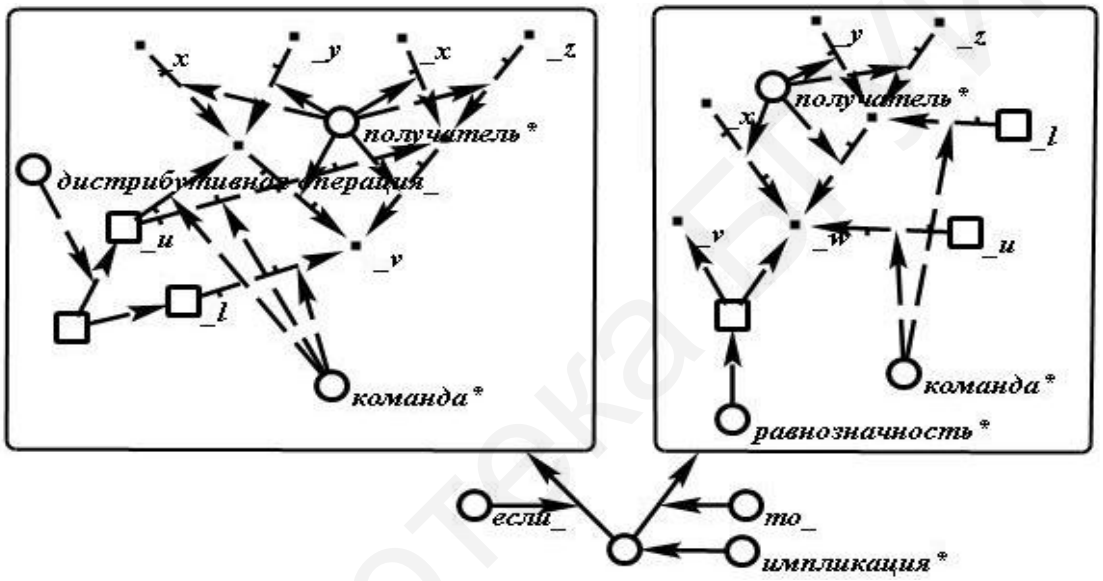


Рисунок 13 – Схема правила преобразования информационного графа с командой дистрибутивной операции

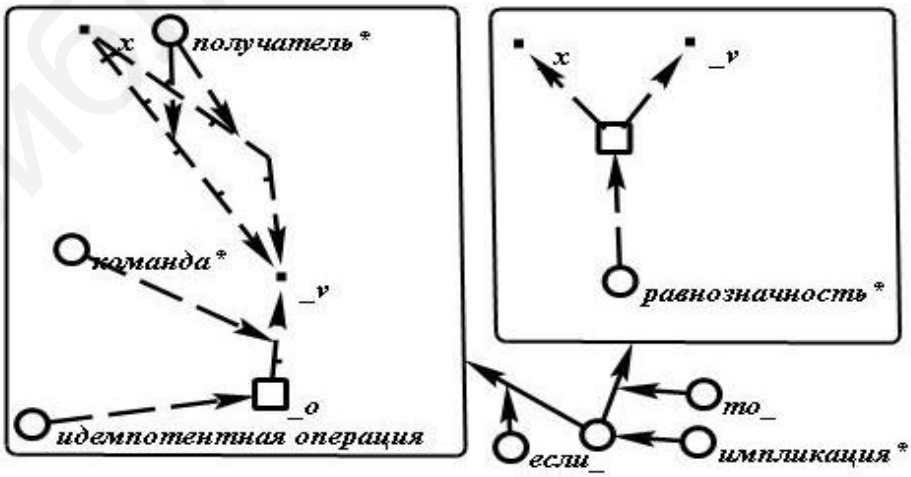


Рисунок 14 – Схема правила преобразования информационного графа с командой идемпотентной операции

Другими преобразованиями, сокращающими временную вычислительную сложность, являются преобразования команд идемпотентных операций и команд операций, обладающих свойством дистрибутивности (рисунки 15 и 16):
$(a \bullet a) = a$; $((a \bullet b) \circ (a \bullet c)) = (a \bullet (b \circ c))$.

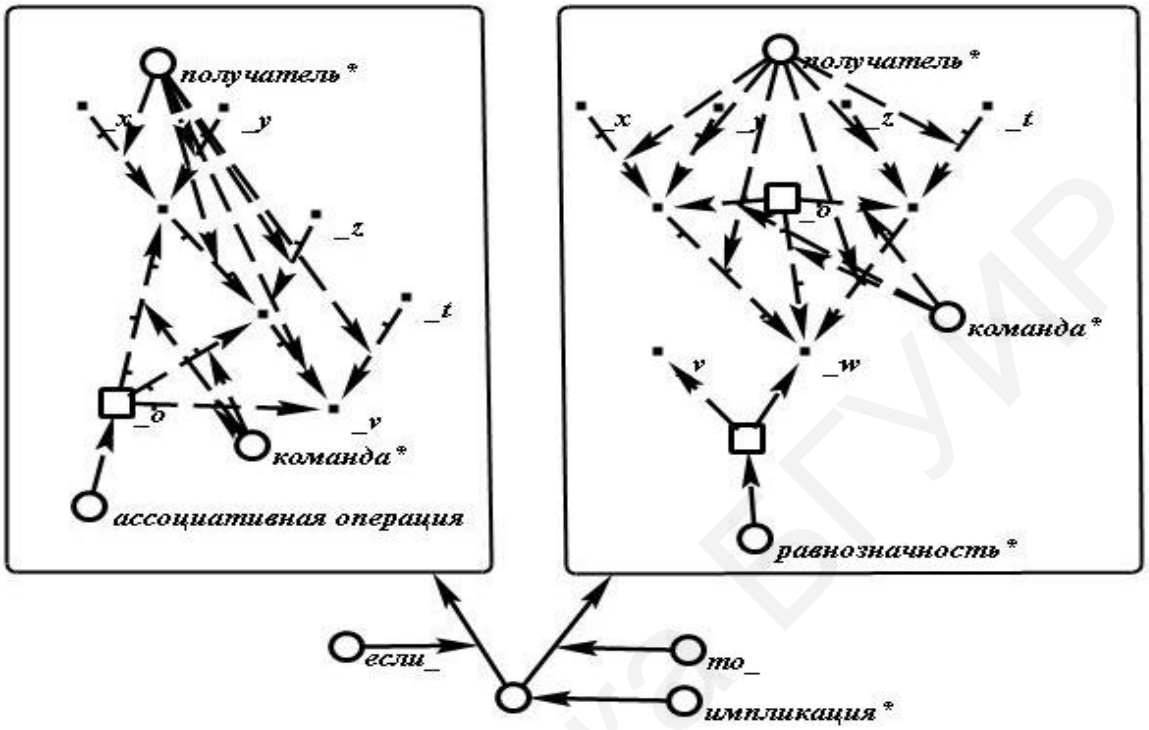


Рисунок 15 – Схема правила преобразования информационного графа с командой ассоциативной операции

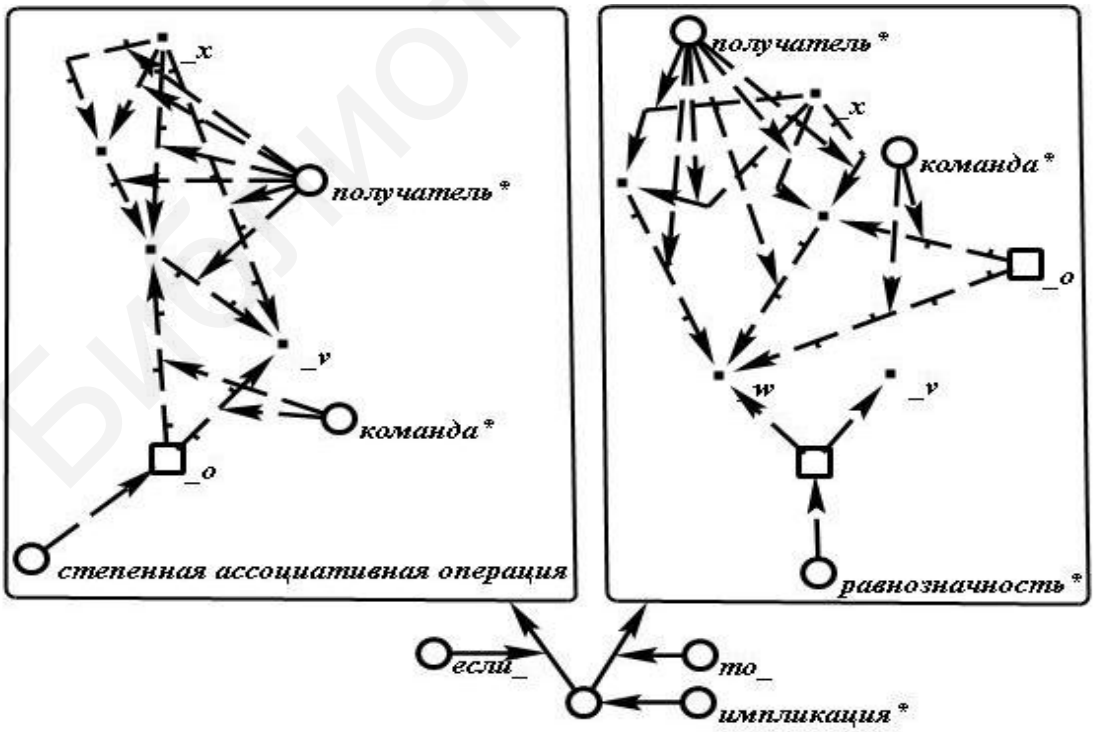


Рисунок 16 – Схема правила преобразования информационного графа с командой операции, обладающей степенной ассоциативностью

Процедура параллельного вычисления редукции массива

Решение *задачи редукции массива* [36] сводится к последовательному применению **операции редукции** ($\bullet$) ко всем элементам массива и текущему результату, начиная с нейтрального элемента. Путём преобразований её решение может быть сведено к эффективному параллельному решению описываемому процедурами, представленными на рисунках 17 и 18 (n – номер потока).

```
⟨a, size⟩ ←                          ⟨a, size, n⟩ ←
i ← (ведущийБит(size) >> 1)          i ← (ведущийБит(size) >> 1)
пока(i ≥ 1)                          пока(i ≥ 1)
| если(i + i > size)                 | если(i + i > size)
| | k ← size − i                     | | k ← size − i
| иначе                              | иначе
| | k ← i                            | | k ← i
| a[0 : k] ← a[0 : k] • a[i : i + k] | если(n < k) a[n] ← a[n] • a[n + i]
| i ← (i >> 1)                       | i ← (i >> 1)
← a[0]                               ← a[0]
```

Рисунок 17 – Процедуры параллельного вычисления редукции массива a с количеством элементов $size$ (слева – для SIMD, справа – для MIMD (SPMD))

```
ведущийБит                           | i ← (i >> k) | i
i ←                                  | k ← (k << 1)
k ← 1                                ← i + 1
пока(i ≠ ((i >> k) | i))
```

Рисунок 18 – Процедура вычисления ведующего бита числа i

Временная сложность этих процедур – $O(\log(n) \cdot f(n))$, где n – количество элементов массива, а $f(n)$ – время доступа к одной ячейке (в массиве).

Процедура параллельного вычисления редукционной прогрессии

Задачу *редукционной прогрессии* можно решить, последовательно решая задачу редукции массива a с сохранением всех текущих результатов операции редукции ($\bullet$). Различают эксклюзивный вариант:

$p_0 \leftarrow e; \quad a_0 \leftarrow e; \quad p_{i+1} \leftarrow a_i \bullet p_i$

и инклюзивный вариант этой задачи (здесь e – нейтральный элемент):

$p_0 \leftarrow e; \quad a_0 \leftarrow e; \quad p_{i+1} \leftarrow a_{i+1} \bullet p_i.$

Частным случаем задачи редукционной прогрессии является задача вычисления кумулятивной инклюзивной или эксклюзивной (префиксной) суммы [36]. Далее приведены процедуры вычисления редукционной прогрессии массива a размером $size$, обладающие одинаковой временной сложностью $O\left(\log(n)\cdot f(n)\right)$, однако первая выполняется до двух раз быстрее, тогда как вторая выполняет примерно в логарифм раз меньшее число команд (рисунки 19 и 20).

$\langle a, size, e, f \rangle \leftarrow$
$q \leftarrow ведущийБит(size)$
$i \leftarrow 1$
$p[0:q] \leftarrow e$
$p[f:size] \leftarrow a[0:size-f]$
$пока(i < q)$
$k \leftarrow q/(i+i)$
$p[0:k, 0:i+i] \leftarrow p[0:q]$
$p[0:k, 0:i] \leftarrow p[0:k, 0:i] \bullet p[0:k, i:i+i]$
$p[0:q] \leftarrow p[0:k, 0:i+i]$
$i \leftarrow i+i$
$\leftarrow p[0:size]$

Рисунок 19 – Процедура быстрого вычисления редукционной прогрессии (SIMD)

$\langle a, size, e, f \rangle \leftarrow$
$q \leftarrow ведущийБит(size)$
$i \leftarrow 1$
$p[0:q] \leftarrow e$
$p[f:size] \leftarrow a[0:size-f]$
$пока(i < q)$
$j \leftarrow i+i$
$k \leftarrow q/j$
$p[0:k, 0:j] \leftarrow p[0:q]$
$p[0:k, j-1:j] \leftarrow p[0:k, i-1:i] \bullet p[0:k, j-1:j]$
$p[0:q] \leftarrow p[0:k, 0:j]$
$i \leftarrow j$
$пока(i > 1)$
$k \leftarrow q/i$
$i \leftarrow (i >> 1)$
$p[0:k, 0:i+i] \leftarrow p[0:q]$
$p[0:k, i+(i>>1)-1:i+(i>>1)] \leftarrow p[0:k, i-1:i] \bullet p[0:k, i+(i>>1)-1:i+(i>>1)]$
$p[0:q] \leftarrow p[0:k, 0:i+i]$
$\leftarrow p[0:size]$

Рисунок 20 – Процедура вычисления редукционной прогрессии (SIMD)
(e – нейтральный элемент, f – флаг эксклюзивности)

Процедура параллельной сортировки

Задача сортировки может быть решена быстрее при использовании параллелизма. На рисунке 21 представлены информационные графы (соответствующе сортировочным сетям) и процедуры, обеспечивающее эффективное решение задачи сортировки [31] на параллельных архитектурах.

$\langle a, size, s\rangle \leftarrow$

$i \leftarrow size$

$q \leftarrow$ *ведущийБит*$(size)$

$p[0:q] \leftarrow s$

$p[0:size] \leftarrow a[0:size]$

$s \leftarrow 1$

пока$(s < size)$

$t \leftarrow 1$

$s \leftarrow s + s$

$b \leftarrow 1$

пока$(t < s)$

$w \leftarrow s / t$

$d \leftarrow w / 2$

$k \leftarrow q / w$

$c[0:k, 0:d] \leftarrow p[0:q/2]$

$u[0:k, b:d-b] \leftarrow p[q/2:q]$

$l[0:k, 0:d] \leftarrow \min(\{c[0:k, 0:d]\} \cup \{u[0:k, 0:d]\})$

$u[0:k, 0:d] \leftarrow \max(\{c[0:k, 0:d]\} \cup \{u[0:k, 0:d]\})$

$p[0:q/2] \leftarrow l[0:k, 0:d]$

$p[q/2:q] \leftarrow u[0:k, b:d-b]$

$t \leftarrow t + t$

$b \leftarrow 0$

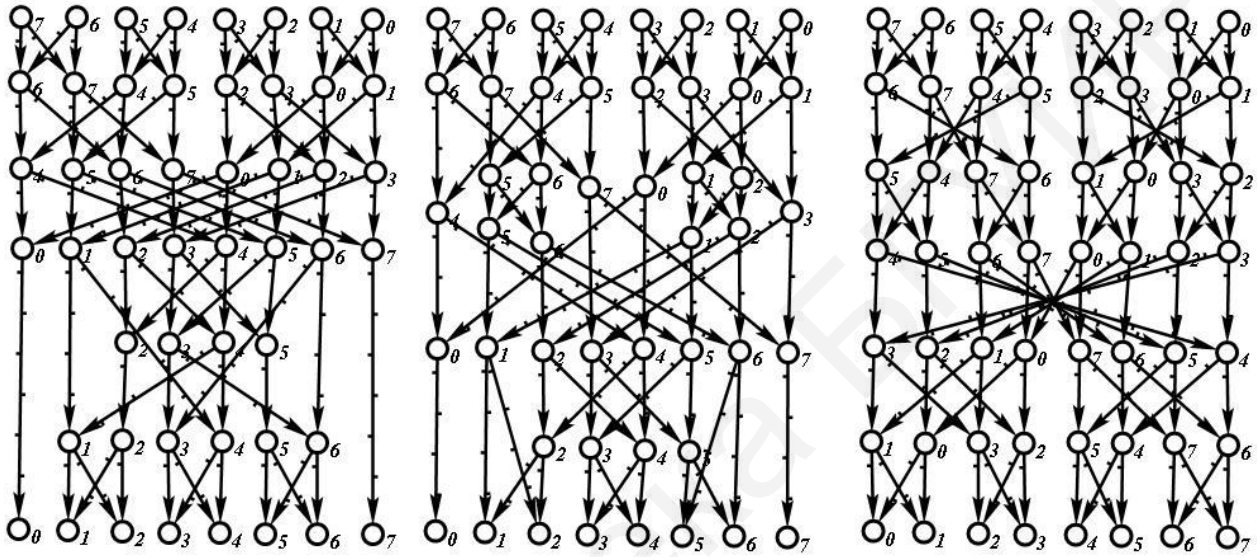
$\leftarrow p[0:size]$

Рисунок 21 – Процедура битонной сортировки массива по возрастанию ($size$ – количество элементов массива a, S – максимальное возможное значение)

В представленных на рисунке 22 информационных графах узлы, в которые входят под наклоном дуги справа, соответствуют операциям вычисления минимума значений аргументов, а узлы, в которые входят под наклоном дуги слева, – операциям вычисления максимума значений аргументов.

Приведенные процедуры обладают одинаковой временной сложностью: $O\left(\log^2(n)\cdot f(n)\right)$.

Платформой для реализации алгоритмов обработки структур данных, обладающих различными формами параллелизма, в том числе естественной, является OpenCL [28, 29].



а б в

а – попарная; б – чётно-нечётная; в – битонная

Рисунок 22 – Примеры информационных графов алгоритмов сортровки

Операции над строками

К операциям над строками [17] относятся: добавление элемента в конец строки (push), удаление элемента с конца строки (pop), удаление элементов строки (popall), конкатенация строк (операция редукции) (concat), разделение строки (split), получение элемента строки по индексу (valueat), начало итерирования (start), итерирование по строке вперёд (increment), итерирование по строке назад (decrement), прерывание итерирования (break), проверка, что символ является пустым (over), получение текущего символа строки (value), удержание элемента (keep), запоминание под (именем) номером (memorize), воспоминание по (имени) номеру (remember), забывание по (имени) номеру (lose).

Реализация последовательных операций над строками рассмотрена в [25] (таблица 16). Операции над строками неограниченной длины могут быть реализованы с помощью указателей переменной разрядности с не менее чем двойным (и не более чем четвертным) запасом по количеству разрядов. При выявлении в структуре строки блока с указателем, имеющим недостаточное (избыточное) количество разрядов, блок переконструируется.

Таблица 16 – Характеристики операций обработки строк и списков данных

Операция	Временная сложность	Оценка пропускной способности
push	$O\left(\log_q(n)\cdot g(n)\right)$	$O\left(\log_n(q)/g(n)\right)$
pop	$O\left(h(r,q,n)+\log_q(n)\cdot g(n)\right)$	$O\left(1/\left(h(r,q,n)+\log_q(n)\cdot g(n)\right)\right)$
popall	$O\left(h(r,q,n)+g(n)\cdot n/q\right)$	$O\left(1/\left(h(r,q,n)+g(n)\cdot n/q\right)\right)$
concat split	$O\left(h(r,q,n)+\log_q(n)\cdot g(n)\right.$ $\left.+f(n)\cdot q\cdot\log_q(n)\right)$	$O\left(1/\left(h(r,q,n)+\log_q(n)\cdot g(n)\right.\right.$ $\left.\left.+f(n)\cdot q\cdot\log_q(n)\right)\right)$
valueat	$O\left(\log_2(n)\cdot f(n)\right)$	$O\left(\log_n(2)/f(n)\right)$
break\start increment decrement	$O\left(h(r,q,n)+\log_q(n)\cdot f(n)\right)$	$O\left(1/\left(h(r,q,n)+\log_q(n)\cdot f(n)\right)\right)$
over	$O\left(f(n)\right)$	$O\left(1/f(n)\right)$
value	$O\left(f(n)\right)$	$O\left(1/f(n)\right)$
keep	$O\left(\log_q(n)\cdot g(n)\right)$	$O\left(\log_n(q)/g(n)\right)$
memorize lose	$O\left(h(r,q,n)+\log_q(n)\cdot g(n)\right.$ $\left.+\left(\log_2^2(n)+q\cdot\log_q(n)\right)\cdot f(n)\right)$	$O\left(1/\left(h(r,q,n)+\log_q(n)\cdot g(n)\right.\right.$ $\left.\left.+\left(\log_2^2(n)+q\cdot\log_q(n)\right)\cdot f(n)\right)\right)$
remember	$O\left(h(r,q,n)+\log_2^2(n)\cdot f(n)\right)$	$O\left(1/\left(h(r,q,n)+\log_2^2(n)\cdot f(n)\right)\right)$

Вопросы

1. Чем является операция редукции?
2. Какова наименьшая временная сложность параллельной реализации алгоритма, вычисляющего префиксную сумму массива из *n* элементов в наихудшем случае (время чтения, записи и сложения двух элементов массива равно одному такту)?
3. Какова наименьшая временная сложность параллельной реализации алгоритма, вычисляющего результат *n* имеющих прямую зависимость команд операции редукции в наихудшем случае (время вычисления одной команды равно одному такту)?
4. Какова наименьшая временная сложность параллельной реализации алгоритма битонной сортировки массива из *n* элементов в наихудшем случае (время чтения, записи и сравнения двух элементов массива равно одному такту)?
5. На что разделены устройства OpenCL (1.2)?
6. На что разделены процессоры устройства OpenCL (1.2)?

7. Какая абстракция OpenCL (1.2) относится к окружению управления ресурсами и объектами, обеспечивая их взаимосвязь?
8. Какая абстракция OpenCL (1.2) является механизмом хоста, способствующим запросам и порядку исполнения действий на устройстве?
9. Как называется функция (процедура) в программе, предназначенная для исполнения на OpenCL (1.2) устройстве?
10. Какого региона память, недоступная с устройств, описывается стандартом OpenCL (1.2)?
11. Как называется регион памяти OpenCL (1.2), доступной для чтения и записи для всех заданий (work-item) во всех рабочих группах (work-group)?
12. Как называется регион памяти OpenCL (1.2) (инициализируемой хостом), из которого задания (work item) могут лишь читать данные?
13. Как называется регион памяти OpenCL (1.2), доступной для чтения и записи лишь заданиям (work-item) одной рабочей группы (work group)?
14. Как называется регион памяти OpenCL (1.2), доступной лишь одному заданию (work item)?
15. Во что группируются задания OpenCL (1.2) (work-item)?
16. Во что собираются рабочие группы OpenCL (1.2) (work-group)?
17. Чем отличается исполняемая гранула от гранулы?
18. Что возвращает процедура get_global_id?
19. Что возвращает процедура get_local_id?
20. Что возвращает процедура get_group_id?

4 Задачи уровня управления знаниями и модели обработки знаний

К задачам этого уровня относятся: анализ и синтез программ, анализ и синтез операций и моделей обработки информации (агентов), синтез многоагентных систем обработки потоков знаний и управление ими. Основными принципами для успешного решения поставленных задач являются: учёт НЕ-факторов, семантическое протоколирование [24], обработка потока знаний.

Статические и динамические модели

Информационный граф и граф состояний зачастую рассматриваются как относящиеся к статическим моделям. Динамические модели в отличие от статических, будучи выраженными на языке в виде текста, обладают не только интерпретируемостью, но и активностью, которая реализуется операционной семантикой. К динамическим моделям могут быть отнесены: сети Петри, автотрансформационный граф (Т-система) [1] и др. У динамической модели граф (гиперграф) или его свойства меняются от состояния к состоянию.

Сети Петри

Сеть Петри задаётся графом [14, 15], соответствиями (множеством разметок) и множеством параметров (меток).

Граф сети Петри является двудольным. В двудольном графе множество вершин разбивается на две доли, два подмножества, каждое из которых является множеством несмежных вершин. Смежными в двудольном графе, т. е. соединёнными рёбрами, могут быть только вершины из разных долей. Одна доля графа сети Петри называется множеством позиций, а другая – множеством переходов.

Операции сети Петри

Операции сети Петри позволяют сменить состояние одной разметки сети Петри на другую разметку сети Петри. Они не являются в общем случае детерминированными операциями и описывают асинхронное срабатывание переходов сети Петри. Задача управления сетями Петри рассматривается в [5].

Условия срабатывания переходов

Для выполнения операции срабатывания перехода сети Петри необходимо, чтобы в текущем состоянии каждая из позиций, из которых выходят дуги в рассматриваемый переход, имела не меньшее количество меток, чем кратность выходящей из позиции в рассматриваемый переход дуги.

Результат срабатывания переходов

В результате выполнения операции срабатывания перехода (рисунок 23) сети Петри формируется новое состояние с новой разметкой, в которой по отношению к прежней разметке:

а) отсутствуют прежние метки в каждой из позиций, из которых выходят дуги в срабатывающий переход, в количестве, равном кратности выходящей из позиции в рассматриваемый переход дуги;

б) присутствуют новые метки в каждой из позиций, в которые входят дуги из срабатывающего перехода, в количестве, равном кратности входящей в позицию из рассматриваемого перехода дуги.

Сети Петри способны описывать разные параллельные системы (рисунок 24).

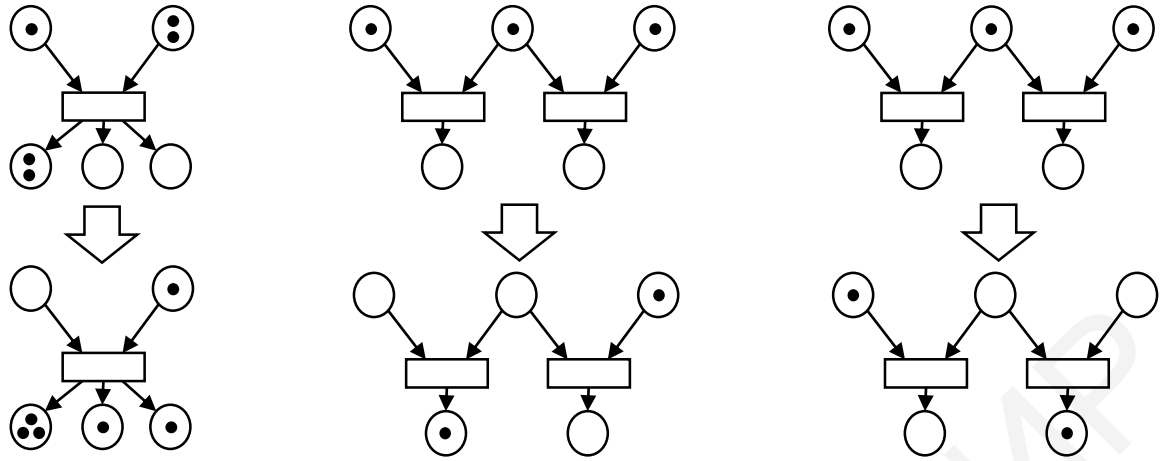
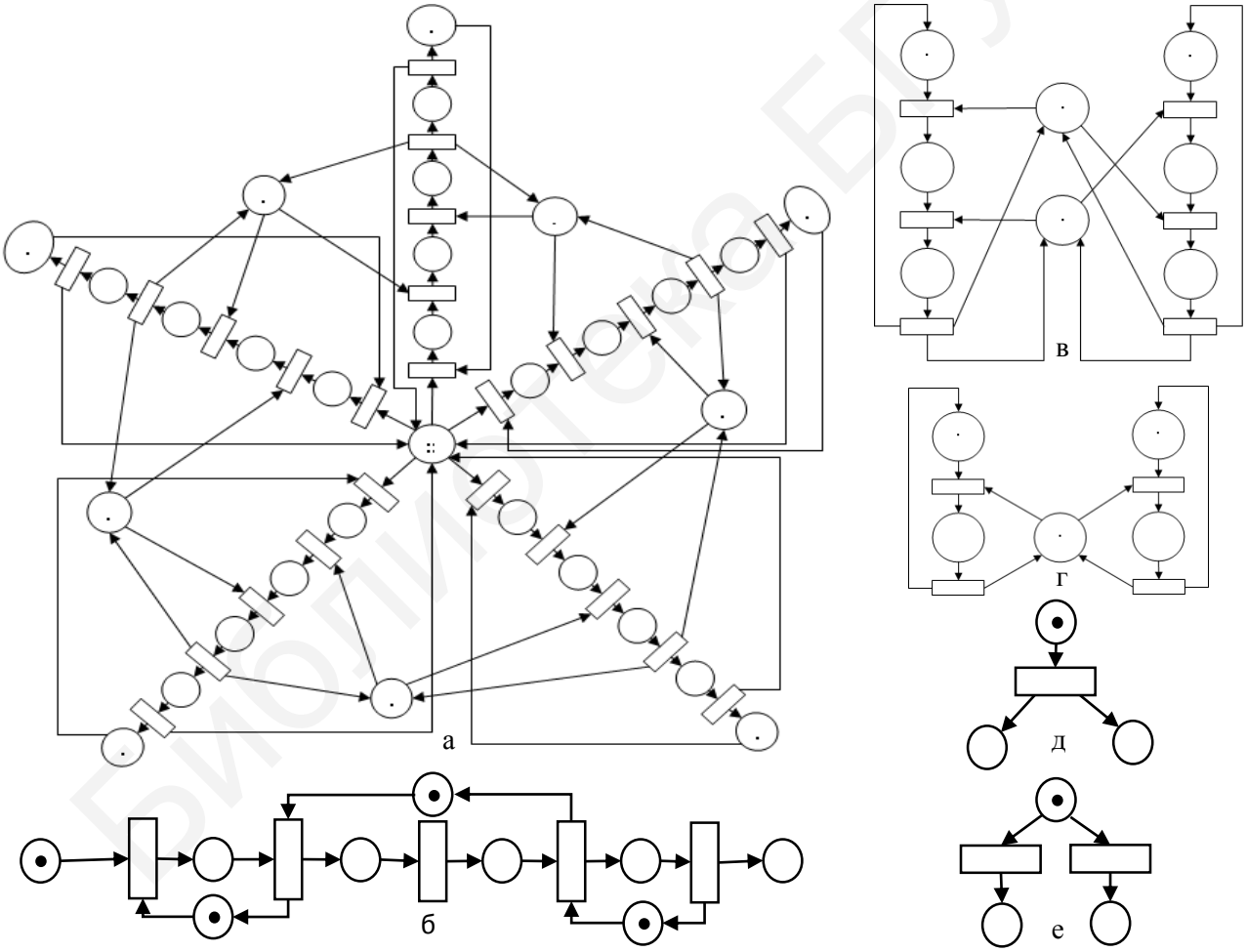


Рисунок 23 – Примеры срабатывания переходов в сетях Петри



а – решение задачи с обедающими философами (возможно «голодание»);
б – конвейер; в – взаимодействие параллельных процессов (возможна взаимная блокировка); г – взаимное исключение процессов;
д – оператор распараллеливания «fork»; е – условный оператор «if-then-else»

Рисунок 24 – Примеры описания сетями Петри параллельных систем

Расширения сетей Петри

Поглотители – дуги, которые выходят из перехода в позицию. При срабатывании перехода удаляют метки из позиции в количестве, соответствующем кратности поглотителя, если они находились в ней.

Ингибиторы – дуги, которые выходят из позиции в переход. Запрещают срабатывание перехода при наличии меток в позиции в количестве, соответствующем кратности ингибитора.

Универсальная сеть Петри, моделирующая работу универсальной машины Тьюринга, может быть построена с использованием сетей Петри и ингибиторов [37].

К расширенным моделям сетей Петри относят: раскрашенные сети Петри, вероятностные сети Петри и др.

Программа, граф потока управления программы

Чтобы решить задачу управления, составляется план или программа управления, реализующие ту или иную стратегию. На уровне управления знаниями объектом управления могут являться программы как частный вид знаний. В этом случае необходимо создать план или программу составления программ в соответствии с тем или иным методом оптимального планирования (программ), которые могут быть заданы графом потока управления программы.

Вершины *графа потока управления программы* соответствуют командам (или базовым блокам), а рёбра – потоку управления (переходам между командами).

В отличие от информационного графа граф потока управления (программы) может содержать *циклы*. Некоторые графы потока управления программы, включающие циклы, позволяют задавать информационные графы с большим количеством команд, чем в графе потока управления программы, когда одна команда в теле цикла соответствует множеству команд информационного графа, один граф потока управления программы, включающий циклы, также позволяет задать множество информационных графов, соответствующих разным диапазонам значений переменной цикла. На рисунке 25 показана схема правила, позволяющего перейти от графа потока управления программы к графу потока управления программы, содержащему цикл.

Коэффициент расхождения [13] программы равен отношению суммарной длины программы к её средней длине:

$$D = \frac{L_{\Sigma}}{L_{cp}}.$$

Суммарная длина L_{Σ} равна сумме длин l_{ik} всех команд графа потока управления программы. Длина команды равна её времени выполнения. Средняя длина L_{cp} равна отношению суммы произведений длины команды на ранг задачи r_{ik}, решаемой ею, к суммарному рангу r всех команд одного яруса ярусного параллельного графа потока управления программы. Команды одного яруса в

ярусном параллельном графе потока управления выполняются одновременно. Суммарный ранг всех команд на каждом ярусе полагается одинаковым:

$$L_{\Sigma} = \sum_k \sum_i^{1 \le i \le s_k} l_{ik} \; ; \; L_{cp} = \frac{\sum_k \sum_i^{1 \le i \le s_k} \left(r_{ik} * l_{ik} \right)}{r}; \qquad r = \sum_i^{1 \le i \le s_k} r_{ik} \; .$$

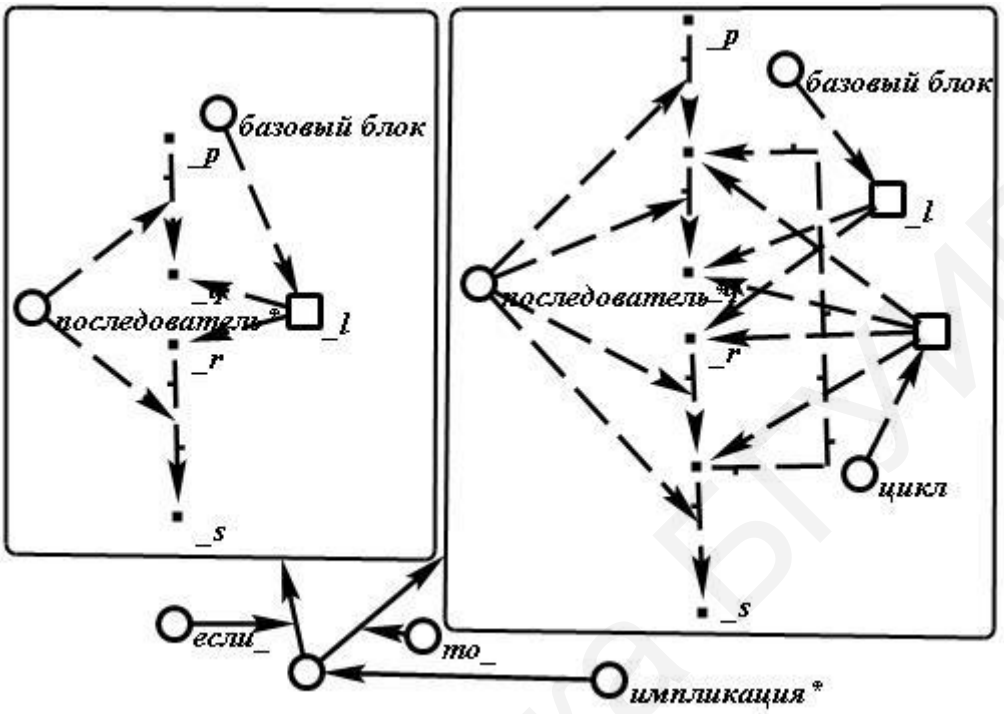


Рисунок 25 – Схема правила преобразования базового блока в тело цикла

Планирование и преобразование программ и их частей

Оптимальное планирование может зачастую требовать распараллеливания кода, поэтому методы оптимального планирования тесно связаны с подходами к распараллеливанию программ, которые включают:

- статическое распараллеливание (ручное [21, 27] и автоматическое);
- динамическое распараллеливание [1].

Среди методов оптимального планирования [20] выделяют:

- глобальные;
- локальные;
- циклических программ;
- ациклических программ.

К одним из основных типов частей программы относятся базовые блоки и гнёзда циклов.

Базовый блок – часть программы без команд условного перехода. Среди методов планирования базовых блоков выделяют метод планирования трасс и другие методы [1, 19, 20].

Гнездо циклов – часть программы, являющаяся циклом, частью тела которого могут быть другие гнёзда циклов. Различают совершенные и несовершенные гнёзда циклов [1, 19, 20]. Методы планирования гнёзд циклов

используют преобразования (трансформации), позволяющие получить для заданной архитектуры программу с меньшим временем исполнения.

Различают следующие виды преобразований [1, 19, 20]: общего назначения (применяются не только к циклам) (таблица 17), реорганизации и реструктуризации циклов (таблицы 18 и 19). Некоторые из преобразований являются частными случаями более общих преобразований, к основным механизмам которых относятся исключение избыточности и запоминание.

Таблица 17 – Примеры преобразований общего назначения для циклов

Преобразование	Исходный цикл	Преобразованный цикл
Чистка цикла	$for(i=0;i<n;i++)\{$ $a[i]=a[i]+sqrt(x);$ $\}$	$if\ (n\geq 0)C=sqrt(x);$ $for(i=0;i<n;i++)\{$ $a[i]=a[i]+C;$ $\}$
Втягивание констант	$n=64;$ $c=3;$ $for(i=0;i<n;i++)\{$ $a[i]=a[i]+c;$ $\}$	$for(i=0;i<64;i++)\{$ $a[i]=a[i]+3;$ $\}$
Подстановка вперёд	$np1=n+1;$ $for(i=0;i<n;i++)\{$ $a[np1]=a[np1]+a[i];$ $\}$	$for(i=0;i<n;i++)\{$ $a[n+1]=a[n+1]+a[i];$ $\}$
		$n=get\_global\_size(0);$ $i=get\_global\_id(0);$ $if\ ((i<n)\&\&(i\geq 0))\{$ $atomic\_add(a+n+1,a[i]);$ $\}$
Понижение силы операций	$for(i=0;i<n;i++)\{$ $a[i]=a[i]+c\cdot i;$ $\}$	$T=c;$ $for(i=0;i<n;i++)\{$ $a[i]=a[i]+T;$ $T=T+c;$ $\}$

Запоминание (мемо(р)изация) – механизм [24], позволяющий преобразовать первичное и повторное решение индивидной задачи к первичному решению задачи с записью (запоминанием) результата и чтению (воспоминанию)

результата по заданному ключу (имени). Это позволяет сократить время решения, если время (повторного) решения задачи превосходит время записи и чтения результата. Частным случаем такого преобразования является *втягивание констант* (и частично *чистка цикла* (рисунок 26)), а обратное преобразование иногда может быть выполнено как *подстановка вперёд*.

Исключение избыточности – преобразование, которое позволяет удалить из программы команды, которые никогда не выполняются, либо их выполнение не влияет на результат и достижение цели.

Понижение силы операций связано с рекурсивным определением операции $(\bullet)$ через себя и другие $(\circ)$, например:

$(a \bullet (i \circ e')) = ((a \bullet i) \circ a)$; $\quad (a \bullet e') = a$.

Когда необходимо в цикле решить задачи, связанные как задачи и их подзадачи, требующие вычисления множества значений операции $(\bullet)$, то эта $(\bullet)$ операция каждый раз не вычисляется, а вычисляется промежуточный рекурсивный результат, который запоминается и используется для решения последующей надзадачи путём применения операции $(\circ)$, на которой выстроена рекурсия.

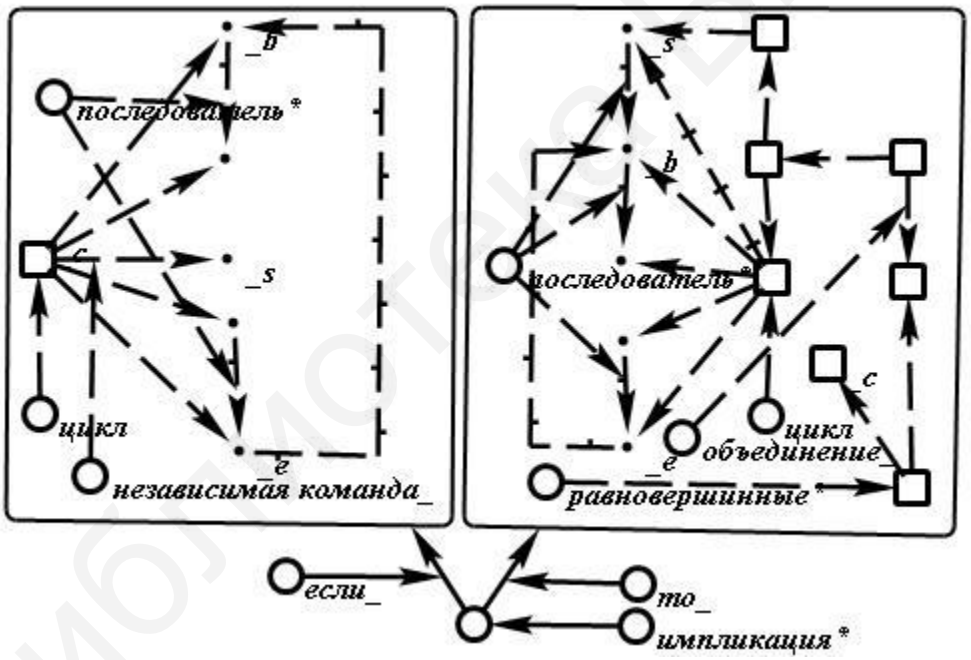


Рисунок 26 – Схема правила преобразования чистки цикла от независимой операции

Для анализа совершенных гнёзд циклов используется модель *итерационного пространства* (совершенного гнезда циклов) [1, 19, 20], заданная множествами итераций совершенного гнезда циклов и отношений зависимости между ними. Каждой итерации совершенного гнезда циклов взаимно однозначно соответствует вектор (ориентированное множество) значений переменных всех циклов гнезда. Количество итераций в совершенном гнезде циклов равно произведению всех количеств всех итераций каждого цикла в

нём. Если в гнезде циклов между командами на двух итерациях существует зависимость (команды не являются независимыми), то эта зависимость может быть задана *дистантным вектором*, равным разности векторов последующей и предыдущей зависимых итераций. Цикл параллелизуем, когда все итерации в нём не имеют зависимых команд и могут быть исполнены параллельно (рисунок 27).

$for(i=0;i<n;i++)\{$	$for(i=1;i<n;i++)\{$
$for(j=1;j<n;j++)\{$	$for(j=0;j<n-1;j++)\{$
$a[i,j]=a[i,j-1]+c;$	$a[i,j]=a[i-1,j]+a[i-1,j+1];$
$\}$	$\}$
$\}$	$\}$

Рисунок 27 – Параллелизуемые циклы (слева – внешний, $\{(0,1)\}$ – дистантный вектор) справа – внутренний, векторы – $\{(1,0),(1,-1)\}$)

В совершенном гнезде циклов параллелизуем p*-й* цикл, если и только если для любого дистантного вектора v
$\left((v_p=0)\vee\left(\exists q\left((q<p)\wedge(v_q>0)\right)\right)\right)$.

Цель преобразований реорганизации – изменить дистантные векторы для итерационного пространства.

Таблица 18 – Примеры преобразований реорганизации циклов

Преобразование	Исходное гнездо циклов	Преобразованное гнездо циклов
1	2	3
Распределение циклов (рисунок 28)	$for(i=0;i<n-1;i++)\{$ $a[i]=a[i]+c;$ $\}$ $for(i=0;i<n-1;i++)\{$ $x[i+1]=x[i]*7+x[i+1]+a[i];$ $\}$	$for(i=0;i<n-1;i++)\{$ $a[i]=a[i]+c;$ $x[i+1]=x[i]*7+x[i+1]+a[i];$ $\}$

1	2	3
Перестановка циклов (рисунок 29)	$for(i=0;i<n;i++)\{$ $for(j=0;j<n;j++)\{$ $total[i]=total[i]+a[i,j];$ $\}$ $\}$	$for(j=0;j<n;j++)\{$ $for(i=0;i<n;i++)\{$ $total[i]=total[i]+a[i,j];$ $\}$ $\}$
Слияние циклов	$for(i=0;i<n-1;i++)\{$ $a[i]=a[i]+c;$ $x[i+1]=x[i]*7+x[i+1]+a[i];$ $\}$	$for(i=0;i<n-1;i++)\{$ $a[i]=a[i]+c;$ $\}$ $for(i=0;i<n-1;i++)\{$ $x[i+1]=x[i]*7+x[i+1]+a[i];$ $\}$
		$n=get\_global\_size(0);$ $i=get\_global\_id(0);$ $if((i<n-1)\&\&(i\geq 0))\{$ $a[i]=a[i]+c;$ $\}$ $barrier(CLK\_MEM\_LOCAL\_FENCE);$ $barrier(CLK\_MEM\_GLOBAL\_FENCE);$ $if(i==0)\ for(;i<n-1;i++)\{$ $x[i+1]=x[i]*7+x[i+1]+a[i];$ $\}$
Обращение цикла	$for(i=1;i<n;i++)\{$ $for(j=0;j<n-1;j++)\{$ $a[i,j]=a[i-1,j+1]+1;$ $\}$ $\}$	$for(i=1;i<n;i++)\{$ $for(j=n-1;j-->0;)\{$ $a[i,j]=a[i-1,j+1]+1;$ $\}$ $\}$

В примере преобразования обращения исходное гнездо циклов имеет дистантный вектор (1, –1), а после обращения внутреннего цикла гнездо циклов имеет дистантный вектор (1,1).

Преобразование распределения циклов является обратным к преобразованию слияния циклов.

Кроме перечисленных преобразований реорганизации циклов к таковым относится преобразование, использующее механизм программного конвейера [1, 19].

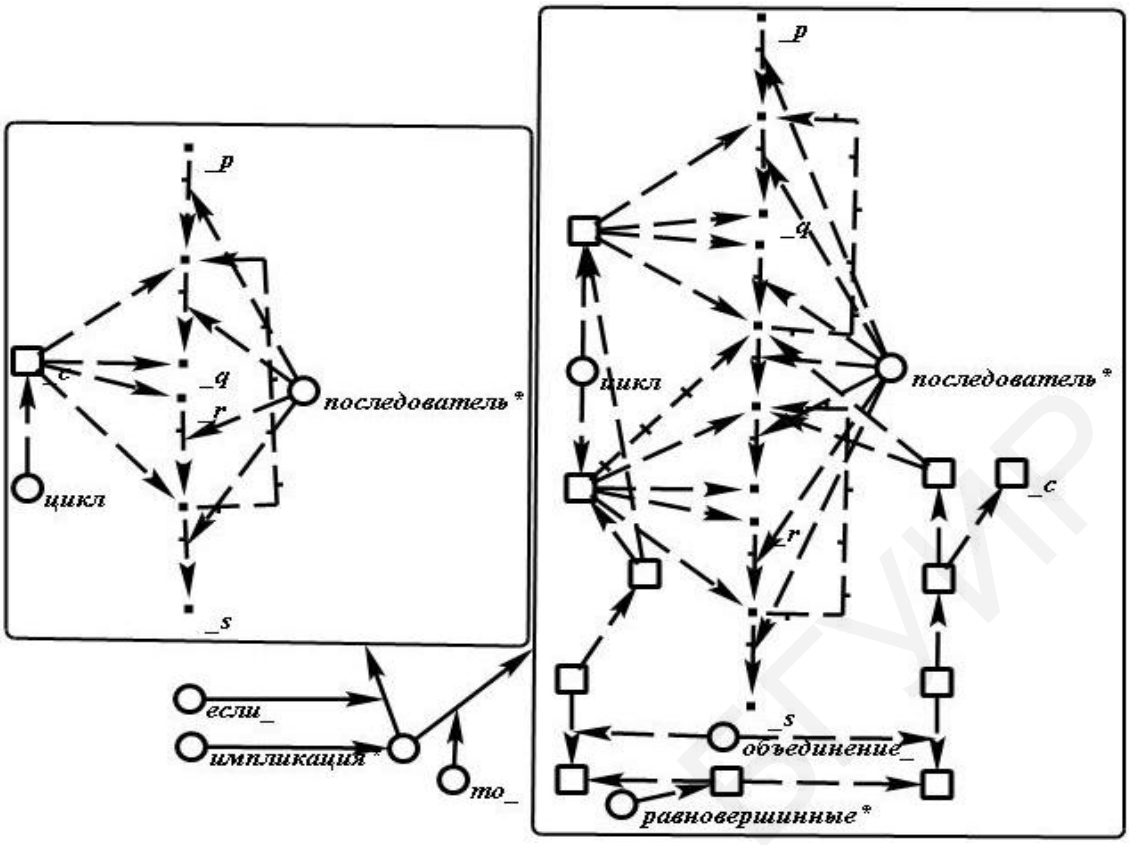


Рисунок 28 – Схема правила распределения циклов

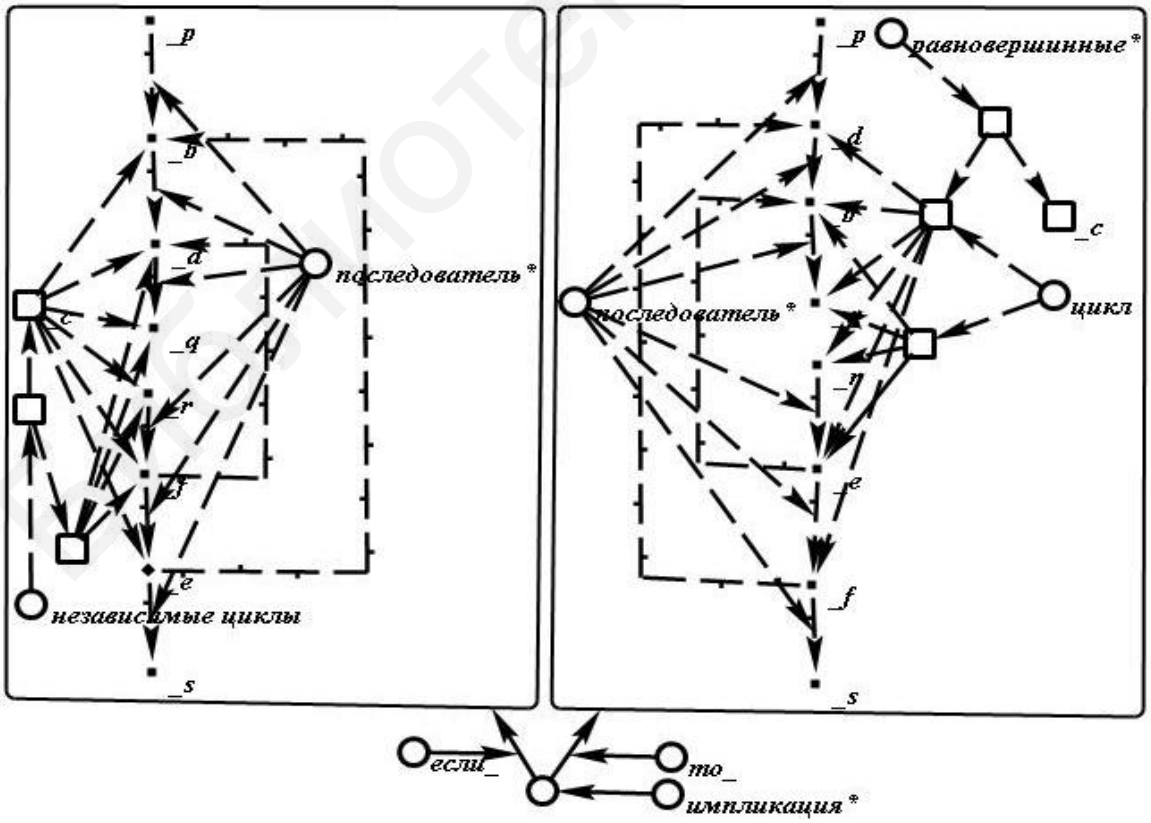


Рисунок 29 – Схема правила перестановки циклов

Таблица 19 – Примеры преобразований реструктуризации циклов

Преобразование	Исходный цикл	Преобразованный цикл
1	2	3
Развёртка цикла	$for(i=1;i<n-1;i++)\{$ $a[i]=a[i]+a[i-1]*a[i+1];$ $\}$	$for(i=1;i<n-2;i+=2)\{$ $a[i]=a[i]+a[i-1]*a[i+1];$ $a[i+1]=a[i+1]+a[i]*a[i+2];$ $\}$ $if((n\%2==1)\&\&(n>2))\{$ $a[n-2]=a[n-2]+a[n-3]*a[n-1];$ $\}$
Разгрузка цикла	$for(i=0;i<n;i++)\{$ $b[i]=b[i]+b[0];$ $\}$ $for(i=1;i<n;i++)\{$ $a[i]=a[i]+c;$ $\}$	$if(n>0)\{$ $b[0]=b[0]+b[0];$ $\}$ $for(i=1;i<n;i++)\{$ $b[i]=b[i]+b[0];$ $a[i]=a[i]+c;$ $\}$
Распознавание редукций	$for(i=0;i<n;i++)\{$ $s=s+a[i];$ $\}$	$n=get\_global\_size(0);$ $i=get\_global\_id(0);$ $T[i]=0;$ $for(j=0;j<n;j+=64)\{$ $if(j+i<n)T[i]=T[i]+a[j+i];$ $\}$ $barrier(CLK\_MEM\_LOCAL\_FENCE);$ $barrier(CLK\_MEM\_GLOBAL\_FENCE);$ $if(i==0)for(;i<64;i++)\{$ $s=s+T[i];$ $\}$

1	2	3
Нормализация цикла	$for(i=0;i<n;i++)\{$ $a[i]=a[i]+c;$ $\}$ $for(i=1;i\le n;i++)\{$ $b[i]=a[i-1]+b[i];$ $\}$	$for(i=1;i\le n;i++)\{$ $a[i-1]=a[i-1]+c;$ $\}$ $for(i=1;i\le n;i++)\{$ $b[i]=a[i-1]+b[i];$ $\}$
Соединение циклов	$for(i=0;i<n;i++)\{$ $for(j=0;j<m;j++)\{$ $a[i,j]=a[i,j]+c;$ $\}$ $\}$	$for(k=0;k<n*m;k++)\{$ $j=k\%m;$ $i=(k-j)/m;$ $a[i,j]=a[i,j]+c;$ $\}$

Параллельная редукция

Параллельная редукция – сведение исходной задачи ко множеству независимых подзадач (*И-подзадач*), решаемых параллельно. Результат исходной задачи, как правило, получается применением операции редукции к результатам решения независимых подзадач, к которым была сведена исходная задача. В простейшем случае в качестве операции редукции может быть использована операция конкатенации строк, представляющих результаты решения И-подзадач. Граф состояний с решением исходной задачи может быть получен из результата прямого произведения графов состояний с решениями И-подзадач. За счёт параллелизма сокращается время решения исходной задачи до времени, складывающегося из сокращённого до p раз времени основной задачи $(p \ge r)$, где p – количество процессов, а r – ранг задачи, времени отображения и передачи входных данных исходной задачи на входы подзадач, а также времени на выполнение операций редукции t_r, которое обычно пропорционально $\log_2(p)$. Это соответствует схеме «map-reduce».

Инверсия задач

Инверсия задач по общему типу (решения). Если задачи имеют одинаковое решение (являются однотипными, частными случаями обобщённой задачи), то в целях экономии вычислительных ресурсов в графе управления команды их решения могут быть расположены последовательно, а не параллельно.

Инверсия задач по (интенсивному) обмену сообщениями. Если гранулы решения задач не являются абсолютно независимыми и доля операций обмена сообщениями между командами гранул значительна, то в целях сокращения накладных расходов на передачу сообщений рекомендуется выполнять гранулы последовательно, возможно, поочередно по частям, при этом операции обмена

сообщениями заменяются на операции передачи управления и операции чтения-записи.

Инверсия задач по необходимости синхронизации по времени. Если требуется, чтобы решение задач повторялось с одинаковым периодом во времени, большим суммарного времени решения задач, то такие задачи могут исполняться последовательно в целях сокращения накладных расходов на поддержку работы параллельных процессов.

Операции над индексами и мерами

Если рассмотреть индикаторные или характеристические функции множеств, то обработке данных с помощью операций можно сопоставить операции алгебры матриц матричных представлений (таблица 20) этих функций или операций алгебры этих функций (таблица 21).

Пусть

$Z = \{0_Z, 1_Z\}$,

тогда индикаторная функция множества X будет иметь вид

$i_X \in Z^X$,

причём

$$i_X(x) = \begin{cases} 0_Z \,|\, x \notin X, \\ 1_Z \,|\, x \in X. \end{cases}$$

Таблица 20 – Выражения теоретико-множественных операций над (нечёткими) характеристическими функциями (предикатами) бинарных отношений и их связь с логическими операциями

Операция	Выражение	Формула
$\varphi \circ \psi$	$m_{\varphi\circ\psi} = m_\varphi \cdot m_\psi$	$m_{\varphi\circ\psi}(\langle\alpha,\beta\rangle) = 1_Z - \prod_\gamma \left(\left(1_Z - m_\varphi(\langle\alpha,\gamma\rangle)\right) \cdot \left(1_Z - m_\psi(\langle\gamma,\beta\rangle)\right) \right)$
$\varphi \circ \psi$	$m_{\varphi\circ\psi} = m_\varphi \cdot m_\psi$	$m_{\varphi\circ\psi}(\langle\alpha,\beta\rangle) = \tilde{\vee}_\gamma \left(m_\varphi(\langle\alpha,\gamma\rangle) \,\tilde{\wedge}\, m_\psi(\langle\gamma,\beta\rangle) \right)$
$\varphi \cap \psi$	$m_{\varphi\circ\psi} = m_\varphi \bullet m_\psi$	$m_{\varphi\cap\psi}(\langle\alpha,\beta\rangle) = m_\varphi(\langle\alpha,\beta\rangle) \,\tilde{\wedge}\, m_\psi(\langle\alpha,\beta\rangle)$
$\varphi - \psi$	$m_{\varphi\circ\psi} = m_\varphi \oplus m_\psi$	$m_{\varphi\oplus\psi}(\langle\alpha,\beta\rangle) = m_{\psi/\varphi}(\langle\alpha,\beta\rangle) \,\tilde{\vee}\, m_{\varphi/\psi}(\langle\alpha,\beta\rangle)$
φ / ψ	–	$m_{\varphi/\psi}(\langle\alpha,\beta\rangle) = 1_Z - \left(m_\varphi(\langle\alpha,\beta\rangle) \,\tilde{\rightarrow}\, m_\psi(\langle\alpha,\beta\rangle) \right)$

Таблица 21 – Варианты логических операций над (нечёткими) характеристическими функциями (предикатами) бинарных отношений

Операция	Вариант 1	Вариант 2	Вариант 3
$\chi \tilde{\wedge} \gamma$	$\min\left(\{\chi\} \cup \{\gamma\}\right)$	$\chi \cdot \gamma$	$\max\left(\{0_Z\} \cup \{\chi + \gamma - 1_Z\}\right)$
$\chi \tilde{\vee} \gamma$	$\max\left(\{\chi\} \cup \{\gamma\}\right)$	$\chi + \gamma - \chi \cdot \gamma$	$\min\left(\{1_Z\} \cup \{\chi + \gamma\}\right)$
$\tilde{\bigwedge\limits_{\chi}} \chi$	$\min\limits_{\chi}\left(\{\chi\}\right)$	$\prod_{\chi} \chi$	$\max\limits_{\chi}\left(\{0_Z\} \cup \left\{1_Z - \sum_{\chi}(1_Z - \chi)\right\}\right)$
$\tilde{\bigvee\limits_{\chi}} \chi$	$\max\limits_{\chi}\left(\{\chi\}\right)$	$1_Z - \prod_{\chi}(1_Z - \chi)$	$\min\limits_{\chi}\left(\{1_Z\} \cup \left\{\sum_{\chi} \chi\right\}\right)$

Исходя из онтологической модели событий, явлений и процессов перейдём от рассмотрения отдельных состояний к событиям их наблюдения и уже от них ко множествам таких событий и явлению их становления (становлений). Таким образом, (отношение) становления событий наблюдения состояний можно рассматривать как некоторую операцию. При этом если среди становлений нет изменений, такая операция реализует отношение бытия, т. е. такие становления, которые могут быть выражены высказыванием «A есть A», где A – рассматриваемое событие. Иначе становления выражают некоторые изменения таких событий, которые могут быть отнесены к одному множеству, классу эквивалентности в рамках транзитивного замыкания симметризованного отношения становления, что может быть выражено высказыванием «A заменяется B», где A и B – объекты одного класса. Таким образом, осуществляется переход от преобразования состояний к композиции операций, продолжая далее, путём перехода от операций к их характеристическим функциям (индикаторам) вместо композиции операций получаем операции над индикаторами операций (матрицами), т. е. например, имея операцию o и состояния d и s, от

$$d = o(s)$$

переходим к преобразованиям вида

$$\left(\{d\} \times \{d\}\right) = \left(o^{-1} \circ \left(\{s\} \times \{s\}\right)\right) \circ o,$$

а затем к

$$m_d = \left(\left(m_o\right)^T \circ m_s\right) \circ m_o,$$

где m_s, m_o, m_d – матрицы, задающие соответствующие индикаторы (характеристические функции): $i_{\{s\}\times\{s\}}$, i_o и $i_{\{d\}\times\{d\}}$.

Следует отметить, что при таком представлении за счёт перехода к метапредставлению (индикаторам) можно задать (характеризовать) не одно состояние, а сразу множество состояний, что позволяет более удобно описывать про-

цессы параллельной обработки информации, включая процессы квантовых вычислений [19].

Для требуемого учёта НЕ-факторов необходимо использовать средства, обладающие достаточными соответствующими качествами. Такими средствами обладает модель унифицированного семантического представления знаний, семантика которой основывается на ситуативных множествах.

Отношения и операции над ситуативными множествами

В простейшем случае сигнатура алгебраической системы ситуативных множеств предполагает следующие отношения и операции над ситуативными множествами.

Пусть p – доля исходов, когда отношение связки между элементом и ситуативным множеством оказывается принадлежностью, а $1-p$ – доля исходов, когда отношение связки оказывается непринадлежностью.

Пусть q – доля исходов выбрать принадлежность, а $1-q$ – доля исходов выбрать непринадлежность для выяснения (средней) доли исходов, в которых связка между элементом и ситуативным множеством оказывается связкой выбранного отношения. Тогда средняя доля исходов – $p\cdot q+(1-p)\cdot(1-q)$ (рисунок 30).

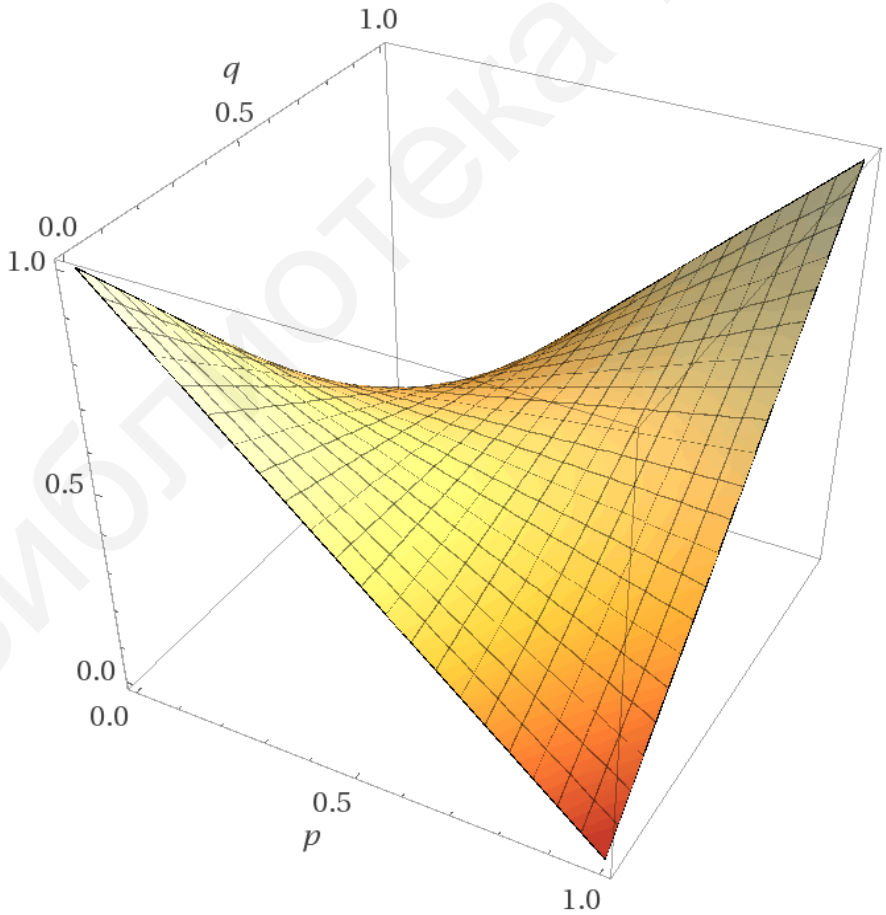


Рисунок 30 – График функции (нечёткой) ситуативной принадлежности аргументов второго порядка ситуативному множеству

Обобщая это понятие, перепишем выражение в общем рекуррентном виде:

$$\pi(s)=\begin{cases}1 \mid s=\varnothing \\ e\cdot\pi\left(s/\{e\}\right)+(1-e)\cdot\left(1-\pi\left(s/\{e\}\right)\right) \mid \{e\}\subseteq s\end{cases}$$

или нерекуррентном:

$$\pi(s)=\frac{1+(-1)^{|s|}\cdot\sum_{i}^{i\in\mathbb{N}\cup\{0\}}(-2)^{i}\cdot\sum_{m}^{m\in\{x\|x|=i\}\cap 2^{s}}\sum_{e}^{e\in m}e}{2},$$

где S – множество долей исходов.

Пусть σ – соответствия (нечёткой) *ситуативной принадлежности* ситуативному множеству, тогда

$$\forall\chi\exists q\forall\gamma\left(\langle\chi,\gamma,q\rangle\in U\times[0;1]^{q}\times\left(\mathbb{N}\cup\{0\}\right)\right)\rightarrow\rho\left(\langle\sigma_{1}(\chi),\gamma\rangle\right);$$

$$\forall\chi\forall\varepsilon\exists q\forall\gamma\left(\langle\chi,\varepsilon,\gamma,q\rangle\in U\times E\times[0;1]^{q}\times\left(\mathbb{N}\cup\{0\}\right)\right)\rightarrow\rho\left(\langle\sigma_{2}(\chi)(\varepsilon),\gamma\rangle\right),$$

где $\forall\psi\left(\psi[1]\in\left\{\psi(\tau)\middle|\left(\kappa\in\mathbb{N}\cup\{0\}\right)\wedge\left(\tau\in\{1\}^{\kappa}\right)\right\}\right);\qquad \{1\}^{0}\underset{def}{=}\{\varnothing\};$

$$\rho\left(\langle\chi,\tau\rangle\right)\underset{def}{=}\left(\chi(\tau)=\pi\left(\{\chi[1]\}\cup\left\{\tau_{\kappa}\middle|\left(\kappa\in\mathbb{N}/\{\varnothing\}\right)\wedge\left(\kappa\le\dim(\tau)\right)\right\}\right)\right).$$

Ситуативное множество является α ситуативным *подмножеством* ситуативного множества β, если и только если:

$$\{\alpha\}\cup\{\beta\}\subseteq\mu;\qquad \alpha_{1}\subseteq\beta_{1};$$

$$\forall\chi\forall\varepsilon\left(\langle\chi,\varepsilon\rangle\in U\times E\right)\rightarrow\left(\left(\alpha_{21}(\chi)[1]\le\beta_{21}(\chi)[1]\right)\wedge\left(\alpha_{22}(\chi)(\varepsilon)[1]\le\beta_{22}(\chi)(\varepsilon)[1]\right)\right).$$

Ситуативные множества из пары ситуативных множеств *равны*, если и только если каждое из них является ситуативным подмножеством другого ситуативного множества.

Ситуативным *пересечением* ситуативных множеств α и β является ситуативное множество γ, такое, что:

$$\{\alpha\}\cup\{\beta\}\cup\{\gamma\}\subseteq\mu;\qquad \gamma_{1}=\alpha_{1}\cup\beta_{1};$$

$$\forall\chi(\chi\in U)\rightarrow\left(\gamma_{21}(\chi)[1]=\varphi\left(\left\langle\gamma_{22}(\chi),\min\left(\{\alpha_{21}(\chi)[1]\}\cup\{\beta_{21}(\chi)[1]\}\right)\right\rangle\right)\right);$$

$$\forall\chi\forall\varepsilon\left(\langle\chi,\varepsilon\rangle\in U\times E\right)\rightarrow\left(\gamma_{22}(\chi)(\varepsilon)[1]=\min\left(\{\alpha_{22}(\chi)(\varepsilon)[1]\}\cup\{\beta_{22}(\chi)(\varepsilon)[1]\}\right)\right),$$

где φ может удовлетворять, например, одному из следующих выражений:

$$\varphi\left(\langle\psi,\chi\rangle\right)\underset{def}{=}\min\left(\{\chi\}\cup\left\{\sup\left(\{\psi(\varepsilon)(\varnothing)|\varepsilon\in E\}\right)\right\}\right),$$

$$\varphi\left(\langle\psi,\chi\rangle\right)\underset{def}{=}\chi,$$

$$\varphi\left(\langle\psi,\chi\rangle\right)\underset{def}{=}1.$$

Ситуативным *объединением* ситуативных множеств α и β является ситуативное множество γ, такое, что:

$$\{\alpha\}\cup\{\beta\}\cup\{\gamma\}\subseteq\mu;\qquad \gamma_{1}=\alpha_{1}\cup\beta_{1};$$

$$\forall\chi(\chi\in U)\rightarrow\left(\gamma_{21}(\chi)[1]=\max\left(\{\alpha_{21}(\chi)[1]\}\cup\{\beta_{21}(\chi)[1]\}\right)\right);$$

$$\forall\chi\forall\varepsilon\left(\langle\chi,\varepsilon\rangle\in U\times E\right)\to\left(\gamma_{22}(\chi)(\varepsilon)[1]=\max\left(\left\{\alpha_{22}(\chi)(\varepsilon)[1]\right\}\cup\left\{\beta_{22}(\chi)(\varepsilon)[1]\right\}\right)\right).$$

Ситуативной *разностью* ситуативного множества β из α является ситуативное множество γ, такое, что:

$$\{\alpha\}\cup\{\beta\}\cup\{\gamma\}\subseteq\mu;\qquad \gamma_1=\alpha_1\cup\beta_1;$$

$$\forall\chi(\chi\in U)\to\left(\gamma_{21}(\chi)[1]=\varphi\left(\left\langle\gamma_{22}(\chi),\max\left(\{0\}\cup\left\{\alpha_{21}(\chi)[1]-\beta_{21}(\chi)[1]\right\}\right)\right\rangle\right)\right);$$

$$\forall\chi\forall\varepsilon\left(\langle\chi,\varepsilon\rangle\in U\times E\right)\to\left(\gamma_{22}(\chi)(\varepsilon)[1]=\max\left(\{0\}\cup\left\{\alpha_{22}(\chi)(\varepsilon)[1]-\beta_{22}(\chi)(\varepsilon)[1]\right\}\right)\right).$$

Операции и метаоперации

Операции и метаоперации, соответствующие логическому уровню обработки знаний, представлены в таблицах 22–25.

Таблица 22 – Первичные операции

Название операции	Обозначение	Выражение	Матричное выражение
Пустая	$\varnothing$	$\varnothing$	$m_{\varnothing}\in\{0_Z\}^{M\times M}$
Универсальная	ω	$\omega\underset{def}{=}M\times M$	$m_{\omega}\in\{1_Z\}^{M\times M}$
Тождественная	ι	$\iota\underset{def}{=}\bigcup_{\chi}^{\chi\in M}\{\chi\}\times\{\chi\}$	$m_{\iota}=\left(\frac{\iota}{\omega}\times\{0_Z\}\right)\cup\left(\iota\times\{1_Z\}\right)$

Таблица 23 – Метаоперации

Название метаоперации	Обозначение	Выражение	Матричное выражение
1	2	3	4
Пересечения	$\cap$	$\varphi\cap\psi$	$m_{\varphi\cap\psi}=m_{\varphi}\bullet m_{\psi}$
Элементов	$\varphi\iota$	$\varphi\cap\iota$	$m_{\varphi\cap\iota}=m_{\varphi}\bullet m_{\iota}$
Симметрической разности	$-$	$\varphi-\psi$ $\varphi=\left(\psi-\left(\psi-\varphi\right)\right)$	$m_{\varphi-\psi}=m_{\varphi}\oplus m_{\psi}$
Дополнения до тождественной операции	$\frac{\iota}{\varphi}$	$\iota-\varphi$ $\varphi=\left(\iota-\left(\iota-\varphi\right)\right)$	$m_{\frac{\iota}{\varphi}}=m_{\iota}\oplus m_{\varphi}$
Дополнения до универсальной операции	$\frac{\omega}{\varphi}$	$\omega-\varphi$ $\varphi=\left(\omega-\left(\omega-\varphi\right)\right)$	$m_{\frac{\omega}{\varphi}}=m_{\omega}\oplus m_{\varphi}$
Обращения	φ^{-1}	$\left\{\langle\chi,\gamma\rangle\middle\vert\langle\gamma,\chi\rangle\in\varphi\right\}$	$m_{\varphi^{-1}}=\left(m_{\varphi}\right)^{T}$
Композиции	$\circ$	$\varphi\circ\psi$	$m_{\varphi\circ\psi}=m_{\varphi}\cdot m_{\psi}$
Нормирующей композиции	$[\varphi]$	$\left(\varphi^{-1}\right)\circ\varphi$	$m_{[\varphi]}=\left(m_{\varphi}\right)^{T}\cdot m_{\varphi}$

1	2	3	4
Множества	$\omega\varphi$	$[\omega \circ \varphi]$	$m_{[\omega\circ\varphi]} = m_{\varphi\circ\omega} \cdot m_{\omega\circ\varphi}$
Существования	$\odot$	$[\varphi \circ \psi]$ $\{\gamma \mid \langle \chi, \gamma \rangle \in \varphi \circ \psi\}^2$	$m_{\varphi\odot\psi} = m_{[\varphi\circ\psi]}$
Абсолютного существования	φ	$[\varphi] \cap \iota$ $\{\langle \gamma, \gamma \rangle \mid \langle \chi, \gamma \rangle \in \varphi\}$	$m_{\varphi} = m_{[\varphi]} \bullet m_{\iota}$
Объединения	$\cup$	$\varphi \cup \psi$	$m_{\varphi\cup\psi} = m_{\omega} \oplus \left(m_{\frac{\omega}{\varphi}} \bullet m_{\frac{\omega}{\psi}} \right)$
Кондукции	$\vec{\bullet}$	$\omega - (\varphi \circ (\omega - \psi))$	$m_{\varphi\vec{\bullet}\psi} = m_{\omega} \oplus \left(m_{\varphi} \circ m_{\frac{\omega}{\psi}} \right)$
Индукции	$\overleftarrow{\bullet}$	$\left(\left(\psi^{-1} \right) \vec{\bullet} \left(\varphi^{-1} \right) \right)^{-1}$	$m_{\varphi\overleftarrow{\bullet}\psi} = m_{\omega} \oplus \left(m_{\frac{\omega}{\varphi}} \circ m_{\psi} \right)$ $m_{\varphi\overleftarrow{\bullet}\psi} = \left(m_{\left(\varphi^{-1}\right)\vec{\bullet}\left(\psi^{-1}\right)} \right)^{T}$
Бикондукции	$\overleftrightarrow{\bullet}$	$(\varphi \vec{\bullet} \psi) \cup (\varphi \overleftarrow{\bullet} \psi)$	$m_{\varphi\overleftrightarrow{\bullet}\psi} = m_{\omega} \oplus \left(m_{\frac{\omega}{\varphi\vec{\bullet}\psi}} \bullet m_{\frac{\omega}{\varphi\overleftarrow{\bullet}\psi}} \right)$
Бииндукции	$\overline{\bullet}$	$(\varphi \vec{\bullet} \psi) \cap (\varphi \overleftarrow{\bullet} \psi)$	$m_{\varphi\overline{\bullet}\psi} = m_{\varphi\vec{\bullet}\psi} \bullet m_{\varphi\overleftarrow{\bullet}\psi}$
Кондуктивной композиции [22]	$\vec{\circ}$	$(\varphi \circ \psi) \cap (\varphi \vec{\bullet} \psi)$	$m_{\varphi\vec{\circ}\psi} = m_{\varphi\circ\psi} \bullet m_{\varphi\vec{\bullet}\psi}$
Индуктивной композиции	$\overleftarrow{\circ}$	$(\varphi \circ \psi) \cap (\varphi \overleftarrow{\bullet} \psi)$	$m_{\varphi\overleftarrow{\circ}\psi} = m_{\varphi\circ\psi} \bullet m_{\varphi\overleftarrow{\bullet}\psi}$
Бикондуктивной композиции	$\overleftrightarrow{\circ}$	$(\varphi \circ \psi) \cap (\varphi \overleftrightarrow{\bullet} \psi)$	$m_{\varphi\overleftrightarrow{\circ}\psi} = m_{\varphi\circ\psi} \bullet m_{\varphi\overleftrightarrow{\bullet}\psi}$
Бииндуктивной композиции	$\overline{\circ}$	$(\varphi \circ \psi) \cap (\varphi \overline{\bullet} \psi)$	$m_{\varphi\overline{\circ}\psi} = m_{\varphi\circ\psi} \bullet m_{\varphi\overline{\bullet}\psi}$
Всеобщности	$\otimes$	$[\varphi \overleftarrow{\circ} \psi]$	$m_{\varphi\otimes\psi} = m_{[\varphi\overleftarrow{\circ}\psi]}$

Если задана метрика ρ над аргументами операций, то между такими операциями можно ввести метрику

$$\delta(\langle \varphi, \psi \rangle) \underset{def}{=} \max\left(\{0\} \cup \left\{ \rho(\langle \chi, \gamma \rangle) \mid \langle \chi, \gamma \rangle \in \varphi - \psi \right\} \right).$$

На основе введённой метрики и некоторой нейтральной операции ψ, например $\varnothing$, можно ввести меру σ_{ψ}, равную $\sigma_{\psi}(\varphi) \underset{def}{=} \delta(\langle \varphi, \varnothing \rangle)$.

Таблица 24 – Отношения над операциями

Название	Требование
Равенство	$\varphi = \psi$
Включение	$\varphi = \varphi \cap \psi$
Вложенная композиция	$\varphi \circ \psi = \varphi \cap (\varphi \circ \psi)$ $\varphi \circ \psi = \psi \cap (\varphi \circ \psi)$
Одноэлементность	$(\neg(\varnothing = \varphi\iota)) \wedge (\varphi = \omega\varphi)$
Транзитивность	$\varphi^{\circ} = \bigcap_{\psi}^{\psi=(\varphi\cup(\psi\circ\varphi))} \psi$ $\varphi = \varphi^{\circ}$
Симметричность	$\varphi = \varphi^{-1}$
Антисимметричность	$\varphi\iota = \varphi^{-1} \cap \varphi$
Рефлексивность	$\iota = \varphi\iota$
Иррефлексивность	$\varnothing = \varphi\iota$
Полнота	$\omega = \varphi^{-1} \cup \varphi$
Функциональность	$[\varphi] = \varphi \ ; (\varphi^{-1} \circ \varphi) = (\varphi^{-1} \circ \varphi) \cap \iota$

Таблица 25 – Соответствие отношений типам дескрипционных логик [34]

Название операции или отношения		Тип дескрипционной логики				
		FL	AL	ALC	S	иной
Пересечения		ι	ι	ι	ι	–
Абсолютного существования		ι	ι	ι	ι	–
Всеобщности		ι	ι	ι	ι	–
Пустая		–	ι	ι	ι	–
Тождественная		–	ι	ι	ι	–
Дополнения до тождественной операции		–	*	ι	ι	C
Существования		–	–	ι	ι	E
Дополнения до универсальной операции		–	–	–	–	¬
Композиции		–	–	–	–	∘
Обращения		–	–	–	–	I
Объединения		–	–	–	–	U
Множества		–	–	–	–	O
Одноэлементности		–	–	–	–	O
Включения		–	–	–	–	H
Вложенной композиции		–	–	–	–	R
Транзитивности		–	–	–	ω	S
Иррефлексивности		–	–	–	–	R
Функциональности		–	–	–	–	F

Задания для лабораторных работ

Задача вычисления матрицы значений

Дано: сгенерированные матрицы ***A***, ***B***, ***E***, ***G*** заданных размерностей ***p*×*m***, ***m*×*q***, ***1*×*m***, ***p*×*q*** соответственно со значениями в рекомендуемом диапазоне [–1;1]:

$$c_{ij} = \underset{k}{\tilde{\wedge}} f_{ijk} \cdot \left(3 \cdot g_{ij} - 2\right) \cdot g_{ij} + \left(\underset{k}{\tilde{\vee}} d_{ijk} + \left(4 \cdot \left(\underset{k}{\tilde{\wedge}} f_{ijk} \mathbin{\tilde{\circ}} \underset{k}{\tilde{\vee}} d_{ijk} \right) - 3 \cdot \underset{k}{\tilde{\vee}} d_{ijk} \right) \cdot g_{ij} \right) \cdot \left(1 - g_{ij}\right),$$

$$f_{ijk} = \left(a_{ik} \mathbin{\tilde{\to}} b_{kj}\right) \cdot \left(2 \cdot e_k - 1\right) \cdot e_k + \left(b_{kj} \mathbin{\tilde{\to}} a_{ik}\right) \cdot \left(1 + \left(4 \cdot \left(a_{ik} \mathbin{\tilde{\to}} b_{kj}\right) - 2\right) \cdot e_k\right) \cdot \left(1 - e_k\right),$$

$$d_{ijk} = a_{ik} \mathbin{\tilde{\wedge}} b_{kj}.$$

Получить: ***C*** – матрицу значений соответствующей размерности ***p*×*q*** в соответствии с вариантами (таблицы 26 и 27).

Таблица 26 – Варианты операций композиции и импликации

Вариант	$\chi \mathbin{\tilde{\circ}} \gamma$	$\chi \mathbin{\tilde{\to}} \gamma$
1	$\min\left(\{\chi\} \cup \{\gamma\}\right)$	$\max\left(\{1_Z - \chi\} \cup \{\gamma\}\right)$
2	$\chi \cdot \gamma$	$\chi \cdot \left(\gamma - 1_Z\right) + 1_Z$
3	$\max\left(\{\chi + \gamma - 1_Z\} \cup \{0_Z\}\right)$	$\min\left(\{1_Z - \chi + \gamma\} \cup \{1_Z\}\right)$
4	–	$\sup\left(\left\{\delta \middle\| \left(\min\left(\{1 - \chi\} \cup \{\delta\}\right) \le \gamma\right) \wedge \left(\delta \le 1_Z\right)\right\}\right)$
5	–	$\sup\left(\left\{\delta \middle\| \left((1 - \chi) \cdot \delta \le \gamma\right) \wedge \left(\delta \le 1_Z\right)\right\}\right)$
6	–	$\sup\left(\left\{\delta \middle\| \left(\min\left(\{1 - \chi + \delta\} \cup \{0\}\right) \le \gamma\right) \wedge \left(\delta \le 1_Z\right)\right\}\right)$

Таблица 27 – Варианты лабораторных заданий

Вариант	$\underset{k}{\tilde{\wedge}} f_{ijk}$	$\underset{k}{\tilde{\vee}} d_{ijk}$	$\underset{k}{\tilde{\wedge}} f_{ijk} \mathbin{\tilde{\circ}} \underset{k}{\tilde{\vee}} d_{ijk}$	$a_{ik} \mathbin{\tilde{\to}} b_{kj}$	$b_{kj} \mathbin{\tilde{\to}} a_{ik}$	$a_{ik} \mathbin{\tilde{\wedge}} b_{kj}$
1	2	2	2	2	2	2
2	2	2	2	5	5	2
3	2	2	1	2	2	2
4	2	2	1	5	5	2
5	2	2	3	2	2	2
6	2	2	3	5	5	5
7	2	2	2	1	1	1
8	2	2	2	4	4	1
9	2	2	1	1	1	1
10	2	2	1	4	4	1
11	2	2	3	1	1	1
12	2	2	3	4	4	1
13	2	2	2	3	3	3
14	2	2	2	6	6	3
15	2	2	1	3	3	3
16	2	2	1	6	6	3
17	2	2	3	3	3	3
18	2	2	3	6	6	3

Вопросы

1. Чем является граф сети Петри?
2. Чем являются в сети Петри: позиции, переходы?
3. Что является разметкой сети Петри?
4. Чему приписываются метки сети Петри?
5. Что соединяют рёбра сети Петри?
6. Как осуществляется срабатывание переходов сети Петри?
7. Что может вызываться срабатыванием переходов сети Петри?
8. Что необходимо для срабатывания перехода сети Петри?
9. Что является результатом срабатывания перехода сети Петри.
10. Если в сети Петри могут сработать два перехода, то какие варианты возможны?
11. Чем могут быть обозначены программные ресурсы в сети Петри?
12. Чем отличается универсальная сеть Петри от классической сети Петри?
13. Как называется несколько вложенных друг в друга циклов?
14. Как называется пространство, имеющее элементы, взаимно однозначно соответствующие итерациям совершенного гнезда циклов?
15. При каком достаточном и необходимом условии в гнезде циклов *i*-й цикл параллелизуем?
16. Как называется значение разности между набором координат в итерационном пространстве итерации исполнения команды текущей операции и набором координат в итерационном пространстве итерации исполнения команды, зависимой от команды текущей операции?
17. Как называется преобразование цикла в цикл с изменённым направлением пересечения итерационного пространства?
18. Как называется преобразование, обратное распределению цикла?
19. Как называется преобразование, изменяющее порядок вложенных циклов в гнезде без изменения их количества?
20. В каких случаях уместна инверсия задач?
21. В каком случае ситуативное множество является ситуативным подмножеством ситуативного множества?
22. Какое множество является ситуативным пересечением (объединением, разностью) ситуативных множеств?
23. Что такое: признак, характеристическая функция?
24. Какие варианты операций для пересечения над характеристическими функциями вам известны?
25. Может ли операция являться языком?
26. Приведите примеры, когда результаты разных операций композиции над одними и теми же аргументами различны.

Список использованных источников

1. Воеводин, В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб. : БХВ-Петербург, 2002. – 608 с.
2. Воеводин, В. В. Вычислительная математика и структура алгоритмов / В. В. Воеводин. – М. : МГУ, 2006. – 112 с.
3. Головкин, Б. А. Параллельные вычислительные системы / Б. А. Головкин. – М. : Наука, 1980. – 520 с.
4. СТБ ГОСТ 7.0–2004 (ГОСТ 7.0–99, IDT) СИБИД. Информационно-библиотечная деятельность, библиография. Термины и определения.
5. Зайцев, Д. А. Уравнение состояний и эквивалентные преобразования временных сетей Петри / Д. А. Зайцев, А. И. Слепцов // Кибернетика и системный анализ. – 1997. – №5. – С. 59–76.
6. Иванов, В. А. Теория дискретных систем автоматического управления : учеб. пособие / В. А. Иванов, А. С. Ющенко. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2015. – 348 с.
7. Ивашенко, В. П. Онтологическая модель пространственно-временных отношений событий и явлений в процессах обработки знаний / В. П. Ивашенко // Вестник БрГТУ. – 2017. – №5(107). – С. 13–17.
8. Ивашенко, В. П. Модели обработки информации в интеллектуальных системах, основанных на семантических технологиях / В. П. Ивашенко, А. С. Бельчиков, А. П. Еремеев // Информационные технологии и системы 2016 (ИТС 2016) : материалы междунар. науч. конф., Минск, 26 октября 2016 г. / Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2016. – С. 106–107.
9. Ивашенко, В. П. Распределение памяти в неограниченном линейном адресном пространстве / В. П. Ивашенко // Информационные технологии и системы 2019 (ИТС 2019) : материалы междунар. науч. конф., Минск, 30 октября 2019 г. / Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2019. – С. 110–111.
10. Ивашенко, В. П. Модели и алгоритмы интеграции знаний на основе однородных семантических сетей / В. П. Ивашенко // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г. / Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2015. – С. 111–132.
11. Ивашенко, В. П. Операции управления массивами данных в линейно адресуемой памяти / В. П. Ивашенко, С. В. Синцов // Доклады БГУИР. – 2016. – №10. – С. 86–93.

12. Ивашенко, В. П. Принципы платформенной независимости и платформенной реализации OSTIS / В. П. Ивашенко, М. М. Татур // Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г. / Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.]. – Минск : БГУИР, 2016. – С. 145–150.
13. Карцев, М. А. Вычислительные системы и синхронная арифметика / М. А. Карцев, В. А. Брик. – М. : Радио и связь, 1981. – 360 с.
14. Мараховский, В. Б. Моделирование параллельных процессов. Сети Петри. Курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления / В. Б. Мараховский, Л. Я. Розенблюм, А. В. Яковлев. – СПб. : Профессиональная литература, АйТи-Подготовка, 2014. – 400 с.
15. Питерсон, Дж. Теория сетей Петри и моделирование систем / Дж. Питерсон. – М. : Мир, 1984. – 264 с.
16. Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг ; пер. с англ. – 2-е изд. – М. : ООО «И. Д. Вильямс», 2015. – 1408 с.
17. Смит, Б. Методы и алгоритмы вычислений на строках / Б. Смит; пер. с англ. – М. : ООО «И. Д. Вильямс», 2006. – 496 с.
18. Цилькер, Б. Я. Организация ЭВМ и систем : учебник для вузов / Б. Я. Цилькер, С. А. Орлов. – 2-е изд. – СПб. : Питер, 2011. – 688 с.
19. Шпаковский, Г. И. Реализация параллельных вычислений: кластеры, многоядерные процессоры, грид, квантовые компьютеры. – Минск : БГУ, 2011. – 176 с.
20. Шпаковский, Г. И. Организация параллельных ЭВМ и суперскалярных процессоров : учеб. пособие / Г. И. Шпаковский. – Минск : БГУ, 1996. – 296 с.
21. Шпаковский, Г. И. Программирование для многопроцессорных систем в стандарте MPI / Г. И. Шпаковский, Н. В. Серикова. – Минск : БГУ, 2002. – 323 с.
22. Backhouse, R. Demonic operators and monotype factors / R. Backhouse, J. van der Woude // Mathematical Structures in Computer Science. – 1993. – 3(4). – P. 417–433.
23. Flynn, M. J. Some Computer Organizations and Their Effectiveness / M. J. Flynn // Computers, IEEE Transactions on. – 1972. – V. C-21 (9). – P. 948–960.
24. Ivashenko, V. P. Semantic logging of knowledge processing based on binary generated events / V. P. Ivashenko // Pattern Recognition and Information Processing (PRIP 2019) : Proceedings of the 14th International Conference, Minsk, 21–23 May 2019. – Minsk, 2019. – P. 172–177.
25. Ivashenko, V. P. String processing model for knowledge-driven systems / V. P. Ivashenko // Доклады БГУИР. – 2020. – Том. 18, №6. – P. 33–40.
26. Kshemkalyani, A. D. Distributed Computing: Principles, Algorithms, and Systems. / A. D. Kshemkalyani, M. Singhal. – Cambridge University Press, 2011. – 756 p.

27. MPI Forum [Электронный ресурс]. – Режим доступа : https://www.mpi-forum.org/. – Дата доступа : 12.02.2020.
28. OpenCL Overview – The Khronos Group Inc [Электронный ресурс]. – Режим доступа : https://www.khronos.org/opencl. – Дата доступа : 12.02.2020.
29. AMD APP SDK OpenCL™ User Guide [Электронный ресурс]. – Режим доступа : http://developer.amd.com/wordpress/media/2013/12/AMD_OpenCL_Programming_User_Guide2.pdf. – Дата доступа : 12.02.2020.
30. PARALLEL.RU – Информационно аналитический центр по параллельным вычислениям [Электронный ресурс]. – Режим доступа : https://parallel.ru. – Дата доступа : 12.02.2020.
31. Parberry, I. The Pairwise Sorting Network / I. Parberry // Parallel Processing Letters, – Vol. 2, №2, 3. – 1992. – P. 205–211.
32. Ramamoorthy, C. V. Pipeline Architecture / C. V. Ramamoorthy, H. F. Li // ACM Computing Surveys. – 1977. – №9, 1. – P. 61–102.
33. Rado, R. Universal graphs and universal functions / R. Rado // Acta Arith. – 1964. – Vol. 9. – P. 331–340.
34. The Description Logic Handbook: Theory, Implementation, and Applications. – Cambridge University Press, 2003. – 587 p.
35. Viotti, P. Consistency in Non-Transactional Distributed Storage Systems / P. Viotti, M. Vukolic // ACM Computing Surveys. – 2016. – 49 (1): 19:1–19:34.
36. Vishkin, U. Thinking in Parallel: Some Basic Data-Parallel Algorithms and Techniques, 104 pages. These class notes are meant to make the fundamental theory of parallel algorithms accessible to computer science and engineering graduate students. They have also been tested successfully for upper division undergraduate students. [Электронный ресурс]. – Режим доступа : http://users.umiacs.umd.edu/~vishkin/PUBLICATIONS/classnotes.pdf. – Дата доступа : 12.02.2020.
37. Zaitsev, D. A. Universal Inhibitor Petri Net / D. A. Zaitsev // Proceedings of the 17-th German Workshop on Algorithms and Tools for Petri Nets, 7–8 October. – Cottbus, Germany. – 2010. – P. 1–15.

Св. план 2020, поз. 13

Учебное издание

МОДЕЛИ РЕШЕНИЯ ЗАДАЧ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

В двух частях
Часть 1

В. П. Ивашенко

Формальные модели обработки информации и параллельные модели решения задач

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 02.12.2020. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 4,77. Уч.-изд. л. 4,8. Тираж 50 экз. Заказ 218.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, Минск