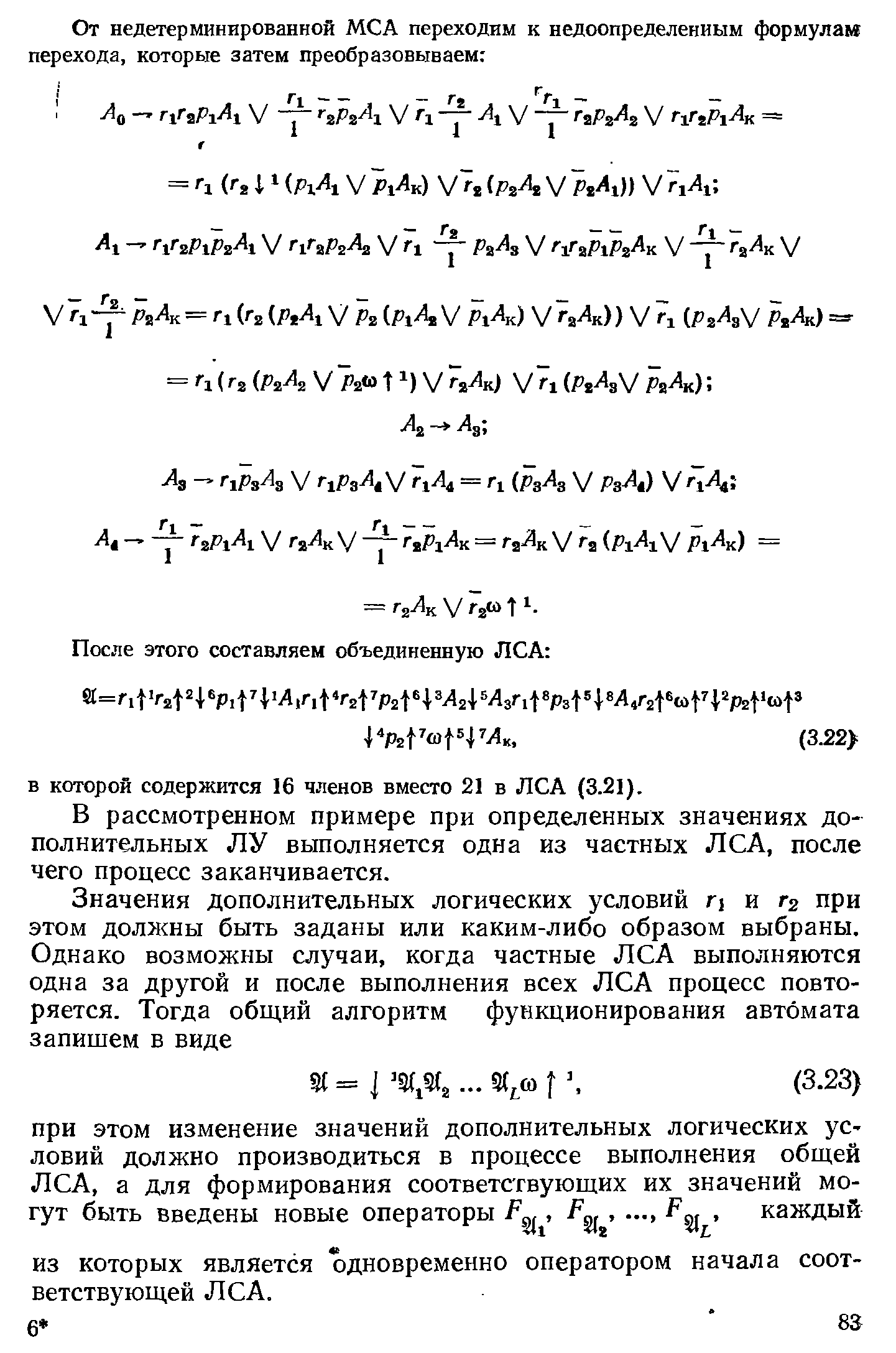
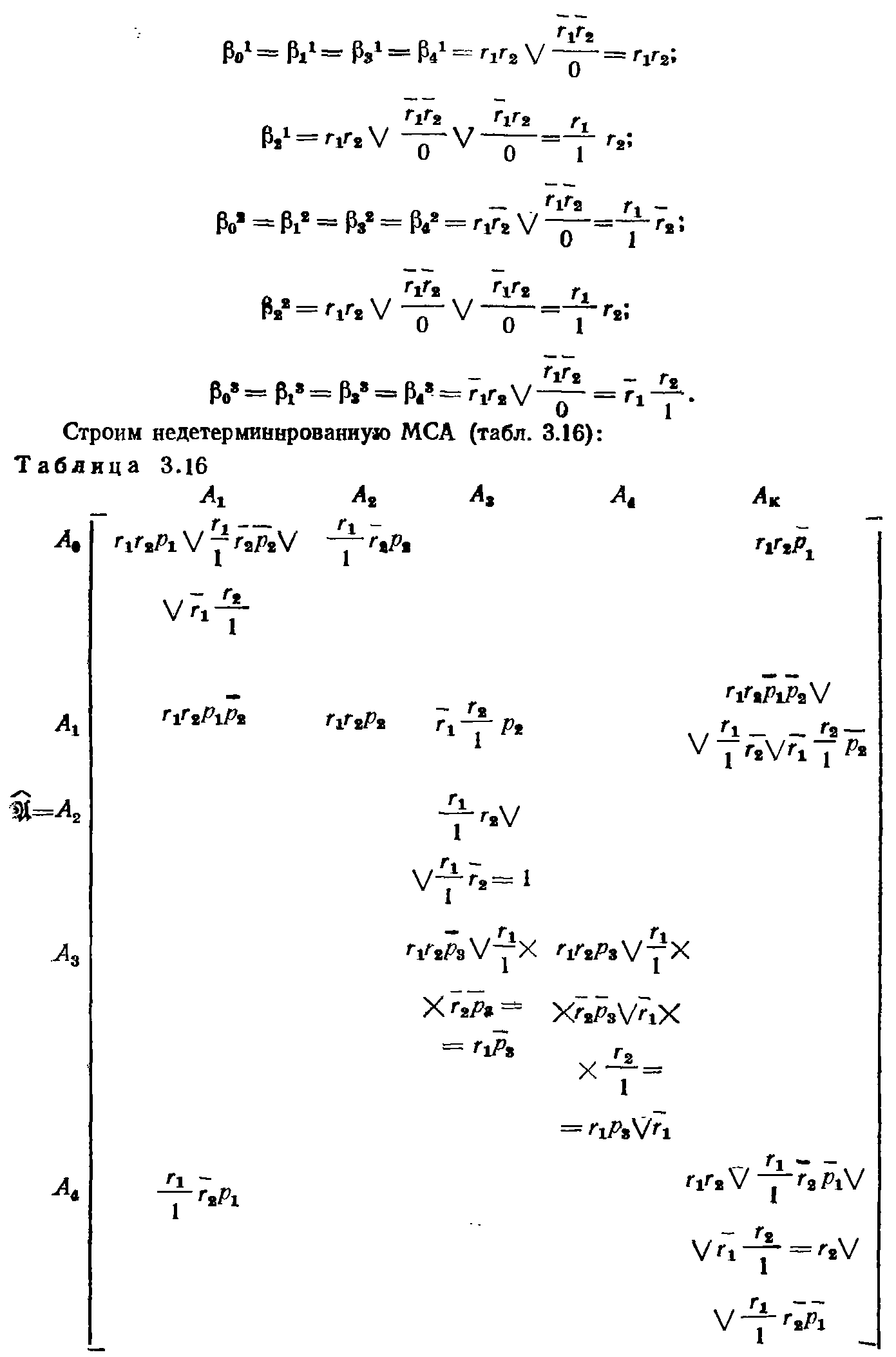
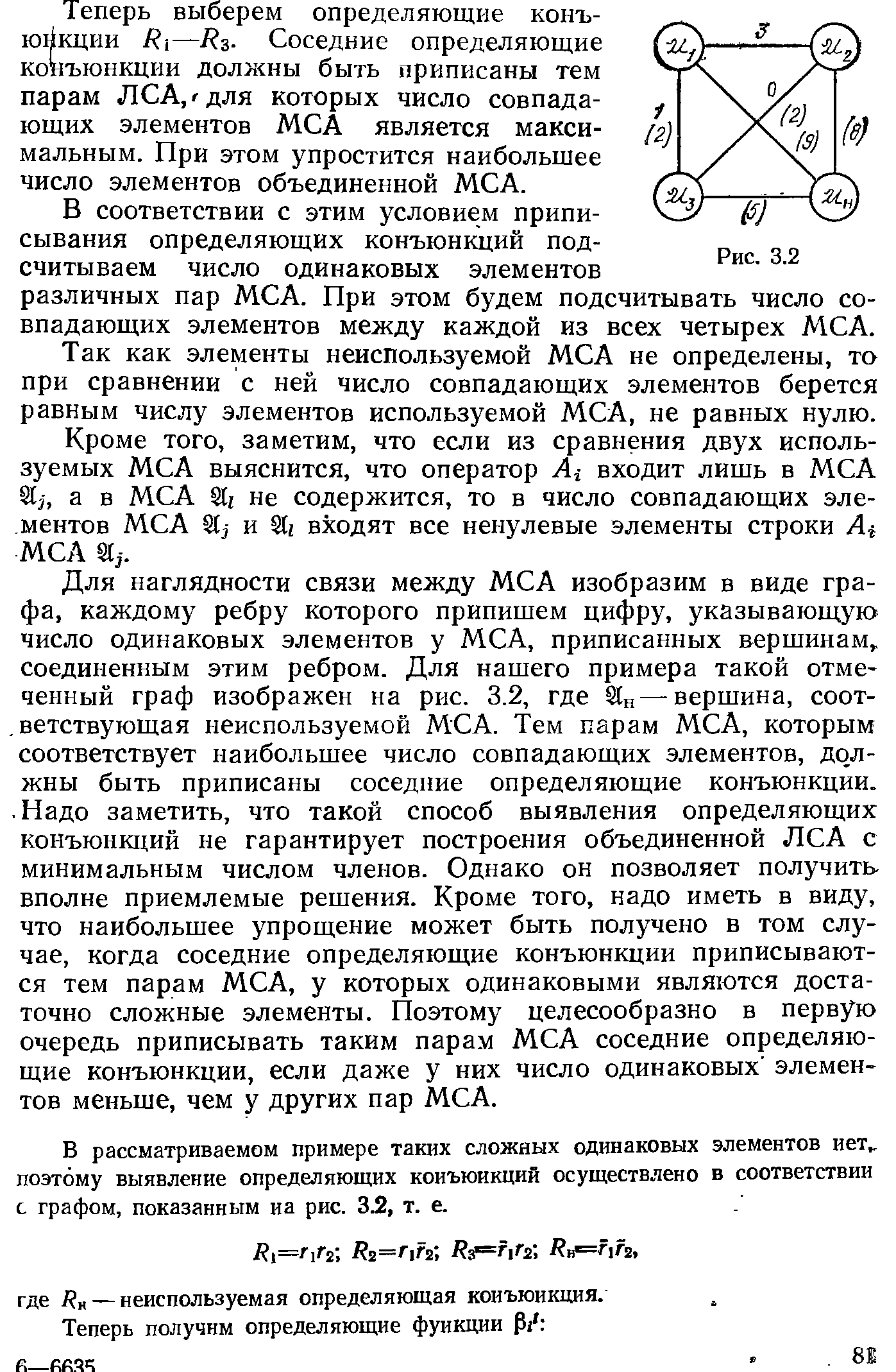
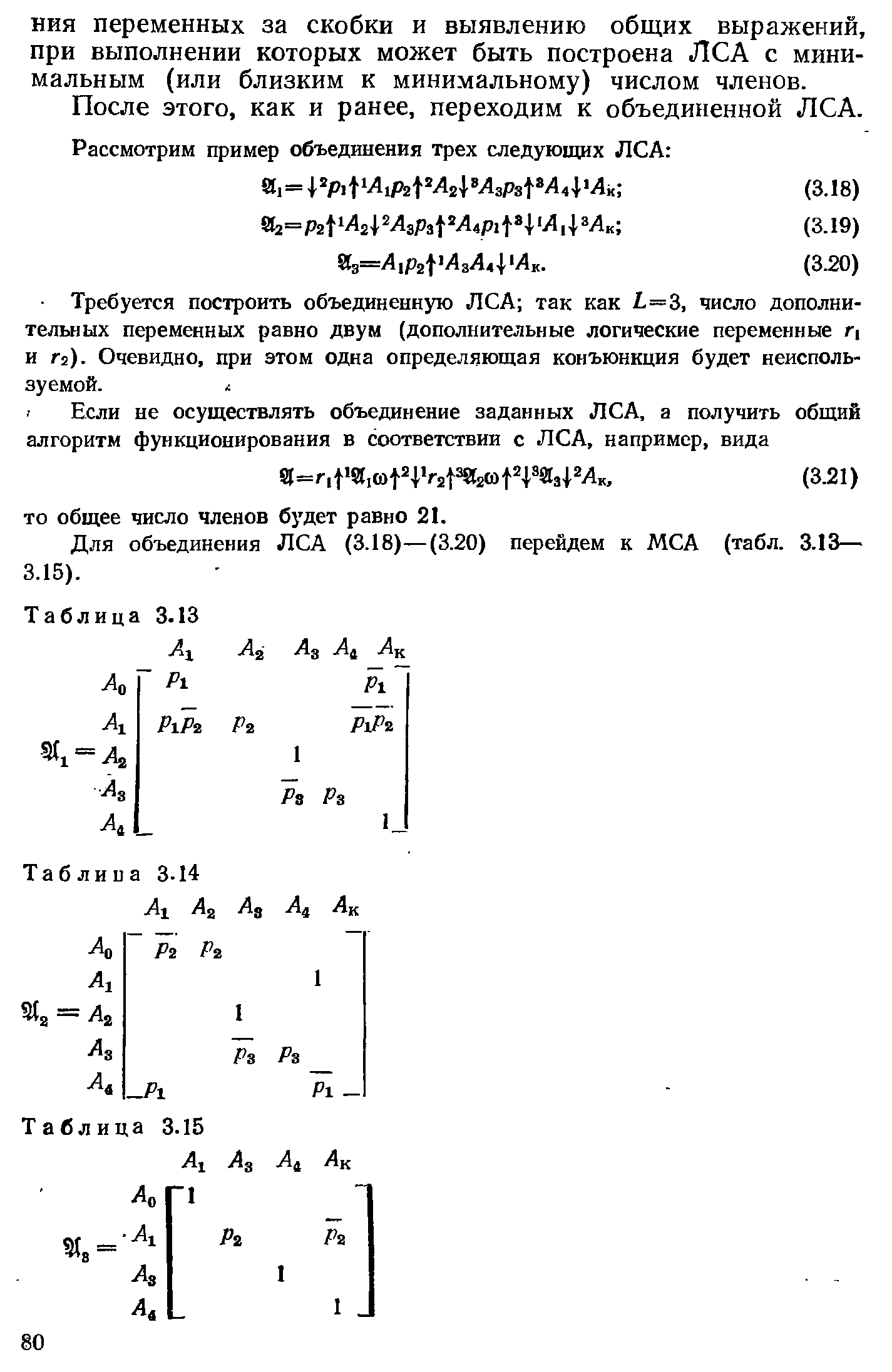
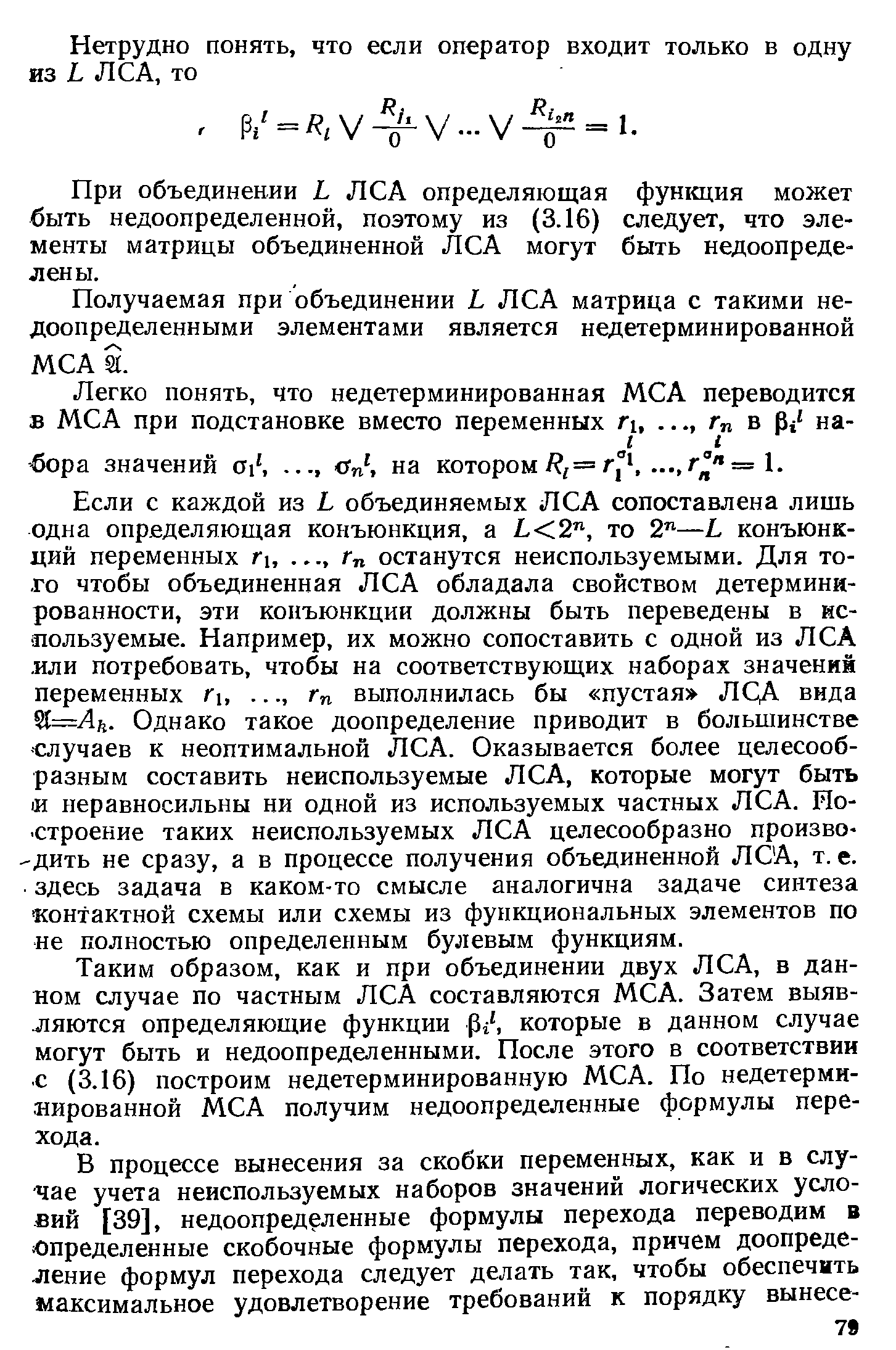
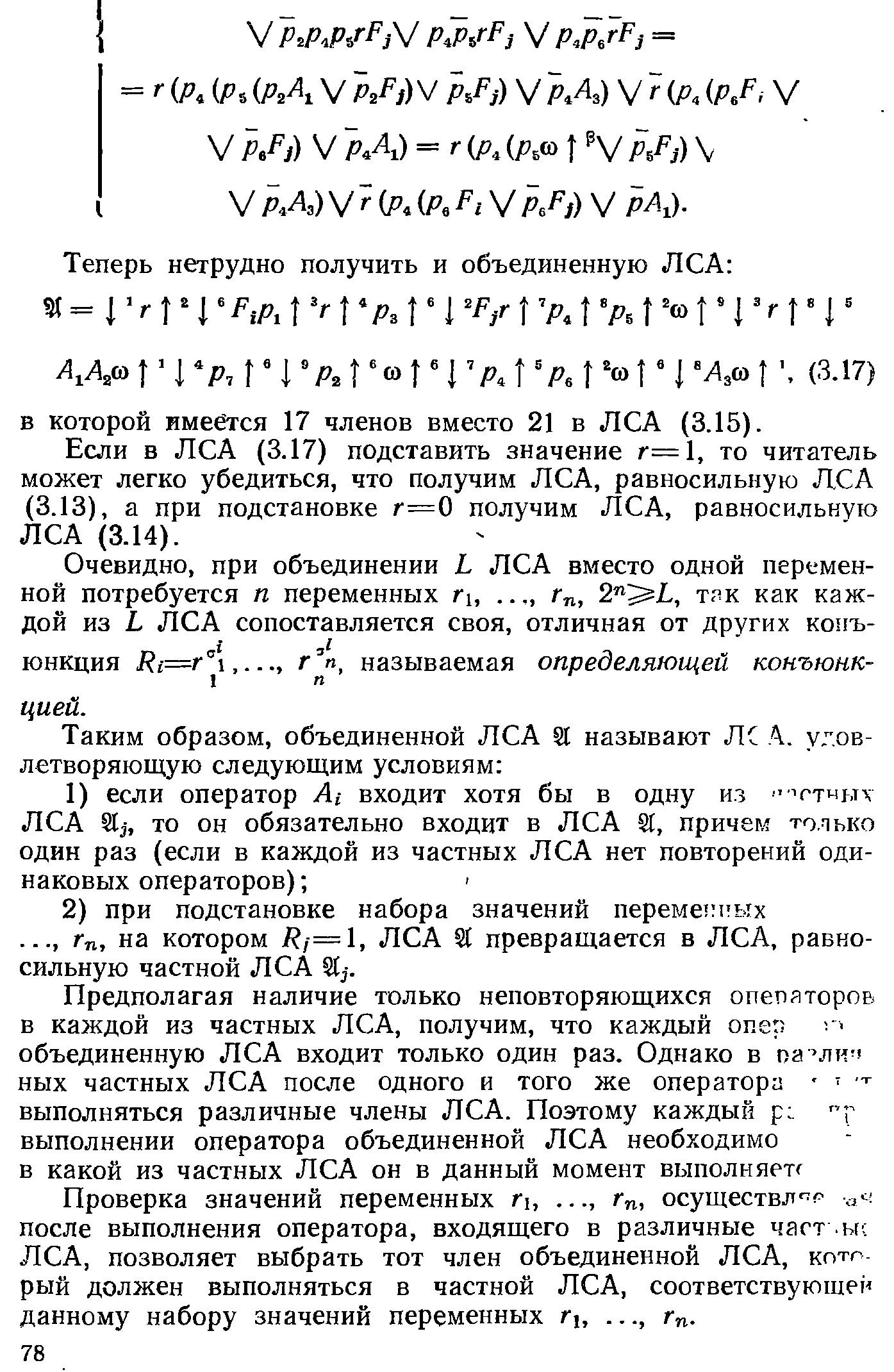
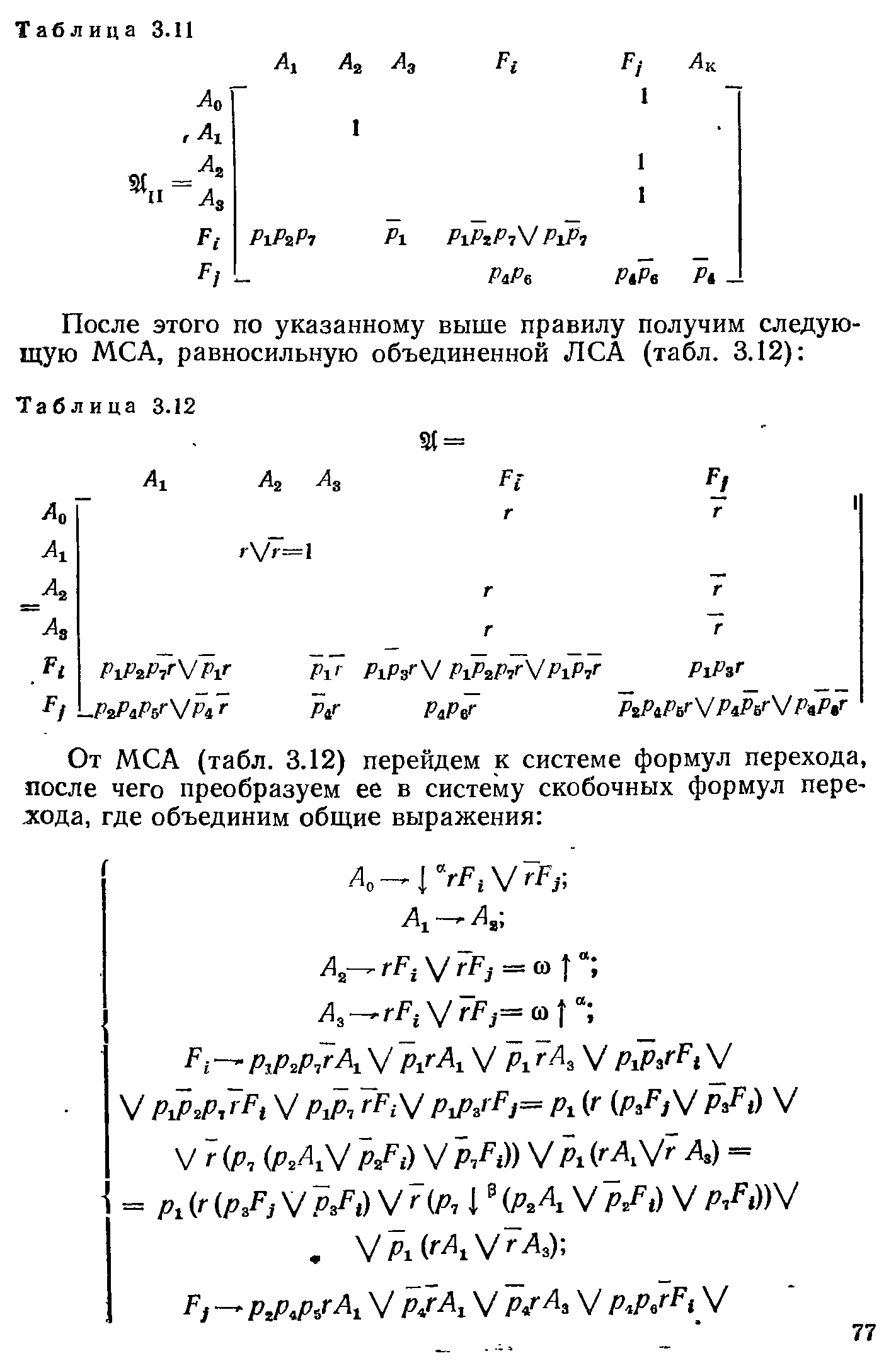
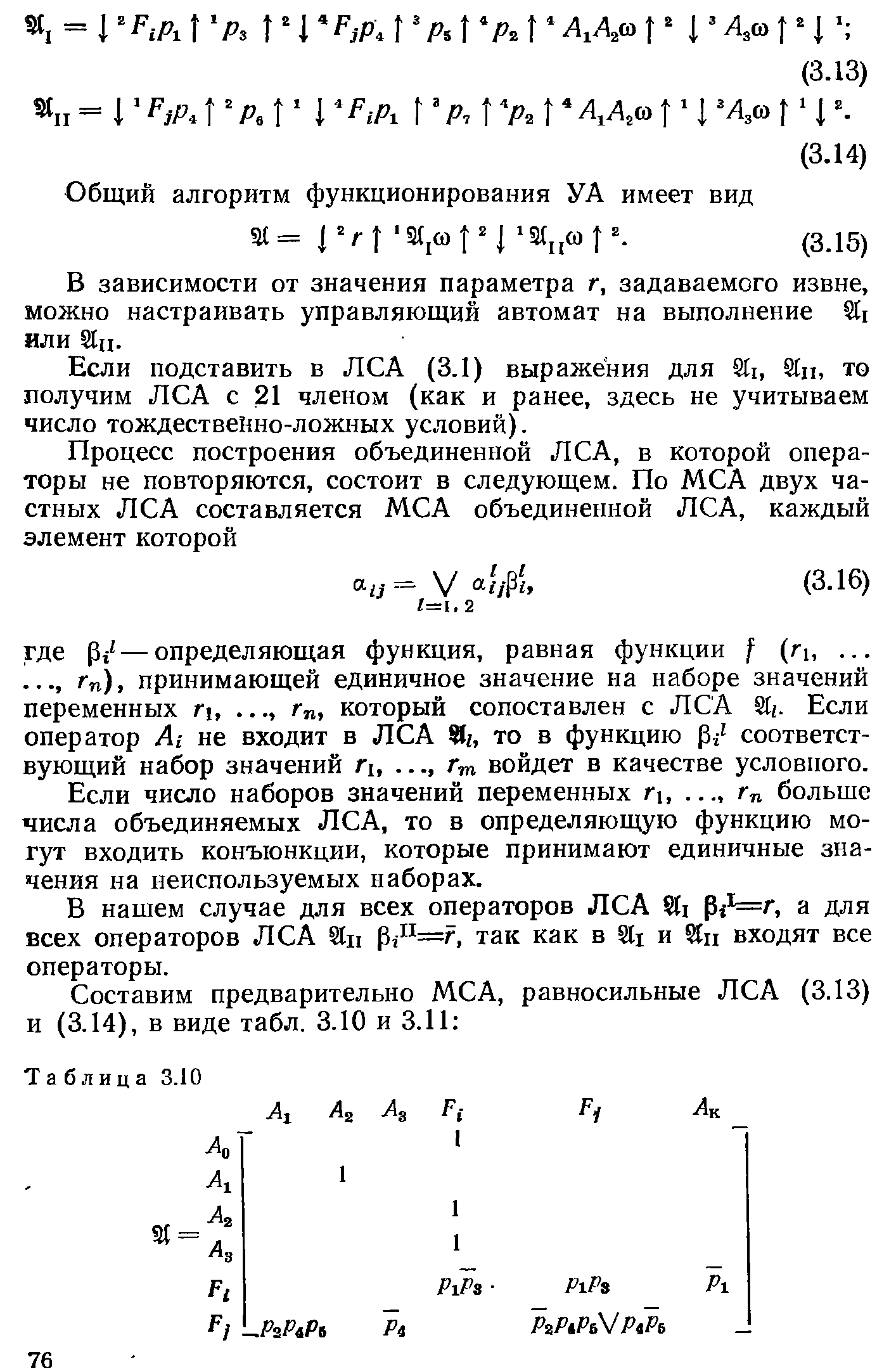
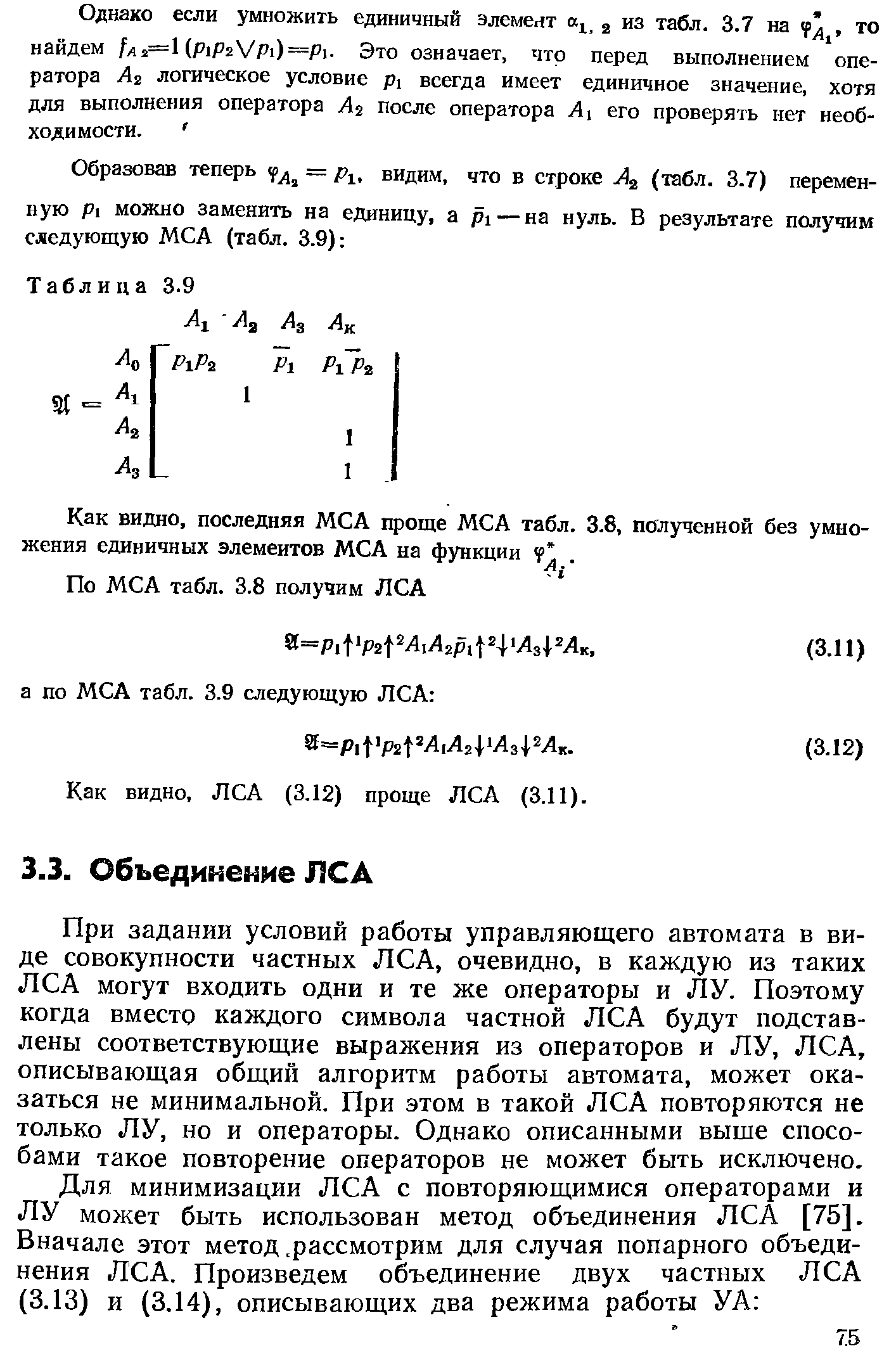
30 Объединение схем алгоритмов. Определение конъюнкции. Кодирование схем.



Это определяющие конъюнкции. Их получают методом парных

кодов, т.е. тем алгоритмам у которых больше всего совпадающих

ячеек в таблицах МСА присваем парные коды. Например тут у R2  и

R3 коды парные.

Определяющие функции. Несколько случаев для их получения:

- Все блоки всех алгоритмов одинаковые – берём определющую конъюнкцию

Алгоритма для которого ищем определющие функции и дизъюнктируем её с опр.

Кон. Пустого алгоритма делённого на нуль.

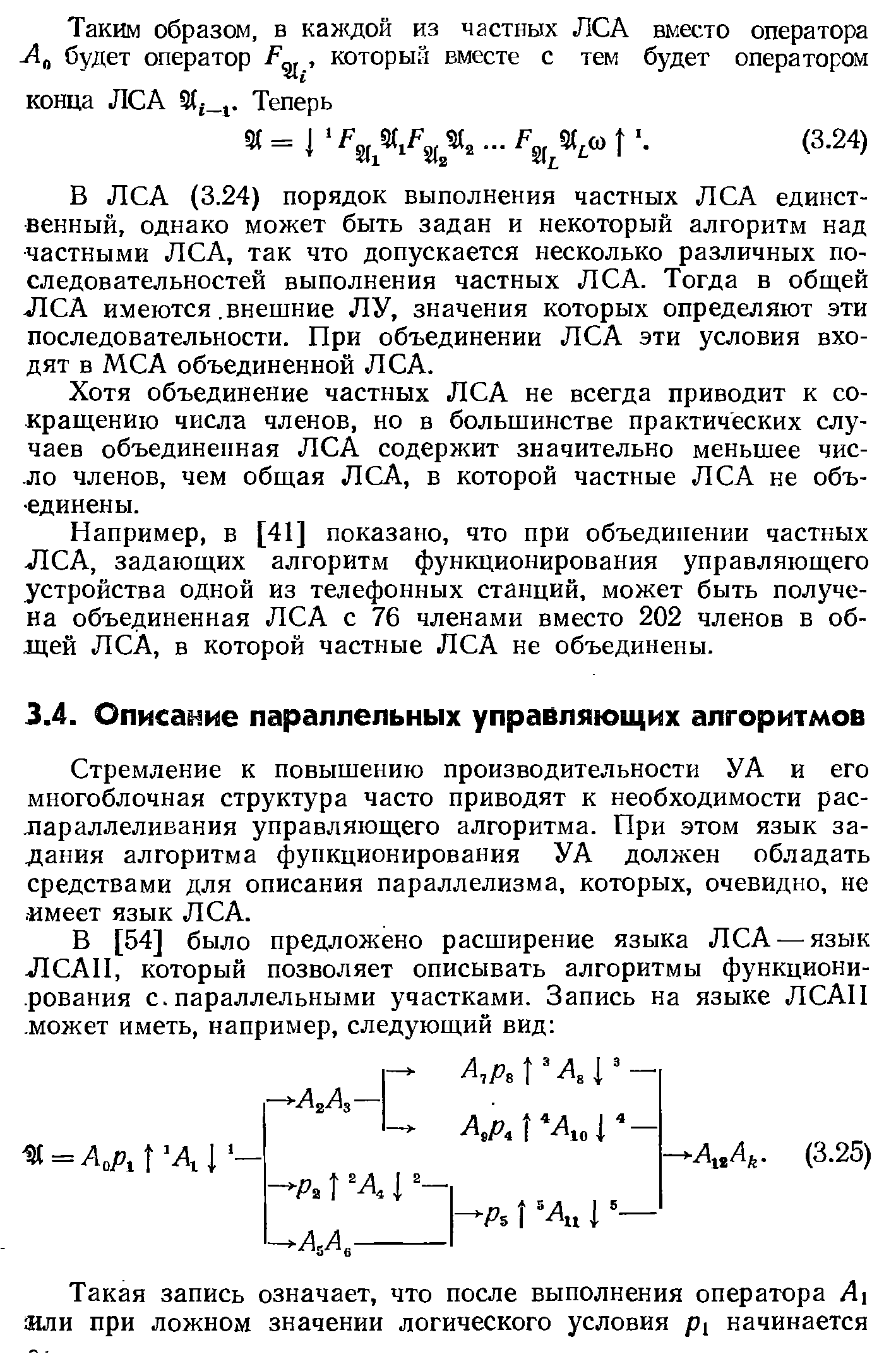
- есть один «уникальный» блок – для него опр.ф. будет 1, т.к. мы дизъюнктируем

опр.кон. алгоритма в котором есть этот «уникальный» блок со всеми опр.кон

делёнными на нуль.

- данный блок есть в двух алгоритмах, но его нет в третьем – дизъюнктируем опр.

кон. Алгоритмов в которых блок есть с опр.кон. пустого алгоритма дел-ым на нуль.



31 Определяющие функции. Процесс доопределения.

Опр.ф. -- в предыдущем вопросе выделены синим.

Как доопределять?

Вот дали тебе на объединение, допустим, три алгоритма. Ты уже построила по ним по отдельности МСА, по этим МСА в виде графа построила схему взаимосвязи между алгоритмами (т.е. где мы считаем у каких алгоритмов больше всего совпадающих ячеек), с помощью этого графа сделала определяющие конъюнкции, потом сделала определяющие функции и наконец построила объединённую МСА (далее ОМСА). Вот теперь и начинается магия. Процесс доопределения прост. Посмотри на таблицу 3.16 из предыдущего вопроса. Там есть записи типа: «r1/1<и далее по тексту>», так вот, доопределение состоит в том, что мы выбираем, что оставить в записи – «r1» или «1». Определённого алгоритма выбора нет, поэтому тут всё зависит от твоих действий. Главное выбирать так, чтобы в конечном итоге алгоритм оказался наиболее компактным. И если ты в одной строке выбираешь из, допустим, !r2/1 (! – тут знак отрицания) 1, то желательно во всей ОМСА выбирать также, но следует помнить, что если в строке есть r1, то должно быть и !r1, если есть !r2, то должно быть и r2.

32 Алгоритмически неразрешимые проблемы. Примеры.

Алгоритмическая проблема — это проблема, в которой требуется найти единый метод (алгоритм) для решения бесконечной серии однотипных единичных задач. Такие проблемы называют также массовыми проблемами. Они возникали и решались в различных областях математики на протяжении всей ее истории. Примеры таких проблем рассматривались ранее.

Математики в начале XX в. столкнулись с тем, что для некоторых массовых проблем не удается подобрать общий алгоритм для их решения. В связи с этим возникла необходимость дать точное определение самому понятию алгоритма.

**Алгоритмически неразрешимые проблемы**

Алан Тьюринг высказал предположение (известное как Тезис Чёрча — Тьюринга), что любой алгоритм в интуитивном смысле этого слова может быть представлен эквивалентной машиной Тьюринга. Уточнение представления о вычислимости на основе понятия машины Тьюринга (и других эквивалентных ей понятий) открыло возможности для строгого доказательства алгоритмической неразрешимости различных массовых проблем (то есть проблем о нахождении единого метода решения некоторого класса задач, условия которых могут варьироваться в известных пределах). Простейшим примером алгоритмически неразрешимой массовой проблемы является так называемая проблема применимости алгоритма (называемая также проблемой остановки). Она состоит в следующем: требуется найти общий метод, который позволял бы для произвольной машины Тьюринга (заданной посредством своей программы) и произвольного начального состояния ленты этой машины определить, завершится ли работа машины за конечное число шагов, или же будет продолжаться неограниченно долго.

В течение первого десятилетия истории теории алгоритмов неразрешимые массовые проблемы были обнаружены лишь внутри самой этой теории (сюда относится описанная выше проблема применимости), а также внутри математической логики (проблема выво-димости в классическом исчислении предикатов). Поэтому считалось, что теория алгоритмов представляет собой обочину математики, не имеющую значения для таких её классических разделов, как алгебра или анализ. Положение изменилось после того, как А. А. Марков и Э. Л. Пост в 1947 году установили алгоритмическую неразрешимость известной в алгебре проблемы равенства для конечнопорождённых и конечноопределённых полугрупп (т. н. проблемы Туэ). Впоследствии была установлена алгоритмическая неразрешимость и многих других «чисто математических» массовых проблем. Одним из наиболее известных результатов в этой области является доказанная Ю. В. Матиясевичем алгоритмическая неразрешимость десятой проблемы Гильберта.

*Алгоритмически неразрешимые проблемы*

За время своего существования человечество придумало множество алгоритмов для решения разнообразных практических и научных проблем. Зададимся вопросом – а существуют ли какие-нибудь проблемы, для которых невозможно придумать алгоритмы их решения?

Утверждение о существовании алгоритмически неразрешимых проблем является весьма сильным – мы констатируем, что мы не только сейчас на знаем соответствующего алгоритма, но мы не можем принципиально никогда его найти.

Успехи математики к концу XIX века привели к формированию мнения, которое выразил Д. Гильберт – «в математике не может быть неразрешимых проблем», в связи с этим формулировка проблем Гильбертом на конгрессе 1900 года в Париже была руководством к действию, констатацией отсутствия решений в данный момент.

Первой фундаментальной теоретической работой, связанной с доказательством алгоритмической неразрешимости, была работа Курта Гёделя – его известная теорема о не-полноте символических логик. Это была строго формулированная математическая проблема, для которой не существует решающего ее алгоритма. Усилиями различных исследователей список алгоритмически неразрешимых проблем был значительно расширен. Сегодня принято при доказательстве алгоритмической неразрешимости некоторой задачи сводить ее к ставшей классической задаче – «задаче останова».

Имеет место быть следующая теорема:

*Теорема 3.1*. Не существует алгоритма (машины Тьюринга), позволяющего по описанию произвольного алгоритма и его исходных данных (и алгоритм и данные заданы 2 символами на ленте машины Тьюринга) определить, останавливается ли этот алгоритм на этих данных или работает бесконечно.

Таким образом, фундаментально алгоритмическая неразрешимость связана с бесконечностью выполняемых алгоритмом действий, т.е. невозможностью предсказать, что для любых исходных данных решение будет получено за конечное количество шагов.

Тем не менее, можно попытаться сформулировать причины, ведущие к алгоритмической неразрешимости, эти причины достаточно условны, так как все они сводимы к проблеме останова, однако такой подход позволяет более глубоко понять природу алгоритмической неразрешимости:

*а) Отсутствие общего метода решения задачи*

*Проблема 1:* Распределение девяток в записи числа π;

Определим функцию f(n) = i, где n – количество девяток подряд в десятичной записи числа π, а i – номер самой левой девятки из n девяток подряд: π=3,141592… f(1) = 5.

Задача состоит в вычислении функции f(n) для произвольно заданного n.

Поскольку число π является иррациональным и трансцендентным, то мы не знаем никакой информации о распределении девяток (равно как и любых других цифр) в десятичной записи числа π. Вычисление f(n) связано с вычислением последующих цифр в разложении π, до тех пор, пока мы не обнаружим n девяток подряд, однако у нас нет общего метода вычисления f(n), поэтому для некоторых n вычисления могут продолжаться бесконечно – мы даже не знаем в принципе (по природе числа π) существует ли решение для всех n.

*Проблема 2:* Вычисление совершенных чисел;

Совершенные числа – это числа, которые равны сумме своих делителей, например: 28 = 1+2+4+7+14.

Определим функцию S(n) = n-ое по счёту совершенное число и поставим задачу вычисления S(n) по произвольно заданному n. Нет общего метода вычисления совершенных чисел, мы даже не знаем, множество совершенных чисел конечно или счетно, поэтому наш алгоритм должен перебирать все числа подряд, проверяя их на совершенность. Отсутствие общего метода решения не позволяет ответить на вопрос о останове алгоритма. Если мы проверили М чисел при поиске n-ого совершенного числа – означает ли это, что его вообще не существует?

*Проблема 3:* Десятая проблема Гильберта;

Пусть задан многочлен n-ой степени с целыми коэффициентами – P, существует ли алгоритм, который определяет, имеет ли уравнение P=0 решение в целых числах?

Ю.В. Матиясевич показал, что такого алгоритма не существует, т.е. отсутствует общий метод определения целых корней уравнения P=0 по его целочисленным коэффициентам.

*б) Информационная неопределенность задачи*

*Проблема 4:* Позиционирование машины Поста на последний помеченный ящик;

Пусть на ленте машины Поста заданы наборы помеченных ящиков (кортежи) произвольной длины с произвольными расстояниями между кортежами и головка находится у самого левого помеченного ящика. Задача состоит установке головки на самый правый помеченный ящик последнего кортежа.

Попытка построения алгоритма, решающего эту задачу приводит к необходимости ответа на вопрос – когда после обнаружения конца кортежа мы сдвинулись вправо по пустым ящикам на М позиций и не обнаружили начало следующего кортежа – больше на лен-те кортежей нет или они есть где-то правее? Информационная неопределенность задачи состоит в отсутствии информации либо о количестве кортежей на ленте, либо о максимальном расстоянии между кортежами – при наличии такой информации (при разрешении информационной неопределенности) задача становится алгоритмически разрешимой.

*в) Логическая неразрешимость (в смысле теоремы Гёделя о неполноте)*

*Проблема 5:* Проблема «останова» (см. теорема 3.1);

*Проблема 6:* Проблема эквивалентности алгоритмов;

По двум произвольным заданным алгоритмам (например, по двум машинам Тьюринга) определить, будут ли они выдавать одинаковые выходные результаты на любых исходных данных.

*Проблема 7:* Проблема тотальности;

По произвольному заданному алгоритму определить, будет ли он останавливаться на всех возможных наборах исходных данных. Другая формулировка этой задачи – является ли частичный алгоритм Р всюду определённым?

***3. Проблема соответствий Поста над алфавитом ∑***

В качестве более подробного примера алгоритмически неразрешимой задачи рассмотрим проблему соответствий Поста (Э. Пост, 1943 г.). Мы выделили эту задачу, поскольку на первый взгляд она выглядит достаточно «алгоритмизуемой», однако она сводима к проблеме останова и является алгоритмически неразрешимой.

Постановка задачи:

Пусть дан алфавит : ∑:|∑| >= 2 (для односимвольного алфавита задача имеет решение) и дано конечное множество пар из ∑+х∑+ , т.е. пары непустых цепочек произвольного языка над алфавитом : , ……, .

Проблема: Выяснить существует ли конечная последовательность этих пар, не обязательно различных, такая что цепочка, составленная из левых подцепочек, совпадает с последовательностью правых подцепочек – такая последовательность называется решаю-щей.

В качестве примера рассмотрим ∑ = {a,b}

1. Входные цепочки: (abbb, b), (a, aab), (ba, b)

Решающая последовательность для этой задачи имеет вид:

(a,aab) (a,aab) (ba,b) (abbb,b), так как : a a ba abbb ≡ aab aab b b

2. Входные цепочки: (ab,aba), (aba,baa), (baa,aa)

Данная задача вообще не имеет решения, так как нельзя начинать с пары (aba,baa) или (baa,aa), поскольку не совпадают начальные символы подцепочек, но если начинать с цепочки (ab,aba), то в последующем не будет совпадать общее количество символов «а», т.к. в других двух парах количество символов «а» одинаково.

В общем случае мы можем построить частичный алгоритм, основанный на идее упорядоченной генерации возможных последовательностей цепочек (отметим, что мы имеем счетное множество таких последовательностей) с проверкой выполнения условий задачи. Если последовательность является решающей – то мы получаем результативный ответ за конечное количество шагов. Поскольку общий метод определения отсутствия решающей последовательности не может быть указан, т.к. задача сводима к проблеме «останова» и, следовательно, является алгоритмически неразрешимой, то при отсутствии решающей последовательности алгоритм порождает бесконечный цикл.

В теории алгоритмов такого рода проблемы, для которых может быть предложен частичный алгоритм их решения, частичный в том смысле, что он возможно, но не обязательно, за конечное количество шагов находит решение проблемы, называются частично разрешимыми проблемами.

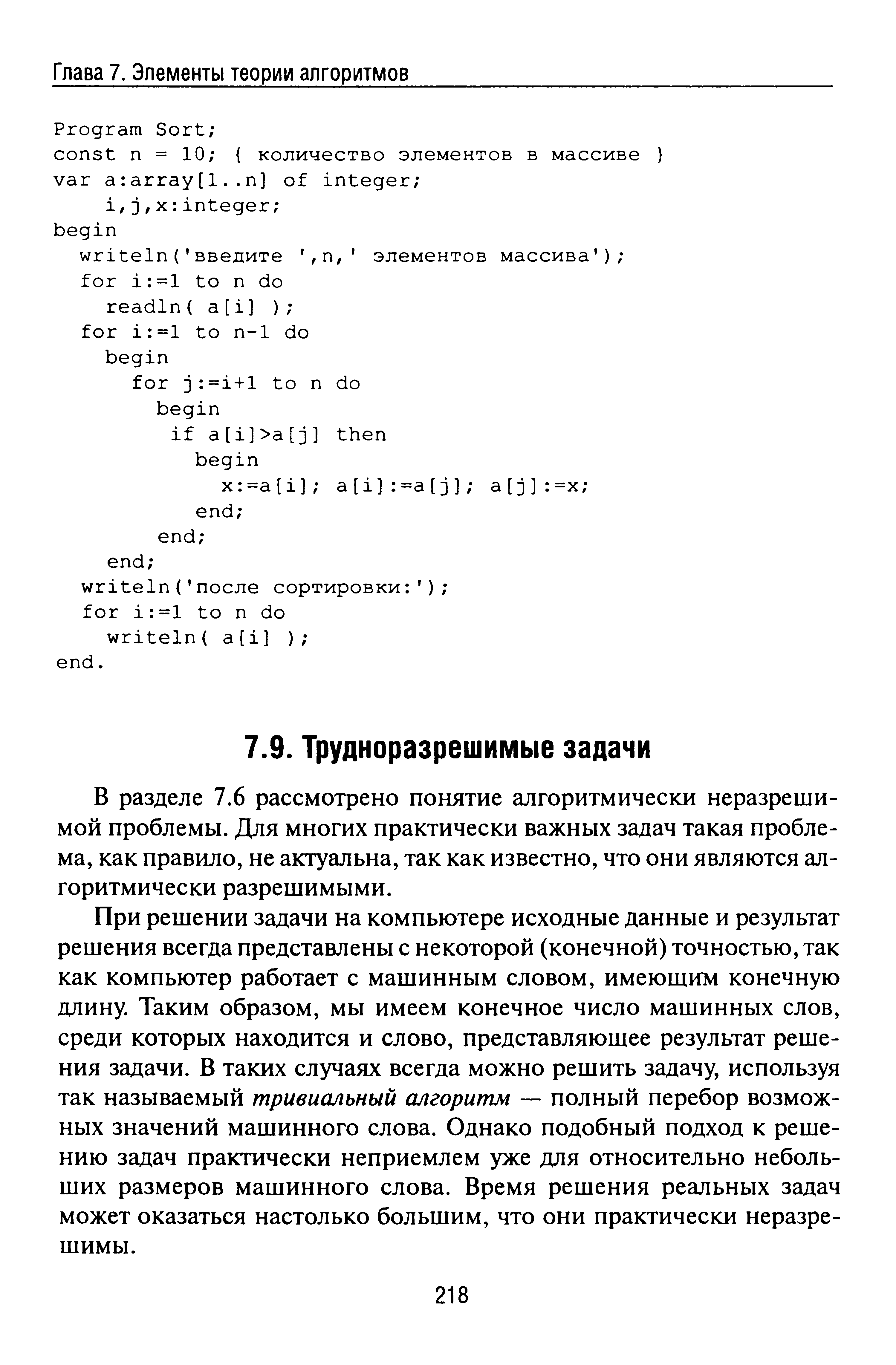
В частности, проблема останова так же является частично разрешимой проблемой, а проблемы эквивалентности и тотальности не являются таковыми.

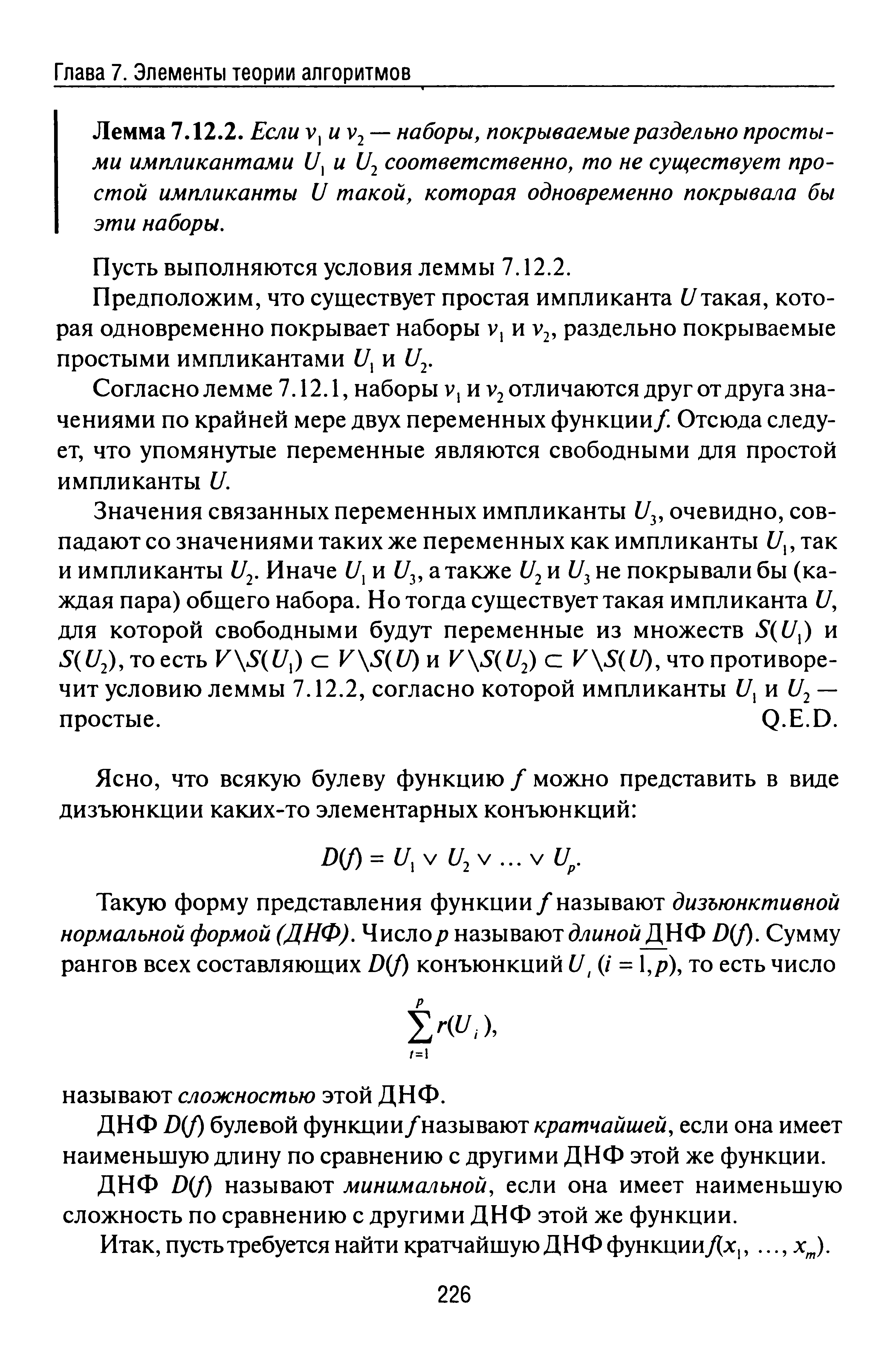
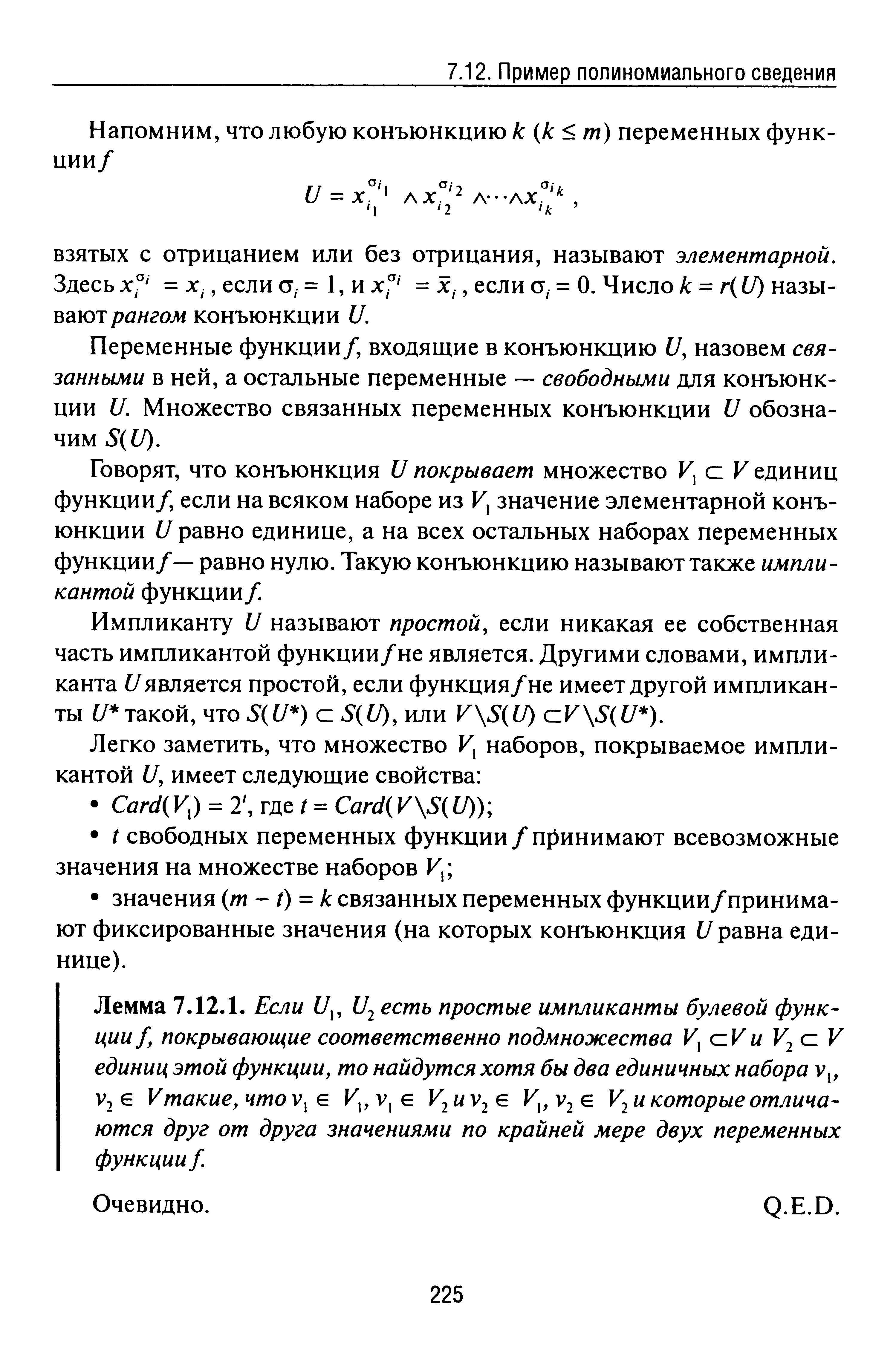
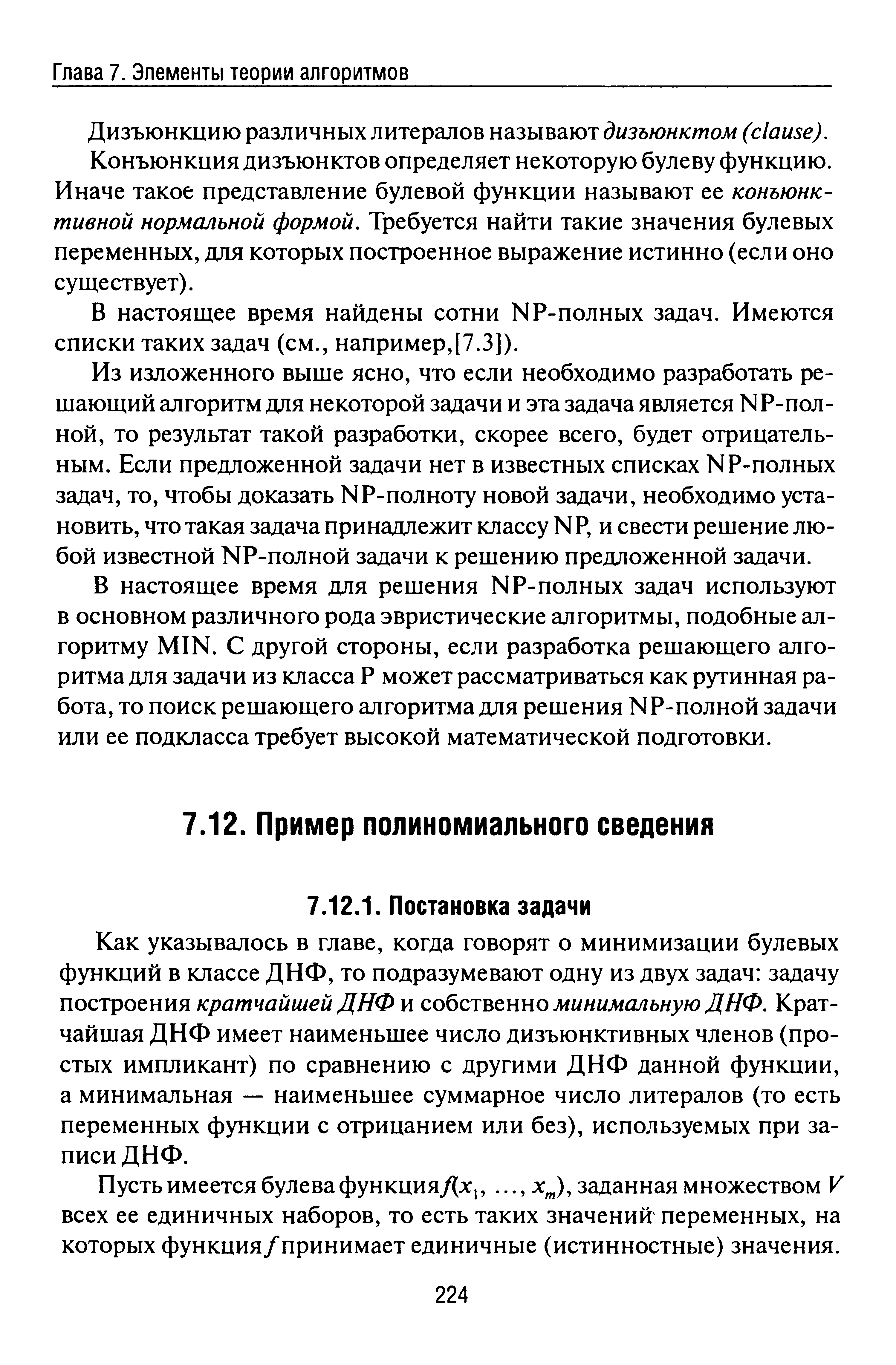
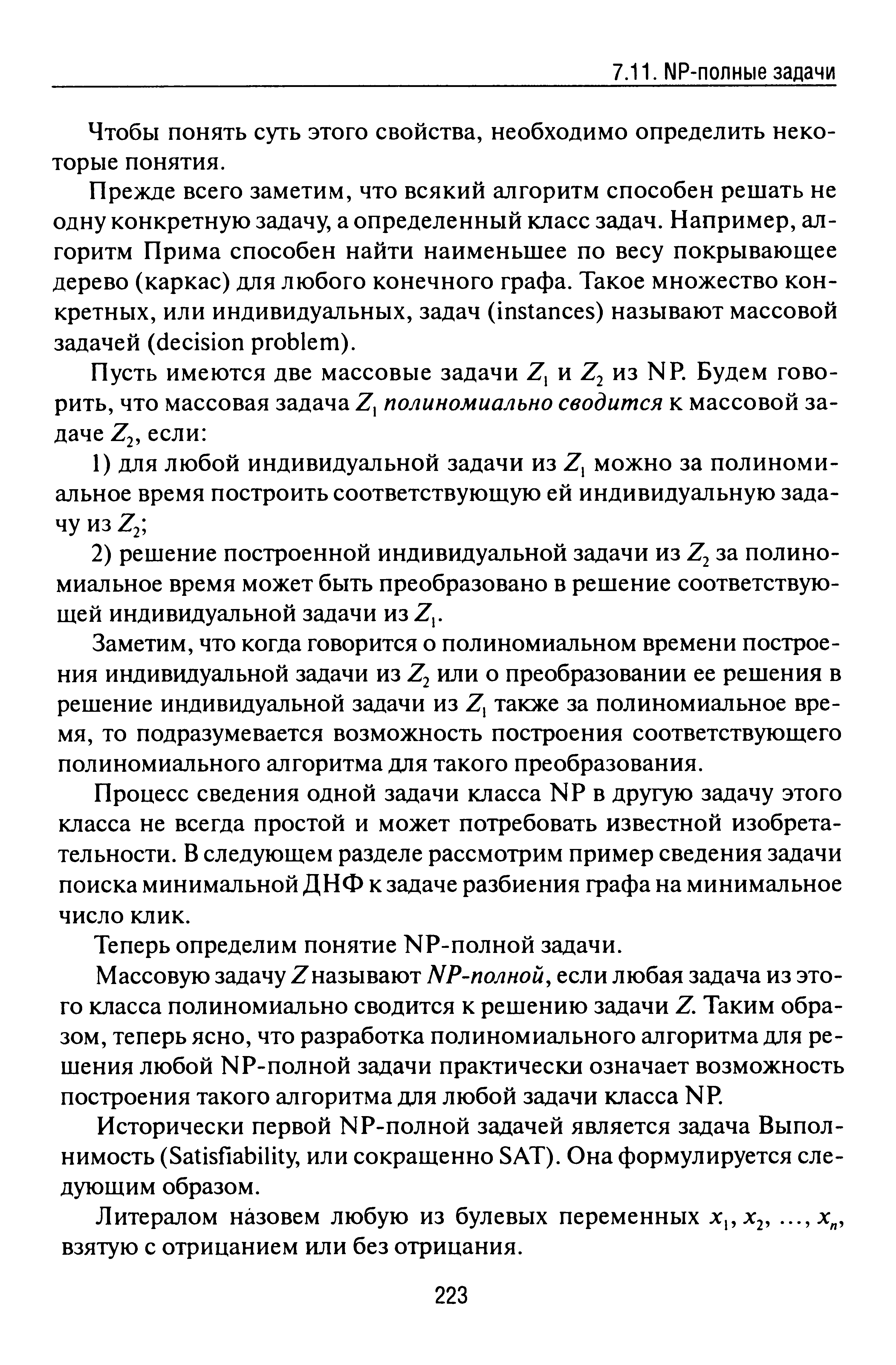
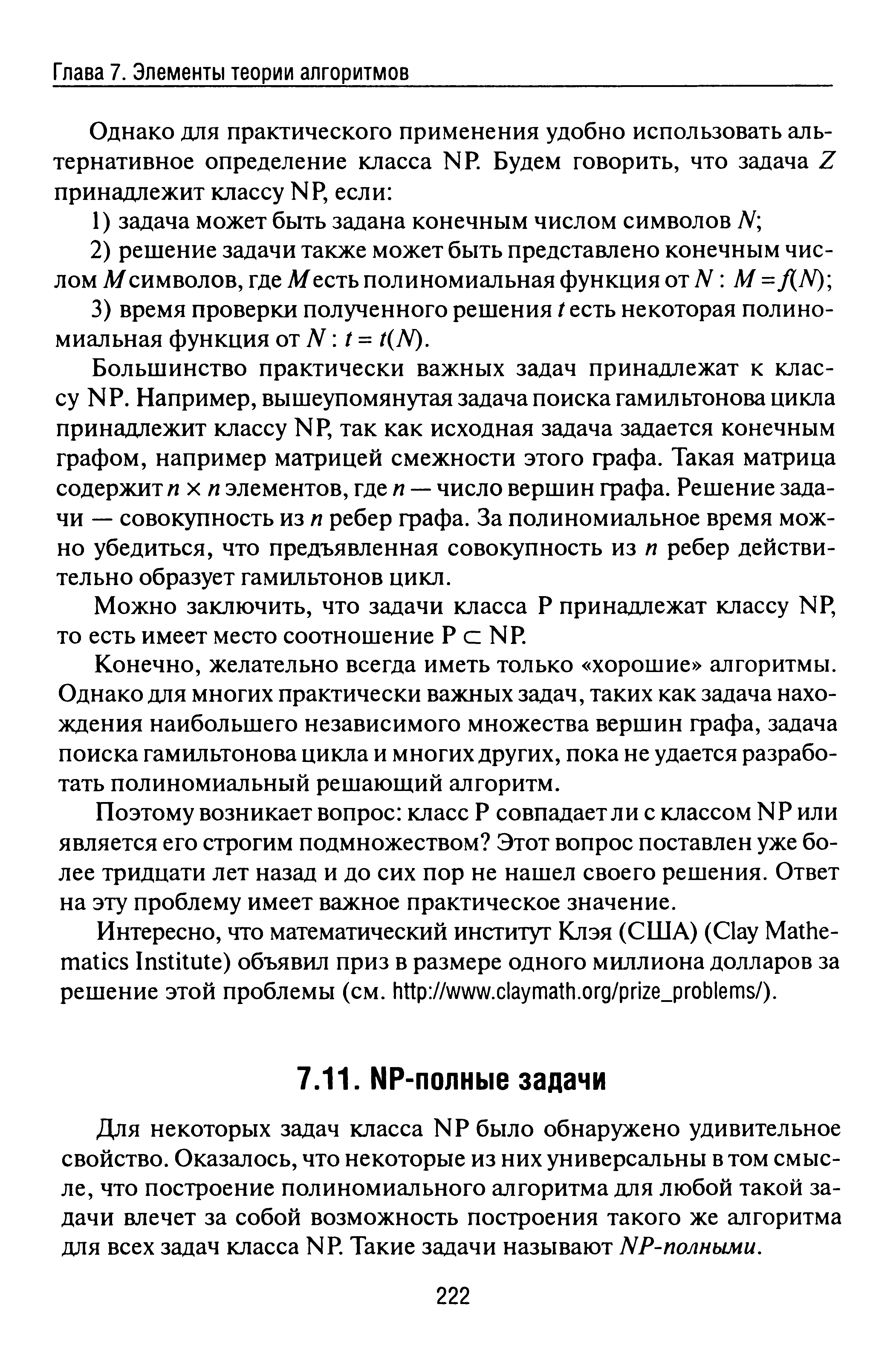
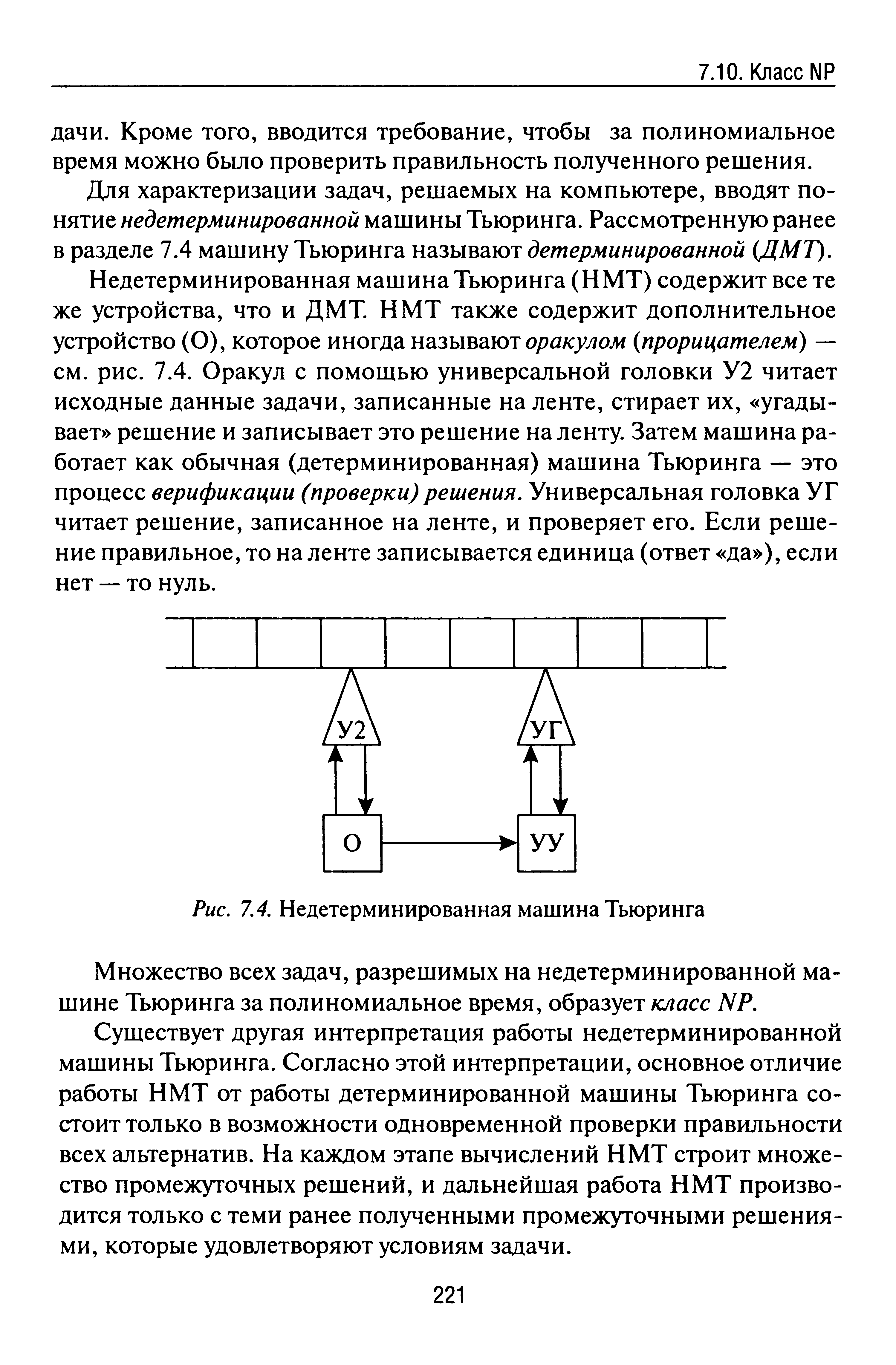
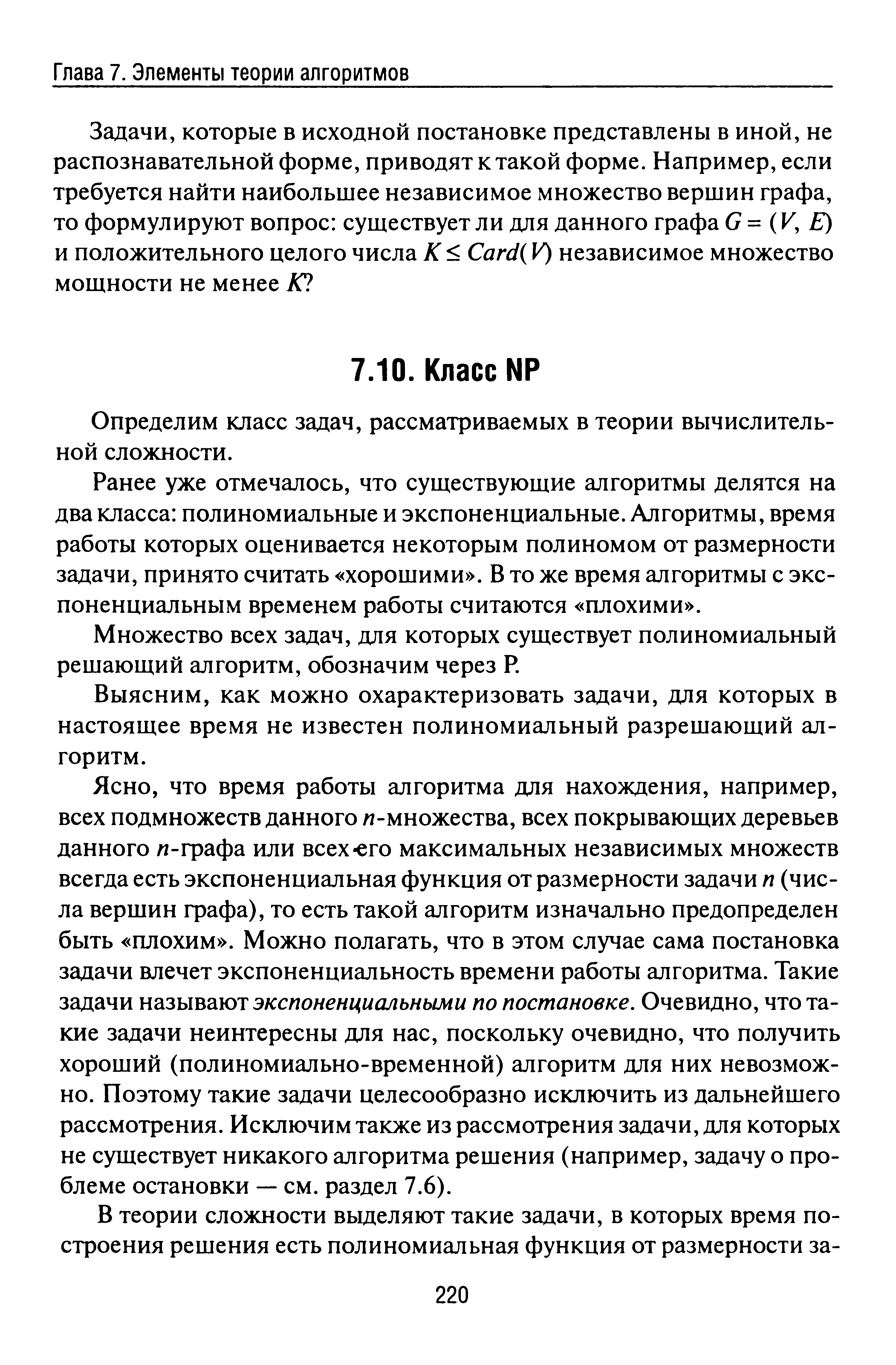
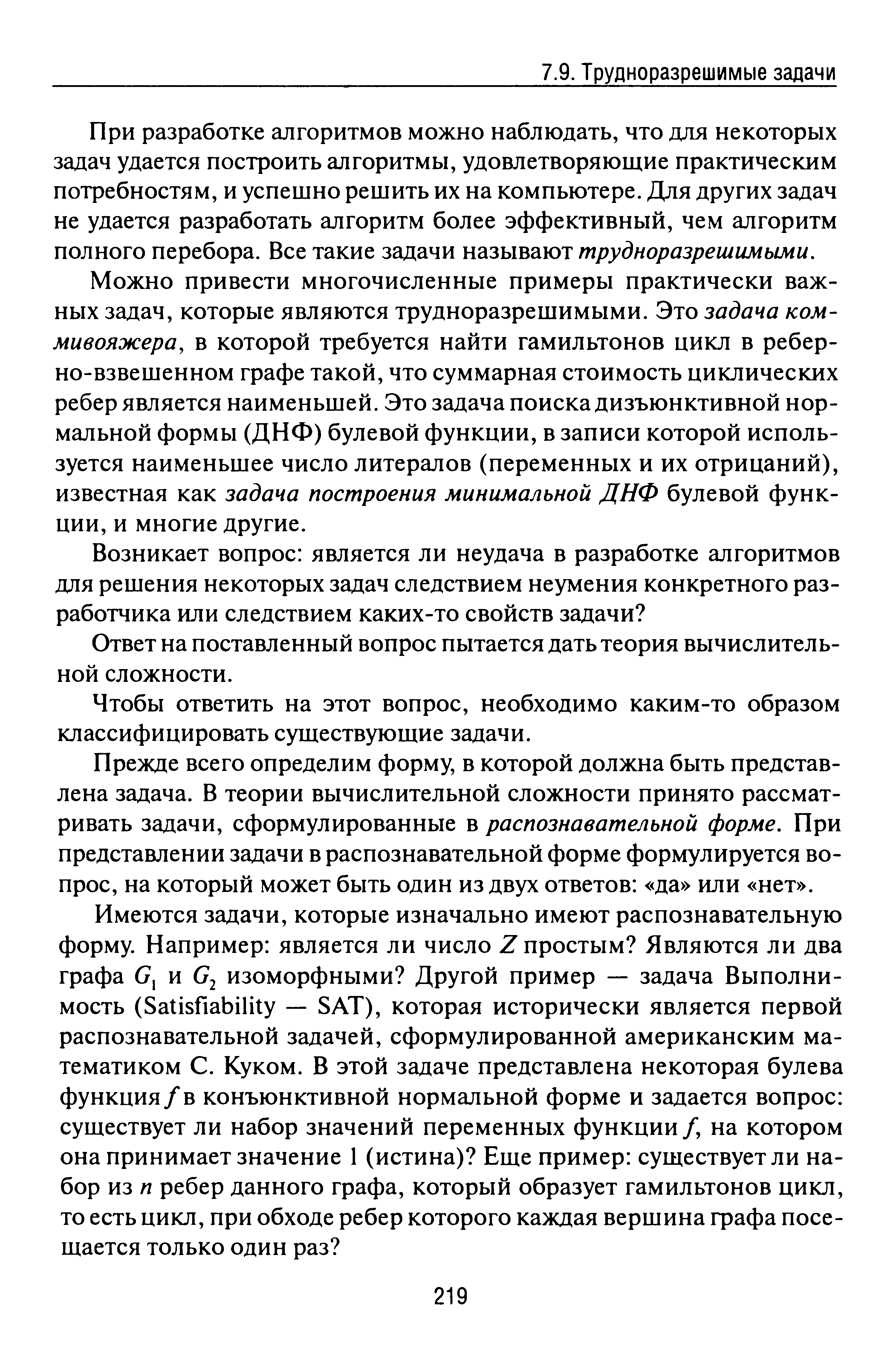
33 Трудноразрешимые проблемы. Примеры

Под задачей понимается некоторый вопрос, на который нужно найти (вычислить) ответ. Задачи бывают массовыми (общими) и частными (индивидуальными). Общая задача определяется: • списком параметров – свободных переменных, конкретные значения которых неопределенны; • формулировкой условий – свойств, которыми должен обладать ответ (решение задачи). Массовые задачи часто называют алгоритмическими проблемами. Частная задача получается из массовой, если всем параметрам массовой задачи придать конкретные значения – здесь исходные данные. В общем случаи для любой алгоритмически разрешимой задачи существует несколько разрешающих алгоритмов. С практической точки зрения важно не просто существование разрешающих алгоритмов, а поиск среди них наиболее простого и наименее трудоемкого. Под сложностью задачи принято понимать минимальную из сложностей алгоритмов, решающих эту задачу. Компьютерные науки пока не накопили достаточно сведений для того, чтобы задача минимизации сложности алгоритма была поставлена математически строго. Поэтому без ответов остаются такие вопросы, связанные с нижней оценкой сложности алгоритмов, а значит, и сложности задач: • Существует ли для заданной задачи алгоритм минимальной сложности? • Как убедиться, что найденный алгоритм действительно минимальный или, напротив, не минимальный? • Можно ли для рассматриваемой задачи доказать, что никакой разрешающий ее алгоритм не может быть проще найденной нижней оценки сложности? При разработке алгоритмов можно наблюдать, что для некоторых задач можно построить алгоритм полиномиальной сложности. Такие задачи называют полиномиальными. Полиноминально разрешимые задачи можно успешно решать на компьютере и даже в тех случаях, когда они имеют большую размерность. Для других задач не удается найти полиномиальный алгоритм. поэтому их называют трудноразрешимыми. К классу трудноразрешимых задач относится большое число задач алгебры, математической логики, теории графов, теории автоматов и других разделов дискретной математики. В большинстве своем это так называемые переборные задачи.Переборная задача характеризуется экспоненциальным множеством вариантов, среди которых нужно найти решение, и может быть решена алгоритмом полного перебора. Переборный алгоритм имеет экспоненциальную сложность и может хорошо работать только для небольших размеров задачи. С ростом размера задачи число вариантов быстро растет, и задача становится практически неразрешимой методом перебора. Многие из переборных задач являются экспоненциальными по постановке. Экспоненциальные по постановке задачи не представляют особого интереса для теории алгоритмов, поскольку для них, очевидно, невозможно получить алгоритм полиномиальной сложности. Возникает вопрос: если известно, что некоторая задача алгоритмически разрешима, то неудача в разработке для нее полиномиального разрешающего алгоритма является следствием неумения конкретного разработчика или следствием каких-то свойств самой задачи? Ответ на этот вопрос дает классическая теория алгоритмов, которая классифицирует задачи по сложности. При этом классифицируются лишь распознавательные задачи – задачи, имеющие распознавательную форму. В распознавательной форме суть задачи сводится к распознаванию некоторого свойства, а ее решение – один из двух ответов: «да» или «нет». С точки зрения математической логики задаче распознавания свойства соответствует предикат Р(х), где х – множество фактических значений входных переменных задачи. Существуют задачи, которые изначально имеют распознавательную форму. Например, являются ли два графа изоморфными? Другой пример – задача о выполнимости булевой функции, которая является исторически первой распознавательной задачей, глубоко исследованной в теории алгоритмов. Многие задачи, которые в исходной постановке представлены в иной форме (к ним относятся задачи дискретной оптимизации), довольно просто приводятся к распознавательной форме. Например, если требуется найти хроматическое число графа, то формулируют вопрос: верно ли, что заданный граф является k – раскрашиваемый, где k – заданное целое положительное число. Между тем, имеются задачи, которые нельзя привести к распознавательной форме. Это. В первую очередь, конструктивные задачи – задачи на построение объектов дискретной математики, обладающих заданными свойствами: генерация всех подмножеств конечного множества; генерация всех n! Различных перестановок; построение плоской укладки графа; построение остовного дерева графа и т. п. Такие задачи могут быть как трудноразрешимыми, так и полиноминально разрешимыми. Они пока не попадают под существующую в теории алгоритмов классификацию.

*Классификация задач по сложности*

Задачи, как и алгоритмы принято классифицировать по сложности. Множество всех распознавательных задач, для которых существует полиномиальный разрешающий алгоритм, образуют класс Р. Ясно, что распознавательные трудноразрешимые задачи не принадлежат классу Р. Класс NP – это множество распознавательных задач, которые могут быть разрешены за полиномиальное время на недетерминированной машине Тьюринга (НМТ). Оракул предлагает решения, которые после проверки верификатором приобретают «юридическую» силу. Таким образом, задачи класса NP являются «полиноминально проверяемыми». Например, в задаче коммивояжера оракул предлагает некоторую перестановку всех вершин графа, а верификатор проверяет, образует ли эта перестановка гамильтонов цикл графа. Ясно, что такую проверку можно выполнить с полиномиальной сложностью – надо лишь проверить смежность соседних вершин. Построить одну перестановку вершин тоже можно с полиномиальной сложностью. Оба шага решения задачи полиномиальные, поэтому задача коммивояжера принадлежит классу NP. Трудноразрешимой ее делает факториальное число повторений этих шагов. Следует отметить, что такую двухшаговую процедуру поиска решения можно применить к любой распознавательной задаче полиномиальной сложности.





Проблема остановки машины Тьюринга

34 Трудноразрешимые проблемы теории графов {???}

* [**Гипотеза Каццетты — Хаггвиста**](https://ru.wikipedia.org/w/index.php?title=%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0_%D0%9A%D0%B0%D1%86%D1%86%D0%B5%D1%82%D1%82%D1%8B_%E2%80%94_%D0%A5%D0%B0%D0%B3%D0%B3%D0%B2%D0%B8%D1%81%D1%82%D0%B0&action=edit&redlink=1) — [ориентированный граф](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84), имеющий *n* вершин, из каждой вершины которого выходит не менее *m* рёбер, имеет замкнутый контур длиной не более \left\lceil \frac{n}{m} \right\rceil.[[58]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-58)
* [**Гипотеза Хадвигера**](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%B7%D0%B0_%D0%A5%D0%B0%D0%B4%D0%B2%D0%B8%D0%B3%D0%B5%D1%80%D0%B0) — каждый [*n*-хроматический](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) граф [стягиваем](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#.D0.A1) к полному графу K_n[[59]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.9B.D0.B5.D0.BA.D1.86.D0.B8.D0.B8_.D0.BF.D0.BE_.D1.82.D0.B5.D0.BE.D1.80.D0.B8.D0.B8_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941990.E2.80.94.E2.80.94264-59).
* **Гипотеза**[**Улама**](https://ru.wikipedia.org/wiki/%D0%A3%D0%BB%D0%B0%D0%BC,_%D0%A1%D1%82%D0%B0%D0%BD%D0%B8%D1%81%D0%BB%D0%B0%D0%B2_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD):[[60]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.9B.D0.B5.D0.BA.D1.86.D0.B8.D0.B8_.D0.BF.D0.BE_.D1.82.D0.B5.D0.BE.D1.80.D0.B8.D0.B8_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941990.E2.80.94.E2.80.9418-60)
  + а) всякий граф с более чем двумя вершинами однозначно определяется *набором* графов, где каждый граф из набора получен удалением одной из вершин исходного графа;
  + б) всякий граф с более чем *тремя* вершинами однозначно определяется *множеством* графов, где каждый граф из множества получен удалением одной из вершин исходного графа.
* **Гипотеза Харари** (слабая форма гипотезы Улама) — если граф имеет более трёх рёбер, то его можно однозначно восстановить по подграфам, полученным удалением единственного ребра[[60]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.9B.D0.B5.D0.BA.D1.86.D0.B8.D0.B8_.D0.BF.D0.BE_.D1.82.D0.B5.D0.BE.D1.80.D0.B8.D0.B8_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941990.E2.80.94.E2.80.9418-60).
* В любом графе, не содержащем мостов (ребер, удаление которых увеличивает число компонент связности графа), можно выбрать множество простых циклов, такое, что каждое ребро принадлежит ровно двум из них.
* В любом [кубическом графе](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#.D0.9A) можно выбрать 6 [1-факторов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#.D0.A4) так, чтобы каждое ребро принадлежало ровно двум из них.
* **Гипотеза Рамачандрана** Любой [орграф](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D1%80%D0%B0%D1%84) N-реконструируем.[[61]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.9B.D0.B5.D0.BA.D1.86.D0.B8.D0.B8_.D0.BF.D0.BE_.D1.82.D0.B5.D0.BE.D1.80.D0.B8.D0.B8_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941990.E2.80.94.E2.80.94286-61)
* **Гипотеза Бержа** Граф G является [совершенным](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B2%D0%B5%D1%80%D1%88%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) тогда и только тогда, когда ни он, ни его дополнение \overline{G} не содержат порождённых подграфов вида C_{2k+1, k} \geqslant 2.[[62]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.9B.D0.B5.D0.BA.D1.86.D0.B8.D0.B8_.D0.BF.D0.BE_.D1.82.D0.B5.D0.BE.D1.80.D0.B8.D0.B8_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941990.E2.80.94.E2.80.94272-62)
* **Гипотеза о восстановлении** Если заданы классы изоморфизма всех k примарных подграфов некоторого графа, то при k \geqslant 3 класс изоморфизма этого графа определяется однозначно.[[63]](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B5_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BF%D1%80%D0%BE%D0%B1%D0%BB%D0%B5%D0%BC%D1%8B#cite_note-.D0.A2.D0.B5.D0.BE.D1.80.D0.B8.D1.8F_.D0.B3.D1.80.D0.B0.D1.84.D0.BE.D0.B2.E2.80.941988.E2.80.94.E2.80.94154-63)

- раскраска графов

- задача независимого множества

- задача вершинного покрытия

- задача коммивояджера