1979
Нотация как средство мышления

К. Е. Айверсон

Исследовательский центр Томаса Уотсона фирмы IBM

Тьюринговская премия ACM за 1979 г. была вручена К. Е. Айверсону председателем Тьюринговского комитета В. Карлсоном на ежегодной конференции ACM в Детройте, шт. Мичиган, 29 октября 1979 г.

Этим выбором Комитет по премиям за общие технические достижения отметил пионерские работы Айверсона по языкам программирования и математической нотации, завершившиеся созданием языка, известного теперь под названием APL. Был отмечен также вклад Айверсона в реализацию интерактивных систем, в обучение средствами языка APL и в теорию и практику языков программирования.

Айверсон родился и вырос в Канаде. Докторскую степень он получил в 1954 г. в Гарвардском университете. Там он работал младшим профессором прикладной математики с 1955 по 1960 г. Затем он заключил контракт с фирмой IBM и в 1970 г. был введен в научный совет IBM в знак признания его вклада в развитие языка APL.

В настоящее время доктор Айверсон работает в компании И. П. Шарпа в Торонто. Он опубликовал множество статей по языкам программирования и написал четыре книги о программировании и математике: «Язык программирования» (1962 г.), «Элементарные функции» (1966 г.), «Алгебра: алгоритмическая интерпретация» (1972 г.) и «Элементарный анализ» (1976 г.).

* * *

Уже давно признана важность классификации, нотации и языка как орудий мышления. Например, в химии и ботанике системы классификации Лавуазье и Линнея во многом стимулировали дальнейшие исследования и определили направления. Относительно языка Дж. Буль в [1, стр. 24] утверждает: «Всеми признана истина, что язык является инструментом человеческого мышления, а не только средством выражения готовых мыслей».

Математическая нотация представляет собой, может быть, самый известный и самый развитый пример языка, специально служащего орудием мышления. Признание важной роли нотации в математике явственно следует из высказываний математиков, приводимых в [2, стр. 332, 331]. Эти высказывания заслуживают того, чтобы прочесть их полностью, но об их общем духе можно судить по следующим цитатам:

«Освобождая мозг от всей необязательной работы, хорошая нотация позволяет ему сосредоточиться на более сложных проблемах и в результате увеличивает умственную мощь цивилизации».

А. Н. Уайтхейд

«Количество смысла, спрессованного в малом пространстве алгебраическими знаками, является еще одним обстоятельством, облегчающим рассуждения, которые мы привыкли выполнять с их помощью».

Чарльз Бэббидж

Тем не менее математическая нотация страдает существенными недостатками. В частности, ей недостает универсальности, и она должна интерпретироваться по-разному — в зависимости от предмета, автора и даже непосредственного контекста. Поскольку языки программирования предназначаются для управления компьютерами, они обладают важными преимуществами как средства мышления. Они не только универсальны (имеют общее назначение), но к тому же исполнимы и недвусмысленны. Их исполнимость позволяет использовать компьютеры для пространных экспериментов над идеями, выраженными на языке программирования, а отсутствие двусмысленностей делает возможными точные мысленные эксперименты. Однако в других отношениях большинство языков программирования, несомненно, слабее математической нотации и мало применяются в качестве инструментов мышления способами, которые, скажем, математик-прикладник счел бы важными.

Основная идея данной работы состоит в том, что обнаруживаемые в языках программирования исполнимость и универсальность могут быть эффективно объединены в одном согласованном языке с преимуществами, предоставляемыми математической нотацией.

(а) В разд. 1 идентифицируются наиболее существенные характеристики математической нотации и на простых примерах иллюстрируются способы представления таких характеристик в исполнимой нотации.

(б) В разд. 2 и 3 это иллюстрирование продолжается с более глубоким рассмотрением ряда аспектов, выбранных из-за общей привлекательности и полезности. В разд. 2 речь идет о полиномах, а в разд. 3 разбираются преобразования представлений, касающиеся ряда тем, в том числе перестановок и ориентированных графов. Хотя эти темы можно назвать математическими, они непосредственно связаны с программированием для компьютеров, а эта связь возрастает по мере того, как продолжается развитие программирования в законную математическую дисциплину.

(в) В разд. 4 приводятся примеры тождеств и формальных доказательств. Многие из этих формальных доказательств относятся к тождествам, которые были неформально установлены и использовались в предыдущих разделах.

(г) В заключительном разделе представлены некоторые общие сопоставления с математической нотацией, упоминаются подходы к другим темам и обсуждается проблема введения обозначений в контекст какой-то конкретной предметной области.

В качестве исполнимого языка мы будем применять APL, язык общего назначения, который был создан с целью обеспечения ясного и точного выражения мыслей при написании и преподавании и который был реализован в виде языка программирования только после нескольких лет его использования и развития [3].

Несмотря на то что среди читателей многие незнакомы с языком APL, я предпочел не давать здесь отдельное введение в APL, а вводить его в контекст по мере надобности. Математическая нотация всегда вводится таким образом, а не излагается в отдельном курсе подобно языкам программирования. Нотация, подходящая в качестве инструмента общения в какой-либо области, должна легко вводиться в контекст этой предметной области. Одно из преимуществ введения языка APL в наш контекст состоит в том, что читатель получает возможность оценить трудность такого введения.

Однако введение в контекст не сравнимо с полным обсуждением всех оттенков каждого элемента нотации, и читатель должен быть готов либо обобщать эти описания очевидным и систематическим способом в соответствии с требованиями дальнейших приложений, либо обратиться к литературе. Вся используемая здесь нотация суммирована в приложении А и полностью представлена на с. 24—60 руководства [4].

Читатели, имеющие доступ к компьютеру, оснащенному системой APL, могут захотеть перевести функциональные описания, приводимые здесь в форме прямого описания [5, с. 10] (с использованием знаков α и ω для представления левых и правых аргументов), в *каноническую* форму, которая требуется для исполнения. Функция для автоматического выполнения этого преобразования приведена в приложении Б.

1. ВАЖНЫЕ ХАРАКТЕРИСТИКИ НОТАЦИИ

Помимо выделенных во введении исполнимости и универсальности хорошая нотация должна обладать характеристиками, знакомыми каждому пользователю математической нотации:

- Легкость выражения конструкций, возникающих в данном классе задач.
- Содержательность.
- Возможность управлять выражением конкретных деталей.
- Экономичность.
- Пригодность для формальных доказательств.

Я не утверждаю, что этот список исчерпывающий, но буду вести последующее обсуждение в его рамках.

Недвусмысленная исполнимость введенной нотации сохраняет важность и будет подчеркиваться приведением под выражениями порождаемых ими результатов в явном виде. Чтобы различать выражения и результаты, мы будем задавать выражения в том виде, как они автоматически представляются на компьютерах с языком APL. Например, *целая* функция ⍳, примененная к аргументу N, порождает вектор первых N целых чисел; *сведение к сумме*, обозначаемое знаками +/, порождает сумму элементов своего векторного аргумента, и мы будем пользоваться следующим изображением:

```
      ⍳5
1 2 3 4 5
      +/⍳5
15
```

Мы будем применять один неисполнимый элемент нотации: символ ←→ между двумя выражениями устанавливает их эквивалентность.

1.1. ЛЕГКОСТЬ ВЫРАЖЕНИЯ КОНСТРУКЦИЙ, ВОЗНИКАЮЩИХ В ЗАДАЧАХ

Чтобы быть эффективным средством мышления, нотация должна допускать удобное выражение не только тех понятий, которые непосредственно вытекают из постановки задачи, но и тех, которые возникают при последующем анализе, обобщении и специализации.

Например, рассмотрим показанную на рис. 1 кристаллическую структуру, в которой атомы из последовательных слоев лежат не непосредственно один на другом, а образуют «плотную упаковку» с атомами из соответствующего нижнего слоя. Поэтому количество атомов в последовательных слоях, начиная с вершины рис. 1, задается как ⍳5, а общее количество задается как +/⍳5.

Трехмерная структура такого кристалла также дает плотную упаковку. Атомы в плоскости, лежащей поверх рис. 1, были бы расположены между атомами нижней плоскости; базовая ячейка состояла бы из четырех атомов. Иначе говоря, полная

```
    o
   o o
  o o o
 o o o o
o o o o o
```

Рис. 1.

трехмерная структура, соответствующая рис. 1, представляла бы собой набор тетраэдров, плоскости которых имеют длины оснований 1, 2, 3, 4 и 5. Поэтому числа элементов в последовательных плоскостях равны *частичным* суммам вектора ι5, т. е. первого элемента, сумме первых двух элементов и т. д. Такие частичные суммы вектора V обозначаются как $+\backslash V$, причем функция $+\backslash$ называется *разверткой суммы*.

```
      +\ι5
1 3 6 10 15
      +/+\ι5
35
```

Последнее выражение дает общее число атомов в тетраэдре.

Сумма $+/\iota 5$ может быть представлена графически и другими способами, например как показано на рис. 2 слева. В сочетании с такой же, но перевернутой конфигурацией справа эта сумма может просто интерпретироваться как число единиц в прямоугольнике, т. е. как произведение.

```
□       □□□□□
□□       □□□□
□□□       □□□
□□□□       □□
□□□□□       □
```

Рис. 2.

Длины строк фигуры, образованной слиянием двух частей рис. 2, задаются сложением вектора ι5 с обращенным вариантом того же вектора. Итак:

```
      ι5
1 2 3 4 5
      ⌽ι5
5 4 3 2 1
      (ι5)+(⌽ι5)
6 6 6 6 6
```

Эта конфигурация, состоящая из пяти повторений числа 6, может быть выражена как 5ρ6, и мы имеем

```
      5ρ6
6 6 6 6 6
      +/5ρ6
30
      6×5
30
```

Факт $+/5\rho 6 \leftrightarrow 6\times 5$ следует из определения умножения как повторяемого сложения.

Из предыдущего следует, что $+/\iota 5 \leftrightarrow (6\times 5)\div 2$ и, в более общем виде, что

$$+/\iota N \leftrightarrow ((N+1)\times N)\div 2. \qquad \text{A.1}$$

1.2. СОДЕРЖАТЕЛЬНОСТЬ

Нотация называется *содержательной,* если формы выражений, возникающих в неком множестве задач, подсказывают соответствующие выражения, находящие применение в других задачах. Теперь мы рассмотрим другие, аналогичные приведенным применения введенных до сих пор функций, а именно:

$$\iota \quad \phi \quad \rho \quad +/ \quad +\backslash$$

Пример

```
      5ρ2
2 2 2 2 2
      ×/5ρ2
32
```

наводит на мысль, что $\times/M\rho N \leftrightarrow N{*}M$, где знак $*$ представляет степенную функцию. Итак, сходство определений степени в терминах умножения и умножения в терминах сложения может быть показано следующим образом:

$$\times/M\rho N \leftrightarrow N{*}M$$
$$+/M\rho N \leftrightarrow N\times M$$

Аналогичные выражения для частичных сумм и частичных произведений можно разработать так:

```
      ×\5ρ2
2 4 8 16 32
      2*ι5
2 4 8 16 32
```

$$\times\backslash M\rho N \leftrightarrow N{*}\iota M$$
$$+\backslash M\rho N \leftrightarrow N\times\iota M$$

Поскольку они представимы треугольником, изображенным на рис. 1, суммы $+\backslash\iota 5$ называются *треугольными* числами. Это частный случай *фигурных* чисел, получаемых повторяемым применением развертки суммы, начиная либо с $+\backslash\iota N$, либо с $+\backslash N\rho 1$. Итак:

```
          5⍴1                          +\+\5⍴1
    1 1 1 1 1                    1 3 6 10 15

          +\5⍴1                        +\+\+\5⍴1
    1 2 3 4 5                    1 4 10 20 35
```

Замена сумм для последовательных целых на произведения порождает факториалы, например:

```
          ⍳5
    1 2 3 4 5
          ×/⍳5                         ×\⍳5
    120                          1 2 6 24 120
          !5                           !⍳5
    120                          1 2 6 24 120
```

Выразительная сила языка отчасти основывается на возможности представления тождеств в кратких, общих и легко запоминаемых формах. Мы проиллюстрируем это утверждение, выразив *двойственность* в форме, включающей законы Де-Моргана, умножение — применив логарифмы, а также приведя другие, менее известные тождества.

Если V — вектор положительных чисел, то произведение ×/V можно получить, взяв натуральные логарифмы от каждого элемента из V (обозначаемые как ⍟V), просуммировав их (+/⍟V) и применив экспоненциальную функцию (*+/⍟V). Итак:

```
×/V ←→ *+/⍟V
```

Поскольку экспоненциальная функция * — это функция, обратная к натуральному логарифму ⍟, то правая часть тождества предлагает общую форму

$$IG\ F/G\ V$$

где IG — функция, обратная G.

Используя знаки ∧ и ∨ для обозначения функций *и* и *или*, а знак ~ для обозначения обратной к самой себе функции логического отрицания, мы можем выразить законы Де-Моргана для произвольного числа элементов так:

```
∧/B ←→ ~∨/~B
∨/B ←→ ~∧/~B
```

Конечно, элементы из B ограничиваются логическими значениями 0 и 1. Воспользовавшись символами отношения для обозначения *функций* (например, $X<Y$ порождает 1, если X меньше

чем Y, и 0 в противном случае), мы можем выразить другие двойственности, такие, как

$$\neq/B \leftarrow\rightarrow \sim=/\sim B$$
$$=/B \leftarrow\rightarrow \sim\neq/\sim B$$

Наконец, используя ⌈ и ⌊ для обозначения функций *максимума* и *минимума*, мы можем выразить двойственности с включением арифметического отрицания:

$$\lceil/V \leftarrow\rightarrow -\lfloor/-V$$
$$\lfloor/V \leftarrow\rightarrow \lceil/-V$$

Можно отметить также, что в любой из приведенных выше двойственностей развертка ($F\backslash$) может заменить сведение ($F/$).

1.3. ДЕТАЛИЗАЦИЯ

Как заметил Бэббидж в высказывании, цитируемом в [2], краткость облегчает рассуждения. Краткость достигается детализацией, и мы рассмотрим здесь три важных способа ее обеспечения: использование массивов, присваивание имен функциям и переменным, употребление операций.

Мы уже видели примеры краткости, обеспечиваемой одномерными массивами (векторами), при рассмотрении двойственности; дальнейшая детализация достигается путем использования матриц и других массивов более высокого ранга, поскольку функции, определенные на векторах, систематически обобщаются на массивы более высокого ранга.

В частности, можно специфицировать ось, к которой применяется функция. Например, $\Phi[1]M$ действует по первой оси матрицы M и обращает каждый из столбцов, $\Phi[2]$ обращает каждую строку; $M,[1]N$ сочленяет столбцы [помещая M над N), а $M,[2]N$ сочленяет строки; $a+/[1]M$ суммирует столбцы, а $+/[2]M$ суммирует строки. Если никакая ось не специфицирована, функция применяется вдоль последней оси. Так, $+/M$ суммирует строки. И наконец, сведение вдоль *первой оси* может обозначаться символом ⌿.

Можно различать два способа использования имен: *постоянные* имена, соотносимые с фиксированными объектами, применяются для объектов, помещенных в весьма общих ситуациях, а *целевые* имена присваиваются (посредством символа ←) величинам, представляющим интерес в более узком контексте. Например, постоянная (имя) 144 соотносится с фиксированным объектом, а имена CRATE, LAYER и ROW, присваемые выражениями

$$\text{CRATE} \leftarrow 144$$
$$\text{LAYER} \leftarrow \text{CRATE} \div 8$$
$$\text{ROW} \leftarrow \text{LAYER} \div 3$$

являются целевыми именами, или именами *переменных*. Предусматриваются также постоянные имена для векторов, например 2 3 5 7 11 для числового вектора из пяти элементов и «ABCDE» для символьного вектора из пяти элементов.

Аналогичные различия существуют между именами функций. Такие постоянные имена, как $+$, $\times$ и $*$, присваиваются так называемым *примитивным* функциям общего назначения. Детализированные описания, например $+/M\rho N$ для $N \times M$ и $\times/M\rho N$ для $N*M$, подчиняются постоянным именам $\times$ и $*$.

Менее известные примеры постоянных имен функций связываются с запятой, которая *сцепляет* свои аргументы, как в примере

$$(\iota 5), (\Phi 5) \longleftrightarrow 1\ 2\ 3\ 4\ 5\ 5\ 4\ 3\ 2\ 1$$

и с функцией *базового представления* $\top$, которая порождает представление своего правого аргумента по основанию, заданному ее левым аргументом. Например:

```
      2 2 2 ⊤ 3 ↔ 0 1 1

      2 2 2 ⊤ 4 ↔ 1 0 0

      BN←2 2 2 ⊤ 0 1 2 3 4 5 6 7
      BN
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1

      BN,ΦBN
0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0
0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0
```

Матрица BN имеет важное значение, потому что ее можно рассматривать несколькими способами. Помимо представления двоичных чисел столбцы представляют все подмножества множества из трех элементов, а также входы в таблицы истинности для трех логических аргументов. Легко увидеть, что общее выражение для N имеет вид $(N\rho 2) \top (\iota 2*N) - 1$, и мы можем захотеть присвоить этой функции целевое имя. Посредством непосредственного описания (приложение Б) этой функции присваивается имя $\underline{T}$ следующим образом:

$$\underline{T}:(\omega\rho 2)\top(\iota 2*\omega)-1. \qquad \text{A.2}$$

Символ ω представляет аргумент функции; в случае двух аргументов левый представляется как α. В соответствии с таким описанием функции $\underline{T}$ выражение $\underline{T}3$ порождает показанную выше булеву матрицу $\overline{BN}$.

Три выражения, разделенные двоеточиями, служат также для следующего описания функции: выражение из середины выполняется первым; если его значение есть нуль, то выполняется первое выражение, в противном случае выполняется последнее выражение. Эта форма удобна для рекурсивных описаний, в которых функция используется для своего собственного описания. Например, функция, порождающая биномиальные коэффициенты для порядка, который указывается в ее аргументе, может быть определена рекурсивно так:

$$BC:(X,\ 0)+(0,\ X\leftarrow BC\ \omega-1):\omega=0:1. \qquad \text{A.3}$$

Итак, $BC\ 0\leftarrow\rightarrow 1$, $BC\ 1\leftarrow\rightarrow 1\ 1$, а $BC\ 4\leftarrow\rightarrow 1\ 4\ 6\ 4\ 1$.

Термин *операция*, используемый в строгом математическом смысле, а не как примерный синоним *функции*, относится к объекту, который применяется к функциям для получения функций; примером является операция дифференцирования.

Мы уже встречались с операциями *сведения* и *развертки*, обозначенными как / и \, и видели, как они способствуют краткости при их применении к различным функциям для получения семейств связанных функций, например +/, ×/ и ∧/. Теперь мы продолжим пояснение этого понятия, введя операцию *внутреннего произведения*, обозначаемую точкой. Функция (например, +/), порожденная неким оператором, будет называться *выведенной* функцией.

Если P и Q — два вектора, то внутреннее произведение +.× описывается так:

$$P+.\times Q\leftarrow\rightarrow +/P\times Q$$

и аналогичные описания имеют место и для других пар функций, отличных от + и ×. Например:

```
      P←2 3 5
      Q←2 1 2
      P+.×Q
17
      P×.*Q
300
      P⌊.+Q
4
```

Каждое из приведенных выше выражений имеет по крайней мере одну полезную интерпретацию: $P+.\times Q$ представляет собой общую стоимость порядковых величин Q для объектов, цены которых заданы через P; если P — это вектор простых чисел, то $P\times.*Q$ является числом, разложение которого на простые множители задается степенями Q; если P задает расстояние от исходного места до пунктов пересадки, а Q задает расстояния от пунктов пересадки до места назначения, то $P\lfloor.+Q$ дает минимальное возможное расстояние.

Функция $+.\times$ эквивалентна внутреннему произведению или математическому скалярному произведению и обобщается на матрицы, как в математике. Аналогично обобщаются и другие варианты, например $\times.*$. Так, если $\underline{T}$ — это функция, описанная формулой А.2, то

```
      T 3
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
      P+.×T 3                     P×.*T 3
0 5 3 8 2 7 5 10          1 5 3 15 2 10 6 30
```

Эти примеры выявляют важное обстоятельство: если значения B булевы, то $P+.\times B$ порождает суммы над подмножествами из P, специфицируемыми значениями 1 из B, а $P\times.*B$ порождает произведения над подмножествами.

Вариант $\circ.\times$ представляет собой специальное применение операции внутреннего произведения для получения выведенной функции, которая дает произведения каждого элемента своего левого аргумента и каждого элемента правого аргумента. Так:

```
      2 3 5∘.×⍳5
2  4  6  8 10
3  6  9 12 15
5 10 15 20 25
```

Функция $\circ.\times$ называется *внешним произведением*, как и в тензорном анализе, а такие функции, как $\circ.+$ и $\circ.*$, описываются аналогичным образом и порождают «таблицы функций» для соответствующих функций.

Например:

```
      D←0 1 2 3
      D∘.⌈D           D∘.≥D           D∘.!D
0 1 2 3         1 0 0 0         1 1 1 1
1 1 2 3         1 1 0 0         0 1 2 3
2 2 2 3         1 1 1 0         0 0 1 3
3 3 3 3         1 1 1 1         0 0 0 1
```

Символ ! обозначает функцию биномиальных коэффициентов, а таблица D∘.!D содержит треугольник Паскаля с вершиной слева; при обобщении на отрицательные аргументы (как в случае D←¯3¯2¯1 0 1 2 3) она будет содержать треугольные числа, а также фигурные числа для фигур более высокого порядка. Такое обобщение на отрицательные аргументы представляет интерес и для других функций. Например, таблица D∘.×D состоит из четырех квадрантов, разделенных строкой и столбцом из нулей, причем эти квадранты наглядно демонстрируют правило знаков для умножения.

Структура этих таблиц функций проявляет другие свойства этих функций, позволяющие кратко формировать утверждения, доказываемые перебором. Например, коммуникативность проявляется как симметрия относительно диагонали. Более точно, если результат применения к таблице T←D∘.fD функции транспонирования ⍉ (которая обращает порядок осей своего аргумента) совпадает с T, то функция f коммуникативна на данной области. Так, T=⍉T←D∘.⌈ιD порождает таблицу из единиц, потому что функция ⌈ коммутативная.

Соответствующие тесты ассоциативности требуют таблицы третьего ранга вида D∘.f(D∘.fD) и (D∘.fD)∘.fD. Например,

```
      D←0 1
D∘.∧(D∘.∧D)  (D∘.∧D)∘.∧D  D∘.≤(D∘.≤D)  (D∘.≤D)∘.≤D

   0 0            0 0           1 1           0 1
   0 0            0 0           1 1           0 1

   0 0            0 0           1 1           1 1
   0 1            0 1           0 1           0 1
```

1.4. ЭКОНОМИЧНОСТЬ

Полезность языка в качестве средства мышления возрастает с расширением круга проблем, к которым он подходит, но она уменьшается с ростом словарного запаса и сложности грамматических правил, которые пользователь должен хранить в своей памяти. Поэтому важное значение имеет экономичность нотации.

Экономичность требует, чтобы большое количество идей выражалось в терминах относительно малого словаря. Фундаментальная схема достижения этого состоит во введении грамматических правил, с помощью которых содержательные фразы и предложения могут конструироваться посредством комбинирования элементов словаря.

Эта схема может быть проиллюстрирована на первом рассмотренном примере — относительно простое и весьма полезное понятие суммы первых N целых чисел было введено не как примитив, а как фраза, сконструированная из двух общих понятий, функции ι порождения вектора целых и функции $+/$ суммирования элементов вектора. Более того, выведеная функция $+/$ сама является фразой, причем суммирование — это выведенная функция, сконструированная из более общего понятия оператора сведения, примененного к конкретной функции.

Экономичность достигается также за счет общности введенных функций. Например, описание функции факториала, обозначаемой через !, не ограничивается целыми числами, и поэтому гамма-функция от X может быть записана как $!X—1$. Аналогично *отношения*, определенные для всех вещественных аргументов, обеспечивают при их применении к логическим аргументам несколько важных логических функций: исключающее или ($\neq$), материальная импликация ($\leqslant$) и эквивалентность ($=$).

Достигаемую для рассмотренных до сих пор понятий экономичность можно подтвердить, вспомнив введенный словарь:

$$\iota \quad \rho \quad \phi \quad \top \quad ,$$
$$/ \quad \backslash \quad .$$
$$+ \ - \ \times \ \div \ \star \ \circledast \ ! \ \lceil \ \lfloor \ \phi$$
$$\vee \ \wedge \ \sim \ < \ \leq \ = \ \geq \ > \ \neq$$

Первостепенный интерес представляют пять функций и три операции, перечисленные в первых двух строках, остальные известные функции были введены, чтобы проиллюстрировать многосторонность операций.

В отличие от экономии функций значительная экономия символов достигается за счет того, что любой символ может представлять как *одноместную функцию* (т. е. функцию от одного аргумента), так и *двуместную* функцию аналогично тому, как знак минус обычно используется и для вычитания, и для отрицания. Поскольку две представленные функции могут оказаться связанными, как в случае знака минус, облегчается запоминание символов.

Например, $X*Y$ и $*Y$ представляют степень и экспоненту, $X\otimes Y$ и $\otimes Y$ представляют логарифм по основанию X и натуральный логарифм, $X\div Y$ и $\div Y$ представляют деление и получение обратного числа, а $X!Y$ и $!Y$ представляют функцию биномиальных коэффициентов и факториал (т. е. $X!Y \longleftrightarrow (!Y)\div(!X)\times\times(!Y—X)$). Символ ρ, используемый для двуместной функции повторения, представляет также и одноместную функцию, которая дает форму аргумента (т. е. $X \longleftrightarrow \rho X \rho Y$), символ Φ, служащий для одноместной функции обращения, представляет также двуместную функцию циклической перестановки, демонстрируемую с помощью 2φι5←→3 4 5 1 2 и ¯2φι5←→4 5 1 2 3 и наконец, запятая представляет не только конкатенацию, но и одноместное *связывание*, порождающее вектор элементов своего аргумента в порядке «старшинства строк». Например:

$$\begin{array}{lll} \underline{T}\ 2 & \qquad & ,\underline{T}\ 2 \\ 0\ 0\ 1\ 1 & & 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ 0\ 1\ 0\ 1 & & \end{array}$$

Важное значение имеет также простота грамматических правил нотации. Поскольку использованные до сих пор правила были знакомы нам по математической нотации, они не выделялись явно, но следует отметить два упрощения порядка исполнения:

(1) Все функции интерпретируются единообразно, и отсутствуют правила старшинства, такие, как исполнение × до +.

(2) Правило, что правый аргумент одноместной функции является значением всего выражения, находящегося справа от него, подразумеваемое в порядке исполнения такого выражения, как *SIN LOG !N*, обобщается на двуместные функции.

У второго правила имеются определенные полезные следствия применительно к сведению и развертке. Поскольку выражение *F/V* эквивалентно помещению функции *F* между элементами вектора *V*, то выражение —/*V* дает знакопеременную сумму элементов из *V*, а ÷/*V* дает знакопеременное произведение. Кроме того, если *B* — булев вектор, то </*B* «выделяет» первую единицу в *B*, потому что все элементы, следующие за ней, становятся нулями. Например:

</0 0 1 1 0 1 1←→0 0 1 0 0 0 0

Дальнейшее упрощение синтаксических правил достигается применением единой формы для всех двуместных функций, которые помещаются между своими аргументами, и для всех одноместных функций, которые помещаются перед своими аргументами. Это единообразие резко отличается от разнообразия правил математики. Так, символы одноместных функций отрицания, факториала и модуля вектора соответственно предшествуют своим аргументам, следуют за ними и окружают их. Двуместные математические функции демонстрируют еще большее разнообразие.

1.5. ДОСТУПНОСТЬ ФОРМАЛЬНЫХ ДОКАЗАТЕЛЬСТВ

Важность формальных доказательств и выводов очевидным образом следует из их роли в математике. Раздел 4 в значительной степени посвящен формальным доказательствам, и здесь мы ограничим обсуждение введением в формы доказательств.

Доказательство исчерпывающим перебором состоит в систематическом исследовании всего конечного количества частных случаев. Такое исчерпывание часто можно выразить просто, применив некоторое внешнее произведение к аргументам, включающим все элементы соответствующей области. Например, если $D \leftarrow 0\ 1$, то $D\circ.\wedge D$ дает все случаи применения функции *и*. Кроме того, закон Де-Моргана может быть доказан исчерпывающим перебором с помощью сравнения каждого элемента матрицы $D\circ.\wedge D$ с каждым элементом из $\sim(\sim D)\circ.\vee(\sim D)$ следующим образом:

```
      D∘.∧D                      ~(~D)∘.∨(~D)
0 0                        0 0
0 1                        0 1
      (D∘.∧D)=(~(~D)∘.∨(~D))
1 1
1 1
      ∧/,(D∘.∧D)=(~(~D)∘.∨(~D))
1
```

С проблемами ассоциативности можно обходиться аналогичным образом; следующие выражения демонстрируют ассоциативность *и* и неассоциативность *не-и*:

```
      ∧/,((D∘.∧D)∘.∧D)=(D∘.∧(D∘.∧D))
1
      ∧/,((D∘.⍲D)∘.⍲D)=(D∘.⍲(D∘.⍲D))
0
```

Доказательство последовательностью тождеств представляется перечислением последовательности выражений, причем каждое выражение аннотируется для поддержания очевидности его эквивалентности с его предшественником. Например, формальное доказательство тождества A.1, представленного в первом рассмотренном примере, было бы представлено следующим образом:

```
+/⍳N
+/⌽⍳N                       + is associative and commutative
((+/⍳N)+(+/⌽⍳N))÷2                           (X+X)÷2←→X
(+/((⍳N)+(⌽⍳N)))÷2          + is associative and commutative
(+/((N+1)⍴N))÷2                                   Lemma
((N+1)×N)÷2                               Definition of ×
```

Четвертое примечание относится к тождеству, которое (после рассмотрения примера в специальном случае $(\iota 5)+(\phi\iota 5)$) можно считать очевидным или же считать заслуживающим формального доказательства в качестве отдельной леммы.

Индуктивные доказательства проводятся в два этапа: (1) предполагается истинность некоторого тождества (называемого индуктивным предположением) для некоего фиксированного значения некоторого параметра N, и это предположение используется для доказательства справедливости этого тождества и для значения $N+1$, и (2) показывается, что это тождество справедливо для некоторого целого значения K. Вывод состоит в том, что тождество верно для всех целых значений N, которые равны или превосходят значение K.

Рекурсивные описания часто обеспечивают удобную основу для индуктивных доказательств. Для примера обратимся к рекурсивному описанию функции BC биномиальных коэффициентов, данному в А.3, и используем его в индуктивном доказательстве того, что сумма биномиальных коэффициентов порядка N равна $2*N$.

В качестве индуктивного предположения примем тождество

$$+/BC\ N \leftarrow\rightarrow 2*N$$

и затем предпримем следующие шаги:

`+/BC N+1`	
`+/(X,0)+(0,X←BC N)`	**A.3**
`(+/X,0)+(+/0,X)`	**+ is associative and commutative**
`(+/X)+(+/X)`	**0+Y←→Y**
`2×+/X`	**Y+Y←→2×Y**
`2×+/BC N`	**Definition of X**
`2×2*N`	**Induction hypothesis**
`2*N+1`	**Property of Power (*)**

Остается показать, что индуктивное предположение верно для некоторого целого значения N. Согласно рекурсивному описанию А.3, значение $BC\,0$ представляет собой значение самого правого выражения, т. е. 1. Поэтому $+/BC\,0$ равняется 1 и, следовательно, равняется $2*0$.

Закончим доказательством того, что закон Де-Моргана для скалярных аргументов, представленный в виде

$$A\wedge B \leftarrow\rightarrow \sim(\sim A)\vee(\sim B) \qquad \text{A.4}$$

и доказываемый перебором, может на самом деле быть обобщен на векторы произвольной длины, как указывалось раньше гипотетическим тождеством

$$\wedge/V \leftarrow\rightarrow \sim\vee/\sim V \qquad \text{A.5}$$

В качестве индуктивного предположения примем, что А.5 является истиной для векторов длины $(\rho V)-1$.

Дадим сначала формальные рекурсивные описания выводимых функций *и-сведения* и *или-сведение* ($\wedge/$ и $\vee/$) с помощью двух новых примитивов, *индексации* и *сбрасывания*. Индексация обозначается выражением вида $X[1]$, где 1 — одиночный индекс или массив индексов для вектора X. Например, если $X \leftarrow 2\ 3\ 5\ 7$, то $X[2]$ есть 3 и $X[2\ 1]$ есть 3 2. Сбрасывание обозначается как $K\downarrow X$ и описывается как отбрасывание $|K$ (т. е. модуль K) элементов из X, начиная с головы, если $K>0$, и с хвоста, если $K<0$. Например, $2\downarrow X$ равняется 5 7 и $^{-}2\downarrow X$ равняется 2 3. Функция *взятие* (которая будет применяться позднее) обозначается как $\uparrow$ и описывается аналогичным образом. Например, $3\uparrow X$ есть 2 3 5 и $^{-}3\uparrow X$ есть 3 5 7.

Формальные описания *и-сведения* и *или-сведения* обеспечиваются следующими функциями:

$$ANDRED: \omega[1] \wedge ANDRED\ 1 \downarrow \omega : 0 = \rho\omega : 1 \qquad \text{A.6}$$

$$ORRED: \omega[1] \vee \ ORRED\ 1 \downarrow \omega : 0 = \rho\omega : 0 \qquad \text{A.7}$$

Индуктивное доказательство тождества А.5 выполняется так:

`∧/V`	
`(V[1])∧(∧/1↓V)`	A.6
`~(~V[1])∨(~∧/1↓V)`	A.4
`~(~V[1])∨(~~∨/~1↓V)`	A.5
`~(~V[1])∨(∨/~1↓V)`	`~~X↔X`
`~∨/(~V[1]),(~1↓V)`	A 7
`~∨/~(V[1],1↓V)`	∨ distributes over ,
`~∨/~V`	Definition of , (catenation)

2. ПОЛИНОМЫ

Если C — вектор коэффициентов и X — скаляр, то полином от X с коэффициентом C может быть записан просто как $+/C\times X*^{-}1+\iota\rho C$, или $+/(X*^{-}1+\iota\rho C)\times C$, или $(X*^{-}1+\iota\rho C)+.\times C$. Однако для применения к нескалярному массиву аргументов X следует заменить степенную функцию $*$ на степенную таблицу $\circ.*$, как показано в следующем определении полиномиальной функции:

$$\underline{P}: (\omega \circ . *^{-}1 + \iota\rho\alpha) + . \times \alpha \qquad \text{B.1}$$

Например: $1\ 3\ 3\ 1\ \underline{P}\ 0\ 1\ 2\ 3\ 4 \leftarrow\!\rightarrow 1\ 8\ 27\ 64\ 125$. Если $\rho\alpha$ заменяется на $1\uparrow\rho\alpha$, то функция применяется также к матрицам коэффициентов и массивам коэффициентов большей размерности, представляющим (вдоль ведущей оси для α) наборы коэффициентов различных полиномов.

Из этого описания ясно видно, что полином является линейной функцией от вектора коэффициентов. Кроме того, если α и ω — это векторы одинаковой формы, то предмультипликатор $\omega\circ.*^{-}1+\iota\rho\alpha$ является матрицей Вандермонда для ω и поэтому обратим, если элементы из ω различны. Следовательно, если C и X — векторы одинаковой формы и если $Y\leftarrow C \underline{P} X$, то обратная проблема (подгонки кривой) очевидным образом решается применением функции ⌹ обращения матрицы к матрице Вандермонда и использованием тождества

$$C \leftarrow \rightarrow (⌹ X\circ.*^{-}1+\iota\rho X)+.\times Y$$

2.1. ПРОИЗВЕДЕНИЯ ПОЛИНОМОВ

Произведением двух полиномов B и C обычно считается вектор коэффициентов D, такой, что

$$D\ \underline{P}\ X \leftarrow \rightarrow (B\ \underline{P}\ X)\times(C\ \underline{P}\ X)$$

Хорошо известно, что можно вычислить D, взяв произведения всех пар элементов из B и C и просуммировав их по подмножествам тех произведений, которые ассоциируются с одной и той же степенью результата. Эти произведения появляются в таблице функции $B\circ.\times C$, и легко показать неформально, что степени X, ассоциирующиеся с элементами $B\circ.\times C$, задаются таблицей сложения $E\leftarrow(^{-}1+\iota\rho B)\circ.+(^{-}1+\iota\rho C)$. Например:

```
      X←2
      B←3 1 2 3
      C←2 0 3
      E←(¯1+ιρB)∘.+(¯1+ιρC)
      B∘.×C              E                X*E
 6 0 9               0 1 2           1  2  4
 2 0 3               1 2 3           2  4  8
 4 0 6               2 3 4           4  8 16
 6 0 9               3 4 5           8 16 32
      +/,(B∘.×C)×X*E
518
      (B P X)×(C P X)
518
```

Из сказанного вытекает следующее тождество, которое будет установлено формально в разд. 4.

$$(B\ \underline{P}\ X)\times(C\ \underline{P}\ X) \leftarrow \rightarrow +/,(B\circ.\times C)\times$$
$$\times X*(^{-}1+\iota\rho B)\circ.+(^{-}1+\iota\rho C) \qquad \text{B.2}$$

Кроме того, вид экспоненциальной таблицы E показывает, что лежащие на диагоналях элементы из $B\circ.\times C$ ассоциируются с одной и той же степенью и что поэтому вектор коэффициентов полинома-произведения задается суммами по этим диагоналям. Иначе говоря, таблица $B\circ.\times C$ обеспечивает отличную организацию ручного вычисления произведений полиномов. В нашем примере эти суммы дают вектор $D\leftarrow 6\ 2\ 13\ 9\ 6\ 9$, и можно увидеть, что $D\ \underline{P}\ X$ равняется $(B\ \underline{P}\ X)\times(C\ \underline{P}\ X)$

Суммы по нужным диагоналям матрицы $B\circ.\times C$ можно получить также, окаймив эту матрицу нулями, «сдвинув» полученную матрицу циклической перестановкой (сдвигом) последовательных строк на число позиций, равное последовательным целым, и затем просуммировав столбцы. Таким образом, мы получаем описание произведения полиномов в следующем виде:

$$PP: +/(1-\iota\rho\alpha)\phi\,\alpha\circ.\times\omega,\ 1\downarrow 0\times\alpha$$

Теперь мы разработаем иной метод, основанный на простом наблюдении, что если $B\ PP\ C$ порождает произведение полиномов B и C, то функция PP линейна по обоим своим аргументам. Следовательно,

$$PP: \alpha+.\times A+.\times\omega$$

где A — это массив, который нужно определить. Массив A должен иметь ранг 3 и зависеть от экспонент левого аргумента $(^{-}1+\iota\rho\alpha)$, результата $(^{-}1+\iota\rho 1\downarrow\alpha,\omega)$ и правого аргумента. «Дефицит» правой экспоненты задан таблицей разностей $(\iota\rho 1\downarrow\alpha,\omega)\circ.-\iota\rho\omega$, и сравнение этих значений с левыми экспонентами порождает A. Итак,

$$A\leftarrow(^{-}1+\iota\rho\alpha)\circ.=((\iota\rho 1\downarrow\alpha,\omega)\circ.-\iota\rho\omega)$$

и

$$PP: \alpha+.\times((^{-}1+\iota\rho\alpha)\circ.=(\iota\rho 1\downarrow\alpha,\omega)\circ.-\iota\rho\omega)+.\times\omega$$

Поскольку $\alpha+.\times A$ является матрицей, из этой формулы следует, что если $D\leftarrow B\ PP\ C$, то можно получить C из D, предварительно умножив D на обратную матрицу $(\boxed{\div}B+.\times A)$; тем самым обеспечивается деление полиномов. Так как матрица $B+.\times A$ не квадратная (в ней больше строк, чем столбцов), то этот способ не годится, но если заменить матрицу $M\leftarrow B+.\times A$ либо ее головной квадратной частью $(2\rho\lfloor/\rho M)\uparrow M$, либо хвостовой квадратной частью $(-2\rho\lfloor/\rho M)\uparrow M$, то получаются два результата, один из которых соответствует делению с остаточными членами нижнего порядка, а другой соответствует делению с остаточными членами более высокого порядка.

2.2. ПРОИЗВОДНАЯ ПОЛИНОМА

Так как производная от $X\times N$ равняется $N\times X*N-1$, мы можем применить правила дифференцирования суммы функций и произведения функции на константу, чтобы показать, что производная полинома $C\,\underline{P}\,X$ является полиномом $(1\downarrow C\times{}^{-}1+\iota\rho C)\underline{P}X$. Из этого результата ясно, что интеграл представляет собой полином $(A,C\div\iota\rho C)\underline{P}X$, где A — произвольная скалярная константа. Выражение $1\text{ф}\overline{C}\times{}^{-}1+\iota\rho C$ порождает также коэффициенты производной в виде вектора такой же формы, как C, с последним элементом, равным нулю.

2.3. ПРОИЗВОДНАЯ ПОЛИНОМА ПО ОТНОШЕНИЮ К ЕГО КОРНЯМ

Если R — вектор из трех элементов, то производные от полинома $\times/X-R$ по отношению к каждому из его корней равны $-(X-R[2])\times(X-R[3])$, $-(X-R[1])\times(X-R[3])$ и $-(X-R[1])\times(X-R[2])$. В более общем случае производная от $\times/X-R$ по отношению к $R[J]$ равна просто — $(X-R)\times.*J\neq\iota\rho R$, а вектор производных относительно каждого из корней равен — $(X-R)\times.*I\circ.\neq I\leftarrow\iota\rho R$.

Выражение $\times/X-R$ для полинома с корнями R применимо только к скаляру X, более общее выражение имеет вид $\times/X\circ.-D$. Поэтому общее выражение для матрицы производных (полинома, вычисленного в $X[I]$ по отношению к корню $R[J]$) задается так:

$$-(X\circ.-R)\times.*I\circ.\neq I\leftarrow\iota\rho R \qquad \text{B.3}$$

2.4. РАЗЛОЖЕНИЕ ПОЛИНОМА

Биномиальное разложение — это выписывание тождества в форме полинома от X для выражения $(X+Y)*N$. Для частного случая $Y=1$ мы имеем хорошо известное выражение в терминах биномиальных коэффициентов порядка N:

$$(X+1)*N\leftarrow\rightarrow((0,\iota N)!N)\,!N)\,\underline{P}\,X$$

В общем случае коэффициенты D являются функциями от Y. Если $Y=1$, они опять зависят только от биномиальных коэффициентов, но в этом случае от нескольких биномиальных коэффициентов разных порядков, конкретно от матрицы $J\circ.!J\leftarrow{}^{-}1+\iota\rho C$.

Например, если $C\leftarrow 3\ 1\ 2\ 4$ и $C\ \underline{P}\ X+1\leftarrow\rightarrow D\ \underline{P}\ X$, то D зависит от матрицы

```
0 1 2 3 ∘.! 0 1 2 3
1 1 1 1
0 1 2 3
0 0 1 3
0 0 0 1
```

и ясно, что величина D должна быть взвешенной суммой столбцов, причем весами являются элементы из C. Итак:

$$D \leftarrow (J \circ .! J \leftarrow {}^{-}1 + \iota\rho C) + . \times C$$

Краткая запись матрицы коэффициентов и выполнение указанного перемножения матриц обеспечивают быстрый и надежный способ организации громоздкого без этих приемов ручного вычисления разложений.

Если B — подходящая матрица биномиальных коэффициентов, то ясно, что $D \leftarrow B + . \times C$ и функция разложения линейна относительно коэффициентов C. Кроме того, разложение для $Y = {}^{-}1$ должно задаваться обратной матрицей $\boxed{\div} B$, которая, как мы увидим, содержит знакопеременные биномиальные коэффициенты. Наконец, поскольку

$$C\ \underline{P}\ X + (K+1) \leftarrow \rightarrow C\ \underline{P}\ (X+K) + 1 \leftarrow \rightarrow (B + . \times C)\ \underline{P}\ (X+K)$$

то, следовательно, разложение для положительных целых значений Y можно задать произведениями вида

$$B + . \times B + . \times B + . \times B + . \times C$$

где B появляется Y раз.

Так как суперпозиция $+ . \times$ ассоциативная, то предыдущее выражение можно записать в виде $M + . \times C$, где M является произведением Y появлений B. Интересно исследовать последовательные степени B, вычисленные либо вручную, либо выполнением на компьютере следующей степенной функции внутреннего произведения (IPP):

$$IPP : \alpha + . \times \alpha\ IPP\ \omega - 1 : \omega = 0 : J \circ . = J \leftarrow {}^{-}1 + \iota\ 1 \uparrow \rho\alpha$$

Сравнение $B\ IPP\ K$ с B для нескольких значений K показывает очевидную закономерность, которая может быть выражена в следующем виде:

$$B\ IPP\ K \leftarrow \rightarrow B \times K * 0 \ulcorner - J \circ . - J \leftarrow {}^{-}1 + \iota\ 1 \uparrow \rho B$$

Интересно, что правая часть этого тождества имеет смысл для нецелых значений K и фактически обеспечивает желаемое выражение для разложения $C\ \underline{P}\ X + Y$:

$$C\ \underline{P} (X+Y) \leftarrow \rightarrow$$

$$(((J\circ.!J)\times Y*0\ulcorner -J\circ.-J\leftarrow {}^{-}1+\iota\rho C)+.\times C)\underline{P}\ X \qquad \text{B.4}$$

Правая часть из B.4 имеет вид $(M+.\times C)\underline{P}\,X$, где матрица M в свою очередь имеет вид $B\times Y*E$ и может быть изображена неформально (для случая $4=\rho C$) следующим образом:

$$\begin{matrix}1&1&1&1\\0&1&2&3\\0&0&1&3\\0&0&0&1\end{matrix} \quad \times Y* \quad \begin{matrix}0&1&2&3\\0&0&1&2\\0&0&0&1\\0&0&0&0\end{matrix}$$

Поскольку $Y*K$ умножено на диагональную матрицу $B\times(K=E)$, то выражение для M можно также записать как внутреннее произведение $(Y*J)+.\times T$, где T — массив ранга 3 (т. е. трехмерный), в котором плоскость с номером K является матрицей $B\times(K=E)$. Такой массив ранга 3 можно сформировать из верхней треугольной матрицы M, образовав массив ранга 3, в котором первая плоскость — это M (т. е. $(1=\iota 1\uparrow\rho M)\circ.\times M$), и поворачивая его вдоль первой оси с помощью матрицы $J\circ.-J$, для которой супердиагональ с номером K имеет значение $-K$. Итак:

```
DS:(I∘.-I)⊖[1](1=I←⍳1↑⍴ω)∘.×ω                B.5

      DS  K∘.!K←¯1+⍳3
1 0 0
0 1 0
0 0 1

0 1 0
0 0 2
0 0 0

0 0 1
0 0 0
0 0 0
```

Подставив этот результат в B.4 и воспользовавшись ассоциативностью суперпозиции $+.\times$, мы получаем следующее тождество для разложения полинома, справедливое как для целых, так и для нецелых значений Y:

$$C\ \underline{P}\ X+Y\leftarrow\rightarrow((Y*J)+.\times(DS\ J*.!J)\leftarrow {}^{-}1$$
$$+\iota\rho C)+.\times C)\underline{P}\,X \qquad \text{B.6}$$

Например:

```
        Y←3
        C←3 1 4 2
        M←(Y*J)+.×DS J∘.!J←¯1+ιρC
        M
 1  3  9 27
 0  1  6 27
 0  0  1  9
 0  0  0  1
        M+.×C
96 79 22 2
        (M+.×C) P X←2
358
        C P X+Y
358
```

3. ПРЕДСТАВЛЕНИЯ

Объекты математического анализа и вычислений могут быть *представлены* разнообразными способами, и каждое представление может обладать конкретными преимуществами. Например, положительное число N можно представить просто N палочками, или менее просто, но более компактно посредством римских цифр, или же менее просто, но более удобно для выполнения сложения и умножения в десятичной системе, или менее привычно, но более удобно для вычисления наименьшего общего кратного и наибольшего общего делителя в схеме разложения на простые множители, которая будет здесь обсуждаться.

Графы, относящиеся к связям между набором элементов, представляют собой пример более сложного объекта, которому соответствует несколько полезных представлений. Так, простой ориентированный граф из N элементов (обычно называемых *узлами*) можно представить булевой матрицей B размера $N \times N$ (обычно называемой матрицей *смежности*), такой, что $B[I;\ J]=1$, если имеется связь *от* вершины I *к* вершине J. Всякая связь, представленная единицей в матрице B, называется *дугой*, и граф может быть представлен также матрицей из $+/,B$ строк и N столбцов, в которой каждая строка показывает узлы, связываемые конкретной дугой.

Функции также допускают различные полезные представления. Например, функция перестановки, порождающая переупорядочивание элементов своего векторного аргумента X, может быть представлена *вектором перестановки* P, таким, что функция перестановки имеет вид просто $X[P]$; *циклическим* представлением, которое более явно показывает структуру этой функции; булевой матрицей $B \leftarrow P \circ . = \iota\rho P$, такой, что функция перестановки — это $B + . \times X$; или же *корневым* представлением R, которое основывается на одном из столбцов матрицы $1 + (\phi\iota N) \top {}^{-}1 + \iota ! N \leftarrow \rho X$ и обладает тем свойством, что $2 | + / R - 1$ является четностью представляемой перестановки.

Для того чтобы удобным образом использовать различные представления, важно уметь ясно и точно выражать преобразования между представлениями. Обычная математическая нотация часто оказывается недостаточной в этом отношении, и настоящий раздел посвящен разработке выражений для преобразований между представлениями, полезными в разнообразных приложениях: в системах счисления, полиномах, перестановках, теории графов и в булевой алгебре.

3.1. СИСТЕМЫ СЧИСЛЕНИЯ

Начнем обсуждение представлений с уже знакомого примера, применения различных представлений положительных чисел и преобразований между ними. Вместо обычно употребляемого *позиционного* представления мы будем использовать *разложение на простые* множители, т. е. представление, интересные свойства которого делают его полезным и для введения понятия логарифмов, и для представления чисел [6, гл. 16].

Если P — вектор из первых ρP простых чисел и E — вектор неотрицательных целых, то E может служить для представления числа $P \times . * E$, и аналогичным образом можно представить любые целые из $\iota \lceil / P$. Например, $2\ 3\ 5\ 7 \times . * 0\ 0\ 0\ 0$ равняется 1, а $2\ 3\ 5\ 7 \times . * 1\ 1\ 0\ 0$ равняется 6, а также

```
      P
2 3 5 7
      ME
0 1 0 2 0 1 0 3 0 1
0 0 1 0 0 1 0 0 2 0
0 0 0 0 1 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0
      P×.*ME
1 2 3 4 5 6 7 8 9 10
```

Сходство с логарифмами видно из тождества

$$\times / P \times . * ME \leftarrow\rightarrow P \times . * + / ME$$

которое может служить для реализации умножения сложением.

Кроме того, если мы обозначим через GCD и LCM наибольший общий делитель и наименьшее общее кратное элементов векторных аргументов, то

```
GCD P×.*ME ↔ P×.*⌊/ME
LCM P×.*ME ↔ P×.*⌈/ME
```

```
      ME               V←P×.*ME
2 1 0                  V
3 1 2          18900 7350 3087
2 2 0                  GCD V                        LCM V
1 2 3          21                              926100
                       P×.*⌊/ME                     P×.*⌈/ME
               21                              926100
```

При описании функции GCD мы воспользуемся операцией / с булевым аргументом B [как в $B/$). Получается функция *сжатия*, которая выбирает элементы из своего правого аргумента в соответствии с единицами в B. Например, 1 0 1 0 1/ι5 равняется 1 3 5. Кроме того, функция $B/$, примененная к матричному аргументу, сжимает строки (выбирая определенные столбцы), а функция $B⌿$ сжимает столбцы для выбора строк. Итак:

```
GCD:GCD M,(M←⌊/R)|R:1≥ρR←(ω≠0)/ω:+/R
LCM:(×/X)÷GCD X←(1↑ω),LCM 1↓ω:0=ρω:1
```

Преобразование в значение числа из его представления в виде разложения на простые множители (VFR) и обратное преобразование в это представление из значения (RFV) задаются так:

```
VFR:α×.*ω
RFV:D+α RFV ω÷α×.*D:∧/~D←0=α|ω:D
```

Например:

```
        P VFR 2 1 3 1
10500
        P RFV 10500
2 1 3 1
```

3.2. ПОЛИНОМЫ

В разд. 2 введены два представления полинома от скалярного аргумента X, первое в терминах вектора коэффициентов C (т. е. $+/C\times X*^{-}1+\iota\rho C$), а второе в терминах его корней R (т. е. $\times/X-R$). Коэффициентное представление удобно для сложения полиномов ($C+D$) и для получения производных ($1{\downarrow}C\times{}^{-}1+\iota\rho C$). Корневое представление удобно для других целей, в том числе для умножения, которое задается посредством R_1, R_2.

Теперь мы разработаем функцию *CFR* (коэффициенты из корней), которая преобразует корневое представление в эквивалентное представление, и обратную функцию *RFC*. Разработка будет неформальной; формальный вывод *CFR* появится в разд. 4.

Выражение для *CFR* будет основываться на симметричных функциях Ньютона, которые порождают коэффициенты как суммы по некоторым из произведений над всеми подмножествами арифметического отрицания (т. е. $-R$) корней R. Например, коэффициент константного члена задается как $\times/-R$, т. е. как произведение над всем множеством, а коэффициент при следующем члене равняется сумме произведений над элементами из $-R$, взятыми по $(\rho R)-1$ каждый раз.

Функция, описанная в А.2, может служить для получения произведений над всеми подмножествами следующим образом:

$$P\leftarrow(-R)\times.*M\leftarrow\underline{T}\ \rho R$$

Элементы из P, суммированные для получения заданного коэффициента, зависят от числа элементов из R, исключенных из конкретного произведения, т. е. от $+/\sim M$, суммы столбцов дополнения булевой матрицы «подмножества» $\underline{T}\rho R$.

Таким образом, суммирование над P может быть выражено как $((0,\iota\rho R)\circ.=+/\sim M)+.\times P$, и полное выражение для коэффициентов C приобретает вид

$$C\leftarrow((0,\iota\rho R)\circ.=+/\sim M)+.\times(-R)\times.*M\leftarrow\underline{T}\ \rho R$$

Например, если $R\leftarrow2\ 3\ 5$, то

```
      M                                  +/~M
0 0 0 0 1 1 1 1                  3 2 2 1 2 1 1 0
0 0 1 1 0 0 1 1                       (0,ιρR)∘.=+/~M
0 1 0 1 0 1 0 1                  0 0 0 0 0 0 0 1
      (-R)×.*M                   0 0 0 1 0 1 1 0
1 ¯5 ¯3 15 ¯2 10 6 ¯30           0 1 1 0 1 0 0 0
                                 1 0 0 0 0 0 0 0
      ((0,ιρR)∘.=+/~M)+.×(-R)×.*M←T ρR
¯30 31 ¯10 1
```

Поэтому функция *CFR* получения коэффициентов из корней может быть описана и использована следующим образом:

```
    CFR:((0,ιρω)∘.=+/~M)+.×(-ω)×.*M←T ρω          C.1

    CFR 2 3 5
¯30 31 ¯10 1
    (CFR 2 3 5) P X←1 2 3 4 5 6 7 8
¯8 0 0 ¯2 0 12 40 90
    ×/X∘.-2 3 5
¯8 0 0 ¯2 0 12 40 90
```

Обратное преобразование *RFC* является более трудным, но может быть выражено как схема последовательных приближений:

```
RFC:(¯1+ιρ1↓ω)G ω
G:(α-Z)G ω:TOL≥⌈/|Z←α STEP ω:α-Z
STEP:(⌹(α∘.-α)×.*I∘.≠I←ιρα)+.×(α∘.*¯1+ιρω)+.×ω

    ⎕←C←CFR 2 3 5 7
210 ¯247 101 ¯17 1
    TOL←1E¯8
    RFC C
7 5 2 3
```

Разумеется, порядок корней в результате не имеет значения. Последним элементом любого аргумента для *RFC* должна быть единица, поскольку любой полином, эквивалентный ×/*X*—*R*, обязательно должен иметь коэффициент 1 для члена самого высокого порядка.

Данное выше описание *RFC* применимо только для коэффициентов полиномов, все корни которых являются вещественными. Левый аргумент для *G* в *RFC* обеспечивает (обычно удовлетворительные) начальные приближения для корней, но в общем случае по крайней мере некоторые из них должны быть комплексными. Следующий пример с использованием единицы как начального приближения корней был выполнен в системе APL, оперирующей комплексными числами:

```
        (*○0J2×(¯1+ιN)÷N←ρ1↓ω)Gω                C.2

        ⎕←C←CFR 1J1 1J¯1 1J2 1J¯2
10 ¯14 11 ¯4 1
        RFC C
1J¯1 1J2 1J1 1J¯2
```

Использованная выше одноместная функция ○ умножает свой аргумент на π.

В методе Ньютона отыскания корня скалярной функции F следующее приближение задается как $A \leftarrow A - (F\,A) \div DF\,A$, где DF — производная от F. Функция $STEP$ представляет собой обобщение метода Ньютона на случай, когда F является векторной функцией от вектора. Она имеет вид $(\boxed{\div}M)+.\times B$, где B — это значение полинома с коэффициентами ω; ω — исходный аргумент для RFC, вычисленный в α, т. е. в текущем приближении для корней. Анализ, подобный выводу формулы В.3, показывает, что M является матрицей производных полинома с корнями α, причем производные вычисляются в α.

Исследование выражения для M показывает, что все внедиагональные элементы этой матрицы представляют собой нули, и поэтому выражение $(\boxed{\div}M)+.\times B$ можно заменить на $B \div D$, где D — это вектор диагональных элементов из M. Поскольку операция $(I, J)\downarrow N$ отбрасывает I строк и J столбцов из матрицы N, то вектор D можно выразить как $\times/0\ 1\downarrow(^{-}1+\iota\rho\alpha)\phi\alpha\circ.-\alpha$; поэтому описание функции $STEP$ можно заменить на более эффективное описание:

$$STEP: ((\alpha\circ.*^{-}1+\iota\rho\omega)+.\times\omega)\div 0\ 1\downarrow(^{-}1+\iota\rho\alpha)\phi\alpha\circ.-\alpha \qquad \text{C.3}$$

Последняя формула реализует изящный метод Кернера [7]. С помощью начальных значений, задаваемых левым аргументом для G в С.2, эта аппроксимация сходится за семь шагов (с отклонением $TOL \leftarrow 1E^{-}8$) для представленного Кернером примера шестого порядка.

3.3. ПЕРЕСТАНОВКИ

Вектор P, элементы которого представляют собой некоторую перестановку его индексов (т. е. $\wedge/1 = +/P\circ.+\iota\rho P$), будет называться вектором *перестановок*. Если D — такой вектор перестановок, что $(\rho X) = \rho D$, то $X[D]$ является перестановкой от X и D будет называться *прямым* представлением этой перестановки.

Перестановка $X[D]$ может быть также выражена как $B+.\times X$, где B — булева матрица $D\circ.=\iota\rho D$. Матрица B будет называться булевым представлением перестановки. Преобразования между прямым и булевым представлениями выглядят так:

$$BFD: \omega\circ.=\iota\rho\omega \qquad DFB: \omega+.\times\iota 1\uparrow\rho\omega$$

В силу ассоциативности перестановки композиция перестановок удовлетворяет следующим отношениям:

$$(X[D1])[D2] \leftarrow\rightarrow X[(D1[D2])]$$

$$B2+.\times(B1+.\times X) \leftarrow\rightarrow (B2+.\times B1)+.\times X$$

Обратным для булева представления B является $\circledslash B$, а для прямого представления обратное — либо $\Uparrow D$, либо $D\iota\rho D$. (Функция $\Uparrow$ *ранжирования* ранжирует свой аргумент, присваивая его элементам вектор индексов в восходящем порядке и сохраняя существующий порядок для равных элементов. Итак, $\Uparrow$ 3 7 1 4 равняется 3 1 4 2 и $\Uparrow$ 3 7 3 4 равняется 1 3 4 2. Функция ι *индекс* определяет наименьший индекс в своем левом аргументе для каждого элемента из своего правого аргумента. Например, $'ABCDE'\iota'BABE'$ равняется 2 1 2 5, и $'BABE'\iota'ABCDE'$ равняется 2 1 5 5 4.)

В *циклическом* представлении также используется вектор перестановки. Рассмотрим вектор перестановки C и сегменты из C, выделенные вектором $C=\lfloor\backslash C$. Например, если $C\leftarrow$7 3 6 5 2 1 4, то $C=\lfloor\backslash C$ равняется 1 1 0 0 1 1 0 и блоки имеют вид

```
7
3 6 5
2
1 4
```

Каждый блок определяет «цикл» в соответствующей перестановке в том смысле, что если R является результатом перестановки X, то

$R[7]$ является $X[7]$
$R[3]$ является $X[6]$ $R[6]$ является $X[5]$ $R[5]$ является $X[3]$
$R[2]$ является $X[2]$
$R[1]$ является $X[4]$ $R[4]$ является $X[1]$

Если ведущий элемент из C является наименьшим (т. е. 1), то C состоит из единственного цикла и предоставляемая перестановка вектора X задается как $X[C]\leftarrow X[1\phi C]$. Например:

```
      X←'ABCDEFG'
      C←1 7 6 5 2 4 3
      X[C]←X[1⌽C]
      X
GDACBEF
```

Поскольку эквивалентны формулы $X[Q]\leftarrow A$ и $X\leftarrow A[\Uparrow Q]$, то, следовательно, эквивалентны также $X[C]\leftarrow X[1\phi C]$ и $X\leftarrow X[(1\phi C)[\Uparrow C]]$, и поэтому эквивалентный C вектор прямого представления D зададется (для специального случая единственного цикла) как $D \leftarrow (1 \phi C)[\uparrow C]$.

В более общем случае поворот полного вектора (т. е. $1 \phi C$) должен быть заменен поворотами отдельных подциклов, выделенных посредством $C \llcorner \backslash C$, как показано в следующем описании преобразования из циклического представления в прямое:

$$DFC:(\omega[\uparrow X + + \backslash X \leftarrow \omega = \llcorner / \omega])\ [\uparrow \omega]$$

Если есть желание соединить ряд несвязанных циклов, чтобы сформировать единый вектор C, такой, что выражение $C = = \llcorner \backslash C$ выделяет отдельные циклы, то каждый цикл $C1$ нужно сначала привести в *стандартный вид* поворотом $(^{-}1 + + CI \iota \llcorner / CI) \phi CI$, а полученные векторы затем соединить в нисходящем порядке их ведущих элементов.

Обратное преобразование из прямого представления в циклическое является более сложным, но к нему можно подступиться, построив матрицу всех степеней от D вплоть до степени ρD, т. е. матрицу, последовательными столбцами которой являются D, $D[D]$, $(D[D])[D]$ и т. д. Этот результат получается применением функции POW к одностолбцовой матрице $D \circ . + ,0$, сформированной из D, где POW описывается и используется следующим образом:

```
        POW:POW D,(D←ω[;1])[ω]:≤/ρω:ω

        ⎕←D←DFC C←7,3 6 5,2,1 4
4 2 6 1 3 5 7
        POW D∘.+,0
 4 1 4 1 4 1 4
 2 2 2 2 2 2 2
 6 5 3 6 5 3 6
 1 4 1 4 1 4 1
 3 6 5 3 6 5 3
 5 3 6 5 3 6 5
 7 7 7 7 7 7 7
```

Если $M \leftarrow POW\ D \circ . + ,0$, то циклическое представление можно получить выбором из M только «стандартных» строк, которые начинаются со своих наименьших элементов (SSR), упорядочивая эти остающиеся строки в нисходящем порядке их ведущих элементов (DOL) и последующим соединением циклов по этим строкам (CIR). Итак:

```
      CFD:CIR DOL SSR POW ω∘.+,0

        SSR:(∧/M=1⌽M←⌊\ω)/ω
        DOL:ω[⍒ω[;1];]
        CIR:(,1,∧\0 1↓ω≠⌊\ω)/,ω

      DFC C←7,3 6 5,2,1 4
4 2 6 1 3 5 7
      CFD DFC C
7 3 6 5 2 1 4
```

В описании *DOL* индексация применяется к матрицам. Индексы для последовательных координат разделяются точками с запятыми, и пустая запись для каждой оси означает, что все элементы вдоль этой оси выбраны. Так, *M*[; 1] выбирает столбец 1 из *M*.

Циклическое представление удобно для определения числа циклов в данной перестановке (*NC* : +/ω=⌊\ω), длин циклов (*CL* : *X*—0,⁻1↓*X*←(1⌽ω=⌊\ω)/ιρω) и *степени* перестановки (*PP* : *LCM CL* ω). С другой стороны, такое представление неудобно для композиции и обращения.

Все !*N* векторов-столбцов из матрицы (⌽ι*N*)⊤⁻1+ι!*N* различны, и поэтому они обеспечивают возможное *корневое* представление [8] для !*N* перестановок порядка *N*. Вместо этого мы будем использовать связанную с ним форму, получаемую увеличением каждого элемента на 1:

```
      RR:1+(⌽ιω)⊤⁻1+ι!ω

      RR 4
1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4
1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Преобразования между этим дополнительным представлением и прямой формой задаются так:

```
      DFR:ω[1],X+ω[1]≤X←DFR 1↓ω:0=ρω:ω
      RFD:ω[1],RFD X−ω[1]≤X← 1↓ω:0=ρω:ω
```

Некоторые характеристики этого варианта представления, возможно, лучше всего выявляются модификацией *DFR* для применения ко всем столбцам матричного аргумента и применением модифицированной функции *MF* к результату функции *RR*.

```
MF:ω[,1;],[1]X+ω[(1 ρX)ρ1;]≤X←MF 1 0↓ω:0=1↑ρω:ω
MF RR 4
1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4
2 2 3 3 4 4 1 1 3 3 4 4 1 1 2 2 4 4 1 1 2 2 3 3
3 4 2 4 2 3 3 4 1 4 1 3 2 4 1 4 1 2 2 3 1 3 1 2
4 3 4 2 3 2 4 3 4 1 3 1 4 2 4 1 2 1 3 2 3 1 2 1
```

Прямые перестановки в столбцах этого результата появляются в *лексикографическом* порядке (т. е. по возрастанию первого элемента, в котором два вектора отличаются); этот факт справедлив и в общем случае, и поэтому дополнительное представление обеспечивает удобный способ получения прямых представлений в лексикографическом порядке.

Дополнительное представление имеет еще одно полезное свойство, состоящее в том, что четность прямой перестановки D задается как $2|+/^-1+RFD\,D$, где $M|N$ представляет остаток от N по модулю M. Четность прямого представления может быть определена также функцией

$$PAR: 2 \mid +/.(I\circ.>I\leftarrow \iota\rho\omega)\wedge\omega\circ.>\omega$$

3.4. ОРИЕНТИРОВАННЫЕ ГРАФЫ

Простой ориентированный граф описывается множеством из K узлов и множеством ориентированных соединений от одной вершины к другой для пар вершин. Ориентированные соединения удобно представляются булевой матрицей *соединений* C размера $K\times K$, в которой значение $C[I;\ J]=1$ обозначает соединение, ведущее *из* узла I *в* узел J.

Например, если четыре узла графа представляются как $N\rightarrow'QRST'$ и если имеются соединения из узла S в узел Q, из R в T и из T в Q, то соответствующая матрица соединений задается так:

$$\begin{matrix} 0&0&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 1&0&0&0 \end{matrix}$$

Соединение от некоего узла к нему самому (называемое нулевой петлей) недопустимо, и поэтому диагональ матрицы соединений должна состоять из нулей.

Если P — это любой вектор перестановки порядка ρN, то $N1\leftarrow N[P]$ является переупорядочиванием узлов, и соответствующая матрица соединений задается как $C[P;\ P]$. Мы можем (и должны) без потери общности употреблять для узлов числовые метки $\iota\rho N$, потому что если N — это любой произвольный вектор имен для узлов, а L — любой список числовых меток, то выражение $Q\leftarrow N[L]$ дает соответствующий список имен, и наоборот, $N\iota Q$ дает список числовых меток.

Матрица соединений C удобна для выражения многих полезных функций на графе. Например, $+/C$ задает *выходные степени* узлов, $+\!\not/C$ задает *входные* степени, $+/,C$ задает число соединений или *дуг*, $\circledslash C$ задает родственный граф с обратными направлениями дуг и $C\vee\circledslash C$ задает родственный «симметричный» или «неориентированный» граф. Кроме того, если мы используем булев вектор $B\leftarrow\vee/(\iota 1\rho C)\circ.=L$ для представления списка вершин L, то $B\vee.\wedge C$ задает булев вектор, который представляет множество узлов, непосредственно достижимых из множества B. Поэтому $C\vee.\wedge C$ задает соединения для путей длины два в графе C и $C\vee C\vee.\wedge C$ задает соединения для путей длины один или два. Это рассуждение приводит к следующей функции для *транзитивного замыкания* графа, дающего все соединения путями любой длины:

$$TC\!:\!TC\quad Z\!:\wedge/,\ \omega=Z\leftarrow\omega\wedge\omega\vee.\wedge\omega:Z$$

Узел J называется *достижимым* из узла I, если $(TC\ C)\ [I;\ J]=1$. Граф называется *сильносвязным*, если всякий узел достижим из любого узла, т. е. $\wedge\ /,\ TC\ C$.

Если $D\leftarrow TC\ C$ и $D(I;\ I)=1$ для некого узла I, то узел I *достижим* из самого себя посредством пути некоторой длины; этот путь называется *контуром*, а об узле I говорят, что он содержится в контуре.

Граф T называется *деревом*, если он не содержит контуров и его входные степени не превышают 1, т. е. $\wedge\ /1\geqslant+\!\not/T$. В дереве любой узел с входной степенью 0 называется *корнем*, и если $K\leftarrow+/0=+\!\not/T$, то T называется K-корневым деревом. Поскольку в дереве нет контуров, то значение K должно быть не меньше 1. Если не указано обратное, то обычно предполагается, что дерево является *однокорневым* (т. е. $K=1$): многокорневые деревья иногда называются *лесами*.

Граф C *покрывает* граф D, если $\wedge/,\ C\geqslant D$. Если G — сильносвязный граф и T — это (однокорневое) дерево, то T называется *стягивающим деревом* для G, если G покрывает T и если все узлы достижимы из корня дерева T, т. е.

$$(\wedge/,\ G\geqslant T)\wedge\ \wedge/R\vee R\vee.\wedge TC\ T$$

где R — это булево представление корня T.

Стягивающее дерево первой глубины [9] для графа G — это стягивающее дерево, порожденное движением из корня через непосредственных «потомков» в G, причем в качестве очередного узла всегда выбирается потомок последнего узла из списка рассмотренных узлов, который еще обладает потомком вне списка. Этот относительно сложный процесс может служить для иллюстрации применения представления в виде матрицы соединений:

```
DFST:((,1)∘.=K) R ω∧K∘.∨~K←α=ι1↑ρω          C.4

R:(C,[1]α)Rω∧P∘.∨~C←<\U∧P∨.∧ω
 :~∨/P←(<\α∨.∧ω∨.∧U←~∨⌿α)∨.∧α
 :ω
```

Воспользуемся примером графа G из [9]:

```
            G                                  1 DFST G
0 0 1 1 0 0 0 0 0 0 0 0          0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0          0 0 0 0 1 0 0 0 0 0 0 0
0 1 0 0 1 1 0 0 0 0 0 0          0 1 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0          0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0          0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0          0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0          0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 1 0          0 0 0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0 0 0 1          0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1          0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0          0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0          0 0 0 0 0 0 0 0 0 0 0 0
```

Функция *DFST* устанавливает левый аргумент рекурсии *R* как однострочную матрицу, представляющую корень, указанный левым аргументом из *DFST*, и правый аргумент как исходный граф с удаленными соединениями, ведущими в корень *K*. Первая строка рекурсии *R* показывает, что рекурсия продолжается присоединением на верх списка узлов, собранных до сих пор в левом аргументе, очередного потомка *C* и удалением из правого аргумента всех соединений, направленных к выбранному потомку *C*, за исключением одного, идущего от его родителя *P*. Потомок *C* выбран среди достижимых из выбранного родителя (*P*∨.∧ω), но еще не затронутых (*U*∧*P*∨.∧ ω), и берется произвольным образом как первый из них (<*U*∧*P*∨.∧ω).

Определения P и U показаны во второй строке. P выбирается из тех вершин, у которых имеются потомки среди незатронутых вершин ($\omega\vee.\wedge U$). Над ними выполняется перестановка, упорядочивающая их согласно порядку узлов для левого аргумента ($\alpha\vee.\wedge\omega\vee.\wedge U$), причем они обретают такой порядок, что последний затронутый узел появляется первым, и, наконец, выбирается узел P как первый из них.

Последняя строка из R показывает, что окончательный результат является новым правым аргументом ω, т. е. исходным графом, в котором все соединения, ведущие в каждый узел, вычеркнуты, кроме соединения с его родителем в стягивающем дереве. Поскольку окончательным значением α является квадратная матрица, задающая узлы дерева в порядке, обратном порядку их посещения, то подстановка ω,ф[1]α (или, что эквивалентно, $\omega,\ominus\alpha$) для ω породила бы результат вида 1 $2\times\rho G$, содержащий стягивающее дерево, за которым следует информация о его «предварительном упорядочении».

Часто используется, по крайней мере неявно, другое представление ориентированных графов в виде списка пар узлов V, W, таких, что имеется связь из V в W. Преобразование матрицы соединений в форму такого списка можно описать и использовать следующим образом:

```
          LFC:(,ω)/1+D⊤¯1+⍳×/D←ρω
    C                       LFC C
0 0 1 1              1 1 2 3 3 4
0 0 1 0              3 4 3 2 4 1
0 1 0 1
1 0 0 0
```

Однако это представление является недостаточным, поскольку само по себе оно не определяет число узлов в графе, хотя для данного примера оно задается как /, $LFC\ C$, потому что узел с наибольшим номером случайно имеет соединение. Соответствующее булево представление обеспечивается выражением ($LFC\ C$)$\circ.=\iota 1\uparrow\rho C$, причем первая плоскость показывает исходящие, а вторая — заходящие соединения.

Представление в виде матрицы *инцидентности*, часто используемое при изучении электрических сетей [10], задается как разность этих плоскостей следующим образом:

$$IFC := -\!\!/\,(LFC\omega)\circ.=\iota 1\uparrow\rho\omega$$

Например:

```
      (LFC C)∘.=⍳1↑⍴C                        IFC C
1 0 0 0                               1  0 ¯1  0
1 0 0 0                               1  0  0 ¯1
0 1 0 0                               0  1 ¯1  0
0 0 1 0                               0 ¯1  1  0
0 0 1 0                               0  0  1 ¯1
0 0 0 1                              ¯1  0  0  1

0 0 1 0
0 0 0 1
0 0 1 0
0 1 0 0
0 0 0 1
1 0 0 0
```

При работе с неориентированными графами иногда пользуются представлением, определенным как операция *или* над этими плоскостями (∨⌿). Это выражение эквивалентно |*IFC C*.

Матрица инцидентности *I* характеризуется рядом полезных свойств. Например, +/*I* равняется нулю, +⌿*I* дает *разность* между входными и выходными степенями каждого узла, ⍴*I* дает число дуг, за которым следует число узлов, а ×/⍴*I* дает их произведение. Однако все это легко выражается через матрицу соединений, и более существенные свойства матрицы инцидентности проявляются в ее использовании в электрических сетях. Так, если дуги представляют элементы схем, включенные между узлами, и если *V* — это вектор напряжений в узлах, то напряжения на отдельных ветвях задаются как *I*+.×*V*, если *BI* — вектор токов ветвей, то вектор узловых токов задается как *BI*+.×*I*.

Обратное преобразование из матрицы инцидентности в матрицу соединений задается следующим образом:

```
CFI:D⍴(¯1+⍳×/D)∊D⊥(1 ¯1∘.=⍵)+.×¯1+⍳1↑D←⌊\⌽⍴⍵
```

Функция *принадлежности множеству* ∊ порождает булев массив той же формы, что и ее левый аргумент; этот массив показывает, какие элементы левого аргумента принадлежат правому аргументу.

3.5. СИМВОЛЬНАЯ ЛОГИКА

Булева функция от *N* аргументов может быть представлена булевым вектором из 2*N элементов различными способами, включая формы, иногда называемые *дизъюнктивной, конъюнктивной, эквивалентностью* и *исключающе дизъюнктивной*. Преобразование между любыми двумя из этих форм может быть представлено в сжатом виде как некоторая матрица размера $2*N$ на $2*N$, сформированная из соответствующего внутреннего произведения, например T ∨.∧ ⍉ T, где T←$\overline{T}N$ — это «таблица истинности», построенная по функции $\overline{T}$, описанной в А.2. Эти проблемы детально обсуждаются в [11, гл. 7].

4. ТОЖДЕСТВА И ДОКАЗАТЕЛЬСТВА

В этом разделе мы введем некоторые широко используемые тождества и представим формальные доказательства для части из них, в том числе для симметричных функций Ньютона и для ассоциативности внутреннего произведения, что редко доказывается формально.

4.1. ОТНОШЕНИЯ ДВОЙСТВЕННОСТИ ВО ВНУТРЕННЕМ ПРОИЗВЕДЕНИИ

Разработанные для сведения и развертки отношения двойственности очевидным образом обобщаются на внутренние произведения. Если функция DF двойственная к F, а DG двойственная к G по отношению к одноместной функции M с обратной функцией MI и если A и B — это матрицы, то:

```
A F. G B ←→ MI (M A) DF. DG (M B)
```

Например:

```
A∨.∧B  ←→  ~(~A)∧.∨(~B)
A∧.=B  ←→  ~(~A)∨.≠(~B)
A⌊.+B  ←→  -(-A)⌈.+(-B)
```

Двойственности для внутреннего произведения, сведения и развертки могут служить для того, чтобы избежать применения логического отрицания в различных выражениях, особенно при использовании в конъюнкции с тождествами следующего вида:

```
   A∧(~B)  ←→  A>B
(~A)∧B     ←→  A<B
(~A)∧(~B)  ←→  A⍱B
```

4.2. ТОЖДЕСТВА РАЗБИЕНИЙ

Разбиение массива ведет к ряду очевидных и полезных тождеств. Например:

$$\times/3\ 1\ 4\ 2\ 6 \leftarrow\rightarrow (\times/3\ 1)\times(\times/4\ 2\ 6)$$

В более общем виде для любой ассоциативной функции F

$$F/V \leftarrow\rightarrow (F/K\uparrow V)\ F\ (F/K\downarrow V)$$
$$F/V, W \leftarrow\rightarrow (F/V)\ F\ (F/W)$$

Если функция F не только ассоциативная, но и коммутативная, то нет надобности ограничивать разбиения «префиксами» и «суффиксами», и разбиение может быть построено сжатием по булеву вектору U:

$$F/V \leftarrow\rightarrow (F/U/V)\ \ F\ \ (F/(\sim U)/V)$$

Если E — пустой вектор $(0=\rho E)$, то сведение F/E порождает тождественный элемент для функции F, и поэтому тождества справедливы в предельных случаях $0=K$ и $0=\vee/U$.

Тождества разбиений очевидным образом обобщаются на матрицы. Например, если V, M и A — массивы рангов 1, 2 и 3 соответственно, то

$$V+.\times M \leftarrow\rightarrow ((K\uparrow V)+.\times(K,\ 1\downarrow\rho M)\uparrow M)+$$
$$+(K\downarrow V)+.\times(K,\ 0)\downarrow M \qquad \text{D.1}$$
$$(I,\ J)\downarrow A+.\times V \leftarrow\rightarrow ((I,\ J,\ 0)\downarrow A)+.\times V \qquad \text{D.2}$$

4.3. СУММИРОВАНИЕ И РАСПРЕДЕЛЕНИЕ

Рассмотрим описание и использование следующих функций:

```
N:(∨⌿<\ω∘.=ω)/ω                                   D.3
S:(Nω)∘.=ω                                        D.4

      A←3 3 1 4 1
      C←10 20 30 40 50

      N A                  S A                  (S A)+.×C
3 1 4                1 1 0 0 0             30 80 40
                     0 0 1 0 1
                     0 0 0 1 0
```

Функция $\underline{N}$ выбирает из векторного аргумента его *существо*, т. е. множество различных элементов, которые в нем содержатся. Выражение $\underline{S}A$ дает булеву «матрицу суммирования», которая связывает элементы из A с элементами существа A. Если A — вектор номеров счетов, а C — соответствующий вектор денежных сумм, то выражение $(\underline{S}A)+.\times C$ вычисляет общие суммы, или «суммирует» сделки по нескольким номерам счетов, представленным в A.

Используемая как последующий множитель в выражении вида $W+.\times \underline{S}\ A$ матрица *суммирования* может служить для распределения результатов. Например, если F — функция, вычисление которой обходится дорого, и её аргумент V содержит повторяющиеся элементы, то, может быть, более эффективно было бы применить F только к существу для V и распределить результаты тем способом, который подсказывается следующим тождеством:

$$F\ V \leftarrow\rightarrow (F\ \underline{N}\ V)+.\times \underline{S}\ V \qquad \text{D.5}$$

Порядок элементов из $\underline{N}V$ такой же, как и их порядок в V, и иногда бывает удобнее использовать *упорядоченное* существо и соответствующее *упорядоченное* суммирование по формулам

$$\underline{O}\underline{N}:\underline{N}\omega[\Delta\omega] \qquad \text{D.6}$$
$$\underline{O}\underline{S}:(\underline{O}\underline{N}\omega)\circ.=\omega \qquad \text{D.7}$$

Тождество, соответствующее формуле D.5, имеет вид

$$F\ V \leftarrow\rightarrow (F\ \underline{O}\ \underline{N}\ V)+.\times \underline{O}\ \underline{S}\ V \qquad \text{D.8}$$

Функция суммирования даёт интересный результат при ее применении к функции I, описанной с помощью A.2:

$$+/\underline{S}+\not{/}\underline{T}N \leftarrow\rightarrow (0,\iota N)!N$$

Иначе говоря, сумма строк матрицы суммирования для сумм столбцов матрицы подмножества порядка N образует вектор биномиальных коэффициентов порядка N.

4.4. ДИСТРИБУТИВНОСТЬ

Дистрибутивность одной функции относительно другой является важным математическим понятием, и теперь мы займемся вопросом представления этого понятия в общем виде. Поскольку умножение дистрибутивно относительно сложения, стоящего справа от него, то мы имеем $\mathbf{a}\times(\mathbf{b}+\mathbf{q})\leftarrow\rightarrow\mathbf{ab}+\mathbf{aq}$, а ввиду дистрибутивности слева получаем $(\mathbf{a}+\mathbf{p})\times\mathbf{b}\leftarrow\rightarrow\mathbf{ab}+\mathbf{pb}$. Отсюда получаем более общие формулы:

```
(a+p)×(b+q) ↔ ab+aq+pb+pq
(a+p)×(b+q)×(c+r) ↔ abc+abr+aqc+aqr+pbc+pbr+pqc+pqr
(a+p)×(b+q)×...×(c+r) ↔ ab...c+...+pq...r
```

Воспользовавшись тем, что $V\leftarrow A,B$ и $W\leftarrow P,Q$ или $V\leftarrow A,B,C$ и $W\leftarrow P$, Q, R и т. д., можно записать левую часть просто через сведение как $\times/V+W$. Для этого случая трех элементов правая часть может быть записана как сумма произведений по всем столбцам следующей матрицы:

```
V[0]  V[0]  V[0]  V[0]  W[0]  W[0]  W[0]  W[0]
V[1]  V[1]  W[1]  W[1]  V[1]  V[1]  W[1]  W[1]
V[2]  W[2]  V[2]  W[2]  V[2]  W[2]  V[2]  W[2]
```

Приведенная выше структура распределения V и W в точности совпадает с распределением *нулей* и *единиц* в матрице $T\leftarrow\underline{T}\rho V$, и поэтому произведения вдоль столбцов задаются как $(V\times.*\sim T)\times(W\times.*T)$. Следовательно,

$$\times/V+W\leftarrow\rightarrow+/(V\times.*\sim T)\times W\times.*T\leftarrow\underline{T}\ \rho V \qquad \text{D.9}$$

Теперь мы представим формально индуктивное доказательство для D.9, приняв индуктивное предположение, что D.9 справедливо для всех V и W формы N (т. е. $\wedge/N=(\rho V),\rho W)$, и доказав, что оно справедливо для формы $N+1$, т. е. для X, V и Y, W, где X и Y — это произвольные скаляры.

Для использования в индуктивном доказательстве мы сначала дадим рекурсивное описание функции $\underline{T}$, эквивалентное A.2 и основанное на следующем утверждении; если $M\leftarrow\underline{T}\,2$ является результатом порядка 2, то

```
      M
0 0 1 1
0 1 0 1
```

```
      0,[1]M              1,[1]M
0 0 0 0              1 1 1 1
0 0 1 1              0 0 1 1
0 1 0 1              0 1 0 1
```

```
      (0,[1]M),(1,(1)M)
0 0 0 0 1 1 1 1
0 0 1 1 0 0 1 1
0 1 0 1 0 1 0 1
```

Итак:

```
T:(0,[1]T),(1,[1]T←Tω-1):0=ω:0 1ρ0                                D.10

+/((C←X,V)×.*~Q)×D×.*Q←Tρ(D←Y,W)
+/(C×.*~Z,U)×D×.*(Z←0,[1] T),U←1,[1] T←TρW                        D.10
+/((C×.*~Z),C×.*~U)×(D×.*Z),D×.*U                               Note 1
+/((C×.*~Z),C×.*~U)×((Y*0)×W×.*T),(Y*1)×W×.*T                   Note 2
+/((C×.*~Z),C×.*~U)×(W×.*T),Y×W×.*T                         Y*0 1←→1,Y
+/((X×V×.*~T),V×.*~T)×(W×.*T),Y×W×.*T                           Note 2
+/(X×(V×.*~T)×W×.*T),(Y×(V×.*~T)×W×.*T)                         Note 3
+/(X××/V+W),(Y××/V+W)                              Induction hypothesis
+/(X,Y)××/V+W                                      (X×S),(Y×S)←→(X,Y)×S
×/(X+Y),(V+W)                                          Definition of ×/
×/(X,V)+(Y,W)                                      + distributes over ,
```

Note 1: $M+.\times N,P \leftarrow\rightarrow (M+.\times N),\ M+.\times P$ (тождество разбиения для матриц)
Note 2: $V+.\times M \leftarrow\rightarrow ((1\uparrow V)+.\times(1,1\downarrow\rho M)\uparrow M)+(1\downarrow V)+.\times 1\ 0\downarrow M$ (тождество разбиения для матриц и описание C, D, Z и U)
Note 3: $(V;\ W)\times P,\ Q \leftarrow\rightarrow (V\times P),\ W\times Q$

Чтобы завершить индуктивное доказательство, нам нужно показать, что предполагаемое тождество D.9 справедливо для некоторого значения N. Если $N=0$, то векторы A и B пустые и поэтому $X,A \leftarrow\!\!-\!\!-,X$ и $Y,B\leftarrow\rightarrow,Y$. Следовательно, левая часть приобретает вид $+/X+Y$ или просто $X+Y$. Правая часть приобретает вид $+/(X\times.\ \sim Q)\times Y\times.\ Q$, где $\sim Q$ — это однострочная матрица 1 0, а Q — это 0 1. Поэтому правая сторона эквивалентна $+/(X,1)\times(1,Y)$ или $X+Y$. Может оказаться поучительным аналогичное исследование случая $N=1$.

4.5. СИММЕТРИЧНЫЕ ФУНКЦИИ НЬЮТОНА

Если X — это скаляр, а R — любой вектор, то $\times/X—R$ является полиномом над X с корнями R. Поэтому он эквивалентен некоторому полиному $C\ \underline{P}\ X$ и признание эквивалентности означает, что C представляет собой функцию от R. Теперь мы используем D.8 и D.9 для вывода этой функции, которая обычно строится на основе симметричных функций Ньютона:

```
×/X-R
×/X+(-R)
+/(X×.*~T)×(-R)×.*T←T ρR                                  D.9
(X×.*~T)+.×P←(-R)×.*T                              Def of +.×
(X*S←+/~T)+.×P                                         Note 1
((X*QN S)+.×QS S)+.×P                                     D.8
(X*QN S)+.×((QS S)+.×P)                     +.× is associative
(X*0,ιρR)+.×((QS S)+.×P)                               Note 2
((QS S)+.×P)P X                                B.1 (polynomial)
((QS +/~T)+.×((-R)×.*T←T ρR))P X                     Defs of S
                                                        and P
```

Note 1: Если X — это *скаляр* и B — булев вектор, то X×.*B←→X*×/B.
Note 2: Поскольку T — булева матрица и содержит ρR строк, то суммы ее столбцов ранжируются от 0 до ρR, и поэтому их упорядоченная сущность равна 0, ιρR.

4.6. ДВУМЕСТНАЯ ТРАНСПОЗИЦИЯ

Обозначаемая через ⍉ двуместная транспозиция представляет собой обобщение одноместной транспозиции, которое представляет оси правого аргумента и (или) формирует «секторы» правого аргумента слиянием определенных осей, определяемых левым аргументом. Мы вводим здесь эту операцию как удобное средство изучения свойств внутреннего произведения.

Двуместная транспозиция будет описана формально с помощью функции выбора

SF: (,ω)[1+(ρω)⊥α—1]

извлекающей из своего правого аргумента элемент, индексы которого задаются векторным левым аргументом. Разумеется, его форма должна равняться рангу правого аргумента. Ранг результата K⍉A равен ⌈/K, и если I — это любой подходящий левый аргумент функции выбора I SF I⍉ A, то:

I SF K⍉A← →(I[K]) SF A D.11

Например, если M — это матрица, то 2 1⍉ M←→⍉M и 1 1⍉ M является диагональю матрицы M; если T — массив ранга 3, то 1 2 2⍉ T — это матрица «диагонального разреза» T, полученная выделением последних двух осей, а вектор 1 1 1⍉ T является главной диагональю для T.

В дальнейшем будет использовано следующее тождество:

J⍉K⍉A← → (J[K])⍉A D.12

Доказательство:

```
I SF J⍉K⍉A
(I[J]) SF K⍉A                 Definition of ⍉ (D.11)
((I[J])[K]) SF A              Definition of ⍉
(I[(J[K])]) SF A              Indexing is associative
I SF(J[K])⍉A                  Definition of ⍉
```

4.7. ВНУТРЕННИЕ ПРОИЗВЕДЕНИЯ

Следующие доказательства сформулированы только для матричных аргументов и для конкретного внутреннего произведения +.×. Они легко обобщаются на массивы более высокого ранга и на другие внутренние произведения $F.G$, где F и G должны обладать только свойствами, предполагаемыми в доказательствах для $+$ и $\times$.

Следующее тождество (известное в математике как сумма по матрицам, сформированным как (внешние) произведения столбцов из первого аргумента на соответствующие строки из второго аргумента) будет использоваться при установлении ассоциативности и дистрибутивности внутреннего произведения:

$$M+.\times N \leftarrow\rightarrow +/1\ 3\ 3\ 2 \otimes M\circ.\times N \qquad \text{D.13}$$

Доказательство: $(I, J)SF\,M+.\times N$ описывается как сумма по V, где $V[K] \leftarrow\rightarrow M[I, K]\times N[K; J]$. Аналогично

$$(I,\ J)SF+/1\ 3\ 3\ 2\otimes M\circ.\times N$$

является суммой по вектору W, такому, что

$$W[K] \leftarrow\rightarrow (I,\ J,\ K)SF\ 1\ 3\ 3\ 2\otimes M\circ.\times N$$

Итак:

```
W[K]
(I,J,K)SF 1 3 3 2 ⍉M∘.×N
(I,J,K)[1 3 3 2]SF M∘.×N
(I,K,K,J)SF M∘.×N
M[I;K]×N[K;J]
V[K]
```

Матричное произведение дистрибутивно относительно сложения следующим образом:

$$M+.\times(N+P) \leftarrow\rightarrow (M+.\times N)+(M+.\times P) \qquad \text{D.14}$$

Доказательство:

```
M+.×(N+P)
+/(J← 1 3 3 2)⍉M∘.×N+P          D.13
+/J⍉(M∘.×N)+(M∘.×P)            × distributes over +
+/(J⍉M∘.×N)+(J⍉M∘.×P)          ⍉ distributes over +
(+/J⍉M∘.×N)+(+/J⍉M∘.×P)        + is assoc and comm
(M+.×N)+(M+.×P)                D.13
```

Матричное произведение ассоциативно согласно формуле

$$M+.\times(N+.\times P) \leftarrow\rightarrow (M+.\times N)+.\times P \qquad \text{D.15}$$

Доказательство: Сначала сводим каждую строку к суммам по разделам внешнего произведения и затем сравниваем суммы. Комментирование второго сведения предоставляется читателям:

```
M+.×(N+.×P)
M+.×+/1 3 3 2⍉N∘.×P                                   D.12
+/1 3 3 2⍉M∘.×+/1 3 3 2⍉N∘.×P                         D.12
+/1 3 3 2⍉+/M∘.×1 3 3 2⍉N∘.×P          × distributes over +
+/1 3 3 2⍉+/1 2 3 5 5 4⍉M∘.×N∘.×P                   Note 1
+/+/1 3 3 2 4 ⍉1 2 3 5 5 4⍉M∘.×N∘.×P                Note 2
+/+/1 3 3 4 4 2⍉M∘.×N∘.×P                             D.12
+/+/1 3 3 4 4 2⍉(M∘.×N)∘.×P                × is associative
+/+/1 4 4 3 3 2⍉(M∘.×N)∘.×P      + is associative and
                                            commutative

(M+.×N)+.×P
(+/1 3 3 2⍉M∘.×N)+.×P
+/1 3 3 2⍉(+/1 3 3 2⍉M∘.×N)∘.×P
+/1 3 3 2⍉+/1 5 5 2 3 4⍉(M∘.×N)∘.×P
+/+/1 3 3 2 4⍉1 5 5 2 3 4⍉(M∘.×N)∘.×P
+/+/1 4 4 3 3 2⍉(M∘.×N)∘.×P

Note 1: +/M∘.×J⍉A ←→ +/((⍳⍴⍴M),J+⍴⍴M)⍉M∘.×A

Note 2: J⍉+/A ←→ +/(J,1+⌈/J)⍉A
```

4.8. ПРОИЗВЕДЕНИЕ ПОЛИНОМОВ

Тождество B.2, использованное для умножения полиномов, будет теперь разработано формально

```
(B P X)×(C P X)
(+/B×X*E←¯1+⍳⍴B)×(+/C×X*F←¯1+⍳⍴C)          B.1
+/+/(B×X*E)∘.×(C×X*F)                      Note 1
+/+/(B∘.×C)×((X*E)∘.×(X*F))                Note 2
+/+/(B∘.×C)×(X*(E∘.+F))                    Note 3
```

Примечание 1: $(+/V)\times(+/W) \longleftrightarrow +/+/V\circ.\times X$, потому что $\times$ распределяется по $+$ и операция $+$ ассоциативна и коммутативна, или см. доказательство в [12, с. 21].
Примечание 2: Эквивалентность $(P\times V)\circ.\times(Q\times W)$ и $(P\circ.\times Q)\times(V\circ.\times W)$ может быть установлена исследованием типового элемента из каждого произведения.
Примечание 3: $(X*I)\times(X*J) \longleftrightarrow X*(I+J)$.

Приведенное выше доказательство представлено в сокращенной форме Ортом [13, p. 52], который описал также функции для композиции полиномов.

4.9. ПРОИЗВОДНАЯ ОТ ПОЛИНОМА

Полиномиальные функции удобно использовать в вводном курсе численного анализа, потому что они пригодны для аппроксимации множества полезных функций и потому что класс этих функций замкнут относительно сложения, умножения, композиции, дифференцирования и интегрирования. Однако в курсе элементарных вычислений рассмотрение таких функций обычно запаздывает, так как подход к производной полинома осуществляется, как отмечалось в разд. 2, окольным путем, т. е. через последовательность более общих результатов.

Ниже показан вывод производной полинома непосредственно из выражения для тангенса угла наклона секущей линии через точки X, FX и $(X+Y)$, $F(X+Y)$:

```
((C P̲ X+Y)-(C P̲ X))÷Y
((C P̲ X+Y)-(C P̲ X+0))÷Y
((C P̲ X+Y)-((0*J)+.×(A←DS J∘.!J←¯1+ιρC)+.×C) P̲ X)÷Y     B.6
((((Y*J)+.×M) P̲ X)-((0*J)+.×M←A+.×C) P̲ X)÷Y               B.6
((((Y*J)+.×M)-(0*J)+.×M) P̲ X)÷Y                 P̲ dist over -
((((Y*J)-0*J)+.×M) P̲ X)÷Y                       +.× dist over -
(((0,Y*1↓J)+.×M) P̲ X)÷Y                                 Note 1
(((Y*1↓J)+.× 1 0 ↓M) P̲ X)÷Y                                D.1
(((Y*1↓J)+.×(1 0 0 ↓A)+.×C) P̲ X)÷Y                         D.2
((Y*1↓J-1)+.×(1 0 0 ↓A)+.×C) P̲ X             (Y*A)÷Y←→Y*A-1
((Y*¯1+ι¯1+ρC)+.×(1 0 0 ↓A)+.×C) P̲ X                Def of J
(((Y*¯1+ι¯1+ρC)+.× 1 0 0 ↓A)+.×C) P̲ X                   D.15
```

Note 1: `0*0←→1←→Y*0` **and** `∧/0=0*1↓J`

Производная представляет собой предельное значение тангенса угла наклона секущей при $Y \to 0$, и последнее из вышеприведенных выражений поддается в этом случае вычислению, так как если $E \leftarrow {}^{-}1+\iota^{-}1+\rho C$ есть вектор экспонент для Y, то все элементы из E неотрицательные. Кроме того, $0*E$ сводится к единице, за которой следуют нули, а поэтому внутреннее произведение для $1\ 0\ 0{\downarrow}A$ сводится к первой плоскости из $1\ 0\ 0{\downarrow}A$ или, что эквивалентно, ко второй плоскости из A.

Если $B \leftarrow J\circ.!J \leftarrow {}^{-}1+\iota\rho C$ является матрицей биномиальных коэффициентов, то A — это $DS\ B$ и, согласно описанию DS в B.5, вторая плоскость из A представляет собой $B \times 1 = -J\circ.-J$, т. е. матрицу B, в которой все элементы, за исключением первой супердиагонали, заменены нулями. Поэтому окончательное выражение для коэффициентов полинома, который является производной от полинома $C\underline{P}\omega$, имеет вид

```
((J∘.!J)×1=-J∘.-J←¯1+ιρC)+.×C
```

Например:

```
      C ← 5 7 11 13
      (J∘.!J)×1=-J∘.-J←¯1+⍳⍴C
0 1 0 0
0 0 2 0
0 0 0 3
0 0 0 0
      ((J∘.!J)×1=-J∘-J←¯1+⍳⍴C)+.×C
7 22 39 0
```

Поскольку супердиагональ матрицы биномиальных коэффициентов $(\iota N)\circ.!\iota N$ представляет собой $(\_1+\iota N-1)!\iota N-1$, или просто $\iota N-1$, окончательный результат равен $1\phi C\times^{-}1+\iota\rho C$ в соответствии с приведенным выше выводом.

Завершая обсуждение доказательств, снова подчеркнем тот факт, что все утверждения (выражения) в предшествующих доказательствах исполнимы (вычислимы), и поэтому для обнаружения ошибок можно использовать компьютер. Например, воспользовавшись узловым описанием канонической функции [4, с. 81], можно следующим образом описать некую функцию F, шагами вывода которого являются четыре строки из предыдущего доказательства:

```
    ∇F
[1] ((C P X+Y)-(C P X))÷Y
[2] ((C P X+Y)-(C P X+0))÷Y
[3] ((C P X+Y)-((0*J)+.×(A←DS J∘.!J←¯1+⍳⍴C)+.×C) P X)÷Y
[4] ((((Y*J)+.×M) P X)-((0*J)+.×M←A+.×C) P X)÷Y
    ∇
```

Затем можно выполнить эти шаги доказательства, присвоив значения переменным и вычислив F следующим образом:

```
      C←5 2 3 1
      Y←5
      X←3            X←⍳10
      F              F
132         66 96 132 174 222 276 336 402 474 552
132         66 96 132 174 222 276 336 402 474 552
132         66 96 132 174 222 276 336 402 474 552
132         66 96 132 174 222 276 336 402 474 552
```

Между строками могут быть включены комментарии, не влияющие на выполнение.

5. ЗАКЛЮЧЕНИЕ

В предшествующих разделах была предпринята попытка разработать тезис о том, что свойства исполнимости и универсальности, присущие языкам программирования, можно объединить в одном языке с хорошо известными свойствами математической нотации, которые делают ее столь эффективным инструментом мышления. Этому важному вопросу следует уделять дальнейшее внимание вне зависимости от успеха или неудачи данной попытки разработать его в терминах языка APL.

В частности, я хотел бы надеяться, что другие исследователи станут заниматься тем же вопросом с помощью иных языков программирования и обычной математической нотации. Если эти рассмотрения окажутся нацеленными на те общие аспекты, которыми мы занимались здесь, то можно будет провести некоторые объективные сравнения языков. Уже доступны для сравнения альтернативные рассмотрения некоторых представленных здесь тем. Например, Кернер [7] выразил алгоритм С.3 как на Алголе, так и в общепринятой математической нотации.

Этот заключительный раздел является более общим и посвящен сравнениям с математической нотацией, проблемам введения нотации, обобщениям языка APL, которые способствовали бы дальнейшему увеличению его полезности, и обсуждению вариантов представления материала предшествующих разделов.

5.1. СРАВНЕНИЕ С ОБЫЧНОЙ МАТЕМАТИЧЕСКОЙ НОТАЦИЕЙ

Для любого недостатка, замеченного в математической нотации, вероятно, можно найти пример его исправления в некотором конкретном разделе математики или в некоторой конкретной публикации; проводимые здесь сравнения ориентированы на более общие и типичные применения математической нотации.

Язык APL подобен обычной математической нотации во многих важных отношениях: в нем используются функции с явными аргументами и явными результатами, употребляются выражения, в которых участвуют функции от результатов других функций; для часто используемых функций имеются графические символы, применяются операторы, которые, подобно математическим операторам дифференцирования и свертки, применяются к функциям для получения функций.

Особенность подхода к функциям в APL состоит в обеспечении точного формального механизма для описания новых функций. Используемая в этой статье форма прямого определения, вероятно, лучше всего подходит для целей демонстрации и анализа, но каноническая форма, упоминаемая во введении и описанная в [4, с. 81], часто оказывается более удобной для других целей.

Интерпретация сложных выражений в APL согласуется с обычной нотацией в употреблении скобок, но отличается отсутствием иерархии, обеспечивающим единообразный подход ко всем функциям (как определенным пользователем, так и примитивным), и применением одного и того же правила к одноместным и двуместным функциям: правым аргументом функции является все выражение, стоящее справа от нее. Из этого правила вытекает важное следствие, состоящее в том, что любая часть выражения, свободная от скобок, может *аналитически* читаться слева направо (потому что на любом этапе самая левая функция является «внешней», или общей, функцией, которая должна быть применена к результату, стоящему справа от нее) и *конструктивно* читаться справа налево (поскольку легко видеть, что это правило эквивалентно правилу *исполнения* справа налево).

Хотя Кайори [2] в своей двухтомной истории математических нотаций даже не упомянул правила для порядка исполнения, тем не менее представляется разумным предположить, что мотивация привычной иерархии (возведение в степень выполняется перед умножением, а умножение — перед сложением или вычитанием) возникла из желания записывать полиномы без скобок. Удобство применения векторов при выражении полиномов, например $+/C\times X*E$, во многом способствует устранению этой мотивации. Кроме того, принятое в языке APL правило также обосновывает применимость без скобок эффективной схемы Горнера для выражения полинома:

$$+/3\ 4\ 2\ 5\times X*0\ 1\ 2\ 3 \leftarrow\!\!\!\rightarrow 3+X\times 4+X\times 2+X\times 5.$$

В обеспечении графических символов для общеупотребимых функций APL идет значительно дальше и предусматривает такие символы для функций (например, для степенной функции), которые неявно отвергаются в математике. Это обстоятельство становится важным при введении операторов; в предшествующих разделах внутреннее произведение $\times.*$ (в котором должен применяться символ для степени) играет такую же роль, как и обычное внутреннее произведение $+.\times$. Запрет на отбрасывание символов функций (например, символа $\times$) делает возможным недвусмысленное употребление многосимвольных имен для переменных и функций.

В отношении использования массивов язык APL сходен с математической нотацией, но характеризуется большей систематичностью. Например, $V+W$ имеет одинаковый смысл в обеих нотациях, и в APL определения других функций обобщаются аналогичным поэлементным способом. Однако в математике такие выражения, как $V\times W$ и $V*W$, определяются по-другому или вовсе не определены.

$$\sum_{j=1}^{n} j \cdot 2^{-j}$$

$$1\cdot2\cdot3 + 2\cdot3\cdot4 + \ldots n \text{ членов} \leftarrow \rightarrow \frac{1}{4}n(n+1)(n+2)(n+3)$$

$$1\cdot2\cdot3\cdot4 + 2\cdot3\cdot4\cdot5 + \ldots n \text{ членов} \leftarrow \rightarrow \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$$

$$\frac{\left[\frac{x-a}{N}\right]^{-q}}{\Gamma(-q)} \sum_{j=0}^{N-1} \frac{\Gamma(j-q)}{\Gamma(j+1)} f\left(x - j\left[\frac{x-a}{N}\right]\right)$$

Рис. 3.

Так, $Y \times W$ обычно означает *векторное произведение* [14]. На языке APL оно может быть выражено различными способами. Определение

$$VP: ((1\phi\alpha)\times{}^{-}1\phi\omega) - ({}^{-}1\phi\alpha)\times 1\phi\omega$$

обеспечивает удобную основу для очевидного доказательства того, что VP является «антикоммутативным» (т. е. $V\ VP\ W \leftarrow\rightarrow -W\ VP\ V$) и (с учетом того, что ${}^{-}1\phi X \leftarrow\rightarrow 2\phi X$ для трехэлементных векторов) позволяет легко доказать, что в трехмерном пространстве V и W ортогональны своему векторному произведению, т. е. $\wedge/0 = V+.\times V\ VP\ W$ и $\wedge/0 = W+.\times V\ VP\ W$.

Язык APL более систематичен также в использовании операций для получения функций на массивах; сведение обеспечивает эквиваленты для обозначений sigma и pi (в виде $+/$ и $\times/$) и для множества аналогичных полезных ситуаций; внешнее произведение обобщает внешнее произведение из тензорного анализа на функции, отличающиеся от $\times$, а внутреннее произведение обобщает матричное произведение ($+.\times$) на многие случаи, например на $\vee.\wedge$ и $\lfloor.+$, для которых часто вводятся описания применительно к конкретному случаю.

Сходство между языком APL и традиционной нотацией становится яснее, если выучить несколько довольно механических подстановок; поучительно и преобразование математических выражений. Например, в таком выражении, как показанное первым на рис. 3, производится простая подстановка ιN вместо всякого появления j и замена знака Σ на $+/$. Итак:

$$+/(\iota N)\times 2*-\iota N \quad \text{или} \quad +/J\times 2*-J\leftarrow\iota N$$

В руководстве по суммированию рядов [15] представлены интересные выражения для таких упражнений, особенно если результаты можно вычислить на компьютере. Например, на с. 8

и 9 из [15] приведены тождества, показанные во втором и третьем примерах на рис. 3. Их можно было бы записать так:

$$+/\times/(^-1+\iota N)\circ.+\iota 3 \longleftrightarrow (\times/N+0,\ \iota 3)\div 4$$

$$+/\times/(^-1+\iota N)\circ.+\iota 4 \longleftrightarrow (\times/N+0,\ \iota 4)\div 5$$

В совокупности они подсказывают следующее тождество:

$$+/\times/(^-1+\iota N)\circ.+\iota K \longleftrightarrow (\times/N+0,\ \iota K)\div K+1$$

Читатель может попытаться переформулировать это общее тождество (или хотя бы его частный случай при $K=0$) в нотации из [15].

Последнее выражение на рис. 3 заимствовано из исследования по функциональному исчислению [16] и представляет аппроксимацию для производной порядка q от функции f. Оно может быть записано так:

$$(S*-Q)\times+/(J\,!\,J-1+Q)\times F\ X-(J\leftarrow{}^-1+\iota N)\times S\leftarrow(X-A)\div N$$

Преобразование на языке APL сводится к простому использованию ιN (как предлагалось выше) в сочетании с очевидным тождеством, сводящим несколько появлений гамма-функции в одно применение функции биномиальных коэффициентов «!», область определения которой, разумеется, не ограничивается целыми числами.

В предыдущем выражении положительное значение параметра Q указывает порядок производной, а отрицательное — порядок интеграла (от A до X). Дробные значения задают дробные производные и интегралы, и, описав сначала функцию F и присвоив подходящие значения для N и A, можно использовать для вычислительного эксперимента с рассмотренными в [16] производными следующую функцию:

$$OS:(S*-\alpha)\times+/(J\,!\,J-1+\alpha)\times F\omega-(J\leftarrow{}^-1+\iota N)\times$$
$$\times S\leftarrow(\omega-A)\div N$$

Несмотря на широкое использование «формальных» манипуляций в математической нотации, по-настоящему формальные манипуляции посредством явных алгоритмов весьма затруднительны. В этом отношении язык APL оказывается гораздо более подходящим. Например, в разд. 2 мы видели, что производная от полиномиального выражения $(\omega\circ.*{}^-1+\iota\rho\alpha)+.\times\alpha$ задается как $(\omega\circ.*{}^-1+\iota\rho\alpha)+.\times 1\phi\alpha\times{}^-1+\iota\rho\alpha$, и набор функций для формального дифференцирования выражений на APL, представленный в [13], занимает менее одной страницы. Другие примеры функций для формальных манипуляций встречаются в [17] применительно к операторам моделирования для векторного исчисления.

Дальнейшее рассмотрение связи с математической нотацией можно найти в [3] и в статье «Алгебра как язык» [6, с. 325].

Заключительное замечание касается печати, которая является серьезной проблемой для традиционной нотации. Хотя в языке APL применяются некоторые символы, как правило еще не доступные для издателей, в нем задействованы только 88 основных символов плюс несколько составных символов, сформированных суперпозициями пар основных символов. Кроме того, он не предъявляет таких требований, как подстрочные и надстрочные линии и уменьшенные шрифты для нижних и верхних индексов.

5.2. ВВЕДЕНИЕ НОТАЦИИ

В самом начале этой статьи предлагалось оценивать удобство нотации по легкости ее включения в контекст, и мы просили читателя наблюдать процесс введения нотации APL. Полезность такого критерия вполне можно счесть тривиальным фактом, но тем не менее она требует некоторого пояснения.

Во-первых, целевая нотация, предусмотренная именно для функций, нужных в некоем конкретном деле, тоже легко вводилась бы в контекст. Необходимо задать дополнительные вопросы, относящиеся к общему объему требуемой нотации, глубине структуры нотации и к тому, в какой мере нотация, введенная для специфической цели, оказывается полезной в более общих случаях.

Во-вторых, важно различать трудность описания и изучения какого-то подмножества обозначений и трудность овладения следствиями. Например, легко изучить правила вычисления произведения матриц, но совсем другое и притом гораздо более трудное дело — овладеть его следствиями (такими, как ассоциативность, дистрибутивность относительно сложения и возможность представлять линейные функции и геометрические операции).

В самом деле, сама содержательность нотации может затруднить ее изучение из-за обилия свойств, исследование которых она подсказывает. Например, нотация +.× для произведения матриц не может затруднить освоение правил его вычисления, потому что эта нотация по крайней мере напоминает, что процесс сводится к суммированию произведений, однако любое обсуждение свойств произведения матриц в терминах этой нотации не может не поставить множество вопросов, таких, например, как: ассоциативна ли операция ∨.∧? относительно какой операции она дистрибутивна? верно ли тождество B∨.∧C←→⍉(⍉C)∨.∧⍉B?

5.3. ОБОБЩЕНИЯ ДЛЯ APL

Чтобы гарантировать корректность и широкую доступность на существующих вычислительных машинах нотации, используемой в этой статье, мы ограничились современным состоянием языка APL, описанным в [4], и более формальным стандартом, опубликованным STAPL, техническим комитетом ACM SIGPLAN по языку APL [17]. Здесь мы кратко прокомментируем потенциальные возможности, которые увеличили бы удобство этой нотации для рассматриваемых нами аспектов и в большей степени приспособили бы ее для других приложений, например для обычного и векторного исчисления.

Один тип расширения уже предлагался, когда мы продемонстрировали выполнение примера (корни полинома) в системе APL, основанной на комплексных числах. Это расширение не влечет за собой изменений функциональных символов, хотя и потребуется расширить области определения некоторых функций. Например, $|X$ будет давать модули не только вещественных, но и комплексных аргументов, $+X$ даст сопряженное значение для комплексного аргумента наряду с тем тривиальным результатом, который ныне получается для вещественных аргументов; соответствующим образом будут обобщены и элементарные функции, как видно из применения $*$ в приведенном примере. Подразумевается возможность осмысленного включения примитивных функций для нулей полиномов, а также для собственных значений и собственных векторов матриц.

Другой тип, также предложенный в предшествующих разделах, включает функции, определенные для конкретных целей, которые могли бы пригодиться для общего пользования. Примерами являются функция $\underline{N}$ *сущность*, определенная в D.3, и функция $\underline{S}$ *суммирования*, определенная в D.4. Эти и другие обобщения обсуждаются в [18]. Макдоннелл [19] предложил обобщения функций *и* и *или* на небулевы значения, чтобы $A \vee B$ являлось наибольшим общим делителем (GCD) для A и B, а $A \wedge B$ являлось наименьшим общим кратным (LCM). Функции GCD и LCD, описанные в разд. 3, могли бы быть потом описаны просто как GCD : $\vee/\omega$ и LCM : $\wedge/\omega$.

Более общая линия развития затрагивает операции; она проиллюстрирована в предыдущем разделе на примерах операций сведения, внутреннего произведения и внешнего произведения. Обсуждение операций, ныне включенных в APL, можно найти в [20] и [17]; предлагаемые новые операции для векторного исчисления рассматриваются в [17], а другие обсуждаются в [18] и [17].

5.4. СПОСОБ ПРЕДСТАВЛЕНИЯ

В предшествовавших разделах рассмотрен ряд несложных тем, причем упор делался на ясность, а не на эффективность получаемых в результате алгоритмов. Оба этих качества заслуживают дополнительных пояснений.

Несколько иная и, возможно, более реалистичная проверка нотации связана с рассмотрением более сложных тем в объеме, достаточном, скажем, для одно- или двухсеместрового курса. В частности, это позволяет лучше оценить, какой объем нотации можно ввести в рамках обычного учебного курса.

Такие анализы проведены для ряда тем, включая высшую алгебру [6], элементарный анализ [5], функциональный анализ [13], проектирование цифровых систем [21], резистивные схемы [10] и кристаллографию [22]. Во всех этих случаях отмечается легкость введения нужной нотации, а в одном обсуждается опыт ее использования. Профессор Блейв, обсуждая проектирование цифровых систем [21], утверждает, что «APL позволяет описывать, что на самом деле происходит в сложной системе... Язык APL особенно удобен для этой цели, так как допускает выражения, относящиеся к верхнему уровню архитектуры, к нижнему уровню и ко всем промежуточным уровням... Изучение языка окупается как внутри, так и вне области проектирования компьютеров».

Пользователи компьютеров и языков программирования часто в первую очередь обращают внимание на эффективность исполнения алгоритмов и поэтому могли бы в итоге пренебречь многими представленными здесь алгоритмами. Такое пренебрежение оказалось бы недальновидным, потому что ясная формулировка алгоритма обычно может служить основой, из которой легко выводятся более эффективные алгоритмы. Например, для функции STEP из разд. 3.2 можно значительно повысить эффективность, выполнив подстановки вида *B*⌹*M* для (⌹*M*)+.×*B*, а в выражениях, содержащих +/*C*×*X**¯1+⍳⍴*C*, можно подставить X⊥⌽C или, если принят противоположный порядок коэффициентов, выражение *X*⊥*C*.

Можно производить и более сложные преобразования. Метод Кернера (C.3) получается из довольно очевидного, хотя формально и не установленного тождества. Аналогично использование матрицы α для предоставления перестановок в рекурсивной функции *R*, служащей для получения стягивающего дерева глубины один (C.4), может быть заменено на, возможно, более компактное применение списка узлов, причем индексы для внутренних произведений представляются довольно очевидным, хотя и не совсем формальным способом. Кроме того, такое рекурсивное описание может быть преобразовано в более эффективные нерекурсивные формы.

Наконец, любой алгоритм, ясно выраженный в терминах массивов, может быть преобразован простыми, хотя и утомительными модификациями в, по-видимому, более эффективные алгоритмы с применением итерации для скалярных элементов. Например, вычисление $+/X$ зависит от каждого элемента из X и не допускает заметного улучшения, но вычисление $\vee/B$ могло бы прекратиться на первом элементе, равном 1, и поэтому может быть улучшено итеративным алгоритмом, выраженным в терминах индексации.

Для математики весьма типично, когда сначала разрабатывается ясное и точное описание процесса безотносительно к эффективности, а затем оно используется в качестве руководства и теста для изучения эквивалентных процессов, обладающих другими характеристиками, например большей эффективностью. Это весьма плодотворная практика, которую не следует нарушать преждевременной заботой об эффективности компьютерного исполнения.

Оценки эффективности часто оказываются нереалистичными, потому что они принимают в расчет «основные» функции, например умножение и сложение, и пренебрегают вспомогательными (индексацией и другими процессами выбора), которые часто играют весьма важную роль в менее прямолинейных алгоритмах. Кроме того, реалистичные оценки в значительной мере зависят от современного проектирования компьютеров и от языковой реализации алгоритмов. Например, поскольку была обнаружена интенсивность использования в языке APL функций над булевыми значениями (таких, как $\wedge/B$ и $\vee/B$), разработчики обеспечили их эффективное выполнение. В конечном счете чрезмерное внимание к эффективности приводит к порочному кругу в проектировании: из соображений эффективности ранние языки программирования отражали характеристики ранних компьютеров и каждое новое поколение компьютеров отражало потребности языков программирования, разработанных для предыдущего поколения компьютеров.

БЛАГОДАРНОСТИ

Я благодарен моему коллеге А. Д. Фалкоффу за предложения, способствовавшие значительному улучшению организации этой статьи, а также профессору Д. Макинтайру за советы, возникшие у него при чтении чернового варианта рукописи.

ПРИЛОЖЕНИЕ А
СВОДКА НОТАЦИИ

*F*ω	СКАЛЯРНЫЕ ФУНКЦИИ		*αF*ω
ω	Сопряженная	+ Плюс	
0—ω	Отрицательная	— Минус	
(ω>0)—ω<0	Знак	× Умножить	
1÷ω	Обратная	÷ Разделить	
⌈ —ω	Модуль	∣ Остаток	ω—α×ωω÷α+ +α=0
Целая часть	Низ	*P* Минимум	(ω×ω<α)+α× ×ω≥α
— — ω	Вверх	⌈ Максимум	—(—α)— — ω
2.71828... *ω	Экспоненциальная	* Степень	×/ωρα
Обратная для *	Натуральный логарифм	⍟ Логарифм	(⍟ω)÷⍟α
×/1+ιω	Факториал	! Биномиальная	(!ω)÷(!α)×!ω—α
3.14159 ...×ω	Умножить на Pi	○	

Булевы: ∨ ∧ ∼ (и, или, не-и, не-или, не)

Отношения: < ≤ = ≥ > ≠ (α*R*ω равняется 1, если верно отношение *R*)

	Sec. Ref.	*V*↔2 3 5 *M*↔1 2 3 / 4 5 6
Integers	1	ι5↔1 2 3 4 5
Shape	1	ρ*V*↔3 ρ*M*↔2 3 2 3ρι6↔*M* 2ρ4↔4 4
Catenation	1	*V*,*V*↔2 3 5 2 3 5 *M*,*M*↔1 2 3 1 2 3 / 4 5 6 4 5 6
Ravel	1	,*M*↔1 2 3 4 5 6
Indexing	1	*V*[3 1]↔5 2 *M*[2;2]↔5 *M*[2;]↔4 5 6
Compress	3	1 0 1/*V*↔2 5 0 1⌿*M*↔4 5 6
Take,Drop	1	2↑*V*↔2 3 ¯2↑*V*↔1↓*V*↔3 5
Reversal	1	⌽*V*↔5 3 2
Rotate	1	2⌽*V*↔5 2 3 ¯2⌽*V*↔3 5 2
Transpose	1, 4	⍉ω reverses axes α⍉ω permutes axes
Grade	3	⍋3 2 6 2↔2 4 1 3 ⍒3 2 6 2↔3 1 2 4
Base value	1	10⊥*V*↔235 *V*⊥*V*↔50
&inverse	1	10 10 10⊤235↔2 3 5 *V*⊤50↔2 3 5
Membership	3	*V*∊3↔0 1 0 *V*∊5 2↔1 0 1
Inverse	2, 5	⌹ω is matrix inverse α⌹ω↔(⌹ω)+.×α
Reduction	1	+/*V*↔10 +/*M*↔6 15 +⌿*M*↔5 7 9
Scan	1	+*V*↔2 5 10 +*M*↔2 3ρ1 3 6 4 9 15
Inner prod	1	+.× is matrix product
Outer prod	1	0 3∘.+1 2 3↔*M*
Axis	1	*F*[*I*] applies *F* along axis *I*

ПРИЛОЖЕНИЕ Б
КОМПИЛЯТОР ИЗ НЕПОСРЕДСТВЕННОЙ В КАНОНИЧЕСКУЮ ФОРМУ

Этот компилятор был заимствован из [22, с. 222]. Он не будет обрабатывать описания, включающие α,: или ω в кавычках. Он состоит из функций *FIX* и *F*9 и матриц символов *C*9 и *A*9:

```
FIX
0ρ⎕FX F9 ⍞

D←F9 E;F;I;K
F←(,(E='ω')∘.≠5↑1)/,E,(⌽4,ρE)ρ' Y9 '
F←(,(F='α')∘.≠5↑1)/,F,(⌽4,ρF)ρ' X9 '
F←1↓ρD←(0,+/¯6,I)↓(-(3×I)++\I←':'=F)⌽F,(⌽6,ρF)ρ' '
D←3⌽C9[1+(1+'α'∊E),I,0;],⍉D[;1,(I←2↓⍳F),2]
K←K+2×K<1⌽K←I∧K∊(>⌿1 0⌽'←⎕'∘.=E)/K←+\~I←E∊A9
F←(0,1+ρE)⌈ρD←D,(F,ρE)↑⍉0 ¯2↓K⌽' ',E,[1.5]';'
D←(F↑D),[1]F[2] '⍝',E
```

```
      C9
   Z9←
  Y9Z9←
  Y9Z9←X9
 )/3→(0=1↑,
    →0,0ρZ9←
```

```
    A9
012345678
9ABCDEFGH
IJKLMNOPQ
RSTUVWXYZ
ABCDEFGHI
JKLMNOPQR
STUVWXYZ⎕
```

Пример:

```
        FIX
FIB:Z,+/¯2↑Z←FIBω-1:ω=1:1

        FIB 15
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610

        ⎕CR'FIB'
Z9←FIB Y9;Z
→(0=1↑,Y9=1)/3
→0,0ρZ9←1
Z9←Z,+/¯2↑Z←FIB Y9-1
⍝FIB:Z,+/¯2↑Z←FIBω-1:ω=1:1
```

ЛИТЕРАТУРА

1. Boole G. An Investigation of the Laws of Thought, Dover Publication, N. Y., 1951. Первая публикация в 1954 г. Walton and Maberly, London and by MacMillan and Co. Cambridge. См. также: Collected Logical Works of

George Boole, Volume 2, Open Court Publishing Co., La Salle, Illinois, 1916.
2. Cajori F. A. A History of Mathematical Notations, Volume 2, Open Court Publishing Co., La Salle, Illinois, 1929.
3. Falkoff A. D. and Iverson K. E. The Evolution of APL, Porceedings of a Conference on the History of Programming Languages, ACM SIGPLAN, 1978.
4. APL Language, Form No. GC26—3847—4, IBM Corporation.
5. Iverson K. E. Elementary Analysis, APL Press, Pleasantville, N. Y., 1976.
6. Iverson K. E. Algebra: An Algorithmic Treatment, APL Press, Pleasantville, N. Y., 1972.
7. Kerner I. O. Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen, Numerische Mathematik, Vol. 8, 1966, pp. 290—294.
8. Beckenbach E. F., ed. Applied Combinatorial Mathematics., John Wiley and Sons, New York, N. Y., 1964.
9. Tarian R. E. Testing Flow Graph Reducibility, Journal of Computer and Systems Sciences, Vol. 9, No. 3, Dec. 1974.
10. Spence R. Resistive Circuit Theory, APL Press, Pleasantville, N. Y., 1972.
11. Iverson K. E. A Programming Language, John Wiley and Sons, New York, N. Y., 1962.
12. Iverson K. E. An Introduction to APL for Scientists and Engineers, APL Press, Pleasantville, N. Y.
13. Orth D. L. Calculus in a New Key, APL Press, Pleasantville, N. Y., 1976.
14. Apoctol T. M. Mathematical Analysis, Addison Wesley Publishing Co., Reading, Mass., 1957.
15. Jolley L. B. Summation of Series, Dover Publications, N. Y.
16. Oldham K. B. and Spanier J. The Fractional Calculus, Academic Press, N. Y., 1974.
17. APL Quote Quad, Vol. 9, No. 4, June 1979, ACM STAPL.
18. Iverson K. E. Operators and Functions, IBM Research Report RC 7091, 1978.
19. McDonnel E. E. A Notation for the GCD and LCM Functions, APL 75, Proceeding of an APL Conference, ACM, 1975.
20. Iverson K. E. Operators, ACM Transactions on Programming Languages and Systems, October 1979.
21. Blaauw G. A., Digital System Implementation, Prentice-Hall, Englewood Cliffs, N. J., 1976.
22. McIntyre D. B. The Architectural Elegance of Crystals Made Clear by APL, An APL Users Meeting, I. P. Sharp Associates, Toronto, Canada, 1978.

ПОСТСКРИПТУМ

НОТАЦИЯ КАК СРЕДСТВО МЫШЛЕНИЯ: 1986

К. Е. Айверсон

Тезис данной работы состоит в том, что обнаруживаемые в языках программирования исполнимость и универсальность могут быть эффективно объединены в одном согласованном языке с преимуществами, предоставляемыми математической нотацией.

В качестве исполнимого языка мы будем применять APL, язык общего назначения, который был создан с целью обеспечения ясного и точного выражения мыслей при написании и преподавании, который был реализован в виде языка программирования только после нескольких лет его разработки и развития.

Выше приведены две цитаты из моей статьи 1980 г. В первой из них указывается, в каком случае следует использовать исполнимую аналитическую нотацию, а во второй указано конкретное средство для ее разработки.

Наиболее очевидным и важным является применение исполнимой аналитической нотации в преподавании. В последующих комментариях подводятся итоги недавнего прогресса в этой области.

ПРЕДМЕТЫ И КУРСЫ

Общей темой упомянутых здесь предметов является неформальное введение необходимой нотации в контекст способом, знакомым по обучению математике. Хорошим примером на уровне высшей школы может служить подход к теории вероятности, использованный в [1]. В [2] язык APL широко используется как средство реализации системы для анализа схем, а в [3] графический ввод и выражения языка APL совместно применяются для описания проектов в экспертной системе.

Направление моей недавней работы описано в письме [4], и теперь доступны наброски двух текстов [5], используемых в курсах. Всем, кто интересуется этими вопросами, следует прочесть статью [6].

РАЗВИТИЕ НОТАЦИИ

Недавно была разработана версия языка APL [7], которая, оставаясь в рамках стандарта ISO для этого языка, упростила его структуру и увеличила его выразительную силу. Она обеспечивает значительно лучшую основу для преподавания, чем нотация, использованная в моей статье 1980 г.

ДОСТУПНОСТЬ РЕАЛИЗАЦИЙ

Хотя APL в течение долгого времени предлагался централизованными вычислительными службами университетов, его применение в преподавании оказывалось непрактичным из-за высоких затрат и отсутствия подходящих терминалов. В наше время доступность систем APL на микрокомпьютерах резко изменила эту ситуацию.

Я считаю наиболее удовлетворительной систему, предоставляемую нашим студентам [8]; **в ней еще** не воплощены такие новые функции, **как сущность, стереть** и **все** (обобщение декартова произведения), но предусмотрены фундаментальные понятия **ранга функции**, блочной функции (для общей обработки представлений или «структур») и операция **под** для важного математического понятия двойственности.

Кроме того, система оперирует комплексными числами (со всеми математическими функциями, подходящим образом обобщенными), обеспечивает детерминант (—.×), перманент (+.×), проверку для латинского квадрата (∨.∧) и связанные с ними функции, порождаемые операцией «точка», обобщения функций **и** и **или** для получения наибольшего общего делителя и наименьшего общего кратного; используются характеристики микрокомпьютера и экрана его монитора для обеспечения «единой» клавиатуры, в которой большинство символов (в частности, скобки и прописные и строчные буквы, употребляемые в именах) находятся в своих обычных позициях, принятых на пишущих машинках.

ЛИТЕРАТУРА

1. Alvord L. Probability in APL. APL press, STSC Corp., Bethesda Md.
2. Spence R. and Burgess J. Circuit Analysis. Prentice-Hall, Englewood Cliffs, N. J., 1986.
3. Hasony Y. Краткий отчет о его работе появился в обзоре конференции Minnowbrook, представленной в APL Quote-Quad 16, 3 (1986).

4. Blaauw G. A. et al. A curriculum proposal for computer science. Commun. ACM, Forum (Sept. 1985).
5. Iverson K. E. Mathematics and Programming; Applied Mathematics for Programmers. Обе работы можно получить из I. P. Sharp Associates, Toronto, Ont., Canada.
6. Pesch R. and Berry M. J. A Style and Literacy in APL. In Proceedings of APL86, ACM, New York, 1986.
7. Iverson K. E. A Dictionary of the APL Language. Можно получить из I. P. Sharp Associates, Toronto, Ont., Canada.
8. Sharp APL/PCX. Computer system for use on IBM AT/370 and XI/370 computers. Можно получить из I. S. Sharp Associates, Toronto, Ont., Canada. Система работает также на обычных IBM PC или AT (значительно более медленно, но приемлемо для учебных целей).