1969

Форма и содержание в информатике

М. Минский

Массачусетский технологический институт
Кембридж, Массачусетс

Чрезмерное увлечение формализмом препятствует развитию информатики. Путаница между формой и содержанием обсуждается применительно к трем областям: теории вычислений, языкам программирования и обучению.

Проблема современной информатики состоит в навязчивой озабоченности формой в ущерб содержанию.

Нет, начать нужно не с этого. По любым прежним стандартам жизненная сила информатики огромна. Какая другая сфера интеллектуальной деятельности когда-либо достигала такого расцвета за двадцать лет? К тому же теория вычислений, по-видимому, некоторым образом включает в себя учение о форме, так что эта озабоченность не так уж неправомерна. Тем не менее я утверждаю, что чрезмерное увлечение формализмом тормозит развитие нашей науки.

Прежде чем должным образом углубиться в обсуждение темы, я хочу выразить признательность моим коллегам и студентам, которые способствовали тому, что я удостоился Тьюринговской премии. Комплекс вопросов, некогда философских, а ныне научных, сопряженных с понятием интеллекта, вызывал чрезвычайный интерес у Алана Тьюринга, и он наряду с немногими другими мыслителями — особенно Уорреном Маккалоком и его молодым сотрудником Уолтером Питтсом — был автором многих ранних исследований, приведших к созданию как самих компьютеров, так и новой технологии, а именно искусственного интеллекта. При оценке этой проблематики факт присуждения данной премии должен привлечь внимание к другим работам членов моего научного коллектива, в первую очередь Р. Соломоноффа, О. Селфриджа, Дж. Маккарти, А. Ньюэлла, Х. Саймона и С. Пейперта, моих ближайших сотрудников по десятилетним исследованиям. В этом эссе нашли отражение взгляды Пейперта.

Данное эссе состоит из трех частей, посвященных недопониманию взаимоотношений между формой и содержанием в *теории вычислений, языках программирования* и в обучении.

1. ТЕОРИЯ ВЫЧИСЛЕНИЙ

Чтобы построить теорию, нужно кое-что знать об основных явлениях предметной области. Применительно к теории вычислений нам просто недостает таких знаний, чтобы излагать этот предмет достаточно абстрактно. Вместо этого мы вынуждены подробнее разбирать конкретные примеры, которые понимаем более основательно, и надеяться, что это поможет нам формулировать и обосновывать более общие принципы. Я говорю это не для того, чтобы с ходу принимать как данность многие положения, вероятно истинные, но еще не доказанные. Я полагаю, что многие наши предположения, основанные на так называемом здравом смысле, в действительности ложны. У нас бывают ложные представления о возможных альтернативных вариантах затрат времени и памяти; о компромиссах между временем работы и сложностью программ, между аппаратным и программным обеспечением, между цифровыми и аналоговыми схемами, последовательными и параллельными вычислениями, ассоциативной и адресуемой памятью и о многом другом.

Целесообразно рассмотреть аналогию с физикой, где можно организовать много базовых знаний как набор довольно компактных законов сохранения. Конечно, это всего лишь один из способов описания; можно было бы использовать дифференциальные уравнения, принципы минимума, законы равновесия и т. д. Так, сохранение энергии можно интерпретировать как определение переходов между различными формами потенциальной и кинетической энергии, например между высотой и квадратом скорости или между температурой и давлением — объемом. Можно основывать разработку квантовой теории на балансе между степенью определенности координат и временем или же времени и энергии. В этом нет ничего необычного; всякое уравнение с достаточно гладкими решениями может рассматриваться как некое определение обменов (преобразований) между его переменными. Однако существует много способов формулирования одних и тех же соотношений, и рискованно отдавать чрезмерное предпочтение одной конкретной форме или закономерности и приходить к убеждению, что это *истинный* фундаментальный принцип. (См. на эту тему диссертацию Фейнмана [1].)

Тем не менее распознавание подобных преобразований (об-

менов, компромиссов) часто является зачатием науки, тогда как их количественное выражение — ее рождением. Что в этом смысле мы наблюдаем в области вычислений? В теории рекурсивных функций мы располагаем наблюдением Шеннона [2], заключающимся в том, что любая машина Тьюринга с Q состояниями и R символами эквивалентна машине Тьюринга с двумя состояниями и nQR символами, а также машине Тьюринга с 2 символами и $n'QR$ состояниями, где n и n' — небольшие числа. Таким образом, произведение числа состояний на число символов, QR, играет роль почти инварианта при классификации машин. К сожалению, это произведение не удается использовать в качестве полезной меры сложности машины, потому что оно, в свою очередь, является плодом компромисса со сложностью процесса кодирования для машин Тьюринга, а этот компромисс представляется слишком непонятным для того, чтобы его можно было с пользой применить.

Рассмотрим более элементарный, но все же озадачивающий компромисс между сложением и умножением. Сколько умножений требуется для вычисления детерминанта 3×3? Если мы выпишем полную форму как шесть произведений по три, то понадобится двенадцать перемножений. Если мы соберем множители, воспользовавшись распределительным законом, то число умножений сократится до девяти. Каково минимальное количество, и как доказать это в данном случае и в случае $n\times n$? Важно не то, что мы нуждаемся в ответе. Суть в том, что мы не знаем, как узнать или доказать, что предлагаемые ответы являются правильными! Для одной формулы, возможно, удалось бы организовать некоторый вид исчерпывающего поиска, но на основании этого нельзя было бы установить общее правило. Одной из главных целей исследования должна быть разработка методов доказательства того, что данные конкретные процедуры в тех или иных смыслах являются вычислительно минимальными.

В диссертации Кука [3], в которой используется результат Тоома, сделано удивительное открытие, касающееся умножения; оно обсуждается в публикации Кнута [4]. Рассмотрим обычный алгоритм умножения десятичных чисел: для двух чисел, каждое из которых имеет n знаков, требуется n^2 произведений однозначных чисел. Обычно полагают, что это минимальное количество. Но предположим, что мы разбиваем числа на две части, так что произведение принимает вид $N=(@A+{}+B)\ (@C+D)$, где @ означает умножение на $10^{n/2}$. (Мы пренебрегаем затратами на операцию сдвига влево.) Тогда легко убедиться, что

$$N=@\ @AC+BD+@\,(A+B)\,(C+D)-@\,(AC+BD).$$

Здесь задействованы только *три* умножения половинной длины вместо четырех, которые, казалось бы, нужны. При большом значении n такое сведение может очевидным образом применяться снова и снова для меньших чисел. За это приходится платить увеличением количества сложений. Объединив эту и другие идеи, Кук показал, что при любом ε и достаточно большом n для умножения требуется меньше чем $n^{1+\varepsilon}$ произведений вместо ожидавшихся n^2. Аналогичным образом В. Штрассен недавно показал, что для умножения двух *матриц* $m \times m$ *число умножений можно свести до* $O(m^{\log_2 7})$, хотя всегда было принято считать, что это число должно быть кубическим полиномом от m, потому что результат содержит m^2 членов и для каждого из них вроде бы требуется отдельное внутреннее произведение с m перемножениями. В обоих случаях обычная интуиция длительное время вводила математиков в заблуждение, настолько пагубное, что, по-видимому, никто и не искал лучших методов. Мы все еще не располагаем набором проверенных методов, которые точно устанавливали бы, какова минимальная цена замены умножений сложениями в случае матриц.

Замена умножений сложениями сама по себе может не казаться жизненно важной, но если мы не в состоянии досконально понять нечто столь простое, то нам тем более следует ожидать серьезных затруднений с чем-то более сложным.

Рассмотрим другой компромисс, между размером памяти и временем вычислений. В книге [5] Пейперт и я поставили простую проблему: если имеется произвольный набор слов по n бит, то сколько обращений к памяти потребуется, чтобы ответить, какое из этих слов является ближайшим (по числу совпадающих бит) к произвольному заданному слову? (Для идентификации *точного* отождествления можно использовать хеширование, и эта проблема достаточно хорошо изучена.) Поскольку существует много способов кодирования «библиотечного» набора слов, причем в одних из них используется больше памяти, чем в других, то более точно проблема формулируется следующим образом: насколько должен возрасти объем памяти для достижения данного сокращения числа обращений к памяти? В определенном смысле ответ тривиален: если память достаточно велика, то требуется только одно обращение, потому что мы можем использовать в качестве адреса сам вопрос и запомнить ответ в регистре с таким адресом. Но если память как раз достаточна для хранения информации в библиотеке, то нужно искать все слова из библиотеки, — *и нам не известно сколько-нибудь значительного промежуточного результата.* Разумеется, это фундаментальная теоретическая проблема восстановления информации, однако, по-видимому, ни у кого нет

каких-либо интересных идей относительно того, как установить хорошую нижнюю границу для этого основного компромисса[1]).

Другая проблема связана с последовательно-параллельным обменом. Предположим, у нас имеется n компьютеров вместо всего лишь одного. Насколько мы можем ускорить те или иные виды вычислений? Для некоторых мы можем уверенно получить выигрыш в n раз. Однако такие вычисления встречаются редко. Для других этот коэффициент составит $\log n$, но такие вычисления трудно обнаружить или доказать, какими свойствами они должны обладать. А для большинства вычислений, по моему мнению, нам трудно выиграть хоть что-нибудь; это тот случай, когда имеется множество весьма разветвленных условий, так что попытки предугадывать возможные будущие ответвления обычно оказываются тщетными. Об этом мы не знаем почти ничего. Большинство людей полагают с совершенно неоправданным оптимизмом, что параллелизм, как правило, благоприятствует ускорению большинства вычислений.

Итак, существует несколько плохо понимаемых проблем относительно вычислительных компромиссов. Здесь нет места для обсуждения остальных, например, аналогово-цифровой проблемы. (Некоторые проблемы соотношения локальных и глобальных вычислений намечены в [5].) И мы очень мало знаем о сопоставимости численных и символьных вычислений.

В современных учебных планах по информатике уделяется слишком мало внимания тому, что известно о таких проблемах. Почти все время отводится формальной классификации синтаксических типов языков, пораженческим теориям неразрешимости, фольклору о системном программировании и преимущественно тривиальным фрагментам «оптимизации логического проектирования» — последнее часто в ситуациях, когда практическое искусство эвристического программирования далеко превзошло «теории» для частных случаев, столь громогласно провозглашаемые и неустанно проверяемые, — а также заклинаниям относительно стиля программирования, которые почти наверняка выйдут из моды к тому времени, когда студент завершит свое образование. Даже, казалось бы, наиболее абстрактные курсы по теории рекурсивных функций и формальной логике, по-видимому, игнорируют немногие известные полезные

---

[1]) В 1964 г. в Ж. вычисл. матем. и матем. физ., т. 4, с. 536—543, В. В. Мартынюк опубликовал статью «Экономная организация поиска и занесения информации при избыточной памяти», в которой предложил алгоритм организации библиотек из k слов, при котором поиск любого слова осуществляется путем i обращений к памяти. Этот алгоритм применим при всех значениях $i=1, 2, 3, \ldots, k$. При любом значении i для хранения библиотеки требуется примерно $ik\sqrt[i]{m/k}$ слов (ячеек), где m — число всех адресуемых ячеек, например $m=2^n$. — *Прим. перев.*

результаты по доказательству свойств компиляторов или эквивалентности программ. В большинстве курсов результаты работы по искусственному интеллекту, некоторым из которых теперь уже исполнилось пятнадцать лет, толкуются как второстепенный набор специальных приложений, хотя на самом деле они представляют одну из важнейших сторон эмпирического и теоретического исследования реальных вычислительных проблем. До тех пор пока все эти предпочтения формы не уступят место вниманию к существенным аспектам вычислений, юному студенту вполне можно посоветовать не обременять себя курсами информатики, учиться программировать, усваивать как можно больше математику и другие фундаментальные научные дисциплины, а также изучать современную литературу по искусственному интеллекту, теориям сложности и оптимизации.

2. ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Даже в области языков программирования и компиляторов слишком много внимания уделяется форме. Я говорю «даже», потому что можно понять, что это именно та область, в которой форма *заслуживает* преимущественного внимания. Но рассмотрим два утверждения: (1) языки оказываются такими, что в них слишком много синтаксиса, и (2) языки описываются с излишними синтаксическими подробностями.

В компиляторах не уделяется должного внимания значению выражений, высказываний и описаний. Применение контекстно-свободных грамматик для описания фрагментов языков ведет к важным достижениям в единообразии как спецификации, так и реализации. Однако, хотя это вполне оправдывается в простых случаях, попытки распространить этот подход на более сложные области могут затормозить научный прогресс. Возникают серьезные проблемы при использовании грамматик для описания самоизменяющихся или саморасширяющихся языков с применением процессов исполнения наряду со спецификациями. Невозможно описать синтаксически — иначе говоря, статически — допустимые выражения языка, который изменяется. Конечно, должны быть описаны механизмы расширения синтаксиса, однако если они задаются в терминах такого современного языка сопоставления с образцом, как Снобол, Convert [6] или Matchless [7], то не должно быть различия между программой грамматического разбора и описанием собственно языка. Будущие языки программирования станут в большей степени сосредоточиваться на целях и в меньшей степени на процедурах, специфицированных программистом. Дальнейшие суждения являются несколько максималистскими,

однако ввиду нынешнего чрезмерного увлечения формой в этой полемической заостренности не будет вреда. (Некоторые мысли заимствованы у К. Хевитта и Т. Винограда.)

2.1. СИНТАКСИС ЧАСТО НЕОБЯЗАТЕЛЕН

Можно обойтись гораздо меньшим объемом синтаксиса, чем обычно считается. Большая часть синтаксиса программирования относится к подавлению скобок или к выражению меток контекста. Существуют варианты, которые используются слишком редко.

Пожалуйста, не думайте, что я выступаю против применения для общения с человеком таких средств, как запись знака бинарной операции между операндами или старшинство операций. Они играют свою роль. Однако их важность для информатики в целом столь явно преувеличивалась, что уже начала вредить воспитанию программистской молодежи.

Рассмотрим общеизвестный алгоритм извлечения квадратного корня в том виде, как его можно записать на современном алгебраическом языке, пренебрегая такими аспектами, как описания типов данных. Требуется значение квадратного корня из A при заданном начальном приближении X и предельно допустимой погрешности E.

```
ОПИСАНИЕ SQRT (A,X,E):
    если ABS (A—X*X)<E, то X иначе SQRT (A, (X+A÷X)÷
    ÷2,E).
```

На воображаемой, но узнаваемой версии Лиспа (см. Левин [8] или Вайсман [9]) ту же процедуру можно записать так:

```
(DEFINE (SQRT A X E)
    (IF(LESS(ABS(—A(*X X)))E)THEN X
    ELSE(SQRT(÷(+X(÷A X))2)E)))
```

Здесь имена функций находятся непосредственно *внутри* соответствующих скобок. Очевидно, что для людей неудобно выписывать все скобки; однако часто недооценивают, насколько удобнее обходиться без необходимости заучивать все соглашения, например, что (X+A÷X) равно (+X(÷AX)) и не равно (÷(+X A) X).

Еще предстоит увидеть, является ли синтаксис с явными разделителями косностью или же представляет собой росток будущего. Он обладает значительными преимуществами для редактирования, интерпретации и для *создания программ другими программами.* Полный синтаксис Лиспа можно выучить примерно за час; интерпретатор компактен и не слишком

усложнен, и студенты часто могут отвечать на вопросы о системе, читая саму интерпретирующую программу. Разумеется, это не даст ответов на все вопросы относительно реальной практической реализации, но их не даст и никакой обозримый набор синтаксических правил. Кроме того, несмотря на громоздкость этого языка, многие работающие на переднем крае науки исследователи считают, что он обладает выдающейся выразительной силой. Почти все исследования процедур решения проблем путем построения и модификации гипотез были написаны с применением этого или сходных языков. К сожалению, разработчики языков, как правило, незнакомы с этой областью и склонны отмахиваться от нее как от специального раздела «методов манипуляции символами».

Можно сделать многое для прояснения структуры выражений в таком «синтаксически слабом» языке, воспользовавшись записью текста с различными отступами и другими графическими средствами, которые выходят за рамки собственно языка. Например, можно употребить символ «отсрочки», относящийся к входному препроцессору, чтобы переписать предыдущее выражение как

```
DEFINE (SQRT A X E)⇓.
   IF↓THEN X ELSE⇓.
    LESS (ABS⇓)E.
                 —A(*XX).
    SQRT A⇓E.
           ÷—⇓2.
           +X(÷—AX)
```

где точка означает «)(», а стрелка означает «вставить здесь следующее выражение, ограниченное точкой и становящееся доступным после замены (рекурсивно) его собственных стрелок». Отступы необязательны. Значительная часть эффекта обычных меток области действия обеспечивается здесь двумя простыми средствами, тривиально реализуемыми программами чтения, и такой текст легко поддается редактированию, потому что подвыражение обычно завершается в каждой строке.

Чтобы ощутить силу и ограниченность операции отсрочки, читателю следует обратиться к своему излюбленному языку и к своим излюбленным алгоритмам и посмотреть, что произойдет. Он обнаружит там много применений для отсрочки и поупражняется в рассуждениях о том, что сказать вначале, какие аргументы выделить и так далее. Разумеется, знак ⇓ не решает всех проблем; техника отсрочки требуется также для фрагментов списков, и для этого нужен свой собственный раз-

делитель. В любом случае это всего лишь шаги в направлении более графически выразительных систем описания программ, так как мы не обязаны навсегда ограничиваться только строками символов.

Д. Скотт предложил другое объяснительное средство, состоящее в использовании других скобок для указания функциональной композиции справа налево, так что можно писать $\langle\langle\langle x\rangle h\rangle g\rangle f$ вместо $f(g(h(x)))$, если есть желание показать более естественно, что происходит с величиной в процессе вычисления. Благодаря этому возникает возможность для различных «акцентов», например, $f(\langle h(x)\rangle g)$ можно прочесть так: «вычислите f от того, что вы получили после того, как сначала вычислили $h(x)$, а затем применили g к результату».

Пожалуй, это проще объяснить по аналогии, чем на примере. Со своей фанатической приверженностью синтаксису проектировщики языков стали слишком односторонне ориентированы на фразу. Пользуясь такими средствами, как знак $\Downarrow$, можно конструировать объекты, более напоминающие абзацы, не возвращаясь полностью к старинным блок-схемам.

Современные языки высокого уровня предлагают слишком мало выразительных возможностей в отношении гибкости стиля. Пользователь не может существенно варьировать последовательность представления мыслей, не меняя самого алгоритма.

2.2. ЭФФЕКТИВНОСТЬ И ПОНИМАНИЕ ПРОГРАММ

Для чего предназначается компилятор? Обычно отвечают примерно так: «для трансляции с одного языка на другой» или «чтобы, имея описание алгоритма, превратить его в программу с заполнением многих мелких подробностей». На будущее требуется более честолюбивый подход. Большинство компиляторов станут системами, которые «порождают алгоритм по заданному описанию того, что он должен делать». Таковы уж современные системы формирования изображений; они выполняют всю творческую работу, тогда как пользователь только предоставляет образцы желаемых форматов: в этом компиляторы более сведущи, чем пользователи. Хорошими примерами являются также языки сопоставления с образцами. Однако, за исключением немногих частных случаев, проектировщики компиляторов мало преуспели в обеспечении написания хороших программ. Распознавание общих подвыражений, оптимизация внутренних циклов, распределение нескольких регистров и прочее — все это приводит лишь к небольшим линейным улучшениям эффективности, причем даже в этом отношении компиляторы делают очень немного. Можно было бы в зна-

чительно большей степени автоматизировать распределение памяти. Но настоящий выигрыш заложен в анализе *вычислительного содержания* самого алгоритма, а не способа записи этого алгоритма программистом. Например, рассмотрим:

ОПИСАНИЕ (FIB(N): если $N=1$ то 1, если $N=2$ то 1,
иначе FIB(N—1)+FIB(N—2)

Это рекурсивное описание чисел Фибоначчи 1, 1, 2, 3, 5, 8, 13... может быть задано на любом приличном языке программирования и приведет к разветвляющемуся дереву шагов вычисления, показанному на рис. 1.

Легко видеть, что количество машинной работы будет возрастать экспоненциально с увеличением N. (Более точно, оно проходит через примерно FIB (N) вычислений согласно описанию.) Существуют лучшие способы вычисления этой функции. Так, мы можем описать два временных регистра и вычислить FIB (N1 1) по формуле

ОПИСАНИЕ FIB(NAB): если $N=1$ то A иначе FIB(N—1 A+ +BA)

которая является однократно рекурсивной и избавляет от разветвляющегося дерева, или даже можем использовать

$A=0$
$B=1$

ЦИКЛ ЗАГРУЗИТЬ AB
если $N=1$ вернуть A
$N=N$—1
$B=A+B$
перейти в ЦИКЛ

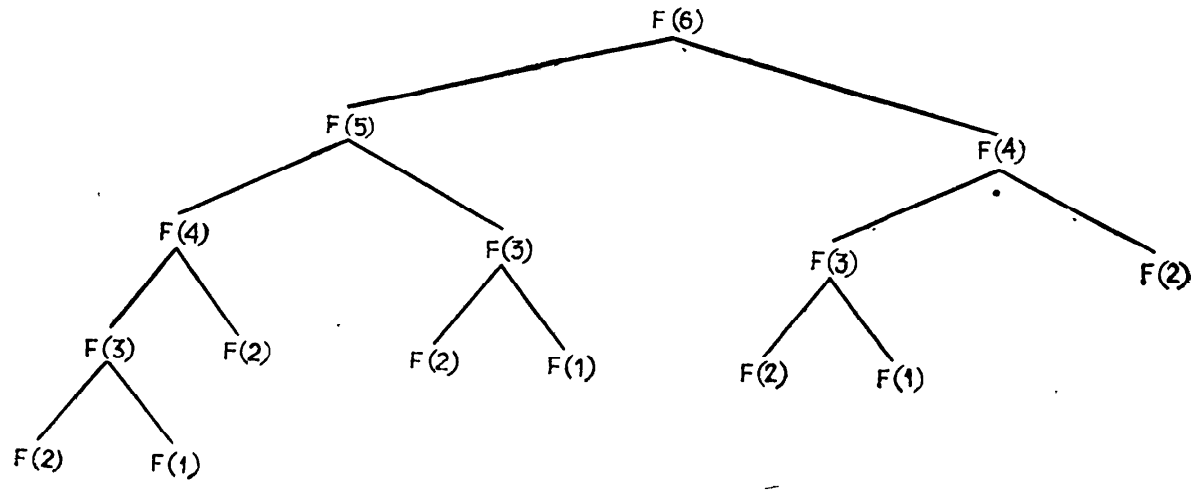

Рис. 1.

Любому программисту быстро придут в голову эти возможности, как только он увидит, что происходит при разветвленном вычислении. Это тот случай, когда «управляемая значениями» рекурсия может быть преобразована в простое повторение. Современные компиляторы не распознают даже простейшие случаи таких преобразований, хотя избавление от экспоненциального роста перевешивает любой возможный выигрыш от локальной оптимизации кода. Нет смысла возражать, что такие выигрыши редки или что подобные преобразования относятся к компетенции программиста. Если важно сэкономить время компиляции, то не следует пренебрегать такими возможностями. Например, для программ, написанных на языках сопоставления с образцом, подобные упрощения действительно часто производятся. Обычно компиляция эффективного дерева грамматического разбора для системы БНФ дает явный выигрыш по сравнению с выполнением примитивного переборного анализа посредством синтеза.

Вне всяких сомнений, систематическая теория таких преобразований трудна. Система должна быть весьма сообразительной для того, чтобы обнаруживать, какие преобразования уместны и когда их применение окупается. Поскольку программист всегда знает свою цель, проблема часто упрощается, если предлагаемый алгоритм сопровождается (или даже заменяется) подходящей декларацией цели работы алгоритма.

Для того чтобы продвигаться в этом направлении, нам нужно знать основы анализа и синтеза программ. В области теории сейчас многое делается для изучения эквивалентности алгоритмов и схем и для доказательства того, что процедуры обладают объявленными свойствами. С практической стороны работы В. Мартина [10] и Дж. Мозеса [11] показывают, как строить системы, оснащенные достаточными знаниями о символьных преобразованиях конкретных математических методов для существенной поддержки прикладных математических возможностей пользователей.

Практически ничего не следует из того факта, что проблема сведения программ в общем случае является рекурсивно неразрешимой. Во всяком случае, можно ожидать появления программ, в конечном счете далеко превосходящих человеческие способности к этой деятельности, и можно воспользоваться большим разнообразием преобразований программ в формализованном и отлаженном виде. Непосредственное применение таких преобразований окажется затруднительным. Вместо этого можно ожидать разработок в духе методов, употреблявшихся в символьном интегрировании, например в работах Слэгла [12] и Мозеса [11]. Сначала был разработан набор простых формальных преобразований, соответствующих

элементарным формулам табличных интегралов. На основании этого Слэгл построил ряд эвристических приемов алгебраического и аналитического преобразования практической задачи в эти уже понятые элементы; при этом привлекались характеристики процедуры сопоставления, которые можно назвать «распознаванием образов». В системе Мозеса процедуры сопоставления и преобразования были столь хорошо отработаны, что в большинстве практических задач стратегия эвристического поиска, игравшая важную роль в эффективности программы Слэгла, становилась малосущественным приращением к точному знанию и его искусному приложению, воплощенному в системе Мозеса. Эвристическая система компиляции в конечном счете потребует гораздо больше *общего знания* и здравого смысла, чем системы символьного интегрирования, поскольку цели такой системы ближе к моделированию всей деятельности математика, а не специалиста по интегрированию.

2.3. ОПИСАНИЕ СИСТЕМ ПРОГРАММИРОВАНИЯ

Вне зависимости от того, как описывается язык, компьютер должен располагать процедурой его интерпретации. Следует помнить, что *при описании языка главная цель состоит в том, чтобы объяснить, как писать на нем программы и что такие программы означают*. Основная цель *вовсе не состоит* в описании синтаксиса.

В рамках статичной структуры синтаксических правил, нормальных форм, правил подстановок Поста и других подобных схем мы получаем эквиваленты логических систем с аксиомами, правилами вывода и теоремами. Проектирование недвусмысленного синтаксиса соответствует тогда проектированию математической системы, в которой каждая теорема имеет ровно одно доказательство! Но в вычислительном контексте это совершенно не важно. Здесь имеется дополнительный аспект — управление, — который выходит за рамки обычной структуры логической системы. Речь идет о дополнительном наборе правил, которые указывают, когда следует применять то или иное правило вывода. Итак, для многих целей недвусмысленность является надуманной проблемой. Если мы рассматриваем программу как процесс, то можем вспомнить, что наиболее мощными средствами описания процессов являются сами программы, а они по самой своей природе недвусмысленны.

Описание языка программирования через программу не является парадоксом. Разумеется, процедурное описание должно быть понято. Можно достичь этого понимания за счет описания на другом языке, который может быть иным, более знакомым или более простым, чем тот, который описывается. Но

часто бывает, что практичнее, удобнее и правильнее использовать тот же самый язык! Чтобы понять такое описание, нужно знать только работу данной конкретной программы, а не все следствия из всех возможных применений языка. Именно эта конкретизация создает возможность для раскрутки, т. е. способа разработки программного обеспечения, при котором сначала разрабатывается ограниченное подмножество языка, используемое для реализации более обширных его вариантов. Это обстоятельство часто озадачивает начинающих, как, впрочем, и кажущихся авторитетными специалистов.

Применение БНФ для описания формирования выражений может задержать развитие новых языков, которые естественным образом включают цитирование, самомодификацию и манипуляции символами в традиционную алгоритмическую схему. А это в свою очередь тормозит прогресс в направлении систем программирования, ориентированных на целевое решение проблем. Как ни парадоксально, хотя идеи современного программирования разрабатывались из-за трудности изображения процессов в классической математической нотации, тем не менее проектировщики обращаются назад, к ранней форме — уравнению — именно в тех ситуациях, где *нужна* программа. В разд. 3, посвященном обучению, наблюдается сходная ситуация в преподавании, возможно, с еще более серьезными последствиями.

3. ОБУЧЕНИЕ, ПРЕПОДАВАНИЕ И «НОВАЯ МАТЕМАТИКА»

Образование — это еще одна область, в которой специалист по информатике путает форму и содержание, но в настоящее время эта путаница неотделима от его профессиональной роли. Он воспринимает свою главную функцию как обеспечение программ и машин для использования в старых и новых методиках обучения. Само по себе это неплохо, но я полагаю, что на нем лежит ответственность за более сложную задачу — вырабатывать и распространять модели самого процесса обучения.

Ниже я кратко обрисую отправную для этого мнения точку зрения (разработанную С. Пейпертом). Следующие утверждения типичны для нашего подхода:

— Помочь людям учиться — это значит помочь им строить в своем сознании различные виды вычислительных моделей.

— Лучше всего с этим справится тот учитель, который имеет в своем сознании разумную модель того, что собой представляет сознание ученика.

— По той же причине студент при отладке своих собственных моделей и процедур должен иметь модель того, что он

делает, и должен знать хорошие приемы отладки, например как формулировать простые, но решающие тестовые примеры.

— Это поможет студенту что-нибудь узнать о вычислительных моделях и программировании. Например, сама идея отладки[1] является весьма мощной — в отличие от беспомощности, навязываемой традиционными представлениями о дарованиях, талантах и склонностях. Эти представления навязывают человеку вывод: «Я не гожусь для этого», вместо того чтобы поощрить его спросить себя: «Как мне усовершенствоваться в этом?»

Эти утверждения звучат вполне разумно, однако они не относятся к числу основных принципов любой из популярных схем обучения, таких, как «оперативное закрепление», «методы раскрытия», «аудиовизуальный синергизм» и др. Дело не в том, что педагоги игнорировали возможность моделей мышления, а в том, что у них просто не было эффективных способов описывать, строить и проверять такие идеи, пока не началась работа по имитации (компьютерному моделированию) процессов мышления.

Мы не можем снисходить здесь до дискуссий со скептиками, которые считают чрезмерным упрощенчеством (если не нечестивостью или непристойностью) сравнение человеческих умов с программами. Можно отослать многочисленных таких критиков к работе Тьюринга [13]. Тем, кто полагает, что ответ не может храниться в цифровых или каких-либо других машинах, можно возразить [14], что машины, обретя интеллект, вполне возможно, станут полагать точно так же. Некоторые обзоры этой области содержатся в работах Фейгенбаума и Фелдмана [15] и Минского [16]. В этой быстро развивающейся отрасли можно не отстать от жизни, только читая современные диссертации и труды конференций по искусственному интеллекту.

Имеется существенный прагматический довод в пользу наших предложений. Ребенку нужны модели; чтобы понять город, он может использовать модель организма: тот тоже должен есть, дышать, выделять, защищаться и т. д. Не самая удачная модель, но достаточно полезная. Обмен веществ в реальном организме он может понять, в свою очередь сравнив его с машиной. Но для моделирования своей собственной сущности он *не может* воспользоваться ни машиной, ни организмом, ни городом, ни телефонным коммутатором; ему для этого не пригодится ничего, кроме компьютера с его программами и их ошибками. В конце концов в начальном обучении само про-

[1] Тьюринг весьма преуспел в отладке аппаратуры. Он проверял ее, не отключая тока, чтобы не потерять «ощущение» предмета. Теперь так делают все, но это не то же самое в наши дни, когда все схемы работают на трех или пяти вольтах.

граммирование станет играть даже большую роль, чем математика. Тем не менее я выбрал именно *математику* в качестве предмета обсуждения в оставшейся части этой работы отчасти потому, что мы понимаем ее лучше, но главным образом из-за того, что существующее предубеждение против программирования как академической науки привело бы к слишком сильному сопротивлению со стороны аудитории. Я полагаю, что сошел бы и какой-нибудь другой предмет, но выводы и понятия математики являются самыми точными и в наименьшей степени подвержены запутывающему влиянию эмоций.

3.1. МАТЕМАТИЧЕСКИЙ ПОРТРЕТ РЕБЕНКА

Представьте себе ребенка в возрасте пяти-шести лет, собирающегося поступать в первый класс. Если мы проэкстраполируем тенденции наших дней, его математическое образование будет проводиться слабо подготовленными учителями и отчасти плохо запрограммированными компьютерами; ни те, ни другие не будут способны сделать намного больше, чем отвечать «правильно» и «ты ошибся»; я уже не говорю, что они будут совершенно не в состоянии разумно интерпретировать то, что ребенок делает или говорит, потому что не будут обладать хорошими моделями ребенка или хорошими теориями детского умственного развития. Ребенок начинает с простой арифметики, теории множеств и начатков геометрии; через десять лет он будет знать немного о формальной теории вещественных чисел, немного о линейных уравнениях, чуть больше о геометрии и почти ничего о непрерывных функциях или предельных переходах. Он станет подростком без особых склонностей к аналитическому мышлению, неспособным применить свой десятилетний опыт к пониманию нового для него мира.

Посмотрим более пристально на нашего ребенка, воспользовавшись составной картиной, извлеченной из работ Пиаже и других наблюдателей умственного развития детей.

Наш ребенок сумеет произнести «один, два, три...» по крайней мере до тридцати, а возможно, и до тысячи. Он будет знать названия некоторых больших чисел, но не сможет увидеть, например, почему десять тысяч равняется сотне сотен. У него будут возникать серьезные затруднения с обратным счетом, если только он недавно не заинтересовался этим. (Хорошее владение обратным счетом облегчит простое вычитание, и имеет смысл несколько попрактиковаться.) У него не особенно развито чувство четности и нечетности.

Он может совершенно надежно сосчитать четыре — шесть предметов, но у него не каждый раз будет сходиться счет пятнадцати рассыпанных предметов. Он будет раздражен этим,

потому что твердо уверен, что должен получать каждый раз одинаковый результат. Поэтому наблюдатель заключит, что ребенок хорошо представляет себе понятие числа, но не столь искусен в его применении.

Впрочем, существенные аспекты его понимания чисел вовсе не будут признаны надежными с точки зрения стандартов, принятых взрослыми. Например, когда предметы переупорядочены перед его глазами, его впечатление об их количестве будет формироваться под влиянием их геометрического размещения. Например, ребенок скажет, что имеется меньше букв x, чем букв y, в следующих строчках:

$$x\,x\,x\,x\,x\,x\,x$$
$$y\,y\,y\,y\,y\,y\,y$$

а когда мы раздвинем буквы x и получим

$$x\;x\;x\;x\;x\;x\;x$$
$$y\,y\,y\,y\,y\,y\,y$$

он скажет, что букв x больше, чем букв y. Несомненно, что он при этом отвечает на другой вопрос — о размере; однако это реальный факт: постоянство (сохраняемость) количества в таких ситуациях в малой степени доходит до него. Он не в состоянии эффективно воспользоваться этим фактом в своих рассуждениях, хотя, если его спросить, он продемонстрирует знание того, что количество вещей не может измениться просто из-за их переупорядочивания. Аналогично, когда вода переливается из одного стакана в другой (рис. 2, *а*), ребенок скажет, что воды в высоком и узком стакане больше, чем в низком и широком. Он плохо оценивает двумерные площади, и поэтому нам не удастся найти такой контекст, в котором ребенок рассматривал бы большую площадь на рис. 2, *б* как четырехкратное увеличение меньшей площади. Кстати, когда он уже взрослый, то, получив два сосуда, один из которых в два раза превосходит другой по всем измерениям (рис. 2, *в*), он будет думать, что один вмещает примерно в четыре раза больше жидкости, чем другой; возможно, ему никогда не удастся достичь умения лучше оценивать объем.

Мы мало знаем о том, что происходит в сознании ребенка, когда он думает о числах. Согласно Гелтону [17], тридцать детей из ста будут ассоциировать малые числа с определенными позициями в пространстве перед своим умственным взором, упорядочив их неким специфическим способом, как показано на рис. 3. Возможно, они сохранят эти ассоциации и в юности и могут пользоваться ими в неясных подсознательных

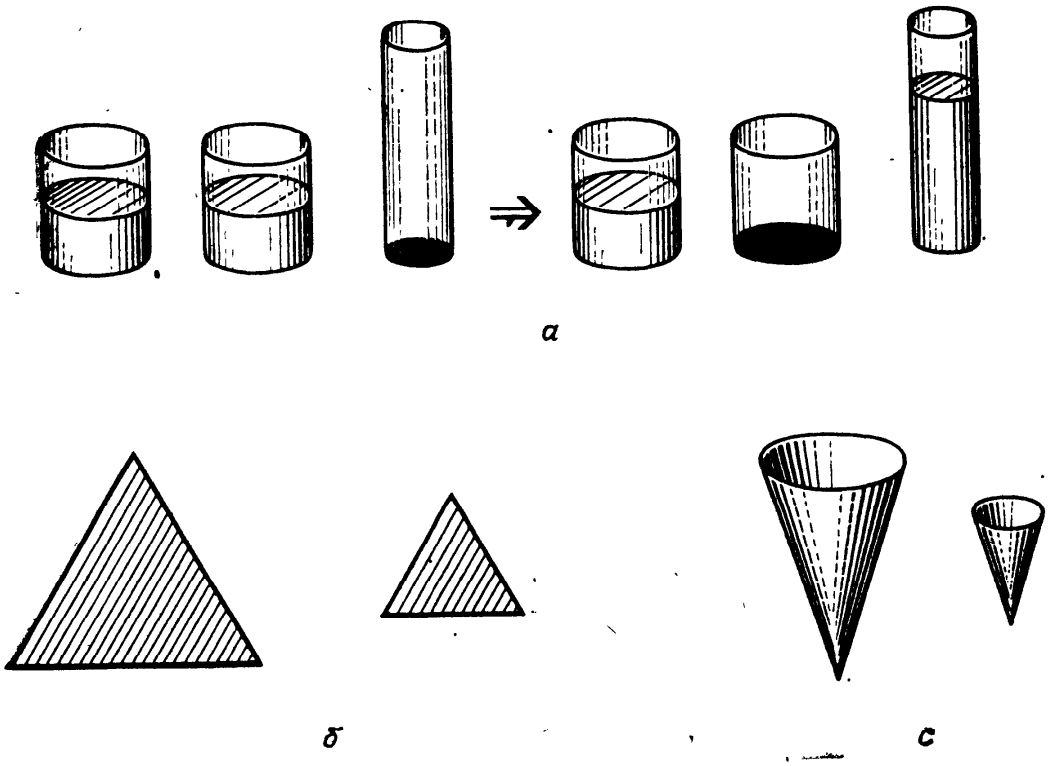

Рис. 2.

процессах запоминания телефонных номеров; а может быть, они сформируют в своем сознании различные пространственно-визуальные представления для исторических дат и т. д. Учитель никогда не почувствует этого, и если ребенок об этом говорит, то даже учитель со своим собственным представлением о числах вряд ли ответит ученику пониманием. (Мой опыт свидетельствует, что требуется ряд тщательно поставленных вопросов, прежде чем один из этих юношей откликнется: «О да! 3 находится над этим местом, чуть сзади».) Когда наш ребенок изучает суммирование столбиком, он может отслеживать переносы цифр в другой разряд, прижимая свой язык к определенным зубам, или использовать некое другое тайное средство временного запоминания, и никто об этом даже не узнает. Возможно, одни способы лучше других.

Детский геометрический мир отличается от нашего. Ребенок не замечает, что треугольники жесткие и этим отличаются от других многоугольников. Он не знает, что аппроксимация окружности стоугольником неотличима от этой окружности, если только она не очень уж велика. Он не начертит куб в перспективе. Он лишь недавно понял, что квадраты становят-

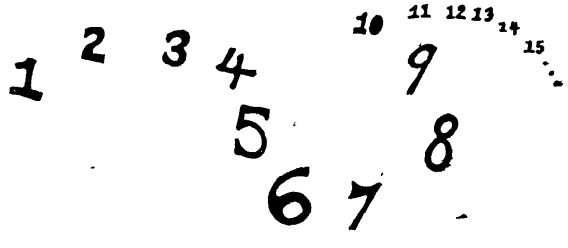

Рис. 3.

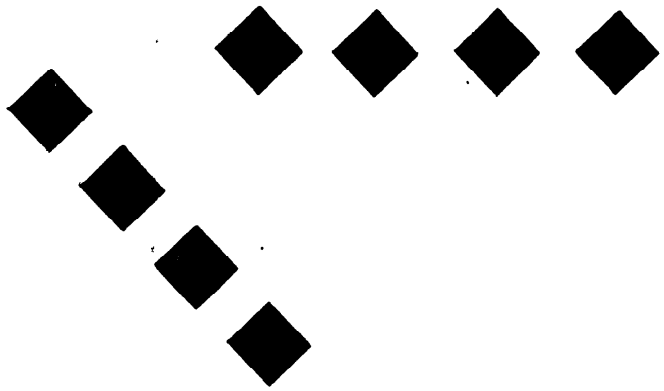

Рис. 4.

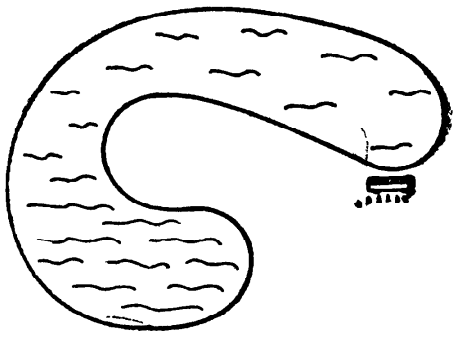

Рис. 5.

ся ромбами, если их повернуть углами вниз. Это перцептивное отличие сохраняется и у взрослых. Так, на рис. 4 мы видим, как заметил Эттниви [18], что восприятие фигур как квадратов или как ромбов зависит от их взаимного расположения на картинке, очевидно, оно определяется нашим выбором осей симметрии, используемых в субъективном описании.

Наш ребенок вполне хорошо понимает топологическую идею замкнутости. Почему же? Это очень сложное понятие классической математики, но в терминах вычислительных процессов оно, по-видимому, оказывается не столь трудным. Но наше дитя почти наверняка будет сбито с толку ситуацией на рис. 5 (см. Пейперт [19]): «Когда автобус начинает экскурсию вокруг озера, мальчик сидит на стороне, удаленной от воды. Окажется ли он на стороне, обращенной к озеру, в некоторый момент экскурсии?» Это затруднение, вероятно, сохранится и на восьмом году жизни ребенка и, по-видимому, связано с его трудностями осознания других абстрактных двойных отрицаний, таких, как вычитание отрицательных чисел или понимание других следствий непрерывности — «в какой точке рейса происходит какое-нибудь внезапное изменение?», — или осознания других мостов между логикой и всеобщностью.

Более детально наш портрет вырисовывается в литературе по психологии развития. Но еще никому не удавалось построить вычислительную модель ребенка, достаточную для того, чтобы увидеть, как эти возможности и ограничения связываются в структуру, совместимую с его столь ярко выраженными способностями к другим делам (а возможно, и логически следующую из этих способностей). Однако такая работа находится в самом начале, и я предполагаю увидеть в следующем десятилетии существенное развитие таких моделей.

Если бы мы больше знали об этих предметах, то смогли бы помочь ребенку. В настоящее время у нас даже нет хорошей диагностики; его явная способность учиться давать правильные ответы на формальные вопросы может свидетельствовать

только о том, что он разработал некоторые изолированные библиотечные подпрограммы. Если они не могут быть вызваны его центральными программами решения проблем из-за использования несовместимых структур данных или по какой-либо причине, мы можем получить проворного решателя тестов, который никогда не будет вполне хорошо мыслить.

До перехода к вычислительным моделям совокупность идей о природе мышления была слишком слабой для того, чтобы поддерживать эффективную теорию обучения и развития. Ни модели с конечным числом состояний бихейвиористов, ни гидравлические и экономические аналогии фрейдистов, ни внезапные озарения гештальт-психологии не дают достаточных оснований для понимания столь затруднительной проблематики. Для этого требуется фундамент из уже отлаженных теорий и решений сопоставимых, но более простых проблем. Теперь же у нас бьют ключом хорошо сформулированные и реализованные идеи для мышления о мышлении, лишь малая часть которых представлена в традиционной психологии:

таблица символов
простая процедура
разделение времени
последовательность вызовов
функциональный аргумент
защита памяти
таблица диспетчирования
сообщение об ошибке
прослеживание вызовов функций
точка прерывания
языки
компилятор
косвенный адрес
макроопределение
список свойств
тип данных
хеширование
микропрограмма
согласование по формату
замкнутые подпрограммы
магазинный список
прерывание
ячейка связи
общая память
дерево решений
компромисс аппаратного и программного обеспечения
последовательно-параллельный компромисс
компромисс затрат времени и памяти
условная точка прерывания
асинхронный процессор
интерпретатор
сборка мусора
списковая структура
блочная структура
упреждение
оглядка
диагностическая программа
управляющая программа

Это лишь немногие идеи, относящиеся к общему системному программированию и отладке; мы ничего не сказали о многих более специфических понятиях из языков, искусственного интеллекта, вычислительной техники и других развитых отраслей информатики. Все они служат сегодня средствами искусно-

го и хитроумного программирования. Но точно так же, как на смену астрологии пришла астрономия, введенная законами Кеплера, открытие принципов эмпирического объяснения интеллектуального процесса в машинах должно привести к возникновению новой науки. (В обучении мы все еще сталкиваемся с таким же состязанием. В разделе комиксов газеты *Boston Globe* содержится астрологическая страница. Помогите справиться с загрязнением интеллектуальной среды!)

Однако вернемся к нашему ребенку. Как наши вычислительные идеи могут помочь ему в связи с его пониманием чисел? В младенчестве он учится распознавать некоторые специальные парные конфигурации, такие, как две руки или два ботинка. Значительно позднее он узнает о некоторых тройках — возможно, с большим затруднением, потому что окружающая среда содержит немного фиксированных троек: если он найдет три пенса, то, наверное, скоро потеряет или проиграет один из них. В конце концов он найдет некую процедуру манипулирования пятью или шестью предметами и выйдет из под власти находок и потерь. Но когда дело касается более чем шести или семи предметов, он останется во власти забывчивости; даже если его словесный счет безупречен, его процедура нумерации будет несовершенна. Он пропустит некоторые элементы, а другие пересчитает дважды. Мы можем помочь, предложив лучшие процедуры; складывание предметов в коробку почти защищает от глупости, поскольку как бы вычеркивает их. Но для неподвижных объектов он по-прежнему будет нуждаться в некой процедуре мысленного группирования.

Прежде всего нужно постараться понять, что делает ребенок, — в этом может помочь слежение за движениями его глаз, — а может быть, достаточно спросить его. Возможно, он выбирает очередной элемент с помощью некоего ненадежного, почти случайного метода, не располагая хорошими средствами для отслеживания того, что уже сосчитано. Можно было бы предложить: *скольжение курсора, изобретение легко запоминаемых групп, нанесение крупных сеток.*

В каждом случае конструкция может быть либо реальной, либо воображаемой. При использовании сеточного метода нужно помнить, что нельзя засчитывать дважды объекты, которые пересекают линии сетки. Учитель должен объяснить, что это хорошо бы планировать заранее, как показано на рис. 6, искривляя сетку, чтобы избежать двусмысленностей.

Математически важный принцип состоит в том, что «любая правильная процедура счета порождает то же самое число». Ребенок поймет, что правилен любой алгоритм, который (1) *подсчитывает все предметы,* (2) *не засчитывает ни один из них дважды.*

Возможно, это процедурное условие кажется слишком простым; даже взрослый может понять его. Во всяком случае, это не та концепция числа, которая принята в том, что теперь носит общее название «новая математика» и преподается в наших начальных школах. Ниже эти вопросы обсуждаются полемически.

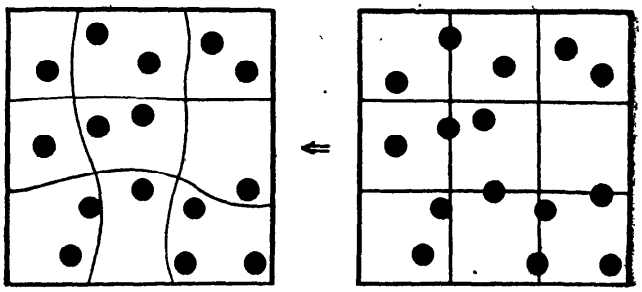

Рис. 6.

3.2. «НОВАЯ МАТЕМАТИКА»

Под «новой математикой» я понимаю некоторые попытки начальной школы имитировать формалистический подход профессиональных математиков. Я полагаю, что эта мода, одновременно охватившая многие школы в связи с новейшими тревогами общественности по поводу состояния начального обучения, плоха из-за свойственных ей разнообразных подмен содержания формой. Последние вызывают трудности и у учителя, и у ребенка.

Вследствие формального подхода учительница не сможет существенно помочь ребенку, у которого возникли затруднения с формулировками. Дело в том, что она будет чувствовать себя неуверенно, заставляя его заучить различие между пустым множеством и ничем или различие между «символами» $3+5$ и цифрой 8, которая является «общим именем» числа восемь, надеясь, что любознательное дитя не спросит, каково общее имя дроби 8/1, которое, наверное, отличается от рационального числа 8/1, и от частного 8/1, и от «указанного деления» 8/1, и от упорядоченной пары (8/1). Педагог воздержится от обсуждения параллельных прямых, поскольку знает, что они обычно не пересекаются, но слышал, что могли бы пересечься, если продлить их достаточно далеко. Он побоится, как бы не произошло что-нибудь наподобие того, что случилось однажды в эксперименте у какого-то русского математика. Впрочем, хватит говорить о проблемах учителя. Рассмотрим теперь три вида протестов с позиции ребенка.

Протесты развития. Очень плохо, когда утверждают, что ребенок хранит свои знания в виде простой упорядоченной иерархии понятий. Чтобы находить то, что ему нужно, он должен обладать многосвязной сетью, чтобы иметь возможность испытывать несколько способов достижения каждой цели. Он может не справиться с приспособлением именно первого способа к условиям задачи. На этом этапе упор на «формальное дока-

зательство» является деструктивным, потому что знания, нужные для поиска доказательств и их понимания, гораздо более сложны (и менее полезны), чем знания, упоминаемые в уже найденных доказательствах. Сеть знаний, требующаяся для понимания геометрии, сплетена из примеров и явлений, а также наблюдений за сходствами и различиями между ними. В детстве не кажется очевидным, что такие сплетения упорядочены, подобно аксиомам и теоремам логической системы, или что ребенок мог бы воспользоваться такой конструкцией, если бы обладал ею. *После* того как явление понято, может иметь большое значение создание для него формальной системы для облегчения понимания более развитых предметов. Но даже в этом случае такая формальная система является всего лишь одной из многих возможных моделей; сочинители новой математики, по-видимому, путают свою аксиомно-теоремную модель с самой системой чисел. Я утверждаю, что в случае аксиом арифметики соответствующий формализм может принести больше вреда, чем пользы для понимания более развитых предметов.

Исторически используемый в новой математике «множественный» подход возник из попытки формалистов вывести интуитивно воспринимаемые свойства континуума из теории почти конечных множеств. Они отчасти преуспели в этом трюке (или в «хекерстве», как выразились бы некоторые программисты), но столь сложным образом, что у тех, кто следует этой модели, исчезает возможность серьезно говорить о вещественных числах, пока не будут усвоены понятия, изучаемые на втором курсе высшей школы. Постижение идей топологии при этом откладывается на еще более поздний срок. Однако дети в свои шесть лет уже обладают хорошо развитыми геометрическими и топологическими идеями, только они плохо умеют манипулировать абстрактными символами и определениями. Нам следовало бы строить преподавание, основываясь на сильных сторонах ребенка, вместо того чтобы оболванивать его, пытаясь заменить то, чем он владеет, на структуры, которыми он еще не в состоянии оперировать. Но это как раз в духе математиков, — наверняка худших в мире растолковывателей, — думать: «Вы можете чему-нибудь научить ребенка, только если пользуетесь достаточно точными определениями»; «Если мы с самого начала все правильно определим, у нас впоследствии не возникнет никаких затруднений». Мы не программируем на Фортране для пустой машины, хуже того — мы лезем внутрь плохо понимаемой нами большой системы, для которой характерно при ее естественном эвристическом поведении использование многократно и различными способами описанных символов.

Интуитивные протесты. В новой математике акцентируется

идея, что число может быть идентифицировано с классом эквивалентности всех множеств, которые могут быть поставлены во взаимно однозначное соответствие друг с другом. При этом рациональные числа определяются как классы эквивалентности пар целых, и прилагается уйма усилий, чтобы *предостеречь ребенка от идентификации рациональных чисел с частными или дробями.* Функции часто интерпретируются как множества, хотя в некоторых текстах представлены «функциональные машины» с поверхностным привкусом понятия алгоритма. Определение «переменной» — это еще одно дьявольское хитросплетение имен, значений, выражений, фраз, предложений, символов, «указанных операций» и тому подобного. (На самом деле при реальном решении проблем возникает так много различных видов данных, что настоящие математики обычно не занимаются их формальной классификацией, а используют для их объяснения контекст, присущий конкретной проблеме.) В погоне за этой формалистической навязчивой идеей учебный курс никогда не представляет сколько-нибудь связной картины реальных математических явлений, т. е. процессов дискретных или непрерывных; алгебры, а не только столь усердно пережевываемого синтаксиса ее нотации; а также геометрии. «Доказываемые» время от времени «теоремы» наподобие такой: «У числа x имеется только одно аддитивное обратное число, $-x$» столь приземлены и очевидны, что ни преподаватель, ни учащийся не понимает цели таких доказательств. «Официальное» доказательство состоит в прибавлении величины y к обеим частям равенства $x+(-y)=0$, применении закона ассоциативности, затем закона коммуникативности, затем закона $y+(-y)=0$ и в заключение аксиом равенства, чтобы показать, что значение y должно равняться x. Детский ум может легче воспринять более глубокие идеи: «В равенстве $x+(-y)=0$, если y меньше x, имелся бы некий излишек в левой части, а если x меньше y, в левой части имелось бы отрицательное число, и поэтому x и y должны быть в точности равны». Ребенку не разрешают пользоваться этим видом упорядоченно-непрерывного мышления преимущественно потому, что «в нем используются более продвинутые знания», поэтому он якобы не дает «настоящего доказательства». Но в нужной ребенку сети идей эта связь имеет равный логический статус и заведомо большее эвристическое значение. Приведу еще один пример. Учащегося заставляют четко различать операцию, обратную сложению, и расстояние, отсчитываемое в обратном направлении, хотя, казалось бы, желательно слияние этих понятий.

Вычислительные протесты. Идея процедуры и сноровка, которая приходит с изучением тестирования, модификации и приспосабливания процедур, могут перейти во многие другие виды

действий ребенка. Такие традиционные академические предметы, как алгебра и арифметика, сравнительно мало влияют на развитие, особенно когда они не подкрепляются интуитивной геометрией. (Вопрос о том, какие виды учебных знаний помогают в других занятиях, является фундаментальным для теории обучения: я снова подчеркиваю наше предположение, что идеи процедуры и отладки окажутся уникальными по своему проникновению в самые различные сферы деятельности.) Мы уже отмечали, что в алгебре понятие «переменной» переусложнено, но в вычислениях ребенок может легко увидеть в «$x+y+z$» описание процедуры (любой процедуры сложения!), причем «x», «y» и «z» указывают на данные для процедуры. Функции легко воспринять как процедуры, но это трудно сделать, представляя их упорядоченными парами. Если вам нужен граф, то опишите машину, которая чертит граф; когда у вас есть граф, опишите машину, которая может читать его для отыскания значений функций. В обоих случаях вы имеете дело с легкими и полезными понятиями.

Давайте избежим культурной ловушки; теоретико-множественные «основы» математики в наши дни популярны у математиков потому, что именно на них эти математики муштровались и воспитывались (в колледже). Эти ученые, как правило, просто незнакомы с вычислениями или с семейством теорий Поста — Тьюринга — Маккалоха — Питса — Маккарти — Ньюэлла — Саймона..., которые будут гораздо важнее, когда ребенок вырастет. Вопреки мнению логиков и издателей, теория множеств не является *единственным* и истинным основанием математики; этот подход отменно хорош для исследования бесконечности, но малопригоден для понимания вещественных чисел и совсем бесполезен для изучения арифметики, алгебры и геометрии.

Если суммировать мои возражения, суть их состоит в том, что новая математика делает упор на формализм и манипуляции символами вместо того, чтобы заниматься эвристическим и интуитивным содержанием существа дела. Предполагается, что ребенок научится решать проблемы, но мы не учим его тому, что знаем сами — ни об учебном предмете, ни о решении задач[1].

[1] В проницательном, хотя и излишне веселом разборе руководств по новой математике Фейнман [20] объясняет следствия проведения различия между вещью и ее представлением. «Окрасьте в красный цвет рисунок мяча», — гласит книга вместо того, чтобы сказать: «Окрасьте мяч в красный цвет». «Должны ли мы красить весь квадрат, в котором присутствует изображение мяча, или только часть внутри окружности мяча?» — спрашивает Фейнман. («Окрасить мячи в красный цвет» должно бы, по-видимому, писаться так: «Окрасить внутренности всех окружностей всех элементов множества мячей» или примерно в этом духе.)

В качестве примера того, как озабоченность формой (в данном случае аксиомами арифметики) может исказить впечатление от существа, изучим странное пристрастие настаивать на том, что сложение в конечном счете является операцией ровно над двумя величинами. В новой математике выражение $a+b+c$ должно «на самом деле» быть либо $(a+(b+c))$, либо $((a+b)+c)$, а выражение $a+b+c+d$ может обрести содержательный смысл только после нескольких применений закона ассоциативности. Это во многих отношениях является глупостью. Ребенок уже обладает хорошим интуитивным представлением о том, что значит собрать несколько множеств вместе; ведь одинаково легко смешать в кучу шарики пяти или двух цветов. Таким образом, сложение уже воспринимается как n-местная операция. Однако прислушаемся к книжной попытке доказать, что это не так:

Сложение... всегда выполняется над двумя числами. На первый взгляд это может показаться необоснованным, так как вы раньше часто складывали длинные строки цифр. Попробуйте проэкспериментировать на себе. Попытайтесь одновременно сложить числа 7, 8, 3. Вне зависимости от того, каким образом вы стараетесь сделать это, вы вынуждены выбрать два числа, сложить их, а затем прибавить третье число к их сумме.

(*Из учебника для девятого класса.*)

Разве высота башни — это результат попарного суммирования высоты ее ступенек в определенном порядке? Получается ли длина или площадь предмета аналогичным образом из его частей? Зачем им понадобилось вводить свои множества и их взаимно однозначные соответствия, чтобы потом упустить суть дела? Очевидно, что они так часто это повторяли, что и в самом деле поверили, будто выбранные ими аксиомы для алгебры обладают неким особым качеством истины в последней инстанции!

Рассмотрим несколько важных и занятных идей, которым не уделяется особого внимания в средней школе. Сначала рассмотрим сумму $1/2+1/4+1/8...$ Интерпретируя ее как площадь, человек овладевает пленительными идеями перегруппировки, продемонстрированными на рис. 7. Научившись делить, ребе-

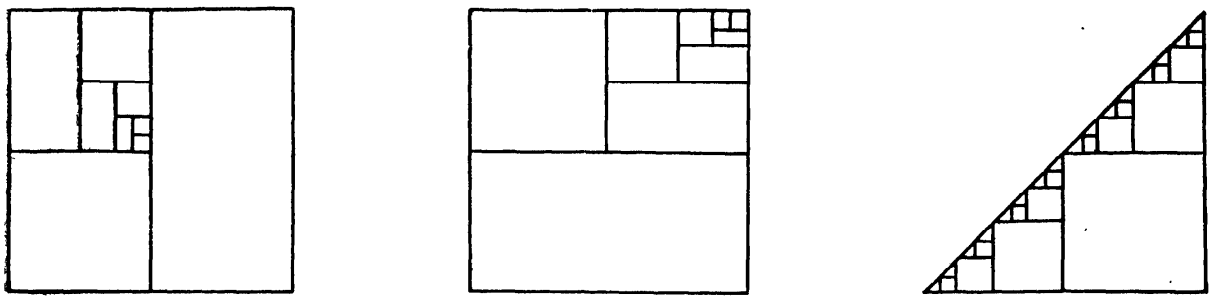

Рис. 7.

нок может вычислять и ощущать некоторые количественные аспекты предельного процесса 0.5, 0.75, 0.875, 0.9375, 0.96875,... и может узнать о складывании и разрезании, об эпидемиях и популяциях. Он мог бы узнать о тождестве $x=px+qx$, где $p+q=1$, и тем самым оценить процесс разведения, он может узнать, что 3/4, 4/5, 5/6, 6/7, 7/8,...$\Rightarrow 1$, и начать понимать многие яркие и здравые геометрические и топологические следствия из таких отвлеченных материй.

Но в новой математике синтаксические границы между рациональными числами, частными и дробями заходят так далеко, что не разрешается для нахождения большей из дробей 3/8 и 4/9 вычислить и сравнить .375 и .4444. От нас *требуют перекрестное перемножение* числителей и знаменателей. Хотя перекрестное перемножение очень привлекательно, ему присущи два дефекта: (1) никто не помнит, в какие стороны нужно переходить в зависимости от результата проверки условия, и (2) мы не получаем информации о том, насколько далеки друг от друга сравниваемые числа. Абстрактное понятие порядка весьма изящно (еще один набор аксиом для очевидного), но дети уже прекрасно понимают упорядоченность и хотят знать количественные соотношения.

Еще одной навязчивой идеей является понятие основания системы счисления. Хорошо, когда ребенок отчетливо понимает, что 223 — это «две сотни» плюс «двадцать» плюс «три», и я думаю, что эту мысль нужно довести до его сознания самым простым образом, а не усложнять ее[1]. Я не считаю эту идею настолько богатой, что требуется муштровать малое дитя в арифметических упражнениях в нескольких системах счисления! Дело в том, что эта хилая идея мало связана с другими предметами, и существует риск изувечить хрупкую арифметику учеников, которые, усвоив с трудом, что $6+7=13$, теперь обнаруживают, что $6+7=15$. Кроме того, я не вижу в учебниках моих детей при всем внимании к основаниям систем счисления хоть каких-нибудь нетривиальных выводов — понятий, которые могли бы оправдать это внимание, например:

> Почему десятичное целое число можно написать только одним способом?
> Почему работает отбрасывание девяток? (Это даже не упоминается.)
> Что произойдет, если пользоваться произвольными нестепенями, например, $a+37b+24c+11d+\ldots$ вместо обычных $a+10b+100c+1000d+\ldots$?

Если они не обсуждают таких вопросов, то должны иметь другую цель. Я предполагаю, что вся суета поднята ради того, чтобы детишки лучше понимали процедуру умножения и деления. Но с точки зрения развития это может оказаться серьезной методической ошибкой для курсов обучения как старой,

[1] См. песню Т. Лейрера «Новая математика» [21].

так и «новой» математике. Стандартный алгоритм деления на бумаге является, мягко выражаясь, громоздким, и большинство детей никогда не воспользуются им для изучения числовых явлений. И хотя довольно любопытно понять, как работает этот алгоритм, выписывание всей картинки заставляет предположить, что педагог верит, будто ребенок должен всякий раз понимать черт знает что! Это неверно. Существенная идея алгоритма, если такая есть, состоит в повторяемом вычитании; все остальное является хитроумным, но не жизненно необходимым программистским трюком.

Если мы хотим научить, хотя бы методом зубрежки, практическому алгоритму деления, очень мило. Но в любом случае давайте снабдим детей маленькими калькуляторами; если это потребует слишком больших расходов, почему бы не отбарабанить правила? Но, пожалуйста, без невыносимых объяснений. Важно справляться с вещественными числами. Принятая в новой математике озабоченность целыми числами столь фанатична, что напоминает (если позволительно упомянуть другую псевдонауку) нумерологию. (Какого мнения об этом *Boston Globe*?)

Теоретико-множественный формализм Дедекинда — Рассела — Уайтхеда достойно дополнил следующую (после Эвклида) серию демонстраций того, что из немногих примитивов можно вывести много математических понятий, хотя бы и длинным и извилистым путем. Однако у ребенка проблема состоит вовсе не в том, чтобы овладевать хоть какими-то понятиями; ему нужно познавать реальный мир. С точки зрения доступных ему понятий весь формализм теории множеств «в подметки не годится» одной-единственной, более древней, более простой и, возможно, более великой идее: представления интуитивной вещественной прямой бесконечной десятичной дробью.

Существует реальное противоречие между целями логика и преподавателя. Логик хочет минимизировать разнообразие идей, и его не беспокоит длинный, тонкий путь вывода. Педагог (в идеале) хочет сделать пути как можно короче, и его не смущают, а на самом деле даже радуют связи со многими другими идеями. И он почти вовсе не беспокоится о направлениях связей.

Что же касается лучшего понимания целых чисел, то я полагаю, что в этом не могут помочь бесконечные упражнения малых детей, понуждаемых чертить схемы взаимно однозначного соответствия. Несомненно, эти упражнения помогут их обучению существенным алгоритмом, но не для чисел, а для важных топологических и процедурных проблем черчения путей без пересечений и тому подобного. Именно этой отнюдь не маловажной проблеме нам следует уделять внимание.

Итак, специалист по информатике несет ответственность за образование. Не потому, как он ошибочно полагает, что он должен будет программировать обучающие машины. И, конечно, не потому, что он искусный пользователь «дискретной математики». Он знает, почему плохо пользоваться трюками, которые потом мстят за себя при отладке и расширении программ. (Так, можно привлечь интерес детишек, связав малые числа с произвольными цветами. Но как пригодится этот фокус в их дальнейших попытках применять понятие числа к площади, объему или к значению функции?) Именно специалист по информатике должен исследовать эти вопросы, потому что он владеет понятием процедуры — секретом, который так долго искали педагоги.

ЛИТЕРАТУРА

1. Feynman R. P. Development of the space-time view of quantum electrodynamics. Science 153, No. 3537 (Aug. 196), 699—707.
2. Shannon C. E. A universal Turing machine with two internal states. In Automata Studies, Shannon C. E., and McCarthy J. ; (Eds.), Princeton U. Press, Princeton, N. J., 1956, p. 157—165.
3. Cook S. A. On the minimum computation time for multiplication. Doctoral diss., Harvard U., Cambridge, Mass., 1966.
4. Knuth D. The Art of Computer Programming, Vol. II. Addison-Wesley, Reading, Mass., 1969.
5. Minsky M. and Papert S. Perceptions: An Introduction to Computational Geometry. MIT Press, Campridge, Mass., 1969.
6. Guzman A. and McIntosh H. V. CONVERT. Comm. ACM 9, 8 (Aug. 1966), 604—615.
7. Hewitt C. PLANNER: A language for proving theorems in robots. In: Proc. of the International Joint Conference on Artificial Intelligence, May 7—9, 1969, Washington, D. C., Walker, D. E. and Norton, L. M. (Eds.), pp. 295—301.
8. Levin M., et al. The LISP 1.5 Programmer's Manual. MIT Press, Cambridge, Mass, 1965.
9. Weissman C. The LISP 1.5 Primer. Dickenson Pub. Co., Belmont, Calif., 1967.
10. Martin W. A. Symbolic mathematical laboratory. Doctoral diss., MIT, Cambridge, Mass., Jan. 1967.
11. Moses J. Symbolic integration. Doctoral diss., MIT, Cambridge, Mass., Dec. 1967.
12. Seagle J. R. A heuristic program that solves symbolic integration problems in Freshman calculus. In Computers and Thought, Feigenbaum E. A. and Feldman J. (Eds.), McGraw-Hill, New York, 1963.
13. Turing A. M. Computing machinery and intelligence. Mind 59 (Oct. 1950), 433—460; перепечатка в Computers and Thought, Feigenbaum E. A. and Feldmen J. (Eds.), McGraw-Hill, New York, 1963.
14. Minsky M. Matter, mind and models. Proc. IFIP Congress 65, Vol. 1, pp. 45—49 (Spartan Books, Washington D. S.). Перепечатка в Semantic Information Processing, Minsky M. (Ed.), MIT Press, Cambridge, Mass., 1968, pp. 425—433.
15. Feigenbaum E. A., and Feldman J. Computers and Thought. McGraw-Hill, New York, 1963.

16. Minsky M. (Ed.). Semantic Information Processing. MIT Press, Cambridge, Mass., 1968.
17. Galton F. Inquiries into Human Faculty and Development. Macmillan, New York, 1983.
18. Attneave F. Triangles as ambiguous figures. Amer. J. Psychol. 81, 3 (Sept. 1968), 447—453.
19. Papert S. Principes analogues a la recurrence. In Problems de la Construction du Nombre, Presses Universitaires de France, Paris, 1960.
20. Feynman R. P. New textbooks for the "new" mathematics. Engineering and Science 28. 6 (March 1965), 6—15 (California Inst. of Technology, Pasandena).
21. Lehrer T. New mat. In That Was the Year that Was, Reprise 6179, Warner Bros. Records.