Доктор физико-математических наук
А. Л. СЕМЕНОВ,
доктор физико-математических наук
В. А. УСПЕНСКИЙ

МАТЕМАТИЧЕСКАЯ ЛОГИКА В ВЫЧИСЛИТЕЛЬНЫХ НАУКАХ И ВЫЧИСЛИТЕЛЬНОЙ ПРАКТИКЕ

Все более существенное место в системе современного научного знания и научной деятельности занимают «вычислительные науки», предметом которых является вычислительная техника, программное и алгоритмическое обеспечение, а также всевозможные применения ЭВМ. В Академии наук СССР этот процесс организационно зафиксирован созданием Отделения информатики, вычислительной техники и автоматизации.

К вычислительным наукам относятся разделы самых различных естественных и общественных наук, выступающих, как правило, одновременно в роли и потребителей, и поставщиков научных и практических результатов для рассматриваемой области — физики, химии, математики, лингвистики, биологии, психологии и т. д. Важное место среди вычислительных наук занимает соответствующая часть математики, называемая по-разному: теоретическая информатика, математическая теория вычислений, вычислительная теория, алгоритмика, кибернетика и т. д. В зарубежных странах используются названия: «computer science», «informatique».

Часть математики, относящаяся к вычислительным наукам, тоже неоднородна. В нее входят разделы математической физики, изучающие уравнения физических процессов, которые протекают в элементах вычислительной техники, теория численных методов решения дифференциальных уравнений, теория массового обслуживания, используемая при построении операционных систем и в приложениях ЭВМ, теория игр и т. д. Однако центральное место в этой связанной с вычислительными науками части математики принадлежит математической логике и теории алгоритмов. Часто обе эти дисциплины объединяют под общим названием «математическая логика», считая теорию алгоритмов составной частью математической логики в широком смысле слова. Мы будем на протяжении статьи придерживаться этой традиции.

С согласия академика А. Н. Колмогорова в основу данной статьи положены доклады А. Н. Колмогорова, А. Л. Семенова и В. А. Успенского, посвященные математическим аспектам вычислительных наук и связанным с этими аспектами вопросам математической подготовки кадров.

Один из самых выдающихся программистов нашего времени Э. Дейкстра в недавно (1986 г.) опубликованной статье называет блестящим предвидением следующее высказывание другого известного специалиста, автора языка программирования ЛИСП Дж. Маккарти, сделанное еще в 1967 г.: «Есть основания полагать, что в следующем столетии связи между вычислительной техникой и математической логикой окажутся столь же плодотворными, какими были связи между математическим анализом и физикой в столетии предыдущем».

Прежде чем перейти к примерам, иллюстрирующим роль математической логики в вычислительных науках, отметим особенность, выделяющую ее среди других разделов математики и объясняющую, на наш взгляд, ее исключительное место в фундаменте вычислительных наук. Дело в том, что среди математических дисциплин математическая логика является единственной дисциплиной, изучающей взаимоотношение между текстами и их смыслом. Математическое исследование и математическое описание этого взаимоотношения приобретают первостепенную важность, когда тексты превращаются из средств общения между людьми в средство взаимодействия с компьютером.

Тексты бывают повествовательные, повелительные, вопросительные. Теория алгоритмов изучает повелительные тексты, называемые в ней программами. Содержание (семантику) такого текста составляет заключенный в ней алгоритм. Связь теории алгоритмов с вычислительной практикой очевидна.

Менее очевидна роль математической логики в узком смысле слова — науки, предметом которой является соотношение между повествовательными текстами логико-математических языков и их смыслами. Тем не менее оказывается, что с развитием программирования эта роль становится все более и более важной: от явного, «процедурного», «повелительного» описания все чаще переходят к «непроцедурному», «функциональному», «повествовательному» программированию. В связи с этим достаточно упомянуть функциональный язык программирования ЛИСП, который за 20 лет существования не только не утратил актуальности, но служит одним из главных инструментов проекта вычислительных машин пятого поколения.

Возникшая в XIX в. (ее «отцом» следует назвать немецкого ученого Г. Фреге) математическая логика, начиная с 10-х годов нашего столетия пережила бурный подъем. В 30-е годы сформировались ее основные понятия и направления. То, что было тогда создано, оказалось математикой, «спроектированной под» еще не существовавшую реальность; такой реальностью оказались ЭВМ, возникшие лишь впоследствии, в середине столетия. Такие ситуации встречались и прежде: геометрия Лобачевского возникла раньше, чем у физиков появилась потребность в ней.

Необходимость изучения основного семантического отношения между текстами и их смыслами привела к задаче математического изучения множеств текстов самих по себе — этим занимается теория формальных грамматик и языков. Вычислительные процессы и устройства изучает теория автоматов и машин, а математические структуры, с помощью которых определяется семантика логических формул, — алгебра и т. д.

Однако центральное место в этой области занимает именно математическая логика. Не случайно Дж. фон Нейману и А. Тьюрингу — ученым, стоявшим у колыбели первых компьютеров, — принадлежит и ряд наиболее фундаментальных вкладов в математическую логику.

Воздействие математической логики па вычислительную практику осуществляется прежде всего в виде формирования общей методологии построения аппаратного и программного обеспечения. Во вторую очередь

это влияние сказывается в выборе языков и вычислительных моделей. Использованию же конкретных теорем и алгоритмов математической логики в вычислительной практике следует, как нам кажется, отдать третье место (хотя для вычислительной теории некоторые фундаментальные теоремы гносеологического характера — такие, как теорема Геделя о неполноте — имеют первостепенное значение).

Общие концепции, понятия, теоремы

Одно из важнейших достижений математики XX в.— открытие общего понятия *вычислимой функции,* то есть функции, вычисляемой каким-либо алгоритмом. Почти одновременными и тесно связанными с этим оказались другие открытия.

Первое из них — *тезис Чёрча.* Попросим любого человека, немного знакомого с работой вычислительной машины, придумать свой язык программирования или описать какой-либо из тех языков, с которыми ему приходилось встречаться. Почти наверняка окажется, что получившийся язык будет *универсальным,* если допустить использование потенциально неограниченных ресурсов. Другими словами, этот язык будет пригоден для записи (может быть, неестественной, неэкономной) программы вычисления любой вычислимой функции. Таким образом, взяв небольшое исходное семейство простейших вычислимых функций и несколько основных алгоритмических конструкций, мы получаем язык программирования, пригодный для задания произвольных вычислимых функций. Для конкретного языка программирования это утверждение (а именно, утверждение о возможности для любой вычислимой функции написать ее программу на рассматриваемом языке) и называется тезисом Чёрча. Этот тезис является математическим выражением следующей идеи: вместо того, чтобы строить специализированные устройства для решения тех или иных конкретных задач, можно построить машину, исполняющую программы некоторого универсального языка программирования, а затем составлять программы решения нужных задач на этом языке.

Эта идея, кажущаяся теперь почти очевидной, была выдвинута создателями теории алгоритмов Э. Л. Постом, А. М. Тьюрингом, А. Чёрчем в 30-е годы и стала важнейшим шагом на пути создания ЭВМ. Недавно она пережила второе рождение в связи с применением универсальных микропроцессоров, заменившим разработку и изготовление различных специализированных устройств автоматики.

Многие понятия программирования были предсказаны в теории алгоритмов. Отметим среди них понятие транслятора, предвосхищенное понятием алгоритмической сводимости нумераций вычислимых функций (А. П. Колмогоров), и, как недавний пример, понятие смешанного вычисления в трактовке А. П. Ершова.

Оставим на время языки повелительных предложений и перейдем к другим. Прежде всего, естественные языки допускают конструкции «разрешающего типа», п о з в о л я ю щ и е (а не п р е д п и с ы в а ю щие, как это делают алгоритмы) выполнять те или иные действия. При этом выбор действия зависит от воли участников, случая и т. д. Возникают понятия исчисления, вероятностного алгоритма, игры.

Остановимся на понятии *исчисления.* Как конкретному алгоритму соответствует конкретная функция, вычисляемая этим алгоритмом,— так конкретному исчислению соответствует конкретное множество, порождаемое этим исчислением. Как общему понятию алгоритма соответствует общее понятие вычислимой функции (то есть функции, вычисляемой каким-л ибо алгоритмом), так общему понятию исчисления соответ-

ствует общее понятие *породимого множества* (множества, порождаемого каким-либо исчислением). Например, породимым является множество всех верных равенств между неопределенными интегралами и формулами, задающими первообразные (ср. термин «интегральное исчисление»). Другим примером породимого множества является множество всех выражений формального языка — языка математической логики, языка программирования и т. д., построенных в соответствии с правилами этого языка. Все эти множества порождаются соответствующими исчислениями.

Для породимых множеств можно аналогично тезису Чёрча сформулировать *тезис Поста* о совпадении интуитивного понятия породимости с понятием породимости некоторым универсальным исчислением (с параметром — описанием конкретных правил порождения). Понятие породимости оказывается тесно связанным с понятием вычислимости. Именно, породимые множества — это в точности множества значений (а также в точности области определения) вычислимых функций. С другой стороны, среди породимых множеств есть такие, для которых невозможен алгоритм распознавания принадлежности к ним. В частности, не существует алгоритма проверки по любой программе на универсальном алгоритмическом языке, не зацикливается ли она при заранее фиксированном исходном данном.

Сама по себе идея отделения текста на языке от его смысла, возможность сопоставления с текстом различных смыслов (и возможность различных способов такого сопоставления) является революционной и очень важной для программирования. Формальные определения семантики для языков математической логики послужили образцом для многих построений в программировании. Тонкое взаимодействие между повествовательной (функциональной) и повелительной (операционной) семантиками является важным моментом в достижении большей надежности математического обеспечения, программ и оборудования. С этими вопросами связана еще одна большая область — доказательство правильности программ. Сама постановка задачи доказательства правильности была бы невозможной без достижений математической логики. Ведь вопрос о том, правильно ли работает программа, допускает тривиальный ответ: она работает в соответствии с самой собой, именно так, как она написана. Этот ответ, конечно, неудовлетворителен.

В действительности один из начальных этапов разработки программы состоит в задании ее *спецификации,* то есть в указании на некотором повествовательном логико-математическом языке сведений о том, в каких условиях должна работать и какого результата должна достигать программа. Доказательство правильности программы — ее соответствия заданной спецификации — представляет собой нечто аналогичное математическому доказательству (наиболее близкий аналог — доказательство в геометрических задачах на построение). Дополнительные трудности, по сравнению с традиционной математикой, состоят в том, что доказательству подлежат свойства программы, связанные с ее выполнением — процессом, разворачивающимся во времени.

Потребность в строгости и достоверности получаемых доказательств привела к созданию формальных систем, предназначенных для доказательства правильности программ. Источником таких систем являются формальные дедуктивные системы математической логики. Естественно, что для систем доказательства правильности программ есть и ограничение на их доказательную силу: в соответствии с теоремой Геделя о неполноте для любой мыслимой системы доказательств найдутся правильные программы, правильность которых невозможно доказать.

Теорема Геделя о неполноте (не смешивать с теоремой Геделя о полноте!) — одно из величайших достижений математической мысли XX в. Она заключается в следующем.

Пусть мы имеем какую-либо достаточно богатую и непротиворечивую систему аксиом, описывающую свойства натурального ряда, то есть свойства натуральных чисел и операций сложения и умножения. Термин «достаточно богатая» означает, что из системы аксиом можно получить (вывести) достаточно содержательные следствия, то есть что определенные, заранее указанные истины относительно натурального ряда могут быть выведены, или доказаны, отправляясь от этой системы. Термин «непротиворечивый» означает, что ни для какого утверждения не может случиться, что в рассматриваемой системе доказывается как само это утверждение, так и его отрицание. Оба требования (достаточное богатство и непротиворечивость) совершенно естественны: системы аксиом, не удовлетворяющие этим требованиям, могут представлять лишь второстепенный интерес.

Пусть далее процесс получения следствия из аксиом формализован настолько, что может быть поручен вычислительной машине (или не слишком квалифицированному, но педантичному и исполнительному лаборанту). Тогда непременно найдется такое истинное утверждение о натуральном ряде, которое не может быть получено указанным формальным образом из рассматриваемых аксиом (если же мы попытаемся присоединить это истинное утверждение к нашей аксиоматике в качестве новой аксиомы, немедленно возникнет новая недоказуемая истина). Здесь важно подчеркнуть, что система аксиом и правила получения следствий из нее выбираются совершенно произвольно — лишь бы эти правила носили механический характер и соблюдались описанные выше естественные требования. Таким образом, невозможна никакая система аксиом и формальных, механических правил вывода, обеспечивающая получение всех истин, относящихся к арифметике натуральных чисел, а тем более всех истин, относящихся к какой-либо математической теории, содержащей арифметику как часть.

Остается прибавить, что практически любая математическая теория (и уж во всяком случае, теоретическое программирование) включает в себя натуральные числа и простейшие операции над ними и, следовательно, содержит арифметику как свою часть. Поэтому теорема о неполноте имеет фундаментальное гносеологическое значение — она показывает неисчерпаемость творческого мышления как средства познания.

Следующее открытие, возникшее в математической логике и играющее сейчас важную роль в практических задачах,— понятие *сложности вычисления.* Это понятие исследовалось и развивалось параллельно и в большей степени независимо от потребностей реальной вычислительной практики. Однако полученные теоретические результаты оказали существенное и очень широкое, хотя и не прямое, влияние на решение реальных задач программирования. Это можно проследить на примерах так называемых переборных задач. Напомним, что в теории сложности вычислений алгоритм принято считать *достаточно быстрым,* если время его работы на данном входе, или аргументе, выражается полиномом от объема аргумента (входа). В течение длительного времени разработчики математического обеспечения затрачивали значительные усилия на попытку построить достаточно быстрый алгоритм решения различных задач, имеющих следующий общий вид.

Пусть задача состоит в отыскании для каждого объекта x либо некоторого объекта y, для которого выполнено заданное соотношение $P(x, y)$, либо сигнала о том, что такого y не существует. Пусть далее ответ на вопрос, выполнено ли соотношение $P(x, y)$ для данных x, y, получается посредством достаточно быстрого алгоритма. Пусть, наконец, заранее известны границы на возможные y, скажем, известно, что объем y не превосходит объема x. Тогда есть тривиальный алгоритм решения по-

ставленной задачи. Он состоит в последовательном переборе всех *y,* не превосходящих заданной границы, и проверке соотношения *P (x, y)* для каждого *y.*

Например, пусть требуется найти в графе *x* путь *y,* проходящий через каждую вершину ровно по одному разу (гамильтонов путь). Тогда *P(x, y)* как раз и означает, что *y* есть гамильтонов путь в графе ж; граница по объему очевидна. Среди других примеров описанного типа существуют различные задачи на графах (раскраска графа, поиск максимального паросочетания и т. д.), задача целочисленного линейного программирования и другие оптимизационные задачи и т. д.

Значительные усилия были затрачены на то, чтобы для каждой из. этих задач найти не просто алгоритм решения, но достаточно быстрый алгоритм. Ни для одной из большого числа задач найти требуемый достаточно быстрый алгоритм не удавалось. Однако поиски продолжались. Прогресс наступил в теории.

Во-первых, было доказано, что все упомянутые задачи эквивалентны в том смысле, что построение достаточно быстрого алгоритма для одной из них автоматически приводит к нахождению достаточно быстрого алгоритма для всех остальных. Эти эквивалентные друг другу задачи получили название *переборных.*

Во-вторых, была сформулирована гипотеза (завоевывающая все большее признание), что ни для одной из переборных задач достаточно быстрого алгоритма не существует.

Почему сочетание этой теоремы об эквивалентности переборных задач и этой гипотезы следует считать прогрессом? Дело в том, что без теоретического осознания ситуации каждая новая переборная задача приводила бы к новым бесплодным попыткам кустарными средствами найти для нее достаточно быстрый алгоритм. Теперь же происходит следующее: специалист-алгоритмизатор, столкнувшись с задачей, кажущейся ему переборной, может обычно довольно быстро доказать, что она действительно такова; тогда он модифицирует ее формулировку, выделяет подклассы, представляющие практический интерес, и т. д., после чего пытается построить достаточно быстрый алгоритм в этих подклассах.

Наряду с доказательством переборности отдельных задач теория сложности дает и другие нижние оценки сложности вычислений, показывающие бесплодность попыток существенного улучшения алгоритмов.

В качестве результата развития теории сложности вычислений можно рассматривать и возможность эффективно оценивать и сравнивать различные конкретные алгоритмы (об отыскании самих алгоритмов речь пойдет ниже).

С точки зрения математика не менее фундаментальным, чем понятие сложности вычислений, является понятие *сложности объекта.* Достаточно давно было замечено, что объем описания объекта может быть значительно меньше, чем объем самого объекта. Например, текст «миллион букв ы» много короче текста «ыы...» из миллиона букв «ы». Объем описания объекта и называется его сложностью — при данном *способе описания.* Однако лишь в середине 60-х годов стало ясно, что среди всевозможных способов описания существуют такие, называемые *оптимальными,* которые позволяют получать описания не хуже любого другого способа («не хуже» здесь означает «длиннее не более чем на некоторую, не зависящую от описываемого объекта константу»). В качестве такого оптимального способа описания можно использовать, например, так называемый универсальный. Описание, подаваемое на вход универсального способа, состоит из программы конкретного способа описания и описания объекта при этом способе. Универсальность — возможность имитировать.

любой конкретный способ — и обеспечивает свойство «быть не хуже». Формализация приведенного рассуждения составляет содержание теоремы об оптимальном способе описания (А. Н. Колмогоров). Объем описания, получаемого при оптимальном способе описания, называется *энтропией.* Теория энтропии объектов может быть положена в основу определения, скажем, случайной последовательности. Классическая теория вероятностей не в состоянии дать такое определение, а важность этого понятия для вычислительной практики несомненна (имея в виду, например, метод Монте-Карло).

Касаясь общеметодологической роли математической логики для вычислительной практики, уместно специально выделить значение отрицательных результатов. Понимание, что в принципе невозможно сделать что-то, имеет для программирования не меньшее значение, чем для техники невозможность нарушить второе начало термодинамики или для физики невозможность превысить скорость света.

Два примера важнейших отрицательных результатов математической логики мы уже привели. Один был связан с теоремой Геделя о неполноте и доказательством правильности программ, другой (условный) — с переборными задачами. Безусловно, практически важными являются и другие отрицательные результаты математической логики. Теорема о нераспознаваемости нетривиальных свойств вычислимых функций показывает невозможность существования транслятора, устанавливающего какие-либо нетривиальные свойства программ, например определяющего, зацикливается ли программа на любом вводе.

Языки

Второй источник приложений математической логики и теории алгоритмов в вычислительной практике — формальные языки, разработанные в этой области математики. Хотя важность этого источника нам кажется меньшей, чем идейное влияние общих концепций, обсужденных выше, но зато здесь это влияние более очевидно и иногда просто поразительно. Большинство логических и алгоритмических языков, изобретенных независимо от компьютеров, так или иначе вошло в языки программирования, описание работы ЭВМ и т. д.

Один из самых впечатляющих примеров — язык ЛИСП. Он до сих пор сохраняет свое лидирующее положение в сфере задач искусственного интеллекта, служит источником многих новых идей и разработок, среди которых ЛИСП-машины 80-х годов. Синтаксис и семантика ЛИСПа построены на базе одного из языков математической логики, а именно Я-исчисления, предложенного Чёрчем в 30-е годы. Вспомним также, что конструкции повелительного языка нормальных алгорифмов, ориентированные на обработку строк символов, вошли во многие языки программирования.

Конечно, языковое влияние на вычислительную теорию и практику не ограничивается использованием конкретных языков. Более фундаментально влияние общих языковых концепций, общего представления о языке как о выразителе смысла — повествовательного, повелительного, вопросительного. Сама идея формальной семантики языка, без которой невозможно современное программирование, принадлежит математической логике. Приведем еще один пример такого концептуального влияния.

Как известно, в современных языках программирования повествовательные и повелительные компоненты тесно переплетены. Одной из основных идей, легших в основу языка ПРОЛОГ, является возможность двоякой интерпретации предложений вида «если *A, B,* С,...,то *X».* Одна

интерпретация — повествовательная: «из истинности *A, B, C...* следует истинность X». Другая — повелительная: «чтобы достичь цели *X,* сделай *A, B,* С...» А ведь идея повелительной интерпретации предложений логического языка — так называемая логика задач (А. Н. Колмогоров) — возникла еще в начале 30-х годов!

Но и влияние конкретных языков бывает весьма существенным. Так, при изучении языка нормальных алгорифмов еще в 50-е годы были открыты основные приемы структурного программирования, вошедшие в практику в 70-е годы.

Еще значительнее роль наиболее известного из языков математической логики — языка узкого исчисления предикатов. В 70-е годы программисты, работающие с большими объемами информации, поняли, что им необходим простой и универсальный язык, позволяющий формулировать утверждения о хранимой информации, и что такой язык может быть построен на основе языка исчисления предикатов. В настоящее время этот язык — под названием «реляционная модель данных» — составляет краеугольный камень быстро развивающейся и актуальной области — построения баз данных.

Применение исчисления предикатов в информатике не ограничивается базами данных. Отметим, например, что исследования по так называемому «естественному выводу», начатые еще в 30-е годы (Г. Генцен), стали основой для систем автоматического поиска вывода, применяемых в задачах искусственного интеллекта. С точки зрения названных задач перспективной кажется и разработка математических исчислений так называемой модальной логики, оперирующей, наряду с традиционными для классической логики оценками утверждений «истинно» и «ложно», также' и такими оценками, как «возможно», «необходимо», «правдоподобно», «произойдет в будущем» и т. д.

Наряду с формальными языками (повелительными, повествовательными, вопросительными) математическая логика поставляет для вычислительной теории и практики и вычислительные модели. Первые универсальные вычислительные устройства — машина Поста, машины Тьюринга—появились в 1936 г. не «в железе», а всего лишь на бумаге — в публикациях их создателей как идеализированные абстрактные схемы. Хотя эти схемы и не были предназначены для конкретной реализации, они и поныне служат мощным инструментом для исследования принципиальных возможностей реально существующих вычислительных устройств.

Таким образом, умозрительные конструкции и вычислительные реалии оказываются во взаимодействии более тесном, чем это можно было предполагать априори. Показательна в этом отношении история абстрактных вычислительных моделей, образованных семейством элементов с одной и той же структурой. Абстрактные вычислительные устройства, составляющие эту модель, впервые рассматривались в начале 50-х годов Дж. фон Нейманом. По-видимому, первоначальные мотивы их изучения носили в основном философский характер и связывались прежде всего с вопросами о возможности имитации искусственными устройствами самовоспроизведения, размножения и других функций живых организмов. Теория этих вычислительных моделей развивалась и время от времени привлекала внимание практиков. При этом название моделей много раз менялось: говорили об автоматах фон Неймана — Чёрча, клеточных автоматах, итеративных сетях, однородных средах и т. д.; сейчас по отношению к этому понятию наиболее часто употребляется термин «систолические структуры».

За последние годы прогресс в области элементной базы вычислитель-

ной техники привел к тому, что работы в указанной области приобрели практическую значимость. Это отразилось на росте публикаций по теории сверхбольших интегральных схем — СБИС, включающей и область рассматриваемых вычислительных моделей. В трудах основных международных конференций по теоретическим основам вычислительных наук такие публикации сейчас занимают 20—30% объема.

Конкретные алгоритмы и теоремы

Наконец, последняя по степени фундаментальности, но все же практически важная сфера влияния математической логики на вычислительную практику связана с конкретными алгоритмами (и лежащими в их основе теоремами). Дело в том, что в сфере интересов математической логики всегда находились выражения формальных языков и в какой-то степени произвольные зиакосочетания (цепочки символов, размеченные графы и тому подобные *конструктивные объекты)*.. Особое внимание, уделяемое математической логикой проблеме эффективности, алгоритмичности построений, привело к тому, что в ней накопился значительный запас алгоритмов для работы с объектами указанного вида. Как известно, расширение сферы использования вычислительной техники связано сейчас прежде всего с обработкой нечисловой информации, представляемой конструктивными объектами различных типов. Так, лингвистические алгоритмы синтаксического анализа в значительной мере используют алгоритмы, выработанные при анализе скобочных и графовых структур математической логики. Использование в вычислительном деле алгоритмов математической логики особенно наглядно видно на примере развития практических работ в области логического вывода и использования неклассических логик, о чем мы уже упоминали выше. Соответствующая теоретическая база давно и успешно строится математической логикой.

При том, что (как мы неоднократно подчеркивали) среди идей, методов и фактов математической логики конкретные т е о р е м ы занимают по значимости лишь третье место (деля его с конкретными алгоритмами), *их роль тоже бывает весьма важной (просто* роль концепций и языков еще важнее). Примером может служить известная *теорема о неподвижной точке* теории алгоритмов, называемая также *теоремой о рекурсии.* Использование ее в вычислительной практике (а именно, в программировании) дает пищу для поучительных размышлений, и мы остановимся на этой теме подробнее. Начнем с общего идейного фона.

Развитие программирования, в частности языков программирования, вначале происходило таким образом, что к текстам на этих языках сначала предъявлялись очень жесткие синтаксические правила и ограничения. Но со временем синтаксис этих языков становился все более неформальным и «человечным»: происходила и продолжается сейчас гуманизация взаимодействия человека и машины. Представление же о том, что, собственно, программа должна делать, если в ней удалось правильно расставить все точки с запятыми и скобки, с самого начала было абсолютно неформальным. Однако необходимость создания больших программ, работающих в соответствии с достаточно сложными спецификациями и не зависящих от типа используемой ЭВМ, привело к попыткам формализации интуитивных представлений о семантике конструкций программирования.

Таким образом, если развитие синтаксиса идет в направлении от формального к неформальному, семантика развивается или пытается развиваться в противоположном направлении — в сторону все большей фор-

мализации. Но создание достаточно простой и последовательной (иначе человек не смог бы ею пользоваться), строгой и непротиворечивой (иначе ее невозможно было бы реализовать в компиляторах и использовать в доказательствах правильности) семантики оказалось весьма сложным делом. Построить такую семантику для ряда языков программирования не удалось по сей день. С другой стороны, во многих случаях был достигнут существенный прогресс. Он основан на широком использовании конструкций, накопленных в математике, и, прежде всего в теории алгоритмов.

Один из важнейших примеров такого использования — построение формальной семантики для рекурсивных программ. В немашинной математике иногда встречаются примеры определения функции через саму себя (классический пример — факториал). Однако роль таких (они называются *рекурсивными)* определений намного важнее в программировании. Типичная ситуация, когда рекурсивные определения возникают, такова.

Мы определяем функцию, применяемую к объектам, которые могут иметь сложную структуру. При этом мы знаем, что дает эта функция в простейших случаях (то есть для простейших объектов). Кроме того, известно, как разбить произвольный объект на сравнительно крупные части — подобъекты — с тем, чтобы применить функцию к этим частям и из результатов применения получить результат действия функции на весь объект. Такое задание функции соответствует естественному для человеческой деятельности подходу — многоступенчатому разбиению задачи на подзадачи, сводящиеся в конце концов к задачам совсем простого вида. В вышеприведенном описании не содержится в явном виде никакого алгоритма вычисления определяемой функции: ведь результаты ее применения к подобъектам тоже не указаны (чтобы получить эти результаты, надо разбить Гфдобъекты на более мелкие подобъекты и так далее — пока не дойдем до простейших объектов). Тем не менее при получении такого описания в вычислительной машине автоматически формируется некоторый алгоритм вычисления. И хотя функция, задаваемая этим алгоритмом, соответствует семантике рекурсивного определения, сама эта семантика носит неалгоритмический характер, она отражает лишь те представления о функции, которыми программист пользовался при ее описании.

Основой для описанной неалгоритмической семантики рекурсивных определений как раз и служит вышеназванная теорема о неподвижной точке. Вообще теоремы о неподвижной точке играют ваяшую роль в различных областях математики. В теории алгоритмов они имеют вид теоремы о неподвижной точке вычислимого оператора, отображающего функции в функции. Теорема эта утверждает, что всякий оператор Имеет неподвижные точки (то есть функции, перерабатываемые этим оператором в себя), причем среди этих неподвижных точек-функций существует *наименьшая* (так что все другие неподвижные точки-функции служат ее продолжением на большие области определения). Таким образом, достаточно описать вычислимый оператор, чтобы получить описание и в конечном счете программу некоторой вычислимой функции (наименьшей неподвижной точки описанного оператора). Именно такое использование теоремы о неподвижной точке и легло в основу построения упоминавшегося уже языка ЛИСП (Дж. Маккарти), а также создания формальной семантики для многих конструкций более поздних языков программирования (Д. Скотт).

СССР традиционно занимает одно из ведущих мест в мире по исследованиям в области математической логики. Однако необходимо отметить

следующие два обстоятельства. Во-первых, проблема создания ЭВМ нового поколения требует интенсивного расширения исследований. Во-вторых, в некоторых областях, где вначале советские ученые занимали ведущее положение, затем произошло отставание. В целом численность специалистов, разрабатывающих проблематику математической логики в Академии наук СССР и университетах, за последние годы сократилась.

Рассмотрим ситуацию более детально. При этом мы не претендуем на полноту картины, а лишь хотим обратить внимание на существующие проблемы.

Логический вывод. В этом направлении в СССР есть серьезные достижения, принадлежащие прежде всего ленинградской группе математических логиков. Разработанный в этой группе обратный метод Маслова не менее эффективен, чем созданный на Западе метод резолюций, и был опубликован раньше него. Для зарубежных стран (прежде всего Японии) в последние годы характерен всплеск работ в этой области.

Формальные грамматики и языки. Уже на раннем этапе развития этого направления в СССР были достигнуты существенные успехи. Однако теперь уровень исследований отстает от зарубежного.

Сложность вычислений. Первые работы по сложности вычислений существенно опередили зарубежные. За последние годы в нашей стране получен ряд важных результатов. Однако масштаб отечественных разработок значительно отстает от ведущихся на Западе, где основные результаты получены группами специалистов по математической логике и теории алгоритмов, сотрудничающими с крупными фирмами.

Сложность {энтропия) конструктивных (конечных) объектов. Эта область пока имеет меньший удельный вес, чем уже перечисленные, но является весьма перспективной. Советские исследователи, которым принадлежат здесь первые работы, продолжают сохранять ведущие позиции.

Логические языки и их семантика. В этой обширной и разветвленной области советским исследователям принадлежит ряд важных достижений, в частности, полученных недавно. Отметим, что программные (динамические) логики, исследование которых было начато в СССР, сейчас наиболее интенсивно развиваются на Западе.

Итак, идейное влияние математической логики, в том числе теории алгоритмов, на формирование вычислительной теории и практики существенно уже сейчас, а в будущем будет возрастать. Поэтому представляется целесообразным расширение как фронта исследовательских работ в этой области, так и масштаба подготовки кадров; должна возрасти и роль .математической логики как компонента обучения на различных его этапах и при подготовке специалистов всех уровней. Затраты на соответствующие мероприятия ничтожны по сравнению с общими затратами на развитие вычислительного дела, а в том, что они многократно окупятся, нет сомнений.

УДК 51