ОТ ЛОГИЦИЗМА К ПРОЦЕДУРАЛИЗМУ
(На автобиографическом материале)

Г.С.Цейтин

Ленинград, СССР

Это рассказ о том, как я изменил свои взгляды, перейдя от убеждения, что хорошее знание всегда должно быть представлено в виде набора логических утверждений в рамках подходящей математической модели действительности, к моей нынешней точке зрения, что знание в основе является алгорифмическим. Необходимо объяснить, почему я решил разбираться в своем прошлом вместо того, чтобы дать систематическое изложение процедуралистской точки зрения. Первая причина в том, что в настоящее время я просто не в состоянии дать такое изложение, не считая нескольких общих положений (для настоящего изложения следовало бы показать все это в действии). Поэтому мне необходимо было найти точку отсчета, с которой можно было бы сопоставлять мои взгляды, и легче всего оказалось выбрать для критики мои собственные ошибки (все ссылки на мои статьи в этом тексте имеют целью иллюстрацию моих заблуждений, а не моих нынешних утверждений). Вторая причина - это то, что мои нынешние взгляды основаны на личном опыте и личной оценке его; быть может, не для каждого он будет убедительным, и я могу лишь показать, как он убедил меня.

Я начал (в начале пятидесятых годов) как чистый математик, с подсознательным убеждением, что математические объекты - это нечто реально существующее и поддающееся исследованию посредством рассуждений, что любой осмысленный вопрос о них имеет "объективный" ответ, который когда-нибудь будет обнаружен, и что, несмотря на теоремы о неполноте, недостающая информация может быть добыта из "действительности" каким-то, пока неясным, образом. (Разумеется, я знал, что математические понятия представляют собой абстракции от реального мира, но это не влияло на мой образ мыслей.) Я умел строить

алгорифмы (первоначально в виде нормальных алгорифмов Маркова), но рассматривал их как особый вид математических объектов, свойства которых следует доказывать посредством стандартных математических рассуждений (в стиле [I]), даже если они непосредственно очевидны. Я знал о критике А.А.Марковым классической теории множеств и принимал участие в его программе построения конструктивной математики, однако не принимал его философской установки и рассматривал свою работу в конструктивизме как своего рода упражнение в самоограничении.

Эта платонистическая позиция далее проявилась в моем подходе к прикладным задачам, который был довольно стандартным, несмотря на несколько необычную область - автоматический перевод с одного языка на другой. Конечным результатом прикладного исследования является обычно вычислительная процедура (в моем случае - алгорифм синтаксического анализа), однако обычно эмпирические процедуры не считаются хорошей формой представления знания. Предполагается, что надо разработать "математическую модель", то есть идеальную конструкцию, составленную из математических объектов и обладающую свойствами, приблизительно соответствующими свойствам реального объекта, а потом все практические процедуры надо получать математическим путем из этой модели. В задаче анализа естественного языка эмпирический подход к построению анализирующих алгорифмов быстро обнаружил свою бесполезность (по крайней мере - добавлю я теперь - при нашей тогдашней, примерно 1960 г., технике программирования, с очень малым опытом в модульности и жесткими ограничениями со стороны оборудования); и тогда наша группа разработала систему логического описания синтаксиса (грамматику зависимостей) и опубликовала статью [2] в поддержку неалгорифмического представления лингвистического знания.

Действительно, преимущества неалгорифмического знания очевидны. Одно уравнение, например, закон Ома (U=IR) заменяет несколько процедур: U:=I*R, I:=U/R и R:=U/I; более того, оно может объединяться с другими уравнениями, образуя описание сложной электрической схемы, после чего можно пользоваться известными математическими методами.

Таким образом получилось, что я с начала 1960-х годов

стал бороться с алгорифмическим представлением знания в разных областях своей работы. В теории алгорифмов и конструктивном анализе я придумал средства для замены прямого построения алгорифмов (в доказательствах существования) действиями над перечислимыми множествами [3,4], что оказалось короче и ближе к обычному математическому стилю. В программировании я осознал, что даже языки высокого уровня (тогда это называли автоматическим программированием) не освобождают программиста от некоторой рутинной работы по "алгорифмизации". Подлинно автоматическая система программирования - думал я - должна уметь строить программу (по крайней мере, ее очевидные части), исходя из какой-то другой формы постановки задачи, в е р о я т н о , из логического описания. Это привело меня к тому, что теперь называют верификацией программ [5,6]; я рассматривал это как шаг в направлении автоматической алгорифмизации. И даже в столь конкретной работе, как построение информационной системы для АСУ, я придумывал логический подход. Я предложил [7] систему аксиом для логической теории первой ступени, включавшую схему построения аксиом, названную принципом полноты; этот принцип утверждал, что любое изменение в системе в пределах заданного промежутка времени связано с одним из явно перечисленных событий.

Мой логицистский подход к прикладным задачам основывался на некотором скрытом предположении, которое заслуживает особого рассмотрения. Оно состояло в том, что исчисление предикатов является универсальным представлением для любых четких рассуждений, то есть предполагалось, что любые четкие рассуждения могут быть выражены в исчислении предикатов при надлежащем изменении обозначений. Некоторые, возможно, возразят, что это крайнее упрощение и что они всегда признавали такие вещи, как эвристику, правдоподобные рассуждения, размытые суждения, интуицию, озарение (или даже ясновидение). На такого рода вещи часто ссылаются в подтверждение ограниченности стандартного логического подхода и иногда на их основе делаются попытки преодоления этой ограниченности. Однако я подчеркиваю, что мое неявное предположение относилось не к такому роду мышления, а именно к тому, что мы обычно воспринимаем как четкое, детерминированное, логичное мышление. (Именно

поэтому я считал, что мышление, связанное с программированием, также основано на логическом выводе, что и привело меня к идее программирования на основе логических описаний.)

Сейчас я могу показать на очень простом примере, в чем была ошибка. Пусть у Пети есть два карандаша, а Аня дала ему еще три; сколько карандашей у Пети теперь? Ответ - пять, и это вполне точное и логичное заключение. Теперь добавим, что один карандаш Петя потерял, и ответ изменится. Может ли быть, чтобы результат, полученный посредством вывода в некоторой системе логики предикатов, стал неверным от добавления дополнительной посылки? Или же в первом случае нужно было отвечать "пять, при условии, что больше ничего не происходило с Петей и/или его карандашами"?

Конечно, это довольно грубый довод, который я не принял бы 15 лет назад. При логицистском подходе указанную трудность можно обойти при помощи принципа полноты или же минимального следования по Дж.Маккарти. Но в логике предикатов есть и более тонкое ограничение, связанное со способом использования имен (или переменных).

Имя рассматривается как особый объект, связанный постоянно или временно (если это переменная) с некоторой вещью, и естественная функция имени - это представлять эту вещь. Не разрешается рассматривать внешний вид имени или его состав (термы и выражения можно рассматривать как составные имена, но их структура описывает операции над денотатами, а не над самими именами); связывание конкретного имени с конкретным объектом является чисто случайным. Если два имени, определенные в разных областях, случайно выглядят одинаково, то благовоспитанный логик должен такого избегать или хотя бы не замечать.

Большинство современных языков программирования (но не ЛИСП) стоят на такой же позиции; в сообщении об Алголе 68 цитируется Шекспир: "Что в имени? То, что зовем мы розой, и под другим названием сохраняло б свой сладкий запах!". Принятое в Алголе правило идентификации для идентификаторов, не описанных в процедуре, где они используются, подражает правилу подстановки из исчисления предикатов (подставляя $f(x)$ вместо y в $\exists x P(x, y)$ получаем $\exists z P(z, f(x))$). Лишь на уров-

не метаязыка допускается рассматривать имя как самостоятельный объект, чем частично можно объяснить строгое разграничение, проводимое между метаязыком и языком (в программировании - между трансляцией и счетом). Немного позже мы увидим, в чем узость такого подхода.

Перемена в моих взглядах была постепенной, и ей способствовали несколько факторов: постепенно проявляющаяся внутренняя слабость концепции "математического мира"; слабые успехи в работах по автоматическому доказательству, автоматическому программированию и представлению семантики естественного языка, основанных на строго логическом подходе, в сопоставлении с более успешными эмпирическими работами в тех же направлениях; мой опыт построения проблемно-ориентированных языков для прикладных задач. Возражений против описательного представления знаний как такового не возникало. Прежде мне приходилось придерживаться такого представления, поскольку только такая форма знания хорошо описывалась при логицистском подходе. Теперь же я могу ее рассматривать как один из возможных типов знания; притом такое знание не может действовать самостоятельно, без помощи процедурного знания. Самое общее и красивое уравнение бесполезно, если нет алгорифма для его решения. (По аналогии с известным принципом Вирхова относительно живых клеток можно сказать, что алгорифмы получаются только из алгорифмов). Итак, переход к процедурализму состоял из трех шагов: снятия ограничений, налагавшихся логицизмом, расширения средств процедурного представления знания и осознания всеобщего характера процедурного знания.

Моя вера в математический мир была серьезно подорвана затруднениями в основаниях математики, положением, при котором ряд вопросов, выглядящих вполне осмысленными (ввиду синтаксической правильности), не мог получить осмысленного ответа. Можно было вполовину соглашаться с объяснениями вроде того, что "множество всех ординалов не существует, а всех вещественных чисел - существует", однако доказательство П.Коэном независимости континуум-гипотезы показало, что дела обстоят гораздо хуже. Обратим внимание на нарастание неполноты в математике: Н.И.Лобачевский, Я.Бояи - в абсолютной геометрии не хватает информации, чтобы решить, какая из д в у х

геометрий верна; К.Гедель - множество теорем, доказуемых в некоторой теории, рекурсивно перечислимо и поэтому не может содержать всех сведений об элементах какого-нибудь неперечислимого множества; П.Коэн - никакое с ч е т н о е множество утверждений не может приблизиться к описанию далеко не счетного множества объектов. Мне пришло в голову, что в математических вопросах, возможно, не больше смысла, чем в вопросах о героях какого-нибудь романа. Чем же в таком случае занимается математика? Конструктивная математика не казалась мне выходом из положения: вводя более тонкие различия между суждениями, она увеличивала количество как будто бы осмысленных вопросов, давая меньше возможностей для ответа на них. Однажды мне пришлось выступать на тему об основаниях математики перед физиками, и мне удалось довести до них свое недоумение относительно предмета математики. Они ответили сочувствием: это означает, сказали они, что математика выходит на уровень физики.

Другой областью, где мои ожидания не оправдались, было применение математической логики к обычному мышлению. В работе над естественным языком я руководствовался представлением о языке как о некотором сложном исчислении, в котором синтаксис определяется некоторой порождающей грамматикой, а семантика выражает смысл каждого порожденного объекта через смыслы его составляющих. Было, однако же, неясно, посредством каких объектов можно представлять эти смыслы.

В конечном счете смысл следовало бы представлять через поведение, но, поскольку такая перспектива была слишком отдаленной, возникла мысль об использовании языка типа логического, для которого поведение (доказательство теорем) формально описано. Таким образом, нужно было строить эквиваленты в логике предикатов для предложений, а также для их составных частей [8].

Этот план терпел неудачи с самого начала, но я потратил больше десяти лет, чтобы понять это. Прежде всего, пришлось ограничиться математическими текстами, ввиду примеров типа "он знает, что...", где подстановка вместо многоточия эквивалентных суждений может привести к неэквивалентным результатам (таким образом, то, что "он" знает, - это т е к с т , а не

суждение; карнаповская теория интенсионалов, основанная на модальности, меня не убедила). Далее, даже в самых однообразных математических текстах, какие я рассматривал, лишь небольшая часть утверждений допускала полноценный перевод на язык исчисления предикатов. Значительная часть текста содержала, явно или неявно, информацию о структуре доказательства и т.п. (что также можно рассматривать как некоторую процедурную информацию).

Для существительных естественным логическим эквивалентом представлялись индивидные переменные (с областью значений, определяемой конкретным существительным). Я пошел в этом направлении дальше и для представления смысла сложных именных групп придумал особый тип подчиненных переменных, у которых область значений зависела от текущих значений других переменных [9]. Получилось очень громоздко. Через несколько лет обнаружилось более простое и общее решение, в котором для представления составляющих, отличных от целого предложения, использовались особые символы типа кванторов, не имеющие собственного смысла. Одновременно группа лингвистов, не интересовавшихся логикой предикатов, смогла дать [10] формальное описание целого ряда семантических эквивалентностей, которые, как я считал, следовало выводить из некоторого еще не известного логического представления. Были и другие примеры успешной семантической обработки фрагментов естественного языка, например, в Коболе или во взаимодействии с базами данных, но логик отбросил бы их с пренебрежением [11] как частные случаи, не указывающие пути к общему решению. (Теперь я считаю, что "общего решения" для естественного языка не может быть, потому что это объединение большого числа относительно независимых систем, использующих общие механизмы нижнего уровня, а не единая заранее спланированная система).

Что касается логической верификации программ, то оказалось, что описывать задание на языке исчисления предикатов не легче, чем просто писать программы. Сперва я думал, что нужен лишь более удобный язык описания, с добавлением "синтаксического сахара" (см. [6]), но и в этом направлении мне не удалось существенно продвинуться. С другой стороны,

Э.Х.Тыугу и другие, вовсе не математики, нашли очень полезный подход [I2] к автоматическому программированию, основанный на соединении заранее подготовленных процедурных "вычислительных моделей". Я был разочарован, когда познакомился с ним, потому что у них новые программы строились из других программ.

В автоматическом доказательстве теорем не было найдено эффективного общего метода, и внимание переместилось на человеко-машинные системы доказательства с тем, чтобы дать возможность человеку внести некий таинственный элемент ("интуицию"), которого не доставало автоматическим системам. (Теперь я считаю, что отсутствовавшим элементом была процедурность, а не какое-то там ясновидение). Существовали более удачные работы по доказательству теорем для ограниченных областей (например, формульные преобразования) или в системах, основанных непосредственно на представлении математических утверждений на естественном языке (с ограниченным набором правил вывода и прямыми указаниями на их использование, которые встречаются в тексте на естественном языке, но теряются в логическом представлении).

В программировании моя работа обычно проходила в форме создания и реализации различных проблемно-ориентированных языков. Таким образом, я имел возможность определять программистские конструкции, более близкие к способу мышления (и выражения) в данной прикладной области, чем к традиционным конструкциям программирования (эта работа в некоторых частях, например, в сложной системе сравнения с образцом оказалась аналогичной разработкам в языках искусственного интеллекта, о которых я узнал гораздо позже). Эта работа показала мне глубокое родство между естественными языками и языками программирования и привела меня к использованию программистских конструкций, наряду с логическими, для представления значения конструкций естественного языка (в самом начале у меня была идея представлять смысл предложения булевской процедурой с боковым эффектом, что могло бы смоделировать использование местоимений; однако это не получилось). Именно в этой области мне встретился (около 1973 г.) пример, сыгравший решающую роль в моем отказе от логицизма.

Мне нужно было построить язык моделирования для некоторо-

го класса экологических систем, включавших несколько популяций рыб (каждая из нескольких возрастных групп), их рост, размножение, питание, хищничество и т.п. Каждая возрастная группа каждой популяции имела по нескольку числовых характеристик, для которых нужно было отводить ячейки памяти. Эти ячейки были единственными объектами, осмысленными с машинной точки зрения, но не с пользовательской. Пользователь мог вообще не знать о какой-нибудь промежуточной величине, используемой в моделировании. Для пользователя имели значение популяции рыб и некоторые явно или неявно задаваемые "законы", например, "количество мальков равно численности, умноженной на плодовитость", или "плодовитость некоторой рыбы выражается такой-то функцией от ее веса" и т.п. Существенно, что закон задается соотношением между определенными характеристиками популяции безотносительно к тому, какие еще у нее есть характеристики. Единственным традиционным способом представления законов было введение для всех популяций универсальной структуры с полным набором характеристик, невзирая на то, что для конкретной популяции и возрастной группы потребуется, возможно, лишь малая часть их. Мне не хотелось делать так, потому что я стремился к системе, открытой для новых законов и характеристик. Примерно после года колебаний я пришел к решению, весьма для меня необычному.

Представлением популяции было просто имя (последовательность букв), которое могло соединяться с и м е н е м характеристики и номером возрастной группы, давая обозначение ячейки памяти для хранения соответствующей величины. Закон применялся явным образом к определенному имени популяции, но имена участвующих в нем характеристик определялись внутри закона. Каждый раз, когда синтезировалось новое обозначение величины, заводилась новая ячейка памяти (для фазы счета), а то, что применения разных законов относятся к одной и той же величине, устанавливалось по совпадению обозначений, а н е н а о б о р о т .

Я был поражен тем, что для получения осмысленного результата нужно было, оказывается, не держаться за смысл имен, а рассматривать их как бессмысленные последовательности букв, и тем, что я пришел к результату, связанному с "поведением",

непосредственно от языка, без какого-либо промежуточного "семантического" представления. Для меня это выглядело так, что на пути от текста к поведению языковой знак, пройдя нетронутым через синтаксические преобразования, вдруг исчезает, но в последний момент ненадолго проявляет себя как физический объект. Я нашел и другие примеры такого рода (например, складывая числа, записанные в десятичной системе, мы действуем с их цифрами). Я сопоставил это также с анализом смысла прилагательных в естественном языке (п р а в и л ь н а я пирамида не обязательно п р а в и л ь н ы й многогранник; х о р о ш и й математик может не быть х о р о ш и м лектором); здесь смысл прилагательного зависит не от определяемого о б ъ е к т а , а от называющего его с л о в а . Теперь представали в новом свете и некоторые другие лингвистические примеры, с которыми я встречался ранее. И это означало, что я дошел до точки, где обычная математическая абстракция теряет силу и уже бесполезно говорить об абстрактном объекте, как будто бы он существует на самом деле, а надо рассматривать его мысленное символическое представление.

Я стал рассматривать объект (точнее, его мысленное представление) как набор именованных атрибутов, значение которых можно выбирать или изменять, указывая имя, аналогично наборам данных и каталогизированным процедурам в операционной системе OS/360 IBM. Таким образом появляется возможность задавать умолчания и отменять их. Разумеется, все это не соответствует логическому образу мыслей: если считать атрибутами прямоугольника основание и высоту, то почему не включить еще и диагональ или площадь? А если включить, то как их изменять?

Я думаю теперь, что функция имени (существительного) в языке состоит не в том, чтобы указывать на определенный объект или класс объектов, а в том, чтобы служить в различных контекстах или ситуациях для указания на предмет, имеющий определенную функцию (в некотором не вполне точном смысле). И когда имя использовано таким образом, оно может указывать на следующую структуру имен и атрибутов. Я считаю это альтернативой к теории экстенсионалов и интенсионалов Р.Карнапа.

В системе моделирования для рыб законы не были процедурами. Это были статические схемы (вроде макросов), используемые

для построения списка "заказов", которые потом упорядочивались и исполнялись в стандартном порядке. Но потом я стал строить процедурные системы, где выборка по имени может производиться в процедурах и, более того, выбранное значение может снова быть процедурой. Я оценил свободу и гибкость такого подхода (это, вероятно, знакомо пользователям ЛИСПа). Для некоторых практических целей процедуры можно рассматривать как непосредственное представление смысла.

Дополнительным возможностям процедурного представления знания я научился из работ по искусственному интеллекту, в особенности, из работ К.Хьюитта, Т.Винограда и, конечно, из теории рамок М.Минского. И это завершило мой переход к процедурализму (примерно, 1976 г.).

Остается ответить на вопрос, почему алгорифмы не признавались ранее в качестве подходящей формы знания. Причина в том, что математическое понятие алгорифма слишком грубо для целей представления знания. Обычно бывает известен не математический алгорифм, а неформальный метод. В чем же существенное различие между ними и что надо добавить к нашим средствам программирования, чтобы справиться с этим различием? Наиболее очевидное различие связано с недетерминированностью, и действительно, очень легко расширить понятие алгорифма в этом направлении.

Часто указывают также на то, что неформальный метод может ссылаться на подцели, не определяя способа их достижения. Однако математическое понятие алгорифма тоже отсылает к неопределенным подцелям (попробуйте приписать к слову букву А, если у вас кончились чернила и т.п.). И если бы ссылки на подцели были главным различием между формальными и неформальными методами, было бы легко воспользоваться понятием относительной рекурсивности и определить неформальный метод как схему сведения цели к подцелям.

Я предлагаю другую интерпретацию этого различия. Алгорифм в математическом смысле полностью замкнут в себе и после того, как заданы исходные данные, не использует никакой другой информации. Напротив, реальная процедура (и, в определенной степени, современная машинная программа) может извлекать информацию из окружения, причем з а р а н е е н е н у ж-

но определять, каким именно образом. Когда мы говорим, что процедура определяет подцель, это значит, что она пытается извлечь способ ее достижения неважно откуда, то есть либо из собственной памяти, либо из окружения, где процедура вызвана. Такое извлечение может иметь вид выборки по имени (намеренно не говорю здесь о вызове по образцу). В других терминах такая организация процедур может быть определена как модульность: каждый раз определяется или изменяется только один модуль, а все остальное относится к окружению.

Есть, видимо, и третье отличие неформальных методов от алгорифмов. Оно состоит в использовании какого-то рода распознавания образов (возможно, поиска рамки, но не классического сравнения с образцом)для определения ситуации и последующего выбора действия. Вероятно, именно здесь в общую систему включается обычное описательное знание.

В этой картине процедурной организации знания описательное знание не утрачивает своей ценности. Однако нужно помнить, что оно надстраивается над сложной процедурной системой и что надо считать удачей такое положение, при котором знание получается в столь общей и сильной форме. Иногда я даже удивляюсь, как к этому удалось прийти.

Литература

I. А.А.Марков. Теория алгорифмов. Тр. Матем. ин-та им. В.А.Стеклова, 1954, т. XLII.

2. Б.М.Лейкина, Т.Н.Никитина, М.И.Откупщикова, С.Я.Фитиалов, Г.С.Цейтин. Система автоматического перевода, разрабатываемая в группе математической лингвистики ВЦ ЛГУ. Научно-техническая информация, 1966, № 1, с. 40-50, № 4, с. 31.

3. Г.С.Цейтин. Один способ изложения теории алгорифмов и перечислимых множеств. Тр. Матем. ин-та им. В.А.Стеклова, 1964, т. LXXII, с. 69-98.

4. Г.С.Цейтин. О верхних границах перечислимых множеств конструктивных вещественных чисел. Тр. Матем. ин-та им.

В.А.Стеклова, 1970, т. СХШ, с. 102-172.
5. Г.С.Цейтин. О логическом подходе к автоматизации программирования. Всесоюз. конф. по проблемам теоретической кибернетики 9-13 июня 1969 г. Тез. докл. Новосибирск, 1969, с. 5-6.
6. Г.С.Цейтин. Некоторые черты языка для системы программирования, проверяющей доказательства. Теория программирования. Тр. симпозиума. Новосибирск, 1972, ч. П, с.234-249.
G.S.Tseytin. Some features of a language for a proof-checking programming system. International Symposium on Theoretical Programming. Lecture Notes in Computer Science, 5. Springer Verlag, Berlin-Heidelberg-N.Y., 1974, p. 394-407.
7. Г.С.Цейтин. Логико-математический подход к построению экономико-информационной системы. Методы вычислений. Изд-во ЛГУ, 1970, вып. 6, с. 107-127.
8. Г.С.Цейтин. Язык математической логики как средство исследования семантики естественного языка. Проблемы прикладной лингвистики. Тез. межвуз. конф. 16-19 декабря. - М.,МГПИИЯ, 1969, ч.П, с. 326-335.
9. Г.С.Цейтин. О промежуточном этапе при переводе с естественного языка на язык исчисления предикатов. Тез. докл. конф. по обработке информации, машинному переводу и автоматическому чтению текста, - М., ВИНИТИ, 1961, с. 107-111.
10. А.К.Жолковский, И.А.Мельчук. О семантическом синтезе. Проблемы кибернетики. - М., Наука, 1967, вып. 19, с. 177-238.
11. G.S.Tseytin. Features of natural languages in programming languages. Proc. of the Fourth International Congress for Logic, Methodology and Philosophy of Science, Bucharest, 1971. North-Holland, Amsterdam-London, 1973, p. 215-222.
Г.С. Цейтин. Черты естественных языков в языках программирования. Машинный перевод и прикладная лингвистика. - М., МГПИИЯ, 1974, вып. 17, с. 134-143.
12. К.А.Тинн, Э.Х.Тыугу, М.И.Унт. Система модульного программирования для ЦВМ Минск-22. ВКП-2.Тр.Всесоюз.конф. по программированию. Новосибирск, 1970, с. 23-39.