В. Ф. ТУРЧИН

МЕТААЛГОРИТМИЧЕСКИЙ ЯЗЫК

УДК 51:681.142.001

Формализованный язык, описываемый в настоящей работе, был первоначально задуман как метаязык для формального описания семантики алгоритмических языков и поэтому получил название *метаалгоритмического*. Первый вариант метаалгоритмического языка описан в [1]. В дальнейшем язык был значительно усовершенствован, хотя основные идеи остались прежними. В статье дано формальное описание метаалгоритмического языка в его окончательном виде; необходимо, однако, вначале коснуться целей создания этого языка и наметить пути их достижения.

ОБЩЕНИЕ ЧЕЛОВЕКА С МАШИНОЙ И ФОРМАЛИЗАЦИЯ СЕМАНТИКИ

Метаязык для описания семантики алгоритмических языков является алгоритмическим языком [1]. На любом универсальном алгоритмическом языке можно описать семантику любого другого алгоритмического языка, поэтому выбор того или иного алгоритмического языка в качестве метаязыка — в принципе, вопрос удобства (хотя на практике «неудобство» часто означает «невозможность»). Так как речь идет о формальных семантических описаниях, предназначенных для использования человеком, выбранный нами алгоритмический метаязык должен быть удобен для человека. Но поскольку алгоритм — в сущности, инструкция для машины, проблема метаязыка — это проблема общения человека с машиной, удобного не для машины, а для человека.

Машинно-независимые алгоритмические языки, широко применяемые в современном программировании (ФОРТАН, АЛГОЛ, КОБОЛ и т. п.), удобны для записи задач из определенной области вследствие того, что они строятся на основе формализации ряда понятий, важных и характерных для данной специальной области. Нам же нужен язык, ориентированный не на какие-либо конкретные понятия, а предназначенный для описания любых языков и понятий (метаязык), поэтому такой язык будет удобен для человека лишь в том случае, если он схватит какие-то чрезвычайно общие и в то же время важные черты человеческого мышления.

В поисках таких черт обратимся к естественным языкам и их продолжению — формализованным языкам математики. Важнейшей чертой этих языков является наличие в них *иерархии понятий*, возникающей благодаря способности мозга путем абстрагирования от многих конкретных ситуаций создавать языковые модели действительности, которые затем используются путем их конкретизации применительно к любой данной ситуации. Естественный язык можно представить себе в виде многоэтажной пирамиды, построенной на почве чувственного опыта. Элементами этой пирамиды при нашем (семантическом) подходе надо считать *морфемы* — минимальные смысловые единицы языка. Складываясь в цепочки, морфемы образуют *языковые объекты* (*лингемы*): слова, группы слов, предложения. Языковые объекты, расположенные на самых низких этажах пирамиды, фиксируют наиболее конкретные, близкие к чувственному опыту понятия: «больно», «холодно», «заяц», «камень» и т. п., на основе которых строятся более абстрактные и более сложные понятия, а на их основе — еще более сложные и т. д.; все эти понятия фиксируются языковыми объектами. Где-то на средних этажах пирамиды расположены понятия «север», «число», «работа», «чин полковника», а где-то на самом верху — «отчуждение», «гомозиготный», «бикомпактность». В естественный язык нельзя ввести точную меру абстрактности или сложности понятия и распределить соответствующие языковые элементы по этажам, поэтому нашу пирамиду следует понимать условно и иллюстративно. Однако принцип образования сложных и абстрактных понятий путем композиции и абстрагирования от более простых и конкретных, несомненно, лежит в основе построения языков.

Возьмем теперь какой-нибудь языковый объект, например слово, и зададим вопрос: что значит понимать это слово? Очевидно, что физический носитель слова не имеет никакого значения, имеют значение лишь связи этого слова с другими словами — «комплексами ощущений», — а в конечном счете с чувственным опытом. Следовательно, понимать слово — значит уметь пройти в обратном направлении путь его построения. Понимать абстрактное понятие — значит уметь его конкретизировать в каждой заданной ситуации, понимать сложное понятие — значит уметь

свести его к ряду более простых. И то, и другое означает замену языкового объекта, занимающего более высокое положение в языковой пирамиде, на ряд объектов, занимающих более низкое положение. Эту операцию будем называть *конкретизацией* языкового объекта. При некоторых видах деятельности мы не доводим ее до комплексов ощущений, однако предполагается, что знаем, как это сделать, иначе слово не имеет для нас никакого смысла. Итак, семантика языкового объекта определяется *правилом его конкретизации*, а семантика языка в целом — *совокупностью правил конкретизации*, которая позволяет путем ряда шагов свести каждый языковый объект к некоторым несводимым элементарным объектам («комплексам ощущений»).

Нарисованная схема определения семантики объектов естественного языка является, без сомнения, очень упрощенной, однако без упрощения не возможна никакая формализация, а ведь наша задача и заключается в формализации семантических описаний. Поэтому при построении метаалгоритмического языка эта схема была принята за основу. Прежде чем приступать к формальному описанию метаалгоритмического языка, опишем кратко и неформально его основные черты.

Для указания того, что данный объект подлежит конкретизации, в метаалгоритмическом языке вводится *символ конкретизации* Ж. Например, если x — некоторая (алгебраическая) переменная, то ее значение будет обозначаться через Жx («конкретизация x») так, что если значение x есть 7, то рано или поздно Жx будет заменено на 7. Для придания необходимой структуры выражениям, в частности для выделения выражения, относящегося к данному символу конкретизации, служат *круглые скобки*. Выражения, заключенные в круглые скобки, и еще некоторые элементарные выражения называются *термами*. Правила конкретизации записываются в виде *предложений*, которые отделяются друг от друга знаком параграфа §, стоящего в начале каждого предложения. Предложение имеет *левую часть* — терм «высокого уровня» и *правую часть* — эквивалентное выражение «более низкого уровня». Например, правило конкретизации, гласящее, что значение x есть 7, записывается в виде предложения §Жx∼ ∼ 7, где символ *подстановки* ∼ означает, что Жx надо заменить на 7. В этом предложении Жx — левая часть, а ∼ 7 — правая часть (правая часть, однако, не всегда начинается с символа ∼). Для записи неполностью определенных выражений в предложениях служат *свободные переменные*, например **S**1, **W**x, **E**a (читается: «символ 1, терм x, выражение a»).

Во всех естественных языках существует возможность введения новых терминов и переопределения старых. Ясно, что необходимо предусмотреть это и в метаалгоритмическом языке, тем более, что иначе при принятом нами способе записи значения переменной с помощью правила конкретизации (а этот способ очень удобен) мы не смогли бы даже присвоить переменной новое значение. В метаалгоритмическом языке допускается занесение в память нового предложения; для этого служит операция, обозначаемая символом →. Обращаясь к будущим предложениям, необходимо отличать входящие в них символы §, Ж и др. от «настоящих» §, Ж и др. Поэтому предложения, рассматриваемые как *выражения*, записываются в специальном, преобразованном виде (μ-преобразование), которое осуществляется с использованием символа ′ (штрих).

Семантика самого метаалгоритмического языка определяется метаалгоритмической машиной, которая выполняет конкретизацию выражения в соответствии с имеющимся набором предложений. Следовательно, работа этой машины моделирует языковую деятельность человека, использующего иерархическую систему понятий. Таким образом, метаалгоритмическая машина «умнее» (и, конечно, сложнее), чем машина Тьюринга, которая моделирует только сам факт языковой деятельности (так сказать, «языковость» деятельности). Ее можно рассматривать как следующий шаг на пути приближения машины к человеку, поэтому и общение с такой машиной протекает более «человеческим» способом и удобнее для человека, чем общение с машиной Тьюринга. Алгоритм (набор предложений) на метаалгоритмическом языке может быть построен как иерархия понятий, все более сложных и специальных.

Поскольку использование иерархии понятий является необходимым (быть может, основным) элементом человеческого мышления, метаалгоритмический язык, который позволяет формализованное рассмотрение иерархий понятий, может оказаться полезным для изучения и моделирования мыслительных процессов. Общение человека с машиной и моделирование человеческого мышления — это, по существу, две стороны одной и той же проблемы.

ФОРМАЛЬНОЕ ОПИСАНИЕ МЕТААЛГОРИТМИЧЕСКОГО ЯЗЫКА

(При чтении этого раздела рекомендуется обращаться к следующему разделу за примерами и пояснениями.)

1.1. Описать язык — значит определить некоторое множество объектов, которые будем называть *лингемами*, и некоторые действия, связанные с этими объектами. Первая часть описания носит название *синтаксиса*, вторая — *семантики*.

1.2. Метаалгоритмический язык является *одномерным знаковым языком*, т. е. существует конечное число нерасчленимых знаков и всякая лингема является *цепочкой знаков*, т. е. конечной последовательностью элементов, каждый из которых отождествляется с одним из знаков[1].

Введем ряд синтаксических понятий, которые понадобятся нам как для определения лингемы, так и для описания семантики.

1.3. В основе синтаксиса одномерных знаковых языков лежит данное выше понятие цепочки знаков (или просто *цепочки*) и понятие *соединения* нескольких цепочек, обозначающее цепочку, полученную последовательным приписыванием цепочек друг к другу в том порядке, в каком они следуют за словом «соединение». Цепочку, не содержащую ни одного знака, будем называть *пустой*.

Значительной части синтаксических понятий могут быть даны простые рекурсивные определения; их удобно описать с помощью нормальной формы Бэкуса [3]. В определениях 2.1—2.14 (кроме 2.3), данных в этой форме, слова в кавычках обозначают синтаксические понятия, которые будем использовать в дальнейшем как обычные русские слова (без кавычек); знак : : = читается «есть по определению»; черта | (читается «или») отделяет различные частные случаи определения; последовательность элементов языка и синтаксических понятий означает соединение соответствующих цепочек; при этом предполагается, что ни кавычки « », ни знаки : : = и |, не входят в число знаков метаалгоритмического языка.

2.1 «*знак*» : : = «*собственный знак*» | «*несобственный знак*»
2.2 «*собственный знак*» : : = < | > | (|) | § | Ж | **S** | **W** | **E** | ′ | ~ | → | ←
2.3 *Несобственным знаком* может быть любой знак, отличный от *собственного знака*.
2.4 «*символ*» : : = «*значащий символ*» |
«*управляющий символ*» | «*скобка*»
2.5 «*значащий символ*» : : = ′ | ~ | → | ← | «*идентификатор*»
2.6 «*идентификатор*» : : = «*несобственный знак*» |
⟨«*несобственная цепочка*»⟩
2.7 «*несобственная цепочка*» : : = «*несобственный знак*» |
«*несобственная цепочка*» «*несобственный знак*»
2.8 «*управляющий символ*» : : = § | Ж, «*указатель переменной*»
2.9 «*указатель переменной*» : : = **S** | **W** | **E**
2.10 «*скобка*» : : = (|)
2.11 «*терм*» : : = «*значащий символ*» |
«*свободная переменная*» | «*пассивная пара*» |
Ж «*терм*» | («*выражение*»)
2.12 «*свободная переменная*» : : =
«*указатель переменной*» «*идентификатор*»
2.13 «*пассивная пара*» : : = «*указатель переменной*»′
2.14 «*выражение*» : : = «*пустая цепочка*» | «*выражение*» «*терм*»

Для краткого обозначения отдельных объектов метаалгоритмического языка будем использовать заглавные курсивные латинские буквы, снабженные, возможно, индексами.

3.1. Управляющий символ в некотором выражении называется *пассивным*, если за ним следует штрих′; в противном случае он называется *активным*.

3.2. Выражение (в частности, терм) называется *активным*, если оно содержит хотя бы один активный символ конкретизации Ж; в противном случае называется *неактивным*.

[1] Обстоятельное обсуждение основных понятий знаковых языков содержится в «Теории алгорифмов» А. А. Маркова [2], где термины *буква* и *слово* имеют то же значение, что *знак* и *цепочка* в нашей работе.

3.3. *Предложением* называется цепочка вида § K W E, где K — несобственная цепочка, W — неактивный терм, а E — выражение, не содержащее таких свободных переменных, которые не входят в W. Цепочка K носит название *комментария*, терм W — *левой части* предложения, выражение E — его *правой части*.

3.4. Соединение некоторого числа (в частности, нуля или единицы) предложений называется *набором предложений*.

3.5. Цепочка знаков называется *лингемой*, если она является выражением или набором предложений.

4.1. Выражение (в частности, терм) называется *полностью определенным*, если оно не содержит свободных переменных. Желая подчеркнуть, что некоторое выражение, быть может, содержит свободные переменные, будем называть его *общим*.

4.2. Свободные переменные вида SI, WI и EI, где I — идентификатор, назовем соответственно свободными переменными *символа*, *терма* и *выражения*. (Свободную переменную символа следовало бы назвать свободной переменной *значащего* символа; для удобства этот термин сокращаем.)

4.3. Будем говорить, что полностью определенное выражение E_d может быть отождествлено как общее выражение E_q, если свободные переменные символов, термов и выражений, входящие в E_q, можно заменить такими значащими символами, термами и выражениями соответственно, что в результате выражение E_q совпадает с E_d. Эти символы, термы и выражения будем называть *значениями* соответствующих свободных переменных. Если свободная переменная входит в E_q более одного раза, все ее вхождения должны быть заменены одним и тем же значением.

4.4. При наличии в общем выражении E_q более одной свободной переменной выражения может оказаться, что существует несколько вариантов придания значений свободным переменным, которые приводят к отождествлению. Чтобы значения свободных переменных были определены однозначно, условимся, что из всех вариантов выбирается тот, в котором самая левая по своему положению в E_q свободная переменная выражения принимает значение, содержащее минимальное число символов. Если это не устраняет неоднозначности, то такой же отбор производится по второй слева свободной переменной выражения и т. д. Это означает, что при попытке отождествления E_d как E_q оба выражения просматриваются слева направо и каждая свободная переменная EK_k[1] в E_q принимает сначала пустое значение, которое затем может удлиняться,— но лишь после того, как станет точно известно, что при данном значении EI_k отождествление невозможно, какие бы значения ни принимали свободные переменные EI_{k+1}, EI_{k+2}, . . . , впервые появляющиеся в E_q правее, чем EI_k.

4.5. Пункты 4.3 и 4.4. определяют *алгоритм синтаксического отождествления*, который применим всякий раз, когда задано некоторое полностью определенное выражение E_d и некоторое общее выражение E_g, и результатом применения которого является, во-первых, заключение о возможности или невозможности отождествления E_d как (частного случая) E_q и, во-вторых, если отождествление возможно,— то сопоставление каждой свободной переменной из E_q некоторого значащего символа, терма или выражения (соответственно типу свободной переменной), называемого значением этой свободной переменной.

4.6. Предложение P будем называть *подходящим* для терма W, если терм W может быть отождествлен как левая часть P.

5.1. Описание тех действий, связанных с лингемами языка, которые не выходят за пределы языка, т. е. сводятся к некоторым преобразованиям лингем, зависящим только от информации, содержащейся в лингемах, назовем *формальной* (или *внутренней*) семантикой языка. Описание действий, связанных с лингемами и выходящих за пределы языка, назовем *интерпретацией* (или *внешней семантикой*) языка. Настоящее описание затрагивает лишь формальную семантику. Интерпретация может быть самой различной.

5.2. Формальная семантика метаалгоритмического языка включает единственное фундаментальное преобразование лингем: *конкретизацию* данного выражения E *в соответствии* с данным набором предложений R, выполнение которого дает новое (вообще говоря) выражение E_1 и новый (вообще говоря) набор предложений R_1. Определим это преобразование с помощью описания выполняющей его *метаалгоритмической машины*. Синтаксис и формальная семантика метаалгоритмического языка образуют математическую *формальную систему* в обычном смысле слова.

6.1. Метаалгоритмическая машина имеет *управляющее устройство* и *запоминающее устройство* (*память*). Запоминающее устройство метаал-

[1] Через I_k обозначим некоторый идентификатор.

алгоритмической машины состоит из: а) бесконечной последовательности *рабочих полей*, которые нумеруются натуральными числами, начиная с нуля; б) такой же последовательности *полей памяти*; в) *указателя поля зрения*.

6.2. Номер n рабочего поля или поля памяти в своей последовательности назовем *уровнем* поля. Рабочее поле и поле памяти одного уровня будем называть *соответствующими* друг другу, поля нулевого уровня — *основными*.

6.3. Рабочее поле — это запоминающее устройство неограниченной емкости, которое в каждый момент времени содержит некоторое полностью определенное выражение. Поле памяти — это запоминающее устройство неограниченной емкости, которое в каждый момент времени содержит некоторый набор предложений. Поля, содержащие пустые цепочки, будем называть пустыми. Указатель поля зрения в каждый момент времени содержит натуральное число, которое может быть сколь угодно большим.

6.4. В каждый момент времени одно из рабочих полей является выделенным и носит название *поля зрения*. Уровень поля зрения меняется в процессе работы, его текущее значение будем обозначать через n_t; оно содержится в указателе поля зрения.

Метаалгоритмическая машина работает таким образом, что на всех уровнях $n < n_t$ рабочие поля не пусты, а на всех уровнях $n > n_t$ пусты как рабочие поля, так и поля памяти. Само поле зрения может, вообще говоря, оказаться пустым (в этом случае, как увидим ниже, произойдет немедленная остановка машины).

6.5. Набор предложений, полученный соединением всех наборов предложений, хранящихся в полях памяти в порядке возрастания уровня, называется *совокупным набором предложений*.

7.1. Работа метаалгоритмической машины складывается из ряда элементарных конкретизаций, называемых *шагами*, которые выполняются управляющим устройством, основываясь на информации, содержащейся в запоминающем устройстве. Управляющее устройство в процессе выполнения шага может проходить через много различных внутренних состояний, но после конца каждого шага оно может оказаться лишь в одном из следующих трех состояний: *рабочее* состояние, при котором машина приступает к выполнению следующего шага, *аварийное* состояние (*аварийная остановка*), которое интерпретируется как наличие ошибки в исходных данных, и, наконец, состояние *нормальной остановки*. Алгоритм выполнения шага всегда однозначен. Таким образом, если после конца каждого шага будем отмечать содержание запоминающего устройства (состояние памяти), то получим последовательность состояний машины, в которой каждое следующее состояние однозначно определяется предыдущим.

7.2. Выполнение шага начинается с исследования поля зрения в поисках выражения, подлежащего конкретизации в первую очередь. Просматривая слева направо символы выражения, содержащегося в поле зрения (это выражение будем в дальнейшем называть просто «поле зрения»), управляющее устройство метаалгоритмической машины (далее будем говорить просто «машина») ищет активный символ λ. Если такового в поле зрения не оказывается, происходит остановка машины, которая будет аварийной, если текущий уровень $n_t > 0$, и нормальной, если $n_t = 0$. Последний случай является единственным случаем, когда происходит нормальная остановка. Он означает, что метаалгоритмическая машина выполнила свою задачу, преобразовав путем ряда конкретизаций исходное активное выражение в конечное неактивное. Найдя активный символ λ (самый левый), машина исследует следующий непосредственно за ним терм, называемый *областью действия* символа λ (он обязательно существует, иначе поле зрения не было бы выражением).

Если найденный терм не активен или если он имеет вид $(\sim E)$, где E — произвольное (возможно, активное) выражение, то стоящий перед ним символ λ получает название ведущего символа конкретизации. Если он активен и не имеет вида $(\sim E)$, то в нем отыскивается первый слева активный символ λ и исследуется его область действия и т. д. В конце концов машина находит самый левый из всех активных символов λ, область действия которых либо не активна, либо имеет вид $(\sim E)$. Этот символ будет ведущим символом конкретизации. Терм, состоящий из ведущего символа конкретизации и следующего за ним терма, называется *ведущим активным термом*.

7.3. Термы вида $\lambda\,(\sim E)$, $\lambda\,(\rightarrow E)$, $\lambda\,(\leftarrow E)$, где E — произвольное выражение, называются соответственно *машинными операциями подстановки*, *занесения в память* и *вывода*. Если ведущий активный терм оказался машинной операцией, то машина выполняет соответствующую операцию, заканчивая тем самым и выполнение шага.

7.4. Выполнение машинной операции *подстановки* $\lambda\,(\sim E)$ заключается в следующем. Если $n_t > 0$, то отыскивается ведущий актив-

ный терм в рабочем поле уровня $n_t - 1$, после чего он заменяется на выражение E. Затем поле зрения и соответствующее ему поле памяти очищаются (делаются пустыми), а n_t уменьшается на единицу. Если $n_t = 0$, то происходит аварийная остановка.

7.5. Для описания операции *занесения в память* введем следующие определения. Назовем μ-преобразованием последовательности символов L метаалгоритмического языка, обозначаемым через $\mu_+ [L]$, последовательность символов, полученную заменой каждого управляющего символа в L пассивной парой, состоящей из этого управляющего символа и штриха $'$. Назовем μ-преобразованием последовательности символов L, обозначаемым через $\mu_- [L]$, последовательность символов, полученную заменой каждой пассивной группы в L соответствующим управляющим символом.

7.6. Машинная операция $\lambda (\rightarrow E)$ выполняется следующим образом. Если $\mu_- [E]$ — набор предложений, то он приписывается к концу набора предложений, стоящего в поле памяти, соответствующем полю зрения (уровень n_t), а ведущий активный терм $\lambda (\rightarrow E)$ уничтожается (стирается) в поле зрения. Этим выполнение шага заканчивается. Если $\mu_- [E]$ не является набором предложений, происходит аварийная остановка.

7.7. Если ведущий активный терм имеет вид $\lambda (\leftarrow E)$, где E — произвольное выражение, то выражение E «*выводится во внешние устройства*» (например, печатается), а ведущий активный терм $\lambda (\leftarrow E)$ стирается в поле зрения и выполнение шага заканчивается. Выражение «выводится во внешние устройства» взято в кавычки, так как внешние устройства не входят в состав метаалгоритмической машины и не описываются. Их может не быть вообще, работа машины от этого не изменится, если только $\lambda (\leftarrow E)$ во́время стирается в поле зрения. Для теоретических рассмотрений, при которых считается, что имеется возможность следить за состоянием памяти машины, операцию $\leftarrow$ можно было бы и не вводить, однако она чрезвычайно полезна при практическом использовании алгоритмов.

7.8. Если ведущий активный терм λW не является машинной операцией, то машина в совокупном наборе предложений находит последнее (по своему положению в наборе) предложение из числа предложений, *подходящих* для ведущего активного терма λW, заносит его правую часть на рабочее поле уровня $n_t + 1$, заменяя входящие в нее свободные переменные значениями, которые они приняли в процессе отождествления, и увеличивает n_t на единицу. Этим выполнение шага заканчивается. Если в совокупном наборе предложений нет подходящего для λW предложения, происходит аварийная остановка.

Описание метаалгоритмической машины закончено; полезно, однако, ввести еще два соглашения, которые усложняют (без принципиальной необходимости) определение метаалгоритмической машины и поэтому могут стать помехой при теоретических исследованиях, но зато упрощают запись, делают ее более естественной.

8.1. *Соглашение о подстановке.* Если правая часть предложения, используемого при выполнении шага, имеет вид $\sim E$, где E — произвольное выражение, то перед занесением на рабочее поле она преобразуется к виду $\lambda (\sim E)$.

8.2. *Соглашение о знаке* $\rightarrow$. Знак $\rightarrow$ воспринимается метаалгоритмической машиной как выражение $\rightarrow \S' \lambda'$.

8.3. Соглашения 8.1 и 8.2 имеют силу во всех случаях, когда не оговорено противное.

9. Поместим в основное рабочее поле метаалгоритмической машины какое-либо полностью определенное выражение E, а в соответствующее поле памяти — какой-либо набор предложений R. Остальные поля очистим, а в указатель поля зрения поместим число $n_t = 0$, так что выражение E окажется в поле зрения машины. Приведем управляющее устройство машины в рабочее состояние и запустим ее. Если после выполнения некоторого числа шагов машина придет в состояние нормальной остановки, то выражение E_1, стоящее в поле зрения (которым будет снова основное рабочее поле), назовем *результатом конкретизации* (или просто *конкретизацией*) E в *соответствии с* набором предложений R. Набор предложений R_1, содержащийся в момент остановки в основном поле памяти, назовем *конечным набором предложений*. Если при указанных начальных условиях машина приходит в состояние аварийной остановки или не останавливается вообще, будем говорить, что конкретизация E в соответствии с R невозможна.

ПРИМЕРЫ, ПОЯСНЕНИЯ И НЕКОТОРЫЕ ТЕОРЕМЫ

Можно провести параллель между элементами метаалгоритмического языка и элементами естественных языков.

Знакам соответствуют фонемы или буквы,—

это элементы, не имеющие сами по себе значения, а являющиеся лишь строительным материалом. В качестве знаков (несобственных) будем использовать буквы, цифры и специальные знаки, например $+$, $-$, $=$, и т. п., не перечисляя их заранее; однако при формулировке теорем предполагается, что мы всегда имеем дело с вполне определенной и заданной в явном виде совокупностью несобственных знаков.

Символам соответствуют морфемы.

Примеры символов: [1]

$$\mathbf{E}\,|\,\S\,|\,A\,|\langle\text{пока не}\rangle\,|+|\langle\text{прив. под. чл.}\rangle|)$$

Угловые скобки позволяют использовать для обозначения символов слова и словосочетания, поясняющие роль символа. Интервал между словами, который оставляется для удобства читателя, можно с равным успехом считать как несуществующим, так и представляющим особый знак.

Выражениям (в частности, термам и значащим символам) соответствуют слова и словосочетания.

Примеры термов:

$$A\,|\,(A)\,|\,(\quad)\,|\,((\alpha+\beta)+\gamma)\,|\,\mathbf{E}4\,|\,(\text{Ж}\gamma-\text{Ж}'\delta)\,\text{ЖЖЖ}\mathbf{S}5$$

Примеры выражений:

$$A\,|\,A+B\,|\,\text{Ж}(\alpha+\beta)+\gamma\,|\,\langle\text{если}\rangle\,a=b\,\langle\text{то}\rangle\,1\,|\,\mathbf{W}1+\mathbf{W}2$$

Теорема 1. Каждое непустое выражение, не являющееся термом, может быть единственным образом представлено как соединение некоторого числа термов.

Теорема 2. Соединение выражений есть выражение.

Примеры цепочек, которые не являются выражениями:

$$\text{если}\rangle\,|\,(\alpha+\beta)\,\text{Ж}\,|\,(x+\text{Ж})\,|\,\mathbf{EW}\,|\,\mathbf{E}(z)\,|\,\S\,26$$

Предложениям метаалгоритмического языка соответствуют предложения естественных языков. Пример предложения:

§ 4.2. Ж (<если> *и* <то> **E1** <иначе> *E*2) ~ **E1** Здесь 4.2 — комментарий, Ж, (<если> *и* <то> **E1** <иначе> **E2**) — левая часть, **E1** — правая часть.

Можно доказать следующую теорему.

Теорема 3. Множество выражений и множество предложений разрешимы.

Следствие. Множество лингем метаалгоритмического языка разрешимо.

[1] Примеры разделяются чертой |, которая не принадлежит к числу знаков языка.

Примеры синтаксического отождествления даны в таблице.

Таблица

Выражения	Выражения могут быть отождествлены как	Свободные переменные	Свободные переменные принимают значения
Ж $(a+1)$	Ж $(\mathbf{W}1+1)$	**W1**	a
$a+b+c+d$	**E1**+**E2**	**E1**	a
		E2	$b+c+d$
$\alpha\beta\alpha\beta(\delta\delta)\alpha\beta(\delta)\alpha\beta(\delta)$	**EA**$\alpha\beta$(**W***B*)**EC**	**EA**	$\alpha\beta\alpha\beta(\delta\sigma)$
		WB	δ
		E C	$\alpha\beta(\delta)$
$a+a+a+a$	**E1**+**E1**	**E1**	$a+a$
E′′′′	**W**α**E**β	**W**α	**E′**
		Eβ	′′′

Однако **E′′′′** не может быть отождествлено как SαEβ, так как **E** не является значащим символом.

Память метаалгоритмической машины для удобного выполнения рекурсивных процедур, которые требуют занесения в память промежуточных результатов, устроена по «стэковому» типу.

Порядок определения ведущего активного терма отражает общепринятый в математике способ мышления в терминах функций и аргументов и тот естественный принцип, что прежде чем вычислять функцию, надо вычислить ее аргументы. Исключение делается лишь для подстановки, к которой мы и переходим.

Приведем пример выполнения подстановки. Пусть ведущим активным термом является выражение Жx. Пусть последнее *подходящее* предложение в совокупном наборе предложений есть §Жx ~ 7. Тогда метаалгоритмическая машина, выполняя очередной шаг, занесет на рабочее поле уровня $n_l + 1$ выражение Ж (~ 7) (по соглашению 8.1) и сделает его полем зрения. В следующем шаге оно будет объявлено ведущим активным термом, и в результате выполнения операции подстановки поле зрения вернется на прежний уровень, а Жx будет заменено на 7. Следовательно, использование следующего уровня в этом случае является чисто формальным, и так будет всегда, когда правая часть начинается с символа ~, так как определение ведущего активного терма содержит принцип *приоритета подстановки*, согласно которому подстановка выполняется прежде, чем конкретизировано подставляемое выражение. Если правые части используемых предложений начинаются

не с символа $\sim$, можно много раз переходить на следующий уровень, однако возвратиться на прежний уровень можно только с помощью операции подстановки, которую теперь придется повторить столько раз, сколько раз мы ее пропустили.

При нашем определении занесения в память можно добиться занесения из поля зрения в соответствующее (и только в соответствующее!) поле памяти любого набора предложений R. Это достигается выполнением машинной операции $\mathfrak{Y}(\to \mu_+[R])$, что очевидно из следующих двух теорем.

Теорема 4. Если R — произвольный набор предложений, то $\mu_+[R]$ — неактивное выражение.

Теорема 5. Для всякой цепочки L цепочка $\mu_-[\mu_+[L]]$ совпадает с L.

Не стирая поле памяти в целом (при переходе на нижний уровень), нельзя стереть в нем какое-либо предложение; в этом, однако, нет необходимости, потому что любое предложение может быть фактически аннулировано, если занести в память предложение с той же левой частью, но с другой (нужной нам) правой частью. Если, например, в поле памяти стоит набор $\S\mathfrak{Y}x \sim 7\ \S\ \mathfrak{Y}x \sim 8\ \S\mathfrak{Y}x \sim 125$, то машина будет действовать в точности так же, как если бы вместо этого набора было одно последнее предложение $\S\ \mathfrak{Y}x \sim 125$.

Естественно поставить вопрос: является ли метаалгоритмический язык универсальным алгоритмическим языком, эквивалентным, скажем, языку нормальных алгорифмов [2]? Ответ на этот вопрос положительный, как это следует из теоремы 6.

Для формулировки теоремы 6 обозначим через A алфавит, состоящий из всех знаков метаалгоритмического языка, через $Б$ — алфавит из всех его несобственных знаков, а через A' — алфавит, полученный добавлением к знакам A разделительной черты $|$, и условимся, что заглавные курсивные латинские буквы будут обозначать как слова (в смысле теории нормальных алгорифмов Маркова), так и совпадающие с ними выражения метаалгоритмического языка.

Теорема 6. 1. Существует нормальный алгорифм $\mathfrak{A}$ над алфавитом A' такой, что если E — произвольное выражение, а R — произвольный набор предложений метаалгоритмического языка, то $\mathfrak{A}$ применим к слову $E|R$ в том и только в том случае, если возможна *конкретизация* E *в соответствии с* R, и перерабатывает слово $E|R$ в слово $E_1|R_1$, где E_1 — *конкретизация* E *в соответствии с* R, а R_1 — *конечный набор предложений.*

2. Каков бы ни был нормальный алгорифм $\mathfrak{B}$ в алфавите $Б$, существует такой набор предложений R, что для всякого слова D_1 в алфавите $Б$, которое $\mathfrak{B}$ перерабатывает в слово D_2, *конкретизация* $\mathfrak{Y}(D_1)$ *в соответствии с* R есть D_2, а для всякого слова D_3, к которому нормальный алгорифм $\mathfrak{B}$ не применим, *конкретизация* $\mathfrak{Y}(D_3)$ *в соответствии с* R невозможна.

Первая часть теоремы 6 следует из того, что работа метаалгоритмической машины была описана нами как алгоритмический процесс, поэтому хотя построение нормального алгорифма $\mathfrak{A}$ довольно громоздко, принципиальная возможность этого очевидна.

Для доказательства второй части достаточно заметить, что схема (т. е. список формул подстановки) нормального алгорифма может быть очень просто переписана в виде такого набора предложений, что метаалгоритмическая машина будет выполнять в точности те же действия, которые предписываются для выполнения нормального алгорифма[1]. Для этого нужно каждую *простую* формулу заменить предложением

$$\S\,\mathfrak{Y}(\mathrm{E}1A\mathrm{E}2) \sim \mathfrak{Y}(\mathrm{E}1B\mathrm{E}2),$$

а каждую *заключительную* формулу $A \to \cdot B$ — предложением

$$\S\,\mathfrak{Y}(\mathrm{E}1A\mathrm{E}2) \sim \mathrm{E}1B\mathrm{E}2,$$

записать их в порядке, обратном тому, в котором они расположены в схеме, и в самом начале набора приписать предложение

$$\S\,\mathfrak{Y}(\mathrm{E}1) \sim \mathrm{E}1.$$

ФОРМАЛИЗАЦИЯ НЕКОТОРЫХ ПОНЯТИЙ ПРОГРАММИРОВАНИЯ

Не имея возможности обсуждать вопрос, приведем просто описание некоторых важнейших понятий программирования (и математики вообще) на метаалгоритмическом языке, предоставив читателю самому проследить, как они «понимаются» метаалгоритмической машиной. Комментарии будем использовать, в частности, для нумерации предложений. Предложениям, относящимся к одному понятию, будем приписывать номера, различающиеся только «десятичной частью», и говорить, что они образуют *группу*

[1] Это замечание принадлежит В. М. Веселову.

предложений. Порядок предложений в группе, как правило, чрезвычайно существенен. При записи длинных цепочек будем переходить на следующую строчку без всякого знака переноса.

§ Правило снятия лишних скобок $Я(Wx)\sim Я Wx$

§. 1.1. $Я(Ea \equiv Eb)\sim л$

§ 1.2. $Я(Ea \equiv Ea)\sim и$

§ 2. $Я(Ea = Eb)\sim Я(Я(Ea)\equiv Я(Eb))$

§ 3. $Я(Wx := Ea)\sim Я(\to x \sim Ea)$

§ 4.1. $Я(\langle$если$\rangle Ec\langle$то$\rangle Ea\langle$иначе$\rangle Eb)\sim$
$\sim Я(\langle$если$\rangle Я(Ec)\langle$то$\rangle Ea\langle$иначе$\rangle Eb)$

§ 4.2. $Я(\langle$если$\rangle и\ \langle$то$\rangle Ea\langle$иначе$\rangle Eb)\sim Я(Ea)$

§ 4.3. $Я(\langle$если$\rangle л\ \langle$то$\rangle Ea\langle$иначе$\rangle Eb)\sim Я(Eb)$

§ 5.1. $Я(\neg Wa)\sim Я(\neg Я Wa)$

§ 5.2. $Я(\neg и)\sim л$

§ 5.3. $Я(\neg л)\sim и$

§ 6.1. $Я(Wa \vee Wb)\sim Я(Я Wa \vee Wb)$

§ 6.2. $Я(и \vee Wb)\sim и$

§ 6.3. $Я(л \vee Wb)\sim Я Wb$

§ 7.1. $Я(Wa \,\&\, Wb)\sim Я(Я Wa \,\&\, Wb)$

§ 7.2. $Я(и \,\&\, Wb)\sim Я Wb$

§ 7.3 $Я(л \,\&\, Wb)\sim л$

§ 8.1. Двоеточие означает последовательное выполнение операторов. $Я(:E1;\ E2)\sim Я(E1)\,Я(:E2)$

§. 8.2. $Я(:)\sim$

§ 9.1. Цикл. $Я(\langle$повторять$\rangle Wa\ \langle$пока не$\rangle$
$Wc)\sim Я(\langle$если$\rangle Wc\ \langle$то$\rangle\ \langle$конец$\rangle\ \langle$иначе$\rangle\ \langle$выполнить$\rangle Wa\ \langle$и возобновить цикл с условием$\rangle Wc$

§ 9.2. $Я(\langle$конец$\rangle)\sim$

§ 9.3. $Я(\langle$выполнить$\rangle Wa\ \langle$и возобновить цикл с условием$\rangle Wc)\sim Я Wa Я(\langle$повторять$\rangle Wa$
$\langle$пока не$\rangle Wc)$

Введя в память метаалгоритмической машины этот набор предложений, программист получает возможность разговаривать с ней на языке типа АЛГОЛа. Из существенных черт АЛГОЛа здесь не нашли отражения переход по метке и блочная структура. Отсутствие перехода по метке легко возместить путем обозначения соответствующих отрезков программы идентификаторами, а возможности, которые дает блочная структура, реализуются благодаря стэковой структуре памяти.

Проиллюстрируем описание на метаалгоритмическом языке некоторых простых понятий, типичных для задач, связанных с преобразованием цепочек символов.

§ 10.1. $Я(\langle$последний символ выражения$\rangle)\sim$

§ 10.2. $Я(\langle$последний символ выражения$\rangle$
$EBSL)\sim SL$

§ 11.1. $Я(\langle$выражение$\rangle Ea\langle$содержит $+$
$+\ \rangle)\sim л$

§ 11.2. $Я(\langle$ выражение$\rangle E1 + E2\langle$ содержит $+\ \rangle)\sim и$

§ 12. $Я(\langle$ левая часть равенства$\rangle EL =$
$= ER)\sim EL$

Теперь становятся формализованными и понятными для машины выражения типа

$Я(\langle$если$\rangle(\langle$выражение$\rangle Яx\langle$ содержит $+\rangle)\,\&\,((\langle$ левая часть равенства $\rangle Яy) = a)$
$\langle$то$\rangle : \alpha := 1;\ \beta := 0;\ \langle$иначе$\rangle\ \langle$повторять $\rangle A\langle$пока не$\rangle\ B)$.

Приведем пример формализации рекурсивных определений. Обычное (словесное определение числа в метаматематике гласит:

1) 0 есть число;

2) если N есть число, то Ns есть число;

3) пункты 1 и 2 определяют *все* числа.

(Так как штрих′ является *собственным символом* метаалгоритмического языка, отношение следования обозначаем с помощью буквы s.)

Определение соответствующего предиката на метаалгоритмическом языке имеет вид

§ 13.1. $Я(E\langle$прочее$\rangle)\langle$есть число$\rangle)\sim л$

§ 13.2. $Я(ENs\langle$есть число$\rangle)\sim Я(EN$
$\langle$есть число$\rangle)$

§ 13.3. $Я(0\langle$есть число$\rangle)\sim и$

Видим, что формализованное определение весьма напоминает обычное (читать надо снизу вверх).

Рекурсивное определение сложения:

§ 14.1. $Я(Ea + 0)\sim Ea$

§ 14.2. $Я(Ea + Ebs)\sim Я(Eas + Eb)$

В заключение коснемся вопроса о построении метаалгоритмической машины «в натуре», т. е. создании вычислительной машины с логикой метаалгоритмической машины. Современные вычислительные машины хорошо приспособлены к решению арифметических задач. Так как действия над многозначными числами, сами по себе сложные и состоящие из большого числа первичных операций, выполняются аппаратно и поэтому, во-первых, быстро, а во-вторых, без необходимости для программиста думать об их выполнении. Для решения же неарифметических задач вычислительная машина может предложить лишь такие элементарные действия как сдвиги, логические сложения и т. п., которые никак не связаны с сущностью решаемых задач. Возникает задача нахождения более крупного, и в тоже время общего, элементарного действия, которое имело бы смысл осуществлять аппаратно. Элементарная конкретизация (один шаг метаалгоритмической машины) может быть предложена в качестве одного из возможных

решений. Главное преимущество метаалгоритмической машины состоит в чрезвычайном упрощении программирования логических, символических, лингвистических и т. п. задач и вытекающих отсюда удобствах откладки, контроля и т. д.,— короче говоря, в удобстве общения человека с машиной. Но если будут разработаны эффективные технические способы осуществления *конкретизации*, то можно рассчитывать и на значительный выигрыш в эффективности машины в целом — за счет увеличения аппаратной части работы и уменьшения программной части.

При отсутствии специальной машины метаалгоритмический язык может быть использован в качестве входного языка программирования для обычных машин. Для исследований в этом направлении автором был написан транслятор (работающий в режиме интерпретации) с метаалгоритмического языка на машину М—20 («КИТ-1»). О результатах использования этого транслятора будет сообщено в следующих работах.

Автор выражает благодарность В. И. Сердобольскому, В. М. Веселову и другим товарищам, совместная работа с которыми способствовала совершенствованию метаалгоритмического языка.

ЛИТЕРАТУРА

1. В. Ф. Турчин, Метаязык для формального описания алгоритмических языков, сб. «Цифровая вычислительная техника и программирование», изд-во «Советское радио», М.— Л., 1966.
2. А. А. Марков, Теория алгорифмов, Труды Математического института им. В. А. Стеклова, Изд-во АН СССР, 1954.
3. Report on the Algorithmic Language ALGOL-60, «Communication of the ACM», 3, N 5, 1960.

Поступила в редакцию
10. IX 1966