MATHEMATICA VOL. 5 (28), 2, 1963, pp. 247—260

МЕТОД АНАЛИЗА ГРАФ-СХЕМНЫХ АЛГОРИФМОВ

ЕМИЛЬ МУНТЯНУ

Клуж

§ 1.

Введение

Алгорифмизация математических и логических задач для их решения при помощи электронных счетных машин привела к созданию различных алгорифмических языков или различных формальных языков, какими являются языки, возникшие из практики необходимого для этой цели программирования.

Некоторые алгорифмические языки, как, например, языки, созданные в математической логике — рекурентные функции, машина Тьюринга, нормальные алгорифмы и другие — не были созданы с целью алгорифмизации математических и логических задач, как рассматривается ныне вопрос алгорифмизации в свете возникновения и быстрого развития электронных счетных машин и, следовательно, они в меньшей мере соответствуют этой цели. Другие же алгорифмические языки — операторные алгорифмы [2], алгорифмы в виде граф-схемы [4], адресные алгорифмы [6], — были созданы с этой целью и применяются в той или иной мере для формальной записи различных задач. Кажется, что из этих языков запись алгорифмов в виде граф-схемы легче применяется и употребляется больше всех в целях формализации. Довольно часто употребляются и формальные языки, как, например, логические схемы алгорифмов [7], матричные схемы [3] и другие, которые, в отличие от алгорифмических языков, хотя и не уточняют ясно понятия об алгорифме, но, возникшие в практике программирования, доказали свою полезность в формальной записи различных задач для их разрешения при помощи электронных счетных машин. Здесь мы касаемся, в первую очередь, логических схем программ.

Также были созданы различные методы перевода с одного формального языка на другой формальный язык и, в особенности, переводов на языки машины (так называемые программы программирования).

Меньше была исследована другая, часто встречающаяся сторона вопросов алгорифмизации. Известно, что существенным этапом этого процесса является анализ алгорифма, написанного на таком языке. В случае универсальных цифровых электронных машин этот этап состоит в анализе программы путём её отладки до её осуществления. Отладка очень трудна, более или менее полна, не всегда удачна, так как применяемые методы отладки являются только искусственными методами и приёмами, возникшими в практике программирования, в большинстве случаев основывающимися на частных случаях предложенной задачи.

Ешё в 1947 году Т. ФОН НЕУМАНН и Х. Х. ГОЛЬДСТИНЕ, предлагая логикоформальную запись программ в виде блок-схемы, выдвигают вопрос об анализе программ, написанных в этом виде. Этот вопрос ставится различными авторами в различных частных видах. В трудах И. Ю. ЯНОВА [3], исходя из записи алгорифмов в виде логической схемы, стало возможным построить алгебру, в которой изучаются преобразования, ведущие эквивалентным схемам в точное копределённом смысле. Самый простой случий такой алгебры, а именно, тот, в котором изучаются только формальные преобразования логических условий, был исследован И. Ю. ЯНОВЫМ в высшеуказанном труде. Ряд полуформальных преобразований логических схем алгорифмов был выявлен в практике программирования и было обнаружено, что они существенно упрощают заданную логическую схему. Трудность исследования таких полуформальных преобразований состояла в том, что язык логических схем казался негодным для их выявления Р. И. ПОДЛОВЧЕНКО [8] вводит новый язык для записи логических схем алгорифмов, отличающийся от языка, употребляемого И. Ю. ЯНОВЫМ, и таким образом исследует эти полуформальные преобразования, связанные с переходом от одной системы параметров к другой. В этих трудах вопрос анализа алгорифмов состоит в осуществлении преобразований в записи алгорифмов, позволяющих переходить к другому алгорифму, эквивалентному заданному алгорифму.

Другие авторы, как например, Р. М. КАРП [5], употребляя некоторые понятия из теории графов, исследуют довольно частный случай анализа алгорифмов, написанных в виде граф-схемы. Метод анализа, предложенный Р. М. КАРПОМ, даёт возможность определить в граф-схеме те операторы, исходя из которых никогда не доходится до конечного оператора, а также те операторы, которые никогда не могут быть достигнуты исходя из начального оператора. В первом случае мы имеем дело с алгорифмом, действие которого никогда не прекращается, а во втором случае алгорифм содержит операторы, которых никогда нельзя достигать. К сходным результатам, но иным путём, приходит и Р. Т. ПРОССЕР [9].

Л. Х. ВЯЗАЛОВ и Ю. И. МОРОЗОВ [11] получили вспомогательную стандартную программу для отладки программ для машины „Стрела". Но эта программа является только формализацией одной из эмпирических процедур, более часто применяемых в работе по отладке. Она

осуществляет анализ путём выполнения по частям проверяемой программы. Уже пренебрегая тем, что деление проверяемой программы на части, для которых необходимо знать путь их выполнения, производится в зависимости от характера задачи и от необходимости отладки той или иной части программы, полная отладка предполагала бы, как и для любой эмпирической процедуры, применяемой в практике программирования, выполнение в целом программы с некоторыми очень упрощёнными начальными данными.

Вопрос об анализе алгорифмов и до сих пор не был ещё сформулирован с более исчерпывающей точки зрения, а именно: правильно или неправильно был построен алгорифм? Это значит, что ещё не известна процедура, или частные случаи алгорифмов, при помощи которых могли бы заранее установить (без применения алгорифма) если конечные результаты, достигнутые путём применения алгорифма к начальным данным задачи, содержат или не содержат конечных данных, требуемых разрешением задачи. Такая процедура, помимо её значения в процессе алгорифмизации различных математических и логических задач ввиду их разрешения при помощи электронных счётных машин, найдёт себе широкого применения в практике программирования. Осуществление такой процедуры на универсальной цифровой электронной машине значило бы универсальную программу полной отладки программ до их выполнения. Это избегало бы, с одной стороны, тяжёлого, а иногда и безрезультатного труда отладки программ программировщиком, а с другой стороны, на много уменьшало бы непродуктивное время машин, предусмотренное для отладки программ.

Кроме того, такая процедура облегчала бы постановку вопроса об автоматическом построении алгорифмов.

На основе такой новой формулировки вопроса об анализе алгорифмов, в настоящем труде мы представим новый метод анализа довольно общего класса алгорифмов, записанных в виде граф-схемы.

Настоящий труд представляет процедуру анализа в том частном случае, когда алгорифмы не содержат циклов, без всяких других ограничний относительно класса задач, для решений которых существуют такие алгорифмы.

§ 2.

Определение граф - схемных алгорифмов

Предметом исследования в настоящем труде являются алгорифмы, записанные при помощи граф-схем или граф-схемные алгорифмы, как мы будем их называть в дальнейшем.

В этом параграфе мы дадим определение понятия о граф-схеме, а также и соответствующую интерпретацию, характерную для намеченной цели, что приведёт нас к определению графсхемного алгорифма.

Под *граф-схемой I'* поцимаем математический объект, описанный следующим образом

Пусть конечное множество объектов

$$T = \{T_1, T_2, \ldots, T_n\}, \tag{1}$$

которых мы называем *преобразователями*, и пусть второе множество объектов

$$D = \{D_1, D_2, \ldots, D_m\}, \tag{2}$$

элементы которого мы называем *распознавателями*.

Граф-схема Г является графом G, конечным, связанным и ориентированным, т. е. конечным множестве точек, названных *узлами*, некоторые из них будучи соединены между собой ориентированными отрезками, названными *стрелами*. Это множество удовлетворяет следующим свойствам:

1. *Один узел графа G, меченный, носит название* входа *и обозначаем его через I. Он является единственным узлом графа, в котором не кончается ни одна стрела.*

2. *Граф G имеет второй меченный узел, который носит название* выхода *и обозначается E. Выход является единственным узлом графа из которого не выходит ни одна стрела.*

3. *Каждому узлу графа, кроме входа и выхода, соответствует либо один преобразователь T_i, и тогда соответствующий узел называется* α-узлом, *либо один распознаватель D_j, и тогда называем его* β-узлом.

4. *Из каждого α-узла всегда выходит только одна стрела.*

5. *Из каждого β-узла всегда выходят две стрелы, одна носит знак* + (*плюс*), *а другая знак* — (*минус*).

Дадим теперь интерпретацию множеств T и D, которая поведёт нас к понятию о граф-схемных алгорифмах.

Если задаются множества T и D, то каждой граф-схеме Г единственным образом может соответствовать один „алгорифм" для вычисления значения функции, если каждый преобразователь T_i понимаем как функцию, преобразующую элементы множества в то же множество и каждый распознаватель D_j понимаем как действие, проверяющее если элемент рассматриваемого множества обладает ли определённым свойством и определяет припенение той или иной функции в зависимости от того, обладает ли или нет соответствующий элемент соответствующим свойством.

Известно, что программа задачи для универсальной цифровой электронной машины является в сущности записью алгорифма решения соответствующей задачи на языке машины. Программа применяется к состоянию памяти машины (к так называемому начальному состоянию), содержащему начальные данные задачи, и преобразует это состояние в другое (называемое конечным состоянием), которое должно содержать результат задачи в том случае, когда программа была правильно составлена.

Для более понятного изложения, ниже проведём аналогию с программой универсальной цифровой электронной машины, и введём для определения граф-схемного алгорифма понятие о памяти и о состоянии памяти.

Называется *памятью* произвольное множество объектов $Z=\{z\}$, элементы которого называются *ячейками*. Рассмотрим множество $M=\{x\}$, элементы которого мы называем состояниями. Делаем так, чтобы каждой ячейке памяти соответствовал элемент множества M. Этот элемент называется *состоянием ячейки z*. (Нескольким ячейкам памяти может соответствовать одно и то же состояние x, но не допускается того, чтобы одной единственной ячейке в определённый момент соответствовало несколько состояний).

Если каждой ячейке памяти соответствует одно состояние x множества M, то будем говорить, что в соответствующий момент подмножество M, содержащее все эти элементы, называется *состоянием памяти* и обозначаем его через $S = \{x\}$. Обозначаем через $H = \{S\}$ множество всех состояний памяти.

Функция

$$A(S) = S' \tag{3}$$

определимая на множестве H и преобразующая это множество в оно же самое называется *оператором*. Пусть Q, $= \{0, 1\}$ множество чисел 0 и 1, тогда функция

$$P(S) \tag{3'}$$

также определимая на множестве H всех состояний памяти и преобразующая это множество во множество Q, называется *логическим оператором* или логическим условием.

Пусть заданы множества $T= \{T_1, \ldots, T_n\}$ и $D = \{D_1, \ldots, D_m\}$. Под их интерпретацией понимаем следующее: даётся множество операторов $A = \{A_1, \ldots, A_p\}$ и множеств ологических операторов $P= \{P_1, \ldots, P_r\}$. Делаем так, чтобы каждому преобразователю T_i соответствовал оператор $A_k \in A$ (допускается и то, чтобы нескольким преобразователям соответствовал один и тот же оператор) и каждому распознавателю D_j соответствовал логический оператор $P_l \in P$. Также даётся состояние $S_0 \in H$. Эту интерпретацию обозначаем символом

$$\{S_0 ; T \to A ; D \to P\} \tag{4}$$

Если задана интерпретация (4), то каждую грав-схему Г можно считать единственным описанием одного процесса преобразования состояния S_0, названного *начальным состоянием*, в другое состояние S_f памяти, которое мы называем *конечным состоянием*.

Граф-схему Г вместе с интерпретацией $\{S_0 ; T \to A ; D \to P\}$ будем называть *граф—схемным алгорифмом* и будем обозначать Γ_{A-P}. Действие алгорифма Γ_{A-P} будем описывать следующим образом:

Начальное состояние S_0, начиная с входа I проходит граф-схему Γ по стрелам, преобразуясь каждый раз, когда проходит через α-узел и оставаясь неизменным каждый раз, когда проходит через β-узел. Граф-схема проходится следующим образом:

а). Предполагаем, что состояние S_0, после некоторого числа преобразований, становится состоянием S', и доходит до α-узла, которому соответствует преобразователь T_i. В этом случае, из этого узла, следуя единственной стреле, которая выходит из этого узла, продолжает свой путь состояние $S''=A_k(S')$, где A_k является оператором, который по интерпретации (4), соответствует преобразователю T_i.

б). Если состояние S' в своём пути встречает β-узел, которому соответствует распознаватель D_j, тогда то же состояние будет продолжать свой путь по стреле, обозначенной знаком + (плюс) в том случае, когда $P_l(S')=1$, или по стреле, обозначенной знаком — (минус), если $P_l(S')=0$. P_l является логическим оператором, который по данной интерпретации соответствует распознавателю D_j.

в). Если в определённом этапе до выхода E граф-схем Γ доходит определённое состояние S_f, то это состояние будем считать результатом применения граф-схемного алгорифма Γ_{A-P}, определённого при помощи граф-схемы Γ интерпретацией $\{S_0, T\to A; D\to P\}$. Это будем обозначать следующим образом

$$\Gamma_{A-P}(S_0)=S_f \tag{5}$$

Если операторы $A_i\in A$ являются нормальными алгорифмами в алфавите N, а логические операторы $P_i\in P$ также являются нормальными алгорифмами в алфавите $NU\{d, a, n, u\}$, то интерпретация называется нормальной, а алгорифм Γ_{A-P} называется квазинормальным [10]. Ясно, что любой нормальный алгорифм в N является квазинормальным алгорифмом в N.

М. П. ТЕР-ЗАХРЯН, в своём труде [10], для любого произвольного квазинормального алгорифма $\mathcal{H}$ строит нормальный в определённом алфавите алгорифм $\mathcal{G}$, эквивалентный квазинормальному алгорифму $\mathcal{H}$. Хотя не имеет никакой связи с тем, что следует ниже мы сделали, это замечание, чтобы подчеркнуть то, что высшеприведенное определение граф-схемных алгорифмов можно считать уточнением понятия об алгорифме, если соответствующая интерпретация является удачной, т. е. если имеем дело с нормальной интерпретацией или с интерпретацией в которой операторы и логические операторы описываются рекурентными функциями, машиной Туринга и т. п.

§ 3.

Описание формальной системы.

Для исследования граф-схемных алгорифмов с изложенной во введении точки зрения, опишем формальную систему, при помощи которой определим понятие о формуле.

Под *переменной* формальной системы понимаем один из символов $\xi, \eta, \ldots, x, y, i, k, \ldots, x_1, y_1, z_1$. Число переменных бесконечно и не будем предявлять никакого ограничения их области значений. Там, где это потребуется, мы уточним природу соответствующего переменного.

Под *константой* формальной системы понимаем один из символов

$$C_1, C_2, \ldots, C_k.$$

Также, что касается природы констант, не предявляем никакого ограничения, следуя, как и в случае пременных, уточнять это всегда, когда оказывается необходимым. Предполагаем, что число констант конечно, так как, при помощи операций, которые мы введём, и высшеуказанных констант мы можем писать любую другую константу любой природы. Это является одновременно и первой гипотезой о константах.

Операции формальной системы мы разделим на два класса: унарные операции и бинарные операции.

Под *унарной операцией* понимаем один из символов: $\exp_a$, $\ln$, $\sqrt[n]{}$, $/_n$ символы тригонометрических функций и их обратных функций, символы гиперболических функций и их обратных функций. Первые 4 символа носят названия: „экспоненциальная с базой a“, „натуральный логарифм“, „корень порядка n“, „степень n“.

Под *бинарной операцией* понимаем один из символов $+, -, \cdot, :$. Название каждого известно.

Поу *реляцией* формальной системы понимаем один из символов

$$=, >$$

с известными соответствующими названиями.

Под *пропозициональной связью* системы понимаем один из символов

$$\neg, \wedge, \vee, \supset, E, A$$

носящих, соответственно, названия: отрицание, конъюнкция, дисъюнкция, импликация, квантор существования и квантор обобщения. Вводим следующее рекурентное опрэдение понятия о терме:

а). Любая переменная и любая константа является термом.

б). Если α и β являются термами, то

$$(\alpha 0\beta) \text{ и } (\theta x)$$

также являются термами. Здесь, для краткости, мы обозначили через 0 любую из высшеопределённых четырёх бинарных операций и через θ любую унарную операцию из введенных нами операций в формальную систему.

Элементарная формула

а) Если α и β являются термами, то

$$(\alpha = \beta) \quad \text{и} \quad (\alpha > \beta)$$

являются элементарными формулами.

Формула. Аналогично, как и для понятия о терме, вводим рекурентное определение.

а). Любая элементарная формула является формулой.

б). Если φ и ψ являются формулами, то

$$(\varphi \wedge \psi),\ (\varphi \vee \psi),\ (\varphi \supset \psi) \quad \text{и} \quad \neg\varphi$$

также являются формулами.

в). Если φ является формулой и ξ-переменной, входящей в φ то

$$(E\xi)(\varphi) \quad \text{и} \quad (A\xi)(\varphi)$$

являются формулами.

Что касается переменной ξ в формулах $(E\xi)(\varphi)$ и $(A\xi)(\varphi)$ будем говорить, что она является *связанной переменной*. Если переменная η входит в формулу φ и не подвергается действию какого-либо оператора говорим, что она является *свободной переменной* в φ.

В том, что следует ниже понятие о верности формулы имеет существенное значение. Мы это понятие будем применять интуитивно, без формального определения. Так, мы будем говорить, что формула

$$(Ax)\,(Ay)\,[(x + y) = (y + x)]$$

верна, а формула

$$(Ax)\,(Ay)\,[(x - y) = (y - x)]$$

ложна.

Конечно, нельзя говорить о формуле, содержащей свободные переменные, что она верна или ложна, так как ей будут удовлетворять определённые значения свободных переменных, а другие нет.

Если φ является формулой, не содержащей свободных переменных, то $\neg\varphi$ верна если и только если φ ложна.

Если φ и ψ являются формулами, не содержащими свободных переменных, то формула $(\varphi \wedge \psi)$ верна если и только если φ и ψ являются обе верными; формула $(\varphi \vee \psi)$ верна если хотя одна из формул φ и ψ верна; формула $(\varphi \supset \psi)$ верна если и только если или формула φ ложна или формула ψ верна.

Если φ является формулой, единственной свободной переменной которой является ξ, то говорим, что формула $(A\xi)(\varphi)$ верна если для любого значения свободной переменной ξ формула φ верна, а о формуле

$(E\xi)(\varphi)$ говорим, что она верна если существует такое значение переменной ξ, чтобы формула φ была верна. Квантор $(A\xi)(\varphi)$ читается: „для любого значения переменной ξ формула φ верна„, а квантор $(E\xi)(\varphi)$ читается: „существует значение переменной ξ, для которого формула φ верна".

§ 4.
Процедура анализа граф-схемных алгорифмов.

Ниже мы упростим обозначения, употребляя для граф-схемного алгорифма $\Gamma_{A\text{-}D}$, данного интерпретацией $\{S_0;\ T \to A;\ D \to P\}$, символ $\mathcal{A}$, подразумевая каждый раз, что определяется граф-схема Г и даётся интерпретация $\{S_0;\ T \to A;\ D \to P\}$.

Дадим точную формулировку упомянутой во введении задачи анализа алгорифмов.

Пусть K класс граф-схемных алгорифмов и пусть алгорифм $\mathcal{A}$ -некоторый элемент этого класса, преобразующий начальное состояние S_0 в конечное состояние S_f, т.е.

$$\mathcal{A}(S_0) = S_f.$$

Пусть $S_0 = \{x_1, x_2, \ldots, x_n\}$ и $S_f = \{y_1, y_2, \ldots, y_m\}$, где $x_i (i = 1, 2, \ldots, n)$ и $y_j\ (j = 1, 2, \ldots, m)$ являются совокупностью элементов состояний S_0 и, соответственно, S_f.

Пусть

$$(F(x_1, x_2, \ldots, x_n; y_1, y_2, \ldots, y_m)) \tag{6}$$

формула, выражающая реляции между элементами начального состояния и элементами конечного состояния. Первым ограничением, которое мы предявляем классу К алгорифмов, является то, что эти алгорифмы относятся к тем задачам, для которых можно построить формулу (6). Довольно трудно уточнить класс задач, для которых можно построить такую формулу. Всё-таки можно с уверенностью утверждать, что этот класс содержит все задачи, требующие численного решения, задачи, требующие выполнения аналитических расчётов, а также многие другие задачи логики.

Дадим простой пример для иллюстрированния того, как строится формула (6), хотя построение этой формулы не является целью нашего труда. Эта формула, вместе со спецификацией, какие из величин задачи являются начальными величинами и какие из них являются конечными величинами, а также и алгорифм $\mathcal{A}$ решения задачи, являются начальными данными, из которых исходим при потроении процедуры анализа.

Рассмотрим задачу составления таблицы функции $y = f(x)$ для значений $t_1, t_2, \ldots, t_p$ аргумента. Следовательно, начальными данными являются $t_1, t_2, \ldots, t_p$, которые, вместе с другими величинами, необ-

ходимыми действию алгорифма, составляют состояние S_0. Конечным состоянием является $S_f = \langle y_1, y_2, \ldots, y_p \rangle$, где $y_i = f(t_i)$ $(i = 1, 2, \ldots, p)$. В этом случае формула (6), выражающая реляции между состояниями, будет иметь вид:

$$(A_k)\ [(1 \leqslant k \leqslant p) \supset (y_k = f(t_k))]$$

или, так как символ $\leqslant$ не фигурирует во введенной формальной системе, её было необходимо написать

$$(A_k)\ [(\neg(k < 1) \wedge [(k < p) \vee (k = p)]) \supset (y_k = f(t_k))]$$

Теперь задача анализа алгорифмов формулируется следующим образом:

Требуется построить алгорифм $\mathcal{B}$, который, применённый к любому алгорифму $\mathcal{A}$ класса K, преобразует этот алгорифм в формулу

$$(F'(x_1, x_2, \ldots, x_n; y_1, y_2, \ldots, y_m)) \tag{7}$$

аналогичную формуле (6), т.е. чтобы имели

$$\mathcal{B}(\mathcal{A}) = F'$$

Алгорифм $\mathcal{B}$, построенный таким образом, приводит к тому, что каждому алгорифму $\mathcal{A}$ класса K соответствует одна формула (7). Будем говорить, что алгорифм $\mathcal{A}$ правильно построен если формула

$$(F' \supset F) \tag{8}$$

верна. Следовательно, для полного разрешения задачи анализа необходимо построить ещё один алгорифм $\mathcal{C}$, при помощи которого могли бы установить если формула формальной системы верна или нет. Такой алгорифм $\mathcal{C}$, применеёный к формуле (8), преобразует эту формулу в ответ „да" если алгорифм $\mathcal{A}$ правильно построен, и в ответ „нет" если алгорифм $\mathcal{A}$ неправильно построен.

Как легко видеть, нет необходимости в том, чтобы между формулами F и F' существовала эквивалентность, т. е. нет необходимости в том, чтобы была верна и импликация $F \supset F'$, так как с нашей точки зрения достаточно если алгорифм $\mathcal{A}$ обеспечивает в конечном счёте результаты, требуемые задачей, т. е. величины $y_j (j = 1, 2, \ldots, m)$.

Об алгорифме $\mathcal{A}$ будем говорить, что он содержит циклы если необходимо применять несколько раз один или несколько операторов до того, пока соответствующее состояние не удовлетворяет одному определённому логическому оператору.

Алгорифм $\mathcal{B}$, построенный ниже, относится к алгорифмам без циклов, следуя, чтобы во второй части труда дать алгорифм построения формулы F для хорошо уточнённого класса циклов, а также и алгорифм $\mathcal{C}$ проверки формулы (8).

Обозначаем узлы граф-схемы Г алгорифма $\mathcal{A}$ следующим образом

Узел I обозначаем L_{p+1}.

Узел E обозначаем L_0.

Остальные узлы обозначаются символами $L_1, L_2, \ldots, L_p$ так, что если стрела направляется от узла L_i к узву L_j имели бы $i > j$ для того случая, когда L_i является α-узлом, а если он является β-узлом имели бы $i > j'$ и $i > j''$, где $L_{j'}$ и $L_{j''}$ являются узлами, к которым направляются из узла L_i стрелы, намеченные знаком $+$, и, соответственно, знаком $-$.

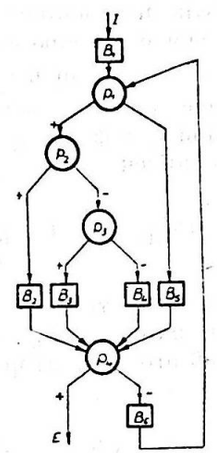
Эти условия не являются другим, чем ограничением, предявленным алгорифму $\mathcal{A}$, а именно то, что он не содержит циклов. Каждый узел $L_i = (i = 1, 2, \ldots, p)$ является либо оператором A_k либо логическим оператором P_l.

Присваеваем узлу формулу

$$F_0 \equiv (0 = 0)$$

Опишем теперь алгорифм, который осуществляет построение формулы F_i, зная формулу F'_i, где $i > i'$. Алгорифм состоит в построение одной формулы F_k на каждом узле L_k. Так как число узлов граф-схемы Г конечен, действие алгорифма $\mathcal{B}$ кончится построением формулы F_{p+1}, которая и является искомой формулой F'.

Граф-схема $\mathcal{G}$ алгорифма $\mathcal{B}$ даётся ниже



Интерпретация преобразователей и распознавателей, встречающихся в граф-смене $\mathcal{G}$ сходующая:

B_1 является оператором, который для заданного j готовит узел L_j граф-схемы Г для анализа.

p_1 является логическим оператором, из которого действие идёт по стреле, обозначенной знаком + в том случае, когда к узлу присоединился оператор $A_k \equiv (y_k = f_k(x_{i_1}, x_{i_2}, \ldots, x_{i_n}))$, или по стреле, обозначенной знаком — в противном случае.

В том случае, когда L_j является оператором A_k, пусть L_{j_1} $(j_1 < j)$ узел, к которому направляется стрела, исходящая из L_j, и пусть формула, построенная алгорифмом при анализе узла L_{j_1}. Формула F_{j_1} построена, так как $j_1 < j$.

Логический оператор p_2 определяет то, входит ли переменная y_k в формулу F_{j_1}. В положительном случае путь продожается по стреле, обозначенной знаком +, а в противном случае, — по стреле, обозначенной знаком —.

Логический оператор p_3 проверяет является ли переменная y_k конечной величиной. В положительном случае путь продолжается по стреле, обозначенной знаком +, а в противном случае, — по стреле, обозначенной знаком —.

Оператор B_2 осуществляется в том случае, когда L_j является оператором A_k, и переменная y_k входящая в формулу F_{j_1} выражает промежуточный результат, который зависит только от исходных данных. В этом положении этот оператор строит формулу

$$F_j \equiv (S^{y_k}_{f_k(x_{i_1}, \ldots, x_{i_n})} F_{j_1}) \tag{9}$$

где под действием S^{ξ}_{φ} подразумеваем замену каждого входа переменной ξ в формуле ψ формулой φ. Если переменная y_k входит и в выражение $f_k(x_{i_1}, \ldots, x_{i_n})$, то осуществляем операцию следующим образом: пусть t переменная, не входящая ни в F_{j_1} и ни в $f_k(x_{i_1}, \ldots, x_{i_n})$. Такая переменная существует, так как число переменных бесконечно. Сначала заменяем каждый вход переменной y_k в формуле F_{j_1} переменной t и потом формула F_j получается операцией

$$F_j \equiv (S^{t}_{f_k(x_{i_1}, \ldots, x_{i_n})} F_{j_1}). \tag{9'}$$

Оператор B_3 осуществляется в том случае, когда L_j является оператором A_k, переменная y_k не входит в F_{j_1}, но она является конечной величиной. В этом положений этот оператор строит следующую формулу F_j

$$F_j \equiv [(y_k = f_k(x_{i_1}, x_{i_2}, \ldots, x_{i_n})) \wedge F_{j_1}]. \tag{10}$$

Оператор B_4 осуществляется в тех же условиях, как и оператор B_2 за исключением последнего условия, которое в этом случае состоит в том, что переменная y_k не зависит только от исходных данных. Он строит формулу F_j следующего вида

$$F_j \equiv \{(Ey_k)[(y_k = f_k(x_{i_1}, x_{i_2}, \ldots, x_{i_n})) \wedge F_{j_1}]\}. \tag{11}$$

Наконец, оператор B_5 осуществляется в том случае, когда F_j является логическим оператором $P_l(x_{i_1}, \ldots, x_{i_n})$ и он строит формулу F_j вида

$$F_j \equiv [(P_l(x_{i_1}, \ldots, x_{i_n}) \supset F_{j'}) \wedge (\neg P_l(x_{i_1}, \ldots, x_{i_n}) \supset F_{j_1})] \tag{12}$$

где $F_{j'}$ является формулой, построенной при анализе узла $L_{j'}$, к которому направляется стрела, обозначенная знаком плюс, из узла L_j, а $F_{j''}$ является формулой, построенной при анализе узла $L_{j''}$, к которому направляется из узла L_j стрела, обозначенная знаком минус.

Логический оператор p_4 проверяет имеет ли переменная j значение $p+1$. В положительном случае действие продолжает свой путь по стреле со знаком + к выходу, а в противном случае, -по стреле со знаком минус к оператору B_6.

Оператор B_6 увеличивает значение переменной j на одну единицу и передаёт действие оператора B_1 для анализа следующего узла граф-схемы алгорифма $\mathcal{A}$.

При выходе граф-схемы $\mathcal{G}$ извлекается формула F_{p+1}.

Как стало видно из описания интерпретации и действия алгорифма $\mathcal{B}$, он анализирует каждый узел граф схемы Г алгорифма $\mathcal{A}$ вместе с соответствующей интерпретацией, начиная с выхода схемы к входу. Это было сделано для упрощения изложения алгорифма $\mathcal{B}$. С таким же успехом могли бы производить анализ от входа к выходу, приходя к той же формуле F', но изложение было бы сложнее.

Как мы уже показали, формула F' аналогична формуле F, следовательно, более точно формулируя нашу мысль, формула F' выражает реляции, при помощи которых, алгорифм добивается конечных данных, исходив из начальных данных. После этого уточнения, то, что импликация (8) верна значит, что между этими реляциями находятся и реляции, выражённые формулой F.

Тот факт, что формула F_{p+1} удовлетворяет этой необходимости, непосредственно вытекает из формул (9), (10), (11), и (12). Можно дать строгое доказательство этому факту при помощи индукции относительно числа узлов граф-схемы, имея в виду, что каждый узел является либо α-узлом либо β-узлом и применяя формулы (9), (10), (11) и (12).

ЛИТЕРАТУРА

[1] Дьяченко В. Ф., *Построение граф-схем алгоритмов*, Проблемы передачи информации, 12 (1963).
[2] Ершов А. П., *Операторные алгорифмы*, Проблемы кибернетики, 3 (1960).
[3] Янов Ю. И., *О логических схемах алгоритмов*, Проблемы кибернетики, 1 (1958).

[4] Калужнин Л. А., *Об алгоритмизации математических задач*, Проблемы кибернетики, 2, (1959).
[5] Karp R. M., *A note on the application of graph theory to digital computer programming*, Inf. and Control, 3, 2 (1960).
[6] Королюк В. С., *О понятии адресного алгоритма*, Проблемы кибернетики, 4, (1960).
[7] Ляпунов А. А., *О логических схемах программ*, Проблемы кибернетики. 1 (1958).
[8] Подловченко Р. И., *О преобразованиях схем программ и их применении в программировании*, Проблемы кибернетики, 7, (1962).
[9] Prosser R. T., *Application of Boobean matrices to the analysis of flow diagrams*, Proc. Eastern Joint Computer Conf. (1959).
[10] Тер-Захрян Н. П., *О нормализуемости граф-схемных алгорифмов*, Труды вычислительного центра А. Н. Армянской ССР и Ереванского университета, 1(1963).
[11] Вязалов Л. Х., Морозов Ю. И., *Вспомогательная стандартная программа для отладки программ*, Проблемы кибернетики, 6 (1961).

Поступило 15. VII. 1963

MATHEMATICA VOL. 5 (28), 2, 1963, pp. 261–277

PROPRIÉTÉ SPÉCIALE DE LA DÉRIVÉE LOGARITHMIQUE DE LA FONCTION $\Gamma(z)$

par

FRANTIŠEK NOŽIČKA

Praha

Le but du travail présent est l'application de certaines propriétés de la fonction $\Gamma(z)$ — précisément de la dérivee logarithmique de cette fonction. La fonction

$$\psi(z) = -K + \sum_{k=1}^{\infty}\left(\frac{1}{z-k}+\frac{1}{k}\right), \tag{1}$$

où K signifie la constante d'Euler (qui est en relation étroite avec la dérivée logarithmique de $\Gamma(z)$), possède quelques propriétés importantes, qui permettent la sommation des certaines séries infinies. A l'aide de la fonction $\psi(z)$ on peut encore parvenir à des relation nouvelles (à des formules récurrentes) pour la fonction $\zeta(z)$.

§ 1.

Notions préliminaires.

La fonction $\pi \operatorname{cotg} \pi z$, laquelle peut s'utiliser pour sommer les séries spéciales, possède les propriétés élémentaires (bien connues) suivantes :

1. *La fonction* $\pi \operatorname{cotg} \pi z$ *est une fonction méromorphe dans le plan complexe ouvert* E ;

2. *elle admet seulement des pôles simples aux points* $z = k$ (k *nombre entier*) *avec les résidus*

$$\operatorname*{res}_{z=k} \pi \operatorname{cotg} \pi z = 1 \quad \textit{pour } k = 0, \pm 1, \pm 2, \ldots;$$