ТЕОРИЯ ВЫЧИСЛИМОСТИ И ЯЗЫКИ СПЕЦИФИКАЦИЙ
(Вычислительные системы)
1991 год Выпуск 139

УДК 519.685+510.52

# О СТРАТЕГИЯХ ИСПОЛНЕНИЯ Σ-ПРОГРАММ

М.Ю.Трифонов

## В в е д е н и е

В последнее время была предложена и развита концепция семантического программирования, в основе которого лежат теория моделей и теория допустимых множеств. Одним из уточнений семантического программирования является Σ-программирование [2]. В Σ-программировании за основу берется некоторая модель $\mathcal{M}$, которая понимается как базовая, а функции и предикаты на этой модели - как встроенные. Наряду с моделью $\mathcal{M}$ рассматривается списочная надстройка над $\mathcal{M}$ - списки элементов $\mathcal{M}$, списки списков и т.д., на которых введены операции head, tail, cons и предикаты "$\in$" и "$\sqsubseteq$". На этом языке просто и естественно записываются свойства многих конструкций логического программирования [1,4]. Расширением Σ-программ являются $\Sigma^+$-программы [3].

Σ-программирование основано на тезисе "вычислимость = Σ-определимости". Это означает, что класс всех вычислимых предикатов допускает эффективное представление в виде Σ-формул. Кроме того, любой набор спецификаций, заданных в виде Σ-формул, имеет процедурную семантику и может считаться программой. Исполнение Σ-программы заключается в проверке истинности запроса на модели. В качестве запроса выступает также Σ-формула.

При этом еще не рассматривалась задача стратегии исполнения программ. Действительно, если запрос имеет вид A & B, то заранее неясно, в каком порядке эффективнее(?!) исполнять подзапросы (параллельное исполнение неспособно разрешить проблему эффективности в принципе, так как существуют конструкции, допускающие бесконечное ветвление, например, рекурсивные предикаты), не затронут МЕТАуровень Σ-программ и другие вопросы, встающие при практическом исполнении. Данной теме и посвящена настоящая работа.

Ввиду вышеизложенного работа во многом носит постановочный характер. Однако ряд конкретных частей (например, алгоритмические) программно реализован автором.

В § 1 вводится понятие системы с информационным обменом, позволяющее проиллюстрировать общий вид системы искусственного интеллекта, составные части которой обмениваются информацией в ходе вычислений, а также задачи, возникающие при этом.Конкретный вид такой системы, а именно, представление запроса в виде так называемой ориентированной блок-сети рассматривается в §2. В §3 и 4 вводятся характеристики, позволяющие оценивать эффективность порядка исполнения, индуцированного ориентированной блок-сетью.

## §1. Системы с информационным обменом

Рассмотрим работу какой-либо системы, сложенной из составных частей таких, что каждая часть при определенных условиях может производить работу в то время, как остальные части бездействуют. Назовем каждую составную часть функционером данной системы. На вход системы подается некоторая начальная ситуация и подразумевается, что на выходе появится некоторая конечная ситуация. Примером системы может быть технологический цикл по производству табурета с помощью дерева, краски, клея, инст-

рументов и человека (функционерами в этой системе могут служить операции: "выпилить ножки", "покрасить то, что получилось") либо система автоматического доказательства теорем (функционеры - правила вывода), либо система $\Sigma$-программирования (функционерами служат, например, процессы проверки истинности подзапросов на модели).

С каждым функционером связано "минимально необходимое состояние входных данных", при условии наличия которых он может работать. Например, функционер "покрасить табурет" не может работать, если на входе нет данных вида: "наличие краски" и "наличие табурета"; правило $\frac{\Phi \,\&\, \psi}{\psi}$ не может быть применимо, если на входе присутствует дизъюнкция и т.д.

Пусть для функционера задано "минимально необходимое состояние входных данных". Тогда он в процессе работы порождает выходные данные (покрашенный табурет, формулу, набор значений выходных переменных).

Работа функционера происходит с некоторой "эффективностью" (быстротой, объемом потребляемой памяти и т.д.). "Эффективность" этой работы зависит от двух моментов: во-первых, от присущих собственно функционеру особенностей (не зависящих от входных данных, работы других функционеров и др.), во-вторых, от дополнительной информации о процессе работы (например, о выходных данных, не являющихся "минимально необходимыми"). Действительно, если на входе функционера "покрасить табурет", кроме "наличия краски" и "наличия табурета" присутствует входное данное "наличие кисти", то работа будет производиться с большей эффективностью (быстрее и качественнее), хотя и наличие последнего входного данного, вообще говоря, не обязательно для работы функционера, ибо покрасить можно и пальцем. Этих дополнительных входных данных может быть много, как то: "наличие желания работать", "наличие помощника", причем мы будем исхо -

дить из предположения, что добавление разных дополнительных входных данных дает разную эффективность, и ее можно измерить, а также то, что функция ЭФФЕКТИВНОСТЬ(ВхДанные) является монотонно неубывающей.

Наконец, примем тот принципиальный для нашего исследования факт, что выходные данные работы одного из функционеров могут служить входными данными для других, при этом как "минимально необходимыми", так и нет. Ясно, что очередность исполнения функционеров имеет существенное влияние как на саму работу системы, так и на эффективность этой работы.

Итак, пусть $\mathcal{M}$ - многосортная модель, $\chi$ - множество переменных, $I: \chi^m \to |\mathcal{M}|^m$ - интерпретация наборов переменных. Назовем формулу $f(x_1, \ldots, x_n)$ функционером при начальных данных $I(x_{i_1}, \ldots, x_{i_k})$, если $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq \{x_1, \ldots, x_n\}$. Обозначим при этом $X_0(f) = \{x_{i_1}, \ldots, x_{i_k}\}$ - входные-1 аргумента функционера $f$, $X_1(f) = \{x_1, \ldots, x_n\} \setminus X_0$.

ОПРЕДЕЛЕНИЕ 1. Рассмотрим совокупность функционеров с их начальными данными: $f_1;I_1,\ f_2;I_2, \ldots,\ f_k;I_k$. Назовем это множество системой, если справедливы условия:

а) $(\exists i \quad x \in X_0(f_i)) \Rightarrow \forall j \quad x \in X_1(f_j)$,

б) $(x \in (X_0(f_i) \cap X_0(f_j)) \ \& \ i \neq j) \Rightarrow I_i(x) = I_j(x)$.

Обозначим через $Z_0 = \bigcup_{i=1}^{k} X_0(f_i)$ входные аргументы системы, а через $Z_1 = \bigcup_{i=1}^{k} X_1(f_i)$ - выходные аргументы системы.

Пусть $Z = Z_0 \cup Z_1 = \{x_1, \ldots, x_l\}$ - аргументы системы.

Набор элементов модели $\mathcal{M}(a_1,\ldots,a_l)$ назовем ответом системы $f_1;I_1,\ldots,f_k;I_k$, если для всех $j = 1,\ldots,k$ проекция этого набора на набор свободных переменных $f_j$ делает эту формулу истинной.

Пусть $f_1;I_1,\ldots,f_n;I_n$ - система. Возьмем произвольную перестановку $s: (1,\ldots,n) \to (j_1,\ldots,j_n)$ и рассмотрим последовательность $SC = (f_{j1};I_{j1},\ldots,f_{jn};I_{jn})$. Свяжем в рамках этой последовательности $SC$ с каждым функционером $f_{jk}$ при его начальных данных два множества переменных:

$X_0^*(f_{jk},SC) = \{x_i \mid x_i \in X_1(f_{jk}) \;\&\; \exists jl\ (jl < jk)\ x_i \in X_1(f_{il})\}$ - входные-2 аргументы функционера $f_{jk}$,

$X_1^*(f_{jk},SC) = X_1 \setminus X_0^*$ - выходные аргументы функционера $f_{jk}$.

Пусть с каждым $f_i$ связано минимально необходимое множество переменных $Minbeg(f_i)$. Будем говорить, что функционер $f_i$ b-самостоятелен, если $Minbeg(f_i) \subseteq X_0(f_i,SC)$, и что p-самостоятелен, если $Minbeg(f_i) \subseteq X_0(f_i,SC) \cup X_0^*(f_i,SC)$.

Ясно, что если $f_i$ b-самостоятелен, то он и p-самостоятелен.

ОПРЕДЕЛЕНИЕ 2. Последовательность, введенная выше, называется системой с информационным обменом, если любой $f_{jk}$ в ней является p-самостоятельным.

Пусть $f_1;I_1,\ldots,f_n;I_n$ - система, $SC_1,\ldots,SC_m$ - различные системы с информационным обменом указанной системы; $E_1,E_2$ - частично упорядоченные множества.

ОПРЕДЕЛЕНИЕ 3. Назовем функцией эффективности функционера $f_k$ в системах с информационным обменом $SC_i$, $i = 1,\ldots,m$, отображение $Eff: (f_k,SC_i) \to E_1$, если справедливо: $Eff(f_k,SC_i) \geq Eff(f_k,SC_j)$ тогда и только тогда, когда $X_0^*(f_k,SC_i) \supseteq X_0^*(f_k,SC_j)$.

ОПРЕДЕЛЕНИЕ 4. Назовем функцией эффективности системы с информационным обменом $SC_i$, $i = 1,\ldots, m$, отображение $Efc: SC_i \to E_2$, если из того, что $Eff(f_k, SC_i) \geq Eff(f_k, SC_j)$ для всех $f_k$, следует, что $Efc(SC_i) \geq Efc(SC_j)$.

Легко видеть, что эффективность системы с информационным обменом есть величина, зависящая от порядка расположения (фактически - порядка исполнения) функционеров.

Исследование частных случаев системы с информационным обменом полезно с точки зрения выработки эффективных стратегий исполнения задач искусственного интеллекта, различных по своему виду. С другой стороны, изучение абстрактной системы поможет решать проблемы общей теории дедукции.

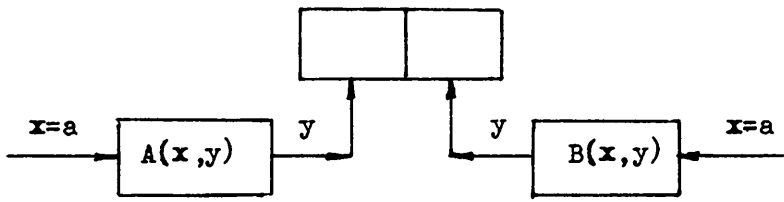
В настоящей работе рассматривается частный случай систем с информационным обменом как исполнение запроса $\Sigma^+$-программы [3]. При этом функционерами будут являться подзапросы, начальными данными - значения входных переменных, а системой с информационным обменом - детерминированная последовательность подзапросов. Будут предложены функции эффективности для данного случая и алгоритм выбора последовательности с наивысшей эффективностью.

## §2. Простые и ориентированные блок-сети

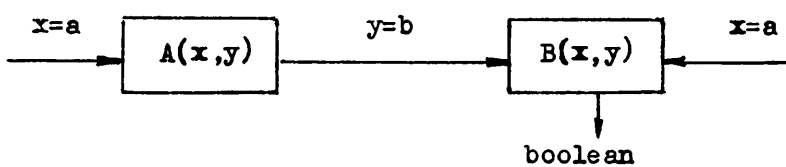
2.1. Исполнение подзапросов - варианты выбора. Рассмотрим работу системы с информационным обменом на примере исполнения $\Sigma^+$-программы. Напомним, что исполнение программы заключается в данном случае в проверке истинности запроса на базовой модели со списочной надстройкой. При этом запрос может иметь (и чаще всего имеет) сложную структуру, позволяющую, однако, исполнять запрос "по частям". Исполнение мы будем считать последовательным, хотя многие результаты могут быть применимы и в случае параллельного исполнения.

Пусть запрос имеет вид "найти такие $y$, что на базовой модели $\mathcal{M}$ истинна формула $A(x,y)\ \&\ B(x,y)$, где значение $x$ задано, $x = a$". Возможно не менее трех различных вариантов исполнения этой формулы.

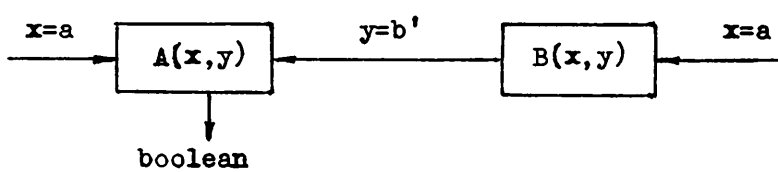
1. Оба члена конъюнкции исполняются независимо друг от друга, выходные значения $y$ сравниваются вплоть до совпадения:




2. Исполняется $A(x,y)$. Пусть $\mathcal{M} \vdash A(a,b)$. После этого аргумент $y$ во втором члене конъюнкции считается известным и равным $b$, происходит проверка истинности на модели формулы $B(a,b)$. Если $\mathcal{M} \vdash B(a,b)$, то ответ получен; если же формула ложна, то происходит возврат и ищутся другие значения переменной $y$:




3. То же, но в начале исполняется второй член конъюнкции:



Возникает задача выбора порядка исполнения предикатов.Заметим, что такая задача встает и при отсутствии информационного обмена между подзапросами, например, если у различных под - запросов множества выходных переменных не пересекаются.Для этого небоходимо:

1) знать все возможные варианты исполнения запроса,

2) выработать критерии, позволяющие с достаточной досто - верностью заключать, какие из вариантов исполнения "эффектив - нее" других.

Если подзапросы не связаны между собой информационным обменом, т.е. не имеют общих выходных переменных или если запрос имеет вид дизъюнкции, то возможен, вообще говоря, любой поря - док исполнения. Пусть запрос имеет вид конъюнкции. Для запроса такого вида вводятся основные понятия простой и ориентирован - ной блок-сети, реализующие потенциальную возможность информа - ционного обмена и конкретные варианты введения такой структуры на запросе.

Заметим, что сходные задачи возникают во многих языках высокого уровня, например, в ПРОЛОГе при выяснении того, в каком порядке эффективнее проверять формулы из правой части запроса (см., в частности, [6]).

2.2. Основные определения. Пусть $P$ - $\Sigma^+$-формула (далее называемая запросом), $FV(P)$ - свободные переменные $P$, причем переменные разделяются согласно [5] на $In(P)$ - входные, $Out(P)$ - выходные, $Ext(P)$ - внешние переменные, где $In(P) \cup Out(P) \cup Ext(P) = FV(P)$, $In(P) \cap Out(P) = In(P) \cap Ext(P) = Out(P) \cap Ext(P) = \emptyset$.

Определим понятие подзапроса $P$.

Если $P$ - атомарная, то $P$ - подзапрос, причем единст - венный.

Если $P = \neg P_1$ или $P = \exists x.P_1$, или $P = \forall x.P_1$, $P = \exists x \in t.P_1$, , или $P = \forall x \in t.P_1$, то $P_1$ - подзапрос , тоже единственный.

Если $P = \&_{i=1}^{n} f_i$ либо $P = \vee_{i=1}^{n} f_i$, то $f_1,\ldots,f_n$ - подзапросы $P$, при этом $\&$ (соответственно $\vee$) называется ключевым знаком $P$.

Далее синтаксис формул соответствует [5].

ПРИМЕР 1. Если $P = (A(x,!y)\ \&\ (B(?z)\ \&\ \&\ \forall x.C(x,?z))$, то $\&$ - ключевой знак, $A(x,!y)$, $B(?z)$, $\forall x.C(x,?z)$ - подзапросы.

Если $P = ((A(!x) \vee B(t))\ \&\ \exists y.(C(y)\ \&\ \&\ T(y,?z)))$, то $\&$ - ключевой знак, $(A(!x) \vee B(t))$, $\exists y.(C(y)\ \&\ T(y,?z))$ - подзапросы.

Будем говорить, что

$x \in In(f_i)$, если $x \in In(P) \cap FV(f_i)$,

$x \in Out(f_i)$, если $x \in Out(P) \cap FV(f_i)$,

$x \in Ext(f_i)$, если $x \in Ext(P) \cap FV(f_i)$.

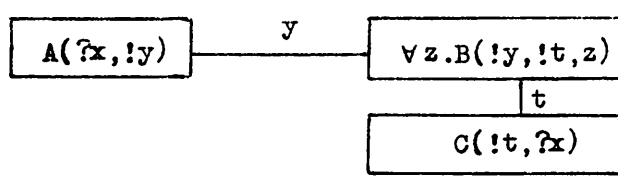
ОПРЕДЕЛЕНИЕ 5. Назовем простой дугой запроса $P$ пару $(\{A,B\},x)$, где $A$, $B$ - различные подзапросы $P$, $x \in \in Out(A) \cap Out(B)$.

ОПРЕДЕЛЕНИЕ 6. Назовем простой блок-сетью запроса $P$ (ПБС(P)) множество всех простых дуг запроса $P$.

ПРИМЕР 2. Если $P = (A(?x,!y)\ \&\ (\forall z.B(!y,!t,z)\ \&\ \&C(!t,?x)))$, тогда

$$ПБС(P) = \{(\{A(?x,!y),\forall z.B(!y,!t,z)\},y),\ (\{C(!t,?x),\forall z.B(!y,!t,z)\},t)\}.$$

Графическое изображение простой блок-сети:



Пусть $P$ - запрос, $P^*(P)$ = ПБС(P).

Введем несколько обозначений. Пусть $S_1, \ldots, S_n$ - все дуги простой блок-сети запроса $P$. Если при этом $S_i = (\{A,B\},x)$, то через $S_i(1)$ обозначим $\{A,B\}$, через $S_i(2) - x$, через $S_i(1,1)$ - $A$ и через $S_i(1,2)$ - $B$.

ЗАМЕЧАНИЕ. Хотя во множестве $\{A,B\}$ порядок заранее не фиксирован, но мы можем считать, что элементы расположены, например, по алфавиту.

Приведем два очевидных утверждения о природе простой блок-сети.

УТВЕРЖДЕНИЕ 1. *Пусть* $\{f_1, \ldots, f_n\} = F(P)$ - *все подзапросы* $P$, $\$ \in \{\&, \vee\}$ - *ключевой знак,* $N'(P) = \{f_{i_1}, \ldots, f_{i_k}\} \subset F(P)$. *Обозначим* $P^*(f_{i_1}, \ldots, f_{i_k}) = \{S_i \in P^*(P) \mid S_i(1,1) \in F'(P) \,\&\, S_i(1,2) \in F'(P)\}$. *Пусть также* $P_1 = \$_{i=1}^{k} f_{i_1}$, $P^*(P_1)$ - *ПБС*$(P_1)$. *Тогда* $P^*(P_1) = P^*(f_{i_1}, \ldots, f_{i_k})$.

ЗАМЕЧАНИЕ. Если $F''(P) = F(P) \setminus F'(P) = \{f_{j_1}, \ldots, f_{j_m}\}$, $P_2 = \$_{i=1}^{m} f_{j_1}$, то $P^*(P) \setminus P^*(P_1) \supseteq P^*(P_2)$, причем если $Out(P_1) \cap Out(P_2) \neq \emptyset$, то неравенство строгое.

УТВЕРЖДЕНИЕ 2. *Пусть* $P_1 = \$_{i=1}^{m} f_i, \ldots, P_n = \$_{k=1}^{s} f_k$, $\$$ *одинаково для всех. Обозначим* $F^\wedge = \{f_1, \ldots, f_t\} = \bigcup_{i=1}^{n} F(P_i)$. *Тогда* $P^*(F^\wedge) = (\bigcup_{i=1}^{n} P^*(P_i)) \cup P^\wedge(P_1, \ldots, P_n)$, *где* $P(P_1, \ldots, P_n) = \{$*все такие дуги* $S_i$, *что один из элементов множества* $\{S_i(1,1), S_i(1,2)\}$ *лежит в* $F(P_l)$ *и не лежит в* $F(P_m)$, *второй - наоборот; для некоторых* $l, m \in \{1, \ldots, k\}$ *и* $S_i(2) \in Out(S_i(1,1)) \,\&\, S_i(2) \in Out(S_i(1,2))\}$.

СЛЕДСТВИЕ. *В условиях утверждения 2* $P^*(P^\wedge) = \bigcup_{i=1}^{n} P^*(P_i)$ *тогда и только тогда, когда выполняются условия:*

а) $\forall i\ \forall j\ (i \neq j)\ N(P_i) \cap F(P_j) = \emptyset$ ,

б) $\forall i\ \forall j\ (i \neq j)\ Out(P_i) \cap Out(P_j) = \emptyset$ .

ОПРЕДЕЛЕНИЕ 7. Назовем ориентированной дугой запроса $P$ упорядоченную тройку $(A,B,x)$, где $A$, $B$ - различные подзапросы $P$, $x \in Out(A) \cap Out(B)$.

УТВЕРЖДЕНИЕ 3. *Если* $(\{A,B\},x)$ *- простая дуга запроса* $P$, *то* $(A,B,x)$ *и* $(B,A,x)$ *- ориентированные дуги запроса* $P$.

Для ориентированной дуги $(A,B,x)$ примем обозначения $0(1)$ - для $A$, $0(2)$ - для $B$, $0(3)$ - для $x$.

ОПРЕДЕЛЕНИЕ 8. Назовем ориентированной блок-сетью запроса $P$ (ОБС(P)) множество ориентированных дуг запроса $P$, если справедливы следующие условия.

1. Если $(A,B,x) \in$ ОБС(P), то

а) если существуют $f_i$ такие, что $(\{A,f_i\},x) \in$ $\in$ ПБС(P), то для всех $f_i$ справедливо $(A, f_i, x) \in$ $\in$ ОБС(P);

б) если существуют $f_i$ $f_j$ такие, что $(\{f_i,f_j\},x) \in$ $\in$ ПБС(P) и $A$ отлично от $f_i$, $f_j$, то $(f_i, f_j, x) \notin$ $\notin$ ОБС(P) и $(f_j, f_i, x) \notin$ ОБС(P).

2. Если $A$ - подзапрос $P$, $x \in Out(A)$, $\mathcal{F}_A(x) =$ $= f_i \mid (\{A,f_i\},x) \notin$ ПБС(P), то существует хотя бы один $f_i \in \mathcal{F}_A(x)$ такой, что $(A,f_i,x) \in$ ОБС(P) или $(f_i, A, x) \in$ ОБС(P) .

3. В ориентированной блок-сети запроса $P$ нельзя найти набор дуг $0_1,\ldots,0_k$ таких, что $0_1(2) = 0_2(1), 0_2(2) = 0_3(1),\ldots$ $\ldots, 0_{k-1}(2) = 0_k(1), \ldots, 0_k(2) = 0_1(1)$ одновременно.

ПРИМЕР 3. Если $P = ((A \& B) \& (C \& D))$, где $Out(A) = \{x,y,t\}$, $Out(B) = \{x,t\}$, $Out(C) = \{y,z\}$, $Out(D) = \{x,y\}$, то $ОБС_1(P) = [(A, B, x), (A,D,x), (A,C,y), (A,D,y), (A,B,t)]$ (рис.1а);

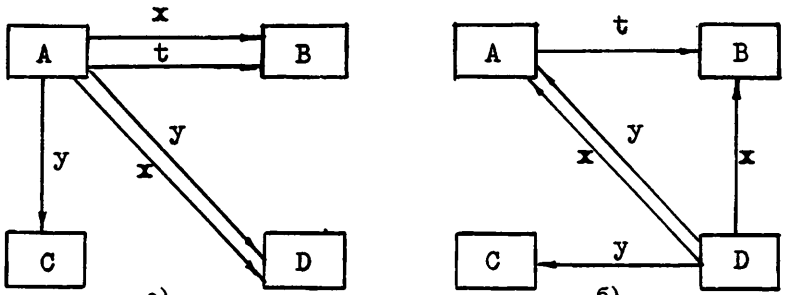



Рис. 1

либо, например, $ОБС_2(P) = [(D, B, x), (D, A, x), (A,B,t), D,A,y), (D,C,y)]$ (рис.1б).

Из примера ясно, что ориентированная блок-сеть запроса P, вообще говоря, неединственна. Построен алгоритм, который по известной блок-сети запроса P строит набор таких блок-сетей. Ниже мы приводим его описание.

ОПРЕДЕЛЕНИЕ 9. Назовем ориентацией простой дуги отображение $R : S_i \to \{O_{i1}, O_{i2}, \emptyset\}$, где, если $S_i = (\{A,B\},x)$, то $O_{i1} = (A,B,x)$ - левая ориентация, $O_{i2} = (B,A,x)$ - правая ориентация. Ориентация простой дуги $S_i$ называется пустой, если $R(P_i) = \emptyset$.

2.3. A1 - алгоритм порождения ориентаций простой блок-сети.

1-й шаг. $R(S_1) = O_{11}$. Если среди $S_2, \ldots, S_n$ есть такие $S_i$, что $S_i(2) = S_1(2)$, то все такие $S_i$ ориентируются следующим образом:

если $O_{11}(1) = S_i(1,1)$, то $R(S_i) = O_{i1}$,

если $O_{11}(1) = S_i(1,2)$, то $R(S_i) = O_{i2}$,

если $0_{11}(1) \neq S_i(1,1)$ и $0_{11}(1) \neq S_i(1,2)$, то $R(S_i) = \emptyset$.

Все ориентированные на этом шаге простые дуги назовем ориентированными на шаге 1, а $S_1$ - базовой дугой шага 1.

$k$-й шаг. Пусть ориентированы простые дуги на предыдущих шагах и существуют еще неориентированные дуги. Пусть $S_j$ - неориентированная простая дуга с наименьшим номером, $R(S_j) = 0_{j1}$. Если среди неориентированных дуг есть такие $S_i$, что $S_i(2) = S_j(2)$, то все такие $S_i$ ориентируются аналогично вышеизложенному на шаге 1, при этом все такие $S_i$ называются ориентированными на шаге $k$, а $S_j$ - базовой дугой шага $k$.

Полученное на шагах $1,\ldots,k$ множество ориентированных дуг проверим на удовлетворение условий 2 и 3 для ориентированной блок сети запроса $P$. Пусть оба условия удовлетворены. Тогда $k$-й шаг заканчивает свою работу переходом на шаг $k+1$. Если хотя бы одно условие не удовлетворяется, то происходит переход на шаг $k^*$.

$k^*$-й шаг. Ориентация, проведенная на шаге $k$, объявляется недействительной. Пусть $S_j$ - базовая дуга на шаге $k$. Проведем повторную ориентацию по закону: если было

$R(S_j) = 0_{j1}$, то $R(S_j) = 0_{j2}$,

$R(S_j) = 0_{j2}$, то $R(S_j) = \emptyset$,

$R(S_j) = \emptyset$, то переход на шаг $(k-1)^*$.

Если стало $R(S_j) = 0_{j2}$, то ориентируем все такие неориентированные дуги $S_i$ так, что $S_i(2) = S_j(2)$ по принципу:

если $0_{j2}(1) = S_i(1,1)$, то $R(S_i) = 0_{i1}$,

если $0_{j2}(1) = S_i(1,2)$ то $R(S_i) = 0_{i2}$,

если $0_{j2}(1) \neq S_i(1,1)$ и $0_{j2}(1) \neq S_i(1,2)$, то $R(S_i) = \emptyset$.

Теперь проверяем множество ориентированных дуг на удовлетворение условиям 2 и 3 ориентированной блок-сети запроса $P$. Если условия удовлетворяются, то происходит переход на шаг $k+1$, если нет - то еще одна попытка ориентации базовой дуги. Подобная ориентация (но без ориентации каких-то других дуг) происходит и в случае $R(S_j) = \emptyset$. Тогда $S_j$ считается ориентированным на шаге $k$ (и ориентированным пусто), перестает быть базовой дугой и ищется другая $S_p$ такая, что $S_j(2) = S_p(2)$ (с повторением шага $k$).

Алгоритм считается завершенным при переходе на шаг $0^*$.

Если при переходе на шаг $m$ выясняется, что неориентированных дуг нет, то ориентированной блок-сетью запроса $P$ объявляется множество дуг, ориентированных непустым образом. При этом происходит переход на шаг $(m-1)^*$ и алгоритм продолжает свою работу, порождая другие блок-сети запроса $P$.

ЗАМЕЧАНИЕ. Покажем, что алгоритм является достаточно гибким, чтобы учесть в своей работе дополнительные сведения по поводу очередности исполнения подзапросов.

Пусть из каких-то соображений задан заранее порядок исполнения некоторых подзапросов, например, "A должен выдавать значение переменной $x$ и передавать его в B". Это означает, что дуга $(A,B,x)$ принадлежит любой ориентированной блок-сети. Зная это, можно провести ориентацию еще нескольких дуг, согласно первому условию для ориентированных блок-сетей. Примем, что ориентация подобного вида происходит на шаге 0, и внесем незначительные изменения в алгоритм. Так как переход на шаг $0^*$ заканчивает работу, то ориентация сохраняется.

Пусть, наоборот, известно заранее, что, например, дуга $(A,B,x)$ не должна принадлежать блок-сети. Данное условие

можно считать дополнительным условием при проверке корректности в процессе построения ориентированных блок-сетей.

Интерес может представлять введение в алгоритм процедур, способных учитывать данные общего вида: "если T дает данные для B, то C должно выполняться перед D, но после E, а если наоборот, то...". Это может быть громоздким, но не нарушит основную идею алгоритма.

Покажем, что алгоритм A1 по данной простой блок-сети запроса P порождает хотя бы одну ориентированную блок-сеть, и, более того, любая такая блок-сеть запроса P будет получена на некотором шаге работы алгоритма.

Действительно, пусть перед исполнением шага III не осталось неориентированных дуг. Множество дуг, ориентированных непусто, удовлетворяет условиям 2 и 3 для ориентированной блок-сети запроса P по построению. Покажем, что первое условие также соблюдается.

ОПРЕДЕЛЕНИЕ 10. Назовем x-подсетью ориентированной блок-сети запроса P совокупность дуг из этой блок-сети вида $(A,B,x)$.

ЛЕММА. *Пусть для множества ориентированных дуг справедливы условия 2 и 3 для ориентированной блок-сети запроса P и для каждой x-подсети справедливо условие 1. Тогда условие 1 справедливо для всего множества ориентированных дуг.*

ДОКАЗАТЕЛЬСТВО непосредственно следует из формулировки условия 1. Рассмотрим какую-нибудь x-подсеть и выберем в ней базовую дугу $S_{i0}$. Она существует, так как в противном случае не выполнялось бы второе правило x-подсети. По построению условия 2 и 3 справедливы для всех $0_j$ из подсети таких, что $j > i0$. Пусть $S_j$ из x-подсети такой, что $j < i0$. Он ориентирован "пусто". Покажем, что если $A = 0_{i0}(1)$, то $S_j(1,2) \neq A$ и $S_j(1,2) \neq A$. Действи-

тельно, пусть это не так и, например, $S_j(1,1) = A$ и $S_j(1,2) = B$. Но тогда легко заметить, что для $B$ несправедливо условие 2. Противоречие. Для доказательства остается заметить, что ориентированная блок-сеть запроса $P$ однозначно определяется набором всех своих базовых дуг и их ориентаций. Из переборного характера алгоритма следует, что ни один такой вариант не будет пропущен.

2.4. Потоки подзапросов. Пусть $F(P) = \{f_1, \ldots, f_n\}$ - все подзапросы $P$.

ОПРЕДЕЛЕНИЕ 11. Подзапрос $f_j$ называется автономным, если для каждого $S_i \in$ ПБС(P) имеем $f_j \neq S_i(1,1)$ и $f_j \neq S_i(1,2)$.

ОПРЕДЕЛЕНИЕ 12. Упорядоченная n-ка $(f_{i1}, \ldots, f_{in})$ называется потоком подзапросов $P$ по ориентированной блок-сети $B$, если $s: (1 \ldots n) \to (i1, \ldots, in)$ - подстановка и справедливо условие

$$\forall il \forall im \, (\exists O_i \in B \, (f_{im} = O_i(2) \,\&\, f_{il} = O_i(1)) \to il < im).$$

Указанный объект конструктивно строится по любой ориентированной блок-сети запроса $P$.

А2 - алгоритм упорядочения подзапросов

1-й шаг. Сформируем множество выбора $M_1$ по принципу:

1) если $f_i$ автономный, то $f_i \in M_1$,

2) если $f_i$ не автономный и не существует $O_k \in B$ таких, что $O_k(2) = f_i$, то $f_i \in M_1$.

Выбираем в качестве $f_{i1}$ любой подзапрос из $M_1$.

k-й шаг. Пусть $f_{i1}, \ldots, f_{ik-1}$ уже выбраны, $O[k-1] = \bigcup_{i1 \le i1 \le ik-1} \{O_i \in B \mid O_i(1) = f_{i1}\}$, $O^* = O \setminus O[k-1]$.
Сформируем множество выбора $M_k$ по принципу:

1) если $f_i$ автономный и еще не выбран, то $f_i \in M_k$,

2) если $f_i$ не автономный и не существует $O_k \in O^*$ таких, что $O_k(2) = f_i$, то $f_i \in M_k$.

Выбираем в качестве $f_{ik}$ любой подзапрос из $M_k$.

Если вновь собранное множество выбора пусто, алгоритм заканчивает свою работу.

Легко показать, что алгоритм А2 корректен. Действительно, нуждается в доказательстве лишь тот факт, что любой подзапрос на некотором шаге попадает во множество выбора. Для автономных это очевидно. Пусть " $\Rightarrow$" - отношение на множестве подзапросов по принципу:

если $\exists O_i$: $O_i(1) = f_j$, $O_i(2) = f_k$, то $f_j \Rightarrow f_k$,

если $f_j \Rightarrow f_l$ и $f_l \Rightarrow f_k$, то $f_j \Rightarrow f_k$.

Пусть $f_j$ - неавтономный подзапрос, $Up(f_j) = \{f_i \mid f_i \Rightarrow f_j\}$. Ясно, что $f_j$ попадает во множество выбора тогда и только тогда, когда все элементы из $Up(f_j)$ уйдут в поток. Пусть $f_i \in Up(f_j)$. Если $f_i$ попадает во множество выбора, то на некотором шаге оно уйдет в поток. Тогда можно рассмотреть множество $Up(f_i)$ и т.д. Остается заметить, что "самые верхние" подзапросы попадут в $M_1$.

ЗАМЕЧАНИЕ. Множества выбора, введенные в алгоритме, иллюстрируют ситуацию, какие подзапросы готовы к исполнению в данный момент времени.

Ориентированная блок-сеть и поток подзапросов характеризуют возможную последовательность исполнения подзапросов. Если теперь будут найдены критерии, позволяющие оценивать эффективность каждой ориентированной блок-сети (потока запросов), то это позволит из множества вариантов исполнения выбрать наилучший с достаточной степенью достоверности.

## §3. Множества истинности

Пусть $P$ - запрос, $\{x_1, \ldots, x_n\} = Out(P)$, $F(P) = \{f_1, \ldots, f_m\}$ - конъюнктивные подзапросы. Чтобы запрос был исполнен успешно, необходимо, чтобы был найден (хотя бы один) набор $(a_1, \ldots, a_n)$ такой, что при подстановке $x_1 = a_1, \ldots, x_n = a_n$ все $f_i$ были бы истинны на базовой модели. Напомним, что исполнение происходит последовательно, и если аргумент $x_j$ означивается в результате исполнения подзапроса $f_k$ и этот же аргумент присутствует в других подзапросах, исполняющихся после $f_k$, то в них $x_j$ уже является входным с полученным значением.

Если подзапрос $f_1$ ложен на модели при данном наборе входных значений и все входные значения $f_1$ совпадают со входными значениями $P$, то запрос дает отрицательный ответ. Если же терпит неудачу подзапрос, получивший некоторые входные значения от подзапросов, исполнявшихся до него, то происходят возврат на место последнего происшедшего информационного обмена и поиск нового набора ответа. Заметим, что этот возврат на практике неизбежен, причем очень частый.

В данных условиях большое значение будет иметь факт: какое количество ответов может выдать каждый подзапрос при условии, что значения некоторых переменных известны (и как значения входных переменных запроса, и как означенные в процессе информационного обмена). Если количество ответов конечно (и достаточно мало), то соответствующее количество возвратов также будет конечно, но если это не так, то вполне возможна ситуация, при которой нужный ответ будет найден за бесконечно долгое время (в теоретическом смысле), т.е. никогда, даже если он и существует .

Заметим, что чем больше аргументов в подзапросе еще не известно, тем больше существует возможных ответов. Очевидно, что

наиболее тривиален в этом отношении случай, когда известны все переменные - происходит простая проверка на истинность, всегда имеющая один ответ. В любом случае важно так определить порядок выполнения подзапросов, чтобы "неисчерпаемых по ответам" было меньше.

Рассмотрим элементарный

ПРИМЕР 4. Пусть предикат maxinlist(l,x) - множество пар (конечный список натуральных чисел l, максимальный элемент в этом списке x). Пусть запрос F имеет вид:

$$F = ((A(!l) \,\&\, maxinlist(!l,!x)) \;\&\; B(!x)) \;,$$

и известно, что подзапросы A и B имеют бесконечное множество ответов. Существуют четыре варианта исполнения (рис.2).

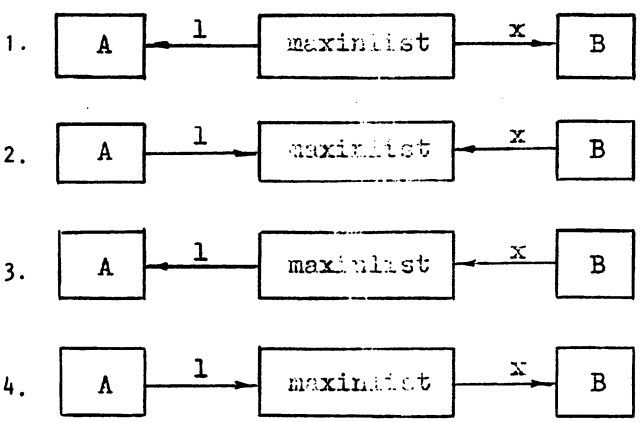



Рис.2

Будет логичным признать, что, вообще говоря, варианты 2 и 3 хуже, чем 1 и 4, так как содержат больше обращений к предикатам с бесконечным числом ответов.

Сложно дело обстоит и с рекурсиями, ибо возможны случаи бесконечного зацикливания. Эмпирически довольно трудно понять, что хуже - бесконечный поиск правильного ответа или бесконеч - ная подготовка к вычислению, сулящему, быть может, малое число ответов. Окончательные результаты здесь могут дать теоретические исследования вида рекурсий и практические эксперименты.

Еще один критерий - формальная сложность подзапросов. Так как перед исполнением формула приводится к плоскому виду (все термы вынесены), то имеет значение факт: сколько термов содержится в формуле.

В нашем случае запросом служит $\Sigma$-формула. Если формула имеет вид дизъюнкции или конъюнкции, то возникает вопрос: какой из членов исполнять первым, какой - вторым и т.д. В свою очередь, каждый член дизъюнкции (конъюнкции) может иметь вид конъюнкции (соответственно дизъюнкции) и т.п.

Пусть $\mathcal{M}$ - базисная модель, $\mathbf{F}$ - формула в известной сигнатуре, $\{x_1,\dots, x_n\} = FV(F)$.

ОПРЕДЕЛЕНИЕ 13. Назовем множеством истинности формулы $\mathbf{F}$ множество $T(F) = \left\{(a_1,\dots,a_n) \mid \mathcal{M} \vdash (F)^{x_1 \dots x_n}_{a_1 \dots a_n}\right\}$.

ОПРЕДЕЛЕНИЕ 14. Мощность множества $T(F)$ назовем абсолютной мощностью формулы $\mathbf{F}$ и будем обозначать эту величину $M(F)$.

ОПРЕДЕЛЕНИЕ 15. Назовем мощностью $\mathbf{F}$ при заданных переменных $x_{i1} = a_{i1},\dots, x_{im} = a_{im}$ мощность множества $T(F; x_{i1} = a_{i1},\dots, x_{im} = a_{im}) := \{(a_1,\dots, a_n) \mid \mathcal{M} \vdash (F)^{x_1 \dots x_n}_{a_1 \dots a_n}$ и таких, что если $j$: $1 < j < n$ и $j = ik$, $i1 < ik < im$, то $a_j = a_{ik}\}$. Будем обозначать эту мощность $M(F: x_{i1} = a_{i1},\dots, x_{im} = a_{im})$.

Пусть $\{x_{i_1},\dots, x_{i_k}\} \subseteq \{x_1,\dots, x_n\}$, $\{y_{j_1},\dots, y_{j_l}\} \subseteq \{x_{i_1},\dots, x_{i_k}\} \cap \{y_{j_1},\dots, y_{j_l}\} = \emptyset$.

Рассмотрим совокупность множеств и их мощностей:

$$T(F; x_{i_1}=a_{i_1},\dots,x_{i_k}=a_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l}),$$
$$M(F; x_{i_1}=a_{i_1},\dots,x_{i_k}=a_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l}),$$
$$T(F; x_{i_1}=b_{i_1},\dots,x_{i_k}=b_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l}),$$
$$M(F; x_{i_1}=b_{i_1},\dots,x_{i_k}=b_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l}),$$
$$\dots \qquad \dots \qquad \dots$$
$$T(F; x_{i_1}=u_{i_1},\dots,x_{i_k}=u_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l}),$$
$$M(F; x_{i_1}=u_{i_1},\dots,x_{i_k}=u_{i_k},y_{j_1}=k_{j_1},\dots,y_{j_l}=k_{j_l});$$
$$\dots \qquad \dots \qquad \dots$$

где $k_{j_1},\dots, k_{j_l}$ - заданный набор элементов модели и пробегаются все возможные на модели наборы аргументов переменных $x_{i_m}$.

ОПРЕДЕЛЕНИЕ 16. Будем называть мощностью $F$ при фиксированных переменных $x_{i_1},\dots,x_{i_k}$ и заданных переменных $y_{j_1}= k_{j_1},\dots, y_{j_l}= k_{j_l}$ suprerum введенных выше множеств. Обозначим эту величину $M(F; x_{i_1},\dots,x_{i_k}; y_{j_1}= k_{j_1},\dots, y_{j_l}= k_{j_l})$.

Введем обозначение $\{ T^*(F; y_{j_1} = k_{j_1},\dots, y_{j_l} = k_{j_l}) := T(F; x_{i_1},\dots, x_{i_s}; y_{j_1} = k_{j_1},\dots, y_{j_l} = k_{j_l})$ таких, что $(x_{i_1},\dots, x_{i_s})$ пробегает все наборы (всех размерностей) переменных из множества $FV(F) \setminus \{ y_{j_1},\dots, y_{j_l}\}\}$.

Зададим отношение "$\geq_*$" на множестве $T^*$ следующим образом:

$$T(F; x_{i_1},\dots, x_{i_k}; y_{j_1}= k_{j_1},\dots, y_{j_l}= k_{j_l}) \geq_*$$

$\geq_*$ $T(F; x_{r1},\ldots, x_{rm}; y_{s1}= k_{s1},\ldots, y_{sn}= k_{sn})$,

если выполняется одно из условий:

1) мощности соответствующих множеств конечны и мощность первого множества больше либо равна мощности второго множества,

2) мощность первого множества счетна, мощность второго множества конечна,

3) мощности обеих множеств счетны, но при этом $\{y_{j1},\ldots, y_{jl}\} \subseteq \{y_{s1},\ldots, y_{sn}\}$, у одинаковых переменных заданные значения совпадают и $\{x_{i1},\ldots, x_{ik}\} \subseteq \{x_{r1},\ldots, x_{rn}\}$.

УТВЕРЖДЕНИЕ 4. *Для любого заранее известного набора переменных и их значений множество* $T^*(F; y_{j1}= k_{j1},\ldots, y_{jl}= k_{jl})$ *является частично упорядоченным по отношению* "$\geq_*$" *с наибольшим элементом* $M(F; y_{j1}= k_{j1},\ldots, y_{jl}= k_{jl})$ *и наименьшим элементом* $M(F; FV(F)\setminus\{y_{j1},\ldots, y_{jl}\}; y_{j1}= k_{j1},\ldots, y_{jl}= k_{jl})$.

УТВЕРЖДЕНИЕ 5. *Для любого набора заданных переменных множество* $M^*$ *конечно и насчитывает* $2^{N-L}$ *элементов, где* N *- количество свободных переменных формулы,* L *- количество заданных переменных.*

ДОКАЗАТЕЛЬСТВО. Пусть $M = N-L$. Тогда искомое число равно

$$1+ C_M^M + C_M^{M-1} + \ldots + C_M^2 + C_M^1 = 1+ (2^M - 1) = 2^M.$$

ОПРЕДЕЛЕНИЕ 17. Если $M(F; x_{i1},\ldots, x_{is}; y_{j1} = k_{j1},\ldots, y_{jl} = k_{jl})$ - счетное и supremum из определения 16 достигается на некотором элементе, то говорится, что формула имеет собственно-счетную мощность, в противном случае - предельно-счетную мощность.

ПРИМЕР 5. Пусть $F(x,y,z) = Plus(x,y,z)$ - простое сложение, где z - сумма, основное множество модели - натуральные числа. Тогда:

M(Plus; x,y) = 1,

M(Plus; x=2,z=1) = 0,

M(Plus; x.z) = 1,

M(Plus; y.z) = 1,

M(Plus; x=4) - собственно-счетно,

M(Plus; x) - собственно-счетно,

M(Plus; y) - собственно-счетно,

M(Plus; z=4) = 5,

M(Plus; z=8) = 9,

M(Plus; z) -предельно-счетно,

M(Plus) - собственно-счетно.

ПРИМЕР 6. Пусть **maxinlist(l,x)** - предикат, выдающий список **l** и максимальный элемент этого списка **x** (модель - списочная надстройка над множеством натуральных чисел). Тогда:

M(maxinlist; l = (0,1,2)) = 1 .

M(maxinlist; l) = 1,

M(maxinlist; x=2) - собственно-счетно,

M(maxinlist; x) - собственно-счетно,

M(maxinlist) - собственно-счетно.

## §4. Критерии сложности ориентированной блок-сети и потоков подзапросов

Пусть $P$ - запрос, $f_1, \ldots, f_n$ - подзапросы, $B_i(P)$ - ориентированная блок-сеть запроса $P$, $C_jB_i(P)$ - поток подзапросов, построенный по $B_i$, $C_jB_i(P) = (f_{i1}, \ldots, f_{in})$.

ОПРЕДЕЛЕНИЕ 18. Определим для каждого подзапроса $f_{ik}$ из потока два множества:

1) $I1(f_{ik})$ - множество фиксированных переменных

$$x \in I1(f_{ik}) \leftrightarrow \exists il\,|\,i1 \le il < ik \;\&\; (f_{il}, f_{ik}, x) \in B_i(P);$$

2) $I2(f_{ik})$ - множество заданных переменных

$$x \in I2(f_{ik}) \leftrightarrow x \in In(f_{ik}).$$

Эти множества являются аналогом множеств $X_0^*$ и $X_0$ из §1.

Поставим в соответствие каждому потоку подзапросов набор числовых параметров: $(RC, RC_{i1}, \ldots, RC_{in}, AR, M, M^*, M1, M2, M1^*, M2^*, S_{i1}, \ldots, S_{in})$, где

RC - общее число рекурсивных предикатов, встречающихся в запросе;

$RC_{ik}$ - число рекурсивных предикатов, встречающихся в подзапросе $f_{ik}$ (степень рекурсивности $f_{ik}$);

$$AR = \sum_{i=1}^{n} \sum_{k=1}^{i-1} (RC_k - RC_i);$$

$M^*$ - количество подзапросов $f_{ik}$ таких, что $M(f_{ik}; I1(f_{ik}); I2(f_{ik}) = In(f_{ik}))$ - счетно (М-условие);

$M1^*$ - количество подзапросов таких, что выполняется М-условие, а счетность - собственная;

$M2^*$ - то же, но счетность предельная;

M - количество подзапросов $f_{ik}$ таких, что выполняется М-условие и имеет место свойство (для $f_{ik}$): $\exists x\, \exists im\, (ik < im \le in \;\&\; (f_{ik}, f_{im}, x) \in R_i(P))$;

M1, M2 аналогичны $M1^*$, $M2^*$ с учетом характера счетности;

$S_{i1}, \ldots, S_{in}$ - количество термов (отличных от переменных) в подзапросах $f_{i1}, \ldots, f_{in}$.

ОПРЕДЕЛЕНИЕ 19. Будем говорить, что поток подзапросов $C_j B_i(P)$ превосходит по Т-параметрам поток подзапросов $C_k B_l(P)$, если: $M1(C_j B_i) < M1(C_k B_l)$ (превосходит

по первому параметру) либо $M1(C_jB_i) = M1(C_kB_l)$ и $AR(C_jB_i) < AR(C_kB_l)$ (превосходит по второму параметру), либо $M1(C_jB_i) = M1(C_kB_l)$ и $AR(C_jB_i) = AR(C_kB_l)$ и если $ik$ - первый номер такой, что $S_{ik}(C_jB_i) \neq S_{ik}(C_kB_l)$, то $S_{ik}(C_jB_i) < S_{ik}(C_kB_l)$ (превосходит по дополнительному параметру).

Примем также, что если такого $ik$ не существует, то первый поток превосходит по $T$-параметрам второй, равно как и второй превосходит по $T$-параметрам первый.

УТВЕРЖДЕНИЕ 6. *Параметры* $RC$, $S_j$ *определяются запросом,* $M$, $M1$, $M2$, $M^*$, $M1^*$, $M2^*$ *- по ориентированной блок-сети,* $RC_i$, $AR$ *- конкретным потоком подзапросов.*

В предыдущем параграфе приведен алгоритм, дающий по известной ориентированной блок-сети поток подзапросов. Порождая различные потоки и высчитывая их параметры, находим наилучший.

Заметим, что отношение "превосходит по $T$-параметрам" ($\geq_t$) устанавливает линейный порядок на множестве всех потоков подзапросов всех ориентированных блок-сетей запроса $P$.

Покажем, что если существует знание о мощности и характере рекурсивности всех предикатов, то существует алгоритм, дающий по запросу $P$ поток подзапросов, наибольший по отношению $\geq_t$.

Действительно, пусть ключевой знак запроса - конъюнкция, $B_1, \ldots, B_r$ - все возможные ориентированные блок-сети запроса $P$, построенные по алгоритму А1, приведенному в § 2; $M' = \min_j M(B_j)$, где $M(B_j)$ считается путем просмотра подзапросов при условии знания о мощностях при фиксированных (выходных из более ранних подзапросов) и входных переменных, $\{B_{i_1}, \ldots, B_{i_s}\}$ - такие, что $M(B_{ik}) = M'$, $\{C_0B_{i_1}, \ldots, C_0B_{i_s}\}$ - соответствующие наилучшие потоки

подзапросов, $AR' = \min_{i1 \le it \le is} AR(C_0B_{it})$, $\{C_0B_{j1}, \ldots, C_0B_{jn}\}$ - такие, что $AR(C_0B_{jt}) = AR'$. Тогда поток, превосходящий остальные по дополнительному параметру, является искомым.

Если ключевой знак запроса - дизъюнкция, то, как было замечено в § 1, информационного обмена не происходит, следовательно, построение ориентированной блок-сети и знание о мощности не нужно. В этом случае наилучший поток определяется по второму и дополнительному параметрам и элементарно строится "от простого подзапроса (в смысле рекурсии и термальной сложности) к сложному".

Здесь следует сделать важное замечание.

В этом параграфе введены ряд параметров, которые, по мнению автора, способны в некоторой степени аппроксимировать понятие "сложность исполнения $\Sigma$-программы". Эти параметры, равно как и отношение $\geq_t$ есть попытка создания некоторых эвристик, и, безусловно, нет абсолютной гарантии, что они окажут действительно серьезное влияние на улучшение эффективности исполнения. В самом деле, например, считается верным, что если запрос имеет вид $\&_{i=1}^{n} A_i$, мощности $A_i$ равны и $A_n$ является наихудшим в смысле рекурсии, то его нужно исполнять в самом конце. Это утверждение доказать теоретически вряд ли представляется возможным, скорее всего, для некоторых запросов это будет так, а для некоторых - совершенно наоборот. То же можно сказать и о других параметрах. Невозможно доказать, почему количество подзапросов с бесконечной мощностью является более существенным, чем расположение рекурсий. Принципиально различаются конечные и счетные мощности, хотя в практическом смысле мало отличаются факты "предикат $A$ имеет бесконечное множество ответов" и "предикат $A$ имеет $10^{10!}$ ответов". По-

добные сложностные оценки с огромным трудом поддаются матема - тической формализации, а их значимость проявляется в конкрет - ных практических машинных экспериментах. В настоящее время автором готовится пакет эвристик различного вида с целью проверки их эффективности. Только после машинного эксперимента можно будет говорить о данных сложностных параметрах с большой уверенностью как о реально существующем.

Следует учесть, что заранее не всегда существует знание о мощностях отдельных предикатов. Хотя некоторые величины удается удовлетворительно оценить сверху, это не всегда возможно.Поэтому имеет смысл ввести еще два дополнительных параметра: $N$ - количество подзапросов таких, что их мощность неизвестна, и соответствующее $N^*$.

## З а к л ю ч е н и е

Данная работа является первым этапом в исследовании стратегии исполнения $\Sigma^+$-программ. Введенные в ней характеристики сложности ("$T$-параметры") и их градация по степеням значимости являются в некоторой степени эмпирическими (первые - менее, вторые - более), хотя, безусловно, и отражающими практический характер вычислений. Окончательные ответы на эти вопросы должны дать машинные эксперименты.

Дальнейшие исследования по теме стратегии будут продолжены такими следующими направлениями, как:

1) общетеоретическое:

а) анализ систем с информационным обменом,

б) применение МЕТАуровня к решению задач $\Sigma$-программирования;

2) конкретно-практическое:

а) поиск и исследование характеристик сложности,

б) усложнение базовой модели и других уровней системы,влияние этих изменений на стратегию.

В заключение автор выражает благодарность Д.И.Свириденко и С.В.Котову за помощь при подготовке работы.

## Литература

1. ГОНЧАРОВ С.С. Теория списков и ее модели // Логические вопросы теории типов данных.- Новосибирск.- 1986. - Вып. 114: Вычислительные системы.- С.84-95.

2. ГОНЧАРОВ С.С., СВИРИДЕНКО Д.И. $\Sigma$-программирование // Логико-математические проблемы СМОЗ.- Новосибирск. - 1985. - Вып.107: Вычислительные системы.- С.3-29.

3. Их же. $\Sigma^+$-программы и их семантика //Логические методы в программировании.- Новосибирск.- 1987.- Вып.120: Вычислительные системы.- С.24-51.

4. СВИРИДЕНКО Д.И. Об одном варианте теории списочных надстроек //Прикладная логика.- Новосибирск.-1986.-Вып.116: Вычислительные системы.- С.67-79.

5. КОТОВ С.В. К операционной интерпретации $\Sigma^+$-программ. Язык $\Gamma$-процедур// Системология и методологические проблемы информационно-логических систем.- Новосибирск.-1990.- Вып.135: Вычислительные системы.- С.131-159.

6. SMITH D.E., GENESERETH M.R. Ordering conjuntive queries //Artificial Intell.- 1985.- № 26.- P.171-215.

Поступила в ред.-изд.отд.
18 июля 1991 года