АНАЛИЗ ВРЕМЕННЫХ РЯДОВ И СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ
(Вычислительные системы)
1991 год Выпуск 141

УДК 519.68

УНИФИКАЦИЯ: АЛГОРИТМИЧЕСКИЙ АСПЕКТ

Н.А.Чужанова

Для двух термов, включающих функциональные символы и переменные, унификация в ее классическом смысле состоит в нахождении подстановок вместо переменных, делающих два терма эквивалентными.

Эквивалентность двух термов может рассматриваться и в рамках некоторой эквациональной теории, т.е. при заданной системе аксиом.

Впервые о практическом использовании унификации было сообщено Робинсоном [1]. Унификация являлась основной операцией предложенного им метода резолюций, используемого для автоматического доказательства теорем. В настоящее время область использования унификации широка - это распознавание образов, компьютерная алгебра, экспертные системы и т.д. (обзор применения унификации можно найти в [2]).

В данном обзоре мы рассмотрим разрешимость проблемы унификации в различных теориях, а также наиболее известные алгоритмы унификации и ее частного случая - идентификации (matching), не использующие типизированных переменных.

1. Определения

Пусть F - множество функциональных символов, не содержащих равенства; V - счетное множество переменных ($F \cap V = \emptyset$).

Множество термов T определяется следующим образом:

а) каждая переменная есть терм;

б) для любого n-местного функционального символа f и термов $t_1, \ldots, t_n$ $f(t_1, \ldots, t_n)$ есть терм (0-местный функциональный символ есть терм, обозначающий константу). Если терм не содержит переменных, то он называется базисным.

Подстановка σ - это эндоморфизм T в T такой, что $\sigma(x) \neq x$ на конечном множестве x-ов. Подстановку будем обозначать как конечное множество $\{t_1/x_1, \ldots, t_n/x_n\}$ или $\{x_1 := t_1, \ldots, x_n := t_n\}$, где x_i - переменная, t_i - терм, отличный от x_i, и все x_i различны. Если $t_1, \ldots, t_n$ - базисные термы, то подстановка называется базисной. Множество всех подстановок обозначим через Σ.

Под равенством понимается формула вида $s = t$, где $s, t \in T$.

Заданная система равенств (аксиом) $E = \langle s_i = t_i \rangle$ определяет эквациональную теорию, т.е. множество всех равенств, выводимых из E посредством хорошо известных правил обращения с равенствами.

Отношение равенства, определяемое E, есть наименьшее отношение конгруэнтности, содержащее E и замкнутое относительно подстановок. Отношение равенства, порождаемое E, будем обозначать $=_E$.

Подстановка $\sigma \in \Sigma$ называется E-унификатором термов s и t, если $\sigma s =_E \sigma t$. В случае пустого множества аксиом ($E = \emptyset$) подстановка σ называется $\emptyset$-унификатором или просто унификатором, если $\sigma s = \sigma t$. Термы s и t называются унифицируемыми, если для них существует унификатор.

Пусть $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ и $\lambda = \{u_1/y_1, \ldots, u_m/y_m\}$ - две подстановки. Композиция $\theta \circ \lambda$ - это подстановка, получаемая из множества $\{t_1\lambda/x_1, \ldots, t_n\lambda/x_n, u_1/y_1, \ldots, u_m/y_m\}$ вычеркиванием всех элементов $t_j\lambda/x_j$, для которых $t_j\lambda = x_j$, и всех элементов u_i/y_i таких, что $y_j \in \{x_1, \ldots, x_n\}$.

Унификатор σ называется наиболее общим унификатором (mgu) термов s и t, когда для любого унификатора θ существует подстановка λ такая, что $\theta = \sigma \circ \lambda$.

Частным случаем унификации является идентификация (matching), именуемая иногда односторонней унификацией.

Термы s и t называются идентифицируемыми тогда и только тогда, когда существует подстановка σ такая, что $\sigma s =_E t$ (или $\sigma s = t$).

Проблема унификации (идентификации) состоит из двух частей и формулируется следующим образом:

а) Является ли рекурсивно разрешимой проблема унифицируемости (идентифицируемости) или неунифицируемости (неидентифицируемости) термов s и t?

б) Если да, то как сгенерировать все возможные (или только наиболее общие) унификаторы (идентификаторы) σ такие, что $\sigma s =_E \sigma t (\sigma s =_E t)$?

Ответ на первый вопрос зависит от эквациональной теории. В случае пустой теории процедура определения унифицируемости является рекурсивно разрешимой и существуют эффективные алгоритмы генерации унификаторов. Рассмотрим более подробно результаты в этих областях.

2. Разрешимость проблемы унификации

Как уже отмечалось, ответ на вопрос, является ли рекурсивно разрешимой проблема унификации, зависит от эквациональной теории. Так, например, в эквациональной теории, представляю -

щей арифметику Пеано, проблема унификации в точности соответствует десятой проблеме Гильберта и, следовательно, она рекурсивно неразрешима.

Рассмотрим некоторые специальные теории, представленные комбинацией следующих аксиом:

A - ассоциативность $f(f(x,y),z) = f(x,f(y,z))$;

C - коммутативность $f(x,y) = f(y,x)$;

D_L - левая дистрибутивность $f(x,g(y,z))=g(f(x,y), f(x,z))$;

D_R - правая дистрибутивность $f(g(x,y),z)=g(f(x,z), f(y,z))$;

D - дистрибутивность $D_R + D_L$;

I - идемпотентность $f(x,x) = x$;

U - существование единицы $f(1,x) = f(x,1) = x$.

В табл.1 представлены теории, в которых проблема унификации рекурсивно разрешима, в табл.2 - теории, в которых проблема унификации рекурсивно неразрешима; в теориях D, D + C, D + U, D_L + A, D_L + U вопрос разрешимости остается открытым (приведенные таблицы заимствованы из [3]).

Таблица 1

Теории	Ссылки
∅	[1,4]
A	[5,6]
C	[6]
D_L, D_R, U	[7]
A + C, A + C + I	[9,10]
A + I, D + A + I	[8]

Таблица 2

Теории	Ссылки
D + A, D + A + C	[8]
D_L + A + U	[7]

3. Алгоритмы Ø-унификации

В данном разделе рассмотрим три алгоритма Ø-унификации и приведем оценки их сложности.

Алгоритм Робинсона [1] основан на поиске рассогласований в термах. Множество рассогласований D непустого множества термов S определяется следующим образом. Находим первую слева позицию, в которой не для всех термов из S стоит один и тот же символ. Выписываем из каждого терма наименьший подтерм, начинающийся в этой позиции. Множество таких подтермов и является множеством рассогласований D термов S для данной позиции.

Алгоритм унификации имеет следующий вид:

S - множество термов;

σ - унификатор для S.

Шаг 1. $k = 0$, $S_k = S_0 = S$, $\sigma_k = \sigma_0 = \emptyset$.

Шаг 2. Если S_k - единичное множество, то останов: σ_k - наиболее общий унификатор для S. В противном случае находим множество рассогласований D_k для S_k

Шаг 3. Если существуют такие элементы v_k и t_k в D_k, что v_k - переменная, не входящая в t_k, то переход на шаг 4. В противном случае останов: множество S неунифицируемо.

Шаг 4. Полагаем $\sigma_{k+1} = \sigma_k \circ \{t_k/v_k\}$ и $S_{k+1} = W_k\{t_k/v_k\}$ (заметим, что $W_{k+1} = W\sigma_{k+1}$).

Шаг 5. Полагаем $k = k + 1$ и переходим к шагу 2.

ПРИМЕР 1. Найти mgu для $S = \{P(a,x,f(g(y))), P(z,f(z),f(u))\}$.

1. $\sigma_0 = \emptyset$, $S_0 = S$. Так как S_0 - неединичное множество, то σ_0 не является mgu для S.

2. Строим множество рассогласований $D_0 = \{a,z\}$. В D_0 существует переменная $v_0 = z$, которая не встречается в $t_0 = a$.

3. Полагаем $\sigma_1 = \sigma_0 \circ \{t_0/v_0\} = a/z$,

$$S_1 = S_0\{a/z\} = \{P(a,x,f(g(y))),\ P(a,f(a),f(u))\}.$$

4. S_1 - неединичное множество. Строим множество рассогласований $D_1 = \{x, f(a)\}$. В D_1 существует переменная $v_1 = x$, которая не встречается в $t_1 = f(a)$.

5. Полагаем $\sigma_2 = \sigma_1 \circ \{t_1/v_1\} = \{a/z, f(a)/x\}$,

$$S_2 = S_1\{f(a)/x\} = \{P(a,f(a),f(g(y))),\ P(a,f(a),f(u))\}.$$

6. S_2 - неединичное множество. Строим множество рассогласований $D_2 = \{g(y), u\}$. В D_2 существует переменная $v_2 = u$, не встречающаяся в $t_2 = g(y)$.

7. Полагаем $\sigma_3 = \sigma_2 \circ \{t_2/v_2\} = \{a/z, f(a)/x\ g(y)/u\}$,

$$S_3 = \{P(a,f(a),f(g(y))),\ P(a,f(a),f(g(y)))\} = \{P(a,f(a),f(g(y)))\}.$$

8. S_3 - единичное множество, поэтому $\sigma_3 = \{a/z,\ f(a)/x, g(y)/u\}$ есть mgu для S.

Трудоемкость алгоритма Робинсона может быть экспоненциальной в наихудшем случае.

Рассмотрим пример, заимствованный из [11].

Пусть

$$S = \{p(x_1, \ldots, x_n),\ p(f(x_0,x_0), \ldots, f(x_{n-1},x_{n-1}))\}.$$

Тогда

$$D_1 = \{x_1, f(x_0,x_0)\} \text{ и } \sigma_1 = \{f(x_0,x_0)/x_1\},$$

$$S\sigma_1 = \{p(f(x_0,x_0), x_2, \ldots, x_n\},$$
$$p(f(x_0,x_0), f(f(x_0,x_0), f(x_0,x_0)),$$
$$f(x_2,x_2), \ldots, f(x_{n-1},x_{n-1}));$$

$D_2 = \{x_2, f(f(x_0,x_0),f(x_0,x_0))\}$ и

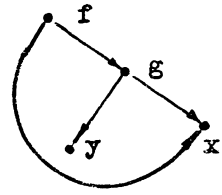
$\sigma_2 = \{f(x_0,x_0)/x_1,$

$f(f(x_0,x_0),f(x_0,x_0))/x_2\}$ и т.д.

Отметим, что второй терм в $S\sigma_n$ имеет (2^k-1) вхождений f в k-й аргумент $(1 \le k \le n)$. Выполнение шага 3 в этом случае требует экспоненциального времени.

Линейный алгоритм унификации, предложенный в [12], основан на представлении термов в виде направленных ациклических графов, а также на использовании приема, известного в литературе под названием "propagating the equivalence" ("размножение эквивалентности").

Направленный ациклический граф (dag) - это граф с вершинами, помеченными функциональными символами и переменными. Из вершины, помеченной n-местным функциональным символом, выходит n дуг. Все общие подвыражения терма представляются одним подграфом, а для каждой переменной существует только одна вершина. Например, терм $t = f(x,g(y,x))$ представляется в виде:



Унифицируемые термы представляются в виде dagов с двумя различными вершинами, соответствующими корневому функциональному символу. Вычисление наиболее общего унификатора для двух термов сводится к вычислению некоторого отношения эквивалентности на вершинах соответствующих dagов.

Отношение эквивалентности для вершин имеет место при выполнении следующих условий:

1) если две функциональные вершины эквивалентны, то соответствующие сыновья попарно эквивалентны;

2) каждый класс эквивалентности является гомогенным, т.е. он не содержит двух вершин с различными функциональными символами;

3) классы эквивалентности могут быть частично упорядочены по отношению частичного порядка на данном графе (условие ациклличности).

Авторы алгоритма доказали, что термы, соответствующие вершинам u и v, унифицируемы тогда и только тогда, когда существует отношение эквивалентности $\equiv$ такое, что $u \equiv v$. В данном случае это также и минимальное, единственное отношение эквивалентности, соответствующее mgu.

Рассмотрим работу алгоритма на примере.

ПРИМЕР 2. Найти mgu для

$$S = \{A(B(v),\ C(u,v)),\ A(B(w),\ C(w,D(x,y)))\}.$$

Графы, соответствующие данным термам, представлены на рис. 1а.

Шаг 1. Создаем неориентированную дугу, связывающую корневые вершины (неориентированные дуги будем изображать пунктирными линиями). Результирующий граф представлен на рис. 1б.

Шаг 2. Анализируя оба графа снизу вверх, находим первую функциональную вершину $D(x,y)$. Проанализируем всех отцов вершины D, запоминая их в порядке просмотра: D-C-A. Вершина A не является сыном никакой другой вершины.

Шаг 3. Анализируем наличие неориентированной дуги из вершины A. Такая дуга существует. Проверяем условие 2 и в случае его выполнения удаляем неориентированную дугу A-A, связываем по условию 1 пары (j-й сын, j-й сын) неориентированными дугами, удаляем все дуги, выходящие из вершин, помеченных символами A (рис.1в). Сохраняемый порядок просмотра: D-C.

Шаг 4. Следующая по порядку просмотра вершина C не имеет отцов. Из нее выходит неориентированная дуга в вершину, также помеченную функциональным символом C. Удаляем дугу C-C, связы-

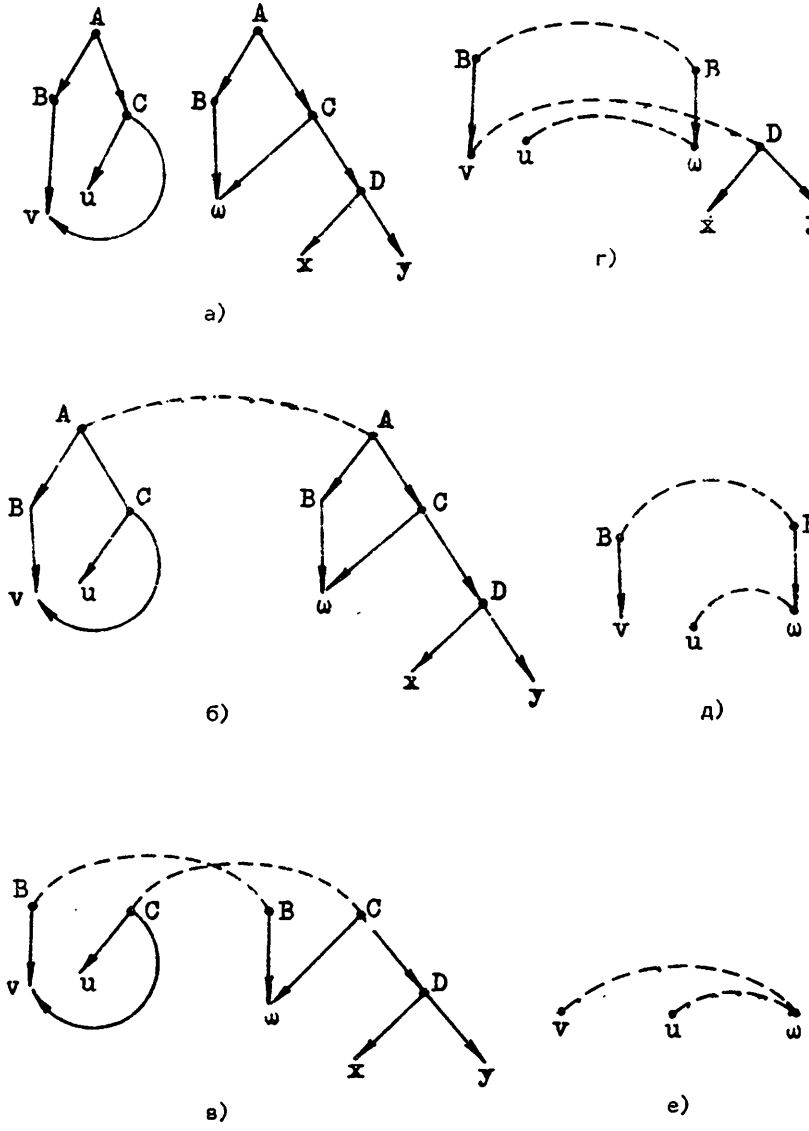


Рис.1. Преобразование графов в алгоритме [12]

ваем соответствующих сыновей и удаляем все дуги, выходящие из вершин, помеченных символом C (рис.1г). Сохраняемый порядок просмотра: D.

Шаг 5. Следующая анализируемая вершина D не имеет отцов. Из нее выходит неориентированная дуга $D-V$. Символы в вершинах дуг не совпадают, и один из символов (V) является переменной. В этом случае выписываем подстановку $V/D(x,y)$, удаляем дугу $D-V$ и все выходящие из D дуги и сохраняем порядок просмотра: V (рис. 1д).

Шаг 6. Следующая анализируемая вершина V имеет отца - вершину B. Запомним порядок просмотра: $V-B$. Вершина B не имеет отцов, но из нее выходит неориентированная дуга в вершину с тем же функциональным символом. Удаляем дугу B-B, связываем соответствующих сыновей и удаляем все дуги, выходящие из вершин B (рис.1е). Сохраняемый порядок просмотра: V.

Шаг 7. Следующая анализируемая вершина V не имеет отцов. Из нее выходит неориентированная дуга $V-\omega$. Так как V и ω - переменные, то записываем подстановку V/ω, удаляем дугу $V-\omega$ и сохраняем порядок просмотра: ω.

Шаг 8. Следующая анализируемая вершина ω не имеет отцов. Из нее выходит неориентированная дуга $\omega-u$. Так как u и ω - переменные, то записываем подстановку ω/u, удаляем дугу $\omega-u$ и сохраняем порядок просмотра: u.

Шаг 9. Из вершины u не выходит ни одной дуги, и она (вершина) не имеет отцов. Просмотр закончен.

Шаг 10. Фиксируем полученный наиболее общий унификатор:

$$\{V/D(x,y),\ V/\omega,\ \omega/u\}.$$

Трудоемкость описанного алгоритма линейна в зависимости от длины термов, однако, необходимо очень аккуратно использовать прием, обозначенный выше как "размножение эквивалентности".

Серьезным источником неэффективности в рассмотренном алгоритме являются используемые структуры данных (направленные ациклические графы), а также выходной формат подстановок.

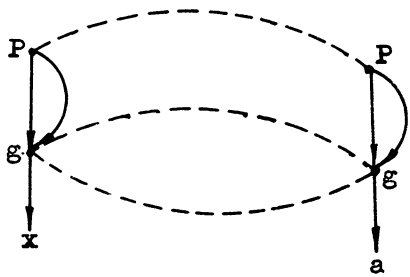


Рис. 2

В некоторых случаях возникает необходимость многократного связывания вершин и, следовательно, многократного их анализа. Так, например,для термов $s = P(g(x),g(x))$ и $t = P(g(a), g(a))$ вершины g-g должны быть связаны дважды (рис.2) как сыновья вершин P по правой и по левой дугам. Такая ситуация может потребовать более чем линейного времени для анализа.

Второй недостаток связан с подстановками типа x/u. Они допустимы лишь в тех случаях, когда u не содержит x, что требует дополнительной проверки на вхождение. Кроме того, при подстановках типа x/u, где $x,u \in V$, важную роль играет выбор последовательности подстановок. Например,при унификации термов $P(x,x,x)$ и $P(g(g(a)),g(g(z)),g(y))$ можно выбрать три варианта подстановок: $\{x := g(g(a))\}$, $\{x := g(g(z))\}$, $\{x := g(y)\}$. Все они в конечном счете приводят к подстановкам вида $\{x := g(g(a)), z := a, y := g(a))\}$, но с разными затратами.

В [13] предложен вариант улучшения алгоритма, позволяющий избежать указанных недостатков.

Метод трансформаций, предложенный в [14] и восходящий к [15], использует трансформацию множества равенств в другое множество, эквивалентное первоначальному и имеющее то же множество унификаторов. К числу таких трансформаций (преобразований) относятся:

1) элиминация тривиальных пар, т.е. преобразования вида:

$$\{ < s, s > \} \cup S \Rightarrow S ,$$

где S - любая (возможно, и пустая) система равенств;

2) редукция термов, т.е. преобразования вида:

$$\{< f(s_1,\ldots,s_n),\ f(t_1,\ldots,t_n) >\} \cup$$

$$\cup S \Rightarrow \{< s_1,t_1 >,\ldots,< s_n,t_n> \} \cup S;$$

3) элиминация переменных, т.е. преобразования вида:

$$\{ <x,t > \} \cup S \Rightarrow \{< x,t > \} \cup S[t/x],$$

где x - переменная, встречающаяся в S, но не встречающаяся в t, $S[t/x]$ - результат подстановки терма t вместо каждого вхождения x в S.

Сложность алгоритма, использующего метод трансформаций, - $O(n + m\log m)$, где m - число различных переменных в термах.

Как отмечается в ряде работ (см., например, [2,3,12]), в типичных приложениях, таких как доказательство теорем, алгоритмы унификации чаще оперируют термами небольшой длины, поэтому асимптотические различия в оценках приведенных алгоритмов не могут быть использованы.

4. Унификация в специальных теориях

Рассмотрим некоторые специальные теории, классифицированные по числу наиболее общих унификаторов [3].

Эквациональная теория называется:

- унитарной, если для каждой пары унифицируемых термов существует в точности один mgu;
- финитарной, если множество mgu непусто и конечно;
- нефинитарной, если множество mgu бесконечно;
- нульарной, если множество mgu пусто.

Согласно приведенной классификации, Ø-теория является унитарной, C-теория является финитарной и т.д.

Стандартный алгоритм Ø-унификации может быть легко адаптирован на случай C-теории. Для унификации, например, двух термов $f(s_1,s_2)$ и $f(t_1,t_2)$, где f коммутативна, необходимо попытаться унифицировать s_1 с t_1 и s_2 с t_2, а также s_1 с t_2 и s_2 с t_1, т.е. проблема C-унификации сводится к Ø-унификациям, и ее трудоемкость экспоненциальна в зависимости от числа вхождений коммутативного функционального символа.

Эквациональная теория A является нефинитарной. Если функция f ассоциативна, например, $f(x,f(y,z)) = f(f(x,y),z)$, то можно не делать различий между переменными и представлять терм списком аргументов $[_f x,y,z] = [x,y,z]$.

Алгоритм, описанный в [3], включает следующие шаги.

Шаг 1. Анализ списков аргументов слева направо и построение множества рассогласований. Если один из списков исчерпался раньше другого, то унификация невозможна.

Шаг 2. Если множество рассогласований не содержит терма t и переменной x, не встречающейся в t, то унификация заканчивается с этой подстановкой.

Шаг 3. В противном случае создаются два унификационных подпроцесса с подстановками $x:=t$ и $x:=f(t,u)$, где u - новая переменная.

ПРИМЕР 3. Унифицировать два терма $f(x,y)$ и $f(a,f(b,c))$. Функция f ассоциативна: $[x,y] = [a,b,c]$.

A. $x:=a$, $[a,y] = [a,b,c]$;

1. $y:=b$, $[a,b] = [a,b,c]$ - отказ;

2. $y:=f(b,v)$, $[a,b,v] = [a,b,c]$;

   1. $v:=c$, $[a,b,c] = [a,b,c]$ - успех;

   2. $v:=f(c,z_1)$, $[a,b,c,z]=[a,b,c]$ - отказ.

5. $x:=f(a,u)$, $[a,u,y] = [a,b,c]$;

1. $u:=b$, $[a,b,y] = [a,b,c]$;

   1. $y:=c$, $[a,b,c] = [a,b,c]$ - успех;

   2. $y:=f(c,z_2)$, $[a,b,c,z_1] = [a,b,c]$ - отказ;

2. $u:=f(b,\omega)$, $[a,b,\omega,y] = [a,b,c]$;

   1. $\omega:=c$, $[a,b,c,y] = [a,b,c]$ - отказ;

   2. $\omega:=f(c,z_3)$, $[a,b,c,z_3,y] = [a,b,c]$ - отказ.

Два наиболее общих унификатора имеют вид:

$$\sigma_1 = \{x:=a,\ y:=f(b,c)\},$$

$$\sigma_2 = \{x:=f(a,b),\ y:=c\}.$$

Алгоритмы унификации для $(A+C)$-теории рассмотрены в работах [9,10,16]. Они сводятся к решению линейных однородных диофантовых уравнений.

5. Алгоритмы идентификации

Напомним, что термы s и t называются идентифицируемыми тогда и только тогда, когда существует подстановка σ такая, что $\sigma_s =_E t$.

Мы рассмотрим два алгоритма, использующих структуры данных и аналогичных направленным ациклическим графам и конструкциям для идентификации цепочек.

Алгоритм, предложенный в [17], использует структуры данных, называемые сжатыми ациклическими графами. Отличие их от направленных ациклических графов состоит в том, что для представления переменных используется единственная вершина, поэтому все подтермы, отличающиеся только именами переменных, представляются одними и теми же подграфами в графе.

При представлении терма сжатым dagом возникает проблема сохранения информации об использовании переменных. Предполагается, что все переменные переименованы и перенумерованы слева направо в порядке их первого вхождения в терм. Для каждой вершины строится карта переменных, позволяющая восстанавливать имена переменных. На рис.3 представлены термы в виде обычного и.сжатого dagов.

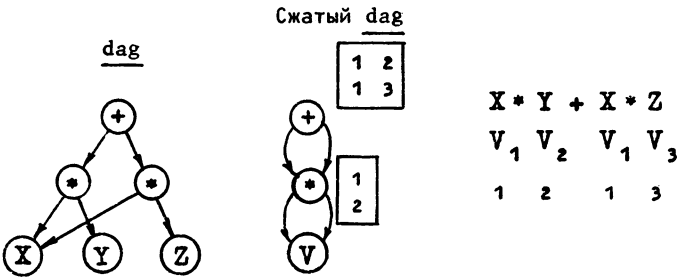


Рис. 3

Алгоритм является алгоритмом унификации одного терма со многими. Множество термов представляется одним сжатым dagом, унифицируемый терм - обычным dagом. Процесс построения сжатого графа по многим термам в рамках алгоритма не рассматривается.

Алгоритм начинает анализ обычного daga снизу вверх, пытаясь идентифицировать концевые вершины с вершиной-переменной в сжатом dage, вершины, помеченные функциональными символами,с соответственно размеченными вершинами в сжатом графе и т.д.

Рассмотрим работу алгоритма на примере.

ПРИМЕР 4. Пусть имеются три выражения $X * Y + X * Z$, $X + Y * X$, $X + Y$ и терм $P * Q + (Q * P) * (P * Q)$, представленные графами на рис.4 (для удобства вершины поименованы буквами).

Идентифицируемые вершины и подстановки представлены в табл. 3.

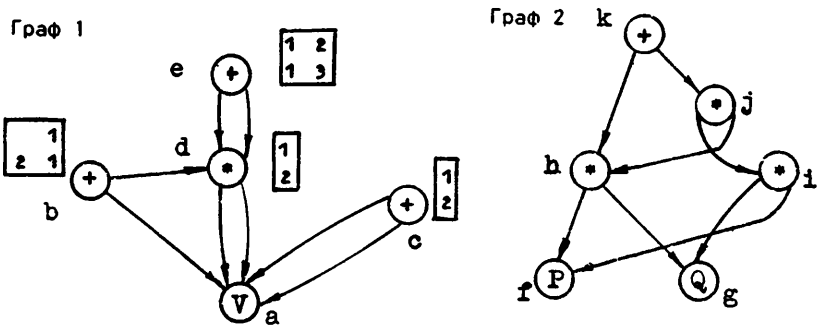


Рис. 4

Таким образом, терм $P*Q+(Q*P)*(P*Q)$ идентифицируется с термами $X+Y*X$ и $X+Y$ с помощью подстановок $\{X:=P*Q,\ Y:=Q*P\}$ и $\{X:=P*Q,\ Y:=(Q*P)*(P*Q)\}$ соответственно.

Трудоемкость описанного алгоритма квадратична в зависимости от числа сравнений в каждой вершине. Число же таких сравнений в каждой вершине может быть довольно большим.

Алгоритм, предложенный в [18], сводится к алгоритмам идентификации цепочек.

Уточним постановку задачи идентификации для рассматриваемого случая: пусть $p_1 \ldots p_k$ - множество термов (образов) с переменными, t - терм, не включающий переменных. Решением задачи идентификации будем считать множество пар (n,i), где n - вершина в t (мы не будем различать сами термы с их представлением в виде графов), а p_i идентифицируется с n. Так, например, терм $a(a(b,c),\ a(a(b,b),b))$ идентифицируется с образом $a(a(b,v),\ v)$ в двух вершинах, помеченных на рис.5 звездочками (здесь v - символ переменной).

Основная идея алгоритма состоит в преобразовании дерева, представляющего терм, во множество цепочек. При построении цепочек учитывается нумерация сыновей, символ же v в конце це-

Таблица 3

Граф 2	Граф 1	Подстановка	Комментарии
f	a	$V_a = f$	
g	a	$V_a = g$	
h	d	$V_{d_1} = f$, $V_{d_2} = g$	Вершина d представляет подтерм вида $V_{d_1} * V_{d_2}$ и может быть унифицирована с вершиной h при подстановках $V_{d_1} = f$ и $V_{d_2} = g$
i	d	$V_{d_1} = g$, $V_{d_2} = f$	$V_d = V_{d_1} * V_{d_2}$ $i = g * f$
j	d	$V_{d_1} = i$, $V_{d_2} = h$	$V_d = V_{d_1} * V_{d_2}$ $j = i * h$
k	b	$V_{b_1} = h$, $V_{b_2} = i$	$V_b = V_{b_1} + V_{b_2} * V_{b_1}$ $k = h + j = h + (i * h)$ УСПЕХ
	e		$V_e = V_{e_1} + V_{e_2} =$ $= V_{d_1} * V_{d_2} + V_{d_1} * V_{d_3}$ $k = h + i * h = f * g +$ $+ i * h$ НЕУСПЕХ
	c	$V_{c_1} = h$, $V_{c_2} = j$	$V_c = V_{c_1} + V_{c_2}$ $k = h + j$ УСПЕХ

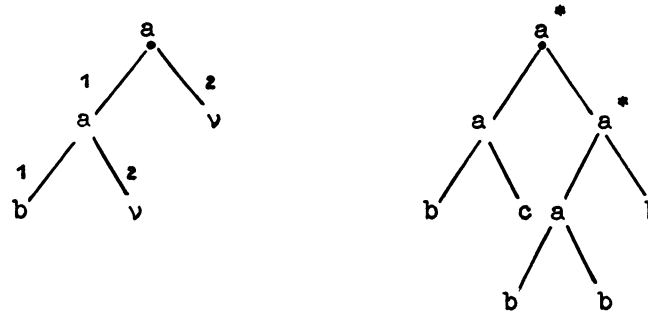


Рис. 5

почки опускают. Так, терм $t = a(a(b,v),v)$ (рис.5) представляется следующим множеством цепочек, ведущих от корня к листьям: $\{a1a1b, a1a2, a2\}$.

Далее используется алгоритм Ахо-Корасика [19] для построения автомата идентификации цепочки. Трудоемкость этого шага пропорциональна размерам дерева, если использовать конструкцию, называемую функцией отказов. Она учитывает возможность самопересечения цепочек.

Функция отказов определяется следующим образом: если j - вершина, в которую мы попадаем, проанализировав цепочку $b_1 \dots b_j$, то функция отказов $f(j)$ - это наибольшее число $s < j$, для которого суффикс проанализированной цепочки $b_1 \dots b_j$ совпадает с ее префиксом, т.е. $b_1 \dots b_s = b_{j-s+1} b_{j-s+2} \dots b_j$.

Если такого $s \geq 1$ нет, то $f(j) = 0$. Функция строится только для вершин, из которых выходит дуга, помеченная символом, а не цифрой.

ПРИМЕР 5. Пусть $t = a(a(b,v), c)$ - терм, представляемый множеством цепочек $\{a1a1b, a1a2, a2c\}$. Цепочки, упакованные в виде дерева, изображены на рис.6а. С помощью функции

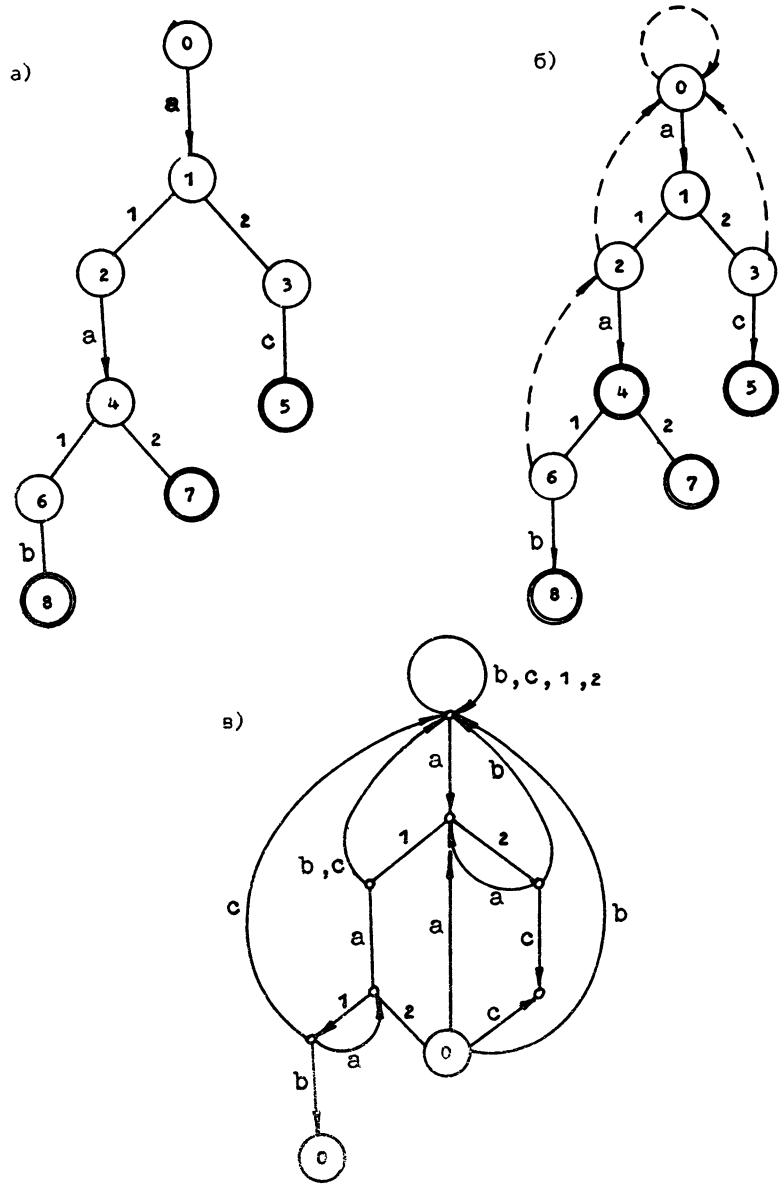


Рис.6. Этапы построения идентифицирующего автомата

отказов это дерево может быть преобразовано к виду 6б. Далее в соответствии с алгоритмом Ахо-Корасика строится детерминированный конечный автомат для идентификации цепочек (рис.6в). Трудоемкость идентификации с использованием построенного автомата линейна.

З а к л ю ч е н и е

В данном обзоре, имеющем алгоритмическую направленность, мы ограничились рассмотрением простейших случаев унификации и идентификации. Различные модификации описанных алгоритмов используются в экспертных системах, дедуктивных базах данных, системах логического программирования.

В некоторых практических приложениях переменные типизированы. Мы не касались таких ситуаций, а также возможностей распространения известных результатов на многосортные алгебры. Некоторые результаты в этом направлении можно найти в [20,21].

Одним из новых направлений в теории унификации является унификация в признаковых структурах [22] и в формальных языках (см., например, [23]), где не делается различий между функциональными символами и константами. Это сильно расширяет области возможных приложений унификации. В частности, в [24] был предложен линейный алгоритм идентификации в языках образов, нашедший применение при распознавании функциональных сайтов в генетических текстах [25].

Л и т е р а т у р а

1. ROBINSON J.A. A Machine Orientated Logic Based on the Resolution Principle //JACM.- 1965.- Vol.12.- P.23-41.

2. SIEKMANN J. Universal Unification //LNCS. - 1984. - N 170. - P. 1-42.

3. SOMMERHALDER R. Unification - an Overview//CWI Tract. 1987. - N 42. - P. 183-204.

4. ROBINSON J.A. Computational Logic: The Unification Computation// Machine Intelligence.- 1971.- Vol.6.

5. PLOTKIN G. Building in Equational Theory //Machine Intelligence.- 1972.- Vol.7.- P.73-90.

6. SLAGLE J.R. Automated Theory-Proving for Theories with Simplifiers, Commutativity and Associativity // JACM. - 1974. - Vol. 21.- P.622-642.

7. ARNBORG S., TIDEN E. Unification Problems with One-Sided Distributivity //LNCS. - 1985.-№ 202.- P.398-406.

8. SHABO P. Theory of First Order Unification. - Thesis.- University of Karlsruhe.- 1982.

9. FAGES F. Associative-Commutative Unification // LNCS. - 1984. - N. 170. -P. 194-208.

10. STICKEL M.E. A Complete Unification Algorithm for Associative-Commutative Function //JACM. - 1981. -Vol. 28.-P.423-434.

11. BIBEL W. Automated Theory Proving.-Vieweg: Braunschweig. 1982.

12. PATERSON M.S., WEGMAN M.N. Linear Unification //J. of Computer and System Science. - 1978. -Vol. 16, N 2. -P.158-167.

13. CHAMPEAUX D. About the Paterson-Wegman Linear Unification Algorithm //J. of Computer and System Science. - 1986. -Vol.32, N 1. -P. 79-90.

14. MARTELLI A.,MONTANARY V. An Efficient Unification Algorithm //ACM Trans.Program.Lang.Syst. - 1982. -Vol. 4.-P.258-282.

15. HERBRAND J. Recherches sur la theorie de la demonstration//Traveaux de la Soc. des Sciences et des hrttres de Varsovie,III.- 1930.- Vol.33.- P.33-160.

16. HUET G. An Algorithm to Generate a Basis of Solutions to Homogeneous Diophantine Equations //Inhrom. Process. Lett. - 1978.- Vol. 7. - P. 144-147.

17. PURDOM P.W., BROWN C.A. Fast Many-to-One Matching Algorithms //LNCS. - N. 202. -P. 407-417.

18. HOFFMANN C.M., O'DONNEL M.J. Pattern Matching in Trees //JACM. - 1982. - Vol. 29. -P. 68-95.

19. AHO A.V., CORASICK M.J. Efficient String Matching: an Aid to Bibliographic Search //CACM. - 1975. - Vol. 18. -P.333-344.

20. SCHMIDT-SCHAUSS B.M. Unification in Many-Sorted Equational Theories // LNCS. - 1986.- № 230.- P.538-552.

21. WALTHER C.A. A Classification of Many-Sorted Unification Problems //LNCS.- 1986.- № 230.- P.525-537.

22. AIT-KACI H., NASR R. Login: A Logic Programming Language with Built-in Inheritance //J.of Logic Programming. - 1986. - Vol.3.- P.185-215.

23. KNIGHT K. Unification: A Multidisciplinary Survey // ACM Computer Survey.-1989.- Vol.21, № 1.- P.93-124.

24. CHUZHANOVA N. Inductive Synthesis of Programms for Symbolic Sequences Processing //LNCS.- 1989.- Vol.397.- P.317-327.

25. GUSEV V., CHUZHANOVA N. The Algorithms of Recognition of the Functional Sites in Genetic Texts //Proc. of the Workshop on Algorithmic Learning Theory, Japan, Tokio.-1990.-P.109-119.

Поступила в ред.-изд.отд.
5 декабря 1990 года