УДК 519.68:510

С. А. Нигиян, Л. О. Хачоян, В. Р. Акопян

Оптимизация систем логического программирования посредством преобразований их программ

(Представлено академиком Н.У. Аракеляном 14/X 2003)

В работе дано определение T-оптимизируемой системы логического программирования, где T - некоторое множество преобразований программ. В качестве иллюстрации общего подхода выбраны интерпретатор системы PROLOG и множество преобразований, которые переставляют и удаляют предложения программ, переставляют атомы в телах правил программ. Для ряда систем логического программирования получены результаты относительно их T-оптимизируемости.

1. *Определения и обозначения.* Зафиксируем три непересекающихся счетных множества Ф, П и X. Ф - множество функциональных символов с приписанной каждому символу местностью, причем для любого $n > 0$ Ф содержит счетное число символов местности n. X - множество предметных переменных. Из элементов множеств Ф и X строятся термы.

1. Каждый 0-местный символ из Ф есть терм.
2. Каждая переменная из X есть терм.
3. Если $t_1, t_2, ..., t_n$ $(n > 0)$ - термы и f - n - местный символ из Ф, то $f(t_1,\ldots,t_n)$ есть терм.
4. Других термов нет.

Множество всех термов, не использующих предметных переменных, обозначим через M. Множество M называется универсумом Эрбрана.

$П=П_1 \cup П_2$, где $П_1$ - множество предикатных символов с приписанной каждому символу местностью, причем для любого $n > 0$ $П_1$ содержит счетное число символов местности n; $П_2$ - некоторое множество интерпретированных предикатных символов (встроенных предикатов), каждый k-местный $(k > 0)$ встроенный предикат представляет собой вычислимое отображение $M^k \to \{true, false\}$.

Атом определяется традиционным образом:

1. Каждый 0-местный символ из $П_1$ есть атом.
2. Если $t_1, \ldots, t_n$ $(n > 0)$ - термы и p - n-местный символ из $П_1$, то $p(t_1,\ldots,t_n)$ есть атом.
3. Других атомов нет.

Атом, использующий предикатный символ из $П_1$, назовем предикатным термом. Атом, использующий встроенный предикат из $П_2$, назовем условием. Множество всех переменных атома A обозначим через Var(A).

Традиционным образом определяется формула логики предикатов первого порядка, использующая логические операции $\neg, \&, \vee, \supset$ и кванторы $\exists, \forall$ (см.[1]).

Опишем рассматриваемые нами интерпретации. Предметным множеством рассматриваемых интерпретаций будет множество M. Функциональные символы интерпретируются следующим образом: каждому 0-местному символу из Φ сопоставляется он сам, каждому n-местному ($n > 0$) символу $f \in \Phi$ сопоставляем отображение $M^n \to M$, которое n-ке $(t_1,\ldots,t_n) \in M^n$, где $t_i \in M$, $i = 1,\ldots,n$, ставит в соответствие терм $f(t_1,\ldots,t_n)$. Каждому 0-местному символу из Π_1 сопоставляется один из элементов множества {true, false}, а каждому n-местному ($n > 0$) символу из Π_1 сопоставляется некоторое отображение $M^n \to$ { true, false}. Обозначим описанное множество интерпретаций через H. Заметим, что интерпретации из H могут отличаться одна от другой лишь отображениями, сопоставляемыми символам множества Π_1.

Пусть F - замкнутая формула и I - интерпретация из H. Значение формулы F на интерпретации I определяется традиционным образом. Формула называется тождественно истинной, если она принимает значение true на любой интерпретации из H. Пусть F и F′ - замкнутые формулы. Мы будем говорить, что формула F′ является логическим следствием формулы F, и обозначать это F|=F′, если формула $F \supset F'$ является тождественно истинной.

Логическая программа P (далее просто программа) есть последовательность предложений $D_1,\ldots,D_n$, $n > 0$. Предложение $D \in \{D_1,\ldots,D_n\}$ является либо фактом A, либо правилом $A{:-}B_1,\ldots,B_m$, которое является импликацией $B_1\&\ldots\&B_m \supset A$, где A - предикатный терм, а каждое из $B_1,\ldots,B_m$ либо предикатный терм, либо условие, $m > 0$. Атом A называется головой предложения D, а последовательность $B_1,\ldots,B_m$ - его телом.

Программе P сопоставляется формула

$$\forall x_1 \ldots \forall x_v (D_1 \& \ldots \& D_n) ,$$

где $x_1,\ldots, x_v$ - переменные, использованные в предложениях $D_1,\ldots, D_n$, $v \geq 0$.

Запрос Q имеет вид $?{-}C_1,\ldots,C_k$, где C_i - либо предикатный терм, либо условие, $i = 1,\ldots,k$, $k > 0$. Запросу Q сопоставляется формула

$$\exists y_1 \ldots \exists y_s (C_1 \& \ldots \& C_k) ,$$

где $y_1, \ldots, y_s$ - переменные, использованные в атомах $C_1,\ldots, C_k$, $s \geq 0$ (см. [2, 3]).

2. *Постановка задачи.* Система логического программирования (кратко LPS) определяется заданием тройки < Prog, Quer, U > , где Prog - некоторое множество программ, Quer - некоторое множество запросов, U - интерпретатор, представляющий собой алгоритм, который для каждой программы $P \in$ Prog и запроса $Q \in$ Quer либо останавливается с положительным ответом (yes), либо с отрицательным ответом (no), либо с неопределенным ответом, либо функционирует бесконечно. Результат применения U к P и Q обозначим U(P,Q). Если U(P,Q) есть yes или no, то будем говорить, что U(P,Q) - определено. Интерпретатор U должен обладать свойством логической корректности:

если U(P,Q) = yes, то P| = Q;

если U(P,Q) = no, то P| ≠ Q.

Пусть P ∈ Prog и Yes(P) = {Q | Q ∈ Quer и P| = Q}.

Будем говорить, что программы P_1,P_2 ∈ Prog эквивалентны (обозначим $P_1 \sim P_2$), если Yes(P_1) =Yes(P_2).

Под преобразованием множества программ Prog мы будем понимать пару (P,P′), где P ~ P′, P,P′ ∈ Prog. Пусть T - некоторое множество преобразований программ Prog, причем для любой программы P ∈ Prog пара (P,P) ∈ T. Будем говорить, что программа P ∈ Prog T-преобразуема в программу P′ ∈ Prog, если существует такая последовательность преобразований (P_1, P_2), …, (P_{n-1},P_n), принадлежащих T, что P_1=P, P_n=P′, n > 1 (см. [4]).

Пусть V - алгоритм, который, получив на вход программу P ∈ Prog, T-преобразует её в программу P′ ∈ Prog (т.е. посредством преобразований множества T преобразует программу P в программу P′). Обозначим через W множество всех таких алгоритмов.

Будем говорить, что система логического программирования (LPS) T-**оптимизируема**, если существует такой алгоритм V_0 ∈ W, что для любого алгоритма V ∈ W, для любой программы P ∈ Prog и для любого запроса Q ∈ Quer имеем:

$$U(V(P),Q) - \text{ определено} \Rightarrow U(V_0(P),Q) - \text{ определено.}$$

3. *Полученные результаты.* Для каждого множества программ Prog определим множество преобразований T, которые переставляют и удаляют предложения программ, переставляют атомы в телах правил программ (см. [4]).

Пусть U - интерпретатор системы PROLOG (см. [3, 5]).

LPS, программы и запросы которой не содержат встроенных предикатов, назовем чистой LPS.

Теорема 1. *Чистая LPS (чистый PROLOG) не является* T-*оптимизируемой.*

Доказательство. Пусть a,b, - 0-местные функциональные символы из Ф, p, q, r - 1-местные предикатные символы из $П_1$, x - переменная из X. Рассмотрим следующую программу P и запросы Q и Q′.

Программа P:

q(a).

r(b).

q(x) : – p(x).

r(x) : – p(x).

p(x) : – q(x).

p(x) : – r(x).

Запрос Q : ? – p(a).

Запрос Q′: ? – p(b).

Легко видеть, что P|=Q, P|=Q′ и для любой собственной подпоследовательности P′′ программы P имеем: P′′ не эквивалентна P. Легко также заметить, что интерпретатор U остановится с положительным ответом (yes) для программы P и запроса Q и будет функционировать бесконечно для программы P и запроса Q′.

Преобразуем программу P, переставив местами её последние два предложения p(x) : – q(x) и p (x) : – r(x). Получим программу P′. Очевидно, что P ~ P′ и интерпретатор U остановится с положительным ответом (yes) для программы P′ и запроса Q′ и будет функционировать бесконечно для программы P′ и запроса Q.

Теорема 1 доказана.

Теорема 2. *Чистая* LPS, *программы и запросы которой не используют переменных,* T-*оптимизируема.*

Доказательство следует из результатов работы [4].

Будем говорить, что программа P содержит повторяющийся предикатный символ q, если число предложений прграммы P, головы которых используют сивол q, больше 1.

Теорема 3. *Чистая* LPS, *программы которой не содержат повторяющихся предикатных символов,* T-*оптимизируема.*

Доказательство следует из результатов работы [4].

LPS, которая использует только 0-местные функциональные символы, 0-местные и 1-местные предикатные символы и встроенные предикаты < , > ,=, называется простой LPS.

Теорема 4. *Чистая простая* LPS, *не является* T-*оптимизируемой.*

Справедливость теоремы 4 следует из доказательства теоремы 1.

Теорема 5. Простая LPS программы которой не используют повторяющихся предикатных символов не является T-оптимизируемой.

Доказательство. Пусть q - 0-местный, p,r,s - 1-местные предикатные символы из Π_1, x - переменная из X. Рассмотрим следующую программу P и запросы Q и Q′.

Программа P:

q : – x > 0.

r(x) : – x > 3, q.

s(x) : – x < 7, q.

p(x) : – r(x), s(x).

Запрос Q : ? – p(2).

Запрос Q′: ? – p(8).

Легко видеть, что P| ≠ Q, P| ≠ Q′ и для любой собственной подпоследовательности P′′ программы P имеем: P′′ не эквивалентна P. Легко также заметить, что интерпретатор U остановится с отрицательным ответом (no) для программы P и запроса Q и остановится с неопределённым ответом для программы P и запроса Q′.

Преобразуем программу P, переставив местами атомы в теле её последнего предложения, т. е. последнее предложение теперь стало иметь p(x) : – s(x), r(x). Получим программу P′. Очевидно, что P ~ P′ и интерпретатор U остановится с отрицательным ответом (no) для программы P′ и запроса Q′ и остановится с неопределённым ответом для программы P′ и запроса Q.

Теорема 5 доказана.

Правило A : – $B_1,\ldots,B_m$ (m > 0) назовем α-правилом, если $Var(B_1)\cup \ldots \cup Var(B_m) \not\subset Var(A)$.

Теорема 6. *Простая* LPS, *программы которой не содержат α-правил и повторяющихся предикатных символов,* T-*оптимизируема.*

Ереванский государственный университет

Литература

1. *Kleene S.C.* Introduction to Metamathematics. D.Van Nostrand Co. 1952.
2. *Lloyd J.W.* Foundadions of Logic Programming. Berlin: Springer-Verlag. 1984.
3. *Нигиян С. А.* - Программирование. 1996. N1. С.30-38. (англ. пер. Nigiyan S. A. - Programming and Computer Software. 1996. V. 22. N. 1. P. 19-25).
4. *Нигиян С. А., Хачоян Л. О.* - Программирование. 1997. N6. С.17-28. (англ. пер. Nigiyan S.A., Khachoyan L.O. - Programming and Computer Software. 1997. V. 23. N. 6. P. 302-309).
5. *Clocksin W. F., Mellish C. S.* Programming in PROLOG. Berlin. Springer-Verlag. 1984. (рус. пер. Клоксин У., Меллиш К. Программирование на языке ПРОЛОГ. М. Мир. 1997).

Ս. Ա. Նիգիյան, Լ. Օ. Խաչոյան, Վ. Ռ. Հակոբյան

Տրամաբանական ծրագրավորման համակարգերի օպտիմիզացիա նրանց ծրագրերի ձևափոխման միջոցով

Աշխատանքում տրված է տրամաբանական ծրագրավորման T-օպտիմիզացվող համակարգի սահմանումը, որտեղ T-ն ծրագրերի ձևափոխությունների ինչ-որ մի բազմություն է: Ընդհանուր մոտեցման ցուցադրման համար ընտրված են $PROLOG$ համակարգի ինտերպրետատորը և այն ձևափոխությունների բազմությունը, որոնք տեղափոխում և հեռացնում են ծրագրերի նախադասությունները, տեղափոխում են ատոմները ծրագրերի կանոնների մարմիններում: Տրամաբանական ծրագրավորման մի շարք համակարգերի համար ստացվել են արդյունքներ նրանց T-օպտիմիզացման վերաբերյալ: