05.13.04
А - 613

ՀՀ ԳԻՏՈՒԹՅՈՒՆՆԵՐԻ ԱԶԳԱՅԻՆ ԱԿԱԴԵՄԻԱ
ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԱՎՏՈՄԱՏԱՑՄԱՆ ՊՐՈԲԼԵՄՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

Համբարձումյան Արամ Մառլենի

# ՏՐԱՄԱԲԱՆԱԿԱՆ ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ԻՆՏԵՐՊՐԵՏԱՏՈՐՆԵՐԻ ՄԱՍԻՆ

Ե.13.04 - հաշվողական մեքենաների, համալիրների, համակարգերի և ցանցերի մաթեմատիկական և ծրագրային ապահովում
մասնագիտությամբ

Ֆիզիկա-մաթեմատիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության

ՍԵՂՄԱԳԻՐ

Երևան –2005

---

ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ И
АВТОМАТИЗАЦИИ НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК РА

Амбарцумян Арам Марленович

# ОБ ИНТЕРПРЕТАТОРАХ СИСТЕМ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата физико-математических наук

по специальности 05.13.04 - математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей

Ереван-2005

© National Library of Armenia OCR by Portmind

Ատենախոսության թեման հաստատվել է Երևանի պետական համալսարանում

Գիտական ղեկավար՝
ֆիզ. մաթ. գիտությունների դոկտոր, պրոֆեսոր Ս.Ա.Նիգիյան

Պաշտոնական ընդդիմախոսներ՝
Ֆիզ. մաթ. գիտությունների դոկտոր, պրոֆեսոր Հ. Բ. Մարանջյան
Ֆիզ. մաթ. գիտությունների թեկնածու Մ. Հ. Խաչատրյան

Առաջատար կազմակերպություն՝
Հայաստանի պետական ճարտարագիտական համալսարան

Պաշտպանությունը կայանալու է 08.07.05 ժամը 16 ին
375014, Երևան, Պ.Սևակի փ. 1 հասցեով, ՀՀԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտում №037 մասնագիտական խորհրդի նիստում:

Ատենախոսությանը կարելի է ծանոթանալ ինստիտուտի գրադարանում:

Սեղմագիրն առաքված է «8» 06 2005թ.

Մասնագիտական խորհրդի գիտական քարտուղար, ֆ.մ.գ.թ. Մ.Ե.Հարությունյան

---

Тема диссертации утверждена в Ереванском государственном университете.

Научный руководитель:
доктор физ.-мат. наук, профессор С.А. Нигиян

Официальные оппоненты:
Доктор физ.-мат. наук, профессор Г. Б. Маранджян
Кандидат физ.-мат. наук М. А. Хачатрян

Ведущая организация:
Государственный инженерный университет Армении

Защита диссертации состоится 08.07.05 в 16 часов на заседании специализированного Совета №037 при Институте проблем информатики и автоматизации НАН РА по адресу: 375014, Ереван, ул. П.Севака, 1.

С диссертацией можно ознакомиться в библиотеке института.
Автореферат разослан «8» 06 2005г.

Ученый секретарь
специализированного совета к.ф.-м.н. М.Е.Арутюнян

Հայաստանի Ազգային գրադարան
2532-2005



## ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

**Актуальность темы.** В работе исследуются возможности расширения области определения и области значений интерпретаторов систем логического программирования. Под системами логического программирования мы понимаем системы хорновского программирования, не использующие встроенные предикаты (системы чистого хорновского программирования). Система логического программирования задается множеством допустимых логических программ *Prog*, множеством допустимых запросов *Quer* и интерпретатором. К программе из множества *Prog* обращаются с запросами из множества *Quer*. Эти обращения обрабатываются интерпретатором системы логического программирования, который должен ответить, является ли запрос логическим следствием программы или нет, и если да, то для каких значений переменных запроса. Интерпретаторы реальных систем логического программирования не всегда справляются с этой задачей — на некоторых запросах они либо зацикливаются, не выдав ни одного определенного ответа, либо оказываются неспособны перечислить все ответы. В этом контексте встает вопрос о расширении возможностей систем логического программирования. Мы исследуем следующие два пути решения этой задачи:

- получение результатов относительно существования интерпретаторов с более широкими областью определения и областью значений;
- расширение области определения и области значений интерпретатора системы путем преобразований программ и запросов. На преобразования программ накладывается т. н. условие Δ-эквивалентности[1]: подмножества запросов, являющихся логическим следствием исходной и преобразованной программ, должны совпадать. Преобразования запросов должны быть такими, чтобы исходный и преобразованный запрос одновременно либо являлись логическим следствием программы, либо нет, причем множества ответов, которые логически следуют из программы, должны совпадать.

**Целью диссертационной работы** является:

- введение понятий тотально разрешающего интерпретатора (т. е. интерпретатора, способного находить все ответы для любой пары "программа-запрос") и тотально логически полного интерпретатора (т. е. интерпретатора, способного находить все ответы в случае, когда запрос является логическим следствием программы);
- исследование языков логического программирования на предмет существования для них тотально разрешающих или тотально логически полных интерпретаторов;

---

[1] *Нигиян С.А., Хачоян Л.О.* О преобразованиях логических программ.//Программирование, №6, 1997, стр. 17-28.

© National Library of Armenia

- введение понятий тотально разрешающего и тотально логически полного интерпретатора относительно систем преобразований программ и запросов;
- исследование интерпретатора простого монадического ПРОЛОГа с одним повторяющимся предикатным символом относительно некоторого (естественного) множества преобразований;
- программная реализация преобразователя логических программ и запросов, расширяющего область определения интерпретатора ПРОЛОГа, основанная на результатах предыдущего пункта.

**Методика исследований** включает в себя методы алгебры, математической логики, теории алгоритмов, а также метод так называемых шаблонов наименьших моделей логических программ[1].

**Научная новизна.**

1. Введены понятия тотально разрешающего и тотально логически полного интерпретаторов и доказано:
   - существование тотально логически полного интерпретатора для языка логического программирования без ограничений;
   - существование тотально разрешающего интерпретатора для языков логического программирования, программы которых имеют конечные шаблоны их наименьших моделей[1].
2. Введены понятия тотально разрешающего и тотально логически полного интерпретаторов относительно заданного множества преобразований и установлена связь с уже существующими понятиями.
3. Исследован интерпретатор простого монадического ПРОЛОГа с одним повторяющимся предикатным символом относительно системы преобразований, переставляющих и удаляющих предложения программ и переставляющих атомы в телах предложений и в запросах. Для этого случая доказаны:
   - невозможность доведения интепретатора до разрешающего;
   - невозможность доведения интерпретатора до тотально логически полного;
   - возможность доведения интерпретатора до логически полного.
4. Реализован преобразователь программ и запросов простого монадического ПРОЛОГа с одним повторяющимся предикатным символом, доводящий интерпретатор до логически полного.

**Практическая ценность.** Результаты, полученные в диссертационной работе носят как теоретический, так и практический характер. Они могут быть использованы для расширения интеллектуальных способностей уже существующих систем логического программирования и при создании новых систем такого рода.

**Апробация работы, публикации.** Основные результаты диссертации докладывались на семинаре кафедры алгоритмических языков ЕГУ, на

[1] *Нигиян С.А., Хачоян Л.О.* К проблеме Δ-эквивалентности логических программ.//ДНАН Армении, том 99, №2, 1999, с.99-103.



семинаре кафедры системного программирования ЕГУ, на общем семинаре факультета информатики и прикладной математики ЕГУ, на семинаре в ИПИА НАН РА, на международной конференции «Conference on Computer Science and Information Technologies (CSIT'99)», Ереван, 1999.

Основные результаты работы отражены в 3-х публикациях.

**Структура и объем работы.** Диссертация состоит из введения, трех глав, заключения и списка цитируемой литературы (76 наименований). Объем работы — 124 страницы.

## СОДЕРЖАНИЕ РАБОТЫ

Во введении описывается предмет исследования, обосновывается актуальность работы и ее новизна, дается краткий обзор содержания.

**В ГЛАВЕ 1** даются определение языка хорновского программирования (в дальнейшем — язык логического программирования) и понятия преобразований его программ и запросов. Язык логического программирования вместе с его интерпретатором образуют систему логического программирования. Приводятся понятия разрешающего и логически полного интерпретатора, описывающие способности интерпретатора находить, соответственно, непустое подмножество ответов для любой пары "программа-запрос" и непустое подмножество ответов для случая, когда запрос является логическим следствием программы[1]. Эти понятия расширяются понятиями тотально разрешающего и тотально логически полного интерпретатора, описывающими способности интерпретатора находить, соответственно, все ответы для любой пары "программа-запрос" и все ответы для случая, когда запрос является логическим следствием программы. Устанавливается существование тотально логически полного интерпретатора для языка логического программирования без ограничений и существование тотально разрешающего интерпретатора для языков, программы которых имеют конечные шаблоны наименьших моделей. Приводятся понятия преобразований программ и запросов и понятия разрешаемости и логической полноты интерпретатора с точки зрения этих преобразований. Последние расширяются понятиями тотальной разрешаемости и тотальной логической полноты интерпретатора с точки зрения преобразований программ и запросов. Глава 1 состоит из трех разделов.

**В разделе 1.1** вводятся используемые определения и результаты.

Зафиксируем три непересекающихся множества $X, F, \Pi$.

$X$ – множество предметных переменных;

$F$ — множество функциональных символов с приписанной каждому символу местностью, причем для любого $n \geq 0$ $F$ содержит счетное число символов местности $n$;

[1] *Нигиян С.А., Хачоян Л.О.* О преобразованиях логических программ.//Программирование, №6, 1997, стр. 17-28.

© National Library of Armenia

$\Pi$ — множество предикатных символов с приписанной каждому символу местностью, причем для любого $n \geq 0$ $\Pi$ содержит счетное число символов местности $n$.

Атомы и термы определяются традиционным образом. Множество всех термов без переменых обозначим через $M$. Множество формул логики предикатов первого порядка, построенных с использованием элементов множеств $X, F, P$, логических операций $\neg, \vee, \&, \supset$ и кванторов $\forall$ и $\exists$, обозначим через $\Phi$.

Опишем рассматриваемые нами интерпретации. Предметным множеством рассматриваемых интерпретаций будет множество $M$. Функциональные символы интерпретируются следующим образом: каждому 0-местному символу из $F$ сопоставляется он сам, каждому $n$-местному ($n > 0$) символу $f \in F$ сопоставляется отображение $M^n \to M$, которое $n$-ке $<t_1, ..., t_n>$, где $t_i \in M$, $i = 1, ..., n$, ставит в соответствие терм $f(t_1, ..., t_n)$. Каждому 0–местному символу из $P$ сопоставляется один из элементов множества $\{true, false\}$, а каждому $n$-местному ($n > 0$) символу из $\Pi$ сопоставляется некоторое отображение $M^n \to \{true, false\}$.

Пусть формулы $A, B \in \Phi$. Будем говорить, что формула $B$ логически следует из формулы $A$ и обозначать это $A|=B$, если формула $A \supset B$ принимает значение *true* на любой интерпретации с предметным множеством $M$.

Логическая программа $P$ есть последовательность предложений $S_1, ..., S_n$, $n > 0$. Каждое предложение $S_i$ ($i = 1, ..., n$) имеет вид $A_i :- B_{i1}, ..., B_{im_i}$, $m_i \geq 0$, которому соответствует хорновский дизъюнкт $A_i \vee \neg B_{i1} \vee ... \vee \neg B_{im_i}$. Атом $A_i$ называется головой, а последовательность $B_{i1}, ..., B_{im_i}$ — телом предложения. В случае $m_i = 0$ предложение называется фактом, а при $m_i > 0$ — правилом. Программе $P$ ставится в соответствие формула

$$\forall x_1 ... \forall x_r (S_1 \& ... \& S_n),$$

где $x_1, ..., x_r$ — все переменные, использованные в $S_1, ..., S_n$. Множество всех логических программ обозначим через $\mathbb{P}$.

Запрос $Q$ имеет вид $?- C_1, ..., C_k$, где $C_1, ..., C_k$ — атомы, $k > 0$. Запросу $Q$ соответствует формула

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

где $y_1, ..., y_s$ — все переменные, использованные в $C_1, ..., C_k$. Множество $\{y_1, ..., y_s\}$ будем обозначать через $Var(Q)$. Множество всех запросов обозначим через $\mathbb{Q}$.

Введем отношение $\Delta$-эквивалентности на множестве логических программ. Пусть $P_1, P_2 \in \mathbb{P}$. Будем говорить, что $P_1$ и $P_2$ $\Delta$-эквивалентны (обозначим $P_1 \overset{\Delta}{\sim} P_2$), если $P_1 |= Q \Leftrightarrow P_2 |= Q$ для любого запроса $Q \in \mathbb{Q}$.

Язык логического программирования определяется как пара $<Prog, Quer>$, где $Prog \subseteq \mathbb{P}$, $Quer \subseteq \mathbb{Q}$ — множества программ и запросов данного



языка. Под языком логического программирования без ограничений будем понимать язык $<\mathbb{P}, \mathbb{Q}>$.

Конечное множество $\sigma = \{t_1 / y_1, ..., t_n / y_n\}$, $n \geq 0$ назовем подстановкой, если оно удовлетворяет следующим условиям:

1. $y_1, ..., y_n$ — переменные, а $t_1, ..., t_n$ — термы;
2. $i \neq j \Rightarrow y_i \neq y_j$, $1 \leq i, j \leq n$;
3. $t_i \neq y_i$, $1 \leq i \leq n$.

Под выражением будем понимать либо терм, либо атом, либо запрос. Применение подстановки $\{t_1 / y_1, ..., t_n / y_n\}$ к выражению $E$ заключается в одновременной замене всех вхождений переменных $y_1, ..., y_n$ в $E$ на $t_1, ..., t_n$ соответственно. Результат этого применения обозначим $E\{t_1 / y_1, ..., t_n / y_n\}$.

Пусть $P \in Prog$, $Q \in Quer$. Обозначим через $Log(P, Q)$ множество ответов на непустой запрос $Q$ к программе $P$, соответствующее логической семантике. Оно определяется следующим образом:

Если $P|\neq Q$, то $Log(P, Q) = \{no\}$;

Если $P|=Q$ и $Q$ не содержит переменных, то $Log(P, Q) = \{yes\}$;

Если $Var(Q) = \{y_1, ..., y_s\}$, $s > 0$, то $Log(P, Q) = \{<t_1, ..., t_s> \mid t_1, ..., t_s \in M$, и $P|=Q\{y_1/t_1, ..., y_s/t_s\}\}$.

Система логического программирования определяется как тройка $<Prog, Quer, U>$, где первые две компоненты определяют язык логического программирования, а $U$ — интерпретатор, результат применения которого к программе $P \in Prog$ и запросу $Q \in Quer$ есть множество $U(P, Q)$, удовлетворяющее следующему условию:

Если $Log(P, Q) \subset \{yes, no\}$, то $U(P, Q) \subseteq Log(P, Q)$;

в противном случае $U(P, Q) \subseteq \{<t_1, ..., t_s> \mid t_1, ..., t_s$ — термы, и $<t_1\theta, ..., t_s\theta> \in Log(P, Q)$ для любой подстановки $\theta$ такой, что $t_i\theta \in M$, $i = 1, ..., n\}$.

Определим следующее множество $Ans(U, P, Q)$:

Если $U(P, Q) \subset \{yes, no\}$, то $Ans(U, P, Q) = U(P, Q)$, иначе

$Ans(U, P, Q) = \{<\tau_1, ..., \tau_s> \mid \tau_1, ..., \tau_s \in M$ и существуют такие кортеж $<t_1, ..., t_s> \in U(P, Q)$ и подстановка $\theta$, что $t_i\theta = \tau_i$, $1 \leq i \leq s$, $s > 0\}$.

Интерпретатор $U$ системы логического программирования назовем разрешающим (соответственно, логически полным), если для любых $P \in Prog$, $Q \in Quer$ $U(P, Q) \neq \varnothing$ (соответственно, $P|=Q \Rightarrow U(P, Q) \neq \varnothing$).

Известно, что не существует разрешающего интерпретатора для языка логического программирования без ограничений[1].

**В разделе 1.2** даются понятия тотально разрешающего и тотально логически полного интерпретаторов и доказываются теоремы 1.2.1 и 1.2.2.

Интерпретатор $U$ системы логического программирования назовем тотально разрешающим (соответственно, тотально логически полным), если

[1] *Нигиян С.А.* Интерпретатор ПРОЛОГа с точки зрения логической семантики.// Программирование, №2, 1994, стр. 64-73.

© National Library of Armenia

для любых $P \in Prog$, $Q \in Quer$ $Ans(U, P, Q) = Log(P, Q)$ (соответственно, $P|=Q \Rightarrow Ans(U, P, Q) = Log(P, Q)$).

**Теорема 1.2.1.** Существует тотально логически полный интерпретатор для языка логического программирования без ограничений.

Заметим, что из несуществования разрешающего интерпретатора для языка логического программирования без ограничений следует несуществование для него тотально разрешающего интерпретатора.

**Теорема 1.2.2.** Существует тотально разрешающий интерпретатор для языка логического программирования, программы которого имеют конечные шаблоны их наименьших моделей.

**Следствие теоремы 1.2.2.** Существует тотально разрешающий интерпретатор для языка логического программирования без функциональных символов местности > 1.

**В разделе 1.3** вводятся понятия преобразований программ и запросов и устанавливаются связи между разрешаемостью, логической полнотой, тотальной разрешаемостью и тотальной логической полнотой как без систем преобразований программ и запросов, так и относительно этих систем.

Пусть $\Phi' \subset \Phi$ и $\Phi' \neq \varnothing$. Под преобразованием формул множеса $\Phi'$ мы будем понимать пару $(A, A')$, где $A, A' \in \Phi'$. Пусть $T$ – некоторое непустое множество преобразований формул множества $\Phi'$. Будем говорить, что формула $A \in \Phi'$ $T$-преобразуема в формулу $B \in \Phi'$ (обозначим $A \xrightarrow{T} B$), если существует такая последовательность преобразований $(A_1, A_2), \ldots, (A_{n-1}, A_n)$, принадлежащих $T$, что $A_1 = A$, $A_n = B$, $n > 1$. В этом случае формулу $B$ назовем $T$-образом формулы $A$. Под преобразованием логических программ и запросов будем понимать преобразования соответствующих им логических формул.

Систему $T$ преобразований логических программ назовем $\Delta$-эквивалентной, если $(P_1, P_2) \in T \Rightarrow P_1 \overset{\Delta}{\sim} P_2$. В дальнейшем под системами преобразований программ будем понимать только системы $\Delta$-эквивалентных преобразований.

Пусть $T$ и $T_q$ — некоторые системы преобразований программ $Prog$ и запросов $Quer$ соответственно, причем $P \in Prog \Rightarrow (P, P) \in T$ и $Q \in Quer \Rightarrow (Q, Q) \in T_q$.

Интерпретатор $U$ назовем $T$-разрешающим (соответственно, $T$-логически полным), если существует алгоритм, переводящий посредством преобразований системы $T$ любую программу $P \in Prog$ в программу $P' \in Prog$ (т. е. $P \xrightarrow{T} P'$) так, что для любого запроса $Q \in Quer$ имеем: $U(P', Q) \neq \varnothing$ (соответственно, $P|=Q \Rightarrow U(P', Q) \neq \varnothing$).

Интерпретатор $U$ назовем тотально $T$-разрешающим (соответственно, тотально $T$-логически полным), если существует алгоритм, переводящий посредством преобразований системы $T$ любую программу $P \in Prog$ в программу $P' \in Prog$ (т. е. $P \xrightarrow{T} P'$) так, что для любого запроса $Q \in Quer$ имеем: $Ans(U, P', Q) = Log(P, Q)$ (соответственно, и $P|=Q \Rightarrow Ans(U, P', Q) = Log(P, Q)$).



Интерпретатор $U$ назовем $T,T_q$-разрешающим (соответственно, $T,T_q$-логически полным), если существуют алгоритм, переводящий посредством преобразований системы $T$ любую программу $P \in Prog$ в программу $P' \in Prog$ и алгоритм, переводящий посредством преобразований системы $T_q$ любой запрос $Q \in Quer$ в запрос $Q' \in Quer$ так, что $Log(P, Q) = Log(P', Q')$ и $U(P', Q') \neq \varnothing$ (соответственно, $P|=Q \Rightarrow U(P', Q) \neq \varnothing$).

Интерпретатор $U$ назовем тотально $T,T_q$-разрешающим (соответственно, тотально $T,T_q$-логически полным), если существуют алгоритм, переводящий посредством пре-образований системы $T$ любую программу $P \in Prog$ в программу $P' \in Prog$ и алгоритм, переводящий посредством преобразований системы $T_q$ любой запрос $Q \in Quer$ в запрос $Q' \in Quer$ так, что $Log(P, Q) = Log(P', Q')$ и $Ans(U, P', Q') = Log(P, Q)$ (соответственно, $P|=Q \Rightarrow Ans(U, P', Q') = Log(P, Q)$).

Легко видеть, что имеет место следующее (для краткости в названиях той или иной логической полноты слово "логическая" может опускаться):

$U$— тот. разреш. $\Rightarrow$ $U$— тот. $T$-разреш. $\Rightarrow$ $U$— тот. $T,T_q$-разреш.

$U$— тот. полный $\Rightarrow$ $U$— тот. $T$-полный $\Rightarrow$ $U$— тот. $T,T_q$-полный

$U$— разреш. $\Rightarrow$ $U$— $T$-разреш. $\Rightarrow$ $U$— $T,T_q$-разреш.

$U$— лог. полный $\Rightarrow$ $U$— $T$-лог. полный $\Rightarrow$ $U$— $T,T_q$-лог. полный

$U$ — тотально разрешающий
$\Downarrow$ $\Downarrow$
$U$ — разрешающий $U$ — тотально лог. полный
$\Downarrow$ $\Downarrow$
$U$ — логически полный

$U$ — тотально $T$-разрешающий
$\Downarrow$ $\Downarrow$
$U$ — $T$-разрешающий $U$ — тотально $T$-лог. полный
$\Downarrow$ $\Downarrow$
$U$ — $T$-логически полный

$U$ — тотально $T,T_q$-разрешающий
$\Downarrow$ $\Downarrow$
$U$ — $T,T_q$-разрешающий $U$ — тот. $T,T_q$-лог. полный
$\Downarrow$ $\Downarrow$
$U$ — $T,T_q$-логически полный

© National Library of Armenia

**В ГЛАВЕ 2** исследуется интерпретатор чистого ПРОЛОГа[1] относительно множества преобразований, переставляющих и удаляющих предложения программ и переставляющих атомы в телах правил и в запросах. Для некоторого подмножества чистого ПРОЛОГа устанавливаются результаты, характеризующие интерпретатор относительно выбранного множества преобразований. Глава 2 состоит из трех разделов.

**В разделе 2.1** вводится определение чистого ПРОЛОГа и описываются рассматриваемые нами преобразования.

Система логического программирования "чистый ПРОЛОГ" (далее — ПРОЛОГ) — это тройка $<\mathbb{P}, \mathbb{Q}, U>$, в которой интерпретатор $U$ использует SLD-резолюцию[2] и стратегию поиска вывода пустого запроса, основанную на порядке предложений программы. Функционирование интерпретатора заключается в построении то удлиняющейся, то укорачивающейся последовательности запросов. Если исходный запрос не содержит переменных и в последовательности появляется пустой запрос, то интерпретатор останавливается с положительным ответом (*yes*). Если исходный запрос содержит переменные и в процессе работы интерпретатора появляются пустые запросы, то для каждого пустого запроса интерпретатор выдает значения переменных, для которых исходный запрос следует из программы. Если дальнейшее конструирование цепочки запросов невозможно и пустые запросы получены не были, то независимо от наличия переменных в исходном запросе интерпретатор останавливается с отрицательным ответом (*no*). В остальных случаях интерпретатор функционирует бесконечно без выдачи ответов.

Поскольку функционирование интерпретатора ПРОЛОГа основано на порядке предложений программы и порядке атомов в телах правил и запросах, то наиболее естественными системами преобразований для него представляются следующие системы[3]:

$T_1 = \{(P, P') \mid P, P' \in Prog, P'$ получена из $P$ путем перестановки предложений $P\}$;

$T_2 = \{(P, P') \mid P, P' \in Prog, P'$ получена из $P$ путем перестановки атомов в телах правил $P\}$;

$T_3 = \{(P, P') \mid P, P' \in Prog, P'$ — получена из $P$ путем удаления предложений $P, P \overset{\Delta}{\sim} P'\}$;

$T = T_1 \cup T_2 \cup T_3$.

$T_q = \{(Q, Q') \mid Q, Q' \in Quer, Q'$ получена из $Q$ путем перестановки атомов $Q\}$.

В дальнейшем под $T$ и $T_q$ будем понимать именно эти системы преобразований.

---

[1] *Lloyd J.W.* Foundations of logic programming.//Springer–Verlag, 1984, 120p.

[2] *Apt K.R., van Emden M.H.* Contribution to the theory of logic programming. Journal of the ACM, v.29, №3, 1982, pp.841–862.

[3] *Нигиян С.А., Хачоян Л.О.* О преобразованиях логических программ.//Программирование, №6, 1997, стр. 17-28.



ПРОЛОГ, использующий предикатные символы местности $\leq 1$ и не использующий функциональные символы местности $> 0$, назовем простым монадическим. ПРОЛОГ, запросы и тела предложений которого содержат не более одного атома, назовем урезанным. Предикатный символ $p$, используемый в программе $P$, назовем повторяющимся, если число предложений $P$, головы которых используют $p$, больше 1. Подмножество ПРОЛОГа, в программах которого используется не более одного повторяющегося предикатного символа, назовем ПРОЛОГом с одним повторяющимся предикатным символом. Известно, что интерпретатор урезанного простого монадического ПРОЛОГа с одним повторяющимся предикатным символом является $T$-разрешающим[1].

**В разделе 2.2** рассматривается простой монадический ПРОЛОГ с одним повторяющимся предиактным символом. Доказывается, что его интерпретатор не является $T,T_q$-разрешающим, а интерпретатор урезанного простого монадического ПРОЛОГа не является тотально $T$-логически полным.

**Теорема 2.2.1.** Интерпретатор простого монадического ПРОЛОГа с одним повторяющимся предикатным символом не является $T,T_q$-разрешающим.

Для доказательства приводится пример программы и запроса из данного подмножества ПРОЛОГа и показывается, что для любого $T$-образа исходной программы и для любого $T_q$-образа исходного запроса интерпретатор будет функционировать бесконечно.

**Теорема 2.2.2.** Интерпретатор урезанного простого монадического ПРОЛОГа с одним повторяющимся предикатным символом не является тотально $T$-логически полным.

Доказательство основано на примере программы и запроса, являющегося логическим следствием программы, таких, что ни для какого $T$-образа исходной программы интерпретатор не способен перечислить все отвеы на запрос.

**Следствие теоремы 2.2.2.** Интерпретатор урезанного простого монадического ПРОЛОГа с одним повторяющимся предикатным символом не является тотально $T$-разрешающим.

**Теорема 2.2.3.** Интерпретатор простого монадического ПРОЛОГа с одним повторяющимся предикатным символом не является тотально $T,T_q$-логически полным.

Доказательство следует из теоремы 2.2.2.

**Следствие теоремы 2.2.3.** Интерпретатор простого монадического ПРОЛОГа с одним повторяющимся предикатным символом не является тотально $T,T_q$-разрешающим.

**Раздел 2.3** посвящен доказательству $T,T_q$-логической полноты интерпретатора простого монадического ПРОЛОГа с одним повторяющимся предикатным символом. Пусть *Prog* и *Quer* — множества программ и запросов простого монадического ПРОЛОГа с одним повторяющимся предикатным символом соответственно.

© National Library of Armenia

**Теорема 2.3.1.** Для любой программы $P \in Prog$ существует такой $T$-образ $P'$, что для любого запроса $Q \in Quer$ существует такой $T_q$-образ $Q'$, что $P' \models Q' \Rightarrow U(P', Q') \neq \varnothing$.

Для доказательства вводится классификация предложений программ, на основе которой осуществляется перестановка и удаление предложений, и критерий, на основе которого переставляются атомы в телах предложений и запросах. Показывается, что после таких преобразований интерпретатор остановится хотя бы с одним ответом в случае, когда запрос логически следует из программы.

Теорема 2.3.1 пока не утверждает, что интерпретатор является $T,T_q$-логически полным, поскольку в ней ничего не сказано об алгоритмах, реализующих последовательность преобразований программы и запроса.

**Теорема 2.3.2.** Интерпретатор простого монадического ПРОЛОГа с одним повторяющимся предикатным символом является $T,T_q$-логически полным.

Доказательство теоремы 2.3.2 основывается на теореме 2.3.1 и заключается в построении алгоритмов, резлизующих преобразования программы и запроса.

**В ГЛАВЕ 3** описывается программная реализация преобразователя, основанного на алгоритмах, приведенных в доказательстве теоремы 2.3.2. Описание дано с двух позиций — интерфейса пользователя (раздел 3.1) и описания классов и деталей реализации (раздел 3.2). В рамках автореферата остановимся только на описании графического интерфейса пользователя.

Преобразователь реализован на языке C++ с использованием графического интерфейса Microsoft Windows и является многопоточным (двухпоточным) приложением: выполнение вычислений (ввод данных и преобразования) происходит в отдельном от пользовательского интерфейса потоке, что позволяет сохранять контроль над приложением.

Графический интерфейс предоставляет пользователю возможность как ручного набора текста исходных программы и запроса, так и их загрузки из файлов. Главное окно приложения разделено на четыре части, предоставленные под тексты исходных и конечных программ и запросов. По команде из меню или с панели инструментов происходит запуск второго потока, который выполняет синтаксический анализ и импорт программы и/или запроса, находящихся в двух левых квадрантах главного окна, с последующим их преобразованием. Результаты преобразований выводятся в двух правых квадрантах главного окна. На время выполнения вычислительного потока на экран выводится окно прогресса, показывающее текущее состояние операции и позволяющего пользователю прервать вычисления.

## ОСНОВНЫЕ РЕЗУЛЬТАТЫ

1. Исследован вопрос расширения интеллектуальных способностей систем логического программирования путем использования интерпретатора с более широкими областью определения и областью значений и



установлены результаты относительно существования таких интерпретаторов:
   - введены понятия тотально разрешающего и тотально логически полного интерпретатора, характеризующие его, соответственно, с точки зрения получения всех ответов для любой пары "программа-запрос" и с точки зрения получения всех ответов в случае, если запрос является логическим следствием программы;
   - установлено существование тотально логически полного интерпретатора для языка логического программирования без ограничений;
   - установлено существование тотально разрешающего интерпетатора для языка, программы которого имеют конечные шаблоны их наименьших моделей.
2. Введены понятия тотально разрешающего и тотально логически полного интерпретатора относительно заданной системы преобразований и установлена их связь с существующими понятиями.
3. Исследованы вопросы разрешаемости, тотальной разрешаемости, логической полноты и тотальной логической полноты интерпретатора простого монадического ПРОЛОГа с одним повторяющимся предикатным символом относительно системы преобразований, переставляющих и удаляющих предложения программ и переставляющих атомы в телах предложений и в запросах. Для этого интерпретатора установлены следующие результаты:
   - невозможность доведения интепретатора до разрешающего;
   - невозможность доведения интерпретатора до тотально логически полного;
   - возможность доведения интерпретатора до логически полного.
4. Реализован преобразователь программ и запросов простого монадического ПРОЛОГа с одним повторяющимся предикатным символом, основанный на результатах предыдущего пункта и доводящий интерпретатор до логически полного.

В заключение выражаю глубокую признательность моему научному руководителю проф. С. А. Нигияну за постановку задач и постоянное внимание к работе.

## ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. *Hambardzumyan A.M.* The Completeness and Solvability Problems for Simple Monadic PROLOG Interpreter.//Proceedings of the Conference on Computer Science and Information Technologies, Yerevan, 1999, pp.36-38.

2. *Амбарцумян А.М.* Об интерпретаторе простого монадического ПРОЛОГа.//Ученые записки ЕГУ, 3, 2001, с.40-48.

3. *Nigiyan S.A., Hambardzumyan A.M.* On Interpreters of Logic Programming Systems.//Transactions of the Institute for Informatics and Automation Problems of NAS RA and YSU: Mathematical Problems of Computer Science, vol.24, pp.34-41, Yerevan, 2005.

© National Library of Armenia

## ԱՄՓՈՓԱԳԻՐ

*Արամ Մառլենի Համբարձումյան*
**«Տրամաբանական ծրագրավորման համակարգերի ինտերպրետատորների մասին»**

Ատենախոսության հիմնական արդյունքներն են.

1. Հետազոտվել է տրամաբանական ծրագրավորման համակարգերի ինտելեկտուալ ունակությունների՝ ավելի լայն որոշման և առժեքների տիրույթներով ինտերպրետատորների օգնությամբ ընդլայնման խնդիրը և ստացվել են արդյունքներ այդպիսի ինտերպրետատորների համար.
   - Սահմանվել են տոտալ տրամաբանորեն լրիվ և տոտալ լուծելի ինտերպրետատորների գաղափարները, որոնք բնութագրում են ինտերպրետատորը, համապատասխանաբար, ցանկացած «ծրագիր-հարցում» զույգի համար բոլոր պատասխանները ստանալու ունակության տեսանկյունից և բոլոր պատասխանները ստանալու ունակության տեսանկյունից այն դեպքում, երբ հարցումը հանդիսանում է ծրագրի տրամաբանական հետևանք:
   - Ցույց է տրվել տոտալ տրամաբանորեն լրիվ ինտերպրետատորի գոյությունը առանց սահմանափակումների տրամաբանական ծրագրավորման լեզվի համար:
   - Ցույց է տրվել տոտալ լուծելի ինտերպրետատորի գոյությունը այն լեզուների դասերի համար, որոնց ծրագրերի փոքրագույն մոդելների շաբլոնները վերջավոր են:
2. Սահմանվել են տվյալ ձևափոխումների համակարգի նկատմամբ տոտալ տրամաբանորեն լրիվ և տոտալ լուծելի ինտերպրետատորների գաղափարները և բացահայտվել է նրանց կապը գոյություն ունեցող հասկացությունների հետ:
3. Ծրագրերի նախադասությունները վերադասավորող ու հեռացնող և կանոնների մարմինների ու հարցումների ատոմները վերադասավորող ձևափոխումների համակարգի տեսանկյունից հետազոտվել են մեկ կրկնվող պրեդիկատային նշանով պարզ մոնադիկ ՊՐՈԼՈԳ-ի ինտերպրետատորի տրամաբանական լրիվության, տոտալ տրամաբանական լրիվության և տոտալ լուծելիության հարցերը: Ստացվել են հետևյալ արդյունքները.
   - Հնարավոր չէ նշված համակարգի ինտերպրետատորը հասցնել լուծելի ինտերպրետատորի:
   - Հնարավոր չէ նշված համակարգի ինտերպրետատորը հասցնել տոտալ տրամաբանորեն լրիվ ինտերպրետատորի (որպես հետևանք՝ նաև տոտալ լուծելի ինտերպրետատորի):
   - Հնարավոր է նշված համակարգի ինտերպրետատորը հասցնել տրամաբանորեն լրիվ ինտերպրետատորի:
4. Հիմնվելով նախորդ կետի արդյունքների վրա՝ իրականացվել է մեկ կրկնվող պրեդիկատային նշանով պարզ մոնադիկ ՊՐՈԼՈԳ-ի ծրագրերի և հարցումների ձևափոխիչը, որը հասցնում ինտերպրետատորը տրամաբանորեն լրիվ ինտերպրետատորի:

*Hambardzumyan A. M.*
«On interpreters of logic programming systems»

1 5 .05. 2014

© National Library of Armenia

© National Library of Armenia