ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Սարգսյան Լուսինե Արամայիսի

SLDNF-ՌԵԶՈԼՅՈՒՑԻԱՅԻ ՄԱՍԻՆ ԺԽՏՈՒՄՈՎ ՏՐԱՄԱԲԱՆԱԿԱՆ ԾՐԱԳՐԱՎՈՐՄԱՆ ՄԵՋ

Ա.01.09 «Մաթեմատիկական կիբեռնետիկա և մաթեմատիկական տրամաբանություն» մասնագիտությամբ ֆիզիկամաթեմատիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության

Ս Ե Ղ Մ Ա Գ Ի Ր

Երևան-2012

ЕРЕВАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Саркисян Лусине Арамаисовна

ОБ *SLDNF*-РЕЗОЛЮЦИИ В ЛОГИЧЕСКОМ ПРОГРАММИРОВАНИИ С ОТРИЦАНИЕМ

А В Т О Р Е Ф Е Р А Т

диссертации на соискание ученой степени кандидата физико-математических наук по специальности 01.01.09 “Математическая кибернетика и математическая логика”

Ереван-2012

Ատենախոսության թեման հաստատվել է Երևանի պետական համալսարանում

Գիտական ղեկավար՝	ֆիզ.մաթ. գիտ. դոկտոր	Ս.Ա. Նիգիյան
Պաշտոնական ընդդիմախոսներ՝	ֆիզ.մաթ. գիտ. դոկտոր	Յու.Մ. Մովսիսյան
	ֆիզ.մաթ. գիտ. թեկնածու	Լ.Օ. Խաչոյան
Առաջատար կազմակերպություն՝	ՀՀ ԳԱԱ Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ	

Պաշտպանությունը կայանալու է 2012 թ. մայիսի 23-ին ժամը 16^{00}-ին ԵՊՀ-ում գործող ԲՈՀ-ի 044 «Մաթեմատիկական կիբեռնետիկա և մաթեմատիկական տրամաբանություն» մասնագիտական խորհրդի նիստում, հետևյալ հասցեով՝ 0025, Երևան, Ալեք Մանուկյան 1:

Ատենախոսությանը կարելի է ծանոթանալ Երևանի պետական համալսարանի գրադարանում:

Սեղմագիրն առաքված է 2012թ. ապրիլի 22-ին:

Մասնագիտական խորհրդի գիտական քարտուղար՝
ֆիզ. մաթ. գիտ. թեկնածու Վ.Ժ. Դումանյան

Тема диссертации утверждена в Ереванском государственном университете

Научный руководитель:	доктор физ.-мат. наук	С.А. Нигиян
Официальные оппоненты:	доктор физ.-мат. наук	Ю.М. Мовсисян
	кандидат физ.-мат. наук	Л.О. Хачоян
Ведущая организация:	Институт проблем информатики и автоматизации НАН РА	

Защита диссертации состоится 23-го мая 2012 г. в 16^{00} часов на заседании действующего в ЕГУ специализированного совета ВАК 044 “Математическая кибернетика и математическая логика” по адресу: 0025 г. Ереван, ул. Алека Манукяна, 1.

С диссертацией можно ознакомиться в библиотеке Ереванского государственного университета.

Автореферат разослан 22-го апреля 2012 г.

Ученый секретарь специализированного совета,
кандидат физ.-мат. наук В.Ж. Думанян

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Диссертационная работа посвящена проблемам логического программирования. Объектами исследования являются логические программы и запросы, использующие отрицание. Предпринимается попытка распространить методы логического программирования на формулы более сложного вида, нежели хорновские дизъюнкты, для того, чтобы увеличить выразительные возможности логических программ. На самом деле невозможно выразить негативные знания посредством хорновских дизъюнктов, а программа в логическом программировании по определению состоит из хорновских дизъюнктов, пригодных для представления только позитивной информации. Для представления и получения негативной информации необходимо ввести специальные правила, руководствуясь принципами, важнейшими из которых являются допущение замкнутости мира (*Closed World Assumption*) и отрицание по неуспеху (*Negation as Failure*). В реальных системах логического программирования программы могут содержать отрицание в телах своих правил, отрицание могут содержать также и запросы. И конечно, системы логического программирования должны быть снабжены механизмом обработки отрицательных подцелей. Метод, который обычно выбирается при таких случаях, это расширение *SLD*-резолюции, основываясь на принципе отрицания по неуспеху (*NF*), при использовании которого отрицательная цель характеризуется неудачей или успехом соответствующей цели без отрицания. Расширенная таким образом резолюция называется *SLDNF*-резолюцией[1]. Семантика отрицания по неуспеху не совпадает с семантикой логического отрицания: нельзя считать что-то неверным, потому что оно является недоказуемым. Чтобы дать теоретическое обоснование корректности *SLDNF*-резолюции и совместить логические и процедурные отрицания, вводится понятие замыкания программы. Основная идея состоит в том, чтобы рассматривать предикаты, описанные в некоторой программе *P* как полностью определенные. Т. е. программе *P* сопоставляется формула логики предикатов первого порядка с равенством *comp(P)* таким образом, чтобы логическая семантика соответствовала допущению замкнутости мира[1,2].

В работах[1,2] рассматривались чистые логические программы и запросы, использующие отрицание. Было дано определение формулы *comp(P)*, сопоставляемой программе *P*, а также определение *SLDNF*-резолюции. Была доказана непротиворечивость *SLDNF*-резолюции в том смысле, что если для некоторого запроса *Q* к программе *P* можно получить положительный ответ, то из формулы *comp(P)* логически следует *Q*, а в случае отрицательного ответа из формулы *comp(P)* логически следует отрицание запроса *Q*[2]. Показано, что в общем случае *SLDNF*-резолюция не обладает свойством полноты: т.е. из *comp(P)* может следовать запрос *Q* или его отрицание, однако посредством *SLDNF*-резолюции невозможно будет получить на него ответ. Естественно возникает вопрос о полноте *SLDNF*-резолюции в некоторых, более узких классах программ и запросов. Также

[1] *Clark K.L.* Negation as Failure. // Logic and Database, H. Galaire and J. Minker (Eds.), Plenum Press, New York, 1978, p. 293-322.

[2] *Lloyd J.W.* Foundations of Logic Programming. // Springer-Verlag, 1984, 120p.

представляет интерес вопрос возможности определения таких классов, в которых *SLDNF*-резолюция не является логически полной, однако в случае, если из формулы *comp(P)* логически следует запрос, *SLDNF*-резолюция дает положительный ответ, а также определения классов, в которых при логическом следствии отрицания запроса из формулы *comp(P)*, *SLDNF*-резолюция дает отрицательный ответ.

В работе[1] описаны основания теории хорновского программирования со встроенными предикатами. Предложена модификация правил *SLD*-резолюции на случай использования встроенных предикатов. Доказаны полнота и непротиворечивость модифицированной *SLD*-резолюции. Однако в реальных системах логического программирования могут быть использованы и встроенные предикаты, и отрицание. Здесь возникает задача модификации *SLDNF*-резолюции на случай использования встроенных предикатов. Представляет интерес исследование модифицированной *SLDNF*-резолюции на непротиворечивость.

В работе[1] рассматривались также правила вывода, используемые в реальных системах логического программирования – практические правила вывода (область определения таких правил уже области определения модифицированных правил *SLD*-резолюции). Была доказана логическая непротиворечивость процедурной семантики, основанной на любом практическом правиле вывода. Представляет интерес задача введения понятия практической *SLDNF*-резолюции и исследования ее на непротиворечивость.

Целью диссертационной работы является:

1. Исследование *SLDNF*-резолюции на вопрос ее полноты в некоторых классах обобщенных логических программ и запросов.
2. Модификация *SLDNF*-резолюции на случай использования встроенных предикатов, исследование модифицированной *SLDNF*-резолюции на непротиворечивость.
3. Введение практических правил *SLDNF*-резолюции (в случае использования встроенных предикатов) и исследование их на непротиворечивость.

Методика исследований. Методика исследований включает в себя методы математической логики, теории алгоритмов, а также методы используемые в логическом программировании.

Научная новизна.

1. Исследована *SLDNF*-резолюция на вопрос полноты для некоторых классов обобщенных логических программ и запросов:
   - Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция не является логически полной, но если из формулы *comp(P)* логически следует запрос, *SLDNF*-резолюция дает положительный ответ.

[1] *Нигиян С.А.* Хорновское программирование со встроенными предикатами. //Программирование, N1, 1996, с.30-38

- Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция не является логически полной, но если из *comp(P)* логически следует отрицание запроса, *SLDNF*-резолюция дает отрицательный ответ.
- Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция логически полна.

2. Модифицирована *SLDNF*-резолюция на случай использования встроенных предикатов. Показана непротиворечивость модифицированной *SLDNF*-резолюции.
3. Введены практические правила *SLDNF*-резолюции и показана их непротиворечивость.

Теоретическая и практическая ценность. Результаты, полученные в диссертации, носят как теоретический, так и практический характер. Они могут быть использованы при создании новых логических систем программирования, а также для обоснования их корректности.

Апробация работы и публикации. Основные результаты диссертации докладывались на международных конференциях CSIT-05 (Ереван, 2005), CSIT-07 (Ереван, 2007), CSIT-09 (Ереван, 2009), на семинаре кафедры программирования и информационных технологий ЕГУ, на общем семинаре факультета информатики и прикладной математики ЕГУ.

Основные результаты работы отражены в 5-ти публикациях.

Структура и объем работы. Диссертация состоит из введения, трех глав, заключения и списка используемой литературы (38 наименований). Объем работы – 82 страницы.

СОДЕРЖАНИЕ РАБОТЫ

Во введении описывается задача диссертационного исследования, обосновывается актуальность темы диссертации и её новизна. Дается краткий обзор содержания работы.

Глава 1. В этой главе даются основные определения, приводятся понятия *SLDNF*-резолюции, *SLDNF*-вывода и конечного *SLDNF*-дерева, терпящего неудачу, обсуждается формула *comp(P)*, которая сопоставляется программе *P*, и доказываются теоремы 1.4.1 – 1.4.3 о логической полноте *SLDNF*-резолюции для некоторых классов обобщенных логических программ и запросов. Глава 1 состоит из четырех разделов 1.1 – 1.4.

В разделе 1.1 приводятся основные определения, используемые в работе.

Зафиксируем три непересекающихся счетных множества *X*, *Φ* и *Π*. *X* – множество предметных переменных. *Φ* – множество функциональных символов с приписанной каждому символу местностью, причем для любого $n \geq 0$ *Φ* содержит счетное число символов местности *n*.

Определение 1.1.1. Из элементов множеств Φ и X строятся термы.

1. каждый 0-местный символ из Φ есть терм;
2. каждая переменная из X есть терм;
3. если $t_1,t_2,\ldots,t_n$ $(n>0)$ – термы и f – n-местный символ из Φ, то $f(t_1,t_2,\ldots,t_n)$ есть терм.

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

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

Определение 1.1.2. Атомы определяются традиционным образом:

1. каждый 0-местный символ из Π есть атом;
2. если $t_1,t_2,\ldots,t_n$ $(n>0)$ – термы и p – n-местный символ из Π, то $p(t_1,t_2,\ldots,t_n)$ есть атом.

Атом, не использующий переменных, назовем основным.

Традиционным образом определяется формула логики предикатов первого порядка, использующая логические операции $\neg$, $\&$, $\vee$, $\supset$, $\sim$ и кванторы $\exists$, $\forall$.

Определение 1.1.3. Опишем рассматриваемые нами интерпретации. Предметным множеством рассматриваемых интерпретаций является множество M. Функциональные символы интерпретируются следующим образом: каждому 0-местному символу из Φ сопоставляется он сам, каждому n-местному $(n>0)$ символу $f\in\Phi$ сопоставляется отображение $M^n\rightarrow M$, которое n-ке $\langle t_1,\ldots,t_n\rangle\in M^n$ ставит в соответствие терм $f(t_1,\ldots,t_n)$. Каждому 0-местному символу из Π сопоставляется один из элементов множества $\{true, false\}$, а каждому n-местному $(n>0)$ символу из Π сопоставляется некоторое отображение $M^n\rightarrow \{true, false\}$.

Обозначим описанное множество интерпретаций через H. Заметим, что интерпретации из H могут отличаться одна от другой лишь отображениями, сопоставляемыми символам множества Π. Каждую интерпретацию $I\in H$ можно отождествлять с множеством тех основных атомов, значения которых на I есть *true*.

Формула, не имеющая свободных переменных, называется замкнутой.

Пусть F – замкнутая формула и I – интерпретация из H. Значение формулы F на интерпретации I определяется традиционным образом и обозначается $Val_I(F)$. Интерпретация I называется моделью формулы F, если значение формулы F на интерпретации I есть *true* и основные атомы, принадлежащие I, используют только предикатные символы формулы F. Будем говорить, что формула F выполнима, если она имеет модель. Пусть F и F' – замкнутые формулы. Будем говорить, что формула F' является логическим следствием формулы F и обозначать это $F\models F'$, если для всякой модели I формулы F, значение формулы F' есть *true*. Будем говорить, что формула F эквивалентна формуле F' и обозначать это $F\approx F'$, если на любой интерпретации $I\in H$ $Val_I(F)=Val_I(F')$.

Определение 1.1.4. Литерал – это атом или его отрицание. Позитивный литерал – это атом, а отрицательный литерал – это отрицание атома.

Пусть K является термом или литералом. Через $Var(K)$ обозначим множество всех переменных используемых в K.

Определение 1.1.5. Подстановка σ есть множество вида: $\{t_1/x_1,\ldots,t_n/x_n\}$, где t_i – терм, x_i – переменная, $t_i \neq x_i$, $i \neq j \Rightarrow x_i \neq x_j$, $i,j=1,..,n$, $n \geq 0$.

$Arg(\sigma) = \{x_1,\ldots,x_n\}$,
$Var(\sigma) = Var(t_1) \cup \ldots \cup Var(t_n)$.

Пусть E – формула, не использующая кванторы. Через $E\sigma$ обозначим формулу, полученную из E путем одновременной подстановки термов $t_1,\ldots,t_n$ вместо переменных $x_1,\ldots,x_n$ соответственно.

Определение 1.1.6. Будем говорить, что атомы A_1 и A_2 унифицируемы, если существует такая подстановка δ, что $A_1\delta = A_2\delta$. Подстановку δ назовем унификатором A_1 и A_2. Унификатор σ назовем наиболее общим унификатором A_1 и A_2 ($\sigma = mgu(A_1,A_2)$), если для любого их унификатора δ существует подстановка γ такая, что $\sigma\gamma = \delta$.

В разделе 1.2 вводятся понятия обобщенной логической программы и обобщенного запроса, дается определение формулы *comp(P)*, которая сопоставляется программе *P*. Определения обобщаются на случай использования программой *P* и запросом *Q* 0-местных предикатных символов. В данном разделе получены также результаты, касающиеся формулы *comp(P)*.

Определение 1.2.1. Обобщенная логическая программа (далее просто программа) есть последовательность предложений $S_1,\ldots,S_r$, $r>0$. Предложение $S \in \{S_1,\ldots,S_r\}$ имеет вид $A{:-}L_1,\ldots,L_m$, $m \geq 0$, где A – атом, а L_i – литерал, $i=1,\ldots,m$. Атом A называется головой предложения, последовательность $L_1,\ldots,L_m$ – его телом, а число m – длиной тела. Если $m=0$, S является фактом, а если $m>0$, S является правилом.

Определение 1.2.2. Множество всех предложений программы, в головах которых используется предикатный символ p, назовем определением предиката p.

Программе *P* сопоставляется формула *comp(P)*.

Определение 1.2.4. Формула *comp(P)* имеет следующий вид:

$$F(p_1) \,\&\, \ldots \,\&\, F(p_u),$$

где $p_1,\ldots,p_u$ предикатные символы программы P ($u>0$), и каждое $F(p)$, где $p \in \{p_1,\ldots,p_u\}$, определяется следующим образом:

Если p – 0-местный предикатный символ, и p не является головой ни одного из предложений программы P, то $F(p)$ есть $\neg p$. Если p является фактом программы P, то $F(p)$ есть p. Если же определением p являются правила p :- B_1 ,…, p :- B_v, где B_i – тело правила p :- B_i, $i=1,\ldots,v$, $v \geq 1$, то $F(p)$ формула следующего вида:

$$p \sim E_1 \vee \ldots \vee E_v.$$

E_i – это формула вида

$$\exists y_1 \ldots \exists y_d (L_1 \& \ldots \& L_m),$$

где $y_1,\ldots,y_d$ ($d \geq 0$) – переменные правила p :- B_i, и B_i имеет вид $L_1,\ldots,L_m$, $m \geq 1$, $i=1,\ldots,v$.

Если p – n-местный ($n \geq 1$) предикатный символ и p не встречается в головах предложений программы P, то $F(p)$ есть формула

$$\forall x_1 \ldots \forall x_n \neg p(x_1,\ldots,x_n).$$

Если же определением p являются предложения A_1 :- B, … , A_v :- B_v, где B_i есть тело (возможно пустое) предложения A_i :- B_i, $i=1,\ldots,v$, $v \geq 1$, то $F(p)$ имеет следующий вид:

$$\forall x_1 \ldots \forall x_n (p(x_1,\ldots,x_n) \sim E_1 \vee \ldots \vee E_v),$$

где $x_1,\ldots,x_n$ - переменные, не встречающиеся в предложениях $A_1 :- B_1,\ldots, A_v :- B_v$ и каждое E_i формула вида

$$\exists y_1 \ldots \exists y_d((x_1=t_1)\&\ldots\&(x_n=t_n)\&L_1\&\ldots\&L_m),$$

где $y_1,\ldots,y_d$ $(d\geq 0)$ – переменные предложения $A_i :- B_i$; A_i имеет вид $p(t_1,\ldots,t_n)$, а B_i есть $L_1,\ldots,L_m$, $m\geq 0$, $i=1,\ldots,v$.

Определение 1.2.5. Обобщенный запрос (далее просто запрос) Q имеет вид $?–L_1,\ldots,L_k$, где L_i - литерал, $i=1,\ldots,k$, $k\geq 0$. Число k назовем длиной запроса Q. Если $k=0$, то запрос называется пустым. Непустому запросу Q сопоставляется формула

$$\exists y_1 \ldots \exists y_s\ (L_1\& \ldots \&L_k),$$

где $y_1,\ldots,y_s$ - все переменные, использованные в литералах $L_1,\ldots,L_k$, $s\geq 0$. Множество $\{y_1,\ldots,y_s\}$ обозначим через $Var(Q)$.

Пусть σ есть некоторая подстановка, Q есть запрос $?–L_1,\ldots,L_k$ $(k>0)$, тогда через $Q\sigma$ обозначим запрос $?–L_1\sigma,\ldots,L_k\sigma$.

В разделе 1.3 приводятся понятия *SLDNF*-резолюции, *SLDNF*-вывода, конечного *SLDNF*-дерева, терпящего неудачу и обсуждаются свойства *SLDNF*-резолюции, аналогичные установленным в [1], но уже с использованием *0*-местных предикатных символов.

Определим класс правил вычисления.

Определение 1.3.1. Каждое правило ρ определяется заданием функции Sel_ρ. Значением аргумента функции Sel_ρ является непустой запрос, и если Q есть запрос $?–L_1,\ldots,L_k$ $(k>0)$, то $Sel_\rho(Q)\in\{1,\ldots,k\}$. Правило вычисления ρ является безопасным (для *SLDNF*-резолюции), если выполняются следующие условия:

1. Правило ρ выбирает в непустом запросе положительный литерал, или основной отрицательный литерал.

2. Выбирая основной отрицательный литерал $\neg A$ в некотором запросе, ρ пытается завершить построение *SLDNF*-дерева, терпящего неудачу с корнем $?-A$, до того, как продолжить вычисление.

Далее, говоря о правиле вычисления, будем иметь ввиду, что оно безопасное.

Определение 1.3.2. Пусть P – программа, Q – непустой запрос и ρ – правило вычисления. *SLDNF*-вывод $Q_1,Q_2,\ldots$ для пары (P,Q) и правила ρ, где $Q_1=Q$, определяется следующим образом:

Пусть Q_i $(i\geq 1)$ – это запрос $?-L_1,\ldots,L_k$ и $Sel_\rho(Q_i)=j$, $1\leq j\leq k$, $k>0$.

Если L_j является атомом, $A:-K_1,\ldots,K_m$ $(m\geq 0)$ – предложение из P, L_j с A унифицируемы, и $\sigma=mgu(L_j,A)$, то запрос Q_{i+1} имеет вид

$$?-\ L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_k\sigma.$$

Если L_j является основным отрицательным литералом и имеет вид $\neg A$, то делается попытка завершить построение *SLDNF*-дерева с корнем $?-A$. И если из пары $(P,?-A)$ выводим пустой запрос, то вывод прерывается. Если же запрос $?-A$ терпит неудачу (т. е. строится конечное *SLDNF*-дерево, терпящее неудачу, с корнем $?-A$), то подцель $\neg A$ удовлетворяется и запрос Q_{i+1} имеет вид

$$?-L_1,\ldots,L_{j-1},L_{j+1},\ldots,L_k.$$

[1] *Lloyd J.W.* Foundations of Logic Programming. // Springer-Verlag, 1984, 120p.

Определение 1.3.3. Пусть P – программа, Q – непустой запрос и ρ – правило вычисления. *SLDNF*-дерево для пары (P,Q) и правила ρ определяется следующим образом:

1. каждая вершина дерева является запросом;
2. корнем является запрос Q;
3. пусть Q' есть вершина $?\text{-}L_1,\ldots,L_k$, $Sel_\rho(Q')=j$ и L_j является атомом $(1\le j\le k,\ k>0)$. Тогда эта вершина для каждого предложения $A\text{:-}K_1,\ldots,K_m$ $(m\ge 0)$ программы P, для которого L_j и A унифицируемы, имеет потомком запрос

$$?\text{-}L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_k\sigma,$$

где $\sigma=mgu(L_j,A)$. Если таких предложений нет, то вершина не имеет потомков.

4. пусть Q' есть вершина $?\text{-}L_1,\ldots,L_k$, $Sel_\rho(Q')=j$ и L_j – основной отрицательный литерал $\neg A$ $(1\le j\le k,\ k>0)$. Если запрос $?\text{-}A$ терпит неудачу, то единственным потомком вершины является запрос

$$?\text{-}L_1,\ldots,L_{j-1},L_{j+1},\ldots,L_k.$$

Если из пары $(P,?\text{-}A)$ выводим пустой запрос, то вершина не имеет потомков.

Пустые запросы не имеют потомков.

Определение 1.3.4. Если правило ρ выбирает отрицательный литерал, который содержит переменную, то запрос отвергается. Если *SLDNF*-дерево для пары (P,Q) и правила ρ имеет конечное число вершин, не имеет ни одного вывода пустого запроса и имеет хотя бы один вывод отвергающегося запроса, то отвергается и запрос Q. *SLDNF*-дерево, которое имеет конечное число вершин, не имеет ни одного вывода пустого запроса, и ни один из его листьев не отвергается, называется *SLDNF*-деревом, терпящим неудачу.

Определение 1.3.5. Если *SLDNF*-дерево для пары (P,Q) и правила ρ имеет вывод, заканчивающийся пустым запросом, то будем говорить, что из пары (P,Q) ρ выводим пустой запрос и обозначать это $(P,Q)\vdash_\rho ?\text{-}$.

Теорема 1.3.1. Пусть P – программа, Q – непустой запрос $?\text{-}\ L_1,\ldots,L_n$ $(n>0)$ и ρ – правило вычисления, тогда:

a) если $(P,Q)\vdash_\rho ?\text{-}$, $Q_1,\ldots,Q_s$ $(s>1)$ есть *SLDNF*-вывод пустого запроса и $\sigma_1,\ldots,\sigma_{s-1}$ – подстановки, соответствующие данному выводу, то

$$comp(P)\models\forall((L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_{s-1});$$

b) если для правила ρ (P,Q) имеет *SLDNF*-дерево, терпящее неудачу, то

$$comp(P)\models\neg Q.$$

В разделе 1.4 вводится понятие системы свойств программ и запросов, с помощью которой задается некоторый класс программ и запросов. Также вводятся понятия несокращаемой логической полноты, несокращаемой положительной полноты и несокращаемой отрицательной полноты таких систем. Для одной системы доказывается ее несокращаемая положительная полнота. Для другой системы, отличающейся от первой только свойствами запросов, доказывается ее несокращаемая отрицательная полнота. А для третьей системы, которая отличается от первых двух свойствами запросов, доказывается ее несокращаемая логическая полнота.

Пусть $Prog_0$ – множество всех программ, для которых имеет место следующее: если $P \in Prog_0$, то *comp(P)* – выполнимая формула. $Quer_0$ – множество всех непустых запросов.

Пусть $Prog \subset Prog_0$, $Quer \subset Quer_0$, и ρ – некоторое правило вычисления.

Определение 1.4.1. Будем говорить, что *SLDNF*-резолюция ρ-положительно полна для пары *(Prog, Quer)*, если для любой программы $P \in Prog$ и для любого запроса $Q \in Quer$ имеем: $comp(P) \models Q \Rightarrow (P,Q) \vdash_\rho ?$-.

Определение 1.4.2. Будем говорить, что *SLDNF*-резолюция ρ-отрицательно полна для пары *(Prog,Quer)*, если для любой программы $P \in Prog$ и для любого запроса $Q \in Quer$ имеем: $comp(P) \models \neg Q \Rightarrow$ для пары *(P,Q)* и правила ρ существует *SLDNF*-дерево, терпящее неудачу.

Определение 1.4.3. Будем говорить, что *SLDNF*-резолюция положительно полна (соответственно, отрицательно полна) для пары *(Prog, Quer)*, если она ρ-положительно полна (соответственно, ρ-отрицательно полна) для любого правила вычисления ρ.

Определение 1.4.4. Будем говорить, что *SLDNF*-резолюция логически полна для пары *(Prog, Quer)*, если она одновременно положительно полна и отрицательно полна для пары *(Prog, Quer)*.

Определение 1.4.4. Любое непустое подмножество $Prog' \subset Prog_0$ назовем свойством программ, а любое непустое подмножество $Quer' \subset Quer_0$ – свойством запросов.

n-ку свойств программ $<Prog_1,...,Prog_n>$ назовем системой свойств программ, если $\bigcap_{i=0}^{n} Prog_i \neq \emptyset$, где $Prog_i \subset Prog_0$, $i=1,\ldots,n$, $n \geq 0$.

n-ку свойств запросов $<Quer_1,\ldots,Quer_n>$ назовем системой свойств запросов, если $\bigcap_{i=0}^{n} Quer_i \neq \emptyset$, где $Quer_i \subset Quer_0$, $i=1,\ldots,n$, $n \geq 0$.

Определение 1.4.5. Пару $<<Prog_1,...,Prog_n>,<Quer_1,\ldots,Quer_k>>$, где $<Prog_1,...,Prog_n>$ система свойств программ, $<Quer_1,\ldots,Quer_k>$ – система свойств запросов, назовем системой свойств программ и запросов, $n,k \geq 0$.

Определение 1.4.6. Будем говорить, что *SLDNF*-резолюция логически полна для системы свойств программ и запросов $<<Prog_1,...,Prog_n>, <Quer_1,\ldots,Quer_k>>$, если она логически полна для пары $(\bigcap_{i=0}^{n} Prog_i, \bigcap_{i=0}^{k} Quer_i)$, $n,k \geq 0$.

Определение 1.4.7. Систему свойств программ и запросов $<<Prog_1,...,Prog_n>, <Quer_1,\ldots,Quer_k>>$ $(n,k \geq 0)$ назовем несокращаемой положительно полной (соответственно, несокращаемой отрицательно полной), если *SLDNF*-резолюция положительно полна (соответственно, отрицательно полна) для пары $(\bigcap_{i=0}^{n} Prog_i, \bigcap_{i=0}^{k} Quer_i)$, но положительно не полна (соответственно, отрицательно не полна) для каждой из пар $(\bigcap_{i=0, i\neq j}^{n} Prog_i, \bigcap_{i=0}^{k} Quer_i)$ и $(\bigcap_{i=0}^{n} Prog_i, \bigcap_{i=0, i\neq h}^{k} Quer_i)$, где $j=1,...,n$, $h=1,...,k$, $n,k \geq 0$.

Определение 1.4.8. Систему свойств программ и запросов $<<Prog_1,...,Prog_n>, <Quer_1,\ldots,Quer_k>>$ $(n,k \geq 0)$ назовем несокращаемой логически полной, если она одновременно является несокращаемой положительно полной и несокращаемой отрицательно полной.

Рассмотрим следующие свойства программ $Prog_1$, $Prog_2$, $Prog_3$ и свойства запросов $Quer_1$, $Quer_2$:

$Prog_1$ – множество программ, не использующих переменных;

$Prog_2$ – множество программ, у которых тела правил имеют длину 1;

$Prog_3$ – множество программ, у которых в головах предложений атомы не повторяются;

$Quer_1$ – множество запросов, не использующих переменных;

$Quer_2$ – множество запросов длины 1.

Теорема 1.4.1. Система свойств программ и запросов<<$Prog_1$, $Prog_2$, $Prog_3$>, <$Quer_1$>> является несокращаемой положительно полной и не является отрицательно полной.

Теорема 1.4.2. Система свойств программ и запросов <<$Prog_1$, $Prog_2$, $Prog_3$>, <$Quer_2$>> является несокращаемой отрицательно полной и не является положительно полной.

Теорема 1.4.3. Система свойств программ и запросов <<$Prog_1$, $Prog_2$, $Prog_3$>, <$Quer_1$, $Quer_2$>> является несокращаемой логически полной.

В главе 2 предлагается модификация *SLDNF*-резолюции на случай использования встроенных предикатов и доказывается непротиворечивость модифицированной *SLDNF*-резолюции.

В разделе 2.1 определения логической программы, формулы *comp(P)*, сопоставляемой программе *P*, запроса, *SLDNF*-резолюции, *SLDNF*-вывода и конечного *SLDNF*-дерева, терпящего неудачу, модифицируются на случай использования встроенных предикатов.

В случае использования встроенных предикатов множество $П$ представляет собой объединение двух непересекающихся множеств: $П=П_1 \cup П_2$, где $П_1$ – множество предикатных символов с приписанной каждому символу местностью, для любого $n \geq 0$ $П_1$ 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 $M^k \rightarrow \{true, false\}$.

Далее атом, использующий предикатный символ из $П_1$, назовем предикатным термом.

Определение 2.1.1. Литерал – это предикатный терм или его отрицание. Позитивный литерал – это предикатный терм, а отрицательный литерал – отрицание предикатного терма.

Определение 2.1.2. Атом, использующий встроенный предикат из $П_2$, назовем условием.

Пусть K является термом, литералом или условием. Через $Var(K)$ обозначим множество всех переменных используемых в K.

Определение 2.1.3. Логическая программа со встроенными предикатами (далее просто программа) представляет собой последовательность предложений $S_1,\ldots,S_r$, $r>0$. Предложение $S \in \{S_1,\ldots,S_r\}$ имеет вид $A:\text{-}L_1,\ldots,L_m$, $m \geq 0$, где A –

предикатный терм, а L_i – литерал или условие, $i=1,\ldots,m$. Предикатный терм A называется головой предложения, последовательность $L_1,\ldots,L_m$ – его телом, а число m – длиной тела. Если $m=0$, S является фактом, а если $m>0$, S является правилом.

Программе P сопоставляется формула $comp(P)$, которая строится точно так же, как и в случае чистого логического программирования.

Определение 2.1.4. Запрос со встроенными предикатами (далее просто запрос) Q имеет вид $?\text{–}L_1,\ldots,L_k$, где L_i - литерал или условие, $i=1,\ldots,k$, $k\geq 0$. Число k назовем длиной запроса Q. Если $k=0$, то запрос называется пустым. Непустому запросу Q сопоставляется формула

$$\exists y_1 \ldots \exists y_s(L_1 \& \ldots \& L_k),$$

где $y_1, \ldots, y_s$– все переменные, использованные в литералах и условиях $L_1,\ldots,L_k$, $s\geq 0$. Множество $\{y_1,\ldots,y_s\}$ обозначим через $Var(Q)$.

Определим множество правил вычисления R, основанных на модифицированной (на случай встроенных предикатов) *SLDNF*-резолюции.

Определение 2.1.5. Каждое правило $\rho\in R$ определяется заданием двух функций: Sel_ρ и Sub_ρ. Значениями аргументов этих функций являются непустые запросы. Пусть Q есть запрос $? - L_1,\ldots,L_k$ ($k>0$). Опишем $Sel_\rho(Q)$ и $Sub_\rho(Q)$.

$Sel_\rho(Q)\in\{1,\ldots,k\}$. Пусть $Sel_\rho(Q)=s$, $1\leq s\leq k$, тогда $Sub_\rho(Q)$ не определено, если L_s - литерал и $Sub_\rho(Q)$ равно некоторому (возможно пустому) множеству подстановок, если L_s – условие.

Пусть $Sub_\rho(Q)$ – определено. Опишем свойства множества $Sub_\rho(Q)$ в этом случае. Каждая подстановка $\sigma\in Sub_\rho(Q)$ должна удовлетворять следующим трем условиям:

a) $Arg(\sigma)\subset Var(L_s)$,

b) $Var(\sigma)\cap(Var(Q)\setminus Var(L_s))=\varnothing$,

c) $Val(L_s\sigma\gamma)=true$ для любой подстановки γ такой, что $Var(L_s\sigma\gamma)=\varnothing$.

Кроме того для любой подстановки δ такой, что $Var(L_s\delta)=\varnothing$ и $Val(L_s\delta)=true$, должны существовать подстановки $\sigma\in Sub_\rho(Q)$ и γ такие, что $L_s\delta=L_s\sigma\gamma$. Заметим, что $Sub_\rho(Q)=\varnothing$, если не существует подстановки δ такой, что $Var(L_s\delta)=\varnothing$ и $Val(L_s\delta)=true$.

Определение 2.1.6. Правило вычисления $\rho\in R$ должно быть безопасным, т. е. должно выполняться следующее:

1. Правило ρ выбирает в непустом запросе условие, положительный литерал, или основной отрицательный литерал.

2. Выбирая основной отрицательный литерал $\neg A$ в некотором запросе, ρ пытается завершить построение *SLDNF*-дерева, терпящего неудачу с корнем $?\text{-}A$, до того, как продолжить вычисление.

Далее, говоря о правиле вычисления, будем иметь ввиду, что оно безопасное.

Определение 2.1.7. Пусть P – программа, Q – непустой запрос и $\rho\in R$. *SLDNF*-вывод $Q_1,Q_2,\ldots$ для пары (P,Q) и правила ρ, где $Q_1=Q$, определяется следующим образом:

Пусть Q_i $(i\geq 1)$ – это запрос $?\text{-}L_1,\ldots,L_k$ и $Sel_\rho(Q_i)=j$, $1\leq j\leq k$, $k>0$.

Если L_j является предикатным термом, $A\text{:-}K_1,\ldots,K_m$ $(m\geq 0)$ – предложение из P такое, что L_j с A унифицируемы, и $\sigma=mgu(L_j,A)$,то запрос Q_{i+1} имеет вид

$$?\text{-}\ L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_k\sigma.$$

Если L_j является условием, $Sub_\rho(Q_i)\neq\varnothing$, то запрос Q_{i+1} имеет вид

$$?\text{-}\ L_1\delta,\ldots,L_{j-1}\delta,L_{j+1}\delta,\ldots,L_k\delta,$$

где $\delta\in Sub_\rho(Q_i)$.

Если L_j является основным отрицательным литералом и имеет вид $\neg A$, то делается попытка завершить построение *SLDNF*-дерева с корнем *?-A*. Если из пары *(P,?-A)* выводим пустой запрос, то вывод прерывается. Если же запрос *?-A* терпит неудачу (т. е. строится конечное *SLDNF*-дерево, терпящее неудачу), то подцель $\neg A$ удовлетворяется и запрос Q_{i+1} имеет вид $?\text{-}L_1,\ldots,L_{j-1},L_{j+1},\ldots,L_k$.

Определение 2.1.8. Пусть P – программа, Q – непустой запрос и $\rho\in R$. *SLDNF*-дерево для пары *(P,Q)* и правила ρ определяется следующим образом:

1. каждая вершина дерева является запросом;
2. корнем является запрос *Q;*
3. пусть *Q'* есть вершина $?\text{-}L_1,\ldots,L_k$, $Sel_\rho(Q')=j$ и L_j является предикатным термом $(1\le j\le k,\ k>0)$. Тогда эта вершина для каждого предложения $A\text{:-}K_1,\ldots,K_m$ $(m\ge 0)$ программы *P*, для которого L_j и *A* унифицируемы, имеет потомком запрос

$$?\text{-}L_1\sigma,\ldots,L_{j-1}\sigma,K_1\sigma,\ldots,K_m\sigma,L_{j+1}\sigma,\ldots,L_k\sigma,$$

где $\sigma=mgu(L_j,A)$. Если таких предложений нет, то вершина не имеет потомков;

4. пусть *Q'* есть вершина $?\text{-}L_1,\ldots,L_k$, $Sel_\rho(Q')=j$ и L_j - основной отрицательный литерал $\neg A$ $(1\le j\le k,\ k>0)$. Если запрос *?-A* терпит неудачу, то единственным потомком вершины является запрос

$$?\text{-}L_1,\ldots,L_{j-1},L_{j+1},\ldots,L_k.$$

Если из пары *(P,?-A)* выводим пустой запрос, то вершина не имеет потомков.

5. пусть *Q'* есть вершина $?\text{-}L_1,\ldots,L_k$, $Sel_\rho(Q')=j$ и L_j – условие $(1\le j\le k)$. Если $Sub_\rho(Q')\neq\varnothing$, то эта вершина для каждого $\delta\in Sub_\rho(?\text{-}L_1,\ldots,L_k)$ имеет потомком запрос

$$?\text{-}\ L_1\delta,\ldots,L_{j-1}\delta,L_{j+1}\delta,\ldots,L_k\delta.$$

Если $Sub_\rho(?\text{-}L_1,\ldots,L_k)=\varnothing$, то вершина не имеет потомков.

6. пустые запросы не имеют потомков.

Определение отвергающегося запроса и *SLDNF*-дерева, терпящего неудачу, совпадают с определениями в случае чистого логичекого программирования.

В разделе 2.2 доказывается теорема 2.2.1 о непротиворечивости модифицированной *SLDNF*-резолюции.

Теорема 2.2.1. Пусть P – программа, Q – непустой запрос $?\text{-}\ L_1,\ldots,L_n$ $(n>0)$ и $\rho\in R$, тогда:

a) если $(P,Q)\vdash_\rho ?\text{-},\ Q_1,\ldots,Q_s$ $(s>1)$ есть *SLDNF*-вывод пустого запроса и $\sigma_1,\ldots,\sigma_{s-1}$– подстановки, соответствующие данному выводу, то

$$comp(P)\models\forall((L_1\&\ldots\&L_n)\sigma_1\ldots\sigma_{s-1});$$

b) если для правила ρ *(P,Q)* имеет *SLDNF*-дерево, терпящее неудачу, то

$$comp(P)\models\neg Q.$$

Глава 3. В данной главе вводится понятие практического правила вычисления. Даются определения *SLDNF*-вывода и *SLDNF*-дерева в случае использования практических правил. Доказывается, что *SLDNF*-резолюция, основанная на любом практическом правиле вычисления, непротиворечива

Определение 3.1.1. Пусть $\rho \in R$, определим множество правил R_ρ. Каждому правилу $\rho' \in R_\rho$, так же, как и правилу ρ, сопоставляются две функции $Sel_{\rho'}$ и $Sub_{\rho'}$. Причем $Sel_{\rho'}=Sel_\rho$, область определения функции $Sub_{\rho'}$ входит в область определения функции Sub_ρ, и если $Sub_{\rho'}(Q)$ определено, то $Sub_{\rho'}(Q) = Sub_\rho(Q)$, где Q – непустой запрос.

Определение 3.1.2. Пусть P – программа, Q – непустой запрос $?\text{-}L_1,\ldots,L_k$, $\rho \in R$ и $\rho' \in R_\rho$. Будем говорить, что Q отвергается правилом ρ', если имеет место один из следующих пунктов:

1. $Sel_{\rho'}(Q)=j$, L_j является отрицательным литералом, содержащим переменные, $1 \le j \le k$;
2. $Sel_{\rho'}(Q)=j$, L_j является условием и $Sub_{\rho'}(Q)$ не определено, $1 \le j \le k$;
3. *SLDNF*-дерево для пары (P,Q) и правила ρ' имеет конечное число вершин, не имеет ни одного вывода пустого запроса и имеет хотя бы один вывод запроса, который отвергается правилом ρ'.

SLDNF-вывод и *SLDNF*-дерево для пары (P,Q) и правила ρ', определяются аналогично тому, как это делалось для правил множества R. И если для пары (P,Q) и правила ρ' существует *SLDNF*-вывод некоторого запроса G, то это обозначим $(P,Q) \vdash_{\rho'} G$.

Определение 3.1.4. *SLDNF*-дерево, которое имеет конечное число вершин, не имеет ни одного вывода пустого запроса, ни один из его листьев не отвергается правилом ρ', называется *SLDNF*-деревом, терпящим неудачу.

Пусть $\rho \in R$. Обозначим через $Reject(\rho')$ множество запросов, которые отвергаются правилом ρ', где $\rho' \in R_\rho$. Введем частичный порядок на множестве R_ρ. Пусть ρ', $\rho'' \in R_\rho$, тогда $\rho' < \rho''$, если $Reject(\rho'') \subset Reject(\rho')$. R_ρ является полной решеткой, наибольшим ее элементом является правило ρ, а наименьшим элементом R_ρ является правило, которое отвергает все запросы, для которых $Sub_\rho(Q)$ определено.

Теорема 3.1.1. Пусть P – программа, Q – непустой запрос, $\rho \in R$ и $\rho', \rho'' \in R_\rho$, тогда:

a) $\rho' < \rho'' \Rightarrow$ если $(P,Q) \vdash_{\rho'} ?\text{-}$, то $(P,Q) \vdash_{\rho''} ?\text{-}$, а если для правила ρ' (P,Q) имеет *SLDNF*-дерево, терпящее неудачу, то (P,Q) имеет также *SLDNF*-дерево, терпящее неудачу для правила ρ'', и эти деревья совпадают;

b) если $(P,Q) \vdash_\rho ?\text{-}$, то $(P,Q) \vdash_\rho ?\text{-}$, а если для правила ρ' (P,Q) имеет *SLDNF*-дерево, терпящее неудачу, то (P,Q) имеет также *SLDNF*-дерево, терпящее неудачу для правила ρ, и эти деревья совпадают.

Из теоремы 2.2.1 и теоремы 3.1.1 следует теорема 3.1.2, которая гласит о непротиворечивости практической *SLDNF*-резолюции.

Теорема 3.1.2. Пусть P – программа, Q – непустой запрос $?\text{-}L_1,\ldots,L_n$ $(n>0)$, $\rho \in R$ и $\rho' \in R_\rho$, тогда:

a) если $(P,Q) \vdash_{\rho'} ?\text{-}$, $Q_1,\ldots,Q_s$ $(s>1)$ есть *SLDNF*-вывод пустого запроса и $\sigma_1,\ldots,\sigma_{s-1}$ – подстановки, соответствующие данному выводу, то
$$comp(P) \models \forall((L_1 \& \ldots \& L_n)\sigma_1 \ldots \sigma_{s-1});$$

b) если для правила ρ' (P,Q) имеет *SLDNF*-дерево, терпящее неудачу, то
$$comp(P) \models \neg Q.$$

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

1. Исследована *SLDNF*-резолюция на вопрос полноты для некоторых классов обобщенных логических программ и запросов:
   - Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция не является логически полной, но если из формулы *comp(P)* логически следует запрос, *SLDNF*-резолюция дает положительный ответ.
   - Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция не является логически полной, но если из *comp(P)* логически следует отрицание запроса, *SLDNF*-резолюция дает отрицательный ответ.
   - Определен класс обобщенных логических программ и запросов, в котором *SLDNF*-резолюция логически полна.

   Классы задаются посредством совокупности свойств, которыми обладают все программы и запросы этого класса. Показывается, что при удалении любого из этих свойств, *SLDNF*-резолюция теряет указанные особенности.
2. Модифицирована *SLDNF*-резолюция на случай использования встроенных предикатов. Показана непротиворечивость модифицированной *SLDNF*-резолюции.
3. Введены практические правила *SLDNF*-резолюции (в случае использования встроенных предикатов) и показана их непротиворечивость.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. *Nigiyan S.A., Sargsyan L.A.* On Negation in Logic Programming.// Proceedings of the International Conference CSIT-05, Yerevan, 2005, p. 109-112.
2. *Sargsyan L.A.* On One Case of Completeness of *SLDNF*-resolution.// Proceedings of the International Conference CSIT-07, Yerevan, 2007, p. 66-67.
3. *Нигиян С.А., Саркисян Л.А.* Об SLDNF-резолюции в чистом логическом программировании с отрицанием. // Вестник РАУ, Серия физико-математические и естественные науки, N1 – Ер.: Изд-во РАУ, 2007, с. 48-67.
4. *Nigiyan S.A., Sargsyan L.A.* Modification of SLDNF-resolution for Built-in Predicates. //Proceedings of the International Conference CSIT-09, Yerevan, 2009, p. 60-61.
5. *Sargsyan L.A.* On *SLDNF*-resolution in Logic Programming with Negation. //Proceedings of the Yerevan State University, Physical and Mathematical Sciences, №3, YSU Press, Yerevan, 2011, p. 31-39.

ԱՄՓՈՓԱԳԻՐ

Սարգսյան Լուսինե Արամայիսի

«*SLDNF*-ռեզոլյուցիայի մասին ժխտումով տրամաբանական ծրագրավորման մեջ»

Ատենախոսությունը նվիրված է տրամաբանական ծրագրավորման պրոբլեմնե-րին: Հետազոտվում են ընդհանրացված տրամաբանական ծրագրեր և հարցումներ (տրամաբանական ծրագրեր և հարցումներ, որոնցում կարող է ժխտում օգտագործվել):

Հորնի ծրագրավորման դեպքում տրամաբանական ծրագիրը բաղկացած է միայն դրական ինֆորմացիա ներկայացնելու համար պիտանի Հորնի դիզյունկտներից: Բացասական ինֆորմացիա ներկայացնելու համար անհրաժեշտ է ընդլայնել ծրագրի սահմանումը՝ թույլատրելով ժխտման օգտագործում ծրագրի կանոնների մարմիններում: Պարզ է, որ տրամաբանական ծրագրավորման համակարգերն այդ դեպքում պետք է օժտված լինեն բացասական ինֆորմացիան մշակելու մեխանիզմներով: Նման դեպքերում սովորաբար կիրառվում է *ժխտումը որպես ձախողում* (*Negation as Failure*) կանոնով ընդլայնված *SLD*-ռեզոլյուցիան՝ *SLDNF*-ռեզոլյուցիան: Հիմնավորելու համար *ժխտումը որպես ձախողում* կանոնի օգտագործումը Կ.Քլարկի կողմից առաջարկվեց ծրագրի «փակման» հասկացությունը, որի առանցքային գաղափարը ծրագրում օգտագործվող բոլոր պրեդիկատներն ամբողջությամբ սահմանված համարելն է: Այսինքն, *P* ընդհանրացված ծրագրին համապատասխանեցվում է առաջին կարգի պրեդիկատների տրամաբանության *comp(P)* բանաձև (որն օգտագործում է հավասարություն)՝ համաձայն փակ աշխարհի ենթադրության: Ապացուցված է *SLDNF*-ռեզոլյուցիայի անհակասելիությունն այն իմաստով, որ եթե *P* ընդհանրացված ծրագրի և *Q* ընդհանրացված հարցման համար *(P,Q)* զույգից *SLDNF*-ռեզոլյուցիայի միջոցով արտածվում է դատարկ հարցում, ապա *comp(P)* բանաձևից տրամաբանորեն բխում է *Q* հարցումը, իսկ եթե *(P,Q)* զույգի համար կառուցվում է ձախողման վերջավոր *SLDNF*-ծառ, ապա *comp(P)* բանաձևից տրամաբանորեն բխում է *Q* հարցման ժխտումը: Հայտնի է, որ ընդհանուր դեպքում *SLDNF*-ռեզոլյուցիան լրիվ չէ, այսինքն հնարավոր է, որ *comp(P)* բանաձևից տրամաբանորեն բխի հարցումը կամ նրա ժխտումը, սակայն *SLDNF*-ռեզոլյուցիայի միջոցով պատասխան ստանալ հնարավոր չլինի: Ուստի հետաքրքրություն է ներկայացնում *SLDNF*-ռեզոլյուցիայի լրիվության ուսումնասիրությունը ծրագրերի և հարցումների ենթադասերում:

Տրամաբանական ծրագրավորման իրական համակարգերն օգտագործում են ներդրված պրեդիկատներ, որոնց իրականացումը տարբերվում է այն իրականացումից, որը կհամապատասխաներ Հորնի դիզյունկտներով նրանց ներկայացմանը: Բացի այդ, ոչ բոլոր ներդրված պրեդիկատներն են ներկայացվում Հորնի մաքուր ծրագրերի միջոցով: Ս.Նիգիյանի կողմից մոդիֆիկացվել է *SLD*-ռեզոլյուցիան ներդրված պրեդիկատներով Հորնի ծրագրավորման համար և ապացուցվել, որ այն լրիվ է ու անհակասելի: Սակայն ներդրված պրեդիկատներ օգտագործող տրամաբանական ծրագրավորման իրական համակարգերը կարող են օգտագործել նաև ժխտում, և առաջանում է *SLDNF*-ռեզոլյուցիայի մոդիֆիկացիայի խնդիրն այս դեպքի համար:

Սույն ատենախոսության նպատակն է ուսումնասիրել *SLDNF*-ռեզոլյուցիայի լրիվությունը ընդհանրացված մաքուր տրամաբանական ծրագրերի և հարցումների ենթադասերում, մոդիֆիկացնել *SLDNF*-ռեզոլյուցիան ներդրված պրեդիկատների օգտագործման դեպքի համար և ուսումնասիրել մոդիֆիկացված *SLDNF*-ռեզոլյուցիայի անհակասելիությունը, ինչպես նաև դիտարկել պրակտիկ *SLDNF*-ռեզոլյուցիայի կանոններն (այդ կանոնների որոշման տիրույթն ավելի նեղ է *SLDNF*-ռեզոլյուցիայի համապատասխան կանոնների որոշման տիրույթից) ու ուսումնասիրել նրանց անհակասելիությունը:

Աշխատանքում կատարված հետազոտությունների արդյունքում ստացվել են հետևյալ հիմնական արդյունքները.

1. Հետազոտվել է *SLDNF*-ռեզոլյուցիայի լրիվությունը ընդհանրացված մաքուր ծրագրերի և հարցումների ենթադասերում:
   - Սահմանվել է դաս, որում *SLDNF*-ռեզոլյուցիան լրիվ չէ, սակայն երբ հարցումը տրամաբանորեն բխում է ծրագրին համապատասխանող բանաձևից, *SLDNF*-ռեզոլյուցիան տալիս է դրական պատասխան:
   - Սահմանվել է դաս, որում *SLDNF*-ռեզոլյուցիան լրիվ չէ, սակայն երբ հարցման ժխտումն է տրամաբանորեն բխում ծրագրին համապատասխանող բանաձևից, *SLDNF*-ռեզոլյուցիան տալիս է բացասական պատասխան:
   - Սահմանվել է ընդհանրացված ծրագրերի և հարցումների դաս, որում *SLDNF*-ռեզոլյուցիան լրիվ է:

   Նշված դասերը նկարագրված են ծրագրերի և հարցումների հատկությունների համակարգերի միջոցով (հատկությունների համակարգին համապատասխանեցվում է այն ծրագրերի ու հարցումների բազմությունը, որոնք միաժամանակ օժտված են համակարգի բոլոր հատկություններով) և ցույց է տրված, որ այդ հատկությունների համակարգերը չկրճատվող են այն իմաստով, որ որևէ հատկություն համակարգից հեռացնելու դեպքում ստացված համակարգով որոշվող ծրագրերի և հարցումների դասում *SLDNF*-ռեզոլյուցիայի առանձնահատկությունները չեն պահպանվում:

2. Մոդիֆիկացվել է *SLDNF*-ռեզոլյուցիան ներդրված պրեդիկատների օգտագործման դեպքի համար և ապացուցվել մոդիֆիկացված *SLDNF*-ռեզոլյուցիայի անհակասելիությունը:

3. Ներմուծվել են *SLDNF*-ռեզոլյուցիայի պրակտիկ կանոններ և ապացուցվել է պրակտիկ *SLDNF*-ռեզոլյուցիայի անհակասելիությունը:

ABSTRACT

Lusine A. Sargsyan

"On *SLDNF*-resolution in logic programming with negation"

The thesis is devoted to problems of logic programming. General logic programs (logic programs with negation) and general goals (goals with negation) are investigated.

A logic program in Horn programming consists of Horn disjuncts, which are useful only to represent positive information. In order to represent negative information, it is important to extend the definition of programs to include negative literals in the bodies of clauses. Logic programming systems also require a mechanism for handling negative information. The method usually chosen is *SLDNF*-resolution, which is the extension of *SLD*-resolution with the negation as failure rule. In order to justify the use of the negation as failure rule Clark introduced the concept of the completion of a general program, the key idea of which is to treat all predicates of a program as fully defined. More precisely a formula *comp(P)* of the first-order logic (with equality) is associated with a program *P* in accordance with the closed world assumption. It is known that *SLDNF*-resolution is sound in the sense that if for a general program *P* and a general goal *Q* there exists a derivation of the empty goal from *(P,Q)*, then *Q* is a logical consequence of *comp(P),* and if there exists a finitely failed *SLDNF*-tree for *(P,Q)*, then the negation of *Q* is a logical consequence of *comp(P).* It is also known that in general *SLDNF*-resolution is not complete, i.e. a goal *Q* or its negation can be a logical consequence of *comp(P)*, but it is not possible to get a response for *(P,Q)* via *SLDNF*-resolution. Naturally the question arises about the completeness of *SLDNF*-resolution in some subclasses of the programs and goals.

Practical logic programming systems use built-in predicates, that are implemented differently to those defined by Horn disjuncts. Moreover, some built-in predicates cannot be represented by pure Horn disjuncts. The modification of *SLD*-resolution for Horn programming with built-in predicates is introduced by S.Nigiyan and its completeness and soundness has been established. However, practical logic programming systems use both negation and built-in predicates, and a modification of *SLDNF*-resolution for built-in predicates is required.

The objectives of this thesis are to investigate the completeness of *SLDNF*-resolution in some subclasses of general programs and goals, to introduce a modification of *SLDNF*-resolution for built-in predicates and investigate the soundness of the modified *SLDNF*-resolution, as well as to study *SLDNF*-resolution rules used in practical systems (the domain of definition of such rules is a subset of the domain of definition of *SLDNF*-resolution rules), and investigate their soundness.

The research carried out in the thesis has produced the following main results.

1. The completeness of *SLDNF*-resolution is investigated in some subclasses of pure general programs and goals.

- A subclass of general logic programs and goals is defined, in which *SLDNF*-resolution is not complete, but if a goal is a logical consequence of the formula corresponding to the program, then *SLDNF*-resolution gives a positive answer.
- A subclass of general logic programs and goals is defined, in which *SLDNF*-resolution is not complete, but if the negation of a goal is a logical consequence of the formula corresponding to the program, then *SLDNF*-resolution gives a negative answer.
- A subclass of general logic programs and goals is defined, in which *SLDNF*-resolution is complete.

These classes are described by means of systems of program and goal properties (a system of properties defines the set of all programs and goals, which possess all the properties of the system simultaneously). It is shown that these systems are irreducible in the sense that if any of system properties is omitted, *SLDNF*-resolution loses the above features.

2. A modification of *SLDNF*-resolution for built-in predicates is introduced and the soundness of the modified *SLDNF*-resolution is proved.

3. *SLDNF*-resolution rules used in practical systems are introduced and their soundness is proved.