Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»
Кафедра информатики

Филипповский Виталий Александрович

Система автоматического поиска доказательств теорем, основанная на специальном варианте обратного метода С.Ю. Маслова

Магистерская диссертация

Допущена к защите.

Зав. кафедрой:

д. ф.-м. н., профессор Косовский Н. К.

_____

подпись

Научный руководитель:

д. ф.-м. н., профессор Оревков В. П.

_____

подпись

Рецензент:

ст. преп., Петухова Н. Д.

_____

подпись

Санкт-Петербург

2015

SAINT-PETERSBURG STATE UNIVERSITY

Computer Science Chair

Filippovskii Vitaly

System of automatic theorem proof search based on special type of Maslov's inverse method

Master's Thesis

Admitted for defence.

Head of the chair:

professor Kossovski N. K.

_____

signature

Scientific supervisor:

professor Orevkov V. P.

_____

signature

Reviewer:

Petuchova N. D.

_____

signature

Saint-Petersburg

2015

Оглавление

Введение

Задача автоматизации поиска доказательств в логических и логико-математических исчислениях является особым направлением исследований в логике и теории искусственного интеллекта. В настоящей работе рассматривается задача поиска вывода в классическом исчислении предикатов первого порядка без равенства и без функциональных знаков. Большинство предлагавшихся алгоритмов поиска вывода для исчисления предикатов основано на известной теореме Ж. Эрбрана (1930) [18], [25], в которой установлены необходимые и достаточные условия выводимости формулы. Применение напрямую теоремы Эрбрана приводит к малоэффективным системам автоматического доказательства теорем. Основываясь на теореме Эрбрана, в 1965 г. Д.А. Робинсон предложил правило резолюции и основанный на нём метод резолюций, который получил в дальнейшем широкое признание и развитие. Метод резолюций был положен в основу логического программирования (язык логического программирования Prolog и др.), на него возлагались большие надежды в решении логическими методами классических проблем теории искусственного интеллекта.

Практически одновременно (1964 г.) с методом резолюций и независимо от него С.Ю. Масловым был предложен так называемый *обратный метод* установления выводимости [13]. Этот метод был разработан и реализован в Ленинградском отделении математической логики Математического института им. В.А. Стеклова (ЛОМИ). Как отмечал Маслов, обратный метод не уступает по эффективности методу резолюции [16]. Но несмотря на это обратный метод не нашёл такого широкого применения, как метод резолюций, поэтому он до сих пор остаётся недостаточно изученным.

В первоначальном варианте обратный метод, как и метод резолюций, был ориентирован на поиск вывода формул определённого вида в классическом исчислении предикатов с функциональными знаками (без равенства). Доказываемые формулы должны были быть приведены к форме дизъюнкции предварённых конъюнктивных нормальных форм. Позже выяснилось, что ограничение на вид выводимых формул не является

для обратного метода существенным, что метод применим для доказательства произвольных формул классического исчисления предикатов, а кроме того, формул других исчислений (исчисления предикатов с равенством, конструктивного исчисления предикатов). Важным свойством обратного метода является то, что он может выступать удобным аппаратом теории выводимости: применение обратного метода к теории разрешимых фрагментов классического исчисления предикатов дало возможность охватить единой схемой доказательства разрешимости почти всех имеющихся в литературе разрешимых классов формул классического исчисления предикатов, расширить известные и получить новые разрешимые классы, а также получить новые классы сведения.

В современной логике есть большое количество алгоритмов поиска вывода в логических исчислениях. Эти алгоритмы сильно различаются между собой, например, могут быть приспособлены только для некоторых (не для всех) разрешимых классов формул, либо предназначены для конкретного типа исчислений. Обратный метод позволяет построить практически эффективный алгоритм поиска логического вывода, являющегося одновременно разрешающим алгоритмом для широких классов формул классического исчисления предикатов и позволяет получать конкретизации для большого числа логических исчислений.

Предметом настоящего исследования является обратный метод поиска доказательств теорем.

Целью исследования является построение программной реализации одного из вариантов обратного метода и сравнение его по эффективности с методом резолюций.

Актуальность исследования обусловлена следующими обстоятельствами.

В последние 10-15 лет обнаруживается возрождение интереса к обратному методу. Так, например, в работе [10] была разработана конкретизация обратного метода для автоэпистемической логики, которая применялась при построении экспертных систем. Совсем недавно была построена реализация обратного метода для модальной логики KT [1]. В течение нескольких последних лет появился ряд работ Т.М. Косовской и Н.Д. Петуховой, в которых обратный метод нашёл применение к решению задач логико-предметного распознавания образов и прочих проблем искусственного интеллекта, в частности, моделирования муравьиного алгоритма [24], [27], [28]. Поскольку обратный метод выступает удобным аппаратом теории выводимости, он часто находит применение в доказательствах теоретических результатов в теории разрешимых классов. Так, в 2004 г. В.П. Оревковым при помощи обратного метода была доказана рарешимость нового хорновского фрагмента исчисления предикатов [19], а в 2010 г. Г.Е. Минцем при

помощи обратного метода была доказана разрешимость класса E [30].

При всём разнообразии интересов к обратному методу, автору неизвестны его реализации на высокоуровневом языке (в открытом доступе программных реализаций нет). Поэтому актуальной становится задача построения программной реализации обратного метода, по крайней мере, одной из его конкретизаций.

Ещё одним актуальным аспектом настоящего исследования является задача сравнения обратного метода с методом резолюции. С.Ю. Маслов отмечал [16], что обратный метод не уступает по эффективности методу резолюции. До сих пор не было проведено тщательного сравнения эффективностей этих двух методов.

Задачами настоящего исследования являются:

1) провести анализ и реконструировать общую схему обратного метода;

2) сравнить разные конкретные варианты обратного метода;

3) выбрать один вариант обратного метода для программной реализации;

4) выбрать класс доказуемых формул;

5) построить программную реализацию выбранного варианта обратного метода;

6) сравнить по эффективности работу алгоритма, построенного на основе обратного метода, с работой алгоритма на базе метода резолюций для выбранного класса формул.

Работа состоит из Введения, 4 глав, заключения и списка использованной литературы. В первой главе «Общая схема обратного метода» рассматривается общая схема обратного метода, вводятся основные для этого метода понятия: «набор», «система зависимостей», «разбивка» и др. Во второй главе «Конкретизации общей схемы обратного метода» проанализированы некоторые конкретные варианты обратного метода, в частности, подробно рассмотрен выбранный для реализации специальный вариант обратного метода. Для специального варианта обратного метода предложена новая нотация и приведены несколько примеров выводов. В третьей главе «Программная реализация алгоритма поиска вывода на основе обратного метода» рассмотрена схема алгоритма АУВ, разработанного группой математической логики ЛОМИ в 60-х гг. 20 в. и предложена блок-схема реализации алгоритма, принятая в настоящем исследовании. В четвёртой главе «Сравнение обратного метода с методом резолюции» были зафиксированы условия проведения экспериментов по сравнению эффективности обратного метода и метода резолюции, в частности, было сформировано тестовое множество формул, которые были получены путём сведения задачи о распознавании клик в неориентированных графах к проблеме выводимости формул необходимого вида.

Глава 1.

Общая схема обратного метода

1.1. Проблема поиска вывода в логических исчислениях

Задача установления выводимости формулы в некотором логическом исчислении ставится обычно следующим образом. Пусть задан некоторый логический язык $\mathcal{L}$, на котором сформулировано некоторое логическое исчисление $\mathbf{G}$ и пусть дана формула F в языке $\mathcal{L}$. Задача установления выводимости формулы F в логическом исчислении $\mathbf{G}$ сводится к исследованию того, можно ли построить вывод формулы F в исчислении $\mathbf{G}$. Если удаётся построить такой вывод, то говорят, что формула F *выводима* в исчислении $\mathbf{G}$:

$$\mathbf{G} \vdash F, \tag{1.1}$$

если такой вывод построить не удаётся, то говорят, что формула F *невыводима* в исчислении $\mathbf{G}$:

$$\mathbf{G} \nvdash F. \tag{1.2}$$

Под алгоритмом *поиска вывода* в исчислении $\mathbf{G}$ понимается алгоритм, который для всех выводимых формул кончает работу и возвращает вывод формулы в исчислении $\mathbf{G}$.

Под *разрешающим алгоритмом* для данного класса формул понимается алгоритм, который для каждой формулы указанного класса кончает работу и возвращает ответ «ДА», если формула выводима в исчислении $\mathbf{G}$, и ответ «НЕТ», если формула невыводима.

В теории доказательств современной логики существует большое количество ме-

тодов поиска вывода. Им соответствуют различные исчисления: аксиоматическое, натуральное, секвенциальное, табличное, резолютивное, и проч. Обратный метод, с одной стороны, по эффективности нисколько не уступает остальным методам. С другой стороны, он не нашёл такого широкого применения, как, например, метод резолюций. Поэтому обратный метод до сих пор остаётся недостаточно изученным.

1.2. Основная идея обратного метода

Ключевой идеей обратного метода является создание специальной записи структуры F-секвенции, своего рода протокола анализа структуры F-секвенции. Эта запись носит название набора. При построении наборов по секвенциям поиск вывода переходит из «плоскости» секвенциального исчисления в «плоскость» специального исчисления наборов. Переход в «плоскость» наборов осуществляется следующим образом: сначала по F-секвенции строятся исходные наборы, которые представляют собой метаописания возможных аксиом, которые могут находиться в листьях дерева поиска вывода в секвенциальном исчислении для рассматриваемой секвенции, а затем по правилам исчисления наборов продуцируются всё новые и новые наборы, пока не будет выведен так называемый пустой набор. Если пустой набор получен, то доказываемая F-секвенция доказуема в секвенциальном исчислении.

Пусть задан некоторый логический язык $\mathcal{L}$, на котором сформулировано некоторое секвенциальное исчисление $\mathbf{G}$, и пусть дана секвенция Σ в языке $\mathcal{L}$, для которой требуется проверить её выводимость в исчислении $\mathbf{G}$. Обратный метод позволяет по произвольному секвенциальному исчислению $\mathbf{G}$ и произвольной секвенции Σ в языке $\mathcal{L}$, на котором сформулировано исчисление $\mathbf{G}$, построить специальное исчисление $\mathbf{G}^*$ *благоприятных* Σ*-наборов*.

Рис. 1.1: Основная идея обратного метода

Конкретные варианты обратного метода формулируются таким образом, чтобы исчисление $\mathbf{G}^*$ отвечало требованиям *корректности* и *полноты* относительно исходного

исчисления $\mathbf{G}$: секвенция Σ выводима в исчислении $\mathbf{G}$ т. т. т., когда в соответствующем ей исчислении $\mathbf{G}^*$ благоприятных Σ-наборов выводим пустой набор $\Box$:

$$\mathbf{G} \vdash \Sigma \Leftrightarrow \mathbf{G}^* \vdash \Box. \tag{1.3}$$

Таким образом, проблема выводимости секвенции Σ в исчислении $\mathbf{G}$ сводится к проблеме выводимости пустого набора $\Box$ в исчислении $\mathbf{G}^*$ благоприятных Σ-наборов[1].

Для того чтобы задать общую схему обратного метода, нужно определить, как по исходному секвенциальному исчислению $\mathbf{G}$ и исходной секвенции Σ строить исходные благоприятные Σ-наборы и как получать новые Σ-наборы из уже имеющихся. Иначе говоря, чтобы задать общую схему обратного метода нужно определить, как задаются аксиомы и правила вывода исчисления $\mathbf{G}^*$ благоприятных Σ-наборов.

Σ-наборы интерпретируются как некоторые списки формул, входящих в ветви дерева поиска вывода секвенции Σ вместе с *зависимостями*, связывающими формулы этих списков[2]. В эти списки попадают не все формулы, входящие в Σ, и не все зависимости между ними. Выбор интересующих формул и зависимостей осуществляется благодаря специально введённым понятиям *разбивки* и *системы зависимостей*. Разбивка секвенции выделяет все необходимые для доказательства подформулы, входящие в Σ, система зависимостей даёт нам язык для выражения именно тех зависимостей между формулами, которые существенны для установления выводимости. Прежде чем определять понятие Σ-набора необходимо определить понятия разбивки и системы зависимостей.

1.3. Понятие разбивки

Определение 1. Будем говорить, что *формула входит в секвенцию* $F_1, ..., F_l \to F_{l+1}, ..., F_{l+m}$, если формула является одной из формул F_i $(1 \leqslant i \leqslant l+m)$.

[1] Аналогичен принцип построения метода резолюций: проблема выводимости формулы F в классическом исчислении предикатов с функциональными знаками сводится к проблеме выводимости пустого дизъюнкта в соответствующем формуле F исчислении резольвент. Однако заметим, что роль пустого набора в обратном методе дуальна роли пустого дизъюнкта в методе резолюций. Выводимость пустого дизъюнкта в исчислении резольвент, соответствующем формуле F означает невыполнимость этой формулы. Поэтому для доказательства данной формулы методом резолюций пытаются вывести пустой дизъюнкт для её отрицания. По-иному обстоит дело в обратном методе. Выводимость пустого набора в исчислении благоприяных наборов данной формулы означает неопровержимость данной формулы.

[2] Существует другая интерпретация понятия набора, предложенная В. П. Оревковым, согласно которой набору содержательно соответствует не вхождение соответствующей последовательности реализаций в ветвь, а совпадение этой последовательности с узлом дерева поиска вывода.

Определение 2. Будем говорить, что *формула входит в формулу* F_i, если она является подформулой формулы F_i $(1 \leqslant i \leqslant l+m)$.

Определение 3. Определим индуктивно понятие *положительного* (*отрицательного*) *вхождения* формулы в *формулу*.

1) F положительно входит в F.

2) Если G положительно (отрицательно) входит в F, то G положительно (отрицательно) входит в формулы вида:

$$\begin{array}{lll} F\&F', & F'\&F, & \forall xF, \\ F\vee F', & F'\vee F, & \exists xF, \end{array} \qquad F'\supset F,$$

и отрицательно (положительно) входит в формулы вида:

$$\neg F \quad F\supset F'.$$

Определение 4. Определим понятие *положительного* (*отрицательного*) *вхождения* формулы в *секвенцию*.

1) Если формула G положительно (отрицательно) входит в формулу F антецедента, то формула G имеет отрицательное (положительное) вхождение в секвенцию.

2) Если формула G положительно (отрицательно) входит в формулу F сукцедента, то формула G имеет положительное (отрицательное) вхождение в секвенцию.

Определение 5. *Разбивкой* секвенции Σ будем называть любой список формул

$$A_1, A_2, ..., A_\delta, \tag{1.4}$$

такой, что при каждом i $(1 \leqslant i \leqslant \delta)$, если A_i не начинается с отрицания, то найдётся положительное вхождение A_i в Σ, а если A_i имеет вид $\neg A_i'$, то найдётся отрицательное вхождение A_i' в Σ.

Для каждой формулы из (1.4) её свободные переменные, отличные от свободных переменных секвенции Σ, будем называть *основными переменными* этой формулы. Результат подстановки вместо основных переменных некоторой формулы разбивки каких-нибудь термов исчисления **G** будем называть *реализацией* этой формулы.

Для классического исчисления предикатов можно упростить понятие разбивки. Без потери общности для классического исчисления предикатов можно рассматривать проблему установления выводимости лишь для секвенций вида

$$\to F, \tag{1.5}$$

где: 1) F содержит среди пропозициональных связок только $\&$, $\vee$, $\neg$; 2) знак $\neg$ входит в F только в составе элементарных формул[3]; 3) связанные переменные формулы F отличны от свободных, и собственные переменные различных вхождений кванторных комплексов из F различны.

Определение разбивки для секвенций вида (1.5) упрощается следующим образом:

Определение 6. *Разбивкой* секвенции (1.5) называется всякий список (1.4), состоящий из формул, имеющих положительное вхождение в секвенцию (1.5).

Замечание. Наиболее надёжным с точки зрения полноты метода[4] способом построения разбивки секвенции Σ является включение в разбивку всех подформул, положительно входящих в Σ, и отрицаний всех формул, входящих в Σ отрицательно. Однако такой способ практически невыгоден ввиду необозримости получаемых разбивок. При конкретизациях обратного метода следует решить вопрос о построении минимальных необходимых для полноты метода разбивок (т. е. разбивок, из которых нельзя выбросить ни одной формулы без нарушения полноты метода) и о целесообразности расширения минимальных разбивок за счёт формул, которые способствуют установлению выводимости, но не являются необходимыми.

1.4. Понятие системы зависимостей

Пусть мы имеем дерево поиска вывода секвенции Σ в некотором секвенциальном исчислении $\mathbf{G}$, обладающем свойством подформульности. Рассмотрим некоторую ветвь дерева поиска вывода секвенции Σ «снизу вверх» (на рисунке ветвь выделена жирным шрифтом).

$$\dfrac{\dfrac{\dfrac{\vdots}{H'_{k+2}} \quad \dfrac{\dfrac{\vdots}{\boldsymbol{H}_{k+3}}}{\boldsymbol{H}_{k+2}}}{\boldsymbol{H}_{k+1}} \quad \dfrac{\dfrac{\dfrac{\vdots}{H''_{k+3}}}{H''_{k+2}} \quad \dfrac{\vdots}{H'''_{k+2}}}{H''_{k+1}}}{\dfrac{\dfrac{\boldsymbol{H}_k}{\vdots}}{\dfrac{\boldsymbol{H}_1}{\Sigma}}}$$

[3]Будем говорить, что формула является *атомарной*, если она не содержит логических знаков, и что формула является *элементарной*, если она является атомарной или отрицанием атомарной.

[4]Под *полнотой* конкретизации обратного метода понимается прямое следствие основного свойства обратного метода (1.3), а именно: то, что из выводимости секвенции Σ в исчислении $\mathbf{G}$ следует благоприятность пустого набора в специальном исчислении $\mathbf{G}^*$ благоприятных Σ-наборов: $\mathbf{G} \vdash \Sigma \Rightarrow \mathbf{G}^* \vdash \Box$.

В различных местах этой ветви возникают те или иные реализации H_i $(i = 1, 2, \ldots)$ тех или иных формул разбивки. Эти реализации связаны между собой некоторыми *зависимостями*.

Пусть f – некоторый двухместный функциональный знак, входящий в язык исчисления $\mathbf{G}$, a – некоторая свободная переменная секвенции Σ. Пусть H_i – реализация некоторой формулы разбивки (1.4) (для любого $i = 1, 2, \ldots$). В связи с задачей установления выводимости полезны, например, следующие зависимости:

(а) при получении H_i $(i = 1, 2, \ldots)$ вместо n-ой основной переменной формулы A_j $(1 \leqslant j \leqslant \delta)$ подставлен терм вида $f(a, t)$ (t – произвольный терм в языке исчисления $\mathbf{G}$);

(б) при получении H_1 вместо n-ой основной переменной формулы A_j $(1 \leqslant j \leqslant \delta)$ подставлен тот же терм, что и подставленный при получении H_2 вместо m-ой основной переменной формулы $A_{j'}$ $(1 \leqslant j' \leqslant \delta)$, и т. д.

Формализация подобных зависимостей может производиться по-разному, поэтому в описании общей схемы обратного метода язык для выражения зависимостей фиксироваться не будет[5]. Формализация в некотором языке подобных зависимостей приводит к понятиям *системы зависимостей* и *допустимой системы зависимостей*.

1.5. Понятие Σ-набора

Пусть заданы: 1) исчисление $\mathbf{G}$, 2) секвенция Σ, 3) алгоритм построения разбивки секвенции Σ и 4) язык для записи систем зависимостей. Определим понятие *набора*.

Определение 7. *Набором* называется любая пара вида $[C; S]$, где C – список номеров формул разбивки (1.4), т. е. список натуральных чисел, не превосходящих δ, S – допустимая система зависимостей. *Длиной набора* называется длина списка C, *элементами* (или *компонентами*) набора называются элементы списка C.

Определение 8. *Свободным набором* называется набор, у которого список S пуст, т. е. набор вида $[C;\ ]$.

Определение 9. *Пустым набором* (специальное обозначение $\square$) называется набор, у которого списки C и S пусты, т. е. набор вида $[\ ;\ ]$.

[5]Выбор языка для выражения систем зависимостей определяется теми зависимостями, которые полезны для установления выводимости, поэтому он зависит от исчисления, в котором производится поиск вывода, и до некоторой степени – от поставленной перед алгоритмом поиска вывода задачи. Поэтому точные определения понятий системы зависимостей и допустимой системы зависимостей вводятся только при построении конкретизаций общего метода.

1.6. Понятие благоприятного Σ-набора

Понятие благоприятного набора является основным понятием обратного метода. Определение благоприятного набора является индуктивным, базой которого является правило получения исходных благоприятных наборов (*правило* **А**), а индукционным переходом – правило построения по некоторой совокупности благоприятных наборов нового благоприятного набора (*правило* **Б**).

Правила **А** и **Б** задают *исчисление благоприятных наборов*. Выводимые объекты этого исчисления – благоприятные Σ-наборы – являются собственным подмножеством множества Σ-наборов, в терминах которых формулируется это исчисление. Выводимость в исчислении благоприятных наборов пустого набора влечёт выводимость исходной секвенции в исходном секвенциальном исчислении.

Уточним способ построения правил **А** и **Б**. Естественным требованием к этим правилам будет разрешимость следующих отношений:

(1) для правила **А**: «набор **Н** является исходным благоприятным набором»;

(2) для правила **Б**: «набор **Н** получен по правилу **Б** из наборов $\mathbf{H}_1$, $\mathbf{H}_2$, ..., $\mathbf{H}_k$».

Реализовать требование (1) удобно следующим образом: выбирается разрешимый фрагмент $\mathbf{G}'$ исчисления **G**, сформулированный в том же языке, что и исчисление **G** (для исчисления предикатов в качестве такого фрагмента удобно выбирать исчисление высказываний). После того, как фрагмент зафиксирован, правило **А** уточняется следующим образом:

Правило **А**. Набор **Н** является исходным благоприятным набором, если верхняя секвенция любой ветви дерева поиска вывода, в которой правильно встречается **Н**, выводима в $\mathbf{G}'$.

Правило **Б** уточняется следующим образом:

Правило **Б**. Пусть $\mathbf{H}_1$, $\mathbf{H}_2$, ..., $\mathbf{H}_k$ – благоприятные наборы и набор **Н** обладает следующим свойством: можно указать полную совокупность $\mathbf{H'}_1$, $\mathbf{H'}_2$, ..., $\mathbf{H'}_k$ продолжений набора **Н**, такую, что для всех i $\mathbf{H}_i$ является поднабором набора $\mathbf{H'}_i$. Тогда набор **Н** считается результатом применения правила **Б** к наборам $\mathbf{H}_1$, $\mathbf{H}_2$, ..., $\mathbf{H}_k$.

Глава 2.

Конкретизации общей схемы обратного метода

2.1. Логический язык и исходное секвенциальное исчисление

Общая схема обратного метода, описанная в предыдущей главе, даёт только общее представление о том, как реализуется алгоритм поиска вывода обртаным методом. Пусть даны исчиление **G**, секвенция Σ в языке этого исчисления, некоторая разбивка секвенции Σ, некоторое точное определение набора и некоторые точно сформулированные правила **А** и **Б**. Тогда секвенция Σ выводима в исчислении **G** т. и т.т., когда в этом исчислении благоприятных наборов выводим пустой набор. Таким образом, для конкретизации обратного метода для выбранного исчисления **G** требуется:

(1) задать алгоритм построения по исходной секвенции соответствующей ей разбивки;

(2) зафиксировать язык описания системы зависимостей между реализациями формул разбивки данной секвенции;

(3) сформулировать правила **А** и **Б**, пригодные для построения благоприятных наборов по исходной секвенции заданного типа.

Поскольку для классического исчисления предикатов проблему установления выводимости произвольной секвенции можно свести к проблеме выводимости секвенций вида

$$\rightarrow F, \tag{2.1}$$

в приводимых в этой главе конкретизациях обратного метода принимается допущение,

что исходная секвенция приведена к виду (2.1), а за исходное исчисление принимается некоторое безантецедентное секвенциальное исчисление, обладающее свойством подформульности.

С целью упрощения будущего изложения, зафиксируем для всех конкретизаций один логический язык и одно исходное секвенциальное исчисление.

2.1.1. Логический язык $\mathcal{L}_0$

Логические знаки

Индивидные переменные: $u, v, w, x, y, z, u_1, v_2, w_3, x_4, y_5, z_6, \ldots$.

Функциональные переменные: $f, g, h, f_1, g_2, h_3, \ldots$.

Предикатные переменные: $P, Q, R, P_1, Q_2, R_3, \ldots$.

Пропозициональные связки: $\&$, $\vee$, $\neg$.

Кванторы: $\forall$, $\exists$.

Нелогические (дескриптивные) знаки

Индивидные константы: $a, b, c, a_1, b_2, c_3, \ldots$.

Функциональные константы: —.

Предикатные константы: —.

Вспомогательные знаки: (,), ', $\to$.

2.1.2. Исходное секвенциальное исчисление G_0

Пусть D и D' – простые дизъюнкции[1], Γ, Γ' и Γ'' – списки формул (возможно, пустые). M – бескванторная формула, $[M]^{x_1,\ldots,x_m}_{b_1,\ldots,b_m}$ – результат одновременной подстановки в M термов $b_1, \ldots, b_m$ вместо переменных $x_1, \ldots, x_m$ соответственно.

Схемы аксиом

1) $\to \Gamma, D, \Gamma'$, где D замкнута[2];

2) $\to \Gamma, D, \Gamma', D', \Gamma''$, где $D \vee D'$ замкнута.

Правила вывода

$$\frac{\to \Gamma, B^1, \Gamma' \,; \ldots \,; \to \Gamma, B^s, \Gamma'}{\to \Gamma, B^1 \& \ldots \& B^s, \Gamma'} \,(\to \&)\,,$$

[1] Под *простой* дизъюнкцией понимается дизъюнкция элементарных формул, т. е. атомарных формул и их отрицаний.

[2] Дизъюнкция называется *замкнутой*, если она содержит формулу вместе с её отрицанием.

где $B^1, \ldots, B^s$ не имеют главным знаком знак &;

$$\frac{\to \Gamma, B^1, \ldots, B^s, \Gamma'}{\to \Gamma, B^1 \vee \ldots \vee B^s, \Gamma'} (\to \vee),$$

где $B^1, \ldots, B^s$ не являются элементарными формулами;

$$\frac{\to \Gamma, [M]^{x_1,\ldots,x_m}_{b_1,\ldots,b_m}, \Gamma'}{\to \Gamma, \forall x_1, \ldots, x_m M, \Gamma'} (\to \forall),$$

где $b_1, \ldots, b_m$ – термы, не входящие в заключение;

$$\frac{\to \Gamma, [M]^{x_1,\ldots,x_m}_{t_1,\ldots,t_m}, \exists x_1, \ldots, x_m M, \Gamma'}{\to \Gamma, \exists x_1, \ldots, x_m M, \Gamma'} (\to \exists),$$

где $t_1, \ldots, t_m$ – термы, не содержащие связанных переменных заключения.

* * *

Таким образом, во всех конкретизациях обратного метода используется один логический язык $\mathcal{L}_0$, одно исходное секвенциальное исчисление $\boldsymbol{G}_0$, а также допущение, что тестируемая секвенция сформулирована в данном логическом языке $\mathcal{L}_0$ и приведена к виду (2.1). Теперь мы готовы сформулировать конкретизацию обратного метода для самого простого случая вида секвенций.

2.2. Конкретизация обратного метода С. Ю. Маслова для секвенциального исчисления $\boldsymbol{G}_0$ без функциональных знаков и без равенства

Рассмотрим одну из конкретизаций обратного метода, предложенную её автором, С.Ю. Масловым [14], [12].

Будем рассматривать секвенции вида:

$$\to P(\bigwedge_{i=1}^{\delta} D_i). \tag{2.2}$$

Введём краткое обозначение формулы, стоящей в сукцеденте этой секвенции:

$$F \rightleftharpoons P(\bigwedge_{i=1}^{\delta} D_i). \tag{2.3}$$

(1) Алгоритм построения разбивки

Разбивкой секвенции (2.2) считается список всех простых дизъюнкций D_i, выписанных в порядке их появления в секвенции (2.2):

$$D_1, D_2, \ldots, D_\delta. \tag{2.4}$$

(2) Язык для выражения систем зависимостей

Обозначим через C произвольный список натуральных чисел

$$\alpha_1, \alpha_2, \ldots, \alpha_h, \tag{2.5}$$

где $h \geqslant 0$ и $\alpha_i \leqslant \delta$ для любого i.

Пусть список

$$S_1, S_2, \ldots, S_h \tag{2.6}$$

составлен из α_1-й, α_2-й, ..., α_h-й, формул разбивки (2.4).

Приставки «F, C-» и «F-» в нижеследующих определениях означают, что определяемые понятия релятивизированы относительно формулы F (2.3) и списка C (2.5).

F, C-атомом будем называть любое выражение вида u^k, где u – основная переменная формулы S_k $(1 \leqslant k \leqslant h)$.

Понятие *F-терма* совпадает с понятием терма в первопорядковой логике без функциональных знаков (т. е. F-термом является любая индивидная переменная или константа).

F, C-зависимостью будем называть выражение вида $A \approx T$, где A – F, C-атом, T – F-терм.

F, C-системой зависимостей будем называть такой список F, C-зависимостей, что каждый F, C-атом упоминается в этом списке не более одного раза.

(3) Правила **А** и **Б**

Правило **А**. Любой замкнутый набор считается благоприятным набором.

Представим правило **А** в операторной записи:

$$\frac{\text{Набор } \mathbf{H} \quad \text{– замкнутый набор}}{\text{Набор } \mathbf{H} \quad \text{– благоприятный набор}}$$

Правило **Б**. Пусть $\mathbf{H}_1$, $\mathbf{H}_2$, ..., $\mathbf{H}_k$ – благоприятные наборы и набор $\mathbf{H}$ обладает следующим свойством: можно указать такую полную совокупность $\mathbf{H}'_1$, $\mathbf{H}'_2$, ..., $\mathbf{H}'_k$ элементарных продолжений набора $\mathbf{H}$, что для всех i $(1 \leqslant i \leqslant k)$ $\mathbf{H}_i$ является поднабором с общим концом набора $\mathbf{H}'_i$. Тогда набор $\mathbf{H}$ считается благоприятным набором.

Представим правило **Б** в операторной записи:

$\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_k$	– благоприятные наборы
$\mathbf{H}'_1, \mathbf{H}'_2, \ldots, \mathbf{H}'_k$	– полная совокупность продолжений набора $\mathbf{H}$
$\forall i\ (1 \leqslant i \leqslant k)\ \mathbf{H}_i$	– поднабор набора $\mathbf{H}'_i$
Набор $\mathbf{H}$	– благоприятный набор

2.3. Специальный вариант обратного метода В. П. Оревкова

Сохраним все допущения предыдущего раздела для построения специального варианта обратного метода, предложенного В.П. Оревковым [20], а именно: 1) будем пользоваться логическим языком $\mathcal{L}_0$, 2) в качестве исходного исчисления будем использовать безантецедентное секвенциальное исчисление $\boldsymbol{G}_0$ (без функциональных знаков и без равенства).

Будем рассматривать секвенции следующего вида, в сукцеденте которых стоит замкнутая формула F:

$$\to \exists x_1, \ldots, x_n \bigwedge_{i=1}^{\delta} A_i, \tag{2.7}$$

где A_i – дизъюнкции атомарных формул и их отрицаний. Формулы A_i будем называть *блоками*.

2.3.1. Алгоритм построения разбивки

Определение 1. *Разбивкой* секвенции (2.7) будем называть список всех блоков A_i, выписанных в порядке их появления в секвенции (2.7):

$$A_1, A_2, \ldots, A_\delta. \tag{2.8}$$

2.3.2. Язык для выражения систем зависимостей

Зафиксируем произвольный список натуральных чисел

$$c_1, c_2, \ldots, c_h, \tag{2.9}$$

представляющий собой список номеров формул разбивки (2.8), где $h \geqslant 0$ и $c_i \leqslant \delta$ для любого i $(1 \leqslant i \leqslant h)$.

Пусть $y_1, \ldots, y_m$ – переменные языка $\mathcal{L}_0$, $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ — константы языка $\mathcal{L}_0$ или выражения вида y_i^k, где y_i – основная переменная c_k-той формулы разбивки (2.8) $(1 \leqslant i \leqslant m)$.

Определение 2. *Системой уравнений S в переменных и константах с неизвестными $y_1, \ldots, y_m$ (связкой)* будем называть систему равенств вида:

$$S \rightleftharpoons \begin{cases} \alpha_1 = \beta_1 \\ \alpha_2 = \beta_2 \\ \ldots \\ \alpha_n = \beta_n \end{cases} \tag{2.10}$$

Решением системы уравнений S будем называть всякий набор $\gamma_1, \ldots, \gamma_m$ значений переменных $y_1, \ldots, y_m$, такой, что в результате одновременной замены переменных $y_1, \ldots, y_m$ на их значения в решении $\gamma_1, \ldots, \gamma_m$ равенства в системе S превратятся в графические равенства, т. е. левые и правые части каждого равенства системы (2.10) совпадут.

Замечание 1. Существует эффективный алгоритм определения, есть у данной системы уравнений решение или нет [7]. Для этого нужно разделить множество $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ на классы эквивалентности, используя равенства из системы равенств S. Затем необходимо просмотреть каждый класс эквивалентности и сделать вывод исходя из следующих правил: 1) если в каком-то классе есть 2 и более константы, то у данной системы уравнений решения нет; 2) если для каждого класса верно, что в нём либо вовсе нет констант, либо всего 1 константа, то у данной системы уравнений есть решение. Тогда решение строится следующим образом. Для классов, в которых есть всего 1 константа, нужно приравнять все переменные этой константе. Для классов, в которых нет констант, по умолчанию нужно взять первую переменную и приравнять все остальные переменные этой переменной.

Пример 1. Рассмотрим систему S_1:

$$S_1 \rightleftharpoons \begin{cases} x = y \\ y = b \\ z = w \\ z = a. \end{cases} \tag{2.11}$$

Система S_1 имеет решение, потому что во всех соответствующих ей классах эквивалентностей присутствует не более одной константы: $\{x, y, b\}$, $\{z, w, a\}$.

Пример 2. Рассмотрим систему S_2:

$$S_2 \rightleftharpoons \begin{cases} x = y \\ y = b \\ y = z \\ z = w \\ z = a. \end{cases} \tag{2.12}$$

Система S_2 не имеет решение, потому что ей соответствует единственный класс эквивалентности, в котором есть две константы: $\{x, y, z, w, a, b\}$.

Определение 3. Систему уравнений вида (2.10) будем называть *допустимой*, если существует по меньшей мере одно решение этой системы уравнений.

2.3.3. Правила образования замкнутых наборов

Утверждение 1. Если секвенция вида (2.7) содержит контрарную пару, то для неё можно построить замкнутый набор.

Одночленный замкнутый набор. Если в какой-то блок A_i входит элементарная формула вида $R(\alpha_1, \ldots, \alpha_n)$ и элементарная формула вида $\neg R(\beta_1, \ldots, \beta_n)$, где $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ – константы и переменные, тогда можно задать систему уравнений S:

$$S \rightleftharpoons \begin{cases} \alpha_1 = \beta_1 \\ \alpha_2 = \beta_2 \\ \ldots \\ \alpha_n = \beta_n. \end{cases} \tag{2.13}$$

Тогда выражение $\{i; S\}$ будет являться одночленным замкнутым набором. Это выражение можно записать проще: $\{i[S]\}$.

Двучленный замкнутый набор. Если в какой-то блок A_i входит элементарная формула вида $R(\alpha_1, \ldots, \alpha_n)$, а в какой-то другой блок A_j входит элементарная формула вида вида $\neg R(\beta_1, \ldots, \beta_n)$, где $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ – константы и переменные, тогда можно

задать систему уравнений S:

$$S \rightleftharpoons \begin{cases} \alpha_1 = \beta_1 \\ \alpha_2 = \beta_2 \\ \dots \\ \alpha_n = \beta_n. \end{cases} \tag{2.14}$$

Тогда выражение $\{i, j; S\}$ будет являться двучленным замкнутым набором. Это выражение можно записать проще: $\{i[S(i)], j[S(j)]\}$, где $S(i)$ означает те равенства из системы уравнений S, которые относятся к переменным блока i.

Определение 4. Замкнутый набор будем называть *допустимым*, если система уравнений S допустима.

Пример 3. Пусть дана секвенция Σ:

$$\rightarrow \exists x(R(a, x) \vee \neg R(x, b)). \tag{2.15}$$

Тогда одночленный набор $\{1; x = a, x = b\}$ является недопустимым, а двучленный набор $\{1, 1; x^1 = a, x^2 = b\}$ – допустимым.

После того как определено понятие замкнутого набора и допустимого замкнутого набора, можно сформулировать основные правила вывода исчисления благоприятных наборов.

2.3.4. Правила вывода исчисления благоприятных наборов

Для рассматриваемого варианта обратного метода кроме двух классических правил вывода **А** и **Б** характерно наличие ещё одного правила вывода – правила склейки. Правило **А** задаёт аксиомы исчисления, исходные благоприятные наборы. Правило **Б** позволяет получать новые благоприятные наборы из уже имеющихся. Как будет видно далее (в частности на примере 3, см. раздел 2.4.) необходимость включения правила склейки обусловлена тем, что без этого правила вывода в некоторых случаях не получается получать наборы меньшей местности, чем той, что обладают наборы в посылках применения правила **Б**. Простым примером является ситуация, когда нужно построить вывод из одних только двучленных наборов. Пользуясь только правилом **Б**, мы не сможем получить пустой набор, вместо этого на каждом шагу будем получать только двучленные наборы. В этих случаях применяется правило склейки.

Правило А. Любой допустимый замкнутый набор является благоприятным набором.

Представим правило **А** в операторной записи:

$$\begin{array}{|c|}\hline \text{Набор } \mathbf{H} \quad - \quad \text{допустимый замкнутый набор} \\ \hline \text{Набор } \mathbf{H} \quad - \quad \text{благоприятный набор} \\ \hline \end{array} \tag{2.16}$$

Правило Б. Пусть на предыдущих шагах вывода получены следующие благоприятные наборы (выделенные компоненты наборов называются *отрезаемыми компонентами*):

$$\{\underbrace{i_1^1, \ldots, i_1^{k_1}}_{\alpha_1}, \boxed{1}\,; S_1\},$$
$$\ldots,$$
$$\{\underbrace{i_\delta^1, \ldots, i_\delta^{k_\delta}}_{\alpha_\delta}, \boxed{\delta}\,; S_\delta\}.$$

Для корректного применения правила **Б** необходимо соблюсти условие на переменные отрезаемых частей, а именно: все одноимённые переменные из отрезаемых частей должны иметь одно и то же значение. Соблюдение этого требования достигается за счёт насыщения новой связки системой равенств:

$$S^\circ \rightleftharpoons \begin{cases} x_1^{k_1+1} = x_1^{k_2+1} & = \ldots = x_1^{k_\delta+1}; \\ & \ldots \\ x_n^{k_1+1} = x_n^{k_2+1} & = \ldots = x_n^{k_\delta+1}. \end{cases}$$

Если замкнуть полученную систему равенств $S_1 \cup \ldots \cup S_\delta \cup S^\circ$ по транзитивности и симметричности и исключить из классов эквивалентностей переменные $x_1^{k_1+1}, \ldots, x_1^{k_\delta+1}, \ldots, x_n^{k_1+1}, \ldots, x_n^{k_\delta+1}$, тогда получим новую систему равенств $S^* = [S^\circ \cup S_1 \ldots \cup S_\delta] \backslash \{x_i^{k_j+1}\}_{i=1\,j=1}^{n\quad\delta}$, где квадратные скобки означают операцию замыкания множества по транзитивности и симметричности.

Тогда можно образовать новый замкнутый набор $\{\alpha_1, \ldots, \alpha_\delta; S^*\}$, где $S^* = [S^\circ \cup S_1 \cup \ldots \cup S_\delta] \backslash \{x_i^{k_j+1}\}_{i=1\,j=1}^{n\quad\delta}$.

Определение 5. Пусть $\mathbf{H}_1$, $\mathbf{H}_2$, …, $\mathbf{H}_\delta$ – благоприятные наборы. Будем говорить, что δ-посылочное правило **Б** *применимо* к посылкам $\mathbf{H}_1$, $\mathbf{H}_2$, …, $\mathbf{H}_\delta$, если набор $\mathbf{H}$, получающийся при его применении, является допустимым (и следовательно, благоприятным набором).

Утверждение 2. δ-посылочное правило **Б** применимо к посылкам $\mathbf{H}_1$, $\mathbf{H}_2$, …, $\mathbf{H}_\delta$ т. и т. т., когда связка S^* допустима.

Представим δ-посылочное правило **Б** в операторной записи:

$$\left.\begin{array}{l} \{\alpha_1, \boxed{1}\!\!\!\diagup\ ; S_1\}, \\ \ldots, \\ \{\alpha_\delta, \boxed{\emptyset}\!\!\!\diagup\ ; S_\delta\}. \end{array}\right\} \quad - \quad \text{благоприятные наборы}$$

$$\text{Связка } S^* = [S^\circ \cup S_1 \cup \ldots \cup S_\delta] \backslash \{x_i^{k_j+1}\}_{i=1 \, j=1}^{n \quad \delta} \text{ допустима} \tag{2.17}$$

$$\{\alpha_1, \ldots, \alpha_\delta; S^*\} \quad - \quad \text{благоприятный набор}$$

Замечание 2. Для секвенции (2.7) имеется всего одно правило **Б**. Но для более сложных секвенций может быть несколько правил **Б**. Например, для секвенции вида

$$\to \bigvee_{i=1}^{n} P_i(\bigwedge_{j=1}^{\delta_i} D_{i,j}) \tag{2.18}$$

количество правил **Б** в общем случае равно n.

Для каждого правила **Б** определено количество необходимых для него посылок. В общем случае количества посылок двух различных правил **Б** различны. Для указанного в (2.18) вида секвенции количество посылок правила **Б** для i-того блока совпадает с числом δ_i.

Правило склейки. Пусть дан благоприятный набор:

$$\{i_1, \ldots, i_l; S\}.$$

Выберем в этом наборе два компонента i_g и i_h $(1 \leqslant g, h \leqslant l)$, такие, что верно равенство: $i_g = i_h$.

Произведём насыщение связки S за счёт равенств:

$$S^\square \rightleftharpoons \begin{cases} x_1^g = x_1^h; \\ \quad \ldots \\ x_n^g = x_n^h. \end{cases}$$

Если замкнуть полученную систему равенств $S \cup S^\square$ по транзитивности и симметричности и исключить из классов эквивалентностей переменные $x_1^h, \ldots, x_n^h$, тогда получим новую систему равенств $S^* = [S \cup S^\square] \backslash \{x_i^h\}_{i=1}^n$, где квадратные скобки означают операцию замыкания множества по транзитивности и симметричности.

Тогда можно образовать новый замкнутый набор $\{i_1, \ldots, i_g, \ldots, i_{h-1}, i_{h+1}, \ldots, i_l; S^*\}$, где $S^* = [S \cup S^\square] \backslash \{x_i^h\}_{i=1}^n$.

Определение 6. Пусть $\mathbf{H}_1$ – благоприятный набор. Будем говорить, что правило склейки *применимо* к посылке $\mathbf{H}_1$, если набор $\mathbf{H}$, получающийся при его применении, является допустимым (и следовательно, благоприятным набором).

Утверждение 3. Правило склейки применимо к посылке $\mathbf{H}_1$ т. и т. т., когда связка S^* допустима.

Представим правило склейки в операторной записи:

$$\frac{\begin{array}{c}\{i_1,\ldots,\boxed{i_g},\ldots,i_{h-1},\cancel{\boxed{i_h}},i_{h+1},\ldots,i_l;S\}\quad\text{– благоприятный набор}\\ i_g=i_h\ (1\leqslant g,h\leqslant l)\\ \text{Связка } S^*=[S\cup S^{\square}]\backslash\{x_i^h\}_{i=1}^n \text{ допустима}\end{array}}{\{i_1,\ldots,i_g,\ldots,i_{h-1},i_{h+1},\ldots,i_l;S^*\}\quad\text{– благоприятный набор}} \tag{2.19}$$

2.3.5. Отношение отображения

Определение 7. Формула A *отобразима в формулу* B ($A\rightsquigarrow B$), если можно найти такую подстановку θ вместо свободных переменных формулы A, в результате применения которой получается формула B: $\theta A\eqsim B$.

Определение 8. Будем говорить, что секвенция Σ *отобразима в секвенцию* Σ_1 ($\Sigma\rightsquigarrow\Sigma_1$), если существует такая подстановка θ вместо свободных переменных секвенции Σ, что в результате этой подстановки получается секвенция Σ_1, являющаяся подсеквенцией Σ: $\theta\Sigma\eqsim\Sigma_1\subseteq\Sigma$.

Утверждение 4. Если секвенция Σ выводима, то любая секвенция, в которую она отобразима, также выводима.

Определение 9. Пусть даны два набора:

$$\begin{aligned}&\mathbf{H}_1\colon\quad\{i_1,\ldots,i_k;S_1\},\\ &\mathbf{H}_2\colon\quad\{j_1,\ldots,j_s;S_2\}.\end{aligned}$$

Будем говорить, что набор $\mathbf{H}_1$ *отобразим в набор* $\mathbf{H}_2$ ($\mathbf{H}_1\rightsquigarrow\mathbf{H}_2$), если соблюдены следующие условия:

1. $\exists\varphi:\{1,\ldots,k\}\overset{\varphi}{\to}\{1,\ldots,s\}$, где k – длина набора $\mathbf{H}_1$, s – длина набора $\mathbf{H}_2$,

2. $i_u=j_{\varphi(u)}$, для любых u $(1\leqslant u\leqslant k,\ 1\leqslant\varphi(u)\leqslant s)$,

3. $\exists\eta=\{x_p^l=y_p^{\varphi(l)}|\forall x_p^l\in i_l(1\leqslant l\leqslant k,1\leqslant p\leqslant n),\forall y_p^{\varphi(l)}\in j_{\varphi(l)}(1\leqslant\varphi(l)\leqslant s,1\leqslant p\leqslant n)\}$, т. е. найдётся система подстановок η в форме равенств вида $x_p^l=y_p^{\varphi(l)}$,

4. если $\alpha=\beta\in S_1$, то $\eta(\alpha)=\eta(\beta)\in S_2$, где η – система подстановок в форме равенств вида $x_p^l=y_p^{\varphi(l)}$ из п. 3 настоящего определения.

Замечание 3. Набор $\mathbf{H}_1$ отобразим в набор $\mathbf{H}_2$, если существует такая подстановка θ вместо свободных переменных набора $\mathbf{H}_1$, что в результате этой подстановки получается набор $\mathbf{H}_2$, являющийся поднабором $\mathbf{H}_1$: $\theta\mathbf{H}_1\eqsim\mathbf{H}_2\subseteq\mathbf{H}_1$. При равенстве длин наборов

$k = s$ имеет место *тривиальное* отображение, в котором не происходит склейки, в противном случае имеем *нетривиальное* отображение со склейкой. В последнем случае в результате подстановки θ получается набор, к которому применимо правило склейки, результатом которого является набор $\mathbf{H}_2$, являющийся строгим поднабором $\mathbf{H}_1$: $\mathbf{H}_2 \subset \mathbf{H}_1$.

2.3.6. Наборы и примеры наборов

Определение 10. Будем называть *F-секвенциями* секвенции, начинающиеся с формулы F:

$$\to F, \ldots . \tag{2.20}$$

Утверждение 5. Если F-секвенция выводима, то в неё отобразим канонический пример некоторого благоприятного набора.

Определение 11. *Примером набора* $\mathbf{H}$ будем называть F-секвенцию вида:

$$\to F, \widetilde{A_{i_1}}, \ldots, \widetilde{A_{i_k}}, \tag{2.21}$$

где $A_{i_1}, \ldots, A_{i_k}$ – компоненты набора $\mathbf{H}$, $\widetilde{A_{i_j}}$ – результат подстановки вместо свободных переменных $x_1, \ldots, x_n$ блока A_{i_j} секвенции (2.7) $(1 \leqslant i_j \leqslant \delta)$, такая, что все равенства вида (2.10) из связки набора $\mathbf{H}$ выполняются.

Утверждение 6. Если замкнутый набор допустим, то любой его пример содержит контрарную пару.

Замечание 4. По любому решению системы уравнений вида (2.10) из связки набора $\mathbf{H}$ можно построить пример набора $\mathbf{H}$.

Определение 12. Решение системы уравнений вида (2.10) будем называть *универсальным*, если это решение отобразимо на все остальные решения.

Утверждение 7. Если система уравнений вида (2.10) имеет решение, то она имеет универсальное решение.

Замечание 5. Всякое решение получается из универсального решения путём насыщения системы равенств вида (2.10), т. е. добавления к ней новых равенств.

Определение 13. *Каноническим примером набора* $\mathbf{H}$ ($\ulcorner\mathbf{H}\urcorner$) будем называть F-секвенцию вида:

$$\to F, \sigma A_{i_1}, \ldots, \sigma A_{i_k}, \tag{2.22}$$

где $A_{i_1}, \ldots, A_{i_k}$ – компоненты набора $\mathbf{H}$, σ – универсальное решение системы уравнений вида (2.10) для переменных блоков $A_{i_1}, \ldots, A_{i_k}$.

Замечание 6. Для пустого набора его каноническим примером является F-секвенция

$$\to F.$$

Замечание 7. Будем считать, что у каждого набора есть только один канонический пример, поскольку все канонические примеры одного набора являются тождественными друг другу в точности до переименования переменных.

Утверждение 8. По любому благоприятному набору можно построить его канонический пример.

Утверждение 9. Если секвенция Σ содержит контрарную пару, то найдётся замкнутый набор $\mathbf{H}$, такой, что его канонический пример $\ulcorner\mathbf{H}\urcorner$ отобразим в эту секвенцию и этот канонический пример содержит контрарную пару.

Утверждение 10. Для всякого набора $\mathbf{H}$ верно, что его канонический пример $\ulcorner\mathbf{H}\urcorner$ отобразим в любой пример этого набора.

Утверждение 11. Пусть $\mathbf{H}_1$ и $\mathbf{H}_2$ – благоприятные наборы и набор $\mathbf{H}_1$ отобразим в набор $\mathbf{H}_2$. Тогда канонический пример $\ulcorner\mathbf{H}_1\urcorner$ отобразим на любой пример набора $\mathbf{H}_2$, в частности в канонический пример $\ulcorner\mathbf{H}_2\urcorner$.

Утверждение 12. Пусть $\mathbf{H}_1$ и $\mathbf{H}_2$ – благоприятные наборы и набор $\mathbf{H}_1$ отобразим в набор $\mathbf{H}_2$ (в частности, применимо правило склейки). Тогда если выводим канонический пример $\ulcorner\mathbf{H}_1\urcorner$, то выводим любой пример набора $\mathbf{H}_2$, в частности в канонический пример $\ulcorner\mathbf{H}_2\urcorner$.

Утверждение 13. Набор допустим (недопустим) т. и т. т., когда существует (не существует) канонический пример набора.

2.3.7. Теоремы о корректности и полноте специального варианта обратного метода

Сформулируем для специального варианта обратного метода основные теоремы о корректности и полноте.

Теорема 1. (теорема о корректности.) *Пусть дана произвольная секвенция Σ в языке $\mathcal{L}_0$ и исходное исчисление $\boldsymbol{G}_0$. Тогда если в соответствующем ей исчислении $\boldsymbol{G}^*$ благоприятных Σ-наборов выводим пустой набор $\square$, то секвенция Σ выводима в исходном исчислении $\boldsymbol{G}_0$*:

$$\boldsymbol{G}^* \vdash \square \;\Rightarrow\; \boldsymbol{G}_0 \vdash \Sigma. \tag{2.23}$$

Теорема 2. (теорема о полноте.) *Пусть дана произвольная секвенция* Σ *в языке* $\mathcal{L}_0$ *и исходное исчисление* $\boldsymbol{G}_0$. *Тогда если секвенция* Σ *выводима в исходном исчислении* $\boldsymbol{G}_0$, *то в соответствующем ей исчислении* $\boldsymbol{G}^*$ *благоприятных* Σ*-наборов выводим пустой набор* $\square$:

$$\boldsymbol{G}_0 \vdash \Sigma \;\Rightarrow\; \boldsymbol{G}^* \vdash \square. \tag{2.24}$$

2.3.8. Новая нотация для специального варианта обратного метода

Нотация обратного метода поиска вывода достаточно сложна как в первоначальных его вариантах, предложенных С. Ю. Масловым, так и в его более поздних версиях. Для эффективной автоматизации поиска вывода обратным методом и построения доказательств при помощи компьютера необходимо упростить нотацию обратного метода.

Рассмотренный специальный вариант обратного метода, предложенный В.П.Оревковым, также не решает вопрос удобства нотации[3]. Эту проблему позволяет решить новый способ записи, который является корректным, кроме того, он проще воспринимается и его легче запрограммировать.

В основе новой нотации лежит следующая простая идея: привязать элементы связки к тем компонентам набора, чьи переменные фигурируют в данном элементе связки.

Пример. Рассмотрим набор:

$$[2, 2; x^1 = a, x^2 = y^1].$$

В связке есть зависимости для переменных из первого и второго компонентов наборов. Если записать рядом с компоненнтом набора относящиеся к нему элементы связки, то получим своего рода запись примера набора с фиксированной системой подстановок:

$$\{2[x = a, y = x], 2[x = y]\}.$$

Алгоритм преобразования традиционной нотации в новую

1. Выражения типа

$$x_n^i = a,$$

[3]Нужно отметить, что на специальном семинаре по теории логического вывода В.П. Оревков представил специальную графовую нотацию для обратного метода. Однако её анализ не вошёл в задачи настоящего исследования.

где a – некоторая константа, преобразуются в $x_n = a$ и записываются в квадратных скобках у i-того компонента набора:

$$i[x_n = a].$$

2. Выражения типа

$$x_n^i = y_m^j,$$

где y_m^j – переменная, преобразуются в два выражения:

(a) $x_n = z$;

(b) $y_m = z$,

для некоторой новой переменной z и записываются в квадратных скобках:

(a) у i-того компонента набора:

$$i[x_n = z],$$

(b) у j-того компонента набора:

$$j[y_m = z].$$

Основной задачей при разработке новой нотации являлось избавление от лишней индексации в выражениях связки наборов. Проблема состоит в том, что при применении правил вывода индексацию приходится корректировать, что представляет собой, вообще говоря, довольно трудоёмкий процесс. Сранение эффективности новой нотации по отношению к традиционной приведено на примере 3 выводов обратным методом (см. разд. 2.4.).

2.4. Примеры построения выводов обратным методом

Рассмотрим несколько примеров построения выводов при помощи специального варианта обратного метода. На первом примере будут подробно разобраны все шаги построения вывода, прояснены все этапы работы, проводимой над секвенцией и получаемыми наборами. В последующих примерах такая детализация будет опущена.

Пример 1. Пусть дана секвенция Σ:

$$\to \exists x(R(a,x) \vee \neg R(x,b)). \tag{2.25}$$

Построим её доказательство в исчислении $\boldsymbol{G}_0$:

$$\dfrac{\dfrac{\to \boxed{R(a,b)} \vee \neg R(b,b), R(a,a) \vee \boxed{\neg R(a,b)}\ , \Gamma}{\to R(a,b) \vee \neg R(b,b), \Gamma} \to \exists}{\to \exists x(R(a,x) \vee \neg R(x,b)) \leftrightharpoons \Gamma} \to \exists$$

Секвенция (2.25) имеет вид:

$$\to \bigvee_{i=1}^{n} P_i(\bigwedge_{j=1}^{\delta_i} D_{i,j}), \tag{2.26}$$

где P_i – произвольные кванторные префиксы, $D_{i,j}$ – простые дизъюнкции, при $n = 1$ и $\delta_1 = 1$. В разбивку секвенции (2.25) входит одна простая дизъюнкция $D_{1,1}$:

$$R(a,x) \vee \neg R(x,b). \tag{2.27}$$

Пронумеруем элементы (2.27) и зафиксируем список C:

$$1. \tag{2.28}$$

Простая диъюнкция $D_{1,1}$ имеет вид:

$$D_{i,j} = \bigvee_{k=1}^{\mu_{i,j}} A_{i,j,k}, \tag{2.29}$$

где $A_{i,j,k}$ – элементарные формулы.

Для наглядности изобразим все обозначения на разбивке (2.27):

$$\overbrace{\underbrace{R(a,x)}_{A_{1,1,1}} \vee \underbrace{\neg R(x,b)}_{A_{1,1,2}}}^{D_{1,1}}, \tag{2.30}$$

Найдём все контрарные пары элементарных формул $A_{i,j,k}$ из (2.30) (т. е. такие пары формул, одна из которых может стать отрицанием другой при соответствующей унификации аргументов). Такая пара всего одна:

$$A_{1,1,1} \text{ и } A_{1,1,2}, \qquad \text{т. е.} \qquad R(a,x) \text{ и } \neg R(x,b). \tag{2.31}$$

Рассмотрим пару формул (2.31). Одночленный набор по этой паре построить нельзя, поскольку нет унификации, делающей эти формулы контрарными. Но можно построить двучленный набор. Для этого нужно осуществить две подстановки в один и тот же элемент развивки и получить два примера этой формулы. Чтобы отличать одноимённые собственные переменные, входящие в разные компоненты набора, снабдим собственные переменные формул верхними индексами:

$$R(a,x^1) \text{ и } \neg R(x^2,b). \tag{2.32}$$

Здесь видно, что в первом компоненте набора нас будет интересовать позитивная подформула, а во втором – негативная. Однако можно выбрать подформулы наоборот:

$$\neg R(x^1, b) \text{ и } R(a, x^2). \tag{2.33}$$

Из (2.32) видно, что:

1) если вместо x^2 подставить терм a, то мы унифицируем первый аргумент этой пары: $R(a, x^1)$ и $\neg R(a, b)$,

2) если теперь вместо x^1 подставить терм b, то мы унифицируем и второй аргумент этой пары: $R(a, b)$ и $\neg R(a, b)$, получив таким образом контрарную пару.

Запишем установленные зависимости в форме:

1) $x^2 = a$;

2) $x^1 = b$.

Теперь мы готовы записать замкнутый набор $\mathbf{H}_1$:

$$[1, 1; x^2 = a, x^1 = b], \tag{2.34}$$

где список $1, 1$ представляет собой список номеров формул разбивки (2.28), который в данном случае означает, что обе формулы, которые образуют контрарную пару, входят в первую формулу разбивки (2.27), а список $x^2 = a, x^1 = b$ представляет собой допустимую систему зависимостей, при которой обе формулы образуют замкнутый набор. По правилу **А**, набор $\mathbf{H}_1$ является благоприятным.

Перепишем набор $\mathbf{H}_1$ в новой нотации:

$$\{1[x = b], 1[x = a]\}. \tag{2.35}$$

Для контрарной пары (2.33) зависимости будут обратными:

1) $x^1 = a$;

2) $x^2 = b$,

а соответствующий замкнутый набор $\mathbf{H}_2$ записывается:

$$[1, 1; x^1 = a, x^2 = b], \tag{2.36}$$

он же в новой нотации:

$$\{1[x = a], 1[x = b]\}. \tag{2.37}$$

Поскольку $\delta_1 = 1$, правило **Б** для данной секвенции имеет всего одну посылку. Применяя дважды правило **Б** к любому из наборов $\mathbf{H}_1$ или $\mathbf{H}_2$, получаем пустой набор.

Покажем это на примере набора $\mathbf{H}_1$: при первом применении правила **Б** получаем сначала набор $\{1[x = b]\}$, а потом пустой набор $\square$.

Запишем получившийся вывод пустого набора для секвенции (2.25) в исчислении благоприятных Σ-наборов $\mathbf{G}^*$:

$$\begin{array}{lll} 1. & \{1[x = b], 1[x = a]\} & \text{А} \\ 2. & \dfrac{\{1[x{=}b],\ 1[\cancel{x{=}a}]\}}{\{1[x{=}b]\}} & \text{Б, 1} \\ 3. & \dfrac{\{\cancel{1[x{=}b]}\}}{\square} & \text{Б, 2} \end{array} \tag{2.38}$$

В этом простом случае не возникает проблем с зависимостями, поэтому применение правила **Б** происходит автоматически. Рассмотрим немного более сложный случай.

Пример 2. Рассмотрим секвенцию Σ:

$$\to \exists x((P(x) \vee \neg P(a)) \ \& \ (P(x) \vee \neg P(b))). \tag{2.39}$$

Построим её доказательство в исчислении $\boldsymbol{G}_0$:

$$\dfrac{\dfrac{\dfrac{\to \boxed{P(a)} \vee \neg P(b),\ P(b) \vee \boxed{\neg P(a)},\ \Gamma \qquad \to \boxed{P(b) \vee \neg P(b)},\ \Gamma_1}{\to P(b) \vee \neg P(a) \ \& \ P(b) \vee \neg P(b), \Gamma_1} \to \& \\ \to P(a) \vee \neg P(b), \Gamma \leftrightharpoons \Gamma_1}{\to \exists} \\ \dfrac{\to \boxed{P(a) \vee \neg P(a)},\ \Gamma}{\dfrac{\to P(a) \vee \neg P(a) \ \& \ P(a) \vee \neg P(b), \Gamma}{\to \exists x(P(x) \vee \neg P(a) \ \& \ P(x) \vee \neg P(b)) \leftrightharpoons \Gamma} \to \exists} \to \&$$

Секвенция (2.39) имеет вид (2.26) при $n = 1$ и $\delta_1 = 2$. Её разбивой является список из двух простых дизъюнкций $D_{1,1}$ и $D_{1,2}$, каждая из которых имеет вид (2.29):

$$P(x) \vee \neg P(a), P(x) \vee \neg P(b). \tag{2.40}$$

Пронумеруем простые дизъюнкции $D_{1,1}$ и $D_{1,2}$ и зафиксируем список C, соответствующий разбивке (2.40):

$$1, 2. \tag{2.41}$$

Тогда разбивку можно представить следующим образом:

$$\overbrace{\underbrace{P(x)}_{A_{1,1,1}} \vee \underbrace{\neg P(a)}_{A_{1,1,2}}}^{D_{1,1}}, \overbrace{\underbrace{P(x)}_{A_{1,2,1}} \vee \underbrace{\neg P(b)}_{A_{1,2,2}}}^{D_{1,2}}, \tag{2.42}$$

Замкнутыми наборами для рассматриваемой секвенции будут:

$$\begin{gathered} [1; x^1 = a],\ [1, 2; x^2 = a], \\ [2; x^1 = b],\ [1, 2; x^1 = b], \end{gathered} \tag{2.43}$$

или, пользуясь другой нотацией:

$$\{1[x = a]\},\ \{1, 2[x = a]\},\\ \{2[x = b]\},\ \{1[x = b], 2\}. \tag{2.44}$$

Поскольку $\delta_1 = 2$, все применения правила **Б** имеют две посылки.

Построим вывод пустого набора для секвенции (2.39) в соответствующем исчислении благоприятных Σ-наборов $\mathbf{G}^*$:

1.	$\{1[x = a]\}$	А
2.	$\{2[x = b]\}$	А
3.	$\{1[x = b], 2\}$	А
4.	$\{1, 2[x = a]\}$	А
5.	$\dfrac{\{1[\cancel{x=a}]\} \quad \{1,\ 2[\cancel{x=a}]\}}{\{1\}}$	Б, 1, 4
6.	$\dfrac{\{2[\cancel{x=b}]\} \quad \{1[\cancel{x=b}],\ 2\}}{\{2\}}$	Б, 2, 3
7.	$\dfrac{\{\cancel{1}\} \quad \{\cancel{2}\}}{\square}$	Б, 5, 6

(2.45)

Стоит заметить, что попытка применить правило **Б** к 1-й и 2-й посылкам (а также к к 3-й и 4-й) не привела бы нас к успешному построению доказательства, поскольку в этом случае был бы получен недопустимый набор с неразрешимой системой зависимостей.

Пример 3. Рассмотрим секвенцию Σ:

$$\to \exists x, y((R(a, x) \vee \neg R(x, b))\ \&\ (Q(a, y) \vee \neg Q(y, b))). \tag{2.46}$$

Построим её доказательство в исчислении $\boldsymbol{G}_0$:

$$\dfrac{\dfrac{\dfrac{\vdots}{\to R(a,b) \vee \neg R(b,b), \Gamma} \qquad \dfrac{\vdots}{\to Q(a,b) \vee \neg Q(b,b), \Gamma}}{\to R(a,b) \vee \neg R(b,b)\ \&\ Q(a,b) \vee \neg Q(b,b), \Gamma}\to\&}{\to \exists x, y(R(a,x) \vee \neg R(x,b)\ \&\ Q(a,y) \vee \neg Q(y,b)) \leftrightharpoons \Gamma}\to\exists$$

Левая ветвь этого доказательства выглядит следующим образом:

$$\dfrac{\dfrac{\to R(a,a) \vee \boxed{\neg R(a,b)}\ , \Gamma_1\big[\boxed{R(a,b)} \vee \neg R(b,b)\big] \qquad \dfrac{\vdots}{\to Q(a,a) \vee \neg Q(a,b), \Gamma_1}}{\to R(a,a) \vee \neg R(a,b)\ \&\ Q(a,a) \vee \neg Q(a,b), \Gamma_1}\to\&}{\to R(a,b) \vee \neg R(b,b), \Gamma \leftrightharpoons \Gamma_1}\to\exists,$$

где недостающая часть следующая:

$$\dfrac{\dfrac{\to R(a,a)\vee\boxed{\neg R(a,b)}\ ,\ \Gamma_2\big[\boxed{R(a,b)}\vee\neg R(b,b)\big] \qquad \to\boxed{Q(a,b)}\vee\neg Q(b,b),\Gamma_2\big[Q(a,a)\vee\boxed{\neg Q(a,b)}\big]}{\to R(a,a)\vee\neg R(a,b)\ \&\ Q(a,b)\vee\neg Q(b,b),\Gamma_2}\to\&}{\to Q(a,a)\vee\neg Q(a,b),\Gamma_1\leftrightarrows\Gamma_2}\to\exists$$

Правая ветвь этого доказательства выглядит следующим образом:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \to R(a,a)\vee\neg R(a,b),\Gamma_3\end{array} \qquad \to Q(a,a)\vee\boxed{\neg Q(a,b)}\ ,\ \Gamma_3\big[\boxed{Q(a,b)}\vee\neg Q(b,b)\big]}{\to R(a,a)\vee\neg R(a,b)\ \&\ Q(a,a)\vee\neg Q(a,b),\Gamma_3}\to\&}{\to Q(a,b)\vee\neg Q(b,b),\Gamma\leftrightarrows\Gamma_3}\to\exists,$$

где недостающая часть следующая:

$$\dfrac{\dfrac{\to\boxed{R(a,b)}\vee\neg R(b,b),\Gamma_4\big[R(a,a)\vee\boxed{\neg R(a,b)}\big] \qquad \to Q(a,a)\vee\boxed{\neg Q(a,b)}\ ,\ \Gamma_4\big[\boxed{Q(a,b)}\vee\neg Q(b,b)\big]}{\to R(a,b)\vee\neg R(b,b)\ \&\ Q(a,a)\vee\neg Q(a,b),\Gamma_4}\to\&}{\to R(a,a)\vee\neg R(a,b),\Gamma_3\leftrightarrows\Gamma_4}\to\exists$$

Секвенция (2.46) имеет вид (2.26) при $n = 1$ и $\delta_1 = 2$. Её разбивой является список из двух простых дизъюнкций $D_{1,1}$ и $D_{1,2}$, каждая из которых имеет вид (2.29):

$$R(a,x)\vee\neg R(x,b), Q(a,y)\vee\neg Q(y,b). \tag{2.47}$$

Разбивка этой секцвенции:

$$\overbrace{\underbrace{R(a,x)}_{A_{1,1,1}}\vee\underbrace{\neg R(x,b)}_{A_{1,1,2}}}^{D_{1,1}},\overbrace{\underbrace{Q(a,y)}_{A_{1,2,1}}\vee\underbrace{\neg Q(y,b)}_{A_{1,2,2}}}^{D_{1,2}}, \tag{2.48}$$

Замкнутыми наборами для рассматриваемой секвенции будут:

$$\begin{array}{l}[1,1;x^1=b,x^2=a],\ [1,1;x^1=a,x^2=b],\\ [2,2;y^1=b,y^2=a],\ [2,2;y^1=a,y^2=b],\end{array} \tag{2.49}$$

или в другой нотации:

$$\begin{array}{l}\{1[x=b],1[x=a]\},\ \{1[x=a],1[x=b]\},\\ \{2[y=b],2[y=a]\},\ \{2[y=a],2[y=b]\}.\end{array} \tag{2.50}$$

Поскольку $\delta_1 = 2$, все применения правила **Б** имеют две посылки.

Построим вывод пустого набора для секвенции (2.46) в соответствующем исчислении благоприятных Σ-наборов $\mathbf{G}^*$:

$$
\begin{array}{lll}
1. & \{1[x=b], 1[x=a]\} & \text{А} \\
2. & \{1[x=a], 1[x=b]\} & \text{А} \\
3. & \{2[y=b], 2[y=a]\} & \text{А} \\
4. & \{2[y=a], 2[y=b]\} & \text{А} \\
5. & \dfrac{\{1[x{=}b],\ \cancel{1[x{=}a]}\} \quad \{2[y{=}b],\ \cancel{2[y{=}a]}\}}{\{1[x{=}b],2[y{=}b]\}} & \text{Б, 1, 3} \\
6. & \dfrac{\{1[x{=}a],\ \cancel{1[x{=}b]}\} \quad \{2[y{=}a],\ \cancel{2[y{=}b]}\}}{\{1[x{=}a],2[y{=}a]\}} & \text{Б, 2, 4} \\
7. & \dfrac{\{1[x{=}b],\ \cancel{2[y{=}b]}\} \quad \{1[x{=}b],\ \cancel{1[x{=}a]}\}}{\{1[x{=}b],1[x{=}b]\}} & \text{Б, 5, 1} \\
8. & \dfrac{\{1[x{=}b],\cancel{1[x{=}b]}\}}{\{1[x{=}b]\}} & \text{склейка, 7} \\
9. & \dfrac{\{\cancel{1[x{=}a]},2[y{=}a]\} \quad \{\cancel{2[y{=}b]},2[y{=}a]\}}{\{2[y{=}a],2[y{=}a]\}} & \text{Б, 6, 3} \\
10. & \dfrac{\{2[y{=}a],\cancel{2[y{=}a]}\}}{\{2[y{=}a]\}} & \text{склейка, 9} \\
11. & \dfrac{\{\cancel{1[x{=}b]}\} \quad \{\cancel{2[y{=}a]}\}}{\square} & \text{Б, 8, 10}
\end{array}
\tag{2.51}
$$

Этот пример показателен тем, что изначально у нас были только двучленные благоприятные наборы. Применением правила **Б** мы из двучленных наборов можем получать только двучленные же наборы. Как видно из этого примера, в таких ситуациях нам помогает дополнительное правило склейки, при помощи которого мы можем понизить местность получающихся наборов.

Для сравнения эффективности новой нотации по отношению к традиционной, приведём вывод той же секвенции в традиционной нотации:

$$
\begin{array}{lll}
1. & \{1,1; x^1 = b, x^2 = a\} & \text{А} \\
2. & \{1,1; x^1 = a, x^2 = b\} & \text{А} \\
3. & \{2,2; y^1 = b, y^2 = a\} & \text{А} \\
4. & \{2,2; y^1 = a, y^2 = b\} & \text{А} \\
5. & \dfrac{\{1,\ 1;\ x^1{=}b,x^2{=}a\} \quad \{2,\ 2;\ y^1{=}b,y^2{=}a\}}{\begin{array}{c}\{1,\ 2,\frac{1}{2};x^1{=}b,\cancel{x^2{=}a},\cancel{y^1{=}b},\cancel{y^2{=}a}\}\\ \{1,\ 2,\cancel{\frac{1}{2}};x^1 = b,\cancel{x^3 = a},y^2 = b,\cancel{y^3 = a}\}\\ \{1,\ 2;\ x^1 = b, y^2 = b\}\end{array}} & \text{Б, 1, 3} \\
6. & \dfrac{\{1,\ 1;\ x^1{=}a,\ x^2{=}b\} \quad \{2,\ 2;\ y^1{=}a,\ y^2{=}b\}}{\begin{array}{c}\{1,\ 2,\frac{1}{2};x^1{=}a,\cancel{x^2{=}b},\cancel{y^1{=}a},\cancel{y^2{=}b}\}\\ \{1,\ 2,\cancel{\frac{1}{2}};x^1 = b,\cancel{x^3 = b},y^2 = a,\cancel{y^3 = b}\}\end{array}} & \text{Б, 2, 4}
\end{array}
$$

$\{1,\ 2; x^1 = a, y^2 = a\}$

7. $\dfrac{\{1,\ 1;\ x^1{=}b,\ x^2{=}a\} \qquad \{1,\ 2;\ x^1{=}b,\ y^2{=}b\}}{\{1,\ 1, \frac{1}{2}; x^1{=}b, \cancel{y^2{=}b}, \cancel{x^1{=}b}, \cancel{x^2{=}a}\}}$ (Б, 5, 1)

$\{1,\ 1, \cancel{\frac{1}{2}}; x^1 = b, \cancel{y^3 = b}, x^2 = b, \cancel{x^3 = a}\}$

$\{1,\ 1; x^1 = b, x^2 = b\}$

8. $\dfrac{\{1,\ 1; x^1{=}b, \cancel{x^2{=}b}\}}{\{1; x^1{=}b\}}$ (склейка, 7)

9. $\dfrac{\{1,\ 2;\ x^1{=}a,\ y^2{=}a\} \qquad \{2,\ 2;\ y^1{=}b,\ y^2{=}a\}}{\{2,\ 2\, \frac{1}{2}; \cancel{x^1{=}b}, \cancel{y^2{=}a}, y^2{=}a, \cancel{y^1{=}b}\}}$ (Б, 6, 3)

$\{2,\ 2, \cancel{\frac{1}{2}}; \cancel{x^3 = b}, y^1 = a, \cancel{y^3 = b}, y^2 = a\}$

$\{2,\ 2;\ y^1 = a,\ y^2 = a\}$

10. $\dfrac{\{2,\ 2;\ y^1{=}a,\ y^2{=}a\}}{\{2;\ y^1{=}a\}}$ (склейка, 9)

11. $\dfrac{\{1; x^1{=}b\} \qquad \{2;\ y^1{=}a\}}{\{\frac{1}{2}; x^1{=}b, y^1{=}a\}}$ (Б, 8, 10)

$\{\cancel{\frac{1}{2}}; \cancel{x^1 = b}, \cancel{y^1 = a}\}$

□

На этом примере видно, что при старой нотации каждое применение правила **Б** требует проведения большого количества промежуточных преобразований наборов, связанных, прежде всего, со смещением индексации при удалении отрезаемых частей наборов.

Глава 3.

Программная реализация алгоритма поиска вывода на основе обратного метода

3.1. Схема машинного алгоритма АУВ

В работе [6] изложена схема машинного алгоритма установления выводимости (АУВ) в классическом ичислении предикатов с функциональными знаками, разработанного группой математической логики ЛОМИ. Насколько известно автору, это единственное описание алгоритма АУВ. Прежде чем описывать структуру реализации алгоритма поиска вывода на основе обратного метода, которая использовалась в настоящей работе, проанализируем схему алгоритма АУВ. Общая блок-схема АУВ представлена на рис. 3.1.

На вход алгоритма приходит формула, она попадает в блок «матрица замкнутых» (мз), где составляется своего рода таблица, в которой хранятся все возможные исходные замкнутые наборы для данной формулы, упорядоченные по числу i, последнему компоненту набора. Одновременно с этим в этом блоке распознаются некоторые простейшие случаи невыводимости (например, когда в формуле нет ни одной потенциальной контрарной пары, в силу чего невозможно построить ни одного замкнутого набора).

По мз строится индексированный список специальных совокупностей наборов – «наборов наборов с лакунами» (ннл), обозначенный в схеме как Q.

Следующий блок алгоритма – «Извлечение из Q». В нём из списка Q выбирается такая пара U, k, что применение правила Б к U с индексом k обещает получение

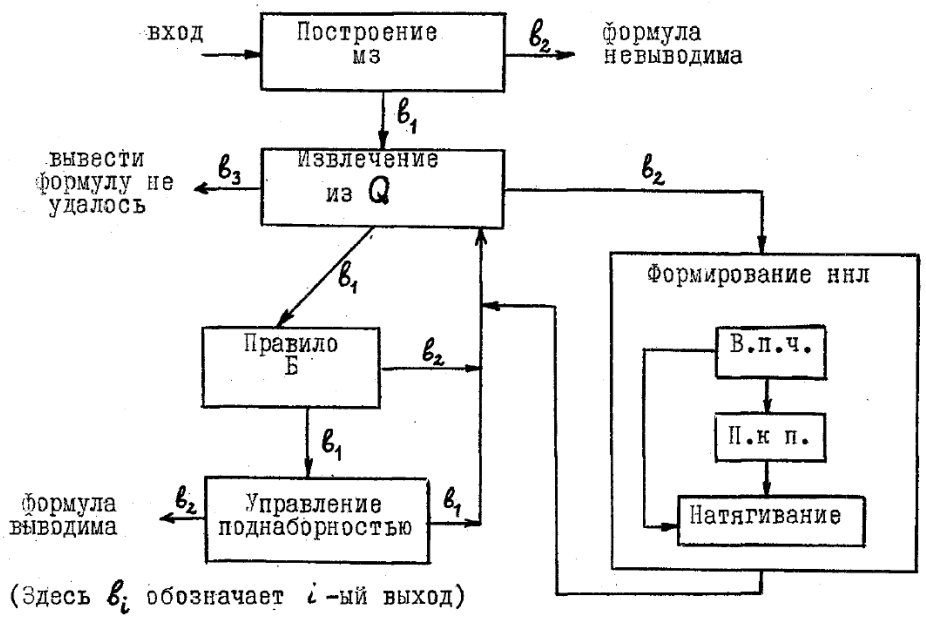

Рис. 3.1: Общая блок-схема АУВ.

наилучшего результата. Одновременно посылается запрос на проверку, возможно ли пополнить Q новыми ннл, которые ещё не использовались и которые обещают более лучший результат. Если такие ннл построить удаётся, происходит обращение в специальный блок «Формирование ннл», в котором эти ннл строятся и затем добавляются в список Q. Для итоговой выбранной пары U, k индекс увеличивается на 1. Если этот индекс превосходит заранее установленное максимальное значение, этот ннл удаляется из Q. Если из Q будут удалены все ннл, то алгоритм возвращает «Вывести формулу не удалось».

После того, как пара U, k выбрана, она попадает в блок «Правило Б», в котором по поступившим наборам строятся в общем случае несколько новых наборов. Если ни одного набора построить не удалось, то алгоритм возвращается к блоку «Извлечение из Q».

Полученные наборы приходят в финальный блок алгоритма – «Управление поднаборностью», в котором анализируются полученные наборы. Если полученный набор является обобщением уже существующего, то этот набор пропускается. Если найден пустой набор, то алгоритм возвращает «Формула выводима», остальные наборы алгоритм записывает в множество всех наборов.

Таким образом, за конечное число шагов алгоритм гарантированно останавливает свою работу. Если алгоритму удалось построить пустой набор, то алгоритм возвращает «Формула выводима», если не удалось, то выводит либо «Формула невыводима», либо

«Вывести формулу не удалось» (если в процессе работы возникали наборы, сложность которых превосходит заранее установленные ограничения, например, на длину записей наборов).

3.2. Схема реализации алгоритма в настоящей работе

В настоящей работе реализовывался специальный вариант обратного метода, описанный в разделе 2.3. Общая блок-схема реализации представлена на рис. 3.2:

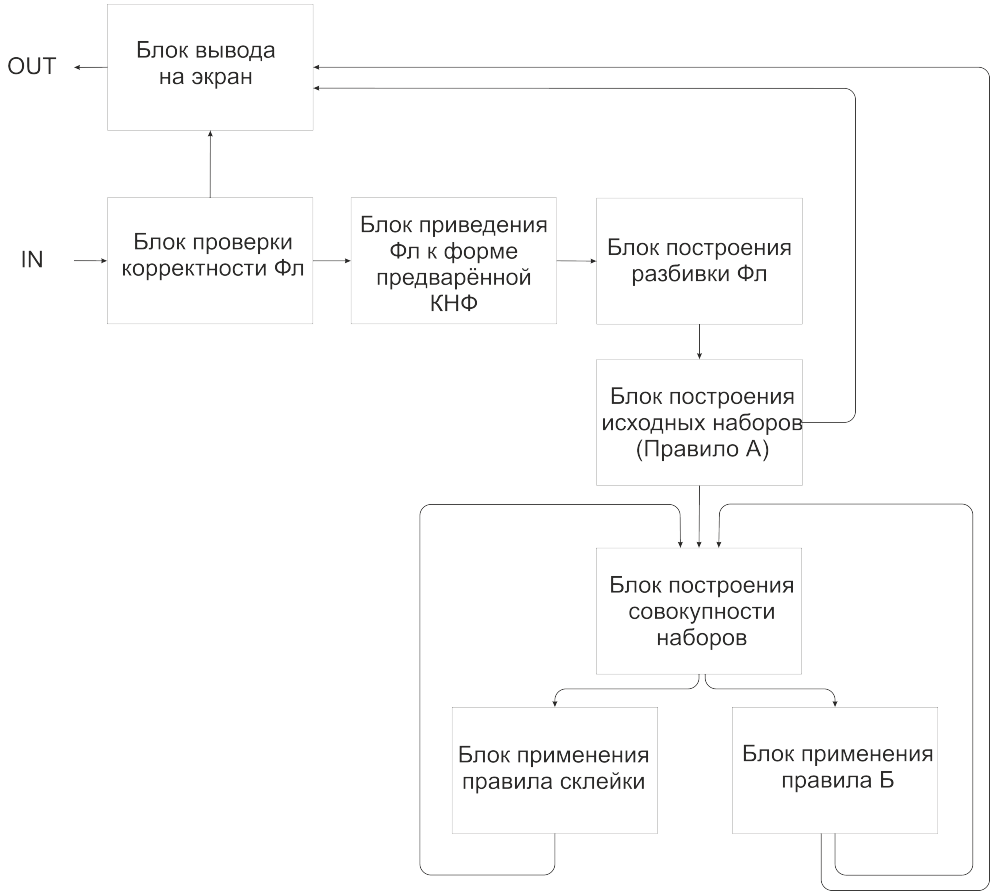

Рис. 3.2: Общая блок-схема программной реализации ОМ настоящей работы.

Из приведённой блок-схемы алгоритма видно, что четыре входных последовательно следующих друг за другом блоков осуществляют проверки и предварительные преобразования входной формулы, а также генерируют исходные наборы для последующего доказательства. Основная работа алгоритма сосредоточена в трёх блоках: «Блоке построения совокупности наборов», обогощаемом на каждой итерации алгоритма новыми наборами, полученными из «Блока применения правила склейки» или «Блока применения правила Б». На выходе работы алгоритма имеется отдельный блок для формирования вывода на экране компьютера.

Глава 4.

Сравнение обратного метода с методом резолюции

4.1. Условия проведения экспериментов

Одной из задач настоящего исследования является сравнение по эффективности алгоритмов обратного метода и метода резолюций. Проведение такого исследования вызвано замечанием С.Ю. Маслова о том, что обратный метод не уступает по эффективности методу резолюции [16]. Это предположение пока должным образом не исследовалось. Некоторые попытки такого исследования для логики высказываний обнаруживаются в работах [22], [23]. Авторы делают заключение, что «для логики высказываний метод Маслова не уступает по эффективности методу резолюций. Полученные результаты подтвердили теоретические оценки С.Ю. Маслова» [23, с. 11-12]. Эти результаты неудовлетворительны по нескольким причинам. Во-первых, сравнение производилось для элементарного подкласса логики предикатов – логики высказываний. Во-вторых, результаты носят абстрактный характер, без указания количественных данных произведённых измерений и сравнений. Таким образом, задача сравнения по эффективности алгоритмов обратного метода и метода резолюций остаётся открытой. На частичное её разрешение было нацелено настоящее исследование.

Для проведения экспериментов по сравнению двух алгоритмов по эффективности нужно ввести некоторые ограничения, чтобы результаты были адекватными. Ключевыми ограничениями на проведение экспериментов являлись:

1) сравнение будет производиться для формул классического исчисления предикатов без функциональных знаков и без равенства;

2) оба метода используются в своём самом базовом варианте, безо всяких оптимизирующих процесс поиска вывода добавлений, тактик и уточнений;

3) множество формул, на которых будет производиться сравнение, должны принадлежать одному классу разрешимости;

4) формулы из тестировочного множества должны обладать одинаковой сложностью, чтобы можно было оценить выборочное среднее времени работы алгоритма.

4.2. Тестовое множество формул

Зафиксирсуем класс формул, на котором будет производиться сравнение работы двух алгоритмов. В качестве тестового множества формул в настоящем исследовании был выбран тривиальный подкласс разрешимого класса ($\forall\exists$) формул логики предикатов вида:

$$\exists x_1, \ldots, x_n \bigwedge_{i=1}^{\delta} A_i. \tag{4.1}$$

Формулы из тестового множества были получены путём сведения задачи о распознавании клик в неориентированных графах к проблеме выводимости формул вида 4.1.

4.2.1. Задача о распознавании клики в графе

Задача о клике [9] относится к классу NP-полных задач в области теории графов. Впервые она была сформулирована в 1972 г. Р. Карпом.

Кликой в неориентированном графе $G = (V, E)$ называется подмножество вершин $V' \subseteq V$, в котором все вершины попарно соединены рёбрами. *Размером* клики называется число содержащихся в ней вершин.

На рис. 4.1 изображён граф, содержащий:

1) 23 клики, содержащие 1 вершину (вершины графа);

2) 42 клики, состоящие из 2 вершин (рёбра графа);

3) 19 клик, состоящих из 3 вершин (закрашенные треугольники);

4) 2 клики, состоящие из 4 вершин (тёмно-синие области).

Задача о клике существует в двух вариантах: 1) (задача распознавания) даны граф G и число k, нужно определить, существует ли в графе G клика размера k? 2) (вычислительный вариант задачи о клике) дан граф G, требуется найти в графе G клику максимального размера.

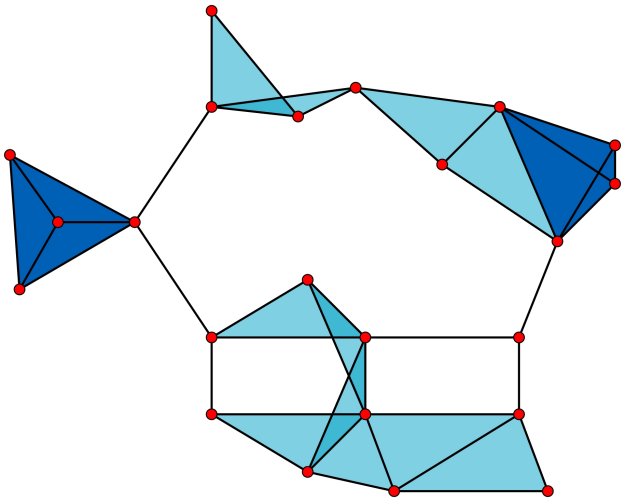

Рис. 4.1: Клики в графе.

4.2.2. Сведение задачи о клике к проблеме распознавания выводимости

В замечании 6.2. работы [19] задача о распознавании клики в графе была представлена как проблема выводимости секвенций вида:

$$E_1, E_2, \ldots, E_k \to \exists x_1, \ldots, x_m \bigwedge_{i=1}^{n} B_i, \tag{4.2}$$

где формулы E_i – замкнутые элементарные формулы вида $R(a_n, a_m)$, описывающие существующие рёбра в графе, формулы B_j – незамкнутые элементарные формулы вида $R(x_s, x_t)$.

Пример. Пусть дан граф G:

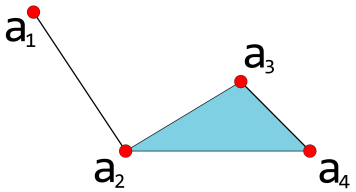

Рис. 4.2: Граф G.

Выразим в логических терминах задачу о распознавании 3-клики в графе G.

Множество рёбер E описывается следующими формулами:

$$R(a_1, a_2), R(a_2, a_3), R(a_2, a_4), R(a_3, a_4). \tag{4.3}$$

Существование 3-клики выражается:

$$\exists x_1, x_2, x_3 \begin{pmatrix} R(x_1,x_1) & \wedge & R(x_2,x_1) & \wedge & R(x_3,x_1) \\ R(x_1,x_2) & \wedge & R(x_2,x_2) & \wedge & R(x_3,x_2) \\ R(x_1,x_3) & \wedge & R(x_2,x_3) & \wedge & R(x_3,x_3) \end{pmatrix}. \tag{4.4}$$

Если считать отношение R рефлексивным, то мы можем устранить излишние отношения вида $R(x_i, x_i)$, если считать его также симметричным, то из пары отношений $R(x_i, x_j)$ и $R(x_j, x_i)$ мы можем оставить только отношения $R(x_i, x_j)$, где $i < j$. Тогда выражение 4.4 упростится:

$$\exists x_1, x_2, x_3 \left(R(x_1,x_2) \wedge R(x_1,x_3) \wedge R(x_2,x_3)\right). \tag{4.5}$$

Полной формулировкой задачи о распознавании 3-клики в графе G будет:

$$R(a_1,a_2), R(a_2,a_3), R(a_2,a_4), R(a_3,a_4) \rightarrow \exists x_1, x_2, x_3 \left(R(x_1,x_2) \wedge R(x_1,x_3) \wedge R(x_2,x_3)\right). \tag{4.6}$$

Перебросив все формулы из антецедента секвенции в сукцедент, внеся их под кванторы существования и применив законы дистрибутивности, получим итоговую секвенцию:

$$\rightarrow \exists x_1, x_2, x_3 \begin{pmatrix} \neg R(a_1,a_2) \vee \neg R(a_2,a_3) \vee \neg R(a_2,a_4) \vee \neg R(a_3,a_4) \vee R(x_1,x_2) \\ \wedge \\ \neg R(a_1,a_2) \vee \neg R(a_2,a_3) \vee \neg R(a_2,a_4) \vee \neg R(a_3,a_4) \vee R(x_1,x_3) \\ \wedge \\ \neg R(a_1,a_2) \vee \neg R(a_2,a_3) \vee \neg R(a_2,a_4) \vee \neg R(a_3,a_4) \vee R(x_2,x_3) \end{pmatrix}. \tag{4.7}$$

Очевидно, что секвенция 4.7 имеет вид:

$$\rightarrow \exists x_1, \ldots, x_n \bigwedge_{i=1}^{\delta} A_i,$$

и, таким образом, формула в сукцеденте секвенции 4.7 имеет требуемый вид 4.1.

4.3. Результаты сравнения

В настоящем исследовании в качестве тестового множества были взяты формулы вида 4.1, описывающие задачи о распознавании клик в графах, подобных рис. 4.2. Тестовое множество формул содержит три подкласса формул, в каждом из которых содержится по 20 формул разной сложности. Сложность формул оценивается по количеству входящих в неё атомарных подформул. Так, сложность формулы в сукцеденте секвенции 4.7 равна 15. Были взяты классы формул со сложностями 7, 15 и 30 соответственно:

Таблица 4.1: Тестовое множество формул.

Сложность формул	**Тип графа**		**Размер клики**
	Количество узлов	**Количество дуг**	
7	≥ 6	4	3
15	≥ 7	9	3
30	≥ 7	9	4

В настоящем исследовании были проведены эксперименты по замеру времени работы алгоритмов на трёх группах формул со сложностями 7, 15 и 30 соответственно. Каждый эсперимент повторялся 50 раз. Результаты проведённых экспериментов по замеру времени работы алгоритма, основанного на обратном методе, обобщены в табл. 4.2. Результаты проведённых экспериментов по замеру времени работы алгоритма, основанного на методе резолюций, – в табл. 4.3.

Таблица 4.2: Результаты замера времени работы алгоритма обратного метода

Сложность формул (количество ЭФ)	**Среднее время вывода (сек.)**	**95%-доверительный интервал**
7	1.3	1.3 ± 0.1
15	28.3	28.3 ± 2.2
30	808.1	808.1 ± 32.4

Таблица 4.3: Результаты замера времени работы алгоритма метода резолюции

Сложность формул (количество ЭФ)	**Среднее время вывода (сек.)**	**95%-доверительный интервал**
7	6.2	6.2 ± 0.3
15	31.8	31.8 ± 3.8
30	963.7	963.7 ± 27.2

Данные двух таблиц представлены в виде графиков на рис. 4.3. Из графиков видно, что имеется незначительное преимущество обратного метода в сравнении с методом

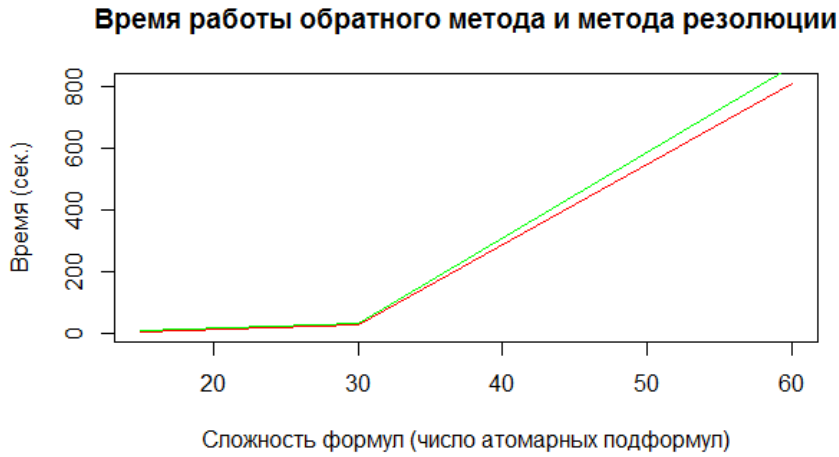

Рис. 4.3: Время работы алгоритмов на основе обратного метода и метода резолюций.

резолюции. Но существенного выигрыша не обнаружено, оба метода обладают приблизительно равной экспоненциальной сложностью.

Заключение

Настоящее исследование посвящено обратному методу поиска выводов в логических исчислениях, автором которого является петербургский математик С.Ю. Маслов. Этот метод был разработан и реализован в Ленинградском отделении математической логики Математического института им. В.А. Стеклова (ЛОМИ). Будучи изобретённым одновременно и независимо от метода резолюций, обратный метод, в отличие от последнего, не нашёл широкого применения и поэтому требует дополнительных исследований. Целью настоящего исследования являлось восполнение этих пробелов в части построения программной реализации одного из вариантов обратного метода, а также сравнения его по эффективности с методом резолюций.

Для достижения целей настоящего иследования необходимо было решение следующих задач:

1) анализ и реконструкция общей схемы обратного метода;

2) сравнение разных конкретизаций обратного метода;

3) выбор одного варианта обратного метода для программной реализации;

4) выбор класса доказуемых формул;

5) программная реализация выбранного варианта обратного метода;

6) сравнение по эффективности работы алгоритма, построенного на основе обратного метода, с работой алгоритма на базе метода резолюций для выбранного класса формул.

Выполнение первой задачи осуществлялось в первой главе «Общая схема обратного метода». В ней была проанализирована общая схема обратного метода, были введены основные понятия: «набор», «система зависимостей», «разбивка» и др.

Вторая задача была решена лишь частично и не вошла в общую часть исследования. При пристальном внимании оказалось, что имеется большое ранообразие конкретизаций обратного метода, сопоставление которых требует большой аналитической работы, время на которую в настоящем исследовании не хватило. Поэтому вместо анализа и

сравнения существующих конкретизаций обратного метода, в настоящем исследовании было принято решение о подборе такого варианта, который бы удовлетворял следующим требованиям: 1) конкретизация должна быть предназначена для классической логики предикатов; 2) конкретизация должна быть для языка логики предикатов, который не включает функциоанльных знаков и равенства; 3) конкретизация должна быть простой, без реализаций тактик поиска вывода; 4) формулировка конкретизации должна быть удобной для программной реализации.

Всем укзанным выше требованиям удовлетворял специальный вариант обратного метода, предложенный В.П. Оревковым, на котором было принято решение остановиться. Особым преимуществом выбранного варианта является простота формулировки правил вывода обратного метода. Таким образом, частичное решение второй и полное решение третьей задачи было дано во второй главе «Конкретизации общей схемы обратного метода», где были проанализированы некоторые конкретные варианты обратного метода и подробно рассмотрен выбранный для реализации специальный вариант обратного метода. Для специального варианта обратного метода была предложена новая нотация и приведены несколько примеров выводов, построенных обратным методом.

Пятая задача была решена в третьей главе «Программная реализация алгоритма поиска вывода на основе обратного метода», в которой была рассмотрена схема алгоритма АУВ, разработанного группой математической логики ЛОМИ в 60-х гг. 20 в. и предложена блок-схема реализации алгоритма, принятая в настоящем исследовании и реализованная на языке высокого уровня Java.

Наконец, четвёртая и седьмая задачи были решены в четвёртой главе «Сравнение обратного метода с методом резолюции», в которой были зафиксированы условия проведения экспериментов по сравнению эффективности обратного метода и метода резолюции. В частности, было сформировано тестовое множество формул, которые были получены путём сведения задачи о распознавании клик в неориентированных графах к проблеме выводимости формул необходимого вида. Был проведён ряд экспериментов по замеру времени работы двух алгоритмов. Было установлено небольшое преимущество обратного метода перед методом резолюции для выбранного класса формул.

Результатами настоящего исследования являются:

1) написана программная реализация специального варианта обратного метода;

2) предложена новая нотация для обратного метода;

3) произведено сравнение алгоритмов обратного метода и метода резолюций по времени работы алгоритмов.

Литература

[1] Бурлуцкий В. В. Реализация обратного метода установления выводимости для модальной логики KT. [Текст]: дис. ... канд. физ.-мат. наук, 05.13.01 – Системный анализ, управление и обработка информации. / В. В. Бурлуцкий; Томский гос. ун-т. – Томск, 2001. – 103 с.

[2] Давыдов Г. В. Метод установления выводимости в классическом исчислении предикатов. // Исследования по конструктивной математике и математической логике. I, Зап. научн. сем. ЛОМИ, 4. – М., 1967, 8-17.

[3] Давыдов Г. В. Некоторые замечания о поиске вывода в исчислении предикатов. // Исследования по конструктивной математике и математической логике. II, Зап. научн. сем. ЛОМИ, 8. – Л.: Изд-во «Наука», 1968, 8-20.

[4] Давыдов Г. В. О корректировании недоказуемых формул. // Исследования по конструктивной математике и математической логике. I, Зап. научн. сем. ЛОМИ, 4. – М., 1967, 18-29.

[5] Давыдов Г. В. Синтез метода резолюций с обратным методом. // Исследования по конструктивной математике и математической логике. IV, Зап. научн. сем. ЛОМИ, 20. – Л.: Изд-во «Наука», 1971, 24-35.

[6] Давыдов Г. В., Маслов С. Ю., Минц Г. Е., Оревков В. П., Слисенко А. О. Машинный алгорифм установления выводимости на основе обратного метода. // Исследования по конструктивной математике и математической логике. III, Зап. научн. сем. ЛОМИ, 16. – Л.: Изд-во «Наука», 1969, 8-19.

[7] Евстигнеев В. А. Применение теории графов в программировании. – М.: Издательство «Наука», 1985. – 352 с.

[8] Катречко С. Л. Обратный метод С. Ю. Маслова и его модификации. // Логика и компьютер (вып. 2): логические языки, содержательные рассуждения и методы поиска доказательства. – М.: «Наука», 1995, 62-75.

[9] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. Издание 3-е. – М.:«Вильям», 2013. – 1328 с.

[10] Ларионов Д. С. Обратный метод установления выводимости для автоэпистемической логики и его применение в экспертных системах. [Текст]: дис. ... канд. техн. наук, 05.13.01 – Системный анализ, управление и обработка информации. / Д. С. Ларионов; Томский политехн. ун-т. – Томск, 2005. – 148 с.

[11] Маслов С. Ю. Обратимый секвенциальный вариант конструктивного исчисления предикатов. // Исследования по конструктивной математике и математической логике. I, Зап. научн. сем. ЛОМИ, 4, М., 1967, 96-111.

[12] Маслов С. Ю. Обратный метод и тактики установления выводимости для исчисления с функциональными знаками. // Логические и логико-математические исчисления. 2, Тр. МИАН СССР, 121, 1972, 14-56.

[13] Маслов С. Ю. Обратный метод установления выводимости в классическом исчислении предикатов. // ДАН СССР, 159, № 1, 1964, 17-20.

[14] Маслов С. Ю. Обратный метод установления выводимости для логических исчислений. // Логические и логико-математические исчисления. I, Тр. МИАН СССР, 98, 1968, 26-87.

[15] Маслов С. Ю. Распространение обратного метода на исчисление с равенством. // Исследования по конструктивной математике и математической логике. IV, Зап. научн. сем. ЛОМИ, 20. – Л.: Изд-во «Наука», 1971, 80-96.

[16] Маслов С. Ю. Связь между тактиками обратного метода и метода резолюций. Зап. научн. сем. ЛОМИ, 16. – Л.: Изд-во «Наука», 1969, 137–146.

[17] Маслов С. Ю. Тактики поиска вывода, основанные на унификации порядка членов в благоприятном наборе. // Исследования по конструктивной математике и математической логике. III, Зап. научн. сем. ЛОМИ, 16. – Л.: Изд-во «Наука», 1969, 126-136.

[18] Минц Г. Е. Теорема Эрбрана. // Математическая теория логического вывода. М.: Изд-во «Наука», 1967, 311-350.

[19] Оревков В. П. Новый разрешимый хорновский фрагмент исчисления предикатов. // Теория сложности вычислений. IX, Зап. научн. сем. ПОМИ, 316, ПОМИ, СПб., 2004, 147-162.

[20] Оревков В. П. Обратный метод поиска вывода. // Адаменко А.Н., Кучуков А.М. Логическое программирование и Visual Prolog. СПб.: БХВ-Петербург, 2003, 952-965.

[21] Оревков В. П. Правило сечения в методе резолюций. // Исследования по конструктивной математике и математической логике. XII, Посвящается памяти Николая Александровича Шанина, Зап. научн. сем. ПОМИ, 407, ПОМИ, СПб., 2012, 111-128.

[22] Павлов В. А., Пак В. Г. Сравнение эффективности методов автоматического доказательства теорем. // XXXIX Неделя науки СПбГПУ : материалы международной научно-практической конференции. Инженерно-технические науки. Часть 2. – СПб.: Изд-во Политехн. ун-та, 2011, 529-531.

[23] Павлов В. А., Пак В. Г. Сравнительное исследование методов автоматического логического вывода для логики предикатов. XXXX Неделя науки СПбГПУ : материалы международной научно-практической конференции. – СПб: Изд-во СПбГПУ 2012, 10-12.

[24] Петухова Н. Д., Косовская Т. М. Решение задач логико-предметного распознавания образов с использованием тактик обратного метода Маслова. // Компьютерные инструменты в образовании. № 3, 2014, 9-20.

[25] Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. – М.: «Наука», 1983.

[26] Degtyarev A., Voronkov A. The Inverse Method. – Handbook of Automated Reasoning, Elsevier Science Publishers B. V., 2001, pp. 180-272.

[27] Kosovskaya T., Petukhova N. The Invers Method for solving Artificial Intelligence Problems in the Frameworks of Logic-Objective Approach and Bounds of its Number of Steps // International Journal «Information Models and Analyses», Vol. 1, 2012, pp. 84-93.

[28] Kosovskaya T., Petukhova N. The Invers Maslov Method and Ant Tactics for Exhaustive Search Decreasing // International Journal «Information Models and Analyses», Vol. 2, 2012, No 1, pp. 81-89.

[29] Lifschitz V. What is the inverse method? – Journal of Automated Reasoning, vol. 5(1), 1989, pp. 1-23.

[30] Mints G. Decidability of the Class E by Maslov's Inverse Method. // Fields of Logic and Computation. Essays Dedicated toYuri Gurevich on the Occasion of His 70th Birthday. / Andreas Blass Nachum Dershowitz Wolfgang Reisig (Eds.). 2010, pp. 529-537.