ПУСТЬ ДОКАЖЕТ КОМПЬЮТЕР

К 250-летию Московского
государственного университета
им. М.В. Ломоносова

Коллективную монографию писали следующие авторы:
Глава I – Болотов А.Е., Бочаров В.А., Горчаков А.Е., Макаров В.В., Шангин В.О.
Глава II – Болотов А.Е., Бочаров В.А., Горчаков А.Е.
Глава III – Шангин В.О.
Глава IV – Макаров В.В.

Книга посвящена рассмотрению проблемы автоматического поиска теорем для классической и интуиционистской логик. В ней предлагаются построенные авторами алгоритмы поиска доказательства для натуральных исчислений классической и интуиционистской логик высказываний, а также для натурального первопорядкового исчисления предикатов. Относительно этих алгоритмов доказываются метатеоремы об их непротиворечивости и полноте. Книга будет интересна всем специалистам, занимающимся проблемами искусственного интеллекта.

ГЛАВА I

ИСТОРИЧЕСКОЕ ВВЕДЕНИЕ

В современном информационном обществе с особой остротой встал вопрос о передаче части мыслительных действий человека машине, что отразилось в возникновении новой сферы исследований – искусственном интеллекте. Одним из многих направлений этой бурно развивающейся отрасли является автоматический поиск доказательства в некоторой формальной системе.

Задачи, связанные с нахождением доказательства некоторого утверждения, интересовали человечество и ранее. Однако понятие автоматического поиска доказательства заставило по-новому рассматривать данные задачи и, в конечном счете, определило один из критериев оценки исчислений. А именно выбор между исчислениями, реализующими одну и ту же логику, производится на основе наименьшей сложности вычислений, которые необходимо произвести для установления общезначимости (доказательства) утверждения. Поэтому становится важным построение таких исчислений, которые обладают приемлемой для практических целей сложностью. Другой проблемой, довольно тесно связанной с первой, является задача представления данных в понятной для человека форме, одним из путей решения которой служит моделирование самой человеческой деятельности автоматом.

Эта монография посвящена рассмотрению вопроса об автоматическом поиске доказательства в классической логике предикатов и интуиционистской логике высказываний, основанных на системе так называемого натурального (естественного) вывода. Еще в 1992 году авторы поставили перед собой задачу осуществить поиск соответствующего алгоритма вывода и доказательства (как линейной последовательности утверждений) для классической логики высказываний, а также осуществить компьютерную реализацию данного алгоритма. Работа осуществлялась сотрудниками кафедры логики философского факультета Московского государственного университета им. М.В. Ломоносова и полностью была завершена в 1997 году. Далее эта программа была расширена на классическую логику предикатов первого порядка и интуиционистскую логику высказываний.

Развитие современных систем автоматического поиска доказательства имело несколько предпосылок: во-первых, – это создание счетных вычислительных машин (калькуляторов), которые могли осуществлять арифметические операции, во-вторых, – это арифметизация логических процедур, что позволило представить логические

операции как операции над числами, в-третьих, – создание теории алгоритмов и языков программирования, что дало возможность, с одной стороны, разрабатывать алгоритмы автоматического поиска доказательства, а с другой – осуществлять компьютеризацию этих алгоритмов, в-четвертых, – строгая формализация логики и создание логических исчислений, для которых как раз и пишутся соответствующие алгоритмы и компьютерные программы, в-пятых, – создание быстрых электронных вычислительных машин. Взаимное развитие этих направлений имело фундаментальное значение не только для становления новой области исследований – искусственного интеллекта, но и для развития всей человеческой цивилизации.

Остановимся теперь кратко на указанных моментах.

1. Исторические предпосылки и современное состояние дел

1.1. *Современная логика*

Считается, что история современной математической логики берет начало в XVII столетии в трудах Г.В. Лейбница. Последний предложил создать некий формальный универсальный язык "lingua characteristica", в котором можно было бы сформулировать любое утверждение, а также построить для него исчисление – "calculus ratiocinator". Если бы удалось создать такое исчисление, то оно могло бы механизировано (технически) решать вопрос об истинности того или иного утверждения. Это стало бы, по Г.В. Лейбницу, освобождением человеческого разума от его собственных представлений о вещах и его исправлением. Эту точку зрения поясняло описание двоих мудрецов, которые для проверки, кто из них прав, переводят свои аргументы на "lingua characteristica" и потом, не споря друг с другом, говорят: "Calculemus" – подсчитаем.

Не останавливаясь на алгебраизации логики, которая была осуществлена в работах Дж. Буля, А. де Моргана и их последователей, укажем, что следующий важнейший вклад в развитие логики и логических систем был сделан Г. Фреге в его работе "Begriffschrift" ("Запись в понятиях" (1879)). В отличие от Дж. Буля Г. Фреге не пытался представить некую абстрактную логику в виде алгебраических формул, а попытался выразить точным образом посредством специальной символики то самое содержание, которое выражается словами в естественном языке. В указанной работе Фреге принимает пропозициональную интерпретацию для булевой алгебры как фундаментальную и развивает достаточно полно ту часть математической логики, которую мы теперь называем исчислением предикатов. Здесь уже встречается употребление кванторов и общее функциональное построение логики. Идеи, заложенные в этом исчислении, Фреге раз-

вил в работе "Основания арифметики" (1884), а в 1893 и 1903 годах выпустил двухтомные "Основные законы арифметики", в которых формализовались арифметика и теория действительных чисел. Эта работа была продолжена и завершена Уайтхедом и Расселом, издавшими в 1910-1913 годах свое фундаментальное исследование "Principia mathematica", содержащее систематическое изложение современной логики и ее применение для обоснования всей математики. Тем самым программа Г.В. Лейбница – свести логическое мышление человека к чисто синтаксическим процедурам – была реализована в важнейшей своей части.

Для нашей темы следующим важным этапом в развитии логической техники явилось создание в начале 30-х годов XX столетия особых (не аксиоматических) форм исчислений. Это работы Г. Генцена, который развил секвенциальное представление логики, а также построил так называемое натуральное исчисление. В этих двух исчислениях процедура вывода была представлена в древесной форме. Вывод строился как некоторое дерево формул или дерево секвенций. В это же время польский логик С. Яськовский публикует свою известную работу, также посвященную натуральному (естественному) выводу. Системы натурального вывода создавались с целью имитировать (насколько это возможно) рассуждения, которые характерны для человеческого мышления.

С тех пор было построено большое количество различных систем натурального вывода. Для нас важно различать натуральные исчисления двух видов. В первом натуральном исчислении вывод строится как дерево формул. Такие натуральные исчисления будем называть *исчислениями генценовского типа*. В натуральных исчислениях второго типа вывод строится как линейная последовательность формул. Такого типа натуральные исчисления будем называть *субординатными исчислениями*.

Следующим важным различением является наличие или отсутствие в системе непрямых правил вывода, т.е. правила формулируются не как разрешенные переходы от формул к формулам, а как переходы от одной выводимости к другой выводимости. Обычно в качестве непрямого правила используется теорема дедукции, а также некоторые другие правила: например, Г. Генцен предложил следующее непрямое правило $\vee$и:

$$\frac{A \vdash C;\ B \vdash C}{A \vee B \vdash C.}$$

В частности, одним из важных моментов натуральных исчислений является наличие прямого (Куайн [1950], Бочаров и Маркин

[1994]) или непрямого (Генцен [1967]) правила удаления квантора существования. Так, в оригинальной системе, предложенной Г. Генценом, имеется непрямое правило удаления ∃:

$$\frac{A(a) \vdash C}{\exists x A(x) \vdash C,}$$

где a – новая, ранее не встречавшаяся в выводе константа. Таким образом, чтобы получить по такому правилу формулу C из формулы ∃xA(x), необходимо построить вспомогательный вывод (он называется подвыводом) формулы C из формулы A(a). Только после построения такого подвывода можно утверждать, что из формулы ∃xA(x) следует формула C.

По-видимому, первым, кто обратил внимание на неудобство применения этого правила и кто предложил альтернативный вариант удаления ∃, был У. Куайн [1950]. В работе "On natural deduction" он сформулировал прямое правило удаления ∃: из формулы ∃xA(x) следует формула A(y), где y – переменная, которая в алфавитном порядке не предшествует ни одной переменной, свободно входящей в ∃xA(x). Как видим, формулировка данного правила предполагает наличие некоторого упорядочения всех переменных из алфавита языка классической логики предикатов. Фактически речь идет о линейном порядке, заданном на множестве переменных языка. Далее У. Куайн показывает, что именно такой порядок гарантирует непротиворечивость и полноту его системы натурального вывода.

Однако подход У. Куайна не является единственным. Например, В.А. Смирнов в системе NεC (классическое натуральное исчисление с ε-термами) предлагает следующую формулировку прямого правила удаления ∃: из формулы ∃xA(x) следует формула A(εxA(x)), где εxA(x) – это ε-терм, содержательно трактующийся как неопределенная дескрипция вида: "некий x, обладающий свойством A" (Смирнов [2000]). Поскольку ε-термы не являются термами классической логики предикатов, то система NεC не эквивалентна системе NC (системе классического натурального исчисления с непрямым правилом удаления ∃, предложенной В.А. Смирновым там же): все, что доказуемо в NC, доказуемо в NεC, но обратное неверно. Однако для NεC можно доказать теорему Гильберта об устранении ε-термов: если в NεC можно из (возможно, пустого) множества посылок Г вывести A и Г, а также A, не содержат ε-термы, то A следует из Г в NC (Смирнов [2000, с. 228], Мендельсон [1976, с. 111-112]).

Важнейшей характеристикой систем натурального вывода с прямым правилом удаления ∃ является тот факт, что в общем случае за-

ключение такого натурального вывода логически не следует из посылок этого вывода. В таких системах появляется наряду с понятием вывода понятие *завершенного* вывода, т.е. вывода, в котором заключение логически следует из посылок этого вывода.

В.А. Бочаров и В.И. Маркин, вслед за Е.К. Войшвилло, предлагают вводить понятие абсолютно ограниченных и относительно ограниченных переменных в выводе. Тогда правило удаления ∃ запишется следующим образом: из формулы $\exists x A(x, z_1,\ldots, z_n)$ следует формула $A(y, z_1,\ldots, z_n)$, где y абсолютно ограничена и $z_1,\ldots, z_n$ относительно ограничены ($z_1,\ldots, z_n$ суть все свободные переменные из $\exists x A(x)$). При этом требуется, чтобы ни одна переменная не была абсолютно ограничена в выводе более одного раза и чтобы ни одна переменная не ограничивала сама себя. В третьей главе будет показано, что такой подход гарантирует непротиворечивость и полноту этой системы.

1.2. *Алгоритмизация вывода*

Что касается проблемы формализации и алгоритмизации процедур доказательства, то здесь, несомненно, ключевая роль принадлежала работам Т.А. Сколема, в которых он развивал метод, позволявший определять выполнимость любого данного множества формул. В двух работах, которые появились в 1920 и 1928 годах, были введены элиминация кванторов и конструкция, известная как эрбрановский универсум (универсум объектов).

В том же 1928 году вышла книга Д. Гильберта и В. Аккермана "Основы теоретической логики". В ней были поставлены две проблемы относительно аксиоматической формализации исчисления предикатов. Во-первых, встал вопрос о полноте этой формализации, т.е. вопрос о том, каждое ли значимое утверждение следует из этой аксиоматизации. Во-вторых, вводилась проблема разрешимости, т.е. проблема существования алгоритма, который для каждого данного утверждения давал бы решение, является данное утверждение истинным или ложным.

Семантическую полноту исчисления, построенного Д. Гильбертом и В. Аккерманом, в 1930 г. доказал К. Гедель в своей докторской работе. А в 1931 г. он же показал, что достаточно богатые системы не являются полными, и дал конструкцию, по которой для каждой достаточно сильной системы можно построить предложение, которое внутри этой системы недоказуемо и неопровержимо.

Немного позже А.М. Тьюрингом и А. Черчем была показана неразрешимость первопорядкового исчисления предикатов. В 1936 году А.М. Тьюринг свел проблему разрешимости к проблеме остановки некоторой абстрактной машины и показал, что вопрос, остано-

вится ли такая машина, неразрешим. В том же году А. Черч показал для своего λ-исчисления, что значение λ-выражения в общем случае неразрешимо.

Однако эти "негативные" результаты не повлияли на надежду представить алгоритмически вывод математических утверждений. Во-первых, работы Т.А. Сколема показали, что для данного множества истинных высказываний можно механически найти их доказательство. Во-вторых, в 1930 году Ж. Эрбран доказал, что для истинного математического предложения можно эту истинность показать. Такая полуразрешимость исчисления предикатов является формальным оправданием всех систем вывода: если утверждение истинно, то можно в конечное число шагов это доказать, если нет, то существуют две возможности – или удается показать ложность утверждения, или программа работает дальше бесконечно долго.

1.3. *Логические машины*

Зачинателем строительства вычислительных логических машин (именно логических, т.е. работающих с текстами, а не цифрами) считается известный средневековый мыслитель Р. Луллий (1234/35-1315), которого сильно увлекла идея создания такого рода машины. Эту идею он называл "Великим и Окончательным Искусством". Основанием возможности построения такой машины для Р. Луллия было бытовавшее в его время представление о науке. Считалось, что во всякой области знания имеется небольшое число исходных понятий, с помощью которых выражаются бесспорные, самоочевидные положения, не нуждающиеся в обосновании. Из комбинаций этих понятий и представлений и возникает знание. В обладании этими комбинациями и тем, что из них вытекает, и заключается подлинная мудрость. Таким образом, логическая машина должна была служить инструментом для порождения таких комбинаций. Построенные Р. Луллием многочисленные модели такой машины все же не заменяли деятельность человека. Человек был необходим для интерпретации понятийных комбинаций и получения, таким образом, окончательного знания. В своих машинах Р. Луллию удалось реализовать одну из важнейших функций вычислительных машин – перебор вариантов.

Идея Р. Луллия захватила многих выдающихся мыслителей. Б. Паскаль в 1641 году построил свою первую механическую счетную машину, а в 1673 году Г.В. Лейбницем была сконструирована механическая счетная машина, которая выполняла умножение, деление, сложение и вычитание. В 1834 году Ч. Беббидж сконструировал цифровое счетное устройство. Замечательной особенностью этой машины было то, что она могла выполнять инструкции, считываемые с перфокарт. В 1842 году А. Лавлейс (дочь лорда Байрона) сделала описание работы анали-

тической машины Бэббиджа и составила первую программу для нее. В дальнейшем стали создаваться машины, совершенствующие модель Бэббиджа. В 1869 году У.С. Джевонс, использовав результаты Дж. Буля, построил усовершенствованную модель машины. А в 1896 году Г. Холлерит создал первую электромеханическую вычислительную машину и основал фирму Tabulating Machine Company, которая впоследствии превратилась в корпорацию IBM.

Короткий промежуток времени между двумя мировыми войнами считается сегодня периодом рождения первого компьютера. В 1927 году в Массачусетском Технологическом Институте (MIT) был изобретен аналоговый компьютер. В 1937 году Дж. Стибитц построил первую вычислительную машину на основе двоичной системы счисления в Bell Telephone Laboratories.

В 1938 году немецкий инженер К. Цузе закончил строительство первой своей машины, названной Z1. Это была механическая программируемая цифровая машина. Модель была пробной и в практической работе не использовалась. Ее восстановленная версия хранится в музее Verker und Technik в Берлине. Именно ее в Германии называют первым в мире компьютером. В этом же году Цузе приступил к строительству машины Z2 – первому в мире электромеханическому программируемому компьютеру, который являлся промежуточной моделью. В 1941 году вместе с несколькими друзьями он построил первый в мире электронный программируемый калькулятор Z3. Оригинал Z3 был, к сожалению, разрушен во время войны. Реконструированную модель можно увидеть в Немецком музее в Мюнхене.

В 1942 году в Университете штата Айова (Iowa State University) Дж. Атанасов и К. Берри создают первый в США электронный цифровой компьютер (Atanasoff-Berry Computer – ABC).

В России созданием "умных" машин занимались П.Д. Хрущов (1849-1909) и А.Н. Щукарев (1864-1936). Создание вычислительных машин активно велось и в советское время.

Следующим революционным переворотом в истории вычислительных машин стало появление в 1975 году в США первого серийного персонального компьютера (ПЭВМ). Широкую популярность ПЭВМ получили в 90-е годы в связи с бурным развитием Интернета.

1.4. *Первые системы автоматического вывода*

Появление компьютеров открыло путь для развития механизируемых исчислений. В 50-е годы были созданы две компьютерные программы, которые заложили два основных направления в области автоматического поиска доказательства. Первая программа была создана в 1954 г. в Америке М. Дэвисом в Institute for Advanced Studies на

ламповом компьютере "Johniac". Эта программа реализовала технику разрешимости для арифметики положительных чисел и проводила доказательство, что сумма двух четных чисел является четным числом – первое в истории доказательство математического утверждения, генерированное компьютером.

Вторая программа, которая могла доказывать ряд предложений из "Principia Mathematica" Б. Рассела и Н. Уайтхеда, была создана А. Невелом, Дж. К. Шоу и Г. Саймоном. Эта программа ориентировалась на человеческий способ рассуждений, пытаясь симулировать общие эвристические подходы и психологические моменты мышления. Результаты этой работы были доложены на первой конференции по только зарождавшейся области знаний – искусственному интеллекту – на Dartmouth Conference в 1956 г. Уже на этой конференции разгорелась дискуссия: необходимо ли пытаться реализовывать на вычислительных машинах процедуры вывода, формулируемые математической логикой, или, как это было у А. Невела, Дж. К. Шоу и Г. Саймона, симулировать человеческие эвристики. Эти дебаты привели к решению, которое будет сформулировано позднее М. Минским в работе [1963]: "Программа решения реальных математических проблем должна комбинировать математические рассуждения... с эвристическими рассуждениями Невела, Шоу и Саймона".

В 1958 году Хао Ван разработал первую логикоориентированную программу по автоматическому доказательству, а позднее, в 1959-1963 годах, представил другие версии. Это был большой шаг вперед по сравнению с иными самыми успешными проектами. Его программа могла доказывать около 350 предложений "Principia Mathematica" для чистого исчисления предикатов с равенством.

В 1954 году А. Робинсон предложил рассматривать точки, линии, окружности, которые нужно строить для решения геометрических проблем, как элементы так называемого эрбрановского универсума, и переходить при доказательстве геометрических утверждений к алгебраическим методам. Одной из первых программ, реализовавших эту идею, была программа М. Дэвиса и Х. Патнэм. Она состояла из двух частей: первая генерировала эрбрановский универсум и делала соответствующие подстановки в формулы, вторая оценивала эти формулы. Основная проблема, которая возникла перед авторами, заключалась в наличии несистематичности подстановок. Ее удалось решить в 1960 г. шведскому логику Д. Правицу посредством механизма, названного унификацией. Позже был предложен и унифицирующий алгоритм.

Независимо от этого над доказательствами математических утверждений работала другая группа ученых в национальной лаборатории Аргонны (Argonne National Laboratory) под руководством

Дж. А. Робинсона. В нее входили Д. Карсон, Дж. А. Робинсон и Л. Уос. Они поставили перед собой задачу разработать такой принцип, в котором объединились бы различные методы в одно общее логическое правило. Такое правило было найдено в 1963-1965 годах и опубликовано Дж. А. Робинсоном в 1965 г. в работе "A machine oriented logic, based on the resolution principle". С тех пор метод резолюции прочно вошел в практически любую образовательную программу по логике и информатике.

Однако скоро стало ясно, что сам по себе метод резолюции не дает алгоритма поиска доказательства. Поэтому возникли два больших направления с соответствующими школами, которые проходят сквозь всю историю теории автоматического поиска доказательства. В то время как первое, самое большое по числу представителей, продолжало отстаивать метод резолюции и совершенствовать его, второе видело своей задачей имитировать эвристические методы, применяемые "человеческими" математиками.

К первому направлению принадлежали ученые национальной лаборатории Аргонны под руководством Л. Уоса и группа под руководством Б. Мелтцера в университете Эдинбурга (Великобритания). Они предложили в период с 1965 по середину 70-х годов десятки усовершенствованных стратегий поиска вывода с помощью метода резолюций. Но так как метод резолюций оказался не всегда удобным, были предложены другие правила вывода, например, гиперрезолюция. Исследовались также возможности представления метода резолюций в виде графоориентированной процедуры. Так, основная структура данных как множество дизъюнктов пополнялась дополнительной информацией, например, представлением возможных шагов в виде графа. Этот механизм оказался достаточно эффективным. Аналоги его теперь можно встретить почти во всех современных алгоритмах автоматического поиска доказательства.

Следующим интересным результатом оказался матричный метод, который разрабатывался независимо друг от друга П. Эндрюсом в Карнеги Меллон университете и В. Бибелем в университете Мюнхена. Если предыдущие методы еще сохраняли некоторую аналогию с человеческими рассуждениями в виде пошаговой выводимости, то матричная процедура отказывается от этой логической традиции и является полностью машиноориентированной. Поиск доказательства протекает в определенной структуре данных, так называемой матрице, и найденное доказательство не имеет ничего общего с нашим обычным пониманием этой процедуры.

Представители второго направления критически относились к такой механизации дедуктивных систем, в особенности М. Минский из Массачусетского Технологического Института (Massachusetts Insti-

tute of Technology). Основной аргумент заключался в том, что единственного метода никогда не будет достаточно для того, чтобы описать и реализовать интеллект. Для этой цели нужно использовать много других компонент.

Жесткая конкуренция этих двух направлений оказалась весьма плодотворной как для развития всей сферы искусственного интеллекта, так и для каждого из них в отдельности. Такие системы автоматического поиска доказательства, как разрабатываемый в 60-е годы прошлого века Л.М. Нортоном эвристический прувер для теории групп и результаты специального алгоритма, разрабатываемого на базе "человекоориентированной логики" А. Невинсом в университете Дж. Вашингтона, оказались не достижимыми для резолюционных процедур того времени. Это придало особый импульс второму направлению. Можно также отметить работу группы исследователей университета Техаса, которая развивала специализированные методы доказательства для различных конкретных областей математики.

На сегодняшний день наиболее эффективными остаются системы автоматического поиска доказательства, основанные на единообразном методе резолюций. Их эффективность подкрепляется темпами развития электронной техники. И если вначале относительно сложные теоремы доказывались исключительно специальными процедурами, то в дальнейшем становилось возможным получать их доказательство посредством единой резолюционной системы. Наиболее результативными в этой области являются группы национальной лаборатории Аргонны (Argonne National Laboratory), Остина (Austin), Карлсруе и Кайзерслаутерна (Karlsruhe und Kaiserslautern). Однако в последнее время наметилась тенденция синтеза различных методов в единой программной оболочке, что позволяет сгладить недостатки отдельных методов. Такую стратегию приняли разработчики наиболее успешной системы OTTER лаборатории Аргонны.

1.5. *Современное состояние дел*

Системы автоматического поиска доказательства долгое время были достаточно эзотерической областью знания, однако в последнее время сфера автоматического вывода – одно из наиболее бурно развивающихся направлений компьютерной науки, непосредственно связанных с решением практических задач в таких областях, как верификация компьютерных систем и программного обеспечения (software and hardware verification), тестирование программ (Байн и Клаланд [1995]), логическое программирование, дедуктивные базы данных, создание интеллектуальных информационных систем. В этот ряд можно включить экспертные системы, робототехнику и другие направления.

В области автоматического поиска доказательства постоянно проходят специальные конференции – международная конференция по автоматическому выводу CADE (International Conference on Automated Deduction), международная конференция TABLEAUX (Automated Reasoning with Analytic Tableaux and Related Methods) и другие. Практически на любой конференции по искусственному интеллекту имеются особые разделы и секции, называемые системами автоматического поиска доказательства теорем (Automated Theorem Proving) и автоматического обоснования (Automated Reasoning). Имеются и специализированные журналы: Journal of Symbolic Computation, Journal of Automated Reasoning и другие.

Системы доказательства теорем разбиваются на две большие группы. Первая группа – это системы интерактивного доказательства теорем (Proof Checker), называемые также редакторами доказательств. Это системы, которые под своим контролем дают возможность пользователю строить доказательство. В большинстве своем они основываются на теории типов (часто с зависимыми типами) и на системах правил заключения. При этом правила вывода обычно охватывают систему натурального вывода Г. Генцена и зависят от желаемого поля применения. Шаги доказательства проверяются системой на применимость и корректность. Так достигается уверенность, что в системе могут быть построены только корректные доказательства. Примером для редактора доказательств является система AUTOMATH, которая создана для проверки математических доказательств. В этой системе доказывается значительная часть теорем из чистой математики. Однако запись доказательств в AUTOMATHe, как правило, в 10-20 раз длиннее, чем в естественном языке.

Из отечественных работ можно привести программу DEDUCTIO, разработанную А.В. Смирновым и А.Е. Новодворским [1995]. Эта программа позволяет работать не только с одной стандартной логической системой, но и делать выбор из довольно широкого списка известных систем. Программа предоставляет также пользователю возможность описать некоторую собственную систему, после чего она автоматически начинает ее поддерживать.

Построение интерактивного доказательства можно сократить, если объединить многие шаги в так называемые тактики. Для этого предлагаются языки программирования, с помощью которых создаются редакторы доказательств. Такие системы были названы тактическими редакторами доказательств. Таким образом, возникает надежда, что через развитие более сильных тактик, может быть достигнут высокий уровень автоматизации. Система NuPRL является примером такого тактического редактора.

Ко второй группе относятся автоматические генераторы доказательств (Automated Prover). В отличие от редакторов доказательств, в которых пользователь сам строит доказательство, роль пользователя в автоматических генераторах доказательств сводится к постановке задания системе – доказать теорему. Далее система работает до тех пор, пока не будет найдено доказательство или не выполнится какой-нибудь обрывающий работу системы критерий. В качестве такого критерия стараются привести построение контр-модели для доказываемого утверждения.

Разработки программ в области автоматической дедукции проводятся в целом ряде крупных мировых научно-исследовательских центров, таких, как, например: Argonne National Laboratory, США (программа OTTER (Organized Techniques for Theorem Proving and Effective Research), основанная на теории резолюций. Она является последним продуктом в цепи систем (AURA, ITP), которые были созданы под руководством Л. Уоса с момента формулирования принципа резолюций. Программа создана для поиска доказательств для классической логики предикатов первого порядка с равенством. OTTER написана на языке Си (примерно 180 Кб.) и способна оперировать большим количеством дизъюнктов (миллиардами), что показывает ее высокую эффективность), University of Cambridge (программа автоматической дедукции *Isabelle*), Carnegie Mellon University (автоматическое доказательство теорем для логики высоких порядков), KAKEHI Laboratory, WASEDA University, Япония (программа автоматического доказательства для секвенциального исчисления) и др.

В отечественной науке подобные программы начали создаваться также достаточно давно. Известен алгоритм машинного поиска логического вывода, разработанный в 60-е годы прошлого века под руководством Н.А. Шанина (см. Шанин и др. [1964]). Программа, реализованная по этому алгоритму, ищет вывод в классическом секвенциальном исчислении высказываний и перерабатывает его далее в натуральный вывод.

Все рассмотренные процедуры автоматического поиска доказательств обычно строятся либо на базе секвенциальных исчислений и теории резолюций, либо на некотором аппарате, который в той или иной мере близок к данным представлениям логики. Надо отметить, что концентрация внимания исследователей в области автоматического вывода на этих популярных дедуктивных методах, конечно же, не случайна, и обусловлена, в первую очередь, практическими соображениями (Янсен [1990]). Так, например, исследования в области теории резолюций тесно связаны с языками логического программирования (в частности, типа *Пролог*), где выводы основаны на методе

резолюций, который изначально являлся как бы составной частью *Пролога* (Братко [1990], Вишняков и др. [1991]).

Еще одной причиной такой ситуации является факт достаточно успешного практического применения неклассических логик в рамках различных проектов тестирования аппаратных средств (hardware verification) (Бохманн [1982], Янсен [1990]), нередко спонсирующихся ведущими производителями таких средств (например, Intel). Здесь самыми распространенными и эффективными на сегодняшний день являются методы тестирования моделей (Model checking) (Пнуэли [1984]), где для описания моделей hardware используется тот или иной дедуктивный аппарат, чаще всего основанный на модальной или временно́й логике (Бохманн [1982], Болотов и Фишер [1997], Девис и Патнэм [1960], Эмерсон и Кларк [1980], Кларк и Эмерсон [1982]), семантика для которых строится в форме аналитических таблиц.

Похожая ситуация складывается и в области автоматического поиска доказательств для интуиционистской логики. Известны системы, основанные на базе теории резолюций (Таммет [1996], Минц [1985]), аналитических таблицах (Оттен [1997]), матрицах (Оттен и Крайтц [1995a]). Во всех рассмотренных приемах, позволяющих решить вопрос о выводимости произвольных утверждений, традиционная процедура выведения одних положений из других или вообще не представлена, или осуществляется в таком виде, который весьма далек от того, что понималось под выводом в истории логики (Бочаров и Маркин [1994], Смирнов [1972]). И секвенции, и метод резолюций, и другие способы "выведения" – это скорее алгоритмы проверки общезначимости утверждений, чем вывод. Все они строятся на основе чисто аналитических процедур, в то время как традиционное понятие вывода представляет собой метод синтеза доказуемого утверждения из имеющихся посылок.

Несомненно, если проблема (а часто именно так она стоит и формулируется) заключается в том, чтобы установить, выводимо некоторое утверждение или нет, то вопрос о способах ее решения может оказаться второстепенным. Главным оказывается сам факт доказательства наличия выводимости, быстродействие алгоритма, его сложность, требуемые ресурсы памяти и т.д. Однако имеются и такие ситуации, когда отвлечение от способов получения нужного результата недопустимо, и тогда наиболее естественным способом представления вывода являются традиционные способы – аксиоматика и натуральные исчисления.

Содержание предлагаемой читателям монографии как раз и посвящено рассмотрению разработанных нами алгоритмов поиска доказательства теорем в натуральных исчислениях – классической и

интуиционистской логиках. Остановимся на вопросе об алгоритмах автоматического поиска доказательства теорем в натуральных исчислениях более подробно.

1.6. *Автоматизация натуральных исчислений*

Следует отметить, что недостаток внимания, уделяемого натуральному выводу, обусловлен некоторым скептицизмом, порожденным именно фактом специфики этого метода доказательства как синтетической процедуры. Наличие правил введения логических связок часто рассматривается как камень преткновения на пути автоматизации натурального вывода. (Заметим, что в алгоритме, предложенном в нашей работе (Болотов и др. [1998]), эта проблема решена кардинальным путем – правила введения логических связок применяются чисто автоматически, по мере достижения целей, и программа работает в линейном режиме.)

Более того, часто то или иное мнение достаточно авторитетных фигур может оказывать заметное влияние на направление исследований. Так, например, М. Фиттинг в известной и популярной монографии по автоматической дедукции, разбирая различные дедуктивные системы и отдавая предпочтение теории резолюций и семантическим таблицам, а также рассматривая проблему сложности, пишет: "Системы Гильберта не подходят для автоматического доказательства теорем. То же самое относится и к натуральному выводу" (Фиттинг [1990, стр. 95]).

Такое мнение обусловлено тем, что системы натурального вывода наряду с системами гильбертовского типа (аксиоматики) являются именно синтетическими (генерирующими) исчислениями, в которых, исходя из аксиом и посылок, на основе правил вывода строится доказательство для некоторой формулы. Это отличает их от аналитических исчислений (секвенциальные исчисления, исчисления на основе метода резолюций и др.), в которых, наоборот, отталкиваясь от доказываемой формулы, приходят по правилам вывода к аксиомам. Такой метод построения доказательства дает аналитическим исчислениям большие преимущества в отношении автоматизации поиска вывода, так как они жестко ограничены выводимой формулой. В связи с этим понятны пессимистические взгляды на автоматизацию поиска доказательства в рамках синтетических систем. Так, например, В. Бибель отмечает, что для применения некоторого генерирующего исчисления в автоматической дедукции необходимо его превратить в аналитическое исчисление на основе обратимости правил вывода. В случае с исчислениями Фреге-Гильберта это приводит к деревьям поиска вывода с бесконечным числом факторов, порождающих ветвление, тем самым эти ис-

числения оцениваются им как непригодные (Бибель [1992, с. 107]). Относительно натуральных исчислений он высказывает аналогичное мнение, определяя для них роль переводчика найденного доказательства в удобную для пользователя форму и возможного источника идей по улучшению аналитических процедур.

И в самом деле, до 90-х годов прошлого века все программы, использующие системы натурального вывода, не притязали на автоматический поиск доказательства, а были предназначены для интерактивного использования. Однако, как будет показано ниже, удается алгоритмизировать определенные эвристики поиска доказательств и тем самым продемонстрировать, что натуральный вывод тоже может находиться в поле автоматической дедукции.

Итак, системы натурального вывода оказались в определенном смысле позабытыми среди логических инструментов автоматического поиска доказательств. Тем не менее, в конце 1980-х – начале 1990-х годов автоматический поиск вывода в натуральных системах начал исследоваться. В числе такого рода работ перечислим следующие алгоритмы поиска натурального вывода в классической первопорядковой логике:

1)программа Thinker, разработанная Д. Пеллетье (Пеллетье [1998]);
2)программа Oscar, разработанная Дж. Поллоком (Поллок);
3)программа ANDP (Automated Natural Deduction Prover), разработанная Д. Ли (Ли [1997]);
4)программа CMU PT (Carnegie Mellon University Proof Tutor), разработанная У. Сигом [1992] и Дж. Бернсом [1999], а также Сигом и Бернсом [1998];
5)программа Symlog (Symbolic Logic), разработанная Ф. Порторaро [1998];
6)программа Prover, разработанная В.А. Бочаровым, А.Е. Болотовым и А.Е. Горчаковым и модифицированная В.О. Шангиным (Болотов и др. [1996], [1998], Шангин [2000]).

В нижеследующей таблице указываются наиболее характерные черты, наличие или отсутствие которых свойственно данным программам:

1)алгоритм работает в первопорядковой логике с равенством;
2)алгоритм работает с неклассическими логиками;
3)множество используемых алгоритмом переменных ограничено;
4)для алгоритма доказываются теоремы о его свойствах (непротиворечивость, полнота и т.д.);
5)алгоритм использует сколемовские термы;

6)алгоритм имеет программную реализацию.

Значением данной таблицы являются символы: + (признак имеется), – (признак отсутствует) и +– (требует доработки).

	Thinker	Oscar	ANDP	CMU PT	Symlog	Prover
1	+	–	–	–	–	–
2	+	+	+	+	–	+
3	+	–	–	–	–	–
4	–	+–	+–	+	–	+
5	–	+	–	+	+	–
6	+	+	+	+	+	+–

Из перечисленных программ только Thinker работает с формулами, содержащими знак равенства. Соответственно, среди предложенных программ только Thinker может построить вывод для всех 75 формул с равенствами, предложенных в (Пеллетье [1986], [1988]). Данный факт неудивителен, поскольку у Thinker'а и у списка 75 тестовых задач для алгоритма поиска вывода один и тот же автор – Д. Пеллетье.

Безусловным преимуществом для алгоритма является возможность работы не только в классической, но и в неклассических логиках. В нашем случае, программа Thinker умеет работать в модальной логике; программа Oscar – в defeasible логике; программа ANDP – в интуиционистской логике, и, наконец, можно рассматривать работы (Макаров [2002], Шангин [2002]) о поиске вывода в интуиционистской логике высказываний как перенесение идей Prover'а на интуиционистскую логику.

Только программа Thinker допускает ограничение (по умолчанию – 20) на количество переменных, которое можно использовать при построении вывода. По желанию пользователя это количество может меняться. Однако наличие данного ограничения принципиально для поиска вывода. Если множество таких переменных исчерпано (т.е. все n переменных использованы в выводе), то поиск вывода прекращается. В остальных программах поиск вывода потенциально не ограничен. Поэтому иногда возможны ситуации, когда при поиске вывода квантор общности снимается бесконечное количество раз.

Строка 4 (Свойства) отмечает разработанность метатеоретических проблем для данных алгоритмов. Например, "+–" в столбце для Oscar и ANDP означает: доказано, что данные алгоритмы семантически непротиворечивы, т.е. все, что доказуемо, является логически общезначимым; и не доказано, что данные алгоритмы семантически полны, т.е. с помощью данных алгоритмов можно доказать все логически

общезначимые формулы. Дж. Поллок, автор Oscar, предполагает, что его программа полна в указанном смысле, однако доказательства этого факта пока нет. Автором было показано, что Oscar работает в 40 раз эффективнее программы OTTER, основанной на методе резолюций. С другой стороны, круг логических проблем, которые решает Oscar, шире, чем аналогичный круг проблем для OTTER.

По разным причинам, метатеоретическая проблематика отсутствует для программ Thinker и Symlog. Ф. Порториро, автор программы Symlog, выражает свое отношение к этой проблеме следующим образом: "Мы считаем вопрос о полноте алгоритма второстепенным: во-первых, он слишком далеко отстоит от темы нашего исследования, во-вторых, он не отвечает задачам алгоритма. Задача нашего алгоритма – представлять всестороннюю помощь при построении вывода, а не болезненно строить формальные выводы определенного вида. Поэтому вопрос о полноте нашего алгоритма отходит на второй план и вовсе исчезает тогда, когда алгоритм начинает оказывать помощь при работе с другими формальными системами, например, с арифметикой или теорией множеств" (Порториро [1998, с. 59]). Мы считаем, что такой подход к проблеме является недостаточно корректным.

Что касается Thinker'а, то данный алгоритм неполон заведомо, если учесть присущее только ему ограничение на количество используемых в выводе переменных (на это указывает сам Д. Пеллетье): "Очень трудно оценить семантическую полноту программы Thinker потому, что он может прийти к пункту J (остановка алгоритма) в различных случаях; в том числе и тогда, когда формула является логически общезначимой, Thinker может остановиться, потому что кончатся переменные" (Пеллетье [1998, с. 33]). С другой стороны, он пишет, что неясен ответ на вопрос, будет ли Thinker семантически полон, если данное ограничение на количество переменных будет снято.

Теорема о семантической полноте доказана для CMU PT. Предложенный У. Сигом метод (см. Сиг [1992], Сиг и Бернс [1998], Бернс [1999]) заключается в том, что по дереву вывода, которое не является доказательством, можно построить контрмодель для данной формулы или для данной выводимости. Если формула не доказуема, то найдется (возможно, бесконечный) контрпример, показывающий, что при данных значениях исходная формула принимает значение "ложь". Сходным образом доказана семантическая полнота для Prover'а.

Не все алгоритмы используют в своей работе сколемизацию. Под сколемизацией здесь имеется в виду явная сколемизация, т.е. добавление в язык логики предикатов особых свободных переменных и предметных функторов, с помощью которых образуется сколемовский терм. Сколемизация осуществляется в Oscar, CMU PT и Symlog.

Однако поиск вывода не обязательно предполагает явную сколемизацию. Существуют различные формы неявной сколемизации. Например, Thinker предполагает в процессе поиска вывода использовать так называемые "шаблоны" (templates), которые при окончательном построении вывода превращаются в формулы. То есть снятие $\forall$ с формулы $\forall x A(x, y)$ порождает шаблон $A(@, y)$. Prover также использует неявную сколемизацию, о чем подробно говорится далее.

Все алгоритмы имеют программные реализации, доступные в сети Интернет. Имеется пропозициональный фрагмент программной реализации Prover'а. Что касается кванторного фрагмента Prover'а, то он находится в стадии разработки.

Из данного анализа видно, что проблема исследований метатеоретических свойств алгоритмов поиска натурального вывода иногда вообще не ставится учеными. Фундаментальными при доказательстве семантической полноты алгоритма являются работы У. Сига. Именно на них во многом базируется и наше исследование. Однако здесь мы отметим те существенные различия, которые лежат между подходом У. Сига и нашим подходом.

Во-первых, существует различие между системами натурального вывода. В нашем случае это система натурального вывода с прямым правилом удаления $\exists$. (Выше уже говорилось о том, что в таких системах натурального вывода заключение в общем случае не является логическим следствием своих посылок.) У. Сиг, в свою очередь, работает с системой натурального вывода с непрямым правилом удаления $\exists$.

Во-вторых, мы непосредственно ищем вывод в нашей системе, а CMU PT использует для поиска вывода некоторые промежуточные исчисления, которые так и называются "intercalation calculi", т.е. вставочные, промежуточные исчисления. По внешнему виду они напоминают секвенциальные исчисления. Вывод в таких исчислениях затем перерабатывается в вывод в натуральном исчислении. Таким образом, мы вновь сталкиваемся с ситуацией, которая стимулировала развитие исследований именно по поиску натурального вывода: сначала вывод строится в другом исчислении (которое более благоприятно с точки зрения автоматического поиска вывода), а затем вывод в этом исчислении конструктивно перестраивается в натуральный вывод (см. Эндрюс [1980]). Таким образом, натуральный вывод в CMU PT строится не в системе натурального вывода, а с помощью промежуточного исчисления.

Что касается Prover'а, то программа строит вывод именно в системе натурального вывода. Спецификой Prover'а является поиск вывода в субординатных натуральных исчислениях типа Куайна с использованием абсолютно и относительно ограниченных переменных. Во-

прос семантической полноты для Prover'а будет рассмотрен в положительном смысле в данной монографии. Оригинальность данной работы состоит в том, что здесь вывод строится как традиционная синтетическая процедура по автоматическому поиску теорем. Таким образом, данная работа заполняет своеобразную лакуну, которая образовалась в области автоматического поиска натурального вывода.

1.7. *Место натурального вывода в автоматизации доказательств*

Говоря об алгоритмах поиска доказательств теорем в натуральных системах, следует отметить и следующий немаловажный факт. Одним из главных критериев оценки эффективности того или иного дедуктивного аппарата (с практической точки зрения) является быстродействие программы, основанной на этом аппарате. Не вдаваясь в детальный анализ этой проблемы, описание которой содержится во многих работах и, в частности, в (Бохманн [1982], Фиттинг [1990]), отметим лишь, что и метод резолюций, и метод секвенций, впрочем, как и все другие допустимые дедуктивные средства, обладают различной скоростью работы на разных формулах: известны типы примеров, где одни методы работают быстрее других и в принципе нет такой системы, которая бы превосходила все другие (см. Фиттинг [1990, стр. 95]). Так, например, теория резолюций (в отношении классической пропозициональной логики) является довольно эффективным методом. Одним из наиболее эффективных методов является так называемая теория линейной резолюции, применяемая по отношению к специальному типу нормальной формы, требуемой для резолюций, когда члены КНФ имеют вид дизъюнктов Хорна (где разрешается не более одного позитивного вхождения какой-то переменной) (Братко [1990]). Существуют стратегии поиска доказательства (комбинация которых является полной по отношению к множеству дизъюнктов Хорна), представляющие собой очень эффективное средство решения логических задач (Бибель [1992], Девис и Патнэм [1960]). Однако очевидно, что далеко не всякое множество формул представимо в виде конъюнкции дизъюнктов Хорна. Так, первый член в следующем множестве не является дизъюнктом Хорна:

$$(p \vee q) \,\&\, (\neg p \vee q) \,\&\, (p \vee \neg q) \,\&\, (\neg p \vee \neg q)$$

и, следовательно, в принципе, никакой стратегии, кроме стандартного перебора всех возможных вариантов применения правила резолюций, здесь применить не удается. Заметим, что в нашем случае натурального вывода процесс его построения – в трудном для теории резолюций случае – является строго детерминированным. Другой тип примеров, сложных для теории резолюций, – это формулы, нормаль-

ные формы которых "взрываются" в аспекте их длины, как, например, в случае

$$((p \equiv q) \,\&\, (q \equiv r)) \equiv (p \equiv r),$$

или в еще более сложном примере

$$(p \equiv (q \equiv r)) \equiv (p \equiv q).$$

Заметим, что и здесь для натурального вывода доказательство подобных формул не проблема, именно в силу следующей кардинальной особенности алгоритма. Сложность алгоритма в случае натурального вывода зависит не от длины формулы, подлежащей доказательству (что является, в совокупности с проблемой порождения "нерелевантных" для доказательства или просто лишних резольвентов, основными моментами, влияющими на сложность метода резолюций) или количества различных переменных в анализируемой формуле (что вызывает экспоненциальный рост таблиц истинности), а структурная сложность формулы.

Подобные примеры служат свидетельством известного факта, что наиболее эффективной является в общем случае простая процедура построения истинностных таблиц. Упомянем здесь работу Д'Агостино и Мондадори [1994], где авторы, модифицируя определенным образом технику аналитических таблиц, доказывают весьма существенный результат, что полученная ими дедуктивная теория может "имитировать" сложность таблиц истинности. Анализируя методику, предложенную в этой работе, можно найти много аналогий и идейного сходства между содержащимися здесь формулировками правил построения таблиц и правилами натурального вывода. В заключение заметим, что существенной является задача определения сложности и эффективности нашего алгоритма, требующая особого исследования; первые же результаты тестирования нашего алгоритма даже на довольно слабых, с современной точки зрения компьютерах, показали достаточно хорошие результаты.

Весьма любопытным в свете вышеизложенного является следующее направление, свойственное современным исследованиям в области автоматической дедукции. Речь идет о попытке симбиоза различных методов, включая теорию резолюций, секвенций, элементы натурального вывода и другие. Это частично характерно для центра исследований OTTER, но наиболее четко проявляется в подходе группы по разработке проекта *Isabelle* (University of Cambridge) [1996], где авторы прямо заявляют о такой стратегии симбиоза. На наш взгляд, этот подход является, во-первых, достаточно обоснованным (в силу специфик и ограниченности каждого из методов дедукции), и во-вторых, много-

обещающим (учитывая не только теоретические моменты, но также и практические возможности применения техники параллельных процессоров). В связи с этим наше исследование в области эвристик для натурального вывода видится весьма перспективным и способным внести свою лепту в решение проблемы эффективной интеграции различных методов автоматической дедукции.

В настоящее время у авторов имеется компьютерная реализация поиска вывода и доказательства для пропозициональной части логики предикатов.

В качестве справочной литературы мы можем порекомендовать следующие работы: Бибель [1998], Бачмейр и Ганзингер [2001].

2. Интеллектуальные системы

2.1. *Практическая сторона проблемы (искусственный интеллект)*

Анализ современного состояния исследований в области автоматической дедукции будет невозможным, если не проанализировать тенденции, заложенные при рождении компьютеристики и искусственного интеллекта как одного из ее разделов. Термин "искусственный интеллект" был предложен Дж. Маккарти на известной Dartmouth Conference в 1956 г., которая считается днем рождения этого раздела современного знания. Конференция собрала ведущих ученых с мировым именем – математиков, логиков, специалистов в области теории автоматов, нейронных сетей, изучения интеллекта. Такие имена, как Марвин Минский, Артур Самуэль, Герберт Саймон, Клод Шеннон, Аллен Ньюэл, говорят сами за себя. К этому времени уже зарекомендовали себя первые программы автоматического доказательства математических теорем, указанные выше. Варрен Маккуллон и Вальтер Питтс в 1943 г. предложили модель искусственного нейрона, показав, что любая вычислимая функция может быть вычислена некоторой сетью нейронов. Первый простой алгоритм обучения нейронных сетей появился в 1949 г. благодаря Дональду Гербу. В 1951 г. Марвин Минский и Дин Эдмонд построили первый нейронный компьютер SNARC. Таким образом, были предложены подходы к автоматизации таких фундаментальных компонентов интеллектуальной деятельности, как дедуктивное рассуждение, обучение и самообучение, что привело к дискуссии о путях объединения усилий ученых из различных областей, связанных с человеческим интеллектом. Рабочее название "искусственный интеллект" прижилось для этого нового междисциплинарного научного направления.

Трудно найти вторую такую научную дисциплину, с более противоречивой историей. С момента своего рождения искусственный

интеллект пережил период энтузиазма и больших ожиданий. Так, например, Герберт Саймон в 1958 г. предсказывал, в частности, что в течение следующих 10 лет шахматный компьютер победит сильнейшего в мире шахматиста и что с помощью компьютера будет доказана новая важная теорема математики. В истории искусственного интеллекта мы находим также массу грандиозных и амбициозных проектов, финансируемых крупнейшими компаниями и государственными агентствами, и даже целыми государствами, как это произошло, например, с так называемой программой создания машин пятого поколения, которая была спонсируема правительством Японии.

В то же время в истории искусственного интеллекта мы видим грандиозное крушений надежд, серьезную критику с последующим свертыванием финансирования исследований почти до нулевого уровня, как это и произошло с упомянутой программой пятого поколения или с исследованиями в Великобритании, где финансирование искусственного интеллекта было значительно сокращено после опубликования в 1973 г. так называемого Lighthill report'a, который содержал критику фундаментальных исследований, проводимых в рамках искусственного интеллекта. Несомненно, такая история связана, в первую очередь, с тем, что основная цель данного раздела компьютеристики – создание искусственного интеллекта – явилась, наверное, наиболее амбициозной, по сравнению с другими научными направлениями. Заметим, кстати, что и сам термин "искусственный интеллект" подвергался всевозможный критике, что во многом определило переименование большого количества университетских кафедр и научных лабораторий (в названии которых ранее фигурировал этот термин). Так, например, стало более популярным употреблять термин "интеллектуальные системы", или "информационные системы", хотя, конечно же, от такого переименования суть дела не меняется – разве что отсутствие термина "искусственный" не вызывает явного недовольства тех, кто связывает со словосочетанием "искусственный интеллект" не более чем выброс денег на ветер.

Мы будем использовать термин "искусственный интеллект", понимая под ним теорию создания компьютерных систем, способных вести деятельность, обычно ассоциируемую с человеческим интеллектом. Заметим, что "интеллект" трактуется здесь в широком смысле слова, как обобщающий различные типы *человеческой активности*, на разных уровнях отражения действительности – от чисто реактивной деятельности до уровня абстрактного мышления (см. Рассел и Норвиг [2003]). Затронув проблему активности, мы с неизбежностью приходим к теме "интеллектуальных агентов".

2.2. *Интеллектуальные агенты*

Под термином "интеллектуальный агент" или "рациональный агент" (от английского "agent" – активный объект) принято понимать такую компьютерную систему, деятельность которой удовлетворяет принципам рациональности и автономии и которая способна функционировать в сообществе аналогичных программ (см. Вулдридж [1999]). Предполагается, что интеллектуальный агент создается для реализации определенных целей (например, для мониторинга работы многопроцессорной системы). Рациональность агента означает, что его активность направлена на достижение этих целей. Под автономией понимается способность системы самой, т.е. без вмешательства человека, выбирать ту или иную линию поведения. Способность системы функционировать в сообществе аналогичных программ связана с тем, что большинство практических задач, для решения которых предназначены данные системы, в силу своей сложности предполагают, что над их решением работает сообщество агентов. Следовательно, при создании рациональных агентов предполагается наделять их способностью коммуникации. Таким образом, мы приходим к следующему определению:

> *Определение 2.2.1.* Рациональный агент – это компьютерная система, созданная для достижения некоторых целей, способная автономно действовать в сообществе аналогичных программ и выбирать оптимальную линию поведения, направленную на достижение этих целей.

Не будет преувеличением сказать, что в современной компьютеристике исследование интеллектуальных агентов – одна из самых популярных тем. Явным подтверждением этому является последнее десятилетие, когда чуть ли не каждая научная конференция по компьютерным наукам в своей программе упоминает теорию агентов. Появились специальные тематические конференции и журналы, посвященные проблемам агентов (International Joint Conference on Autonomous Agents and Multi-Agents Systems, International Workshop on Agent-Based Simulation, From Agent Theory to Agent Implementation, International Workshop on Engineering Societies in the Agents, World International Conference on Intelligent Agent Technology, International Symposium on Intelligent Data Analysis, Workshop on Agent-Oriented Information Systems). Образовалось множество научно-исследовательских центров. Крупнейшие из них работают в Массачусетском Технологическом Институте (http://agents.media.mit.edu/-ndex.html), Карнеги Меллон Университете (http://www-2.cs.cmu.-u/~softagents), Ливерпульском Университете (http://www.csc.liv.ac.-

k/research/agents), в Немецком Исследовательском Центре Искусственного Интеллекта (http://www.dfki.de/mas). Появились стабильно функционирующие координационные центры, такие, например, как AgentLink (http://www.agentlink.org), аккумулирующие теоретическую и практическую информацию по интеллектуальным агентам.

Все больше отечественных и зарубежных исследователей из различных областей компьютеристики и вне ее обращаются к данной теме. Так, например, мы находим идеи применения агентов в теории безопасности компьютерных сетей (см. Городецкий и Котенко [2002]). Конечно, не обошли стороной теорию агентов и робототехника (см. Тимофеев [2000]), и теория связи (см. Чекинов [2001]).

Среди областей, смежных с компьютеристикой или использующих ее методы и достижения, мы находим идеи применения агентов в образовании, например, при создании системы дистанционного образования на базе мультиагентных технологий (см. Кудинов [2003]). Методы применения агентов распространяются на экономические теории планирования (см. Городецкий [1997]). Но, пожалуй, наиболее привлекательной областью для реализации агентов является Интернет. Здесь, как кажется, имеются неисчерпаемые ресурсы и для любителей, и для профессионалов. Уже сейчас Интернет насыщен так называемыми "ботами", выполняющими роль мобильных агентов, т.е. программ, которые "путешествуют" по Интернету, выполняя некоторые заданные функции. Заметим, кстати, что первыми агентами явились именно мобильные агенты. Например, ботами, которые *претендуют* на информационные фильтры (фильтры новостей), снабжаются Интернетовские сайты. Любителям предоставляется широкое поле деятельности по видоизменению ботов, так как доступными становятся программы-источники. В последнее время мы наблюдаем настоящий бум в области создания так называемых поисковых агентов, предназначенных для эффективизации поиска информации в Интернете. Этот список областей, где термин "агент" фигурирует в том или ином виде, может быть значительно расширен (см. Вулдридж [1999], Городецкий [1999]).

Отсылая заинтересованного читателя к указанным литературным источникам, мы остановимся только еще на одной области, связанной с агентами, – дистрибутивной компьютеристике. Дистрибутивная компьютеристика, или дистрибутивный искусственный интеллект, это область знания, заинтересованная в методах анализа и переработки информации, распределенной среди множества ресурсов. Идея распределенной обработки информации часто связывается с появлением первых супер-ЭВМ – от советской ЭВМ БЭСМ-6 и американской ILLIAC-IV, машин так называемого второго поколения, до последних суперкомпьютеров. Кстати, и сам Интернет в том виде, в ка-

ком мы его знаем сейчас, – это развитие идей дистрибутивной компьютеристики, результат одного из проектов, разрабатывавшихся в 1989 г. в CERN (Европейской Организации Ядерных Исследований).

Говоря столь подробно об агентах, мы защищаем тезис, согласно которому интеллектуальные агенты – это современный этап развития искусственного интеллекта: наиболее серьезные подходы в этой области являются продолжениями главных концепций искусственного интеллекта, и проблемы, с которыми мы сталкиваемся при анализе агентов и их реализаций, по сути своей, – это традиционные проблемы искусственного интеллекта. Тем не менее, несмотря на обилие исследовательских проектов, реальные воплощения интеллектуальных агентов только сейчас начали появляться на свет. Здесь мы рассмотрим три примера работающих компьютерных систем, обладающих основными характеристиками интеллектуальных агентов. Примеры взяты из различных областей.

Астрономия. Недавно английский телескоп на Гавайях был снабжен системой интеллектуальных агентов, созданных по проекту "Электронные телескопы для астрономических исследований" (eSTAR). Телескоп предназначен для изучения критических явлений во Вселенной, таких как массивные взрывы и астероиды, которые могут представлять потенциальную опасность столкновения с Землей. Агенты, работающие в компьютерном обеспечении телескопа, объединены в систему, ответственную за изменчивость в поступающих данных. Например, они фиксируют звезды, ставшие ярче, или астероид, видимый телескопом, информируя об этих явлениях ученых. При этом система, фиксируя некоторое явление, продолжает наблюдение и сбор информации до тех пор, пока она не решила, что достигла нужного уровня, и тогда данные обозрения передаются ученым. Более того, создателями этого программного обеспечения предусмотрена возможность для астрономов конструировать своих собственных агентов, снабжая их определенными целями, что обеспечивает эффективное распределение полученной информации: теперь ученый, занимающийся Солнечной Системой, с помощью своего агента, будет непосредственно информирован только о тех событиях, которые его интересуют.

Интернетовская компьютеристика. В свою очередь, IBM предлагает агента, названного Web Browser Intelligence, предназначенного для более эффективного получения, распределения и контроля интернетовской информации (http://www.networkinh.ibm.con/iag/iaghome.html.). Это программное обеспечение представляет собой так называемого персонализированного агента, запоминающего особенности работы пользователя с интернетовскими страницами. Агент может помочь вызвать страницы, ранее просматривавшиеся пользо-

вателем, организовать поиск в этих страницах по ключевым словам. Заметим, что, как указывают разработчики, главной интеллектуальной особенностью агента является его способность изучения манеры пользования Интернетом.

Робототехника. Разработки так называемых реактивных роботов, симулирующих различные моторные аспекты гуманоидной активности, – одно из выдающихся достижений упомянутой выше лаборатории Брукса в Массачусетском Технологическом Институте (см. Брукс [1985], [1986], [1991]). В Карнеги Меллон Университете создан робот, в задачи которого входит сопровождение пациентов на консультации с врачами-специалистами в больницах. Робот снабжен техникой машинного обучения, что позволяет ему ориентироваться в помещении госпиталя, находить нужную палату, находить пациента и сопровождать его на консультацию врача.

Наконец, назовем ежегодные футбольные турниры команд роботов – robocup (см. сборник статей под редакцией Ханнебауэра, Вендлера и Пагелло [2001]). Эти турниры являются эффективным средством тестирования современной экспериментальной технологии, применяемой в робототехнике. В центре внимания коллективов, моделирующих футбольные команды роботов, – проблема конструирования автономной мультиагентной системы, играющей в футбол без вмешательства человека (см. Стоун [2002]). Конечно, параллельно с решением данной проблемы сама конструкция роботов представляет собой огромное поле для исследований – ведь симуляция моторных аспектов деятельности человека, таких, как, например, балансирование при ходьбе или беге, – это очень сложная проблема. Так, первый турнир команд роботов-гуманоидов прошел только совсем недавно, в 2003 г. в Токио. Стоит отметить, что в основе некоторых таких мультиагентных систем лежит идея так называемой BDI архитектуры (от английских Belief – убеждение, Desire – стремление и Intension – намерение), что вплотную связывает разработки в этой области с автоматической дедукцией, как это будет показано в нашем обзоре ниже.

Сам термин "агент" начал активно использоваться в лексиконе проблематики искусственного интеллекта после опубликования известной книги М. Минского "Сообщество мышления" (The Society of Mind [1988]). В этом труде, который сам автор рассматривает как своего рода популярное введение в предмет теории мышления, доступно объясняется понимание мышления как сложного агента, деятельность которого – это функция множества более или менее простых агентов. Например, на анализе, казалось бы такого простого акта, как поднятие рукой кружки чая, Минский показывает, что на самом деле здесь мы видим координацию и кооперацию мини-

сообщества агентов: ответственных за взятие чашки (grasping agent), ответственных за сбалансирование чашки (balancing agents), ответственных за утоление жажды (thirsty agents), ответственных за поднесение чашки к губам (moving agents). Автор утверждает, что интеллектуальная активность человека – это результат кооперации, координации и коммуникации сообщества агентов, каждый из которых обладает определенной независимостью.

Заметим, что существует прямая параллель между этим подходом к мышлению и одной из самых плодотворных архитектур робототехники – так называемой архитектуры Р. Брукса [1985], в основе которой лежит идея о сложном поведении системы как функции более простых актов деятельности. Подчеркнем также, что указанная работа Минского и успехи в области распределенной обработки информации оказали существенное влияние на развитие исследований мультиагентных систем. Отметим некоторые "канонические" работы по теории агентов: Вулдридж [1999], Джорджеф, Пэлл, Поллак, Тамбе и Вулдридж [1998], Рао [1998], в которых введены основные понятия теории интеллектуальных агентов, приводится классификация агентов, дается их общая архитектура и объясняются различные подходы к построению агентов.

2.3. *Абстрактная архитектура интеллектуальной системы*

В научных исследованиях по искусственному интеллекту в большинстве случаев авторы хотя бы коротко указывают на возможные приложения разрабатываемых проблем или полученных результатов. Как правило, такие ссылки на практическое (потенциальное) применение теоретической работы связаны с идеей обоснования, мотивации и демонстрации ее важности. Часто авторы прибегают к следующему довольно эффективному методу. Создается некоторая гипотетическая реальность и значимость работы показывается именно в контексте такой гипотетической реальности.

Итак, в нашей гипотетической реальности предполагается, что с интеллектуальной системой S связана некоторая конкретная задача, и конечной целью является автоматическое продуцирование некоторого множества (рациональных) действий $A = \{a_1, \ldots, a_n\}$. Система определена в некоторой окружающей среде E, воспринимая от среды сигналы (информацию). Предполагается, что S находится в одном из возможных миров, w_i из множества (в общем случае бесконечного) возможных состояний системы $W = \{w_0, \ldots, w_k \ldots\}$. Результаты восприятия сигнала проецируются на модель данного состояния, w_i, модель преобразуется соответствующим образом, и система переходит в новое состояние (новый возможный мир) w_{i+1}, порождая соответствующее действие из множества A. Ожидается, что порожденное

действие является рациональным, т.е. соответствует целям, поставленным перед системой, и окружающая среда и система вступают в очередной цикл взаимодействия.

Рассмотрим только один реальный пример – шахматную компьютерную программу, такую, как Deep Fritz, различные версии которой играли в 2002 г. в Бахрейне против Владимира Крамника и в 2003 г. в Нью-Йорке против Гарри Каспарова (оба матча закончились с ничейным результатом) и которая способна вычислять 3,5 миллиона операций в секунду. Текущая позиция на шахматной доске – это окружающая среда для шахматной программы. Ход соперника – сигнал, поступающий из окружающей среды, проецируемый на модель состояния системы w_i на данный момент времени. Программа строит новое свое состояние, w_{i+1}, и ищет свой ответ на ход соперника.

Общая модель интеллектуальной системы, конечно, реализуется поразному, в зависимости от характеристик ее главных компонентов. Одним из ключевых моментов здесь является отношение изменчивости окружающей среды и времени от момента поступления сигнала до момента вычисления действия. Чем более изменчива окружающая среда, тем более быстро система должна вырабатывать решения. Здесь мы подходим к кардинальной проблеме, связанной с характером выработки решений в интеллектуальных системах, – проблеме совокупной сложности одного цикла интеллектуальной системы, обрисованного выше. Во-первых, это кодирование сигнала, поступающего из окружающей среды. Во-вторых, это преобразование модели, находящейся в памяти системы. В-третьих, это вычисление соответствующего действия. Вся эта картина становится куда более сложной, если мы включим такие параметры, как, например, планирование: каждое изменение в модели системы будет сказываться на плане (а ведь исследования по автоматическому планированию находятся, скорее всего, только на начальном этапе своего теоретического развития).

2.4. *Подходы к построению интеллектуальной системы*

Мы подошли к проблеме применения автоматической дедукции в конструировании интеллектуальных систем. Второй и третий этапы преобразования информации – это те области, где автоматическая дедукция или уже успешно применяется или является предметом разработок. Формальная модель интеллектуальной системы S – это не что иное, как система пошаговых переходов (transition system), определяемая на множестве миров $W = \{w_0, \ldots, w_k, \ldots\}$. Поскольку система должна начать работу, некоторое данное изначальное состояние системы w_0 соответствует входу в систему. Переход от одного состояния к другому – это шаг системы. Система получает сигналы, поступающие из окружающей среды, и кодирует их. Так, находясь в неко-

тором мире w_i и преобразовав полученный сигнал в некоторую символическую форму e_j, система применяет заранее определенную многозначную функцию f пошагового перехода $f(w_i, e_j)$. Результатом применения этой функции будет некоторое подмножество миров из W. Таким образом, наша система недетерминистическая, и какое именно последующее состояние должно быть ею выбрано, будет зависеть от внешних параметров, таких, например, как планирование. Вполне естественно понимать нашу систему как автомат.

Теперь наступает время вычисления оптимального действия, соответствующего новому состоянию системы. В зависимости от характера этого вычисления в современной литературе выделяют различные типы интеллектуальных агентов (см. Вулдридж [1999]). Мы можем, например, непосредственно связывать состояния системы с некоторым действием и затем вычислять реакцию системы как производную от множества возможных реакций. С другой стороны, реакция системы может вычисляться дедуктивно. Наконец, мы можем рассматривать вычисление действия системы как функцию компонентов целенаправленной деятельности.

В первом случае мы работаем в идеологии так называемого субсимволического подхода в построении интеллектуальных систем. Здесь система конструируется как иерархия модулей, где каждый модуль представляет собой состояние системы, непосредственно ответственное за определенное действие. Получив сигнал, система, основываясь на субординации модулей, вычисляет, какой именно модуль должен быть включен, что детерминирует производимое действие.

Во втором случае мы находимся в рамках идеологии символического, или логического, подхода к построению интеллектуальных систем. Здесь вычисление действия непосредственно связано с применением методов автоматической дедукции: возможные действия из множества $A = \{a_1, \ldots, a_n\}$ оцениваются на факт совместимости или несовместимости с состоянием w_i. Алгоритм, по которому система выбирает возможные действия, будет следующим: для любого действия a_i, если a_i несовместимо с состоянием w_i, тогда a_i отбрасывается, если a_i совместимо с состоянием w_i, тогда a_i выполняется.

В последнем случае конструирование агента основано на анализе компонентов целенаправленной деятельности и их моделировании. Здесь, как мы упоминали, вычисление реакции системы – это функция компонентов целенаправленной деятельности. Такой подход реализуется в BDI моделировании интеллектуальных систем.

2.5. *Логические агенты*

Логические интеллектуальные системы явились первыми реализациями идей искусственного интеллекта. Развитие этих систем было

инициировано задачей автоматического доказательства математических теорем. Так появились первые системы автоматической дедукции, о которых уже шла речь выше. Успех этих систем послужил основной причиной для становления программы символического (классического) искусственного интеллекта: символического описания окружающей среды интеллектуальной системы и желаемого поведения системы с последующим синтаксическим манипулированием полученными формализмами. В случае, когда формализмы выражены непосредственно на некотором логическом языке, мы имеем дело с логическим интеллектуальным агентом, где синтаксическое манипулирование формулами – это не что иное, как автоматизированная логическая дедукция.

Заметим, что создание экспертных систем, являвшихся центральным объектом искусственного интеллекта в течение долгого времени, напрямую связано с развитием логического подхода, так как экспертная система может быть рассмотрена как чисто логический интеллектуальный агент. В основе экспертной системы лежат информационная база Г, относящаяся к некоторой специализированной области, например, медицинская база данных, и совокупность выражений – импликативных правил R, типа "Если А, то В", где А и В – это некоторые (простые или сложные) выражения, такие, что интерпретация их нелогических терминов в данной специализированной области выполняет формулу $A \supset B$. Например, если А и В интерпретируются, соответственно, как "имеется высокая температура и тело покрыто сыпью" и "проверить характер сыпи“, то мы получаем выполнимую импликацию $A \supset B$: "если имеется высокая температура и тело покрыто сыпью, то проверить характер сыпи". Получая новую фактическую информацию из окружающей среды I, экспертная система включает механизм автоматической дедукции, и проверяет наличие отношения логического следования между {Г, I} и антецедентами правил из R. Если наличие такого следования установлено, т.е. антецедент А некоторого правила $A \supset B$ выводим из {Г, I}, то в силу modus ponens получаем, что В выводимо из {Г, I}.

Конечно, здесь мы использовали лишь общую схему того, как работает экспертная система. В реальности формализация информационной базы экспертной системы требует богатых выразительных возможностей логического языка, например, языка логики предикатов первого порядка. Следовательно, процесс установления наличия логического следования будет куда более сложным.

В силу своей специфики логические интеллектуальные агенты непосредственно связаны с результатами автоматической дедукции. Логические методы позволяют проводить анализ свойств интеллектуальных систем на высоком абстрактном уровне, что дает возмож-

ность строить системы, которые могут работать с различными информационными базами, а следовательно, могут быть потенциально использованы в различных областях.

В то же время нельзя не сказать об одном, на наш взгляд, самом значительном ограничении, связанном с конструированием этих агентов, – очевидной проблеме символизации информационной базы системы и получаемого ей сигнала из окружающей среды. Несмотря на то что эта проблема является внешней для автоматической дедукции, она непосредственно сказывается на практических применениях логических интеллектуальных систем. Ведь даже в случае эффективного аппарата автоматической дедукции, такого, как, например, теория резолюций, конструирование действительно практически значимой интеллектуальной системы требует также и *автоматизации символизации знания*, задачи куда более сложной, чем интересующая нас проблема автоматизации дедукции.

2.6. *Конструирование интеллектуальных агентов* (*социология*)

Одной из характерных особенностей сегодняшнего состояния дел в компьютерной науке является *мощная специализация исследований*, даже в рамках, казалось бы, одной, уже довольно узкой, области. Возьмем, например, такую область, как временны*е* логики. С момента появления первых работ по применению временны*х* логик в компьютерной науке (Пнуэли [1977]) прошло немногим более 25 лет. Это означает, что на развитие временно́й логики в этом направлении приходится жизнь только одного поколения исследователей. А ведь в потенциале временно́й логики сегодня целый спектр работ – от достаточно прозрачных (синтаксически и семантически) логик линейного времени до куда более "витиеватых" логик ветвящегося будущего, или еще более сложных логических систем, формализующих понятие неподвижной точки, определенной на некоторой временно́й структуре (см. Стёрлинг и Брэдфилд [2001]). Более того, синтаксические и семантические свойства получаемых систем непосредственно зависят от принятия той или иной временно́й линейной или древесной структуры в качестве "онтологической основы". Далее, для этих систем строятся соответствующие теории вывода, основанные на теории резолюций или теории аналитических таблиц. Не будет большим преувеличением сказать, что в большинстве случаев сложность получаемых исчислений такова, что понимание всех тонкостей и деталей становится возможным только для очень узкого круга специалистов (хотя, конечно, в широком смысле, "суть дела" понятна для логика-профессионала, независимо от его специализации). Более того, дальнейшая разработка темы связана с постановкой еще более

специальных проблем и становится возможной в большинстве случаев при наличии коллектива авторов, свободно ориентирующихся в данной области. А это означает, что та или иная специальная тема имеет шансы на будущее, если она принята в программу некоторой исследовательской группы, которая обладает необходимыми материальными и людскими ресурсами.

Представим себе, что некоторой специальной теме повезло, и ею заинтересовалась некоторая исследовательская группа. Теперь многое зависит от таких параметров, как отношение общественного мнения (в научном сообществе) к данной теме, наличие или потенциал финансирования исследований. В лучшем случае рождение темы отвечает требованиям времени, как это произошло с теорией резолюций после опубликования Робинсоном [1965] работы "A machine oriented logic, based on the resolution principle". Множество исследовательских центров оказались ориентированными на поиск доказательства в теории резолюций. Дополнительный импульс для развития теории резолюций был дан разработкой языка логического программирования Prolog в начале 80-х годов прошлого столетия: синтаксис Prolog – это фактически синтаксис "специализированного" языка логики предикатов первого порядка, где выражения представлены в конъюнктивной нормальной форме, и дизъюнкты имеют максимум один позитивный литерал. Процедурная основа Prolog – это применение принципа резолюций к некоторому запросу, выраженному в клаузальной форме.

Другой пример – теория тестирования моделей. После появления первых работ в данной области (см. детальное изложение проблематики теории тестирования моделей в Кларк [1999]) общественное мнение в среде компьютерной науки на десятилетия повернулось в сторону данной области. Более того, крупнейшие университетские центры, такие, как, например, Карнеги Меллон университет, оказались вовлеченными в исследования по тестированию моделей. В такой ситуации, когда ученые с мировым именем (Пнуэли, Варди, Кларк) концентрируют вокруг себя огромные материальные и исследовательские ресурсы, становится совершенно очевидным, что именно здесь будет сосредоточен и интерес бизнеса. Именно так и произошло с теорией тестирования моделей – эта область стала привлекательной для бизнеса, в первую очередь для крупных компаний и организаций, таких, как Intel, Microsoft или NASA. Все это приводит к значительным прорывам в практическом применении методов, разработанных в теории тестирования моделей. Так, например, с помощью применений методов тестирования моделей удалось обнаружить более ста программных "дыр" в процессоре Intel 4 (см. Бентли [2001]). Более двадцати проблем оказались серьезными, и при отсутствии примененной методики тестирования некоторые из них, такие,

как неправильные инструкции для вычисления плавающей точки, могли оказаться незамеченными.

2.7. *Перспективы развития*

Принимая во внимание предпринятый анализ интеллектуальных агентов на современном этапе искусственного интеллекта, основных тенденций в развитии методов спецификации и верификации компьютерных систем, мы попытаемся определить перспективы дальнейшего развития данной области и, в частности, место натурального вывода.

Как было показано, второй и третий этапы преобразования информации в конструировании интеллектуальных систем (см. раздел 2.4.) – это области, где автоматическая дедукция или уже успешно применяется или является предметом разработок. Выше мы рассматривали формальную модель интеллектуальной системы S как систему пошаговых переходов. Но мы можем понимать эту модель и как автомат, определенный на множестве миров $W = \{w_0, \ldots, w_k, \ldots\}$. Поскольку система должна начать работу, некоторое данное изначальное состояние системы w_0 соответствует входу в автомат. Переход от одного состояния к другому – это шаг автомата. Сигналы, поступающие из окружающей среды и кодированные системой, – это компоненты конечного или бесконечного слова или дерева (но всегда составленного из элементов конечного алфавита), которое автомат читает. Применяя заранее определенную многозначную функцию *f* пошагового перехода, автомат осуществляет выбор своего следующего состояния.

Поскольку предполагается, что интеллектуальный агент должен иметь формальную модель своих состояний (как настоящего состояния, так и прошлых состояний), то встает задача применения соответствующих методов спецификации. Здесь в качестве спецификационного языка наиболее приемлемым является язык временны*ы*х логик, а верификация полученных представлений может осуществляться или путем применения теории автоматов или путем дедукции. Не случайны поэтому попытки совмещения методов тестирования моделей и дедуктивных методов, включая натуральный вывод (см. Кларк, Джа и Марреро [1998]).

Автомато-теоретические и дедуктивные методы как бы дополняют друг друга – там, где применение одного метода ограничено, другой метод работает эффективно. Например, при построении автомата нам нужна система с конечным количеством миров, в то время как для дедуктивных методов такого ограничения нет. Построение автоматов из системы с конечным множеством миров связано с проблемой экспоненциального взрыва. В то же время методы верификации автомата, т.е. проверка его непустоты, обычно представлены процедурами, легче поддающимися алгоритмизации, чем поиск доказательства в автомати-

ческой дедукции. Кроме того, для ряда автоматов сложность этого поиска – полиномиальна, в то время как сложность автоматического поиска доказательства для дедуктивных методов является, по крайней мере, экспоненциальной.

Заметим также, что интеллектуальный агент оперирует с информационной базой, которая, в свою очередь, нуждается в репрезентации. Здесь мы сталкиваемся с традиционной проблемой искусственного интеллекта – формализацией информационной базы и наделением агента способностью автоматической дедукции. Именно здесь, как нам кажется, и выявляется роль алгоритмического натурального вывода. Ведь оперирование информацией на этом уровне работы интеллектуальной системы – это не просто тестирование некоторых утверждений. Важен, например, процесс выбора гипотез, прослеживание того, какие именно утверждения участвуют в выводе, и так далее. Не случайно поэтому появление ряда работ, в которых намечено такое использование натурального вывода (Мейер [2000], Бэзйин, Мэттьюс и Вигано [1998], Де Лима и Лингенфельдер [2000], Кларк, Джа и Марреро [1998], Поляков и Пфеннинг [1999]).

Наконец, заметим, что в последнее время исследователи, работающие в области автоматического доказательства, проявляют все больший интерес к так называемым индексированным дедуктивным системам (Labelled Deductive Systems (см. Габбай [1996])). Как нам кажется, с идейной точки зрения, замысел индексированных дедуктивных систем связан с попыткой переноса основных технических проблем на уровень работы с индексами. В нашей работе в последней главе тоже будет использован метод индексации.

Аналогичные явления можно заметить в появившихся недавно попытках разработки методов натурального вывода для временны*х* логик (Расмуссен 2001], Барателла и Масини [2003]), модальных логик (Брода, Фингер и Руссо [1999]) и релевантных логик (Бэйзин, Мэттьюс и Вигано [1998]). В этих работах авторы индексируют формулы, к которым применяются правила натурального вывода, и в формулировках самих правил фундаментальным является именно техника работы с индексами.

ГЛАВА II

АВТОМАТИЗАЦИЯ ВЫВОДА В КЛАССИЧЕСКОЙ ЛОГИКЕ

Ниже будет в общем плане описана идея автоматизации вывода и доказательства для натурального исчисления классической логики. В основу дедуктивного аппарата положено исчисление, представленное в работе Бочарова и Маркина [1994]. Для понимания дальнейшего нам потребуется четко ввести язык этого исчисления и задать принципы работы в нем. Однако для цели описания алгоритма поиска доказательств мы введем вначале более богатый язык, а затем укажем, что в его составе будет относиться к языку исчисления.

1. Язык и натуральное исчисление предикатов

1.1. *Язык алгоритмического представления исчисления предикатов*

1. $x, y, z, x_1, y_1, z_1, x_2,...$ – бесконечный список *индивидных переменных*,
2. $a, b, c, a_1, b_1, c_1, a_2,...$ – список *индивидных констант*,
3. $v_1, v_2,...$ – список *вре́менных переменных*,
4. $p, q, r, p_1, q_1, r_1, p_2,...$ – бесконечный список *пропозициональных переменных*,
5. $P, Q, R, P_1, Q_1, R_1, P_2,...$ – список *предикаторов* различной местности (местность не указывается, так как она однозначно высчитывается по количеству аргументов),
6. $f, g, h, f_1, g_1, h_1, f_2,...$ – список *функторов* различной местности (местность не указывается по вышеуказанной причине),
7. $\&, \vee, \supset, \neg, \forall, \exists$ – *логические термины*,
8. (,), , – *технические символы*.

Понятие псевдотерма

1. Любая индивидная переменная есть псевдотерм,
2. Любая индивидная константа есть псевдотерм,
3. Любая вре́менная переменная есть псевдотерм,
4. Если Ф – *n*-местный функтор, а $t_1, t_2,..., t_n$ – псевдотермы, то выражение вида $Ф(t_1, t_2,..., t_n)$ – псевдотерм,
5. Ничто иное не есть псевдотерм.

Понятие терма

Термом называется псевдотерм, в котором отсутствуют вре́менные индивидные переменные.

Понятие псевдоформулы

1. Если П – *n*-местный предикатор, а t_1, t_2,..., t_n – псевдотермы, то выражение вида $П(t_1, t_2,..., t_n)$ – псевдоформула,
2. Каждая пропозициональная переменная есть псевдоформула,
3. Если А – псевдоформула, то $\neg$А – псевдоформула,
4. Если А и В – псевдоформулы, то (А & В), (А $\vee$ В), (А $\supset$ В) – псевдоформулы,
5. Если А – псевдоформула и α – индивидная переменная, то выражения вида $\forall\alpha$А и $\exists\alpha$А – псевдоформулы,
6. Ничто иное не есть псевдоформула.

Псевдоформулы, определяемые пунктами 1 и 2, называются *элементарными*. Все остальные псевдоформулы называются *сложными*.

В выражениях вида $\forall\alpha$А и $\exists\alpha$А псевдоформула А называется *областью действия кванторов* $\forall$ и $\exists$, *взятых по переменной* α.

Индивидная переменная или вре́менная индивидная переменная s входит в псевдоформулу А, если в ее составе есть хотя бы одно вхождение s.

Вхождение индивидной переменной α называется *связанным*, если данная переменная расположена непосредственно справа от некоторого квантора или она находится в области действия квантора, взятого по переменной α. Остальные вхождения индивидной переменной α называются *свободными вхождениями*.

Понятие формулы

Формулой называется псевдоформула, в которой нет вхождений вре́менных индивидных переменных.

Пусть t – будет псевдотермом, а α – индивидной переменной. Тогда выражения вида $A(\alpha/t)$ обозначают результат правильной подстановки в псевдоформулу $A(\alpha)$ вместо всех свободных вхождений индивидной переменной α соответственно псевдотерма t. Подстановка считается правильной, если никакая индивидная переменная, входящая в терм t, не окажется связанной в местах подстановки терма t.

К языку исчисления предикатов относятся только те объекты, которые задаются понятиями терма и формулы. Понятия же псевдоформулы и псевдотерма необходимы лишь для описания работы алгоритма.

1.2. *Правила вывода*

$\&$в $\dfrac{A,\ B}{A\ \&\ B}$ $\qquad$ $\&$и $\dfrac{A\ \&\ B}{A}$ $\qquad$ $\dfrac{A\ \&\ B}{B}$

$\vee$в $\dfrac{A}{A \vee B}$ $\qquad$ $\dfrac{B}{A \vee B}$ $\qquad$ $\vee$и $\dfrac{A \vee B,\ \neg A}{B}$

$\supset$в $\dfrac{B}{C \supset B}$, где C – последняя посылка $\qquad$ $\supset$и $\dfrac{A \supset B,\ A}{B}$

$\neg$в $\dfrac{B,\ \neg B}{\neg C}$, где C – последняя посылка $\qquad$ $\neg$и $\dfrac{\neg\neg A}{A}$

$\forall$в $\dfrac{A(\alpha/\beta, \gamma_1,..., \gamma_n)}{\forall\alpha A(\alpha, \gamma_1,..., \gamma_n)}$ β – абс. огр., $\gamma_1,..., \gamma_n$ – огр. $\qquad$ $\forall$и $\dfrac{\forall\alpha A(\alpha)}{A(\alpha/t)}$

$\exists$в $\dfrac{A(\alpha/t)}{\exists\alpha A(\alpha)}$ $\qquad$ $\exists$и $\dfrac{\exists\alpha A(\alpha, \gamma_1,..., \gamma_n)}{A(\alpha/\beta, \gamma_1,..., \gamma_n)}$ β – абс. огр., $\gamma_1,..., \gamma_n$ – огр.

В формулировках этих правил считается, что абсолютная переменная β ограничивает каждую из переменных $\gamma_1,..., \gamma_n$.

Выводом в системе называется непустая конечная последовательность формул $C_1, C_2,..., C_k$, удовлетворяющая условиям:

(1) каждая C_i есть либо гипотеза (посылка), либо получена из предыдущих формул по одному из правил вывода,

(2) если в выводе применялись правила $\supset$в или $\neg$в, то все формулы, начиная с последней гипотезы и вплоть до результата применения данного правила, в дальнейших шагах построения вывода не принимают участия (их дальнейшее применение в выводе исключается),

(3) ни одна свободная индивидная переменная не характеризуется в выводе дважды как абсолютная переменная,

(4) ни одна свободная индивидная переменная не ограничивает в выводе сама себя.

Особо отметим, что вывод, согласно данному определению, является именно последовательностью формул, а не пседоформул.

При осуществлении вывода необходимо строго следить, чтобы ни одна переменная не ограничивала сама себя. Здесь надо иметь в виду, что такое самоограничение может возникнуть за счет транзитивности отношения *ограничения*, а именно: если некоторая переменная s при применении данных правил ограничивает переменную s_1, а при другом применении правил переменная s_1 будет ограничивать s, то по транзитивности возникнут два случая самоограничения – s будет ограничивать s и s_1 будет ограничивать s_1. Ни та, ни другая ситуация недопустима.

Вывод C_1, C_2,..., C_k есть вывод вида A_1 A_2,..., $A_n \vdash B$, если A_1, A_2,..., A_n – это в точности все неисключенные посылки (гипотезы), а формула B графически совпадает с C_k.

Вывод называется завершенным, если ни одна свободная индивидная переменная, которая характеризовалась в выводе как абсолютная переменная, не встречается ни в посылках вывода, ни в заключении.

Доказательством (*завершенным доказательством*), как обычно, считается вывод (завершенный вывод) из пустого множества посылок.

Далее будем псевдоформулы, стоящие в правилах вывода над чертой, называть, *посылками правил вывода*, а псевдоформулы, стоящие под чертой, – *заключением правил вывода*.

Для описания алгоритма нам потребуется теперь ввести важные понятия: *компоненты псевдоформулы* и *подобия псевдоформул*. Мы их введем совместным определением.

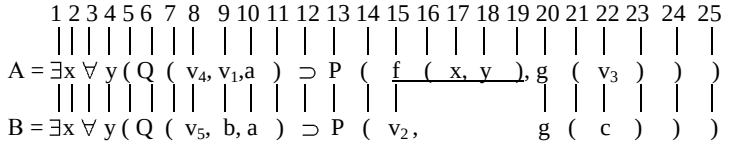
Две псевдоформулы A и B будем считать *подобными*, если и только если при их познаковом сравнении слева направо устанавливаются следующие факты:

1) первый (самый левый) знак псевдоформул A и B называется 1-й компонентой этих псевдоформул и эти первые компоненты графически совпадают,
2) если следующий знак в псевдоформулах A и B, идущий за их j-ми компонентами, не является вре́менной переменной ни в псевдоформуле A, ни в псевдоформуле B, то этот знак объявляется j+1-й компонентой этих псевдоформул и эти компоненты графически совпадают,
3) если следующий знак или в псевдоформуле A, или в псевдоформуле B, идущий за их j-ми компонентами, является вре́менной переменной v_n, то эта переменная в псевдоформуле, где она встречается, объявляется j+1 компонентой; тогда j+1 компонентой другой псевдоформулы объявляется:

(а) если это не функциональный знак, то следующий знак, идущий за j-й компонентой этой псевдоформулы, и он есть или вре́менная переменная v_n, или вре́менная переменная v_m ($n \neq m$), или свободная индивидная переменная,

(б) если это функциональный знак Ф, то комплекс знаков, идущий за j-й компонентой этой псевдоформулы, и этот комплекс есть выражение Ф(...).

Иначе говоря, две псевдоформулы подобны, если они покомпонентно подобны. Рассмотрим в качестве примера две псевдоформулы и укажем нумерацией их компоненты:



Итак, эти две псевдоформулы являются подобными. Отметим, что в ходе установления подобия каждая вре́менная переменная получила некоторое значение. В нашем примере вре́менные переменные получили следующие значения: $v_4 = v_5$, $v_1 = b$, $v_2 = f(x, y)$, $v_3 = c$. Если теперь в обеих псевдоформулах произвести замену вре́менных переменных на соответствующие псевдотермы (причем, если значением вре́менной переменной является другая вре́менная переменная, то вре́менная переменная с большим индексом меняется на вре́менную переменную с меньшим индексом), то эти две псевдоформулы будут не просто подобными, но станут тождественными. Такую процедуру будем называть далее *унификацией*. Более точно это понятие будет введено в следующей главе.

2. Алгоритм поиска вывода

2.1. *Общая стратегия алгоритмического поиска вывода*

Стратегия состоит в том, что по некоторому заданному метаутверждению о выводимости, которое надо автоматически обосновать, создаются две последовательности. Первая последовательность – это последовательность формул C_1, C_2,..., C_k, которая и представляет собой формируемый вывод. На некоторых шагах работы алгоритма эта последовательность может временно содержать и псевдоформулы. Поэтому псевдоформулы, входящие в данную последовательность, называются *псевдоформулами вывода*. Вторая же последовательность – это последовательность *целей*, в качестве которых могут выступать либо некоторые конкретные псевдоформулы, либо метка "⊥", означающая, что целью является противоречие. При этом на каждом шаге вывода однозначно задается та цель, которая на данный момент является последней и должна быть *достигнута*, т.е. которую мы стараемся получить в ходе построения вывода.

Все правила вывода мы подразделим на *аналитические* и *синтетические*. К первым относятся те правила, которые автоматически выполняются в выводе при наличии в нем соответствующих псевдоформул. Этими правилами являются &и, ∨ и, ⊃и, ¬и, ∀и и ∃и, т.е. все правила исключения логических связок. Все остальные правила яв-

ляются синтетическими и их применение регулируется *достижением* некоторой цели. Как это конкретно происходит, описывается в самом алгоритме.

2.2. *Процедуры, выполняемые в ходе поиска вывода*

(I) По псевдоформулам вывода осуществляются следующие процедуры:

Процедура 1 – формирует последовательность псевдоформул вывода. Она состоит в нахождении одной или двух псевдоформул, к которым применяются соответствующие правила исключения логических связок. Если такие псевдоформулы обнаруживаются, то вывод пополняется результатом применения данного правила.

Процедура 2 – формирует новые цели. Эта процедура выполняется в том случае, когда все возможные, указанные в процедуре 1, правила вывода применены, последней целью в последовательности целей является $\perp$ (противоречие) и при этом данная цель не достигнута (см. процедуру 3). Какие именно вводятся цели, определяется видом содержащихся в выводе псевдоформул, и будет объяснено далее.

Процедура 3 – осуществляет проверку достижимости последней цели в списке целей. Если последней целью является некоторая псевдоформула, то она считается достигнутой, если в выводе имеется подобная ей псевдоформула, что устанавливается унификацией. Если же последней целью является $\perp$, то она считается достигнутой, если в выводе содержатся две псевдоформулы вида A и $\neg$A, где формулы A в обеих выражениях подобны, что также устанавливается унификацией. Достигнутая цель из списка целей устраняется, а очередной целью становится предыдущая цель в списке целей.

Процедура 4 – осуществляется переход к циклу. Эта процедура применяется в том случае, когда все выводы сделаны, все цели взяты, последней целью является $\perp$, эта цель не достигнута и в выводе содержатся *псевдоформулы цикла*. Что представляют собой псевдоформулы цикла, будет разъяснено ниже.

(II) По последовательности целей осуществляются процедуры:

Процедура 5 – формирует подцели, которые помещаются в последовательность целей.

Процедура 6 – выбирает новые дополнительные посылки, помещаемые под очередным номером в последовательность псевдоформул вывода.

Процедуры 5 и 6 опишем вместе, так как они в определенных случаях применяются одновременно. Данные процедуры начинают использоваться тогда, когда уже применена процедура 1, целью является некоторая псевдоформула, и при этом она не достигнута. В этом

случае последовательность целей дополняется новой подцелью или совокупность посылок в выводе дополняется новой посылкой. Что конкретно происходит, зависит от вида очередной (последней) цели и будет объяснено в формулировке алгоритма.

Процедура 7 – сигнализирует о необходимости автоматического применения в выводе правил введения логических связок. Более детально это описывается ниже.

2.3. *Описание алгоритма*

1. Определяется *главная цель* вывода. Таковой является псевдоформула, стоящая справа от знака выводимости в заданном метаутверждении о выводимости. Данная псевдоформула помещается в качестве начальной в последовательность целей. Если слева от знака выводимости стоят псевдоформулы, то все они помещаются в качестве начальных в последовательность псевдоформул вывода. Указывается, что эти формулы являются посылками.

Переход к 2.

2. Анализ множества псевдоформул вывода.

2.1. Если множество псевдоформул вывода пусто, то переход к 4.

2.2. Если множество псевдоформул вывода непусто, то переход к 3.

3. Умозаключения по псевдоформулам вывода.

3.1. Осуществляются все умозаключения по аналитическим правилам; при этом на посылки правил вывода вида A & B, A $\vee$ B, A $\supset$ B, $\neg$A, $\forall\alpha$A и $\exists\alpha$A, к которым были применены, соответственно, правила &и, $\vee$ и, $\supset$и, $\neg$и, $\forall$и и $\exists$и, ставится *метка* "V0", указывающая на два обстоятельства: (а) недопустимость вторичного применения к ним указанных правил, (б) невозможность их использования для порождения новых целей; на псевдоформулу $\forall\alpha$A дополнительно ставится метка "V1", указывающая, что данная псевдоформула является *псевдоформулой цикла*; на заключения правил вывода ставится метка "V+0" (смысл этой метки разъясняется далее). При этом правила $\forall$и и $\exists$и применяются следующим образом:

3.1.1.$\forall$и.

(а) Квантор общности снимается по правилу $\forall$и с псевдоформулы $\forall\alpha$A по первой вре́менной переменной в алфавитном порядке, которая отсутствует как в псевдоформулах вывода, так и в целях.

(б) Если в псевдоформуле A вообще отсутствует связанная индивидная переменная α, то результатом снятия квантора общности будет сама псевдоформула A.

3.1.2. $\exists$и.

(а) Квантор существования снимается по правилу ∃и с псевдоформулы $\exists\alpha A$ по первой свободной индивидной переменной в алфавитном порядке, которая отсутствует как в псевдоформулах вывода, так и в целях.

(б) Рядом с так полученной псевдоформулой $A(\alpha/s)$ пишется "s - абс, s_1, s_2,..., s_n – огр.", где s_1, s_2,..., s_n – все вре́менные или свободные индивидные переменные, входящие в псевдоформулу $\exists\alpha A$.

(в) Если в псевдоформуле $A(\alpha/s)$ встречается некоторая вре́менная переменная v_i, то отмечается и запоминается, что $v_i \neq s$ или $\neq$ любому сложному терму, который содержит переменную s. Эта пометка делается для того, чтобы абсолютно ограниченная переменная не ограничивала сама себя.

(г) Если в псевдоформуле А вообще отсутствует связанная индивидная переменная α, то результатом снятия квантора общности будет сама псевдоформула А.

Переход к 11.

3.2. Если ни одно аналитическое правило не применимо, то – 4.

4. Анализируется главный знак очередной последней цели – W.

4.1. W – элементарная псевдоформула. Переход к 5.

4.2. Главный знак W – ¬. Переход к 5.

4.3. W $\rightleftharpoons W_i \supset W_j$. Переход к 6.

4.4. W $\rightleftharpoons W_i$ & W_j. Переход к 7.

4.5. W $\rightleftharpoons W_i \vee W_j$. Переход к 8.

4.6. W $\rightleftharpoons \forall\alpha A$. Переход к 9.

4.7. W $\rightleftharpoons \exists\alpha A$. Переход к 10.

4.8. Цель W – противоречие. Переход к 14.

5. ¬W включается в последовательность псевдоформул вывода в качестве *посылки*, *подцелью* становится получение противоречия, тип устраняемости – ¬. Переход к 3.

6. Псевдоформула W_i включается в последовательность псевдоформул вывода в качестве посылки, подцелью становится W_j, а тип устраняемости – ⊃. Переход к 3.

7. W_i и W_j становятся подцелями. Начинаем всегда работать с W_i.

7.1. Подцель – W_i. Псевдоформула W_i, а также, начиная с этого момента, все новые псевдоформулы вывода и цели помечаются меткой – "V-1".

Если W_i *достижима*, то со всех псевдоформул вывода снимается метка "V-1", переход к 7.2.

В противном случае – стоп. Алгоритмически доказать метаутверждение невозможно.

7.2. Подцелью становится W_j. Псевдоформула W_j, а также, начиная с этого момента все новые псевдоформулы вывода и цели помечаются меткой – "V-2".

Если W_j *достижима*, то со всех псевдоформул вывода снимается метка "V-2", переход к 12.

В противном случае – стоп. Алгоритмически доказать метаутверждение невозможно.

8. W_i или W_j становятся подцелями. Начинаем работать с W_i.

8.1. Цель – W_i. Псевдоформула W_i, а также все новые псевдоформулы вывода и цели помечаются меткой – "V-3".

Если цель W_i *достижима*, то 12.

В противном случае – 8.2.

8.2. Цель – W_j. Из вывода и целей устраняются все псевдоформулы вывода и все цели, отмеченные меткой "V-3". Псевдоформула W_j, а также все новые псевдоформулы вывода и цели помечаются меткой – "V-4".

Если цель W_j *достижима*, то – 12.

В противном случае – 8.3.

8.3. Из последовательности целей и псевдоформул вывода устраняются все подцели и все псевдоформулы с метки "V-4".

(а) Если на цели $W_i \vee W_j$ не стоит метка "V2", то псевдоформула $\neg(W_i \vee W_j)$ берется в качестве новой посылки и помечается меткой "V2", а в качестве новой цели берется противоречие $\perp$. Переход к 3.

(б) В противном случае, т.е. если на цели $W_i \vee W_j$ стоит метка "V2", осуществляется временный выход из доказательства исходного метаутверждения, с псевдоформулы вывода $\neg(W_i \vee W_j)$ снимаются все метки, цель $W_i \vee W_j$ устраняется, а в качестве новой цели берутся две псевдоформулы $\neg W_i$ и $\neg W_j$ и строится вывод $\neg(W_i \vee W_j) \vdash \neg W_i$ и $\neg W_j$. После осуществления этого вывода, а он всегда осуществим, на псевдоформулу вывода $\neg(W_i \vee W_j)$ ставится метка "V0". Переход к 3.

9. Подцелью становится псевдоформула $A(\alpha/s)$, где s – первая в алфавитном порядке свободная индивидная переменная, не встречающаяся ни в псевдоформулах вывода, ни в целях.

(а) Рядом с так полученной псевдоформулой $A(\alpha/s)$ пишется "s – абс, s_1, s_2,..., s_n - огр.", где s_1, s_2,..., s_n – все вре́менные или свободные индивидные переменные, входящие в псевдоформулу $\forall\alpha A$.

(б) Если в псевдоформуле $A(\alpha/s)$ встречается некоторая вре́менная переменная v_i, то отмечается и запоминается, что $v_i \neq s$ или $\neq$ любому сложному терму, который содержит переменную s. Эта пометка дела-

ется для того, чтобы абсолютно ограниченная переменная не ограничивала в выводе сама себя.

(в) Если в псевдоформуле A вообще отсутствует связанная индивидная переменная α, то результатом применения данного пункта алгоритма будет сама псевдоформула A.

Переход к 11.

10. Подцелью становится формула $A(\alpha/v_i)$, где v_i – первая в алфавитном порядке вре́менная переменная, не встречающаяся ни в псевдоформулах вывода, ни в целях.

Если в псевдоформуле A вообще отсутствует связанная индивидная переменная α, то результатом снятия квантора существования будет сама псевдоформула A.

10.1. Цель $A(\alpha/v_i)$, а также все новые псевдоформулы вывода и цели помечаются меткой – "V-5".

Если данная цель *достигнута*, то переход к 12.

В противном случае, переход к 10.2.

10.2. Из последовательности целей и псевдоформул вывода устраняются все подцели и все псевдоформулы с метки "V-5".

(а) Если на цели $\exists\alpha A$ не стоит метка "V3", то псевдоформула $\neg\exists\alpha A$ берется в качестве новой посылки и помечается меткой "V3", а в качестве новой цели берется противоречие. Переход к 3.

(б) В противном случае, т.е. если на цели $\exists\alpha A$ стоит метка "V3", осуществляется временный выход из доказательства исходного метаутверждения, с псевдоформулы вывода $\neg\exists\alpha A$ снимаются все метки, цель $\exists\alpha A$ устраняется, а в качестве новой цели берется псевдоформула $\forall\alpha\neg A$ и осуществляется вывод $\neg\exists\alpha A \vdash \forall\alpha\neg A$. После осуществления этого вывода, а он всегда осуществим, на псевдоформулу вывода $\neg\exists\alpha A$ ставится метка "V0". Переход к 3.

11. Проверка достижимости последней цели.

(а) Если последней целью является псевдоформула A, то она считается достигнутой тогда, когда среди неисключенных псевдоформул вывода имеется псевдоформула B такая, что A и B подобны друг другу, что устанавливается посредством унификации.

В противном случае цель считается не достигнутой.

(б) Если последней целью является противоречие $\perp$, то она считается достигнутой тогда, когда среди неисключенных псевдоформул вывода найдутся две псевдоформулы A и $\neg$B такие, что A и B подобны друг другу, что также устанавливается посредством унификации.

В противном случае цель считается не достигнутой.

Если цель достигнута, то переход к 12.

Если цель не достигнута, то переход к 13.

12. Достигнута цель W_n.

12.1. Если W_n – *главная цель*, т.е. самая верхняя цель в их последовательности, то W_n помечается как *достигнутая*, выход из алгоритма – выводимость обоснована.

12.2. Если W_n – противоречие и некоторые вре́менные переменные в результате унификации получили значение, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя.

(в) W_n как цель устраняется,

(г) Новой целью становится предыдущая цель, а к псевдоформулам вывода применяется правило ¬в. Результатом является псевдоформула ¬B, где B – последняя посылка в выводе,

(д) Делается отметка, что все псевдоформулы, начиная с B и заканчивая псевдоформулой, предшествующей ¬B, в дальнейших шагах вывода принимать участия не могут,

(е) Если в число этих псевдоформул попадает псевдоформула с меткой "V+0", являющаяся заключением некоторого правила вывода, то с соответствующей псевдоформулы, являющейся посылкой данного применения правила вывода, снимается метка "V0". Переход к 3.

12.3. Если предыдущей к достигнутой цели W_n является цель $W_{n-1} = W_n \vee W_k$ или $W_k \vee W_n$ и некоторые вре́менные переменные при этом получили значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя.

(в) Из последовательности целей устраняются цели W_n и W_{n-1},

(г) Новой целью становится предыдущая цель, а в последовательность псевдоформул вывода включается псевдоформула W_{n-1}, которая является результатом применения правила ∨в к псевдоформуле W_n,

(е) Если на цели W_{n-1} стояла метка "V4", то с *псевдоформулы вывода, породившей данную цель* (об этом будет сказано ниже) снимается данная метка. Переход к 3.

12.4. Если предыдущей к достигнутой цели W_n является цель W_{n-1} $=W_k \supset W_n$ и некоторые вре́менные переменные при этом получили значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя.

(в) Из последовательности целей устраняются цели W_n и W_{n-1},

(г) Очередной целью становится ближайшая ранее не достигнутая цель, к псевдоформуле вывода W_n применяется правило $\supset$в, т.е. псевдоформула W_{n-1} переносится в вывод, и в последовательности псевдоформул вывода делается отметка, что все псевдоформулы, начиная с W_k и заканчивая псевдоформулой W_n, в дальнейших шагах вывода принимать участия не могут,

(д) Если в число последних псевдоформул попадает псевдоформула с меткой "V+0", являющаяся заключением некоторого правила вывода, то с соответствующей псевдоформулы, являющейся посылкой данного применения правила вывода, снимается метка "V0",

(е) Если на цели W_{n-1} стояла метка "V4", то с *псевдоформулы вывода, породившей данную цель*, снимается метка "V4". Переход к 3.

12.5. Если предыдущей к достигнутой цели W_n является цель W_{n-1} $=W_k$ & W_n и некоторые вре́менные переменные при этом получили значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя,

(в) W_n и W_{n-1} устраняются из последовательности целей,

(г) Очередной целью становится ближайшая ранее не достигнутая цель, к псевдоформулам вывода W_k и W_n применяется правило &в, т.е. псевдоформула W_{n-1} переносится в вывод и в последовательности псевдоформул вывода на псевдоформулу W_k & W_n ставится метка "V0",

(д) Если на цели W_{n-1} стояла метка "V4", то с *псевдоформулы вывода, породившей данную цель*, снимается метка "V4". Переход к 3.

12.6. Если предыдущей к достигнутой цели W_n является цель W_{n-1} $=\forall\alpha A$ и некоторые вре́менные переменные при этом получили значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя,

(в) W_n и W_{n-1} устраняются из последовательности целей,

(г) Очередной целью становится ближайшая ранее не достигнутая цель, к псевдоформуле вывода W_n применяется правило $\forall$в, т.е. псевдоформула W_{n-1} переносится в вывод и в последовательности псевдоформул вывода на псевдоформулу $\forall\alpha$A ставятся метки "V1" и "V0",

(д) Если на цели W_{n-1} стояла метка "V4", то с *псевдоформулы вывода, породившей данную цель*, снимается метка "V4". Переход к 3.

12.7. Если предыдущей к достигнутой цели W_n является цель $W_{n-1} \neq \exists\alpha$A и некоторые вре́менные переменные при этом получили значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя,

(в) W_n и W_{n-1} устраняются из последовательности целей,

(г) Очередной целью становится ближайшая ранее не достигнутая цель, к псевдоформуле вывода W_n применяется правило $\exists$в, т.е. псевдоформула W_{n-1} переносится в вывод и в последовательности псевдоформул вывода на псевдоформулу $\exists\alpha$A ставится метка "V0",

(д) Если на цели W_{n-1} стояла метка "V4", то с *псевдоформулы вывода, породившей данную цель*, снимается метка "V4". Переход к 3.

12.8. Если достигнутая цель имеет метку "V4" и некоторые вре́менные переменные при этом получили свое значение посредством унификации, то:

(а) Во всех псевдоформулах вывода и целей, содержащих данные вре́менные переменные, осуществляется приписывание значений вре́менным переменным,

(б) Проверяется, не возникла ли при приписывании значений вре́менным переменным ситуация, когда некоторая переменная ограничивает сама себя,

(в) Данная цель из последовательности целей убирается, новой становится предыдущая цель.

Переход к 3.

13. Цель W_n не достигнута.

Если последней целью является псевдоформула и она не достигнута, то 4.

Если последней целью является противоречие и она не достигнута, то 14.

14. Выбор новых целей по формулам вывода.

14.1. Если при просмотре сверху вниз всех псевдоформул вывода в их последовательности имеются сложные псевдоформулы, не отмеченные метками "V0" или "V4", то к ним применяются следующие процедуры:

(а) Если в выводе имеется импликативная псевдоформула вида $W_i \supset W_j$, то в качестве подцели берется W_i. Псевдоформула $W_i \supset W_j$ и цель W_i помечаются метками "V4". Переход к 4.

(б) Если в выводе имеется дизъюнктивная псевдоформула вида $W_i \vee W_j$, то в качестве подцели берется $\neg W_i$. Псевдоформула $W_i \vee W_j$ и цель $\neg W_i$ помечаются метками "V4". Переход к 4.

(в) Если в выводе встречается псевдоформула $\neg W_i$, где W_i сложная формула, то в качестве цели берется W_i. Псевдоформула $\neg W_i$ и цель W_i помечаются метками "V4". Переход к 4.

14.2. В противном случае переход к 15.

15. Если при просмотре сверху вниз всех псевдоформул вывода в их последовательности имеются псевдоформулы, отмеченные меткой "V1", то 16.

В противном случае 17.

16. Вхождение в цикл.

Фиксируется последняя цель. Цикл начинается с применения ко всем псевдоформулам вывода, отмеченным меткой "V1", правил вывода.

Если, применяя далее алгоритм, мы вновь пришли к пункту 16 и при этом последняя фиксированная цель не была достигнута, то цикл считается *непродуктивным*, переход к 17.

17. (а) Если последняя цель – противоречие – отмечена меткой "V-1", то выход из алгоритма – вывод неосуществим.

(б) Если последняя цель – противоречие – отмечена меткой "V-2", то выход из алгоритма – вывод неосуществим.

(в) Если последняя цель – противоречие – отмечена меткой "V-3", то переход к 8.2.

(г) Если последняя цель – противоречие – отмечена меткой "V-4", то переход к 8.3.

(д) Если последняя цель – противоречие – отмечена меткой "V-5", то переход к 10.2.

(е) Если последняя цель – противоречие – не отмечена указанными выше метками, то выход из алгоритма – вывод неосуществим.

На этом описание алгоритма заканчивается.

2.4. *Компьютерная реализация алгоритма*

Нами была составлена компьютерная программа, реализующая изложенный ранее алгоритм в той его части, который относится к классической логике высказываний. При разработке той части программы, которая обеспечивает доказательство теорем этой логики, достаточно при установлении подобия формул ограничиться сравнением строковых выражений. Это означает, что две формулы подобны тогда и только тогда, когда все символы, использованные для их представления, одинаковы в каждой компоненте. В этом случае при написании программы достаточно использовать инструментальные средства, обеспечивающие эффективную работу со строковыми переменными. Однако при доказательстве теорем логики предикатов такое формальное сравнение бессмысленно. В случае логики предикатов для установления эквивалентности двух формул необходимо осуществлять не только формальное сравнение (например, если на одной и той же позиции в них стоит предикатор, то он должен быть одним и тем же), но и проводить сравнение значений элементов формул (например, для вре́менных переменных). Из этого следует, что элементы, составляющие формулу логики предикатов, должны обладать некоторыми свойствами (например, вре́менные переменные должны принять значения). Для программной реализации такого рода переменных наиболее подходят объектно-ориентированные языки программирования, развитие которых в последнее время идет бурными темпами. Для реализации алгоритма авторами был выбран объектно-ориентированный язык программирования C++.

Использование этого языка программирования позволило создать компактную программу, осуществляющую доказательство теорем классической логики высказываний в автоматическом режиме. Программа работает в среде Windows 3.11 и занимает на диске около 180 Кбайт. Однако следует учитывать, что указанный объем памяти относится лишь к исполнимому коду программы и не учитывает памяти, которая требуется для построения вывода (сегмент данных). При работе программы для хранения формул вывода и формул целей память выделяется автоматически. Таким образом, оценить полный объем памяти, необходимый для работы программы, довольно сложно, так как это количество зависит от числа шагов вывода и формул целей, требуемых для доказательства данной формулы. Время доказательства одной теоремы составляет несколько секунд для процессора Intel-386-DX с тактовой частотой 40 Мгц. Программа имеет пользова-

тельский интерфейс и может читать заранее заготовленные формулы из файла. Тестирование программы проводилось на множестве формул, выбранных из учебников по математической логике, в частности, из книги А. Черча [1960]. Это множество – около ста теорем классической логики высказываний и различных их модификаций – достаточно репрезентативно, что позволяет рассматривать предложенный алгоритм как эффективную процедуру осуществления субординатного вывода компьютерными средствами.

Мы приведем здесь два примера компьютерного осуществления вывода. Один из них – доказательство закона Пирса.

Требуется доказать: $((p \supset q) \supset p) \supset p$.

	вывод		анализ	цели
	0.			$((p \supset q) \supset p) \supset p$ – достигнуто
	1.	$(p \supset q) \supset p$	посылка	p – достигнуто в силу 10
	2.	$\neg p$	посылка	$\perp_1$ – достигнуто в силу 2 и 8
	3.	p	посылка	$p \supset q$ – достигнуто в силу 7
	4.	$\neg q$	посылка	q – достигнуто в силу 6
	5.	$\neg\neg q$	¬в из 2 и 3	$\perp_2$ – достигнута в силу 2 и 3
	6.	q	¬и из 5	
	7.	$p \supset q$	⊃в к 6	
	8.	p	⊃и из 1 и 7	
	9.	$\neg\neg p$	¬в из 2 и 8	
	10.	p	¬и из 9	
	11.	$((p \supset q) \supset p) \supset p$	⊃в к 10	

Объясним шаги вывода, строившегося по описанному выше алгоритму. Так как требуется доказать формулу $((p \supset q) \supset p) \supset p$, то исходный список вывода пуст, а главной целью является сама эта формула. По этой формуле, согласно пункту 4, начинает формироваться последовательность формул вывода и в качестве первой посылки берется формула $(p \supset q) \supset p$, а целью становится формула p. Так как эта цель на данном шаге вывода не достигнута, то по пункту 5 в вывод помещается посылка $\neg p$, а целью становится $\perp_1$. Цель $\perp_1$ не достигнута, так как в выводе нет противоречащих друг другу формул, а поэтому по процедуре 14 первая посылка служит источником новой цели, которой становится формула $p \supset q$. Вновь по 4, а также по пункту 6 в качестве новой посылки берется формула p, а в качестве новой цели – формула q. Опять-таки, последняя цель не достигнута, а поэтому по 5 берется новая посылка $\neg q$, а новой целью становится $\perp_2$. На этом шаге автоматического поиска вывода машина устанавливает, что цель $\perp_2$ достигнута, так как в выводе имеется противоречие – формулы p и $\neg p$ (шаги вывода 2 и 3). Это служит сигналом для применения в выводе правила "введения отрицания". Именно таким об-

разом в выводе появляется 5-й шаг – формула $\neg\neg q$. При этом все формулы, начиная с последней посылки (формулы 4) и вплоть до результата применения этого правила считаются из вывода устраненными (что помечено слева от номеров шагов вертикальной линией, длина которой равна числу строчек исключаемых формул). Заметим, что хотя в выводе уже ранее содержалось противоречие, это правило не применялось, так как на это не указывала никакая из целей. Цель $\perp_2$, так как она достигнута, исключается и целью становится предыдущая цель – формула q. Формула q легко достигается, так как к 5-му шагу вывода можно применить правило "исключения отрицания". Достижение формулы q служит сигналом для автоматического применения правила "$\supset$в", что дает формулу 7 вывода. Остальные шаги вывода очевидны и могут быть легко восстановлены.

Рассмотрим теперь пример доказательства предикатной формулы.

Требуется доказать: $\forall x(P(x) \supset Q(x)) \supset (\forall xP(x) \supset \forall xQ(x))$.

1.	$\forall x(P(x) \supset Q(x))$	посылка
2.	$P(y) \supset Q(y)$	$\forall$и из 1
3.	$\forall xP(x)$	посылка
4.	$P(y)$	$\forall$и из 3
5.	$\neg Q(y)$	посылка
6.	$Q(y)$	$\supset$и из 2, 4
7.	$\neg\neg Q(y)$	$\neg$в из 5, 6
8.	$Q(y)$	$\neg$и из 7
9.	$\forall xQ(x)$	$\forall$в, из 8, y – абс.
10	$\forall xP(x) \supset \forall xQ(x)$	$\supset$в к 9
11.	$\forall x(P(x) \supset Q(x)) \supset (\forall xP(x) \supset \forall xQ(x))$	$\supset$в к 10

Вопрос, почему возникла данная последовательность шагов, требует детального обсуждения, что и будет сейчас сделано. Для понимания дальнейшего мы будем помещать цели с правой стороны страницы, а псевдоформулы вывода сдвигать к ее левой стороне. Итак, требовалось обосновать выводимость формулы: $\forall x(P(x) \supset Q(x)) \supset (\forall xP(x) \supset \forall xQ(x))$. Исходный список псевдоформул вывода пуст, а главной целью является сама эта формула:

$$\forall x(P(x) \supset Q(x)) \supset (\forall xP(x) \supset \forall xQ(x)).$$

По этой формуле, согласно пункту 6 алгоритма, начинает формироваться последовательность псевдоформул вывода и в качестве первой посылки берется псевдоформула (помечена цифрой 1):

1. $\forall x(P(x) \supset Q(x))$, V1, V0 (значение меток объясняется далее),

а целью становится консеквент исходного выражения

$$\forall xP(x) \supset \forall xQ(x).$$

Так как к псевдоформуле 1 можно применить правило $\forall$и, мы по пункту 3 алгоритма получаем псевдоформулу 2:

2. $P(v1) \supset Q(v1)$, V4 (значение метки объясняется далее).

При этом на псевдоформулу 1 вешается метка "V1", говорящая о том, что данная псевдоформула является цикловой, а также метка "V0", говорящая, что к данной формуле нет необходимости еще раз применять правило снятия квантора общности. (Отметим, что псевдоформула 2 отличается от того выражения под номером 2, которое содержится в построенном выводе.)

Из данных двух псевдоформул сделать дальнейшие шаги вывода по аналитическим правилам невозможно, а процедура унификации показывает, что подобных формул нет. Поэтому переходим к анализу последней цели. Ее анализ позволяет по пункту 6 получить в качестве еще одной посылки псевдоформулу вывода, помеченную цифрой 3:

3. $\forall x P(x)$, V1, V0 (значение меток было объяснено выше),

а последней целью сделать псевдоформулу

$\forall x Q(x)$.

Теперь мы вновь можем сделать некоторый шаг вывода, что по пункту 3 алгоритма дает нам псевдоформулу 4:

4. $P(v2)$.

При этом на псевдоформулу 3 ставятся метки "V1" и "V0". Отметим, что и в данном случае полученная по алгоритму псевдоформула 4 отличается от того, что мы в этом пункте имеем в окончательном варианте вывода.

Так как никаких дальнейших шагов вывода применить к псевдоформулам, содержащимся в выводе, нельзя, а процедура унификации показывает, что подобных формул нет, переходим к анализу последней цели. Она недостижима, но дает нам по пункту 9 алгоритма в качестве новой последней цели формулу

$Q(y)$, y – абс. огр.,

где "y" – новая свободная индивидная переменная.

Теперь мы попали в ситуацию, когда ни одного шага вывода сделать нельзя и в то же время процедура унификации показывает, что последняя цель (а ею является элементарная формула) не достигнута. Тогда по пункту 5 алгоритма данная цель со знаком отрицания переносится в последовательность псевдоформул вывода в качестве посылки, а последней целью становится $\perp_1$ – противоречие:

5. $\neg Q(y)$

противоречие $\perp_1$.

Вновь возникшая ситуация характеризуется следующим: ни одного шага вывода сделать нельзя, унификация показывает, что цель – противоречие – не достигается. Поэтому по пункту 14 в результате анализа псевдоформул вывода получаем новую цель. Ею становится антецедент псевдоформулы 2, так как только это выражение может породить новую цель. Итак,

P(v1), V4,

при этом на псевдоформулу 2 и полученную новую цель ставится метка "V4".

Так как ни одного шага вывода по-прежнему сделать нельзя, то переходим к анализу последней цели и с помощью унификации устанавливаем, что данная цель является достижимой, а именно: псевдоформула цели P(v1) подобна псевдоформуле 4 вывода. При установлении подобия определяем, что v1 = v2. Из этого равенства определяем, что вре́менной переменной v2 должна быть приписана в качестве значения переменная v1, т.е. во всех псевдоформулах вывода и целей переменная v2 должна быть заменена на переменную v1. Это означает, согласно пункту 12, что псевдоформула 4 в выводе должна быть заменена на P(v1). Итак,

4. P(v1).

Так как цель P(v1) достигнута, то она из целей убирается, а новой целью становится предыдущая цель $\perp_1$ – противоречие:

противоречие $\perp_1$.

Кроме того, с формулы 2 убирается метка V4.

Теперь, согласно пункту 3, к которому нас отсылает пункт 12, мы можем по правилу $\supset$и получить из 2 и 4 псевдоформулу

6. Q(v1).

Последней целью является противоречие, которое, согласно процедуре унификации достижимо в силу того, что в выводе имеются две противоречащие друг другу псевдоформулы. Это псевдоформулы 5 и 6. При этом вре́менная переменная v1 принимает в качестве значения свободную индивидную переменную "y". Тогда, согласно пункту 12, мы перестраиваем вывод, который теперь будет иметь вид:

1. $\forall x(P(x) \supset Q(x)$
2. $P(y) \supset Q(y)$
3. $\forall xP(x)$
4. $P(y)$
5. $\neg Q(y)$
6. $Q(y)$

и применяем правило ¬в, что даст нам псевдоформулу

7. $\neg\neg Q(y)$.

При этом псевдоформулы 5 и 6 в дальнейших шагах вывода уже принимать участия не могут, что и было отмечено скобкой. Цель – противоречие – из последовательности целей устраняется. Последней целью теперь становится псевдоформула

$$Q(y).$$

Эта цель легко достигается, так как к псевдоформуле вывода 7 должно быть применено правило ¬и, что дает:

8. $Q(y)$.

Достижимость псевдоформулы $Q(y)$ означает, согласно пункту 12, что из последовательности целей устраняется как сама цель $Q(y)$, так и предшествующая ей псевдоформула $\forall x Q(x)$. Последняя псевдоформула при этом переносится в вывод, т.е.:

9. $\forall x Q(x)$ – у – абс.,

а новой целью становится псевдоформула

$$\forall x P(x) \supset \forall x Q(x).$$

Дальнейшие шаги построения вывода по описанному алгоритму очевидны и мы их приводить не будем.

3. Метатеорема о корректности алгоритма

3.1. *Корректность алгоритма для классической логики высказываний*

С целью доказательства корректности работы предложенного алгоритма на уровне логики высказываний мы выделим из него те пункты, которые относятся к пропозициональной логике, и еще раз кратко опишем их.

Процедура 0. Итак, идея алгоритма состоит в том, что по некоторому заданному утверждению о выводимости создаются две последовательности формул (два стека). Первая последовательность представляет собой собственно формулы вывода, вторая – формулы-цели. Изначально эти две последовательности формируются следующим образом:

а) если необходимо осуществить вывод некоторой формулы B из множества посылок $A_1,..., A_k$, то формулы $A_1,..., A_k$ составляют исходный список формул вывода, а формула B помещается в список целей. При этом формула B считается главной целью и если эта цель достигается, то вывод считается успешно законченным;

б) если необходимо осуществить вывод B из пустого множества посылок, т.е. требуется осуществить доказательство B, то список вы-

вода является пустым, а формула B помещается в список целей. Естественно, эта формула и является главной целью.

Далее автомат работает следующим образом.

Процедура 1 состоит в нахождении одной или двух формул в множестве формул вывода, к которым можно применить соответствующие правила исключения логических связок. Если такие формулы обнаруживаются, то вывод (множество формул вывода) пополняется результатом применения данного правила.

Используются следующие аналитические *правила вывода*:

$$\&\text{и}\ \frac{\text{вывод}\quad *A \,\&\, B}{A,\ B} \qquad \supset\text{и}\ \frac{\text{вывод}\quad *A \supset B,\ A}{B} \qquad \neg\text{и}\ \frac{\text{вывод}\quad *\neg\neg A}{A} \qquad \vee\text{и}\ \frac{\text{вывод}\quad *A \vee B,\ \neg A}{B}$$

Здесь звездочкой показаны формулы, на которые вешается метка, говорящая о том, что данные формулы не могут: 1) еще раз использоваться для осуществления этих же выводов, 2) быть формулами, порождающими новые цели. Снятие этой метки позволяет вновь использовать эти формулы в обоих качествах.

Процедура 2. Если последней целью является некоторая формула, то она считается достижимой просто по факту наличия в выводе графически равной ей формулы. Если же последней целью является противоречие $\perp$, то оно считается достигнутым, если в выводе содержатся формулы вида A и $\neg$A. Достигнутая цель из списка целей устраняется, а очередной целью становится предыдущая.

Процедура 3. Данная процедура начинает использоваться в том случае, когда уже применена процедура 1, целью является некоторая формула, и при этом она не достигнута. В этом случае в зависимости от вида очередной (последней) цели поступаем следующим образом.

3.1. Если целью является формула вида $A \supset B$, то B становится новой целью, а A помещается в формулы вывода в качестве посылки.

3.2. Если целью является формула вида A & B, то A и B становятся новыми целями. При этом сначала в обязательном порядке требуется достигнуть цель A и только после ее достижения – цель B.

3.3. Если целью является формула вида $A \vee B$, то подцелями становятся A или B. Если ни одна из этих подцелей не достигается, то в качестве новой посылки берется формула $\neg(A \vee B)$, бывшие цели A и B убираются, а новой целью становятся:

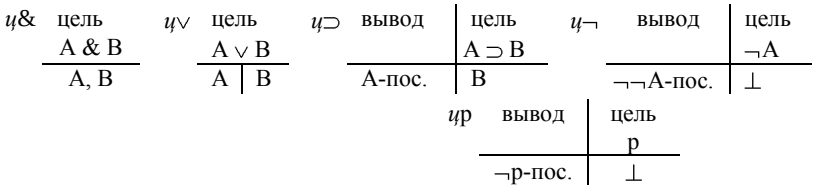
$\perp$, если на цели $A \vee B$ не стояла метка +,

$\neg A$ и $\neg B$, если на цели $A \vee B$ стояла метка +.

3.4. Если целью является пропозициональная переменная p, то в качестве новой посылки берется формула $\neg p$, а целью становится $\perp$.

3.5. Если целью является формула вида ¬A, где A – произвольная формула, то в качестве новой посылки берется формула ¬¬A, а новой целью становится ⊥.

Кратко данную процедуру можно представить следующими алгоритмическими правилами поиска доказательства – *правилами целей* (*ц*-правила):



Процедура 4. Эта процедура состоит в том, что по формулам, содержащимся во множестве формул вывода, на которых не стоят метки, определяемые описанной выше процедурой 1, а также самой рассматриваемой сейчас процедурой, порождаются новые цели. Выполняется данная процедура в том случае, когда все возможные указанные в процедуре 1 правила вывода применены, последней целью в последовательности является ⊥ (противоречие), но при этом данная цель не достигнута. Какие именно вводятся подцели, определяется видом содержащихся в выводе формул:

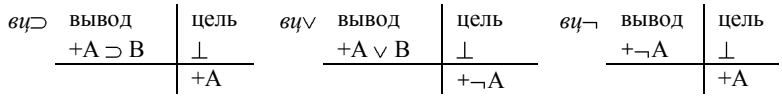
4.1. Если в выводе имеется импликативная формула вида A ⊃ B, то в качестве подцели берется A,

4.2. Если в выводе имеется дизъюнктивная формула вида A ∨ B, то в качестве подцели берется ¬A,

4.3. Если в выводе встречается формула ¬A, то в качестве подцели берется формула A.

Никакие другие формулы не служат источником новых целей, поскольку к ним автоматически применяется процедура 1. В этом смысле мы не интересуемся формулами вида A & B и ¬¬A, так как они никогда не участвуют в качестве формул, порождающих новые цели.

Будем называть эти алгоритмические правила поиска вывода *правилами взятия целей* (*вц*-правилами). Их можно представить в следующей таблице:



Здесь знаком "+" показана метка, которая ставится при выполнении данной процедуры на формулы вывода, которые породили новые цели, и на сами эти цели. Данная метка запрещает использовать соот-

ветствующие формулы вновь в качестве формул, порождающих цели. Однако если порожденная этими формулами цель достигнута, то с формул вывода данная метка снимается и они вновь могут служить в качестве формул, порождающих цели. В правиле *вц*¬ формула A не должна иметь вид формулы ¬B и не является пропозициональной переменной.

Процедура 5. Каждый случай достижения некоторой цели сигнализирует о необходимости выполнить некоторые шаги вывода. В большинстве случаев необходимо в автоматическом режиме выполнить соответствующее правило введения логической связки. Более детально это описывается следующим образом:

5.1. Если непосредственной надцелью является формула вида A & B, а последними в списке целей непосредственными подцелями этой формулы являются формулы A и B, и если формула A достигнута, то это служит сигналом необходимости перейти к достижению цели B. В том же случае, когда достигнута и цель B, т.е. достигнуты обе подцели, то это служит сигналом к осуществлению в выводе правила "&в" и получению тем самым в последовательности вывода формулы A & B. Обращаем внимание, что это означает достижение и цели A & B.

5.2. Если непосредственной надцелью является формула вида A ∨ B, а последними в списке целей являются формулы A, B, и если одна (любая) из этих целей достигнута, то это служит сигналом к осуществлению в выводе правила "∨в", т.е. включению в последовательность вывода формулы A ∨ B. Тем самым достигается и цель A ∨ B.

5.3. Если непосредственной надцелью является формула вида A ⊃ B, а последней в списке целей непосредственной ее подцелью является формула B, и она достигнута, то это служит сигналом к осуществлению в выводе правила "⊃в", т.е. включению в последовательность вывода формулы A ⊃ B. Тем самым достигается и цель A ⊃ B.

5.4. Если непосредственной надцелью является формула A, где A – формула произвольного вида, а непосредственной ее подцелью является цель ⊥, и цель ⊥ достигнута, то это служит сигналом применения в выводе правила "¬в". Это либо сразу же приводит к получению в выводе формулы A, либо она получается после применения правила "¬и". В любом из этих случаев цель A будет достигнута.

Нижеследующая таблица показывает, как применяются правила введения логических связок.

&в вывод	цели
	A & B
	^A, ^B
A & B	

∨в вывод	цели	
	A ∨ B	
	^A	^B
A ∨ B		

⊃в вывод	цели
	A ⊃ B
	^B
A ⊃ B	

Здесь знаком "^" помечены достигнутые цели. В правиле ¬в представлены три случая: 1) если A ≕ p (пропозициональной переменной), то B ≕ ¬¬p; 2) если A ≕ C, где C – сложная формула и C не начинается со знака отрицания (¬), то B ≕ ¬¬C, 3) если A ≕ C и формула C начинается со знака отрицания (¬), то B ≕ ¬¬¬C.

3.2. *Теорема о корректности*

Теперь мы можем сформулировать и доказать метаутверждение о конечности работы алгоритма:

> *При работе алгоритма не возникают циклы, а количество формул, участвующих в порождении новых целей, последовательно уменьшается до нуля.*

Ясно, что доказательства этого утверждения достаточно, чтобы показать конечность работы алгоритма. Так как любая доказываемая формула имеет конечную длину, а в выводе и целях могут появляться только подформулы доказываемой формулы или отрицания этих подформул, то количество формул, порождающих цели (сокращенно – "*фпц*"), должно быть конечным. Поэтому алгоритмический вывод, уходящий в бесконечность, может возникнуть только при неоднократном использовании одних и тех же формул в качестве *фпц*, а это возможно при возникновении в алгоритмическом выводе циклов. Отсюда вытекает, что отсутствие циклов и последовательное уменьшение количества *фпц* до нуля должно гарантировать остановку работы алгоритма.

И еще одно предварительное замечание. Мы будем доказывать теорему для случая, когда алгоритм строит доказательство некоторой теоремы B. Это нисколько не уменьшает общности наших рассуждений, так как случай вывода формулы B из посылок $A_1,..., A_k$, в силу теоремы дедукции, всегда можно свести к доказательству некоторой общезначимой формулы.

Доказательство ведется методом математической индукции по *n* – функциональной сложности доказываемой формулы Ф. Обычно под сложностью формулы понимается количество логических связок, входящих в нее. Для нашего же случая важна не сложность формулы Ф, а ее функциональная сложность. Под последним термином мы по-

нимаем количество *фпц*, которые реально выступают в этом качестве при попытке доказательства Ф. Сделаем соответствующие пояснения.

Рассмотрим доказательство формулы $(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$. Обычная сложность этой формулы равна 6, так как именно такое количество логических связок встречается в ней. Функциональная же сложность данной формулы равна 0, так как при доказательстве этого выражения ни одна формула не будет использоваться в качестве *фпц*. Действительно, рассмотрим следующее доказательство:

вывод		цели	анализ
0.		$(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$	Процедура 0
1.	$p \supset (q \supset r)$ – пос.	$(p \supset q) \supset (p \supset r)$	*и*$\supset$ к 0
2.	$p \supset q$ – пос.	$p \supset r$	*и*$\supset$ к 1
3.	p – пос.	r	*и*$\supset$ к 2
4.	$q \supset r$		$\supset$и к 1, 3
5.	q		$\supset$и к 2, 3
6.	r		$\supset$и к 4, 5, цель 3 достигнута
7.	$p \supset r$		$\supset$в к 6, цель 2 достигнута
8.	$(p \supset q) \supset (p \supset r)$		$\supset$в к 7, цель 1 достигнута
9.	$(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$		$\supset$в к 8, цель 0 достигнута

С другой стороны, рассмотрим доказательство формулы $\neg(p \vee q) \supset \neg p$. Обычная сложность этой формулы равна 4, функциональная же сложность будет равна 1, так как именно такое количество формул будет использоваться в качестве *фпц*. Действительно:

вывод		цели	анализ
0.		$\neg(p \vee q) \supset \neg p$	Процедура 0
1.	$+\neg(p \vee q)$	$\neg p$	*и*$\supset$ к 0
2.	p – пос.	$\perp$	*и*$\neg$ к 1
0.1.		$+p \vee q$	*ви*$\neg$ к 2
0.2.		p	*и*$\vee$ к 0.1., цель 0.2. достигнута
3.	$p \vee q$		$\vee$в к 2, цель 0.1. и 2. достигнуты
4.	$\neg p$		$\neg$в к 1, 3, цель 1 достигнута
5.	$\neg(p \vee q) \supset \neg p$		$\supset$в к 4, цель 0 достигнута

Перед началом доказательства отметим несколько моментов.

Во-первых, нас не будут далее интересовать формулы вида A & B и $\neg\neg$A, так как они никогда не участвуют в качестве *фпц*. Иначе говоря, нас интересуют только случаи появления в выводе формул вида $A \supset B$, $A \vee B$ и $\neg$A (здесь A – сложная формула), которые могут служить источником новых целей.

Во-вторых, из правил порождения целей видно, что каждое из них применяется только тогда, когда в целях появляется указание на взятие противоречия ($\perp$). Это вводит естественное разделение алгоритмиче-

ского вывода на блоки, каждый из которых начинается введением по этим правилам новой цели и заканчивается появлением цели $\perp$. При этом, если цель $\perp$ достигнута, то все ее надцели в блоке, являющиеся формулами, достигаются автоматически путем применения *правил целей* снизу вверх, порождая соответствующие формулы в выводе.

В-третьих, мы будем фиксировать только правила, связанные с целями, так как правила, связанные с формулами вывода, осуществляются автоматически.

Доказательство.

I. Если формула Ф алгоритмически доказана, то алгоритмический вывод не имел циклов и остановился (стоп).

II. Допустим, что формула Ф алгоритмически не доказывается.

а) $n = 0$.

Рассмотрим здесь два подслучая: 1) когда сложность формулы Ф и в обычном смысле равняется 0 и 2) когда сложность формулы А в обычном смысле больше 0.

а1) В первом подслучае формула Ф графически совпадает с некоторой переменной, скажем, переменной р (Ф $\rightleftharpoons$ р). Тогда алгоритмический вывод имеет следующий вид:

вывод		цели	анализ
0.		р	Процедура 0
1.	$\neg$р – пос.	$\perp$	*ц*р к 0
			стоп

Алгоритм закончил работу, так как применить *правила вывода* нельзя, применить *правила взятия целей* нельзя, последней целью является $\perp$ (противоречие), которое не достигнуто и, кроме того, к этой цели ц-правила не применяются. Алгоритмический вывод циклов не содержал.

а2) Во втором подслучае обычная сложность формулы Ф больше 0. Тогда алгоритмический вывод выглядит следующим образом (здесь и далее приводится общая схема вывода):

вывод		цели	анализ
0.		Ф	Процедура 0
1.	A_1	B_1	*ц*-правило к 0
2.	A_2	…	…
…	…	$B_м$	*ц*-правило
к	$A_к$ – пос	$\perp$	либо *ц*р, либо *ц*$\neg$ к $B_м$
			стоп

В этом выводе посылка $A_к$ – либо некоторая переменная, либо отрицание этой переменной. Формулы A_1, A_2,… не могут быть сложными формулами, которые либо сами является *фпц*, либо в дальней-

шем могут породить формулу, которая будет использована как *фпц*, так как, по нашему допущению, в рассматриваемом случае таких формул нет.

Цель $\perp$ не достигнута, так как в противном случае формула Ф была бы доказана, а, по условию, она алгоритмически не доказуема. Все возможные правила вывода применены, все возможные ц-правила тоже применены. Применение *вц*-правил невозможно, так как *фпц* нет, последней целью является $\perp$. Как видим, теорема в этом случае верна, т.е. циклов нет и работа алгоритма останавливается.

б) Рассмотрим теперь случай, когда функциональная сложность Ф = 1, т.е. в выводе имеется ровно одна формула, по которой можно брать новые цели. Схема алгоритмического вывода тогда имеет несколько форм, которые зависят от вида единственной *фпц*. Напомним, что это могут быть формулы вида $A \supset B$, или $A \vee B$, или $\neg A$, где A не является пропозициональной переменной и не имеет вида формулы $\neg B$. Рассмотрим каждый из этих случаев.

б1) *Фпц* графически совпадает с $A \supset B$. Схема алгоритмического вывода имеет вид:

вывод		цели	анализ
0.		Ф	Процедура 0
1.	A_1	B_1	*ц*-правило к 0
2.	A_2	B_2	…
…	…	…	…
r.	$+ A \supset B$	…	…
…	…	$\perp_1$	*ц*-правило
s.	…	$+A$	*вц*$\supset$ к r
…	…	…	*ц*-правило
…	…	$B_м$	…
к.	$A_к$ – пос.	$\perp_2$	либо *ц*р, либо *ц*$\neg$ к $B_м$
			стоп

В этом выводе посылка $A_к$ – либо некоторая переменная, либо отрицание этой переменной. Остальные формулы A_1, A_2,… не могут быть сложными формулами, которые либо сами являются *фпц*, либо в дальнейшем могут породить формулу, которая будет использована как *фпц*, так как, по нашему допущению, в рассматриваемом случае такой формулой является единственная формула $A \supset B$, стоящая на r-ом месте в выводе.

Цель $\perp_2$ либо достижима, либо нет.

Если она недостижима, то алгоритм останавливает свою работу, так как все возможные *правила вывода* и *правила целей* уже применены, а так как $A \supset B$ является единственной *фпц*, то выбор новых целей после появления цели $\perp_2$ невозможен.

Если же цель $\perp_2$ достижима, то достижима и цель $B_м$, а следовательно, и цель А. В этом случае с формулы $A \supset B$ снимается метка +, но ставится метка *, так как теперь возможно осуществить вывод формулы В из $A \supset B$ и А. Последняя метка не позволяет использовать формулу $A \supset B$ вновь для порождения целей. Поэтому число формул *фпц* уменьшается на единицу и становится = 0.

При этом цель $\perp_1$ недостижима, так как в противном случае формула Ф была бы алгоритмически доказана, что противоречит нашему допущению о ее недоказуемости. Стоп. Действительно, при этих условиях случай б1) сводится к рассмотренному выше случаю а).

б2) Случай, когда единственная *фпц* графически совпадает с формулой $A \vee B$, аналогичен только что рассмотренному, а потому на нем специально останавливаться не будем.

б3) Единственная *фпц* графически совпадает с формулой $\neg A$, где А не является пропозициональной переменной и не имеет вида формулы $\neg B$.

Схема алгоритмического вывода тогда имеет следующий вид:

вывод		цели	анализ
0.		Ф	Процедура 0
1.	A_1	B_1	*ц*-правило к 0
2.	A_2	B_2	...
…	…	…	…
r.	$+\neg A$	…	…
…	…	$\perp_1$	*ц*-правило
s	…	+A	*вц*$\neg$ к r
…	…	…	*ц*-правило
…	…	$B_м$	…
к.	$A_к$ – пос.	$\perp_2$	либо *ц*р, либо *ц*$\neg$ к $B_м$
…			стоп

В этом выводе посылка $A_к$ – либо некоторая переменная, либо отрицание этой переменной. Остальные формулы A_1, A_2,… не могут быть сложными формулами, которые либо сами являются *фпц*, либо в дальнейшем могут породить формулу, которая будет использована как *фпц*, так как, по нашему допущению, в рассматриваемом случае такой формулой является единственная формула $\neg A$, стоящая на r-ом месте в выводе.

Цель $\perp_2$ либо достижима, либо нет.

Если цель $\perp_2$ достижима, то достижимы последовательно цель $B_м$ и цель А. Но тогда в выводе будут присутствовать одновременно формулы $\neg A$ и А, т.е. будет противоречие, в силу чего станет достижимой цель $\perp_1$, а потому и формула Ф станет доказуемой, что противоречит нашему условию об алгоритмической недоказуемости Ф.

Итак, показано, что цель $\perp_2$ должна быть недостижимой. Но тогда недостижима и цель А, а потому снять с формулы $\neg$А метку + нельзя. Таким образом, формула $\neg$А не может более участвовать в порождении новых целей. Тем самым и в этом случае число *фпц* уменьшается на единицу и становится равным 0. Исходя из этого, мы приходим к выводу, что все *правила вывода* и *правила целей* применены, в выводе не осталось ни одной формулы, к которой можно было бы применить *правило выбора целей*. Следовательно, стоп.

Как видим, во всех рассмотренных случаях циклов нет и алгоритм останавливает свою работу, т.е. теорема верна.

Индуктивное предположение.

Допустим, что теорема верна для случая, когда функциональная сложность формулы Ф меньше, чем *n*. Покажем, что она верна и когда количество *фпц* = *n*.

в) Рассмотрим случай, когда функциональная сложность Ф = *n*.

Схема алгоритмического вывода тогда имеет несколько форм, которые зависят от последовательности использования *фпц*. Собственно говоря, все они сводятся к двум случаям, которые определяются, чем является вторая (считая сверху вниз) *фпц* в выводе – посылкой или она получена из предыдущих по одному из аналитических правил.

1. $\dfrac{+A(C_1)}{+Д(C_2) - пос.}$ 2. $\dfrac{+A(C_1)}{+Д(C_2)}$

Здесь под записью "$+A(C_1)$" имеется в виду формула А, которая выступает в качестве первой (сверху вниз) *фпц*, а C_1 есть та часть А, которая как цель будет порождена формулой А при построении алгоритмического вывода. Соответствующее значение имеет и запись второй (сверху вниз) *фпц* "$+Д(C_2)$". При этом формулы А и Д могут быть либо импликативными (главный знак – "$\supset$"), либо дизъюнктивными (главный знак – "$\vee$"), либо сложными формулами, начинающимися со знака отрицания (главный знак – "$\neg$").

в1) Алгоритмический вывод в общем случае имеет следующий вид:

вывод		цели	анализ
0.		Ф	Процедура 0
1.	A_1	B_1	*ц*-правило к 0
…	…	…	*ц*-правило
r.	$+A(C_1)$	…	…
…	…	$\perp_1$	…
…	…	$+C_1$	*вц*-правило к r
…	…	…	*ц*-правило
…	…	B_{s-1}	…
s.	$+Д(C_2)$ – пос.	…	*ц*-правило к B_{s-1}
…	…	…	*ц*-правило

…	…	$\perp_2$	…
…	…	$+C_2$	*вц*-правило к s
…	…	…	*ц*-правило
…	…	B_m	…
к.	$A_к$ – пос.	$\perp_{n+1}$	либо *ц*р, либо *ц*¬ к $B_м$

Рассмотрим вывод, идущий после появления цели $\perp_2$. В этом выводе количество подформул или их отрицаний, взятых из формулы Ф, которые могут быть использованы в качестве *фпц*, меньше, чем во всем выводе. По индуктивному предположению это означает, что в этой части вывода по отношению к содержащимся здесь формулам *фпц* не возникали циклы. Таким образом циклы могут возникнуть только за счет формул *фпц* $+А(C_1)$ и $+Д(C_2)$. Рассмотрим последнее более внимательно.

Так как в выводе, идущем после появления цели $\perp_2$, циклов нет, то это означает, что построение алгоритмического вывода дошло до некоторой цели $\perp_{n+1}$ и либо происходит остановка действия алгоритма (стоп), так как цель $\perp_{n+1}$ недостижима, а все правила вывода и правила порождения новых целей были уже использованы (в этом случае теорема верна), либо цель $\perp_{n+1}$ достигнута и алгоритм начинает последовательно достигать надцели $\perp_{n+1}$. При этом цель $+C_2$ либо достигается, либо нет.

Если цель $+C_2$ не достигается, то с формулы $+Д(C_2)$ нельзя снять метку + и работа алгоритма останавливается. Теорема верна. Если же цель $+C_2$ достигается, то все будет зависеть от цели $\perp_2$, которая может быть введена либо непосредственно сразу же после применения к формуле B_{s-1} некоторого ц-правила и порождения посылки $+Д(C_2)$, либо опосредованно.

в1.1) Цель $\perp_2$ была введена непосредственно сразу после применения к формуле B_{s-1} ц-правила и порождения посылки $+Д(C_2)$. В этом случае (см. ц-правила) мы имеем: посылка $+Д(C_2)$ есть формула $\neg(К \vee L)$, формула B_{s-1} и формула $C_2 \subset (К \vee L)$.

Так как цель $+C_2 \subset (К \vee L)$ достигнута, то с нее и с посылки $+Д(C_2)$ снимается знак +. Однако в силу того, что теперь в выводе достигается противоречие ($\neg(К \vee L)$ и $(К \vee L)$), мы применяем правило ¬в и тем самым исключаем из вывода посылку $+Д(C_2)$. Она в дальнейших шагах вывода принимать участия уже не может. Но устраняя посылку $+Д(C_2)$, мы тем самым достигаем и цель B_{s-1} и цель $+C_1$. Последняя цель не может противоречить формуле $+А(C_1)$, ибо в таком случае было бы достижимо противоречие $\perp_1$ и формула Ф оказалась бы доказанной. Однако мы рассматриваем случай, когда формула Ф недоказуема. Таким образом, формула $+А(C_1)$ должна иметь вид: либо $(N \supset M)$, в этом случае C_1 есть N; либо $(N \vee M)$, в этом случае C_1

есть $\neg N$. В любом из этих случаев к формулам $A(C_1)$ и C_1 применяются правила: либо $\supset$и, либо $\vee$и. Тем самым на формулу $A(C_1)$, с которой снята метка +, ставится метка *, которая запрещает использовать данную формулу в качестве *фпц*.

Так как формула $A(C_1)$ являлась самой первой формулой, к которой можно было применить *вц*-правила, то дальнейшие наши шаги вывода по попытке доказать формулу Ф ее уже не будут затрагивать, а потому снять метку * никогда не удастся, но тем самым мы уменьшили число *фпц* по крайней мере на 1 и теорема верна.

в1.2) Цель $\perp_2$ была введена опосредованно. Это возможно (см. *ц*-правила) в следующих пяти случаях: если формула B_{s-1} графически совпадает (а) с формулой $(K \vee L) \supset P$, (б) с формулой $(K \supset L) \supset P$, (в) с формулой $\neg(K \vee L) \supset P$, (г) с формулой $\neg(K \supset L) \supset P$, (д) с формулой $\neg(K \,\&\, L) \supset P$. Тогда, соответственно, формула $+Д(C_2)$ будет графически равна либо $(K \vee L)$, либо $(K \supset L)$, либо $\neg(K \vee L)$, либо $\neg(K \supset L)$, либо $\neg(K \,\&\, L)$, а между целями B_{s-1} и $\perp_2$ появится по крайней мере промежуточная цель P. При этом формула $+C_2$ будет графически совпадать, соответственно, с $\neg K$, K, $(K \vee L)$, $(K \supset L)$, либо $(K \,\&\, L)$.

В случае (в), (г) и (д) достижение цели $+C_2$ означает получение в выводе противоречия. Тем самым цель $\perp_2$ будет достигнута, а, следовательно, по правилу $\neg$в посылка $+Д(C_2)$ вообще исключается из вывода. Но достижение цели $\perp_2$ означает, что будет достигнута и цель P, а тогда по правилу $\supset$в в выводе появятся формулы $\neg(K \vee L) \supset P$, либо $\neg(K \supset L) \supset P$, т.е. будет достигнута цель B_{s-1}, и тем самым достигается цель C_1.

Последняя цель не может противоречить формуле $+A(C_1)$, ибо в таком случае было бы достижимо противоречие и формула Ф оказалась бы доказанной. Однако мы рассматриваем случай, когда формула Ф недоказуема. Таким образом, формула $+A(C_1)$ должна иметь вид: либо $(N \supset M)$, в этом случае C_1 есть N; либо $(N \vee M)$, и в этом случае C_1 есть $\neg N$. В любом из этих случаев к формулам $A(C_1)$ и C_1 применяются правила: либо $\supset$и, либо $\vee$и. Тем самым на формулу $A(C_1)$, с которой снята метка +, ставится метка *, которая запрещает использовать данную формулу в качестве *фпц*.

Так как формула $A(C_1)$ являлась самой первой формулой, к которой можно было применить *вц*-правила, то дальнейшие наши шаги вывода по попытке доказать формулу Ф ее уже не будут затрагивать, а потому снять метку * никогда не удастся, но тем самым мы уменьшили число *фпц* по крайней мере на 1 и теорема верна.

В случае (а) и (б) достижение цели $+C_2$ означает, что с $+C_2$ и посылки $+Д(C_2)$ снимается метка +, но ставится метка *, так как теперь можно применить правила $\vee$и, либо $\supset$и. При этом цель $\perp_2$ до-

стигнута или не достигнута. Если эта цель не достигается, то стоп. Все правила применены, ни одной *фпц* нет, и мы находимся в состоянии противоречия. Тем самым теорема верна. Если же цель $\perp_2$ достигнута, то достигнута и надцель C_1, о которой верны все предыдущие утверждения. Таким образом, в случае в1) теорема верна.

в2) Пусть теперь формула $+Д(C_2)$ будет не посылкой, а получена из предыдущих формул по одному из правил вывода. Тогда алгоритмический вывод имеет следующий вид:

вывод		цели	анализ
0.		Ф	Процедура 0
1.	A_1	B_1	*ц*-правило к 0
…	…	…	*ц*-правило
r.	$+A(C_1)$	…	…
…	…	$\perp_1$	…
		$+C_1$	*вц*-правило к r
…	…	…	*ц*-правило
s.	$+Д(C_2)$	…	…
…	…	$\perp_2$	…
…	Р – пос.	…	…
…	…	$+C_2$	*вц*-правило к s
…	…	…	*ц*-правило
…	…	B_m	…
к.	$A_к$ – пос.	$\perp_{n+1}$	либо *ц*р, либо *ц*$\neg$ к $B_м$

Как и в случае в1), по индуктивному предположению, мы считаем, что в выводе, идущем после появления цели $\perp_2$, циклов нет. Это означает, что построение алгоритмического вывода дошло до некоторой цели $\perp_{n+1}$ и либо происходит остановка действия алгоритма (стоп), так как цель $\perp_{n+1}$ недостижима, а все правила вывода и правила порождения новых целей были уже использованы (в этом случае теорема верна), либо цель $\perp_{n+1}$ достигнута и алгоритм начинает последовательно достигать надцели $\perp_{n+1}$. При этом цель $+C_2$ либо достигается, либо нет. Если цель $+C_2$ не достигается, то с формулы $+Д(C_2)$ нельзя снять метку + и работа алгоритма останавливается. Теорема верна.

Допустим, что цель $+C_2$ достигается. Рассмотрим формулу $+Д(C_2)$. Она может иметь один из следующих видов (см. *вц*-правила): (а) $(К \vee L)$, (б) $(К \supset L)$, (в) $\neg К$. Тогда формула $+C_2$ должна быть, соответственно, $\neg К$, К, либо еще раз К.

В случае (в), при достижении цели $+C_2$, с $+C_2$ и $+Д(C_2)$ устраняется метка + и, следовательно, формула $+Д(C_2)$ вновь может служить *фпц*. В то же время в выводе образуется противоречие (формулы $\neg К$ и К), что приводит, с одной стороны, к устранению по-

сылки Р, а, с другой – к достижению цели $\perp_2$. Это, в свою очередь, влечет достижение цели C_1. При этом цель $\perp_1$ не достигается, так как иначе будет доказана формула Ф, которая по нашему допущению недоказуема.

Таким образом, формула $+A(C_1)$ должна иметь вид: либо $(N \supset M)$, в этом случае C_1 есть N; либо $(N \vee M)$, и в этом случае C_1 есть $\neg N$. В любом из этих случаев к формулам $A(C_1)$ и C_1 применяются правила: либо $\supset$и, либо $\vee$и. Тем самым на формулу $A(C_1)$, с которой снята метка +, ставится метка *, которая запрещает использовать данную формулу в качестве *фпц*.

Так как формула $A(C_1)$ являлась самой первой формулой, к которой можно было применить *вц*-правила, то дальнейшие наши шаги вывода по попытке доказать формулу Ф ее уже не будут затрагивать, а потому снять метку * никогда не удастся, но тем самым мы уменьшили число *фпц* по крайней мере на 1 и теорема верна.

В случае (а) и (б), при достижении цели $+C_2$, с $+C_2$ и $+Д(C_2)$ устраняется метка +, но так как теперь можно применить либо правило $\vee$и, либо правило $\supset$и, на формулу $Д(C_2)$ ставится метка *, которая запрещает использовать данное выражение в качестве *фпц*. При этом цель $\perp_2$ либо достигается, либо нет. Если она не достигается, то стоп. Все правила вывода применены, новых формул *фпц* нет и мы находимся в состоянии противоречия. Теорема верна. Если же цель $\perp_2$ достигается, то тогда достигается и цель C_1, о которой верно будет сказать, все, что говорилось в предыдущих трех абзацах. Итак, и в данном случае число *фпц* уменьшается по крайней мере на 1. Теорема верна. □

Таким образом, показано, что независимо от того, общезначима доказываемая формула Ф или нет, алгоритм всегда завершает работу в конечное число шагов. При этом, если алгоритм завершает работу без построения доказательства формулы Ф, то по форме доказательства мы получаем контрпример – распределение значений "истина" и "ложь" для пропозициональных переменных Ф, при которых данная формула принимает значение ложь.

Следовательно, если Ф недоказуема, то она не общезначима. Отсюда по контрапозиции получаем, что если формула Ф общезначима, то она доказуема, что и обосновывает семантическую полноту алгоритма. Более подробно последнее будет рассмотрено в следующей главе.

3.3. *Некоторые замечания*

В заключение данной главы рассмотрим некоторые особенности предложенного алгоритма.

Будем называть вывод *нормальным*, если никакая формула, которая *является большей посылкой* какого-либо правила удаления, не является в то же время *заключением* некоторого правила введения. Будем далее говорить, что вывод формулы В из множества гипотез Г удовлетворяет критерию *подформульности*, если каждая формула, входящая в вывод, является элементом множества подформул Г или элементом множества подформул В. Будем называть *полем поиска* вывода мощность множества формул, которые могут быть использованы как цели.

Одним из наиболее часто встречающихся аргументов, выдвигающихся в пользу точки зрения, рассматривающей натуральный вывод как непригодную для автоматизации теорию дедукции, является следующий – в системах натурального вывода нарушается свойство *нормальности*. В принципе это действительно так, если система натурального вывода просто сформулирована в виде совокупности правил введения и удаления логических связок. Ничто здесь не запрещает совершать операции, нарушающие критерий нормальности.

Заметим, что проблема нормальности связана с другим свойством вывода – подформульностью. Нормальный вывод обязательно обладает свойством подформульности. Как известно, при секвенциальном построении логики свойство подформульности нарушается при применении правила сечения, и особое место занимают теоремы об устранимости сечения, которые позволяют (если они доказаны) любой вывод с применением сечения перестроить в вывод без сечения. Очевидно, что с точки зрения автоматизации, наличие свойства подформульности является одним из ключевых требований, предъявляемых к выводу.

В принципе оба свойства, нормальности и подформульности, связаны с проблемой поиска вывода. Действительно, при выполнении этих условий поиск, по крайней мере потенциально, может быть алгоритмизирован, в то время как нарушение этих условий может привести к крайне неприятным для алгоритмизации фактам, таким как циклы в выводе (если ничто не запрещает получать некоторую формулу в выводе как результат правила введения и затем использовать ее как большую посылку в правиле удаления связки), или, при нарушении свойства подформульности, поле поиска может быть (в худшем случае) бесконечным. Последнее связано с наличием, например, в нашей формулировке правил натурального вывода, правил введения дизъюнкции и негласно принимающимся принципом введения допущений в вывод, согласно которому в вывод в качестве допущения можно ввести любую произвольную формулу.

Например, пусть мы должны вывести В из множества гипотез Г. Согласно формулировкам правил, в качестве результата применения

правила введения дизъюнкции к некоторой формуле А в нашем выводе может быть получена формула А $\vee$ С, где С – это произвольная формула (возможно не входящая ни в множество гипотез Г, ни в В). С другой стороны, если введение допущений в вывод никак не обусловлено, то при наличии упомянутого принципа введения допущений, последствия для алгоритмизации будут катастрофическими. Исследования в области автоматизации дедукции как раз и обходят стороной натуральный вывод именно в силу проблем, связанных с правилами введения, и, в первую очередь, правилом введения дизъюнкции.

Однако, заметим, что подобная критика натурального вывода, если и являлась обоснованной, то до 1991 года, когда Г. Шталмарк [1991] показал, что любой натуральный вывод может быть нормализован, следуя некоторым специальным процедурам. Таким образом, нападки на натуральный вывод с точки зрения его "ненормальности" сейчас уже являются некорректными. С точки зрения нормальности правила введения оказываются вне критики, оставляя, однако, в запасе у критиков натурального вывода еще одно орудие – *поле поиска*. Действительно, весьма характерно, что в немногочисленных работах по автоматизации натурального вывода обычно принимается следующий подход. Казалось бы совсем безобидное и *естественное* правило удаления дизъюнкции (так оно и формулируется в нашем построении) также считается затруднительным для процедур поиска и заменяется правилом, известным в истории логики под названием "рассуждение по случаям" (если В выведено из Г и А, и В выведено из Г и С, и при этом имеется вывод дизъюнкции А $\vee$ С из Г, то тогда В считается выведенным из Г). Отметим здесь лишь, что такая замена правила удаления дизъюнкции, конечно, не решает проблемы поиска, создавая одновременно дополнительные сложности для поиска.

Таким образом (по имеющейся у авторов информации), на сегодняшний день ситуация такова. Учитывая решенность проблемы нормализации натурального вывода при формулировке системы натурального вывода с правилом рассуждения по случаям, в принципе можно сформулировать некоторый базовый алгоритм, работающий на поле поиска, состоящем из множества подформул исходных формул, входящих в метаутверждение о требуемом доказательстве или выводе. Этот алгоритм может быть представлен, например, в духе последовательного перебора возможных вариантов (так называемый breadth first search) и отсеивания тупиковых ветвей в дереве поиска. В силу того, что множество формул поиска конечно и в силу конечности числа возможных ветвей в дереве поиска алгоритм всегда завершит свою работу, помечая тупиковый путь меткой "Нет" и путь, ведущий к решению, меткой "Да". Однако при таком подходе встает серьезная проблема возможной "недетерминированности" алгоритма

практически на каждом шагу в дереве поиска. Таким образом, актуальной является задача снятия (ликвидации) этой "недетерминированности" (по крайней мере, насколько это окажется возможным) путем введения дополнительных поисковых приемов. Именно эту задачу и решает представленный в данной работе подход.

Итак, мы демонстрируем, что сформулированный нами алгоритм решает следующие проблемы:

– нормализацию вывода (и, следовательно, проблему подформульности),

– максимально решает проблему "недетерминированности" поиска вывода.

Что касается первой из указанных проблем, то свойство нормальности достигается в принципе за счет особого механизма, обеспечивающего строго "детерминированное" применение правил введения логических связок совместно со специальным механизмом накладывания меток на формулы вывода, ограничивающего выбор новых целей и применение правил удаления связок.

Теперь, принимая во внимание, что выбор целей начинается с анализа формулы, требующей доказательства (или вывода), легко установить, что выполняется и свойство подформульности, так как теперь все формулы, попадающие в вывод, являются членами множества подформул этой исходной формулы или их отрицаний.

ГЛАВА III

ПОИСК ВЫВОДА В КЛАССИЧЕСКОЙ ЛОГИКЕ ПРЕДИКАТОВ

В данной главе будет рассмотрен алгоритм автоматического поиска логического вывода в натуральной системе субординатного вывода типа Куайна в классической логике предикатов. Натуральные системы типа Куайна, в отличие от натуральных исчислений типа Генцена, содержат *прямое* правило удаления квантора существования. Как следствие, в натуральных системах типа Куайна между посылками и заключением не имеет места отношение логического следования.

Мы будем основываться на кратком описании алгоритма поиска натурального вывода Prover для классической логики предикатов без предиката равенства, рассмотренного в предыдущей главе. Спецификой натуральной системы, в которой ведется доказательство, является использование абсолютно и относительно ограниченных переменных. В процессе поиска вывода Prover использует также сколемовские термы. В данной главе будет показано, что вопрос о семантической полноте для Prover'а решается положительно.

Алгоритм будет реализовывать работу в системе натурального вывода, которую мы называем системой "BMV" (Бочаров, Маркин, Войшвилло). Формулировка этой системы была дана в предыдущей главе. Однако здесь для целей алгоритмического доказательства теорем будет использоваться другая формулировка системы BMV, в силу чего нам потребуется еще раз ее описать.

1. Описание системы натурального вывода BMV

1.1. *Формулировка системы* BMV

Задается алфавит языка L классической логики предикатов:

1. x, y, z, x_1, y_1, z_1, x_2,... – бесконечный список индивидных переменных,
2. a, b, c, a_1, b_1, c_1, a_2,... – бесконечный список индивидных констант,
3. f, g, h, f_1, g_1, h_1, f_2,... – бесконечный список предметно-функциональных констант (функторов) различной местности,
4. P, Q, R, P_1, Q_1, R_1, P_2,... – бесконечный список предикаторных констант (предикаторов) различной местности,
5. &, $\vee$, $\supset$, $\neg$, $\forall$, $\exists$ – логические символы,

6. (,), , – технические символы.

В дальнейшем буквы α, β, γ, α_1, β_1, γ_1, α_2,… обозначают произвольные индивидные переменные, буквы k, k_1, k_2,… обозначают произвольные индивидные константы, буквы Ф, $Ф_1$, $Ф_2$,… обозначают произвольные функторы и, наконец, буквами П, $П_1$, $П_2$,… обозначаются произвольные предикаторы.

Определение 1.1.1. Термом является объект, обладающий следующими свойствами:

1. Произвольная индивидная переменная есть терм.
2. Произвольная индивидная константа есть терм.
3. Если Ф – *n*-местный функтор, а t_1, t_2,..., t_n – термы, то выражение $Ф(t_1, t_2,..., t_n)$ – терм.
4. Ничто иное не является термом.

Определение 1.1.2. Формулой является объект, обладающий следующими свойствами:

1. Если П – *n*-местный предикатор, а t_1, t_2,..., t_n – термы, то выражение $П(t_1, t_2,..., t_n)$ – формула.
2. Если А – формула, то $\neg$А – формула.
3. Если А и В – формулы, то (А & В), (А $\vee$ В), (А $\supset$ В) – формулы.
4. Если А – формула и α – индивидная переменная, то $\forall\alpha$А и $\exists\alpha$А – формулы.
5. Ничто иное не является формулой.

Формулы, определяемые пунктом 1, называются *элементарными*. Все остальные формулы называются *сложными*. Буквами А, В, С,… с индексами и без них обозначаются произвольные формулы.

Зададим несколько стандартных определений, необходимых для формулировки правил исчисления. В выражениях $\forall\alpha$А и $\exists\alpha$А формула А называется *областью действия* кванторов $\forall$ и $\exists$, взятых по индивидной переменной α. Переменная α входит в А, если в А есть хотя бы одно вхождение α.

Вхождение индивидной переменной α называется *связанным*, если α расположена непосредственно справа от квантора или α находится в области действия квантора, взятого по α. Остальные вхождения α называются *свободными*. Индивидная переменная α является *свободной* (*связанной*) в формуле А, если она имеет хотя бы одно свободное (связанное) вхождение в формуле А.

Вхождение квантора в выражениях $\forall\alpha$А и $\exists\alpha$А называется *вырожденным*, если α не является свободной переменной в А. Напри-

мер, вхождение квантора $\forall$ по переменной x в формуле $\forall x \exists y P(y)$ вырождено. Вырождено первое (слева направо) вхождение квантора $\exists$ по переменной z в $\exists z \exists z R(z, z)$.

Пусть t – терм, α – индивидная переменная. Тогда выражение вида $A(\alpha/t)$ обозначает результат правильной подстановки в формулу $A(\alpha)$ вместо всех свободных вхождений α терма t. Подстановка $A(\alpha/t)$ считается *правильной*, если ни одна переменная, входящая в t, не окажется связанной на местах, где терм t появился в результате подстановки. Частным случаем $A(\alpha/t)$ является выражение $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$. Данное выражение обозначает результат правильной подстановки переменной β вместо переменной α. При этом $\{\gamma_1,\ldots, \gamma_n\}$ обозначает множество (возможно, пустое) свободных переменных в A и $\beta \notin \{\gamma_1,\ldots, \gamma_n\}$.

Правила системы BMV (BMV-правила) полностью совпадают с теми правилами, которые были заданы в предыдущей главе, а потому мы не будем их еще раз формулировать. Отметим только, что в правилах $\supset$в и $\neg$в формула C является *последней неисключенной* посылкой. В правилах $\forall$в и $\exists$и запись "β – абс.; $\gamma_1,\ldots, \gamma_n$ – огр." означает, что β абсолютно ограничена в выводе, а переменные $\gamma_1,\ldots, \gamma_n$ – относительно ограничены. В дальнейшем для краткости мы вместо "β относительно ограничена в выводе" будем писать "β ограничена", а вместо "β абсолютно ограничена в выводе" – "β абсолютна". Переменные, не ограниченные абсолютно в выводе, будут называться *неабсолютными* (содержательный смысл понятий "абсолютно ограниченная переменная" и "относительно ограниченная переменная" см. в работе Бочарова и Маркина [1994, с. 133-134]).

Определение 1.1.3. Выводом в системе BMV (BMV-выводом) называется непустая конечная последовательность формул, удовлетворяющая следующим условиям:

1. Каждая формула есть либо посылка, либо получена из предыдущих по одному из BMV-правил;
2. Если в выводе применялись правила $\supset$в и $\neg$в, то все формулы, начиная с последней посылки и вплоть до результата применения данного правила, исключаются из дальнейших шагов вывода;
3. Ни одна индивидная переменная в выводе не ограничивается абсолютно более одного раза;
4. Ни одна переменная не ограничивает в выводе сама себя.

Доказательство есть BMV-вывод из пустого множества посылок.

Определение 1.1.4. Завершенным BMV-выводом называется BMV-вывод, в котором никакая переменная, абсолютно ограни-

чивавшаяся в выводе, не встречается свободно ни в неисключенных посылках, ни в заключении. Завершенное доказательство есть завершенный BMV-вывод из пустого множества посылок.

Покажем необходимость вводимых ограничений. Требование, чтобы в правилах $\supset$в и $\neg$в формула C была последней неисключенной посылкой, обосновывается следующим выводом:

1	A	пос.
2	B	пос.
…	….	…
n	C	BMV-правило
n+1	$A \supset C$	$\supset$в к n

В данном рассуждении мы от вывода $A, B \vdash C$ по правилу $\supset$в перешли к выводу $\vdash A \supset C$. Однако такой переход семантически неверен. Аналогично обосновывается данное требование в формулировке правила $\neg$в.

Что касается запрета на самоограничение переменной, которого нет у Куайна [1950], то он обосновывается невозможностью вывода в нашей системе формулы $\exists x \forall y A(x, y)$ из $\forall y \exists x A(x, y)$ (см. Бочаров и Маркин [1994, с. 136]). Обоснование требования не ограничивать абсолютно некоторую переменную более одного раза, а также требования наличия завершенного вывода можно найти в работе Куайна [1950, p. 97-99]. С другой стороны, можно показать, что в завершенном выводе заключение всегда следует из своих посылок. В качестве примеров приведем завершенные BMV-выводы $\neg(A \vee B) \vdash \neg A \,\&\, \neg B$ и $\neg \exists x P(x) \vdash \forall x \neg P(x)$, которые нам потребуются при формулировке алгоритма поиска вывода.

1.	$\neg(A \vee B)$	пос.
2.	A	пос.
3.	$A \vee B$	$\vee$в к 2
4.	$\neg A$	$\neg$в к 1, 3
5.	B	пос.
6.	$A \vee B$	$\vee$в к 5
7.	$\neg B$	$\neg$в к 1, 5
8.	$\neg A \,\&\, \neg B$	&в к 4, 7

1.	$\neg \exists x P(x)$	пос.
2.	P(x)	пос.
3.	$\exists x P(x)$	$\exists$в к 2
4.	$\neg P(x)$	$\neg$в к 1, 3
5.	$\forall x \neg P(x)$	$\forall$в к 4, x – абс.

Так как наш алгоритм будет автоматически строить доказательство теорем именно в системе BMV, необходимо прежде всего убедиться в корректности самой этой системы, т.е. необходимо показать ее непротиворечивость и полноту. Что касается проблемы полноты BMV, то она решается достаточно просто при ее сравнении с системой K, описанной Мендельсоном [1976].

Схемы аксиом К:

(1) Схемы аксиом пропозициональной логики;
(2) $\forall\alpha A(\alpha) \supset A(t)$, где t есть терм, свободный для α в $A(\alpha)$;

При этом терм t называется *свободным для переменной* α в формуле A, если никакое свободное вхождение α не лежит в области действия никакого квантора $\forall\beta$, где β – переменная, входящая в t.

(3) $\forall\alpha(A \supset B) \supset (A \supset \forall\alpha B)$, если A не содержит свободных вхождений α.

Задаются следующие два правила вывода:

$$\supset\text{и}\quad \frac{A \supset B,\ A}{B} \qquad\qquad \text{Gen}\quad \frac{\vdash A}{\vdash \forall\alpha A(\alpha)}$$

Определение 1.1.5. Доказательством формулы A в системе К является непустая конечная последовательность формул, каждая из которых является либо аксиомой, либо получена из предыдущих по одному из правил вывода.

1.2. *Семантическая полнота системы* BMV

Доказательство теоремы о семантической полноте BMV состоит из двух шагов. Вначале покажем, что все дедуктивные принципы системы К имеют место в системе BMV. Затем, используя теорему о семантической полноте К (см. Мендельсон [1976, с. 78]), мы получаем теорему о семантической полноте BMV.

Лемма 1.2.1. Система К *включается по множеству доказуемых теорем в систему* BMV, *т.е. в* BMV *справедливы все дедуктивные принципы* К.

С целью доказательства леммы рассмотрим схемы аксиом К. Что касается схем аксиом пропозициональной логики, то показано, что пропозициональный фрагмент системы BMV семантически полон (см. Бочаров и Маркин [1994, с. 129]).

Аксиомная схема (2) доказуема в BMV следующим образом:

1.	$\forall\alpha A(\alpha)$	пос.
2.	$A(\alpha/t)$	$\forall$и к 1
3.	$\forall\alpha A(\alpha) \supset A(\alpha/t)$	$\supset$в к 2

Обратим внимание на взаимоотношение между определением правильной подстановки в BMV и понятием "терм t свободен для переменной α в формуле A" в формулировке аксиомы (2) системы К.

Сравнение последнего понятия и вышеприведенного определения правильной подстановки показывает их синонимичность. Таким образом, можно говорить, что формула $A(\alpha)$, которая встречается в формулировке аксиомы (2) (в этом случае на нее наложено ограничение, что терм t свободен для переменной α в формуле A), и формула $A(\alpha/t)$, которая встречается в заключении вышеприведенного вывода в системе BMV (в этом случае на нее было наложено требование правильной подстановки на шаге 2), есть одна и та же формула.

Аксиомная схема (3) доказуема в BMV следующим образом:

1.	$\forall\alpha(A \supset B)$	пос.
2.	$A \supset B$	$\forall$и к 1
3.	A	пос.
4.	B	$\supset$и к 2, 3
5.	$\forall\alpha B$	$\forall$в к 4, α – абс.
6.	$A \supset \forall\alpha B$	$\supset$в к 5
7.	$\forall\alpha(A \supset B) \supset (A \supset \forall\alpha B)$	$\supset$в к 6

Данный вывод является завершенным, так как, по условию, абсолютно ограниченная переменная α не входит свободно в A, а значит, и в заключение $\forall\alpha(A \supset B) \supset (A \supset \forall\alpha B)$.

Что касается правила $\supset$и, то это правило имеется в системе BMV.

Рассмотрим теперь Gen: если A доказуема в K, то $\forall\alpha A$ доказуема в K.

Допустим, что A доказуема в BMV, т.е. $\vdash A$. Значит, существует завершенное доказательство формулы A в BMV. Рассмотрим переменную α. Если α не входит в A, или входит в A связано, то $\vdash A \equiv \forall\alpha A$. Значит, существует завершенное доказательство $\forall\alpha A$ в системе BMV, т.е. $\vdash \forall\alpha A$. Если же α входит в A свободно, то, в силу завершенности доказательства $A(\alpha)$, т.е. $\vdash A(\alpha)$: (1) переменная α не была абсолютно ограничена в BMV-доказательстве формулы $A(\alpha)$ и (2) ни одна переменная, абсолютно ограниченная в доказательстве, не входит свободно в A. Следовательно, так как $A(\alpha)$ не посылка, а доказуемая формула, мы из $\vdash A(\alpha)$ получаем $\vdash \forall\alpha A(\alpha)$ по правилу $\forall$в. Свойство (1) гарантирует, что в доказательстве формулы $\forall\alpha A$ ни одна переменная не будет абсолютно ограничена дважды. Свойство (2) гарантирует, что в доказательстве формулы $\forall\alpha A$ ни одна переменная не будет ограничивать сама себя. Таким образом, доказательство формулы $\forall\alpha A$ будет завершенным. □

Показав, что все дедуктивные принципы K имеют место в BMV, получаем

Теорему 1.2.2. Система BMV *семантически полна, т.е.* $\Gamma \models B \Rightarrow \Gamma \vdash_{BMV} B$, где знак "$\Rightarrow$" – метаязыковая импликация.

Доказательство: из леммы 1.2.1 и теоремы о семантической полноте К (см. Мендельсон [1976, с. 78]). □

1.3. *Семантическая непротиворечивость системы* BMV

В. Куайн [1950] дал *синтаксическое* доказательство непротиворечивости своей системы натурального вывода, суть которого состоит в перестройке имеющегося вывода в вывод, непротиворечивость которого уже доказана. В данной главе мы предлагаем *семантическое* доказательство непротиворечивости. Поэтому сначала зададим стандартную семантику для первопорядковой логики, а потом уже непосредственно перейдем к доказательству непротиворечивости системы BMV (Шангин [2003]).

Зададим стандартную возможную реализацию для классической логики предикатов.

> Возможной реализацией называется любая пара <U, I>, такая что U – непустое множество, а I – функция, удовлетворяющая следующим условиям:
>
> (1) $I(k) \in U$,
> (2) $I(\Pi^n) \subseteq U^n$,
> (3) $I(\Phi^n)$ есть *n*-местная операция, заданная на U.

Значение произвольного терма t в некоторой модели <U, I> при некотором приписывании значений предметным переменным φ (обозначается записью "$|t|^M_\varphi$") определяется следующим образом:

(T1) Если терм t есть предметная константа k, то его значением в модели <U, I> при приписывании φ является тот индивид, который интерпретационная функция I сопоставляет константе k, т.е. $|t|^M_\varphi = I(k)$.

(T2) Если терм t есть предметная переменная α, то его значением в <U, I> при приписывании φ является тот индивид, который приписывается переменной α посредством φ, т.е. $|\alpha|^M_\varphi = \varphi(\alpha)$.

(T3) Пусть t есть сложный терм $\Phi^n(t_1, t_2,..., t_n)$. Для того чтобы установить его значение в возможной реализации <U, I> при приписывании φ, необходимо, во-первых, выделить операцию, которую функция I сопоставляет предметно-функциональной константе Φ^n, т.е. найти $I(\Phi^n)$; во-вторых,

установить значения термов $t_1, t_2,..., t_n$ в той же возможной реализации при том же приписывании, т.е. найти $|t_1|^M_\varphi$, $|t_2|^M_\varphi,\ldots$, $|t_n|^M_\varphi$; и, в-третьих, применить операцию $I(\Phi^n)$ к аргументам $|t_1|^M_\varphi$, $|t_2|^M_\varphi,\ldots$, $|t_n|^M_\varphi$. Результат применения данной операции к указанным объектам как раз и является значением терма $\Phi^n(t_1, t_2,..., t_n)$ в <U, I> при приписывании φ, т.е.

$$|\Phi^n(t_1, t_2,..., t_n)|^M_\varphi = [I(\Phi^n)](|t_1|^M_\varphi, |t_2|^M_\varphi,\ldots, |t_n|^M_\varphi).$$

Значение произвольной формулы A в <U, I> при некотором приписывании значений предметным переменным φ (обозначается записью "$|A|^M_\varphi$") определяется следующим образом (далее "***и***" обозначает "истина", а "***л***" обозначает "ложь"):

(Ф1) $|\Pi^n(t_1, t_2,..., t_n)|^M_\varphi = \boldsymbol{и} \Leftrightarrow <|t_1|^M_\varphi, |t_2|^M_\varphi,\ldots, |t_n|^M_\varphi> \in I(\Pi^n)$.
$|\Pi^n(t_1, t_2,..., t_n)|^M_\varphi = \boldsymbol{л} \Leftrightarrow <|t_1|^M_\varphi, |t_2|^M_\varphi,\ldots, |t_n|^M_\varphi> \notin I(\Pi^n)$.

(Ф2) $|\neg A|^M_\varphi = \boldsymbol{и} \Leftrightarrow |A|^M_\varphi = \boldsymbol{л}$.
$|\neg A|^M_\varphi = \boldsymbol{л} \Leftrightarrow |A|^M_\varphi = \boldsymbol{и}$.

(Ф3) $|A \,\&\, B|^M_\varphi = \boldsymbol{и} \Leftrightarrow |A|^M_\varphi = \boldsymbol{и}$ и $|B|^M_\varphi = \boldsymbol{и}$.
$|A \,\&\, B|^M_\varphi = \boldsymbol{л} \Leftrightarrow |A|^M_\varphi = \boldsymbol{л}$ или $|B|^M_\varphi = \boldsymbol{л}$.

(Ф4) $|A \vee B|^M_\varphi = \boldsymbol{и} \Leftrightarrow |A|^M_\varphi = \boldsymbol{и}$ или $|B|^M_\varphi = \boldsymbol{и}$.
$|A \vee B|^M_\varphi = \boldsymbol{л} \Leftrightarrow |A|^M_\varphi = \boldsymbol{л}$ и $|B|^M_\varphi = \boldsymbol{л}$.

(Ф5) $|A \supset B|^M_\varphi = \boldsymbol{и} \Leftrightarrow |A|^M_\varphi = \boldsymbol{л}$ или $|B|^M_\varphi = \boldsymbol{и}$.
$|A \supset B|^M_\varphi = \boldsymbol{л} \Leftrightarrow |A|^M_\varphi = \boldsymbol{и}$ и $|B|^M_\varphi = \boldsymbol{л}$.

Для формулировки условий истинности произвольных формул вида $\forall\alpha A$ и $\exists\alpha A$ договоримся, что α, α_1, $\alpha_2,\ldots$, α_n – список всех индивидных переменных, содержащихся в формуле $\forall\alpha A$ (или в $\exists\alpha A$), и что φ приписывает α индивид u из U, а переменным α_1, $\alpha_2,\ldots$, α_n – соответственно индивиды u_1, $u_2,\ldots$, u_n из U.

(Ф6) $|\forall\alpha A|^M_\varphi = \boldsymbol{и} \Leftrightarrow$ для любого $v \in U$: $\varphi'(\alpha) = v$, $\varphi'(\alpha_1) = u_1$, $\varphi'(\alpha_2) = u_2,\ldots$, $\varphi'(\alpha_n) = u_n$ и $|A|^M_{\varphi'} = \boldsymbol{и}$.
$|\forall\alpha A|^M_\varphi = \boldsymbol{л} \Leftrightarrow$ существует $v \in U$: $\varphi'(\alpha) = v$, $\varphi'(\alpha_1) = u_1$, $\varphi'(\alpha_2) = u_2,\ldots$, $\varphi'(\alpha_n) = u_n$ и $|A|^M_{\varphi'} = \boldsymbol{л}$.

(Ф7) $|\exists\alpha A|^M_\varphi = \boldsymbol{и} \Leftrightarrow$ существует $v \in U$: $\varphi'(\alpha) = v$, $\varphi'(\alpha_1) = u_1$, $\varphi'(\alpha_2) = u_2,\ldots$, $\varphi'(\alpha_n) = u_n$ и $|A|^M_{\varphi'} = \boldsymbol{и}$.
$|\exists\alpha A|^M_\varphi = \boldsymbol{л} \Leftrightarrow$ для любого $v \in U$: $\varphi'(\alpha) = v$, $\varphi'(\alpha_1) = u_1$, $\varphi'(\alpha_2) = u_2,\ldots$, $\varphi'(\alpha_n) = u_n$ и $|A|^M_{\varphi'} = \boldsymbol{л}$.

В формулировках последних условий приписывание значений предметным переменным φ' отличается от приписывания значений предметным переменным φ не более чем приписыванием значения

переменной α. С другой стороны, для любой переменной α_i, $1 \le i \le n$, из списка α_1, α_2,…, α_n верно, что $\varphi(\alpha_i) = \varphi'(\alpha_i)$. В таком случае мы будем говорить, что φ' – это α-альтернатива φ.

Теперь дадим определение общезначимой и выполнимой формулы классической логики предикатов, а также зададим отношение логического следования.

(31) Формула А является законом классической логики предикатов (общезначимой формулой), если и только если А принимает значения "истина" в каждой возможной реализации и при любом приписывании значений предметным переменным.
Формула А является необщезначимой, если и только если существует возможная реализация и существует приписывание значений предметным переменным, при котором А принимает значение "ложь".

(32) Формула А языка логики предикатов является выполнимой, если и только если существует возможная реализация и приписывание значений предметным переменным, при которых А принимает значение "истина".
Формула А является невыполнимой, если и только если А принимает значение "ложь" в каждой возможной реализации и при каждом приписывании значений предметным переменным.

(33) Пусть Г – непустое множество формул языка логики предикатов, тогда из Г логически следует формула В (обозначается "$\Gamma \models B$"), если и только если не существует возможной реализации и приписывания значений предметным переменным, при которых каждая формула из Г принимает значение "истина", а формула В – значение "ложь".

На этом построение семантики классической логики предикатов завершено.

Теперь введем следующие обозначения. Пусть $\Gamma \vdash B(\alpha_1,\ldots, \alpha_n)$ обозначает незавершенный вывод формулы В из Г, в котором *n* раз применялись правила $\forall$в или $\exists$и, приведшие к абсолютному ограничению переменных $\alpha_1,\ldots, \alpha_n$, где $\alpha_1,\ldots, \alpha_n$ – полный список таких переменных. Далее $\Gamma \vdash B[\alpha_1,\ldots, \alpha_n]$ обозначает вывод, в котором абсолютно ограниченные переменные $\alpha_1,\ldots, \alpha_n$ не входят свободно ни в Г, ни в В. Таким образом, $\Gamma \vdash B[\alpha_1,\ldots, \alpha_n]$ есть завершенный вывод. При этом запись $\Gamma \vdash B[]$, которая является частным случаем выраже-

ния $\Gamma \vdash B[\alpha_1,\ldots, \alpha_n]$, обозначает завершенный вывод, в котором правила $\forall$в и $\exists$и ни разу не применялись.

Теорема 1.3.1. Система BMV *семантически непротиворечива, т.е.* $\Gamma \vdash_{BMV} B[\alpha_1,\ldots, \alpha_n] \Rightarrow \Gamma \models B$.

Докажем эту теорему в два этапа.

1-й этап. Рассмотрим завершенный вывод вида $\Gamma \vdash_{BMV} B[]$ и покажем, что в этом случае будет справедливо метаутверждение $\Gamma \models B$. Доказательство осуществляется полной индукцией по k – количеству применений правил вывода при построении завершенного вывода $\Gamma \vdash_{BMV} B[]$.

Пусть $k = 0$.

Тогда $B \eqcirc A_i$, где A_i есть посылка из Г, и мы имеем завершенный вывод $A_i \vdash A_i$. Тривиально, так как $A_i \models A_i$, а следовательно имеет место $\Gamma \models B$.

Индуктивное предположение: теорема верна для $j < k$.

Индуктивный шаг: теорема верна для $j = k$.

Действительно, в этом случае в завершенном выводе $\Gamma \vdash_{BMV} B[]$ формула B получена k-ым применением только одно из правил – &в, &и, $\vee$в, $\vee$и, $\supset$в, $\supset$и, $\neg$в, $\neg$и, $\forall$и или $\exists$в. Но каждое из этих правил инвариантно относительно свойства "логического следования", т.е. если посылки этих правил логически следуют из Г, то и заключение тоже следует из Г. Таким образом, и в этом случае получаем, что $\Gamma \models B$.

2-й этап. Теперь докажем теорему для случая, когда в выводе применялись правила $\forall$в и $\exists$и. Собственно говоря, этот пункт является центральным и единственным, который надо доказать, так как именно при применении этих правил не сохраняется логическое следование. Доказательство будет вестись методом возвратной индукции по n применений правил $\forall$в и $\exists$и в этом выводе.

Индуктивное предположение: допустим, что теорема верна для $i = n$-1. Значит, в завершенном выводе с n-1 количеством применений $\forall$в и $\exists$и заключение следует из своих посылок, т.е. $\Gamma \vdash_{BMV} B[\alpha_1,\ldots, \alpha_{n-1}] \Rightarrow \Gamma \models B$.

Покажем, что теорема верна для $i = n$, т.е. $\Gamma \vdash_{BMV} B[\alpha_1,\ldots, \alpha_n] \Rightarrow \Gamma \models B$.

(а) Пусть в выводе $\Gamma \vdash_{BMV} B[\alpha_1,\ldots, \alpha_n]$ формула B получена применением правила $\forall$в. Тогда вывод имеет следующий вид:

$\Gamma,\ldots, C(\alpha_n),\ldots, \forall\beta C(\beta)$, где Γ – неисключаемые посылки, α_n абсолютно ограничена и $\forall\beta C(\beta)$ есть B. Пусть далее, для определенности, Γ есть $\{A_1,\ldots, A_h\}$. Покажем, что $A_1,\ldots, A_h \vDash \forall\beta C(\beta)$. Рассматриваемый вывод, содержит подвывод $\Gamma \vdash_{BMV} C(\alpha_n)[\alpha_1,\ldots, \alpha_{n-1}]$. По индуктивному предположению, $\Gamma \vDash C(\alpha_n)[\alpha_1,\ldots, \alpha_{n-1}]$. Тогда:

1.	$\Gamma \vdash C(\alpha_n)[\alpha_1,\ldots, \alpha_{n-1}]$	индуктивное предположение
2.	$\Gamma \nvDash \forall\beta C(\beta)$	пос.
3.	$\dot\forall$ M $\dot\forall$ φ($\lvert A_1,\ldots, A_h\rvert^M_\varphi = \boldsymbol{и} \Rightarrow \lvert C(\alpha_n)\rvert^M_\varphi = \boldsymbol{и}$)	∀ к 1
4.	$\dot\exists$ M $\dot\exists$ φ($\lvert A_1,\ldots, A_h\rvert^M_\varphi = \boldsymbol{и}$ и $\lvert\forall\beta C(\beta)\rvert^M_\varphi = \boldsymbol{л}$)	∀ к 2
5.	$\lvert A_1,\ldots, A_h\rvert^M_\varphi = \boldsymbol{и}$ и $\lvert\forall\beta C(\beta)\rvert^M_\varphi = \boldsymbol{л}$	$\dot\exists$ и к 4; M, φ – абс. огр.
6.	$\lvert A_1,\ldots, A_h\rvert^M_\varphi = \boldsymbol{и}$	&и к 5
7.	$\lvert\forall\beta C(\beta)\rvert^M_\varphi = \boldsymbol{л}$	&и к 5
8.	$\dot\exists$ φ'(φ'(β) = v и $\lvert C(\beta)\rvert^M_{\varphi'} = \boldsymbol{л}$)	Ф6 к 7; φ' β–альтернативно φ
9.	φ'(β) = v и $\lvert C(\beta)\rvert^M_{\varphi'} = \boldsymbol{л}$	$\dot\exists$ и к 8; φ' – абс. огр.
10.	φ'(β) = v	&и к 9
11.	$\lvert C(\beta)\rvert^M_{\varphi'} = \boldsymbol{л}$	&и к 9
12.	$\lvert A_1,\ldots, A_h\rvert^M_{\varphi^*} = \boldsymbol{и} \Rightarrow \lvert C(\alpha_n)\rvert^M_{\varphi^*} = \boldsymbol{и}$	$\dot\forall$ и к 3 (2 раза), где φ*(α_n) = φ'(β) = v и φ* – α_n-альтернатива φ.

Так как α_n не свободна в Γ (по условию завершенного вывода) и φ* – это α_n-альтернатива φ, то получаем:

13.	$\lvert A_1,\ldots, A_h\rvert^M_\varphi = \boldsymbol{и} \Leftrightarrow \lvert A_1,\ldots, A_h\rvert^M_{\varphi^*} = \boldsymbol{и}$	
14.	$\lvert C(\alpha_n)\rvert^M_{\varphi^*} = \boldsymbol{и}$	⊃и к 6, 12, 13 (2 раза)

противоречие 11, 14, так как φ*(α_n) = φ'(β) = v, а потому:

15. посылка неверна и имеет место $\Gamma \vDash \forall\beta C(\beta)$.

(б) Несколько сложнее доказывается теорема для случая, когда абсолютно ограниченная переменная появляется за счет использования правила ∃и. Так как переход по этому правилу от формулы $\exists\beta C(\beta)$ к формуле $C(\alpha_n)$ не является завершенным выводом, то последний должен быть продолжен до получения некоторой формулы B, в которой переменная α_n уже не является свободной.

Рассмотрим предварительно случай, когда формула $\exists\beta C(\beta)$, к которой в самый последний раз было применено правило ∃и, входит в множество посылок Γ. Тогда завершенный вывод имеет следующий вид: $\Gamma', \exists\beta C(\beta),\ldots, C(\alpha_n),\ldots, B$, где формула B не получена по правилу ∀в, а переменная α_n не свободна в Γ и в B. $\Gamma = \{\Gamma' \cup \exists\beta C(\beta)\}$. Пусть, для определенности, $\Gamma' = \{A_1,\ldots, A_h\}$.

Итак, нам надо обосновать утверждение о том, что если $\Gamma', \exists\beta C(\beta) \vdash_{BMV} B[\alpha_1,\ldots, \alpha_n]$, то $\Gamma', \exists\beta C(\beta) \vDash B$. С этой целью рассмотрим но-

вый вывод Г', $C(\alpha_n) \vdash_{BMV} B[\alpha_1,\ldots, \alpha_{n-1}]$, в котором вместо посылки $\exists\beta C(\beta)$ взята другая посылка – $C(\alpha_n)$. Поскольку количество применений правил $\forall$в и $\exists$и в выводе Г', $C(\alpha_n) \vdash_{BMV} B[\alpha_1,\ldots, \alpha_{n-1}]$ меньше, чем в выводе Г', $\exists\beta C(\beta) \vdash_{BMV} B[\alpha_1,\ldots, \alpha_n]$, то, по индуктивному предположению, Г', $C(\alpha_n) \vDash B$. Покажем, что в этом случае Г', $\exists\beta C(\beta) \vDash B$.

1.	$\dot\forall M \dot\forall \varphi(\lvert A_1 \&\ldots\& A_h \& C(\alpha_n)\rvert^M_\varphi = \textbf{\textit{и}} \Rightarrow \lvert B\rvert^M_\varphi = \textbf{\textit{и}})$	индуктивное допущение
	Допустим в то же время, что Г', $\exists\beta C(\beta) \nvDash B$, т.е.	
2.	$\dot\exists M \dot\exists \varphi(\lvert A_1 \&\ldots\& A_h \& \exists\beta C(\beta)\rvert^M_\varphi = \textbf{\textit{и}}$ и $\lvert B\rvert^M_\varphi = \textbf{\textit{л}})$	пос.
	По переименованию связанной переменной, получаем	
3.	$\dot\exists M \dot\exists \varphi(\lvert A_1 \&\ldots\& A_h \& \exists\alpha_n C(\alpha_n)\rvert^M_\varphi = \textbf{\textit{и}}$ и $\lvert B\rvert^M_\varphi = \textbf{\textit{л}})$	
4.	$\lvert A_1 \&\ldots\& A_h \& \exists\alpha_n C(\alpha_n)\rvert^M_\varphi = \textbf{\textit{и}}$ и $\lvert B\rvert^M_\varphi = \textbf{\textit{л}}$	$\dot\exists$ и к 3; M, φ – абс.
5.	$\lvert A_1 \&\ldots\& A_h\rvert^M_\varphi = \textbf{\textit{и}}$	&и, Ф3 к 4
6.	$\lvert B\rvert^M_\varphi = \textbf{\textit{л}}$	&и к 4
7.	$\lvert\exists\alpha_n C(\alpha_n)\rvert^M_\varphi = \textbf{\textit{и}}$	&и, Ф3 к 4
8.	$\dot\exists \varphi' \lvert C(\alpha_n)\rvert^M_{\varphi'} = \textbf{\textit{и}}$	Ф7 к 7; φ' α_n-альтернативно φ
9.	$\lvert C(\alpha_n)\rvert^M_{\varphi'} = \textbf{\textit{и}}$	$\dot\exists$ и к 8; φ' – абс.
	Так как α_n не свободна в Г и φ' – это α_n-альтернатива φ, то получаем	
10.	$\lvert A_1 \&\ldots\& A_h\rvert^M_{\varphi'} = \textbf{\textit{и}} \Leftrightarrow \lvert A_1 \&\ldots\& A_h\rvert^M_\varphi = \textbf{\textit{и}}$	
11.	$\lvert A_1 \&\ldots\& A_h\rvert^M_{\varphi'} = \textbf{\textit{и}}$	$\supset$и к 5, 10
	Так как α_n не свободна в В и φ' – это α_n-альтернатива φ, то получаем	
12.	$\lvert B\rvert^M_{\varphi'} = \textbf{\textit{л}} \Leftrightarrow \lvert B\rvert^M_\varphi = \textbf{\textit{л}}$	
13.	$\lvert B\rvert^M_{\varphi'} = \textbf{\textit{л}}$	$\supset$и к 6, 12
14.	$\lvert A_1 \&\ldots\& A_h \& C(\alpha_n)\rvert^M_{\varphi'} = \textbf{\textit{и}}$	Ф3 к 9, 11
15.	$\lvert A_1 \&\ldots\& A_h \& C(\alpha_n)\rvert^M_{\varphi'} = \textbf{\textit{и}} \Rightarrow \lvert B\rvert^M_{\varphi'} = \textbf{\textit{и}}$	$\dot\forall$ и к 1 (2 раза)
16.	$\lvert B\rvert^M_{\varphi'} = \textbf{\textit{и}}$	$\supset$и к 14, 15
	противоречие 13, 16	

т.е. посылка (выражение 2) неверна и имеет место Г', $\exists\beta C(\beta) \vDash B$.

При доказательстве этого утверждения использовалось переименование связанных переменных, т.е. использовалась эквивалентность вида $\exists\beta C(\beta) \equiv \exists\alpha_n C(\alpha_n)$. Общезначимость этой формулы легко семантически устанавливается.

(c) Рассмотрим теперь случай, когда формула $\exists\beta C(\beta)$ не входит в множество посылок Г. Тогда вывод имеет следующий вид: Г,…, $\exists\beta C(\beta)$,…, $C(\alpha_n)$,…, $B[\alpha_1,\ldots, \alpha_n]$, где α_n не свободна в Г и в В; $\Gamma = \{A_1,\ldots, A_h\}$.

Мы только что показали, что из посылок Г', которая совпадет с Г в рассматриваемом случае, и посылки $\exists\beta C(\beta)$ логически следует

формула B[α_1,…, α_n], т.е. Г, $\exists\beta C(\beta) \vDash B[\alpha_1,\ldots, \alpha_n]$. Допустим теперь, что вывод формулы $\exists\beta C(\beta)$ из посылок Г является завершенным. В силу того, что в этом выводе применяется меньшее, чем *n*, количество использования правил $\forall$в и $\exists$и, то, по индуктивному допущению, получаем, что имеет место: $\Gamma \vDash \exists\beta C(\beta)$. Применяя сечение, получаем, что $\Gamma \vDash B$.

Допустим теперь иное: что вывод формулы $\exists\beta C(\beta)$ из посылок Г не является завершенным. Это возможно только в единственном случае – когда формула $\exists\beta C(\beta)$ графически совпадает с формулой $\exists\beta C(\beta, \gamma)$, γ является абсолютно ограниченной переменной, а сама формула получена по правилу $\exists$и из формулы $\exists\gamma\exists\beta C(\beta, \gamma)$. Вывод теперь имеет следующий вид Г,…, $\exists\gamma\exists\beta C(\beta, \gamma)$,..., $\exists\beta C(\beta, \gamma)$,…, $C(\alpha_n)$,…, B[α_1,…, α_n], где α_n не свободна в Г и в B.

Мы только что показали (пункт (б)), что если верно Г, $C(\alpha_n) \vDash B$, то верно и Г, $\exists\beta C(\beta, \gamma) \vDash B$. Применяя вновь рассуждение типа (б) к последнему выражению, получаем, что тогда верно: Г, $\exists\gamma\exists\beta C(\beta, \gamma) \vDash B$. Допустим теперь, что вывод формулы $\exists\gamma\exists\beta C(\beta, \gamma)$ из посылок Г является завершенным. В силу того, что в этом выводе применяется меньшее, чем *n*, количество использования правил $\forall$в и $\exists$и, то, по индуктивному допущению, получаем, что имеет место: $\Gamma \vDash \exists\gamma\exists\beta C(\beta, \gamma)$. Применяя сечение, получаем, что $\Gamma \vDash B$.

Если же вывод $\Gamma \vdash \exists\gamma\exists\beta C(\beta, \gamma)$ не является завершенным, то, повторяя нужное количество раз выше описанное рассуждение, мы на каком-то шаге получим, что, с одной стороны, имеет место $\Gamma \vdash \exists\gamma_1\ldots\exists\gamma_{n-1}\exists\beta C(\beta, \gamma_1,\ldots,\gamma_{n-1})[]$, а потому (по первому этапу доказательства) будет верно $\Gamma \vDash \exists\gamma_1\ldots\exists\gamma_{n-1}\exists\beta C(\beta, \gamma_1,\ldots,\gamma_{n-1})$, а с другой – будет верно, что из посылок Г и посылки $\exists\gamma_1\ldots\exists\gamma_{n-1}\exists\beta C(\beta, \gamma_1,\ldots,\gamma_{n-1})$ логически следует формула B, т.е. Г, $\exists\gamma_1\ldots\exists\gamma_{n-1}\exists\beta C(\beta, \gamma_1,\ldots,\gamma_{n-1}) \vDash B$. Применяя теперь сечение, мы окончательно получаем: $\Gamma \vDash B$.

Итак, пункты (а), (б) (в) совместно со случаем, когда имеет место $\Gamma \vdash B[]$, доказывают теорему. □

2. Алгоритм автоматического поиска вывода в BMV

2.1. *Изменение формулировки системы* BMV

Доказав в предыдущем параграфе теоремы о непротиворечивости и полноте, мы показали надежность нашей системы. Теперь встает вопрос о поиске вывода в исчислении, а точнее, вопрос об автомати-

ческой процедуре поиска вывода. В целях эффективности автоматического поиска вывода добавим к множеству BMV-правил следующие правила исключения логических связок:

$$\neg\exists\text{и}\ \frac{\neg\exists\alpha A(\alpha)}{\forall\alpha\neg A(\alpha)} \qquad\qquad \neg\vee\text{и}\ \frac{\neg(A \vee B)}{\neg A, \neg B}$$

Выше было показано, что правила $\neg\vee$и, $\neg\exists$и выводимы в BMV, а потому их добавление не влияет на множество теорем исчисления. Значит, допущение этих правил не влияет на полноту и непротиворечивость системы.

Теперь подробно объясним необходимость введения именно таких правил. Отсутствие этих правил приводит к следующим затруднениям (Болотов и др. [1998]): (ситуация "а") пусть в алгоритме необходимо доказать формулу $A \vee B$, но в выводе нет ни A, ни B. Значит, (ситуация "б") в качестве посылки в множество вывода помещается $\neg(A \vee B)$ и теперь необходимо получить противоречие в выводе. Допустим, что противоречие не получено (и вывод, следовательно, не построен). Значит, (ситуация "в") алгоритм использует формулу $\neg(A \vee B)$ в качестве *источника для взятия целей* и теперь необходимо получить $A \vee B$ снова, т.е. алгоритм возвращается к ситуации "а". Значит, в работе алгоритма имеет место цикл.

Дабы избежать подобной неприятности авторы предлагают переходить от ситуации "б" к ситуации "в", только если этим двум ситуациям не предшествовала ситуация "а", т.е. формула $\neg(A \vee B)$ не была помещена в качестве посылки тогда, когда необходимо было получить $A \vee B$. В противном случае от ситуации "б" алгоритм переходит не к ситуации "в", а к ситуации "$в_1$": происходит "выход из алгоритма" и обосновывается метавыводимость формул $\neg A$, $\neg B$ из $\neg(A \vee B)$. Так как алгоритм всегда построит вывод формул $\neg A$, $\neg B$ из формулы $\neg(A \vee B)$, отмечается, что формула $\neg(A \vee B)$ не может быть *источником для взятия целей* и не возникнет необходимости получать $A \vee B$. Таким образом, от ситуации "$в_1$" алгоритм не перейдет к ситуации "а" и цикл в работе алгоритма ликвидирован. Нетрудно видеть, что метавыводимость формул $\neg A$, $\neg B$ из $\neg(A \vee B)$ в ситуации "$в_1$" есть нечто иное, как вводимое сейчас правило $\neg\vee$и. Что касается правила $\neg\exists$и, то объяснение необходимости данного правила сходно с объяснением необходимости правила $\neg\vee$и, учитывая аналогию между $\vee$ и $\exists$.

Еще раз обращаем внимание на то, что алгоритм поиска вывода в BMV может быть построен и без правил $\neg\vee$и, $\neg\exists$и (Болотов и др. [1998], [1996]). Однако введение этих правил позволяет облегчить

поиск вывода. Другими словами, вместо того чтобы переходить от ситуации "а" к ситуациям "в" или "$в_1$", если в выводе имеется формула $\neg(A \vee B)$ или $\neg\exists\alpha A(\alpha)$, мы предлагаем сразу применять к этим формулам, соответственно, правило $\neg\vee$и или $\neg\exists$и.

Теперь рассмотрим правила $\forall$в и $\exists$и, т.е. те правила системы, которые приводят к появлению в выводе абсолютных переменных.

$$\forall\text{в}\quad \frac{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)}{\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)}\quad \begin{array}{l}\beta \text{ – абс. огр.}\\ \gamma_1,\ldots, \gamma_n \text{ – огр.}\end{array}$$

$$\exists\text{и}\quad \frac{\exists\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)}{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)}\quad \begin{array}{l}\beta \text{ – абс. огр.}\\ \gamma_1,\ldots, \gamma_n \text{ – огр.}\end{array}$$

В формулировке $\forall$в и $\exists$и переменные $\gamma_1,\ldots, \gamma_n$ суть все свободные переменные, соответственно, из $\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)$ и $\exists\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)$. Отметим, что β не может принадлежать множеству $\{\gamma_1,\ldots, \gamma_n\}$, так как, согласно определению BMV-вывода, ни одна переменная не может ограничивать сама себя (*Определение 1.1.3*).

Покажем, что система BMV неявно использует сколемовские функции в правилах $\forall$в и $\exists$и. (Явно сколемовские термы используются при формулировке правила $\exists$и в предложенной Е.К. Войшвилло [1967] системе натурального вывода, т.е. в той системе, модификацией которой является BMV.) В качестве доказательства приведем вариант записи данных правил с использованием сколемовских функций σ, σ_1 и т.д. (Мендельсон [1976, с. 111]). Для $\forall$в: из $A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$ следует $\forall\alpha A$; для $\exists$и: из $\exists\alpha A$ следует $A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$, где σ – новая сколемовская функция, $\gamma_1,\ldots, \gamma_n$ – все свободные переменные в $\exists\alpha A$ и в $\forall\alpha A$. При этом нульместные сколемовские функции обозначаются константами a, b и т.д. Например, сколемовской формой формулы $\exists v\forall w\exists x\forall y\exists z(Q(v, w, y, z) \supset P(f(x, y), g(z)))$ будет выражение $Q(a, w, y, \sigma_1(w, y)) \supset P(f(\sigma(w), y), g(\sigma_1(w, y)))$. Теперь сравним запись в нашей системе с неявным использованием сколемовских термов и запись с использованием таких термов.

сколемовский вариант	вариант BMV
$A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$, где σ – новая функция, если $n > 0$; $A(\alpha/a)$, где a – новая константа, если $n = 0$	$A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$, где β – абс., $\gamma_1,\ldots, \gamma_n$ – отн., если $n > 0$; $A(\alpha/\beta)$, где β – абс., если $n = 0$

В обоих вариантах множества $\{\gamma_1,\ldots, \gamma_n\}$ совпадают. Подстановка везде правильна. Различия заключаются в следующем: (1) вариант BMV не использует функторы для образования сколемовского терма; (2) в случае $n = 0$ сколемовский вариант использует кон-

станты, а вариант BMV – абсолютные переменные. Более того, сколемовский вариант требует, чтобы константы и функторы были новыми, а вариант BMV не требует, чтобы абсолютные переменные были новыми.

Данные различия исчезают, если интерпретировать содержательную трактовку абсолютной переменной как константы, обозначающей некий объект, для которого выполняется условие A в записи $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ или $A(\alpha/\beta)$: "Переменная β в этом случае *абсолютно ограничена* в том смысле, что должна теперь рассматриваться как имя какого-то вполне определенного объекта, удовлетворяющего условию A" (см. Бочаров и Маркин [1994, с. 133]). Таким образом, везде, где в сколемовском варианте используются константы, вариант BMV использует *абсолютные переменные*, ведущие себя как константы. Значит, второе различие устраняется. Что же касается первого различия в записях $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ и $A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$, то содержательно они обозначают одно и то же: выбор значения для объекта, обозначенного в одном случае как "β – абс., $\gamma_1,\ldots, \gamma_n$ – отн.", а в другом – как "$\sigma(\gamma_1,\ldots, \gamma_n)$", ограничивает значения для остальных свободных переменных $\gamma_1,\ldots, \gamma_n$. Требованию в сколемовском варианте, чтобы сколемовская функция или константа была новой, соответствует в варианте BMV условие, что в BMV-выводе ни одна переменная не может быть абсолютно ограничена дважды (не более одного раза). Таким образом, на самом деле при каждом применении $\forall$в, $\exists$и используется новая переменная.

Итак, анализ показал, что в системе BMV неявным образом используется сколемизация. Значит, при задании процедуры поиска вывода в нашей системе можно обойтись без введения сколемовских термов. Специфика нашей системы – наличие абсолютных переменных, содержательно трактующихся как константы, а не как переменные, а также наличие условий корректного использования таких переменных. Применительно к унификации это означает, что абсолютные переменные наряду с константами не подлежат замене на другие термы, иными словами, подстановка термов происходит только вместо *неабсолютных* переменных.

Другой важной проблемой при поиске вывода является отслеживание переменных, которые ограничивают сами себя (непосредственно или по транзитивности). Проиллюстрируем необходимость запрета на самоограничение переменной примерами из (см. Бочаров и Маркин [1994, с. 135]): именно данный запрет, а не требование правильной подстановки не позволяет вывести в системе $A(y, y) \vdash \forall xA(x, y)$ и $\exists xB(x, y) \vdash B(y, y)$. Отмечается, что "при таком переходе переменная y ограничивает сама себя". При этом надо "учитывать,

что ситуация, когда переменная ограничивает сама себя, может возникнуть не только прямым образом,… но и косвенно. Здесь имеется в виду то обстоятельство, что отношение x ограничивает y является транзитивным, т.е. для него верно соотношение: если α ограничивает β, а β ограничивает γ, то α ограничивает γ". В силу этого в нашей системе не проходит доказательство формулы перестановочности кванторов: $\forall x \exists y P(x, y) \supset \exists y \forall x P(x, y)$.

Пусть далее запись $\beta < \gamma$ означает, что переменная β абсолютно ограничена и переменная γ ограничена относительно, т.е. при применении правила $\forall$в или $\exists$и по переменной β была относительно ограничена переменная γ, входящая свободно в формулу $\forall\alpha A(\alpha)$ или $\exists\alpha A(\alpha)$. Если при одном применении данных правил по переменной β были относительно ограничены несколько свободных переменных $\gamma_1,\ldots, \gamma_n$, то мы будем записывать это как $\beta < \gamma_1,\ldots, \beta < \gamma_n$.

Свойства отношения ограничения на множестве переменных задаются двумя аксиомами: $\forall\alpha\neg(\alpha < \alpha)$ – иррефлексивность (ни одна переменная не ограничивает сама себя); $\forall\alpha\forall\beta\forall\gamma(((\alpha < \beta)\ \&\ (\beta < \gamma)) \supset (\alpha < \gamma))$ – транзитивность (если α ограничивает β и β ограничивает γ, то α ограничивает γ). Таким образом, отношение ограничения есть отношение строгого порядка. Заметим, что теория строго порядка разрешима.

Совокупность (возможно, пустую) таких неравенств для некоторого BMV-вывода мы будем называть *системой неравенств* вывода и обозначать T_0, T_1, T_2,… Таким образом, каждому выводу соответствует некоторая (возможно, пустая) система неравенств.

> Система неравенств T_i (*не*)*противоречива*, если в ней (не) имеется или (не) может быть получено по транзитивности из неравенств, имеющихся в T_i, неравенство вида $\alpha < \alpha$.

Для разрешения вопроса о непротиворечивости произвольной системы неравенств T мы предлагаем метод таблиц. Сама таблица, как и аналитическая таблица, представляет рассуждение в виде некоторой последовательности шагов. Причем, каждому шагу рассуждения соответствует некоторая строка этой таблицы. Любая строка таблицы содержит одну систему неравенств. Первая строка таблицы содержит исходную систему неравенств T, противоречивость или непротиворечивость которой устанавливается. Переход от какой-либо строки таблицы (строки с номером n) к следующей строке (строке с номером $n + 1$) осуществляется с помощью правила *склейки* (С), представляющего собой формализацию свойства транзитивности отношения ограничения и перестановки (П). Данные правила позволяют заменять на $n + 1$-м шаге

какую-либо систему неравенств в строке с номером *n* на другую систему неравенств.

$$\text{С}\ \frac{\Gamma, \alpha < \beta, \beta < \gamma, \Delta}{\Gamma, \alpha < \beta, \beta < \gamma, \alpha < \gamma, \Delta} \qquad \text{П}\ \frac{\Gamma, \alpha < \beta, \omega < \gamma, \Delta}{\Gamma, \omega < \gamma, \alpha < \beta, \Delta}$$

Определение 2.1.1. Таблицей называется конечная последовательность строк $I_1, I_2, \ldots, I_k$, в которой каждая строка I_j содержит конечную систему неравенств. Каждая последующая строка I_{j+1} получается из предшествующей I_j заменой системы неравенств на одну новую систему неравенств на основании правила склейки или перестановки.

Таблица называется противоречивой, если система неравенств, находящаяся в последней строке таблицы, является противоречивой.

Система неравенств T противоречива, если и только если существует противоречивая таблица, первая строка которой содержит исходное множество неравенств T.

Покажем на примере, как работают таблицы.

Пусть необходимо выяснить, противоречива ли $T = \{x < y, z < x, y < x\}$.

1. $x < y, z < x, y < x$
2. $x < y, y < x, z < x$ — П к 1
3. $x < y, y < x, x < x, z < x$ — С к 2

Таблица противоречива (полученная на шаге 3 система неравенств содержит неравенство $x < x$). Значит, $T = \{x < y, z < x, y < x\}$ противоречива.

Теперь рассмотрим процедуру, которая позволяет за конечное число шагов определить, противоречива ли произвольная система неравенств T. Для этого введем несколько определений.

Неравенство $\alpha < \beta$ *пройдено* в T_i, если $\alpha < \beta \in T_i$.

Применение правила С в T_i *избыточно*, если результат этого применения правила С пройден в T_i.

Таблица *полна*, если таблица противоречива, или каждое применение правила С в системе неравенств T_i, которая находится на последней строке этой таблицы, избыточно, или правило С вообще не применимо.

Теорема 2.1.2. Любую таблицу можно достроить до полной за конечное число шагов.

Доказательство. Мы строим таблицу только для конечной системы неравенств T. Следовательно, количество переменных в T конечно. Правила C и П позволяют переходить от T к другим конечным системам неравенств T_1, T_2,... Значит, количество возможных применений правил C и П в системах T_1, T_2,... конечно. Так как множество переменных в T_1, T_2,… не отличается от количества переменных в T и избыточные (т.е. уже встречавшиеся выше) применения правила C блокируются, то мы, в конце концов, придем к полной таблице для T. □

Мы не приводим подробного алгоритма, который строит полную таблицу для T и устанавливает ее противоречивость или непротиворечивость, так как данный алгоритм тривиален.

2.2. *Унификация*

Введем основные определения, следуя Чень и Ли [1983]. Особенность нижеследующего алгоритма унификации – работа с абсолютными и неабсолютными переменными, а также работа с множествами ограниченных переменных. Поскольку семантически такие переменные суть константы, алгоритм запрещает подстановку одной абсолютной переменной вместо другой абсолютной переменной.

Определение 2.2.1. Подстановка – это конечное множество вида $\{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$, где

1. каждая α_i – неабсолютная переменная,
2. каждый t_i – терм, отличный от α_i,
3. все α_i различны.

Свойство 1 говорит, что подстановка заменяет только вхождения неабсолютных переменных и не заменяет вхождения абсолютных переменных, констант, предметных функторов, предикаторов, логических связок и кванторов. Свойство 2 исключает подстановку неабсолютной переменной вместо нее самой. Свойство 3 исключает различные подстановки для одной и той же неабсолютной переменной.

Пусть $\theta = \{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$ – подстановка и A – выражение. Тогда $A\theta$ – выражение, полученное из A заменой одновременно всех вхождений α_i $(1 \le i \le n)$ в A на t_i.

Пусть $\theta = \{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$ и $\lambda = \{\beta_1/t^*_1, \ldots, \beta_m/t^*_m\}$ – две подстановки. Тогда *композиция* θ и λ есть подстановка $(\theta \circ \lambda)$, которая получается из множества $\{\alpha_1/t_1\lambda, \ldots, \alpha_n/t_n\lambda, \beta_1/t^*_1, \ldots, \beta_m/t^*_m\}$ вычеркиванием (1) всех элементов $\alpha_j/t_j\lambda$, для которых $t_j\lambda = \alpha_j$, и (2) всех элементов β_i/t^*_i таких, что $\beta_i \in \{\alpha_1, \ldots, \alpha_n\}$.

Свойство (1) отбрасывает подстановки неабсолютной переменной вместо самой себя, а свойство (2) запрещает различные варианты подстановок вместо одной и той же неабсолютной переменной.

Композиция подстановок ассоциативна. Пустая подстановка ε есть одновременно и левое и правое тождество, т.е. $(\theta \circ \lambda) \circ \mu = \theta \circ (\lambda \circ \mu)$ и $\varepsilon \circ \theta = \theta \circ \varepsilon$ для всех θ, λ, μ.

Определение 2.2.2. Подстановка θ называется унификатором для множества $\{A_1\theta, \ldots, A_k\theta\}$ т.т.т., когда $A_1\theta = A_2\theta = \ldots = A_k\theta$. Говорят, что множество $\{A_1, \ldots, A_k\}$ унифицируемо, если для него существует унификатор.

Например, подстановка $\theta = \{x/y, z/v\}$, где x, z – неабсолютные переменные, y, v – абсолютные переменные, в формулы A(x, v) и A(y, z) приводит к тому, что $A(x, v)\theta = A(y, v) = A(y, z)\theta$.

Определение 2.2.3. Унификатор σ для множества выражений $\{A_1, \ldots, A_k\}$ будет наиболее общим унификатором т.т.т., когда для каждого унификатора θ для этого множества существует такая подстановка λ, что $\theta = \sigma \circ \lambda$.

Понятие наиболее общего унификатора очень важно в процессе поиска необходимой подстановки. Содержательно нахождение наиболее общего унификатора означает, что каждый возможный унификатор для некоторого множества формул может быть получен с помощью наиболее общего унификатора для этого множества формул.

Теперь зададим *алгоритм унификации*. Отметим, что, согласно нижеследующему описанию алгоритма поиска вывода, унификации подлежат формулы, обладающие одинаковой (*подобной*) структурой и различающиеся между собою только входящими термами.

Например, алгоритм не будет стараться унифицировать формулы P(x, y) и Q(x, y), поскольку данные формулы различаются между собою входящими предикаторами. Он не будет также унифицировать формулы P(x, y) и Q(x, y, f(z)), так как эти формулы отличаются друг от друга различной местностью.

Для целей нашего исследования можно ограничиться унификацией только *пар* формул.

Множество рассогласований D для непустого множества подобных формул $\{A_1, A_2\}$ получается выявлением первой (слева) позиции, на которой символы, входящие в формулы A_1 и A_2, не совпадают, и затем выписыванием из формул A_1 и A_2 каждого терма, занимающего эту позицию. Множество этих термов есть множество рассогласований в $\{A_1, A_2\}$.

Например, для формул $A_1 = P(x, v)$ и $A_2 = P(y, z)$, где x, z – неабсолютные переменные, y, v – абсолютные переменные, множество рассогласований D есть {x, y}.

Поскольку x – неабсолютная переменная и y – абсолютная переменная, то возможна подстановка $\theta_1 = \{x/y\}$, которая устраняет множество рассогласований D_1 (индекс 1 обозначает по порядку номер позиции, на которой возникло данное множество рассогласований).

Однако θ_1 не является унификатором для {P(x, v), P(y, z)}, поскольку $P(x, v)\theta = P(y, v) \neq P(y, z)\theta$.

Формулы P(y, v) и P(y, z) не совпадают на второй позиции, образуя множество рассогласований $D_2 = \{v, z\}$. Поскольку z – неабсолютная переменная и v – абсолютная переменная, то возможна подстановка $\theta_2 = \{z/v\}$, которая устраняет множество рассогласований D_2.

Таким образом, унификатором для формул P(x, v) и P(y, z) является композиция подстановок $\theta_1 \circ \theta_2 = \{x/y, z/v\}$.

В процессе унификации возможно появление унификатора (подстановки, унифицирующей две формулы), который является композицией подстановок. Значит, первоначальные формулы, унификация которых является исходной задачей, в процессе поиска унификатора изменяются с помощью новых подстановок. В предыдущем примере из формулы P(x, v) при помощи подстановки {x/y} получили формулу P(y, v), а из формулы P(y, z) при помощи подстановки {z/v} получили формулу P(y, v).

Другими словами, если мы обозначим исходную формулу P(x, v) как A_0 и исходную формулу P(y, z) как B_0, то A_1 есть P(y, v) и $B_1 = B_0 = P(y, z)$. Далее $A_2 = A_1 = P(y, v)$ и B_2 есть P(y, v). Таким образом, $A_2 = B_2$, т.е. P(x, v) и P(y, z) унифицируемы. При этом подстановка {x/y, z/v} – унификатор для формул P(x, v) и P(y, z) – является композицией подстановок {x/y} и {z/v}.

Поскольку {A, A} = {A}, мы будем называть множество вида {A, A} *одноэлементным.*

Алгоритм унификации двух подобных формул A *и* B

Символами { } обозначены комментарии.

Шаг 1. Множество k = 0, $A_k = A$, $B_k = B$, $\sigma_k = \varepsilon$ (пустая подстановка). {Алгоритм берет исходные две формулы A и B, для которых необходимо найти унификатор. Исходным унификатором при этом является пустая подстановка ε.}

Шаг 2. Если ни одна переменная в выводе не ограничивает сама себя, то перейти к шагу 3.

Иначе стоп: A_k и B_k не унифицируемы. Выход из алгоритма. {Попытка унифицировать формулы A_k и B_k приводит к появлению в выводе переменной, которая ограничивает сама себя.}

Шаг 3. Если $\{A_k, B_k\}$ – *одноэлементное* множество, то стоп: σ_k – наиболее общий унификатор для A_k и B_k. {Устранив k множеств рассогласований между формулами A_0 и B_0 и получив из формулы $A_0(B_0)$ формулу $A_k(B_k)$, алгоритм добился, что A_k графически совпадает с B_k. Значит, последняя подстановка является наиболее общим унификатором для формул A и B.}

Иначе найдем множество рассогласований D_k для A_k и B_k.

Шаг 4. Если существуют такие элементы α_k и t_k в D_k, что α_k – неабсолютная переменная, которая не входит в t_k, то перейти к шагу 5. {Подстановка, во-первых, осуществляется только вместо неабсолютной переменной, и другие термы, а также предикаты и логические связки не подлежат замене. Во-вторых, осуществляя подстановку терма вместо неабсолютной переменной, необходимо следить, чтобы заменяемая переменная не входила в этот терм.}

Иначе стоп: A_k и B_k не унифицируемы. Выход из алгоритма.

Шаг 5. Пусть $\sigma_{k+1} = \{\alpha_k/t_k\}$, $A_{k+1} = A_k\{\alpha_k/t_k\}$ и $B_{k+1} = B_k\{\alpha_k/t_k\}$. {Осуществляем подстановку σ_{k+1} в формулу $A_k(B_k)$ и получаем формулу $A_{k+1}(B_{k+1})$.}

Заметим, что $A_{k+1} = A\sigma_{k+1}$ и $B_{k+1} = B\sigma_{k+1}$. {Это означает, что формула $A_{k+1}(B_{k+1})$ может быть получена из множества формул $A_0(B_0)$ с помощью подстановки $\sigma_{k+1} = ((\sigma_1 \circ \sigma_2) \circ \sigma_3) \ldots \circ \sigma_k))$, т.е. с помощью композиции подстановок, осуществляемых на предыдущих k шагах.}

Шаг 6. Присвоить значение k+1 и перейти к шагу 2.

Теорема 2.2.4. Вышеприведенный алгоритм конечен и всегда находит наиболее общий унификатор для унифицируемого множества A.

Доказательство: наш алгоритм унификации отличается от оригинального алгоритма унификации, предложенного Чень и Ли в [1983], наличием шага 2, т.е. проверкой наличия переменной, которая ограничивает сама себя.

Данная проверка конечна в силу разрешимости теории строгого порядка, которая описывает абсолютно и относительно ограниченные переменные в выводе (теорема 2.1.2). С другой стороны, данная проверка сужает множество подстановок, допустимых согласно алгоритму унификации в Чень и Ли [1983]. Поскольку алгоритм унификации Чень и Ли конечен и всегда находит наиболее общий унификатор,

наш алгоритм унификации также конечен и всегда находит наиболее общий унификатор. □

Поясним данный алгоритм примерами.

Рассмотрим несложный пример, когда необходимо унифицировать формулы P(x) и P(y), где x – неабсолютная переменная и y – абсолютная переменная.

Пусть A есть P(x) и B есть P(y). Помещаем эти формулы в алгоритм. Тогда A_0 = P(x) и B_0 есть P(y), k = 0, $\sigma_k = \varepsilon$ (пустая подстановка) (шаг 1).

Допустим, что в вывод не входит переменная, которая ограничивает сама себя (шаг 2).

Тогда найдем область рассогласований $D_0 = \{x, y\}$ для формул A_0 и B_0 (шаг 3).

Поскольку неабсолютная переменная x не входит в y, переходим к осуществлению подстановки (шаг 4).

Осуществляем подстановку $\sigma_1 = \{x/y\}$ в формулы A_0, B_0 и получаем формулы A_1, B_1. Отметим, что $A_1 = A_0\sigma_1$ и $B_1 = B_0\sigma_1$ (шаг 5).

Присвоить параметру k значение 1 и вернуться на шаг 2.

Допустим, что в вывод не входит переменная, которая ограничивает сама себя (шаг 2).

A_1 = P(y) и B_1 = P(y). Значит, $\{A_1, B_1\}$ одноэлементное множество. Стоп: $\sigma_1 = \{x/y\}$ – наиболее общий унификатор для P(x) и P(y) (шаг 2).

Если в вышеприведенном примере переменная x, как и переменная y, является абсолютной, то формулы не унифицируемы и подстановка $\theta = \{x/y\}$ невозможна, так как в формулах, согласно определению подстановки, замене подлежат только неабсолютные переменные. В таком случае алгоритм унификации остановится и выдаст ответ, что данное множество формул не унифицируемо.

Следующий пример иллюстрирует необходимость проверки наличия переменной, которая ограничивает сама себя, для поиска унификатора.

Пусть требуется унифицировать формулы $P(x_1, y_1)$, $P(y_2, x_2)$, где $y_1 < x_1$ и $y_2 < x_2$ и x_1, x_2 суть неабсолютные переменные. Напомним, что $y_i < x_i$ означает, что переменная y_i абсолютно ограничена и x_i относительно ограничена.

Пусть A есть $P(x_1, y_1)$ и B есть $P(y_2, x_2)$. Помещаем эти формулы в алгоритм. Тогда $A_0 = P(x_1, y_1)$ и B_0 есть $P(y_2, x_2)$, k = 0, $\sigma_k = \varepsilon$ (пустая подстановка) (шаг 1).

Поскольку в $y_1 < x_1$ и $y_2 < x_2$ ни одна переменная не ограничивает сама себя (шаг 2), то найдем область рассогласований $D_0 = \{x_1, y_2\}$ для формул A_0 и B_0 (шаг 3).

Поскольку неабсолютная переменная x_1 не входит в y_2, переходим к осуществлению подстановки (шаг 4).

Осуществляем подстановку $\sigma_1 = \{x_1/y_2\}$ в формулы A_0, B_0 и получаем формулы $A_1 = P(y_2, y_1)$ и $B_1 = P(y_2, x_2)$. Отметим, что $A_1 = A_0\sigma_1$ и $B_1 = B_0\sigma_1$ (шаг 5).

Присвоить параметру k значение 1 и вернуться на шаг 2.

Поскольку в $y_1 < y_2$ и $y_2 < x_2$ ни одна переменная не ограничивает сама себя (шаг 2), то найдем область рассогласований $D_1 = \{y_1, x_2\}$ для формул A_1 и B_1 (шаг 3).

Поскольку неабсолютная переменная x_2 не входит в y_1, переходим к осуществлению подстановки (шаг 4).

Осуществляем подстановку $\sigma_2 = \{x_2/y_1\}$ в формулы A_1, B_1 и получаем формулы $A_2 = P(y_2, y_1)$ и $B_2 = P(y_2, y_1)$. Отметим, что $A_2 = A_1\sigma_2$ и $B_2 = B_1\sigma_2$ (шаг 5).

Присвоить параметру k значение 2 и вернуться на шаг 2.

В $y_1 < y_2$ и $y_2 < y_1$ переменная $y_1(y_2)$ ограничивает сама себя по транзитивности (шаг 2). Значит, подстановку $\sigma_2 = \{x_2/y_1\}$ в формулы A_1, B_1 делать нельзя.

Из этого следует, что формулы $P(x_1, y_1)$, $P(y_2, x_2)$, где $y_1 < x_1$ и $y_2 < x_2$ и x_1, x_2 суть неабсолютные переменные, неунифицируемы.

2.3. *Правила поиска вывода в системе* BMV

Основная идея алгоритма поиска вывода в нашей системе – построение двух непустых последовательностей формул. Первая последовательность формул – это последовательность формул вывода (ПВ), т.е. последовательность уже имеющихся в выводе формул. ПВ состоит из посылок и формул, полученных из предыдущих по одному из BMV-правил, т.е. ПВ представляет собою, собственно говоря, BMV-вывод. Вторая последовательность – это последовательность целей (ПЦ), которые алгоритм стремится достигнуть с помощью формул из ПВ. ПЦ не является BMV-выводом, так как, во-первых, целью может быть достижение противоречия в ПВ, т.е. целью необязательно является формула, и, во-вторых, ПЦ не является BMV-выводом, так как цели получаются из предыдущих целей не по правилам системы BMV.

Основными понятиями для ПЦ являются понятия "главная цель" и "текущая цель". Главная цель – это *первая* цель в ПЦ, т.е. формула. Текущая цель – это последняя цель в ПЦ, т.е. цель, вывод которой алгоритм строит для того, чтобы построить вывод главной цели. Отметим, что на каждом шаге работы алгоритма текущая цель всегда единственна. Таким образом, главная цель может быть текущей (когда в ПЦ нет других целей, кроме главной) и текущая цель необязательно является главной.

Другим важным понятием для ПЦ является понятие "достигнутой цели". Если текущей целью является формула, то текущая цель *достижима*, если она унифицируется с некоторой неисключенной формулой из ПВ. Если текущей целью является противоречие, то текущая цель *достижима*, если в выводе содержатся две неисключенные формулы вида A и ¬B, причем для A и B существует унификатор. В противном случае цель считается *недостижимой*. Понятие "недостижимая (недостигнутая) цель" следует понимать в том смысле, что данная цель недостижима в результате одного конкретного действия алгоритма унификации. Таким образом, цель может быть недостижимой в какой-то определенный момент построения вывода, но может стать достижимой в каком-то следующем моменте построения вывода.

Правила вывода системы BMV можно рассматривать и как правила поиска вывода в этой системе. Действительно, правила введения и правила исключения логических связок обладают общим свойством: с помощью каждого правила вывода в ПВ добавляются новые формулы. Таким образом, в процессе поиска вывода эти правила используются для выведения новых формул. Такие правила поиска вывода мы будем в данной главе называть *синтетическими*, потому что они "позволяют в ходе решения поисковой задачи добавлять к текущему выводу новые формулы" (см. Смирнов и др. [1996, с. 4]).

Запишем теперь BMV-правила в качестве правил поиска вывода, т.е. с использованием меток алгоритма M0, M1 и M2 (смысл меток приводится ниже, в описании алгоритма). В определенном смысле мы повторяем здесь то, что было уже сказано в предыдущей главе. Однако так как мы теперь имеем дело с логикой предикатов, причем ее формулировка дается без вре́менных переменных, здесь появляются новые моменты.

Синтетические правила исключения:

$$\&\text{и}\ \frac{A \,\&\, B^{M0}}{A^{M2},\ B^{M2}} \qquad \supset\text{и}\ \frac{A \supset B^{M0},\ A}{B^{M2}} \qquad \neg\vee\text{и}\ \frac{\neg(A \vee B)^{M0}}{\neg A^{M2},\ \neg B^{M2}}$$

$$\vee\text{и}\ \frac{A \vee B^{M0},\ \neg A}{B^{M2}} \qquad \neg\text{и}\ \frac{\neg\neg A^{M0}}{A^{M2}} \qquad \forall\text{и}\ \frac{\forall\alpha A(\alpha)^{M0,\,M1}}{A(\alpha/t)^{M2}}$$

$$\neg\exists\text{и}\ \frac{\neg\exists\alpha A(\alpha)^{M0}}{\forall\alpha\neg A(\alpha)^{M2}} \qquad \exists\text{и}\ \frac{\exists\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)^{M0}}{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)^{M2}}$$

β – абс.; $\gamma_1,\ldots, \gamma_n$ – огр

Синтетические правила введения:

$$\&\text{в}\ \frac{A,\ B}{A \,\&\, B} \qquad \supset\text{в}\ \frac{B}{C \supset B}$$

$$\vee\text{в}\ \frac{A\ (B)}{A \vee B} \qquad \neg\text{в}\ \frac{B,\ \neg B}{\neg C}$$

$$\exists\text{в}\ \frac{A(\alpha/t)}{\exists\alpha A(\alpha)} \qquad \forall\text{в}\ \frac{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)}{\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)} \qquad \beta \text{ – абс.; } \gamma_1,\ldots, \gamma_n \text{ – огр.}$$

В правилах $\supset$в и $\neg$в формула С является последней неисключенной посылкой. Отметим, что применение синтетических правил введения детерминируется текущими целями. Как это конкретно происходит будет сказано в описании алгоритма.

Для организации процедуры поиска вывода недостаточно только синтетических правил. Другая группа правил поиска вывода – *аналитические* правила. Их свойством является сведение "задачи поиска некоторого вывода к одной или нескольким подзадачам по поиску вспомогательных выводов" (Смирнов и др. [1996, с. 4]). Заметим, что нынешние понятия аналитичности и синтетичности расходятся с введенными в предыдущей главе понятиями. Это связано с тем, что там на аналитические и синтетические делились правила вывода системы BMV. Мы же делим на соответствующие классы не правила вывода, а правила поиска вывода. Можно сказать, что теперь аналитические правила суть переходы от текущей цели к *подцели*, которая, в свою очередь, сама становится текущей целью. Соответственно, цель, из которой получена подцель, называется *предыдущей* целью.

Например, необходимо доказать формулу A & B. Формула A & B становится текущей целью. Мы можем достигнуть A & B с помощью BMV-правил из имеющихся в выводе посылок. Однако поиск вывода существенно облегчается, если применить к текущей цели A & B некоторое аналитическое правило и получить подцели A и B. Если подцели A и B достижимы (а значит, A и B находятся в ПВ), то, применяя &в, можно получить цель A & B. Если необходимо достигнуть текущую цель A $\supset$ B, то лучше взять формулу A в качестве посылки и стараться достигнуть как подцель B. Если подцель B достижима (а значит, B находится в ПВ), то, применяя $\supset$в, можно получить цель A $\supset$ B. Если необходимо достигнуть текущую цель $\neg$A, то лучше взять в качестве посылки формулу A и стараться достигнуть как подцель противоречие. Если подцель противоречие достижима, то, применяя $\neg$в, можно получить цель $\neg$A. Такие советы по поиску вывода в системе BMV иногда называются *эвристиками*. Общее для всех этих эвристик состоит в том, что благодаря их использованию достижение цели сводится к достижению одной или нескольких подцелей. При этом степень подцелей меньше, чем степень цели. Степенью формулы A, g(A), является количество вхождений пропозициональных связок и кванторов. Например, g(p $\supset$ p) = 1, g($\forall$xP(x)) = 1, g(P(α)) = g($\perp$) = 0, где $\perp$ обозначает противоречие.

Как и ранее, мы называем такие аналитические правила поиска вывода для целей *правилами целей* (сокращенно – "*пц*").

В алгоритме задаются следующие правила целей с использованием меток алгоритма М3-М6 (смысл меток приводится ниже, в описании алгоритма).

$$пц\& \ \frac{A \,\&\, B \in ПЦ}{A^{M3} \in ПЦ,\ B \in ПЦ} \qquad пц\supset \ \frac{A \supset B \in ПЦ}{A \in ПВ,\ B \in ПЦ}$$

$$пц\neg \ \frac{A \in ПЦ}{\neg A \in ПВ,\ \bot \in ПЦ} \qquad пц\vee \ \frac{A \vee B \in ПЦ}{A^{M4} \in ПЦ \text{ или } B^{M5} \in ПЦ}$$

$$пц\forall \ \frac{\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n) \in ПЦ}{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n) \in ПЦ} \quad \beta \text{ – абс.; } \gamma_1,\ldots, \gamma_n \text{ – огр.}$$

$$пц\exists \ \frac{\exists\alpha A(\alpha) \in ПЦ}{A(\alpha/t)^{M6} \in ПЦ}$$

В правиле *пц*$\neg$ A либо элементарная формула, либо $\neg$C, либо C $\vee$ D, либо $\exists\alpha$C.

Проанализируем правила целей. Эти правила строятся таким образом, чтобы достижение подцели заключения влекло достижение предыдущей цели. Ибо если синтетические правила строят вывод, так сказать, сверху вниз (forward reasoning, от посылок к заключению), то правила целей строят вывод в обратном направлении, снизу вверх (backward reasoning, от заключения к посылкам). Например, *пц*$\forall$ позволяет от цели $\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)$ перейти к подцели $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$, где β – новая абсолютно ограниченная переменная и $\gamma_1,\ldots, \gamma_n$ относительно ограничены. Если подцель $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ достигнута (т.е. в ПВ имеется формула B: B и $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ унифицируемы с помощью подстановки Ω), то к BΩ можно применить правило $\forall$в по переменной β, вывести формулу $\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)\Omega$ и достигнуть цель $\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)\Omega$.

Правила целей представляют собой переход от формул большей степени к формулам меньшей степени из ПЦ. Поскольку алгоритм работает только с формулами конечной длины, количество применений правил целей к произвольной цели конечно. С другой стороны, правила целей не работают с целью противоречие $\bot$. Это позволяет сформулировать простое свойство правил целей:

Процесс сведения произвольной цели из множества целей с помощью правил целей к $\bot$ конечен.

Помимо правил целей существует еще группа аналитических правил поиска вывода, которые применяются не к целям из ПЦ, а к формулам из ПВ. Например, пусть текущей целью является проти-

воречие, она не достигнута и в ПВ имеется формула $A \supset B$. Тогда применяем к $A \supset B$ некоторое аналитическое правило поиска вывода и текущей целью становится формула A. На первый взгляд, необходимость данного правила поиска вывода неясна: каким образом достижение (причем, A не всегда достижима) цели A связано с достижением цели противоречия? Однако, если мы сможем достигнуть цель A, то, применив $\supset$и к $A \supset B$ и A, получим в ПВ формулу B. Наличие в ПВ формул A и B, наряду с формулой $A \supset B$, облегчает достижение цели противоречия. Поиск цели A играет важную роль и тогда, когда цель A недостижима: именно поиск A тогда, когда в ПВ имеется $A \supset B$, позволяет говорить, что ПВ содержит контрпример, опровергающий данную выводимость (этой темы мы подробно касаемся в следующем параграфе данной главы). Если необходимо получить цель противоречие, когда в ПВ имеется формула $A \vee B$, то применяем к $A \vee B$ некоторое аналитическое правило поиска вывода и текущей целью становится формула $\neg A$. Если $\neg A$ достижима, можно применить $\vee$и к формулам $A \vee B$, $\neg A$ и получить формулу B. Если не удается достигнуть цель $\neg A$, то ее поиск способствует построению контрпримера. Если необходимо получить цель противоречие, когда в ПВ имеется формула $\neg A$, то применяем к $\neg A$ некоторое аналитическое правило поиска вывода и текущей целью становится формула A. Если A достижима (т.е. формула A входит в ПВ), то в ПВ входят одновременно A и $\neg A$. Значит, цель противоречие достижима. Если A не удается получить, то ее поиск также способствует построению контрпримера.

Такие правила поиска вывода мы будем называть, как и ранее, правилами введения цели по формуле вывода или, кратко, *вц*. Отметим, что правила введения цели применяются только, если текущей целью является противоречие.

Задаются следующие правила порождения цели с использованием метки M7 (ее смысл приводится ниже, в описании алгоритма).

$$вц\vee\ \frac{A \vee B^{M7} \in \text{ПВ}}{\neg A^{M7} \in \text{ПЦ}} \qquad вц\supset\ \frac{A \supset B^{M7} \in \text{ПВ}}{A^{M7} \in \text{ПЦ}}$$

$$вц\forall\ \frac{\forall \alpha A^{M7} \in \text{ПВ}}{A(\alpha/t)^{M7} \in \text{ПВ}} \qquad вц\neg\ \frac{\neg A^{M7} \in \text{ПВ}}{A^{M7} \in \text{ПЦ}}$$

для *любого* t из ПВ

В правиле $вц\neg$ A не имеет вида элементарной формулы, или $C \vee D$, или $\exists \alpha C$. Отметим, что правило $вц\forall$ отлично от правила $\forall$и. Правило $вц\forall$ разрешает снимать квантор общности по *всем* термам из последовательности вывода, в то время как согласно правилу $\forall$и квантор общ-

ности снимается по первой новой неабсолютной переменной из последовательности вывода. Таким образом, применение правила *вц*$\forall$ представляет собой конечное число применений правила $\forall$и.

2.4. *Описание алгоритма поиска вывода в системе* BMV

Алгоритм описывается следующим образом. Сначала задаются основные процедуры алгоритма. Описываются их особенности и порядок применения. Далее описывается механизм меток: объясняется, когда и на какие формулы и цели ставятся (снимаются) метки. Затем следует пошаговое описание алгоритма поиска вывода в системе BMV.

К формулам из ПВ применяются следующие процедуры, которые мы делим на три группы: процедуры 1-4 применяются только к формулам из ПВ; процедура 5 применяется только к целям из ПЦ, и процедуры 6-7 применяются как к формулам из ПВ, так и к целям из ПЦ.

Процедура 1 определяет применение синтетических правил исключения. Процедура 1 состоит в нахождении формул, к которым применимо правило исключения. Если такие формулы обнаруживаются, то ПВ пополняется результатом применения данного правила.

Задается следующий порядок применения: пропозициональные правила (в произвольном порядке) применяются в первую очередь. Среди кванторных правил первым применяется $\neg\exists$и. Затем применяется $\exists$и. Правило $\forall$и применяется только, если остальные правила исключения не применимы.

Пропозициональные правила $\vee$и и $\supset$и применяются следующим образом:

$\supset$и. Пусть $A_2 \supset B \in$ ПВ и формула $A_2 \supset B$ не отмечена меткой М0 (смысл данной метки см. далее). Если $A_1 \in$ ПВ и найдется унификатор θ: $A_1\theta = A_2\theta$, то из $A_1\theta$ и $A_2 \supset B\theta$ по $\supset$и выводима $B\theta$.

$\vee$и. Аналогично $\supset$и.

Процедура 2 определяет применение синтетических правил введения. Процедура 2 состоит в нахождении формул, к которым применимо правило введения. Если такие формулы обнаруживаются, то ПВ пополняется результатом применения данного правила. Отметим, что применение правил введения к формулам из ПВ детерминировано целями из ПЦ.

Процедура 3 определяет применение *вц*-правил. Алгоритм ищет первую сверху вниз сложную формулу из ПВ, которая не отмечена метками М0 или М5 (смысл этих меток см. далее). Эта формула является *формулой, порождающей цели* (*фпц*). Вид *фпц* определяет, какое именно *вц*-правило применимо.

Процедура 4 применяется после всех остальных процедур, если в ПВ содержатся временно запрещенные для применения формулы $\forall\alpha A(\alpha)$. В таком случае метка М1 (см. далее) снимается со всех таких формул и квантор общности снимается по всем термам из ПВ.

К формулам из ПЦ применяется следующая процедура:

Процедура 5 определяет применение правил целей. Алгоритм анализирует текущую цель. Вид цели определяет, какое именно *пц*-правило применимо.

По всему дереву поиска вывода применяется следующая процедура:

Процедура 6 осуществляет с помощью алгоритма унификации глобальную подстановку Ω в ПВ и ПЦ, если Ω – наиболее общий унификатор, а также осуществляет проверку достижимости текущей цели. Если в процессе поиска вывода произошло достижение некоторой цели с помощью Ω, то алгоритм осуществляет данную подстановку Ω во всех (исключенных и неисключенных) формулах и целях из ПВ и ПЦ. Одновременно с осуществлением унификации происходит и проверка корректности работы с абсолютно ограниченными переменными.

Важную роль в работе алгоритма играет механизм меток. Следующие метки ставятся на формулы из ПВ:

М0 ставится на посылку (в однопосылочных) или на большую посылку (в двухпосылочных) правил исключения после применения данного правила. Цель этой метки состоит в запрете: 1) вторичного применения одного и того же правила к одним и тем же формулам, 2) их использования в качестве *фпц*.

М1 ставится на формулы вида $\forall\alpha A$, временно запрещая повторное применение к этим формулам правила исключения $\forall$и. Смысл этой метки состоит в отслеживании циклов, которые могут возникнуть в ходе поиска вывода.

М2 ставится на формулы, полученные в результате применения правил исключения. Эта метка позволяет отслеживать формулы, которые "исключаются" из вывода за счет применения правил $\supset$и и $\neg$и. В этом случае метка М0 снимается с соответствующей формулы.

Следующие метки ставятся на цели из ПЦ и формулы ПВ:

М3 ставится на левый конъюнкт A_i, полученный из цели $A_i \,\&\, A_j$. Метка сигнализирует, что идет поиск вывода формулы A_i.

М4 ставится на левый дизъюнкт A_i, полученный из цели $A_i \vee A_j$, а также на все формулы ПЦ и ПВ, которые после этого будут получены

в ходе поиска вывода формулы A_i. Данная метка позволяет в случае, если A_i не достигнута, убрать из ПЦ и ПВ все формулы, которые были получены в ходе попытки вывода формулы A_i.

М5 ставится на правый дизъюнкт A_j цели $A_i \vee A_j$, а также на все формулы ПЦ и ПВ, которые после этого будут получены в ходе поиска вывода формулы A_j. Смысл метки аналогичен смыслу метки М4.

М6 ставится на формулу $A(\alpha/v)$, полученную из цели $\exists\alpha A(\alpha)$,а также на все формулы ПЦ и ПВ, которые после этого будут получены в ходе поиска вывода формулы $A(\alpha/v)$. Смысл метки аналогичен смыслу метки М4 и М5.

М7 ставится на формулы из ПВ и ПЦ при применении правил, порождающих цели. Данная метка запрещает вторичное использование одних и тех же формул для порождения целей и, кроме того, если цель, отмеченная этой меткой, достигается, то позволяет снимать с соответствующих *фпц* данную запрещающую метку.

Описание алгоритма

Определяется *главная цель* вывода. Таковой является формула А, стоящая справа от знака выводимости в заданном метаутверждении о выводимости $\Gamma \vdash A$. Данная формула помещается в качестве текущей цели в ПЦ. Если слева от знака выводимости стоят формулы, то все они помещаются в качестве начальных посылок в ПВ. Все свободные переменные из Г, А помечаются как абсолютно ограниченные. Естественно, данная пометка должна быть убрана после построения вывода.

Если ПВ не пусто, то переход к *Процедуре 1*.

В противном случае переход к *Процедуре 5*.

Применяется *Процедура 1*. Применяются правила исключения. При этом на посылки правил вывода вида $A \,\&\, B$, $A \vee B$, $A \supset B$, $\neg\neg A$, $\neg(A \vee B)$, $\neg\exists\alpha A$, $\forall\alpha A$, $\exists\alpha A$, к которым были применены соответственно правила &и, $\vee$и, $\supset$и, $\neg$и, $\neg\vee$и, $\neg\exists$и, $\forall$и, $\exists$и, ставится метка М0. На заключения правил вывода ставится метка М2.

Правила $\exists$и, $\forall$и применяются следующим образом:

$\forall$и. Квантор общности снимается по правилу $\forall$и с формулы $\forall\alpha A$ по новой неабсолютной переменной. На формулу $\forall\alpha A$ дополнительно ставится М1, временно запрещающая повторное применение к этой формуле правила $\forall$и. Если вхождение $\forall$ по переменной α в формуле А вырожденно, то результатом снятия $\forall$ будет А.

$\exists$и. Квантор существования снимается по правилу $\exists$и с формулы $\exists\alpha A$ по новой абсолютной переменной. Рядом с полученной таким образом формулой $A(\alpha/\beta)$ пишется "β – абс., γ_1, γ_2,..., γ_n – огр.", где γ_1,

γ_2,..., γ_n – все свободные переменные, входящие в $\exists\alpha A$. Если вхождение $\exists$ по переменной α в формуле А вырожденно, то результатом снятия $\exists$ будет А.

Переход к *Процедуре 6*.

6.1. Достигнута текущая цель A_n.

1) Если A_n – *главная цель*, то A_n устраняется из ПЦ. Выход из алгоритма – вывод построен.

2) Если A_n не является *главной целью*, то:

Если A_n есть $\perp$, то, A_n устраняется из ПЦ. Текущей целью становится предыдущая цель и к формулам из ПВ применяется правило $\neg$в, результатом применения которого является формула $\neg C$, где С – последняя неисключенная посылка в ПВ. Делается отметка, что все формулы, начиная с формулы С и заканчивая формулой, предшествующей $\neg C$, исключаются из дальнейших шагов вывода. Если в число этих формул попадает формула с меткой М2, то с соответствующей формулы, породившей данную формулу, снимается метка М0. Переход к *Процедуре 1*.

Если A_n есть формула, то (переход к *Процедуре 2*) найдется цель A_{n-1}, которая является предыдущей к цели A_n в ПЦ. В зависимости от вида A_{n-1} рассматриваются следующие случаи:

а) $A_{n-1} = A_n \vee A_k$ или $A_k \vee A_n$. Тогда A_n и A_{n-1} устраняются из ПЦ. Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\vee$в к формуле A_n. Если на A_{n-1} стояла метка М7, то с соответствующей *фпц* снимается метка М7. Переход к *Процедуре 1*.

б) $A_{n-1} = A_k \supset A_n$. Тогда A_n и A_{n-1} устраняются из ПЦ. Текущей целью становится предыдущая цель и к формулам из ПВ применяется правило $\supset$в, результатом применения которого является формула $A_k \supset A_n$, где A_k – последняя неисключенная посылка в ПВ. Делается отметка, что все формулы, начиная с A_k и заканчивая формулой A_n, исключаются из дальнейших шагов вывода. Если в число этих формул попадает формула с меткой М2, то с соответствующей формулы, породившей данную формулу, снимается метка М0. Если на цели A_{n-1} стояла метка М7, то с соответствующей *фпц* снимается метка М7. Переход к *Процедуре 1*.

в) $A_{n-1} = A_k \,\&\, A_n$. Тогда:

Если A_n есть формула A_k из выражения $A_k \,\&\, A_n$, то со всех формул из ПЦ снимается метка М3, формула A_k устраняется из целей, последней целью становится формула A_n. Переход к *Процедуре 6*.

Если же достигнутая цель есть формула A_n из выражения $A_k \,\&\, A_n$, то A_n и A_{n-1} устраняются из ПЦ. Текущей целью становится

предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила &в к формулам A_n и A_k. На формулу A_k & A_n из ПВ ставится метка М0. Если на цели A_{n-1} стояла метка М7, то с соответствующей *фпц* снимается метка М7. Переход к *Процедуре 1*.

г) $A_{n-1} = \forall\alpha A$. Тогда A_n и A_{n-1} устраняются из ПЦ. Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\forall$в к формуле $A(\alpha/\beta, \gamma_1, \gamma_2,..., \gamma_n)$. Делается отметка, что β абсолютно ограничена и $\gamma_1, \gamma_2,..., \gamma_n$ относительно ограничены. На формулу $\forall\alpha A$ ставится метка М0. Если на цели A_{n-1} стояла метка М7, то с соответствующей *фпц* снимается метка М7. Переход к *Процедуре 1*.

д) $A_{n-1} = \exists\alpha A$. Тогда A_n и A_{n-1} устраняются из ПЦ. Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\exists$в к формуле $A(\alpha/\beta)$. На формулу $\exists\alpha A$ ставится метка М0. Если на цели A_{n-1} стояла метка М7, то с соответствующей *фпц* снимается метка М7. Переход к *Процедуре 1*.

е) $A_{n-1} = \perp$. Тогда A_n устраняется из ПЦ. Текущей целью становится предыдущая цель A_{n-1}. Переход к *Процедуре 6*.

6.2. Не достигнута текущая цель A_n.

Если A_n – формула, то переход к *Процедуре 5*.

Если $A_n = \perp$, то:

а) если цель $\perp$ отмечена меткой М3, то переход к *Процедуре 3*,

б) если цель $\perp$ отмечена меткой М3, но применить *Процедуру 3* невозможно, и в ПВ нет формул вида $\forall\alpha A$, то остановка алгоритма. Обосновать выводимость нельзя,

в) если цель $\perp$ отмечена меткой М3, применить *Процедуру 3* невозможно, и в ПВ имеются формулы вида $\forall\alpha A$, то переход к *Процедуре 4*, а после ее выполнения к *Процедуре 1*,

г) если цель $\perp$ отмечена меткой М4, то переход к *Процедуре 3*,

д) если цель $\perp$ отмечена меткой М4, но применить *Процедуру 3* невозможно, и в ПВ нет формул вида $\forall\alpha A$, то все цели и все формулы из ПВ, отмеченные меткой М4, удаляются. Новой текущей целью становится правый член дизъюнкции. На эту формулу, а также на все новые формулы из ПЦ и ПВ, полученные в ходе построения вывода, ставится метка М5. Переход к *Процедуре 5*,

е) если цель $\perp$ отмечена меткой М4, но применить Процедуру 3 невозможно, и в ПВ имеются формулы вида $\forall\alpha A$, то переход к *Процедуре 4*, а после ее выполнения к *Процедуре 1*,

ж) если цель $\perp$ отмечена меткой М5, то переход к *Процедуре 3*,

з) если цель $\perp$ отмечена меткой М5, но применить *Процедуру 3* невозможно и в ПВ нет формул вида $\forall\alpha A$, то все цели и все формулы из ПВ, отмеченные меткой М5, удаляются. Новой текущей целью становится противоречие $\perp$, которая не отмечается никакой меткой, а формула $\neg(A_i \vee A_j)$ помещается в ПВ в качестве посылки. Переход к *Процедуре 1*,

и) если цель $\perp$ отмечена меткой М5, применить *Процедуру 3* невозможно, и в ПВ имеются формулы вида $\forall\alpha A$, то переход к *Процедуре 4*, а после ее выполнения к *Процедуре 1*,

к) если цель $\perp$ отмечена меткой М6, то переход к *Процедуре 3*,

л) если цель $\perp$ отмечена меткой М6, но применить *Процедуру 3* невозможно, и в ПВ нет формул вида $\forall\alpha A$, то все цели и все формулы из ПВ, отмеченные меткой М6, удаляются. Новой текущей целью становится противоречие $\perp$, которая не отмечается никакой меткой, а формула $\neg\exists\alpha A(\alpha)$ помещается в ПВ в качестве посылки. Переход к *Процедуре 1*,

м) если цель $\perp$ отмечена меткой М6, применить *Процедуру 3* невозможно, и в ПВ имеются формулы вида $\forall\alpha A$, то переход к *Процедуре 4*, а после ее выполнения к *Процедуре 1*,

н) если цель $\perp$ не отмечена никакой меткой, то переход к *Процедуре 3*,

о) если цель $\perp$ не отмечена никакой меткой, применить *Процедуру 3* невозможно, и в ПВ нет формул вида $\forall\alpha A$, то остановка алгоритма. Обосновать выводимость нельзя,

п) если цель $\perp$ не отмечена никакой меткой, применить *Процедуру 3* невозможно, и в ПВ имеются формулы вида $\forall\alpha A$, то переход к *Процедуре 4*, а после ее выполнения к *Процедуре 1*.

Применяется *Процедура 5* к текущей цели А:

а) если А – элементарная формула, то формула $\neg A$ помещается в ПВ в качестве посылки. Новой текущей целью становится получение $\perp$. Переход к *Процедуре 1,*

б) если главным знаком формулы А является знак "$\neg$", то формула $\neg\neg A$ помещается в ПВ в качестве посылки. Новой текущей целью становится получение $\perp$. Переход к *Процедуре 1,*

в) если $A = A_i \supset A_j$, то формула A_i помещается в ПВ в качестве посылки. Новой текущей целью становится A_j. Переход к *Процедуре 1,*

г) если $A = A_i \,\&\, A_j$, то новой текущей целью становится формула A_i. На формулу A_i, а также на все новые формулы из ПЦ, ставится метка М3. Переход к *Процедуре 6,*

д) если $A \rightleftharpoons A_i \vee A_j$, то новой текущей целью становится формула A_i. На формулу A_i, а также на все новые формулы из ПЦ и ПВ, ставится метка М4. Переход к *Процедуре 6*,

е) если $A \rightleftharpoons \exists\alpha A$, то новой текущей целью становится формула $A(\alpha/v)$, где v – новая неабсолютная переменная. На формулу $A(\alpha/v)$, а также на все новые формулы из ПЦ и ПВ, ставится метка М6. Переход к *Процедуре 6*,

ж) если $A \rightleftharpoons \forall\alpha A$, то новой текущей целью становится формула $A(\alpha/\beta)$, где β – новая абсолютная переменная. Рядом с так полученной формулой $A(\alpha/\beta)$ пишется "β – абс., γ_1, γ_2,..., γ_n – огр.", где γ_1, γ_2,..., γ_n – все свободные переменные, входящие в $\forall\alpha A$. Если вхождение квантора $\forall$ по переменной α в формуле А вырожденно, то результатом снятия квантора $\forall$ будет сама формула А. Переход к *Процедуре 6*.

Применяем *Процедуру 3*.

Если в ПВ имеются сложные формулы, не отмеченные метками М0 или М7, то поступаем следующим образом:

а) если в выводе имеется формула вида $A_i \supset A_j$, то в качестве текущей цели берется A_i. Формула $A_i \supset A_j$ и текущая цель A_i помечаются меткой М7. Переход к *Процедуре 5,*

б) если в выводе имеется формула вида $A_i \vee A_j$, то в качестве текущей цели берется $\neg A_i$. Формула $A_i \vee A_j$ и цель $\neg A_i$ помечаются меткой М7. Переход к *Процедуре 5*,

в) если в выводе встречается формула $\neg A_i$, где A_i не имеет вида $A_n \vee A_k$, $\exists\alpha A$ или элементарной формулы, то в качестве текущей цели берется A_i. Формула $\neg A_i$ и цель A_i помечаются меткой М7 (к формулам же вида $\neg(A_n \vee A_k)$ и $\neg\exists\alpha A$ применяются, соответственно, правила исключения $\neg\vee$и, $\neg\exists$и). Переход к *Процедуре 5*.

Конец описания.

3. Метатеоретические свойства алгоритма

3.1. *Семантическая непротиворечивость алгоритма*

В предыдущем параграфе мы детально описали алгоритм поиска натурального вывода в системе BMV. Вследствие этого возникает вопрос о семантической непротиворечивости нашего алгоритма, т.е. не возникнет ли ситуация, что алгоритм докажет формулу, которая не является общезначимой согласно вышеприведенной семантике для классической логики предикатов?

Есть несколько различных способов доказательства теоремы о семантической непротиворечивости алгоритма. Например, Дж. Поллок

[Pollock] использует с этой целью второпорядковое исчисление предикатов.

Другой способ состоит в доказательстве, казалось бы, очевидного факта, что все выводы, которые строит алгоритм, суть выводы в системе BMV. Учитывая, что система BMV семантически непротиворечива, получаем по транзитивности, что алгоритм также семантически непротиворечив. Для доказательства такой теоремы необходимо ввести ряд новых понятий, центральным из которых является понятие алгоритмического BMV-вывода.

Алгоритмическим BMV-выводом (алго-выводом) называется непустая последовательность формул вывода (ПВ) совместно с непустой последовательностью целей (ПЦ), удовлетворяющих следующим условиям:

а) каждая формула в ПВ является или исходной посылкой, или посылкой, полученной по правилам целей, или формулой, полученной из предыдущих по одному из BMV-правил, причем, правила введения применяются таким образом:

– правило $\&$в применяется к формулам D и B, если D и B суть текущие достигнутые цели и предыдущей целью по отношению к D и B является формула D & B;

– правило $\vee$в применяется к формуле D или к формуле B, если D или B является текущей достигнутой целью и предыдущей целью по отношению к D или B является формула $D \vee B$;

– правило $\supset$в применяется к формуле B, если B является текущей достигнутой целью, формула C является последней неисключенной посылкой и предыдущей целью по отношению к B является формула $C \supset B$;

– правило $\neg$в применяется к формулам B, $\neg$B, если цель противоречие является текущей достигнутой целью, формула $\neg$C является последней неисключенной посылкой и предыдущей целью по отношению к цели противоречия является формула C;

– правило $\exists$в применяется к формуле D(t), если D(t) является текущей достигнутой целью и предыдущей целью по отношению к D(t) является формула $\exists\alpha D(\alpha)$;

– правило $\forall$в применяется к формуле $D(\beta)$, где β – абсолютно ограниченная переменная, если $D(\beta)$ является текущей достигнутой целью и предыдущей целью по отношению к $D(\beta)$ является формула $\forall\alpha D(\alpha)$.

– если в ПВ применялись правила $\supset$в и $\neg$в, то все формулы, начиная с последней посылки и вплоть до результата применения данного правила, исключаются из ПВ;

б) каждая цель является или исходной (главной) целью, или получена по одному из правил *пц* или *вц*.

Будем далее говорить, что некоторый алго-вывод является обосновывающим алго-выводом формулы А из посылок Г (и записывать это посредством $\Gamma \vdash_{ал} A$), если и только если множество неисключенных посылок в ПВ образует множество Г, формула А является *главной* целью и А достигнута.

Обосновывающий алго-вывод формулы А из посылок Г ($\Gamma \vdash_{ал} A$) всегда конечен, так как в нем последовательность формул вывода и последовательность формул целей конечны.

Пусть $\Gamma \vdash_{BMV} A$ означает *завершенный* BMV-вывод формулы А из множества (возможно, пустого) посылок Г. Тогда имеет место следующая

Лемма 3.1.1. Если $\Gamma \vdash_{ал} A$, *то последовательность формул вывода в* $\Gamma \vdash_{ал} A$ *образует завершенный* BMV*-вывод* $\Gamma \vdash_{BMV} A$, *т.е.*

$$\Gamma \vdash_{ал} A \Rightarrow \Gamma \vdash_{BMV} A.$$

Доказательство. Рассмотрим $\Gamma \vdash_{ал} A$.

В алго-выводе $\Gamma \vdash_{ал} A$ каждая формула в ПВ является или посылкой или получена из предыдущих по одному из BMV-правил (по определению алго-вывода). Таким образом, в обосновывающем алго-выводе $\Gamma \vdash_{ал} A$ выполняются все условия завершенного вывода, которые не касаются переменных.

Теперь рассмотрим ограничения на переменные, зафиксированные в определении завершенного BMV-вывода.

Появление абсолютно ограниченных переменных происходит по правилам $\forall$в, $\exists$и. В алго-выводе при применении $\exists$и каждый раз абсолютно ограничивается новая (ранее не встречающаяся ни в ПВ, ни в ПЦ) переменная. Применение $\forall$в в ПВ регулируется *пц*$\forall$ (определение алго-вывода), при применении которого каждый раз абсолютно ограничивается новая (ранее не встречающаяся ни в ПВ, ни в ПЦ) переменная. Таким образом, в $\Gamma \vdash_{ал} A$ ни одна переменная не ограничивается абсолютно более одного раза.

Согласно описанию алгоритма, все свободные переменные из Г, А помечаются как абсолютно ограниченные перед началом постро-

ения алго-вывода $\Gamma \vdash_{ал} A$. Выше показано, что в $\Gamma \vdash_{ал} A$ ни одна переменная не ограничивается абсолютно более одного раза. Значит, в $\Gamma \vdash_{ал} A$ все свободные переменные из Г, А не ограничиваются абсолютно.

По определению алго-вывода, в $\Gamma \vdash_{ал} A$ формула А является *главной* целью в ПЦ и формула А достигнута. Если А достигнута, то имела место процедура унификации, которая с помощью склейки неравенств (раздел 2.2) проверяет, не появилась ли в $\Gamma \vdash_{ал} A$ переменная, которая ограничивает сама себя. Значит, в $\Gamma \vdash_{ал} A$ ни одна переменная не ограничивает сама себя.

Таким образом, обосновывающий алго-вывод $\Gamma \vdash_{ал} A$ является завершенным BMV-выводом $\Gamma \vdash_{BMV} A$. □

Ранее была показана семантическая непротиворечивость исчисления BMV (теорема 1.3.1). Значит, семантическая непротиворечивость алго-вывода устанавливается следующей

Теоремой 3.1.2. В обосновывающем алго-выводе заключение следует из своих посылок, т.е. $A_1,\ldots, A_n \vdash_{ал} B \Rightarrow A_1,\ldots, A_n \models B$.

Доказательство: из леммы 3.1.1 и теоремы 1.3.1. □

В процессе построения любого алго-вывода зафиксируем каждое применение *вц*. Интуитивно представим процесс построения алго-вывода как последовательность переходов от одного применения некоторого *вц* к другому. Такие множества формул и целей в алго-выводе, полученные в результате применения одного *вц* вплоть до другого применения *вц*, будем называть *блоками*. В общем случае блок может состоять только из формул вывода (т.е. множество целей может быть пусто), если данный блок получен по *вц*$\forall$. Так как понятие блока фундаментально для нашего исследования, дадим строгое его определение.

Определение 3.1.3. Блоком α_i, $i \in N$, является непустая последовательность вывода $\{A_1, A_2,\ldots, A_n\}$ и (возможно, пустая) последовательность целей $\{B_1, B_2,\ldots, B_k\}$:

– B_1 есть *главная цель алго-вывода* (в этом случае $i = 1$), или B_1 получена по одному из *вц* (в этом случае $i > 1$);

– каждая цель, кроме B_1, получена по *пц*;

– последней целью блока является либо $\perp$, либо формула из последовательности целей, которая унифицируется с некоторой

формулой из последовательности вывода (тогда последняя цель считается *достигнутой*).

– каждая формула из последовательности вывода в α_i либо получена по *пц* из цели, содержащейся в α_i (в этом случае формула является посылкой), либо есть результат применения BMV-правила.

В качестве примера приведем алго-вывод формулы $((p \supset q) \supset p) \supset p$ с указанием на разделение последовательностей формул и целей на блоки.

	вывод	анализ	блоки	цели
1.	$(p \supset q) \supset p$	пос.		$((p \supset q) \supset p) \supset p$
2.	$\neg p$	пос.	блок 1	p
				$\perp$
3.	p	пос.		$p \supset q$
4.	$\neg q$	пос.		q
5.	$\neg\neg q$	$\neg$в к 2, 3		$\perp$
6.	q	$\neg$и к 5	блок 2	
7.	$p \supset q$	$\supset$в к 6		
8.	p	$\supset$и к 1, 7		
9.	$\neg\neg p$	$\neg$в к 2, 8		
10.	p	$\neg$и к 9		
11.	$((p \supset q) \supset p) \supset p$	$\supset$в к 10		

Здесь видно, что формулы $(p \supset q) \supset p$ и $\neg p$ из ПВ, и цели $((p \supset q) \supset p) \supset p$, p и $\perp$ из ПЦ образуют блок 1 (горизонтальная черта как раз и показывает границу между блоком 1 и блоком 2). Поскольку после получения посылки $\neg p$ (вторая строчка алго-вывода) ни одно правило исключения в ПВ неприменимо, а текущей целью является противоречие $\perp$, алгоритм осуществляет поиск формул из ПВ, к которым можно применить правило *вц*, т.е. алгоритм ищет сложные формулы из ПВ, к которым не применялись правила исключения. В данном случае имеется одна такая формула – $(p \supset q) \supset p$. Таким образом, мы применяем к ней *вц*$\supset$ и начинаем строить новый блок 2. Последний содержит формулы ПВ 3-11, а также цели $(p \supset q)$, q и $\perp$, находящиеся ниже горизонтальной черты.

Введенное понятие блока определяет понятие *дерева поиска вывода* в нашем исчислении. Процесс поиска вывода представляется в виде дерева, вершинами которого являются блоки, а ребрами правила порождения целей (*вц*), по которым из одного блока получается другой блок.

3.2. *Свойства алго-вывода*

Итак, в любом алго-выводе последовательность вывода и последовательность целей разбиты на не пересекающиеся между собою блоки. Конечность длины произвольного блока устанавливается

Леммой 3.2.1. В алго-выводе возможны блоки только конечной длины.

Доказательство: полная индукция по количеству *n* блоков в алго-выводе формулы A.

Базис: $n = 1$.

Первой целью блока α_1 является A – *главная* цель алго-вывода.

Последовательность целей в α_1 конечна (свойство *пц*). Конечная последовательность целей в α_1 порождает конечное множество посылок в последовательности вывода. Количество применений правила $\forall$и в одном блоке конечно. Значит, количество применений правил исключения среди конечного множества посылок вывода конечно (свойство BMV-правил). При этом, если хотя бы одна из текущих подцелей блока α_1 достигнута, то обязательно достигается и главная цель A. Тем самым данный блок будет конечен, завершаясь выводом A. Если же ни одна из текущих целей не достигнута, то, в силу конечности последовательности вывода и последовательности целей в α_1, конечен и сам α_1, завершаясь целью $\perp$.

Индуктивное предположение: допустим, лемма верна для $i < n$.

Индуктивный шаг: покажем, что лемма верна для $i = n$.

Пусть X_{n-1} обозначает множество неисключенных формул в алго-выводе для формулы A, где α_{n-1} – последний блок с последней целью – противоречием $\perp$. По индуктивному предположению, $\alpha_1,\ldots, \alpha_{n-1}$ суть конечные блоки, а значит, X_{n-1} конечно.

Рассмотрим 2 случая в зависимости от *вц*, по которому был получен блок α_n: 1) блок α_n получен по *вц* для $\vee$, $\supset$ или $\neg$; 2) блок α_n получен по *вц*$\forall$.

Случай 1. Пусть формула Y является первой целью в блоке α_n и формула Y получена по одному из *вц* для $\vee$, $\supset$ или $\neg$. Последовательность целей в α_n конечна (свойство *пц*). Конечная последовательность целей в α_n порождает конечное множество посылок вывода. Количество применений правила $\forall$и в одном блоке конечно. Значит, конечно количество применений правил исключения среди конечного множества посылок вывода (X_{n-1} конечно плюс свойство правил исключения). При этом, если хотя бы одна из подцелей блока α_n достигнута, то обязательно достигается первая цель Y. Тем самым α_n конечен, завершаясь выводом Y. Если же ни одна из целей не достигнута, то, в

силу конечности последовательности вывода и последовательности целей в α_n, конечен и сам α_n, завершаясь целью $\perp$.

Случай 2. Пусть формула Y является первой формулой вывода в блоке α_n, которая получена по *вц*$\forall$. Последовательность целей в α_n пуста, так как по *вц*$\forall$ в последовательность целей не добавляются новые цели. Значит, в последовательность вывода не вводятся новые посылки. Количество применений правил исключения в блоке α_n конечно (X_{n-1} конечно; свойство правил исключения; количество применений правила $\forall$и в одном блоке конечно).

Отметим, что последней целью в последовательности целей является последняя цель блока α_{n-1}, т.е. противоречие. Пусть формула Z является последней неисключенной посылкой в последовательности вывода при применении *вц*$\forall$, т.е. при порождении блока α_n. Если цель противоречие достижима, то в последовательности вывода происходит применение BMV-правила $\neg$в, появляется формула $\neg$Z и все формулы, начиная с формулы Z и вплоть до формулы $\neg$Z, исключаются из последовательности вывода. Значит, из последовательности вывода исключается также формула Y, которая в последовательности вывода находится ниже, чем формула Z. Таким образом, данный блок конечен. Если же ни одна из целей не достигнута, то, в силу конечности множества вывода и пустоты множества целей в α_n, конечен и сам α_n, завершаясь целью $\perp$ из блока α_{n-1}. □

Разбив алго-вывод на последовательность блоков, перейдем к анализу самих блоков.

Определение 3.2.2. Если блок α_i получен в результате применения *вц*$\neg$, *вц*$\supset$ или *вц*$\vee$, то α_i является *достигнутым* (д-блоком), при том, что α_i – последний блок алго-вывода и последняя текущая цель в α_i достигнута. Если блок α_i получен в результате применения *вц*$\forall$, то α_i является *достигнутым* (д-блоком), если α_i – последний блок алго-вывода и первая формула в α_i, т.е. формула, полученная по *вц*$\forall$, исключена. Напомним, что в таком блоке последовательность целей пуста. В противном случае блок является *недостигнутым*.

Определение 3.2.3. Блок α_n *непосредственно следует* за блоком α_k ($k < n$), если при порождении α_n последним недостигнутым блоком является α_k.

Если блок α_n получен по *вц*$\neg$, или *вц*$\forall$, то α_n *сильно* непосредственно следует за α_k. Если блок α_n получен по *вц*$\supset$ или *вц*$\vee$, то α_n *слабо* непосредственно следует за α_k.

Различение слабого и сильного непосредственного следования связано с тем, что, если α_n – д-блок и имеет место сильное непосредственное следование α_n за α_k, то α_k также является д-блоком. Дело в том, что в этом случае достижение первой цели α_n, полученной по *вц*¬, означает, что в множестве формул вывода имеются формулы A (первая цель α_n) и ¬A (посылка правила *вц*¬), а это автоматически ведет к достижению последней цели $\perp$ в блоке α_k. При этом достижение последней цели в α_k влечет достижение первой цели в α_k. Значит, α_k – д-блок. Что касается *вц*$\forall$, то, по определению д-блока, α_n достижим, если из последовательности вывода исключена первая формула A из α_n. Исключение формулы A возможно в данном случае только при применении ¬в, причем, формула C (последняя неисключенная посылка в последовательности вывода), исключаемая в результате применения правила ¬в, находится выше, чем формула A. Так как применение правил введения детерминируется целями, в выводе достигнута последняя цель $\perp$ блока α_k. Значит, α_k – д-блок.

Если имеет место слабое непосредственное следование α_n за α_k и α_n станет д-блоком, то α_k может и не стать д-блоком.

Теперь зададим понятие *поискового дерева блоков вывода*.

Определение 3.2.4. Поисковым деревом блоков вывода является частично упорядоченное множество $\{\alpha_1, \alpha_2, \ldots\}$ блоков, распределенных по непересекающимся "уровням" следующим образом:

– нулевой уровень состоит из единственного блока α_1, называемого *начальным* блоком;

– каждый блок i+1-го уровня соединен в точности с одним блоком i-го уровня;

– каждый блок α i-го уровня либо не соединен ни с одним блоком i+1-го уровня, либо соединен с некоторым множеством блоков i+1-го уровня (эти блоки i+1-го уровня называются непосредственно следующими за блоком α);

– блок i-го уровня, не соединенный ни с какими блоками i+1-го уровня, называется *заключительным* блоком.

В дальнейшем мы будем называть эту структуру "П-деревом". Вершинами его являются блоки, а ребрами – правила порождения цели по *фпц* (*вц*-правила), по которым один блок получается из другого. Между множеством алго-выводов и множеством П-деревьев имеет место следующее отношение: каждому алго-выводу соответствует

единственное П-дерево; каждому П-дереву – счетное множество алго-выводов.

Определение 3.2.5. Линейно упорядоченная последовательность блоков α_1, α_2, ... называется *нитью* данного П-дерева, если:
– α_1 – *начальный* блок,
– $\forall\, \alpha_n$ (α_n непосредственно следует за α_{n-1}).

Интуитивно П-дерево представляется как последовательность блоков, полученных по одному из *вц*-правил. В такой ситуации большую роль играют формулы вывода (*фпц*), к которым применимы *вц*-правила. Такими формулами являются формулы вида $\neg A$, $A \supset B$, $A \vee B$, не отмеченные метками М0 или М7, т.е. к ним не применялись правила исключения и правила *вц*, и формулы вида $\forall\alpha A(\alpha)$. К этим и только этим формулам применяются *вц*-правила.

Поиск вывода осуществляется путем выведения в последовательности вывода подформул имеющихся в ней формул. С одной стороны, выведение подформул осуществляется с помощью правил исключения, представляющих собой переход от формулы к ее подформуле(ам). Однако, естественно, не ко всем формулам вывода применимы правила исключения. В такой ситуации при поиске вывода необходимо применять *вц*-правила, которые также (это несложно установить) суть переходы от формулы к ее подформуле(ам). Поэтому нижеследующее определение выделяет роль таких формул в процессе поиска вывода.

Определение 3.2.6. Пусть при построении П-дерева α_n является последним недостигнутым блоком и j – количество нитей, построенных ранее, следующих из блока α_n и содержащих только д-блоки. Тогда $E^j_n = \{G_1, G_2,\ldots, G_s, H_1, H_2,\ldots, H_t\}$, $s \geq 0$, $t \geq 0$, назовем *порождающим множеством формул* блока α_n нити j+1, где $G_1, G_2,\ldots, G_s, H_1, H_2,\ldots, H_t$ – это формулы из ПВ, являющиеся *фпц*; причем, $G_1, G_2,\ldots, G_s$ – формулы вида $\neg A$, $A \supset B$, $A \vee B$, не отмеченные меткой М0 или М7, т.е. к ним не применялись правила исключения и они еще не служили в качестве *фпц*, и $H_1, H_2,\ldots, H_t$ – формулы вида $\forall\alpha A(\alpha)$.

Множество E^j_n конечно для любых конечных j, n, так как все блоки П-дерева конечны (*лемма 3.2.1*). В свою очередь, конечность множества E^j_n позволяет эффективно определять формулу (формулы, если их несколько) наибольшей степени из E^j_n.

Степень множества E^j_n (обозначается "$g(E^j_n)$") есть степень формулы наибольшей степени из E^j_n.

Теперь, после того как формально задано понятие алго-вывода, мы опишем процесс поиска вывода для произвольной формулы A:

1. Строим α_1.

Если α_1 не является д-блоком, то 2;

Если α_1 является д-блоком, то алго-вывод для формулы A построен;

2. Рассматриваем множество $E^0_1 = \{G_1, G_2,\ldots, G_s, H_1, H_2,\ldots, H_t\}$,

Если s = 0, то:

– если t = 0, то A недоказуема. Выход из алгоритма: последовательность формул вывода содержит *конечный контрпример*.

– если $t \neq 0$, то переход к 4.

Если $s \neq 0$, то переход к 3.

3. По формуле G_1 из множества E^0_1 строится новый блок α_2; переход к 5.

4. По формулам $H_1, H_2,\ldots, H_t$ из E^0_1 строится новый блок α_2; переход к 5.

5. Проверяется, является ли α_2 д-блоком.

Если да, то рассматриваем α_1.

Если нет, то рассматривается множество $E^0_2 = \{G'_1, G'_2,\ldots, G'_s, H'_1, H'_2,\ldots, H'_t\}$.

И т.д.

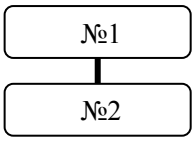
Грубо говоря, мы начинаем работать с самой левой нити в П-дереве. Если блок становится д-блоком, то нить, содержащая д-блок, обрывается, в нити мы переходим на один "уровень" выше (такая процедура называется бэктрэкингом от английского "back-tracking") и начинаем работать с блоком, из которого непосредственно следует д-блок. Если этот блок не становится д-блоком, мы рассматриваем порождающее множество этого блока. Если оно пусто, то построение нити (а вместе с ней и всего П-дерева) обрывается, так как, по определению П-дерева, мы не можем далее строить блоки, непосредственно следующие за этим блоком. Если порождающее множество этого блока непусто, мы строим новый блок, непосредственно следующий за данным. Допускается бесконечное порождение блоков (например, по правилу *вц*$\forall$), каждый из которых непосредственно следует за предыдущим, т.е. допускается наличие бесконечной нити в П-дереве, а значит, бесконечных П-деревьев. Все блоки в такой нити являются не достигнутыми (иначе с помощью бэктрэкинга нить обрывается). Значит, ветвление в этой нити невозможно, что, в свою очередь, влечет *единственность бесконечной нити в П-дереве*.

Ветвление в П-дереве образуется следующим образом. Пусть из блока 1 следует блок 2 и блок 2 является д-блоком, тогда, по определению П-дерева, мы возвращаемся (бэктрэкинг) к блоку 1. При

этом порождающее множество блока 1 непусто и сам блок 1 не является д-блоком. Значит, мы строим блок 3, который также непосредственно следует за блоком 1. Таким образом, в данном П-дереве из блока 1 непосредственно следуют два блока – блок 2 и блок 3, т.е. имеет место ветвление. Далее мы покажем, что в П-дереве имеет место только конечное ветвление, т.е. в П-дереве не существует блока, из которого непосредственно следует бесконечное количество блоков. Таким образом, алгоритм *по очереди* (слева направо) просматривает все нити П-дерева.

Приведем некоторые П-деревья и соответствующие им алго-выводы.

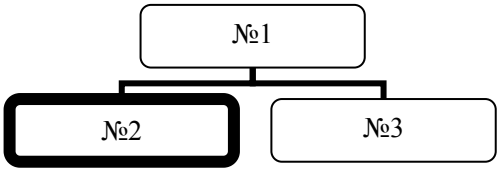
Пример 1.



В данном П-дереве блок 2 непосредственно следует из блока 1. Такому П-дереву соответствует следующий алго-вывод формулы $q \supset r$ из формулы $p \vee q$ (горизонтальная черта обозначает границу между блоком 1 и блоком 2):

вывод		анализ	цели	блоки
1.	$p \vee q^{M7}$	пос.	$q \supset r$	
2.	q	пос.	r	блок 1
3.	$\neg r$	пос.	$\perp$	
4.	p	пос.	$\neg p^{M7}$	
5.	$\neg\neg p$	пос.	$\perp$	блок 2
6.	p	$\neg$и к 5		
		стоп		

Пример 2.



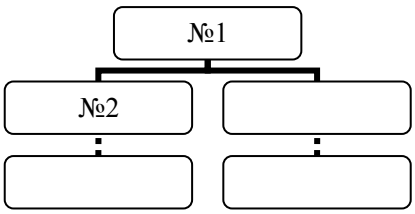
В данном П-дереве: (а) блок 2 непосредственно следует из блока 1; (б) блок 2 является д-блоком (обозначается жирным контуром) и, значит, (с) алгоритм посредством бэктрэкинга вернулся на блок 1, который не является д-блоком, после чего был порожден блок 3, который непосредственно следует из блока 1. Такому П-дереву соответствует

следующий алго-вывод формулы $q \supset r$ из формулы $(p \vee q) \supset q$ (горизонтальная черта обозначает границу между блоком 1 и блоком 2):

вывод		анализ	цели	блоки
1.	$(p \vee q) \supset q^{M7, M0}$	пос.	$q \supset r$	
2.	q	пос.	r	блок 1
3.	$\neg r$	пос.	$\perp$	
4.	$p \vee q^{M7}$	$\vee$в к 2.	$p \vee q^{M7}$	блок 2
5.	q	$\supset$и к 1, 4	p или q	
6.	$\neg\neg p$	пос.	$\neg p$	
7.	p	$\neg$и к 6	$\perp$	блок 3
		стоп		

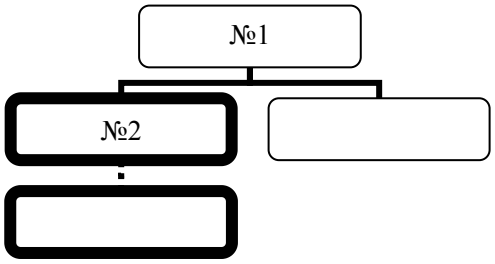
Приведем некоторые примеры деревьев, которые не могут быть П-деревьями.

Пример 3.



Данное дерево содержит блок, из которого непосредственно следуют две бесконечные (обозначается пунктирной линией) нити, состоящие из не достигнутых блоков.

Пример 4.



Данная схема означает дерево, содержащее нить, расположенную *правее* бесконечной нити.

В пропозициональных логиках (классической и интуиционистской) (Шангин [2000], [2002]), *по очереди* работая со всеми ветками П-дерева, мы либо посредством бэктрэкинга вернемся на *начальный* блок П-дерева (и формула А тогда является доказанной), либо остановимся (и тогда формула А недоказуема). При этом множе-

ство неисключенных литералов (пропозициональные переменные и их отрицания в формуле A) образует контрпример. Таким образом, данные процедуры являются разрешающими. Ситуация же в первопорядковой логике сложнее. В силу результата А. Черча [1936a], [1936b] о неразрешимости классической логики предикатов, в общем случае процесс поиска вывода не может быть остановлен: мы не можем за конечное число шагов определить, доказуема ли произвольная формула или нет.

С другой стороны, Ж. Эрбран показал, что любая общезначимая формула логики предикатов доказуема за конечное число шагов (Минц [1967], Чень и Ли [1983]). В такой ситуации встает проблема выбора стратегии, позволяющей за конечное число шагов вывести произвольную общезначимую формулу классической логики предикатов. Более того, процедура поиска вывода должна задаваться таким образом, чтобы гарантировать построение контрпримера (множества Хинтикки) даже в случае "ухода в бесконечность" (Хинтикка [1955], Сиг [1992]).

С помощью понятия *порождающего множества* мы исследуем взаимоотношения между формулами, которые входят в блоки, (непосредственно) следующие друг за другом в одной нити П-дерева. Данные взаимоотношения характеризуются

Леммой 3.2.7. Пусть α_n *непосредственно следует за* α_k *и* E_n, E_k *– соответственно, порождающие множества из* α_n *и* α_k, *тогда* $g(E_n) \leq g(E_k)$, *т.е. при увеличении числа блоков в нити степень новых порождающих множеств не увеличивается.*

Доказательство: индукция по количеству *n* блоков в нити.

Индуктивное предположение: допустим, лемма верна для $i < n$.

Индуктивный шаг: покажем, что лемма верна для $i = n$.

Рассмотрим E_k и E_n. E_n (возможно) отличается от E_k новыми формулами, которые добавляются в последовательность вывода либо 1) по некоторому правилу введения, либо 2) по некоторому правилу исключения.

Рассмотрим случай 1. Пусть $G \in E_k$ – формула, к которой применяется некоторое правило, порождающее цели, при порождении α_n, и M – первая цель (формула) в α_n. Значит, по свойству *вц*-правил, $g(M) < g(G)$. По свойству *пц*, $g(M') < g(M)$, где M' – произвольная (не обязательно первая) подцель из α_n. Значит, $g(M') < g(G)$. Поскольку применение правил введения в α_n детерминируется целями этого блока, $g(Y) \leq g(M)$, где Y – произвольная формула, полученная по правилам введения в α_n. Из $g(Y) \leq g(M)$ и $g(M) < g(G)$ следует $g(Y) < g(G)$.

Рассмотрим случай 2. Если формула Y получена по некоторому правилу исключения, то $g(Y) < g(X)$, где X – (большая) посылка этого правила и $X \in E_k$.

Итак, добавленные новые формулы (образующие E_n) имеют степень меньшую, чем степень формул из E_k. С другой стороны, из E_k в E_n перешли все формулы вида $\forall\alpha A(\alpha)$. Отсюда, $g(E_n) \leq g(E_k)$.

По индуктивному предположению, лемма верна для всех порождающих множеств из предыдущих блоков. Значит, $g(E_n) \leq g(E_k)$, для любых *n, k*. □

3.3. *Семантическая полнота алгоритма*

Доказательство теоремы о семантической полноте алгоритма стандартно. Сначала мы показываем конечность ветвления в произвольном П-дереве, т.е. из некоторого блока непосредственно следует конечное число блоков. Конечность ветвления, в свою очередь, позволяет применить к П-дереву обычную технику с использованием леммы Кенига (Сиг, Бернс [1998]) для определения бесконечной нити в П-дереве. Данная нить создается канонически: в этой нити возможно построение модельного множества (Хинтикка [1955]). Рассматривая П-дерево, отметим, прежде всего, что в общем случае для логики предикатов оно не будет конечным, как для логики высказываний.

Лемма 3.3.1. В произвольном П-*дереве за каждым блоком непосредственно следует только конечное число блоков*.

Доказательство: полная индукция по степени g порождающего множества E_k из произвольного блока α_k.

От противного: допустим в П-дереве существование блока α_k, из которого непосредственно следует бесконечное количество блоков.

Базис: $g = 0$.

Так как $E_k = \varnothing$, то за блоком α_k не следует ни одного блока. Базис доказан.

Индуктивное предположение: лемма верна для $g < n$.

Индуктивный шаг: $g = n$.

Итак, в П-дереве имеем блок α_k, из которого *непосредственно* следует бесконечное количество блоков. Согласно свойствам П-дерева, из этого факта вытекает следующее: (1) блок с такими свойствами единственен в данном П-дереве, так как наличие бесконечного ветвления в одном блоке П-дерева и механизм перехода от блока к блоку с помощью бэктрэкинга запрещает возникновение второго блока с бесконечным ветвлением, (2) все следующие за α_k блоки суть д-блоки, но сам α_k остается не достигнутым блоком (ав-

томат порождает блоки, следующие из данного блока, затем посредством бэктрэкинга опять возвращается к α_k и т.д., т.е. алгоритм "зацикливается" на блоке α_k.

Из (1) и индуктивного предположения следует, что все нити в таком дереве конечны. Так как бесконечность длины нити в П-дереве – это результат потенциально неограниченного количества применений *вц*$\forall$, то все непосредственно следующие за α_k блоки не были получены по правилу *вц*$\forall$. Более того, так как α_k остается не достигнутым блоком и все непосредственно следующие за α_k блоки суть д-блоки, то между этими блоками и α_k имеет место только *слабое* непосредственное следование, (в противном случае достижимость этих блоков привела бы к достижимости α_k), т.е. для любого j, $\forall\alpha A(\alpha) \notin E^j_k$, $\neg A \notin E^j_k$, $A \vee B \in E^j_k$, $A \supset B \in E^j_k$. Таким образом, для любого j, E^j_k состоит только из формул вида $A \vee B$, $A \supset B$.

Пусть в E^1_k, скажем, t формул вида $A \vee B$, $A \supset B$. Допустим, что ко всем этим формулам алгоритм применяет *вц*-правила, порождая, таким образом, в П-дереве t блоков, которые непосредственно следуют из блока α_k. Тогда в E^{t+1}_k не входят формулы из E^1_k. По лемме 2.3.7, $g(E_{k+1}) \leq g(E_k)$, если в E_k входят формулы вида $\forall\alpha A(\alpha)$ или некоторые формулы максимальной степени из E_k перешли в E_{k+1}. Так как для любого j, $\forall\alpha A(\alpha) \notin E^j_k$, то $\forall\alpha A(\alpha) \notin E^t_k$. С другой стороны, мы показали, что в E^{t+1}_k не входят формулы из E^1_k. Отсюда $g(E^{t+1}_k) < g(E^1_k)$, т.е. добавленные в порожденное множество блока α_k формулы из блоков, следующих за α_k, имеет степень меньшую, чем формулы, входящие в E^1_k.

Поскольку для любого j, E^j_k конечно, алгоритм в конце концов исчерпает E^j_k, скажем, на шаге h, t < h. Тогда $g(E^h_k) = \varnothing$, что противоречит условию о бесконечном ветвлении блока α_k. □

Прежде чем перейти к доказательству теоремы полноты для нашего алгоритма, зададим несколько стандартных определений, следуя работе (Непейвода [2000]).

Определение 3.3.2. П-дерево является *достигнутым*, если оно содержит только д-блоки.

Если П-дерево содержит только д-блоки, то П-дерево конечно. Таким образом, некоторому достигнутому П-дереву соответствует множество обосновывающих конечных алго-выводов. Согласно теореме о непротиворечивости алго-вывода, все эти алго-выводы суть *завершенные* BMV-выводы. Теперь нужно ответить на вопрос, множество каких алго-выводов соответствует недостигнутому П-дереву.

Определение 3.3.3. Назовем нить φ *незапертой*, если или φ бесконечна или заключительный блок α_k из φ не является д-блоком и $E_k = \varnothing$.

Незапертая нить, таким образом, бывает либо бесконечной (в этом случае она состоит только из недостигнутых блоков), либо конечной (в таком случае она оканчивается недостигнутым блоком и дальнейшее построение блоков в данной нити невозможно).

Определение 3.3.4. П-дерево является *недостигнутым*, если оно содержит незапертую нить.

Таким образом, недостигнутому П-дереву соответствует либо множество конечных алго-выводов, либо множество бесконечных алго-выводов.

После того, как мы установили конечность ветвления в произвольном П-дереве, можно применить *лемму Кенига*, согласно которой бесконечное П-дерево с конечным ветвлением содержит бесконечную нить (Непейвода [2000]). Напомним, что в П-дереве бесконечная нить, если она существует, единственна. Поэтому зададим построение этой нити так, чтобы она всегда содержала опровергающую модель. То есть множество формул, принадлежащих этой нити, должно образовывать множество Хинтикки (Хинтикка [1953] [1955]). Пусть φ – незапертая (конечная или бесконечная) нить в П-дереве, Σ – множество неисключенных формул вывода, принадлежащих φ, и $T(\Sigma)$ – множество термов из формул в Σ. Тогда справедлива

Лемма 3.3.5. Для любых формул A *и* B *из* Σ *верно*:

$A \in \Sigma \Rightarrow \neg A \notin \Sigma$,
$\neg\neg A \in \Sigma \Rightarrow A \in \Sigma$,
$A \,\&\, B \in \Sigma \Rightarrow A \in \Sigma$ *и* $B \in \Sigma$,
$\neg(A \,\&\, B) \in \Sigma \Rightarrow \neg A \in \Sigma$ *или* $\neg B \in \Sigma$,
$A \vee B \in \Sigma \Rightarrow A \in \Sigma$ *или* $B \in \Sigma$,
$\neg(A \vee B) \in \Sigma \Rightarrow \neg A \in \Sigma$ *и* $\neg B \in \Sigma$,
$A \supset B \in \Sigma \Rightarrow \neg A \in \Sigma$ *или* $B \in \Sigma$,
$\neg(A \supset B) \in \Sigma \Rightarrow A \in \Sigma$ *и* $\neg B \in \Sigma$,
$\forall\alpha A(\alpha) \in \Sigma \Rightarrow A(\alpha/t) \in \Sigma$, *для всех* t *из* $T(\Sigma)$,
$\neg\forall\alpha A(\alpha) \in \Sigma \Rightarrow \neg A(\alpha/t) \in \Sigma$, *для некоторого* t *из* $T(\Sigma)$,
$\exists\alpha A(\alpha) \in \Sigma \Rightarrow A(\alpha/t) \in \Sigma$, *для некоторого* t *из* $T(\Sigma)$,
$\neg\exists\alpha A(\alpha) \in \Sigma \Rightarrow \neg A(\alpha/t) \in \Sigma$, *для всех* t *из* $T(\Sigma)$.

Доказательство. Разберем сначала условия 1-8. (1) По условию, φ незаперта, значит, она содержит только недостигнутые блоки. При-

чем, последней целью в этих блоках является цель – противоречие. Значит, эта цель не достигнута, т.е. в множестве вывода нет противоречащих формул. Условия (2), (3) и (6) соответствуют, в том же порядке, BMV-правилам ¬и, &и, ¬∨и, которые применяются автоматически. (4) Если ¬(A & B) ∈ Σ, то по правилу *вц*¬ текущей целью становится формула A & B. Разбиваем эту цель на подцели A и B, начиная работать с первой. Если A недостижима, то в Σ вводится ¬A в качестве посылки. Если A достижима, то в качестве посылки в Σ вводится ¬B. (Один из этих случаев должен иметь место, так как A & B недостижима (φ незаперта)). Случай 8 доказывается аналогично. (5) Если A ∨ B ∈ Σ и ¬A ∈ Σ, то формула B выводима по ∨и. Если A ∨ B ∈ Σ и ¬A ∉ Σ, то к формуле A ∨ B применяется *вц*-правило *вц*∨ и текущей целью становится формула ¬A. Из нее по правилу *пц*¬ вводим формулу ¬¬A в качестве посылки в множество Σ. Из ¬¬A по ¬и получаем A. Случай 7 доказывается аналогично.

Теперь разберем условия 9-12. (9) Если ∀αA(α) ∈ Σ, то квантор общности снимается по новой неабсолютной переменной x; в выводе появляется формула A(x) и на формулу ∀αA(α) ставится метка M1, временно запрещающая применение к этой формуле правила ∀и. Данная метка в дальнейшем снимается, если: (а) все формулы вида ∀αA(α) из множества Σ отмечены M1 и (б) все иные формулы из множества Σ отмечены M0, т.е. к ним применены те или иные правила алгоритма. В такой ситуации к каждой формуле вида ∀αA(α) из множества Σ применяется правило *вц*∀. Согласно формулировке этого правила, квантор общности снимается по всем термам множества Σ, т.е. по всем термам из T(Σ). Случай (12) доказывается аналогично: к формуле ¬∃αA(α) применяется правило ¬∃и и выводится формула ∀α¬A(α). (10) Если ¬∀αA(α) ∈ Σ, то по правилу вы¬ текущей целью становится формула ∀αA(α). Применяем к этой цели правило *пц*∀ и получаем формулу A(y), где y – новая абсолютная переменная. Так как A(y) недостижима, то применяем к ней правило *пц*¬ и в множество Σ вводится ¬A(y) в качестве посылки. При этом на формулу ¬∀αA(α) ставится метка M5, не позволяющая еще раз применить к ней правило вы¬. (11) Если ∃αA(α) ∈ Σ, то по правилу ∃и получаем формулу A(y), где y – новая абсолютная переменная. При этом на формулу ∃αA(α) ставится метка M0, не позволяющая еще раз применить к ней правило ∃и. □

Определение 3.3.6. Литералами для формулы A называется множество элементарных формул или их отрицаний в A.

Например, для формулы $\forall x(\forall y P(x, y) \supset \forall z Q(z, x))$ литералами будут следующие формулы $P(t, t_1)$, $Q(t_2, t_3)$, $\neg P(t, t_1)$, $\neg Q(t_2, t_3)$, где t, t_1, t_2, t_3 – произвольные термы логики предикатов.

Теорема 3.3.7. Если в алгоритме из Γ *не доказуема* A, *значит*, $\Gamma \not\models A$, *т.е., по контрапозиции, если* $\Gamma \models A$, *то* $\Gamma \vdash_{ал} A$.

Доказательство. Если в алгоритме из Γ не доказуема A, то имеется *недостигнутое* П-дерево (связь алго-вывода и П-дерева). По определению 3.4.4, недостигнутое П-дерево содержит незапертую нить φ. Множество формул вывода Σ из φ является множеством Хинтикки (*лемма 3.3.5*). Значит, множество *литералов* из Σ представляет собой такую интерпретацию, при которой все формулы из Γ истинны, а формула A ложна (Хинтикка [1955]). Таким образом, если в алгоритме из Γ не доказуема A, то $\Gamma \not\models A$. □

ГЛАВА IV

ГЕНЕРИРОВАНИЕ ТЕОРЕМ В ИНТУИЦИОНИСТСКОЙ ЛОГИКЕ

1. Интуиционистская логика

1.1. *Основные идеи интуиционистской логики*

Интуиционистская логика возникла как результат развития одноименного направления в основаниях математики, начало которому положил известный голландский математик Л. Э. Я. Брауэр в своей докторской диссертации "Об обосновании математики", написанной в 1907 году. Появление интуиционизма было тесно связано с обнаружением парадоксов в теории множеств Г. Кантора, основанной на понятии актуальной бесконечности, т.е. на идее "отвлечения от незавершенности и незавершимости процесса образования бесконечного множества, от невозможности задать такое множество полным списком его элементов" (Успенский [1960]).

Как известно, математика и философия античности использовала в основном понятие потенциальной бесконечности, которое выражает возможность всегда продолжить процесс, возможность "всегда-дальше-конструирования". Проблема актуальной бесконечности была поднята прежде всего в христианской теологии, где Богу приписывали как свойство бесконечности, так и свойство актуальности. В конце Средних веков и в Новое время появились ученые и философы, которые выступали против употребления понятия актуального бесконечного. К ним принадлежали такие известные математики, как Декарт, Гаус, Кронекер, и философы – Гоббс и Кант.

Представители интуиционистской математики проводят резкую грань между логикой и математикой. А. Гейтинг так описывает эту ситуацию: "Интуиционистская математика – это некоторая мыслительная деятельность, и каждый язык, в том числе и формализованный, является для нее только вспомогательным средством сообщения. Принципиально невозможно установить систему формул, которая была бы равноценна с интуиционистской математикой, так как возможности мышления невозможно охватить конечным числом ранее установленных правил... Для построения математики установление общезначимых логических законов не обязательно; эти законы открываются в каждом отдельном случае рассматриваемой математической системы. Но языковое сообщение, образованное потребностями ежедневной жизни, выступает в форме логического закона, который она предпосылает как данный" (Бохеньский [1978, с. 342]).

Интуиционисты считают логику частью математики. Логическое предложение есть, по их мнению, математическое предложение только крайнего уровня обобщения. Таким образом, логика не предпосылается, а абстрагируется из математики. Коль скоро это произошло, то ее можно формализовать, но это уже второстепенное дело. Логика не может служить, согласно этому пониманию, основой математики. С. Клини цитирует Вейля: "Согласно его (Л. Брауэра. – *Авт*.) взглядам и пониманию истории, классическая логика была абстрагирована от математики конкретных множеств и их подмножеств.... Забывая об этом ее ограниченном происхождении, впоследствии эту логику приняли ошибочно за нечто высшее и первичное по отношению ко всей математике и в конце концов стали применять без какого бы то ни было оправдания к математике бесконечных множеств" [1973, с. 235]. Поэтому интуиционисты считают, что законы и правила логики не являются универсальными, и что существуют только логики, специфицированные для конкретных областей (более полно о взглядах интуиционистов см. Суханов [1973]).

"Интуиционизм основывается также на некоторых существенных идеализациях. Так, принимается принцип сохранности, заключающийся в том, что, если истинность некоторого утверждения установлена, то оно остается истинным и в будущем. Используется также абстракция потенциальной осуществимости (бесконечности), состоящая в том, что мы отвлекаемся от ограниченности наших ресурсов в пространстве и во времени. Итак, одна из целей, породивших интуиционизм (и конструктивный подход в целом), состоит в том, чтобы не допускать использования в науке неэффективных методов. Таким образом, логика, в которой строятся интуиционистские теории, должна существенным образом отличаться от классической логики, ибо последняя не в состоянии гарантировать конструктивности рассуждений" (Шрамко [1997, с.22-23]). По этой причине в интуиционистской логике не принимаются законы исключенного третьего и снятия двойного отрицания, а также отбрасывается целый ряд других логических законов.

Интуиционизм имел довольно длинную историю развития в математике. Его предшественниками считаются Л. Кронекер и Г. Пуанкаре, а основателем, как было сказано, явился Л. Е. Я. Брауэр. Впервые сформулировал и формализовал интуиционистскую логику А. Гейтинг в 1930 году. В дальнейшем, трудами многих ученых значительная часть математики была перестроена на интуиционистских принципах.

1.2. *Интуиционистское понимание логических операторов*

Главная особенность интуиционистского понимания суждений заключается в том, что всякое утверждение А истолковывается в

смысле "А установимо", "А конструктивно доказано". Из сказанного ясно, что в интуиционизме смысл логических операторов понимается иначе, чем в классической логике. Рассмотрим это понимание.

Постулаты установимости для логики высказываний:

1. Конъюнкция A & B установима тогда и только тогда, когда установимо A и установимо B.
2. Дизъюнкция A $\vee$ B установима тогда и только тогда, когда установимо A или установимо B.

Здесь очевидно отличие от классической дизъюнкции, так как в последнем случае мы всегда имеем право утверждать A $\vee \neg$A – даже тогда, когда нам не известно, имеет ли место A или имеет место $\neg$A.

3. Импликация A $\supset$ B установима тогда и только тогда, когда установимость B может быть сведена к установимости A, т.е. если всякий раз, когда будет установлено A, из этого факта можно извлечь установимость B.
4. Константа ложь ($\perp$) понимается как некоторое фиксированное высказывание, которое заведомо не является установимым.
5. Высказывание $\perp \supset$ B считается установимым ввиду того, что установимость антецедента $\perp$ невозможна, и потому можно говорить о некотором тривиальном сведении установимости консеквента B к установимости антецедента.
6. Отрицание $\neg$A установимо тогда и только тогда, когда A неустановимо. Установимость $\neg$A равносильно установимости A $\supset \perp$.

1.3. *Интуиционистское исчисление высказываний*

Как уже было сказано, первый вариант интуиционистской логики принадлежит А. Гейтингу. Однако сам он не считал этот вариант интуиционистского исчисления высказываний полной формализацией интуиционистских принципов рассуждения. Он предпочитал говорить скорее о том, что существует несколько исчислений интуиционистского типа, каждое из которых может называться интуиционистской логикой. Здесь мы приводим эквивалентное системе А. Гейтинга конструктивное исчисление высказываний, изложенное П.С. Новиковым в работе [1977]. В силу равносильности (A $\supset \perp$) и $\neg$A в рассматриваемой системе не вводится знак $\neg$.

Следующие девять выражений служат схемами аксиом:

A1: A $\supset$ (B $\supset$ A),
A2: (A $\supset$ (B $\supset$ C)) $\supset$ ((A $\supset$ B) $\supset$ (A $\supset$ C)),

A3: $\bot \supset A$,
A4: $(A \,\&\, B) \supset A$,
A5: $(A \,\&\, B) \supset B$,
A6: $(A \supset B) \supset ((A \supset C) \supset (A \supset (B \,\&\, C)))$,
A7: $A \supset (A \vee B)$,
A8: $B \supset (A \vee B)$,
A9: $(A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C)))$.

Правилом вывода является modus ponens:

$$\frac{A,\ A \supset B}{B\ .}$$

Определения вывода и доказательства являются стандартными, т.е. аналогичны классической логике высказываний. По определению вводятся следующие логические связки:

(а) $(A \equiv B) \Leftrightarrow (A \supset B) \,\&\, (B \supset A)$,
(б) $\neg A \Leftrightarrow (A \supset \bot)$.

Зафиксируем некоторые факты, относящиеся к интуиционистской логике.

Во-первых, укажем, что следующие формулы являются невыводимыми в исчислении:

1. $(A \supset B) \supset (\neg A \vee B)$,
2. $\neg(A \,\&\, \neg B) \supset (A \supset B)$,
3. $\neg(\neg A \,\&\, \neg B) \supset (A \vee B)$,
4. $(\neg A \supset B) \supset (A \vee B)$,
5. $((A \supset B) \supset B) \supset (A \vee B)$,
6. $\neg(\neg A \vee \neg B) \supset (A \,\&\, B)$,
7. $\neg(A \supset \neg B) \supset (A \,\&\, B)$,
8. $(\neg A \supset \neg B) \supset (B \supset A)$,
9. $(\neg A \supset B) \supset (\neg B \supset A)$,
10. $(\neg A \supset B) \supset ((\neg A \supset \neg B) \supset A)$,
11. $A \supset (\neg B \vee B)$,
12. $\neg\neg A \supset A$,
13. $(\neg A \supset A) \supset A$,
14. $\neg A \vee A$,
15. $((A \supset B) \supset A) \supset A$,
16. $(\neg\neg A \supset A) \supset (\neg A \vee A)$,
17. $(A \supset B) \vee (B \supset A)$,
18. $\neg\neg(A \vee B) \supset (\neg\neg A \vee \neg\neg B)$,
19. $\neg\ (A \,\&\, B) \supset (\neg A \vee \neg B)$.

Во-вторых, так как множество теорем интуиционистской логики высказываний составляет подмножество множества теорем классической логики, то из этого вытекает, что интуиционистское исчисление высказываний является непротиворечивым.

В-третьих, интуиционистское исчисление высказываний не имеет точной конечной модели (Гедель). Однако существует точная счетная модель интуиционистского исчисления высказываний.

В-четвертых, если формула А выводима в классическом исчислении высказываний, то формула $\neg\neg$A выводима в интуиционистском исчислении высказываний. (Гливенко).

1.4. *Семантика типа Крипке для интуиционистской логики*

В настоящее время для интуиционистской логики существует хорошая семантика типа Крипке. В этой семантике интуиционистская модельная структура представляет собой пару M = <W, R>, где W есть непустое множество "возможных миров", R – рефлексивное и транзитивное отношение на W (отношение достижимости между мирами).

Пусть $|A|^M_\alpha = \boldsymbol{и}$ означает "А истинно в мире α", а $|A|^M_\alpha = \boldsymbol{л}$ означает "А ложно в мире α". Тогда истинность и ложность пропозициональных переменных определяется с соблюдением следующих условий (для произвольных α, $\beta \in W$ и для любой пропозициональной переменной p_i):

а) $|p_i|^M_\alpha = \boldsymbol{и}$ и $R\alpha\beta \Rightarrow |p_i|^M_\beta = \boldsymbol{и}$ – монотонность (принцип сохранности для пропозициональных переменных),

б) $|p_i|^M_\alpha = \boldsymbol{и}$ или $|p_i|^M_\alpha = \boldsymbol{л}$ – полнота,

в) Неверно, что одновременно $|p_i|^M_\alpha = \boldsymbol{и}$ и $|p_i|^M_\alpha = \boldsymbol{л}$ – непротиворечивость.

Условия истинности и ложности для формул, не являющихся пропозициональными переменными, задаются посредством следующих определений:

(1) $|A \,\&\, B|^M_\alpha = \boldsymbol{и} \Leftrightarrow |A|^M_\alpha = \boldsymbol{и}$ и $|B|^M_\alpha = \boldsymbol{и}$,
$|A \,\&\, B|^M_\alpha = \boldsymbol{л} \Leftrightarrow |A|^M_\alpha = \boldsymbol{л}$ или $|B|^M_\alpha = \boldsymbol{л}$,

(2) $|A \vee B|^M_\alpha = \boldsymbol{и} \Leftrightarrow |A|^M_\alpha = \boldsymbol{и}$ или $|B|^M_\alpha = \boldsymbol{и}$,
$|A \vee B|^M_\alpha = \boldsymbol{л} \Leftrightarrow |A|^M_\alpha = \boldsymbol{л}$ и $|B|^M_\alpha = \boldsymbol{л}$,

(3) $|\neg A|^M_\alpha = \boldsymbol{и} \Leftrightarrow \dot\forall\, \beta(R\alpha\beta \Rightarrow |A|^M_\beta = \boldsymbol{л})$,
$|\neg A|^M_\alpha = \boldsymbol{л} \Leftrightarrow \dot\exists\, \beta(R\alpha\beta$ и $|A|^M_\beta = \boldsymbol{и})$,

(4) $|A \supset B|^M_\alpha = \boldsymbol{и} \Leftrightarrow \dot\forall\, \beta(R\alpha\beta \Rightarrow (|A|^M_\beta = \boldsymbol{л}$ или $|B|^M_\beta = \boldsymbol{и}))$,
$|A \supset B|^M_\alpha = \boldsymbol{л} \Leftrightarrow \dot\exists\, \beta(R\alpha\beta$ и $|A|^M_\beta = \boldsymbol{и}$ и $|B|^M_\beta = \boldsymbol{л})$,

Формула А истина в некоторой модельной структуре, если и только если данная формула истинна в каждом мире из данной структуры.

А является общезначимой, если и только если А истина во всех структурах.

Относительно так заданной семантики доказываются метатеоремы о непротиворечивости и полноте интуиционистской логики.

2. Системы генерирования доказательств для интуиционистской логики

2.1. *Секвенциальные исчисления*

Как выше говорилось, известны различные системы генерирования теорем для интуиционистской логики высказываний, основанные на базе теории резолюций, семантических таблицах, матрицах. Весьма разнообразны алгоритмы поиска доказательства, разрабатываемые для секвенциальных исчислений (некоторые из них в связи с родством с системами натурального вывода будут рассмотрены ниже).

Первую разрешающую процедуру для интуиционистской логики высказываний предложил сам Г. Генцен. Она основывалась на секвенциальном исчислении, формулировку которого мы с незначительными изменениями приводим по его работе "Исследования логических выводов" (см. Генцен [1967]).

Основной секвенцией является секвенция вида $A \to A$.

Схемы структурных фигур заключения:

Утончение

$$\frac{\Gamma \to \Theta}{A, \Gamma \to \Theta} \qquad \frac{\Gamma \to \Theta}{\Gamma \to \Theta, A}$$

Сокращение

$$\frac{A, A, \Gamma \to \Theta}{A, \Gamma \to \Theta} \qquad \frac{\Gamma \to \Theta, A, A}{\Gamma \to \Theta, A}$$

Перестановка

$$\frac{\Delta, A, B, \Gamma \to \Theta}{\Delta, B, A, \Gamma \to \Theta} \qquad \frac{\Gamma \to \Theta, A, B, \Delta}{\Gamma \to \Theta, B, A, \Delta}$$

Сечение

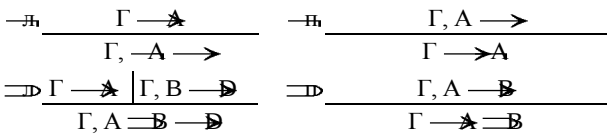
$$\frac{\Gamma \to A \quad A, \Delta \to \Theta}{\Gamma, \Delta \to \Theta}$$

Схемы логических фигур заключения:

$$\&л\ \frac{\Gamma, A \to \Theta}{\Gamma, A \,\&\, B \to \Theta} \qquad \frac{\Gamma, B \to \Theta}{\Gamma, A \,\&\, B \to \Theta}$$

$$\&и\ \frac{\Gamma \to A \mid \Gamma \to B}{\Gamma \to A \,\&\, B} \qquad \vee л\ \frac{\Gamma, A \to \Theta \mid \Gamma, B \to \Theta}{\Gamma, A \vee B \to \Theta}$$

$$\vee и\ \frac{\Gamma \to A}{\Gamma \to A \vee B} \qquad \frac{\Gamma \to B}{\Gamma \to A \vee B}$$



Данные схемы фигур обладают двумя важными свойствами:

(1) все схемы являются обратимыми, т.е. схемы, полученные заменой верхней секвенции на нижнюю и наоборот, являются выводимыми в данном исчислении.

(2) все схемы фигур, кроме сечения, обладают свойством подформульности, т.е. верхние секвенции содержат формулы, являющиеся подформулами формул, входящих в нижнюю секвенцию.

Доказательство основной теоремы об устранимости сечения в интуиционистском секвенциальном исчислении решает для него проблему разрешения.

Система генерирования доказательств предполагала создание множества основных секвенций $A \to A$, где A являлась подформулой доказываемой секвенции, и построение всевозможных постепенно усложнявшихся выводов на основе этого множества. Процедура заканчивала работать, когда доказываемая секвенция попадала во множество секвенций с построенным выводом, или когда начинались доказываться более сложные секвенции, чем заданная. Таким образом, помимо нужной секвенции процедура доказывала ряд других секвенций, что усложняло поиск доказательства. Сложность процедуры вывода заметно росла с ростом числа переменных и логических связок формул исходной секвенции.

Основываясь на доказательстве полноты интуиционистской логики М. Фиттинга [1969], Дж. Андервуд [1990] рассмотрела, как доказательство разрешимости интуиционистской логики может быть включено в формальную систему типа NuPRL. Введя соответствие между секвенциями и узлами таблицы Фиттинга, она доказала, что каждое множество секвенций, соответствующее таблице, содержит доказуемую секвенцию или существует модель Крипке такая, что каждая секвенция из этого множества опровержима на определенной вершине. Доказательство ведется индукцией по числу формул, которые могут быть добавлены к множеству секвенций согласно правилам таблиц. Программа Дж. Андервуд принимает на входе секвенцию, а возвращает на выходе доказательство этой секвенции или контрмодель. Программа использует проверку циклов и пытается построить бесповторный вывод снизу вверх в секвенциальном исчислении, похожем на исчисление Г. Генцена. Наличие процедуры проверки циклов заметно усложняло вычисления, производимые программой, и встала проблема построения исчисления, для ко-

торого разрешающая процедура не содержала бы проверки циклов, и тем самым нахождения более эффективного метода автоматического поиска доказательства теорем для интуиционистской логики.

Одним из путей решения этой проблемы является ограничение применения структурных правил вывода секвенциального исчисления, которые зачастую оказываются причиной возникновения циклов. В силу доказанной основной теоремы Генцена правило сечения можно не принимать для интуиционистской логики. Правило утончения тоже устраняется, если принять в качестве основной секвенцию вида $\Gamma, A, \Delta \to A$. Наиболее трудным из всех структурных правил является правило перестановки. От него частично возможно отказаться, если ввести шкалу последовательности применения логических правил вывода, основываясь на работе С.К. Клини [1967].

Правило сокращения же элиминируется путем существенной перестройки логических правил вывода. Одним из вариантов такой перестройки является секвенциальное исчисление, свободное от сокращений (англ. "contraction free"), предложенное Р. Дикхофом [1992] и Дж. Худельмаером [1988], которые переоткрыли результаты работ Н.Н. Воробьева [1958]. Следуя Д. Корну [1999], назовем данное исчисление "LJ_{cf}".

Основными секвенциями исчисления LJ_{cf} являются секвенции вида

1. $\Gamma_1, A, \Gamma_2 \to A$,
2. $\Gamma_1, \bot, \Gamma_2 \to C$, где $\bot$ константа ложь.

Освобождение от сокращения достигается путем введения различных правил для импликации слева, применение которых зависит от вида формулы А.

1. Если $A = A1 \supset A2$, то тогда применяется следующее правило:

$$\supset\supset\text{л}\ \frac{\Gamma, A_2 \supset B, A_1 \to A_2 \quad | \quad \Gamma, B \to D}{\Gamma, (A_1 \supset A_2) \supset B \to D}$$

Это правило является комбинацией двух правил:

$$\cfrac{\cfrac{\Gamma, A_2 \supset B, A_1 \to A_2}{\Gamma, (A_1 \supset A_2) \supset B \to A_1 \supset A_2} \quad | \quad \Gamma, B \to D}{\Gamma, (A_1 \supset A_2) \supset B \to D}$$

На первом шаге (если смотреть снизу вверх) применяется обычное правило $\supset$л, а на втором вводится уже новое правило – некоторая модификация правила $\supset$п, которая заменяет сохраненную импликацию $(A_1 \supset A_2) \supset B$ на $A_2 \supset B$. Такое правило является вполне конструктивным, так как оно доказуемо в оригинальном генценовском исчислении:

$$\dfrac{\dfrac{\dfrac{A_2 \to A_2}{A_1, A_2, \Gamma \to A_2} \quad B \to B}{A_2, \Gamma \to A_1 \supset A_2} \quad B, A_2, \Gamma \to B}{A_2, \Gamma, (A_1 \supset A_2) \supset B \to B} \quad A_1, A_2 \supset B, \Gamma \to A_2}{\Gamma, (A_1 \supset A_2) \supset B \to A_2 \supset B} \quad A_2 \supset B, \Gamma \to A_1 \supset A_2}{\Gamma, (A_1 \supset A_2) \supset B \to A_1 \supset A_2}\ \text{Cut}$$

2. Если $A = \neg A_1$.

У Р. Дикхофа отрицание представлено в виде импликации $A_1 \supset \bot$, где $\bot$ является синтаксическим выражением противоречия. Поэтому для отрицания правило является тем же самым, что и для импликации. Но в следующей версии исчисления Р. Дикхофа, сделанной У. Москато (см. Миглиоли и др. [1994]), было предложено существенное изменение этого правила. Так как $\to \bot \supset B$ является аксиомой исчисления, то правило для отрицания предстает в следующем виде:

$$\neg\supset\text{л}\ \frac{\Gamma, A_1 \to \bot \ \mid\ \Gamma, B \to D}{\Gamma, \neg A_1 \supset B \to D\,.}$$

3. Если $A = A_1 \,\&\, A_2$, то импликация $A \supset B$ заменяется следующей формулой $A_1 \supset (A_2 \supset B)$ и правило имеет вид:

$$\&\supset\text{л}\ \frac{\Gamma, A_1 \supset (A_2 \supset B) \to D}{\Gamma, (A_1 \,\&\, A_2) \supset B \to D.}$$

4. Если $A = A_1 \vee A_2$, то импликация $A \supset B$ заменяется формулой $(A_1 \supset B) \,\&\, (A_2 \supset B)$ и правило имеет вид:

$$\vee\supset\text{л}\ \frac{\Gamma, (A_1 \supset B) \,\&\, (A_2 \supset B) \to D}{\Gamma, (A_1 \vee A_2) \supset B \to D.}$$

5. Если A атомарная формула, скажем p, то применяется обычное правило разложения импликации слева, но без дублирования ее в верхней секвенции.

$$\supset\text{л}^*\ \frac{\Gamma \to p \ \mid\ \Gamma, B \to D}{\Gamma, p \supset B \to D.}$$

В версии У. Москато возникает дополнительное правило: импликация $p \supset B$ заменяется формулой B, если в антецеденте заключения имеется p.

$$p\supset\text{л}\ \frac{\Gamma, p, B \to D}{\Gamma, p, p \supset B \to D.}$$

Таким образом, исчисление LJ_{cf} может быть представлено так:

Основные секвенции:

1. $\Gamma, A, \Delta \to A$,
2. $\Gamma, \bot, \Delta \to D$.

Правила вывода:

$$\&л\ \frac{\Gamma, A, B \to D}{\Gamma, A \,\&\, B \to D} \qquad \&п\ \frac{\Gamma \to A \mid \Gamma \to B}{\Gamma \to A \,\&\, B}$$

$$\vee л\ \frac{\Gamma, A \to D \mid \Gamma, B \to D}{\Gamma, A \vee B \to D}$$

$$\vee и\ \frac{\Gamma \longrightarrow A}{\Gamma \longrightarrow A \vee B} \qquad \frac{\Gamma \longrightarrow B}{\Gamma \longrightarrow A \vee B}$$

$$\supset п\ \frac{\Gamma, A \to B}{\Gamma \to A \supset B} \qquad \vee\supset л\ \frac{\Gamma, (A_1 \supset B) \,\&\, (A_2 \supset B) \to D}{\Gamma, (A_1 \vee A_2) \supset B \to D}$$

$$\supset\supset и\ \frac{\Gamma \longrightarrow A \mid \Gamma, B \longrightarrow D}{\Gamma, A \supset B \longrightarrow D} \qquad \supset\supset и\ \frac{\Gamma, A \longrightarrow B}{\Gamma \longrightarrow A \supset B}$$

$$\&\supset л\ \frac{\Gamma, A_1 \supset (A_2 \supset B) \to D}{\Gamma, (A_1 \,\&\, A_2) \supset B \to D} \qquad p\supset л\ \frac{\Gamma, p, B \to D}{\Gamma, p, p \supset B \to D}$$

$$\neg\supset л\ \frac{\Gamma, A_1 \to \bot \mid \Gamma, B \to D}{\Gamma, \neg A_1 \supset B \to D} \qquad \supset\supset л\ \frac{\Gamma, A_2 \supset B, A_1 \to A_2 \mid \Gamma, B \to D}{\Gamma, (A_1 \supset A_2) \supset B \to D, \text{ где } A_2 \neq \bot}$$

Определение вывода является стандартным. Показана (Дикхоф [1992], Миглиоли и др. [1994]) семантическая корректность и полнота исчисления LJ_{cf}, а также его разрешимость. В качестве примера можно привести поиск доказательства для $\neg\neg A \supset A$.

$$\cfrac{\cfrac{A \to \bot \mid \bot \to A}{\neg\neg A \to A}\ \supset п}{\to \neg\neg A \supset A}\ \neg\supset л$$

В работе К. Вайха [1998] предложены две разрешающие процедуры для исчисления Р. Дикхофа: одна объединяет результаты более ранних работ (Худельмайер [1993]), вторая является альтернативой первой для импликативного фрагмента интуиционистской логики высказываний.

Первая процедура является алгоритмом, строящим доказательство или контрмодель секвенции $\Gamma \to A$. Алгоритм состоит из двух частей: поиск 1 и поиск 2. Вначале происходит поиск 1, т.е. применяются снизу вверх правила $\supset$п, &п, &л, $\vee$л, $\vee\supset$л, &$\supset$л, р$\supset$л, которые обладают свойством сохранения контрмодели при переходе от посылок к заключению. Если доказательство не построено, происходит дальнейший анализ – поиск 2. А именно анализируются верхние секвенции, которые не являются основными. Если они имеют вид $\Gamma_1, (B_1 \supset B_2) \supset C, \Gamma_2 \to D$, то согласно правилу $\supset\supset$л происходит переход к доказательству секвенций $B_1, \Gamma_1, B_2 \supset C, \Gamma_2 \to B_2$ и $\Gamma_1, C, \Gamma_2 \to D$, кото-

рое происходит далее в режиме поиска 1. Аналогично в случае правила ¬⊃л. В противном случае алгоритм пытается применить ∨п, если это не удается, то он выдает контрмодель. При построении контрмодели используются следующие правила опровержения:

$$\frac{\Gamma \nvdash B_1 \supset C_1 \mid \cdots \mid \Gamma \nvdash B_n \supset C_n}{\Gamma \nvdash p/\bot} \quad \text{Опр1}, \; n \geq 0$$

$$\frac{\Gamma \nvdash B_1 \supset C_1 \mid \cdots \mid \Gamma \nvdash B_n \supset C_n \mid \Gamma \nvdash A_0 \mid \Gamma \nvdash A_1}{\Gamma \nvdash A_0 \vee A_1} \quad \text{Опр2}, \; n \geq 0$$

где:

а) $\Gamma \nvdash B_1 \supset C_1$ означает недоказуемость секвенции $\Gamma \to B_1 \supset C_1$,

б) $\Gamma \nvdash p/\bot$ означает $\Gamma \nvdash p$ или $\Gamma \nvdash \bot$,

в) $\Gamma = \Gamma_0, (B_1 \supset C_1) \supset D_1, \ldots, (B_n \supset C_n) \supset D_n, \Gamma_n$,

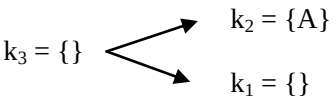
г) к секвенциям $\Gamma \to p$, $\Gamma \to \bot$, $\Gamma \to A_0 \vee A_1$ уже нельзя применить правила поиска 1,

д) импликации $(B_1 \supset C_1) \supset D_1 \ldots (B_n \supset C_n) \supset D_n$ – это все подобного вида формулы, которые входят в Γ.

Таким образом, контрмоделью для секвенции $\Gamma \to A$ (являющейся заключением правил Опрl и Опр2) будет дерево К с корнем ω, в котором истинны все формулы из множества {p: p – атомарная формула, являющаяся элементом списка Γ} и из которого достижимы вершины, опровергающие посылки правил Опрl и Опр2. Например,

$$\frac{\begin{array}{c|c} & A \to \bot \\ \hline \to A & \to \neg A \end{array}}{\to A \vee \neg A} \quad n = 0$$

Миром, опровергающим $\to A$, является $k_1 = \{\}$, миром, опровергающим $\to \neg A$, является $k_2 = \{A\}$. Таким образом, контрмоделью для секвенции является дерево с корнем $k_3 = \{\}$, из которого достижимы k_1 и k_2.



Вторая процедура ограничивается импликативным фрагментом (хотя вполне вероятно, что существует и полная версия второй процедуры для всей пропозициональной части интуиционистской логики, однако Вайху не удалось ее опубликовать до сих пор). Она состоит в следующем. Посредством последовательного применения правила ⊃п секвенция приводится к виду $\Gamma \to p$, где p – это пропозициональная переменная, Γ – последовательность атомарных или им-

пликативных формул, общим видом которых является $((A'_1 \supset Q_1) \supset \ldots \supset ((A'_k \supset Q_k) \supset Q)...)$, где $(A'_i \supset Q_i)$ есть формула $(A_{i1} \supset ... \supset (A_{in} \supset Q_i)\ldots)$, $n \geq 0$, $k \geq 0$. Формула Q называется вожаком (head) импликативной формулы. Дальнейшее доказательство секвенции $\Gamma \rightarrow p$ сводится к анализу тех формул, вожак которых эквивалентен сукцеденту.

Вводится понятие цепи и сигнификата. Цепью в Г от p_i до p_n будет называться список импликаций $p_1 \supset p_2,..., p_{n-1} \supset p_n$, каждая из которых находится в Г. Атомарная формула p_i называется сигнификатом для $\Gamma \rightarrow p$, если существует цепь от p_i до p в Г. Множество сигнификатов определяется следующим образом: если $p_1 \supset p_2$ принадлежит Г и p_2 сигнификат для $\Gamma \rightarrow p_n$, то и p_1 тоже является сигнификатом для $\Gamma \rightarrow p_n$. Имеет место сследующий факт: если имеется цепь от p_1 до p_n в Г и доказуемо $\Gamma \rightarrow p_1$, то доказуемо $\Gamma \rightarrow p_n$.

Далее вводится следующее обобщенное правило для импликации.

$$\frac{A'_1, \Gamma_1, Q_1 \supset Q, \Gamma_2 \rightarrow Q_1 \mid \ldots \mid A'_k, \Gamma_1, Q_k \supset Q, \Gamma_2 \rightarrow Q_k}{\Gamma_1, (A'_1 \supset Q_1) \supset \ldots ((A'_k \supset Q_k) \supset Q \ldots), \Gamma_2 \rightarrow Q} \supset^*$$

Путем разложения подобных формул, где Q – сигнификат, с помощью этого правила приходим либо к построению вывода, либо к получению контрмодели.

Однако в этих разрешающих процедурах включается перебор последовательностей применения (снизу вверх) правил вывода к импликативным формулам, находящимся в антецеденте секвенции, а также к дизъюнктивной формуле, находящейся в сукцеденте секвенции. Кроме того, для исчисления характерно отсутствие свойств подформульности и обратимости правил вывода. Это, конечно, осложняет вычисления и не может считаться преимуществом данной системы.

Другим вариантом решения проблемы ограничения применения правила сокращения является так называемая техника блокирования импликации (die Implikationssperrung), объединенная с введением списка переходов (die Überganglisten). Она развивается в рамках программы "ft" автоматического поиска доказательства теорем для интуиционистской логики, предложенной Д. Салиным (Салин и др. [1992]). Основанием этой программы является многосукцедентное секвенциальное исчисление. Назовем его вслед за Д. Корном [1999] исчислением LJ_{ft}.

Список переходов. Список переходов – это последовательность формул $[X_1,..., X_n]$ секвенции $\Gamma \rightarrow A$ в выводе π, которые являются левыми подформулами всех тех формул $X_1 \supset Y_1,\ldots, X_n \supset Y_n$, к которым было применено снизу вверх правило $\supset$п (формулировку правила см. ниже) в секвенциях, находящихся в выводе π ни-

же секвенции $\Gamma \to D$. Обозначение "список переходов" возникло из того факта, что X_i, при применении правила $\supset$п переходит в антецедент секвенции. Например, пусть L – некоторый список переходов для секвенции $\Gamma \to X \supset Y, \Delta$. Тогда посредством применения правила $\supset$п к этой секвенции получается посылка $\Gamma, X \to Y$, в которой уже отсутствует Δ и которой соответствует список переходов LX, т.е. $[X_1,..., X_n, X]$, где $L = [Xi,..., X_n]$. Такую версию правила $\supset$п обозначим $\supset$п-1.

С помощью идеи списка переходов возможно сформулировать достаточный критерий прекращения непродуктивного применения правила $\supset$п. Если список переходов для секвенции $\Gamma \to X \supset Y, \Delta$ уже содержит формулу X, то следует переход к секвенции $\Gamma \to Y$ без включения X в список переходов. Такую версию правила $\supset$п обозначим $\supset$п-2. Так как перехода некоторой формулы в антецедент не было, то список переходов остался без изменений. В общем случае в список переходов формула может быть занесена не более одного раза.

В правилах вывода список переходов будет отделяться от секвенции двоеточием. Выражение $L: \Gamma \to \Delta$ означает секвенцию $\Gamma \to \Delta$ со списком переходов L.

Блокирование импликации. Дабы запретить непродуктивное применение правила $\supset$л, в него вводится отмеченная импликация $\supset_\in$ как показатель, что к данной импликативной формуле уже нельзя применять снизу вверх правила вывода. Применение к этой импликации правила $\supset$л не разрешается. Эта форма импликации применяется к продублированной импликативной формуле посылки правила $\supset$л.

Соответственно, все формы отмеченной импликации устраняются из секвенции при применении правила $\supset$п-1 и заменяются обычной импликацией. Результат такого устранения и замены импликаций обозначим $\Gamma_{\not\in}$, что является сокращением для

$$\Gamma_{\not\in} = (\Gamma \setminus \{X \supset_\in Y | X \supset_\in Y \in \Gamma\}) \cup \{X \supset Y | X \supset_\in Y \in \Gamma\}).$$

Таким образом, исчисление LJ_{ft} может быть представлено так:

Правила вывода:

$$\&л\ \frac{L: \Gamma, A, B \to \Delta}{L: \Gamma, A \,\&\, B \to \Delta} \qquad \&п\ \frac{L: \Gamma \to A, \Delta \mid L: \Gamma \to B, \Delta}{L: \Gamma \to A \,\&\, B, \Delta}$$

$$\vee л\ \frac{L: \Gamma, A \to \Delta \mid L: \Gamma, B \to \Delta}{L: \Gamma, A \vee B \to \Delta} \qquad \vee п\ \frac{L: \Gamma \to A, B, \Delta}{L: \Gamma \to A \vee B, \Delta}$$

$$\supset н\text{-}1\ \frac{LA: \Gamma_{\not\in}, A \to B}{L: \Gamma, \to A \supset B, \Delta, \text{ где } A \notin L} \qquad \supset н\text{-}2\ \frac{L: \Gamma \to B}{L: \Gamma \to A \supset B, \Delta, \text{ где } A \in L}$$

$$\supset\text{л}\quad \frac{L\colon \Gamma, A \supset_{\in} B \to A, \Delta \qquad L\colon \Gamma, B \to \Delta}{L\colon \Gamma, A \supset B \to \Delta}$$

Отрицание $\neg A$ представлено в виде импликации $A \supset \bot$. Основными секвенциями исчисления LJ_{ft} являются секвенции вида:

1. $L\colon \Gamma_1, A, \Gamma_2 \to \Delta_1, A, \Delta_2$,
2. $L\colon \Gamma_1, \bot, \Gamma_2 \to \Delta$.

Понятие вывода: LJ_{ft}-выводимость для конечной секвенции $[]\colon \to A$ некоторой данной формулы A является LJ_{ft}-выводимостью A.

Показана (К о р н, К р е й т ц [1996], [1997]) корректность и полнота исчисления LJ_{ft}, а также его разрешимость. В качестве примера приведем поиск доказательства для $\neg\neg A \supset A$:

$$\begin{array}{ll}
[\neg\neg A, A]\colon \neg_{\in}\neg A, A \to \bot & \supset\text{п-2} \\ \hline
[\neg\neg A, A]\colon \neg_{\in}\neg A, A \to \neg A & \supset\text{л} \\ \hline
[\neg\neg A, A]\colon \neg\neg A, A \to \bot & \supset\text{п-1} \\ \hline
[\neg\neg A]\colon \neg_{\in}\neg A \to \neg A, A & \supset\text{л} \\ \hline
[\neg\neg A]\colon \neg\neg A \to A & \supset\text{п-1} \\ \hline
[]\colon \to \neg\neg A \supset A &
\end{array}$$

Как и исчисление Р. Дикхофа LJ_{cf}, данное исчисление не обладает свойством обратимости правил, так как правило $\supset$п-1 не является таковым. Несмотря на это в отношении подформульности данное исчисление является улучшением системы LJ_{cf}. Что касается перебора последовательностей применения правил, то как и в исчислении LJ_{cf}, он сохраняется. Оба исчисления LJ_{cf} и LJ_{ft} требуют правила перестановки, т.е. поиск доказательства в обоих исчислениях сопряжен с применением этого правила. Например, доказать следующую секвенцию в данных исчислениях $((((A \supset B) \supset A) \supset A) \supset B), (C \supset D) \supset R \to B$ невозможно, не применяя структурное правило перестановки (или процедуру перебора). Рассмотрим следующую последовательность применения правил вывода в исчислении LJ_{ft}:

$$\frac{\dfrac{\dfrac{\ldots \quad \Big| \quad LC\colon (C \supset D) \supset R, C, B \to D}{LC\colon ((((A \supset B) \supset A) \supset A) \supset B), (C \supset D) \supset R, C \to D}}{L\colon ((((A \supset B) \supset A) \supset A) \supset B), (C \supset D) \supset_{\in} R \to (C \supset D), B} \qquad \dfrac{\ldots}{L\colon ((((A \supset B) \supset A) \supset A) \supset B), R \to B}}{L\colon ((((A \supset B) \supset A) \supset A) \supset B), (C \supset D) \supset R \to B}$$

Секвенция $(C \supset D) \supset R, C, B \to D$ является недоказуемой. Негативную роль в этой неудачной попытке доказательства секвенции $((((A \supset B) \supset A) \supset A) \supset B), (C \supset D) \supset R \to B$ сыграло так называемое несущественное дополнение в виде формулы $(C \supset D) \supset R$. Однако если применять структурное правило перестановки, т.е. при-

менить правило импликации слева к формуле ((((A ⊃ B) ⊃ A) ⊃ A) ⊃ B), то данная секвенция оказывается доказуемой. Причиной такого негативного влияния являются логические правила с вычеркиванием, которые относятся к числу необратимых правил: для односукцедентных исчислений – это правило ⊃л, а для многосукцедентных – правило ⊃п. Именно здесь происходит потеря существенного для построения вывода вхождения формулы в сукцедент секвенции. Это приводит к необходимости включения в алгоритм поиска доказательства перебор последовательностей применения правила вывода ⊃л к различным импликативным формулам. Число таких последовательностей порядка n! по каждой секвенции вывода, где n – число таких импликативных формул.

2.2. *Префиксные исчисления*

Весьма интересным решением задачи поиска доказательства в интуиционистской логике является попытка отражения в синтаксической структуре исчисления некоторой дополнительной информации, например, информации о возможных мирах, в которых истинна та или иная формула. Такая идея реализуется, в частности, в префиксных исчислениях.

Понятие префикса использует в своих работах уже М. Фиттинг [1983], под которым он понимает последовательность натуральных чисел. В дальнейшем эту идею развивали С. Шмит и К. Крайтц [1996], а также Д. Корн [1999]. Приведем более разработанный вариант (с точки зрения автоматического поиска доказательства) такого исчисления – систему LKa А. Ваалера [2000]. Его исчисление для интуиционистской логики является переработкой классической системы LK Г. Генцена путем добавления специальных структур – меток, которые выполняют ту же роль, что и префиксы.

Вводятся новые объекты языка:

a, b, c,... – параметры вывода (inference-parameters),
U, V, W,... – переменные вывода (inference-variables).

Термом вывода считается параметр или переменная вывода. Меткой (label) является последовательность термов вывода, в том числе пустая последовательность ε. Причем, отождествляются следующие метки p, εp и pε, т.е. p = εp = pε. Определяется отношение ≤: p ≤ q, если существует такой t, что pt = q. Если некоторый терм вывода t входит в метку q, то p является *префиксом терма* t в q, если pt ≤ q. Множество меток Θ удовлетворяет условию *уникального префикса* (unique prefix property), если для любого терма вывода существует единственный префикс во всех метках из Θ.

Определяется функция σ подстановки меток как функция из множества переменных вывода в множество меток. Запись Zσ означает результат применения подстановки σ к некоторому объекту Z. Подстановка σ называется идемпотентной, если Zσσ = Zσ для любого Z. Подстановка σ считается связанной множеством меток Θ, если выполняется следующее условие: если t является последним термом в Uσ, то существуют такие метки pU и qt в Θ, что (pU)σ = (qt)σ.

Имеет место следующий факт:

Пусть Θ выполняет условие уникального префикса и σ является идемпотентной подстановкой, связанной этим множеством Θ, тогда Θσ тоже будет выполнять условие уникального префикса.

Правила LKa:

$$\&л\ \frac{\Gamma, p{:}A, p{:}B \to \Delta}{\Gamma, p{:}A \,\&\, B \to \Delta} \qquad \&п\ \frac{\Gamma \to p{:}A, \Delta \mid \Gamma \to p{:}B, \Delta}{\Gamma \to p{:}A \,\&B, \Delta}$$

$$\vee л\ \frac{\Gamma, p{:}A \to \Delta \mid \Gamma, p{:}B \to \Delta}{\Gamma, p{:}A \vee B \to \Delta} \qquad \vee п\ \frac{\Gamma \to p{:}A, p{:}B, \Delta}{\Gamma \to p{:}A \vee B, \Delta}$$

$$\supset л\ \frac{\Gamma, p{:}A \supset B \to pU{:}A, \Delta \mid \Gamma, pU{:}B \to \Delta}{\Gamma, p{:}A \supset B \to \Delta} \qquad \supset п\ \frac{\Gamma, p^*[n]{:}A \to p^*[n]{:}B, \Delta}{\Gamma \to p{:}A \supset B, \Delta}$$

где U – переменная вывода, не встречающаяся в заключении

где n – параметр, не встречающийся в заключении

$$\neg л\ \frac{\Gamma, p{:}\neg A \to pU{:}A, \Delta}{\Gamma, p{:}\neg A \to \Delta} \qquad \neg п\ \frac{\Gamma, p^*[n]{:}A \to \Delta}{\Gamma \to p{:}\neg A, \Delta}$$

где U – переменная вывода, не встречающаяся в заключении

где n – параметр, не встречающийся в заключении

Здесь знаком "*" помечена операция конкатенации (сочленения). Вхождения формул p:¬A и p:A ⊃ B в правилах ¬л и ⊃л, соответственно, называются сокращениями. Боковые (левое и правое), основные (главные) и внешние вхождения формул определяются обычным образом. Считается, что правила ⊃п и ¬п относятся к терму вывода n.

Вводится отношение контекстуальной эквивалентности на вхождениях формул в LKa-дерево:

– вхождения внешних формул в посылки правила вывода контекстуально эквивалентны вхождениям этих формул в заключение правила,

– вхождения боковых формул и сокращения не являются контекстуально эквивалентными вхождениям основной формулы заключения,

– если два логических правила вывода одного и того же типа, например, ¬л, имеют контекстуально эквивалентные основные вхождения формул, тогда их левые боковые вхождения формул,

правые боковые вхождения формул и, если есть, сокращения, тоже контекстуально эквивалентны.

Условие LKa-вывода: если правила одного и того же типа относятся к одному и тому же терму вывода n, то основные вхождения формул соответствующих правил являются контекстуально эквивалентными, а боковые вхождения формул обладают одинаковыми префиксами (метками).

δ есть σ-выводимость, если σ идемпотентно и σ связано относительно множества префиксов δ. Аксиомой считается секвенция формы Γ, p:A $\rightarrow$q:A, Δ, где p $\leq$ q. Секвенция S есть σ-аксиома, если Sσ является аксиомой. Конечная σ-выводимость δ с самой нижней секвенцией S является доказательством S, если все листья $\delta\sigma$ (т.е. самые верхние секвенции) являются аксиомами. Тогда считается, что σ закрывает δ или δ есть σ-доказательство.

В качестве примера можно привести поиск доказательства для $\neg\neg$A $\supset$ A:

$$\dfrac{\dfrac{\dfrac{}{\text{a:}\neg\neg\text{A, aUb:A} \rightarrow \text{a:A}}\ \neg\text{п}}{\text{a:}\neg\neg\text{A} \rightarrow \text{aU:}\neg\text{A, a:A}}\ \neg\text{л}}{\text{a:}\neg\neg\text{A} \rightarrow \text{a:A}}\ \supset\text{п}$$
$$\rightarrow \varepsilon\text{:}\neg\neg\text{A} \supset \text{A}$$

Однако не существует такой подстановки σ, при которой секвенция a:$\neg\neg$A, aUb:A $\rightarrow$ a:A стала бы аксиомой. Впрочем, можно применять снизу вверх дальше правила $\neg$л, однако желаемого построения вывода не достичь, напротив, это приведет к зацикливанию процедуры построения вывода. Таким образом, наряду с унификацией, для поиска вывода в данном исчислении требуется процедура проверки циклов, которая еще больше усложняет вычисления.

А. Ваалер вводит различные стратегии поиска вывода в описанном выше интуиционистском исчислении высказываний. Так, он выделяет два способа введения новых термов вывода в правилах $\supset$п и $\neg$п:

1. Антидублирование (duplication-free): в правилах $\supset$п и $\neg$п вводится новый уникальный терм вывода n, т.е. не входящий ни в какую ветвь LKa-выводимости.

2. Супердублирование (duplication-maximal): в правиле $\supset$п или $\neg$п вводится такой терм вывода, который уже вводился другим правилом $\supset$п или $\neg$п в LKa-выводимости, если основные вхождения формул в обоих правилах контекстуально эквивалентны.

Анализируя эти два способа поиска вывода, А. Ваалер отмечает, что:

– супердублирование допускает перестановочность применения правил вывода, т.е. последовательность применений правил вывода не влияет на успех построения вывода, однако формулирование усло-

вия окончания поиска вывода (когда надо закончить анализ вхождений формул – сокращений) является нетривиальным, так как приходится учитывать информацию о префиксах во всем дереве LKa-выводимости, во всех его ветвях,

– антидублирование же требует жесткого порядка применения правил вывода, однако допускается локальное условие окончания применения правил вывода (относительно одной отдельной ветви).

Предлагая далее промежуточное исчисление, А. Ваалер формулирует линейную поисковую процедуру, которая должна строить множества выводимостей для некоторой данной секвенции, что по сути дела является перебором возможных последовательностей применения правил вывода.

Представленные методы оптимизации поиска доказательства отвечают на вопросы формулирования условия окончания поиска вывода (прекращения зацикливания), нахождения оптимальной стратегии поиска доказательства (отказ от перебора), однако они достаточно ограниченно уменьшают сложность поиска доказательства и возникающие исчисления обладают рядом недостатков. Более значительное уменьшение сложности вычислений возможно при синтезе этих методов, что будет сделано в последующих разделах данной главы в рамках создания процедуры поиска натурального вывода для интуиционистской логики высказываний.

3. Поиск доказательств теорем в натуральном исчислении

В данном разделе будет рассмотрена возможность использования синтаксических средств префиксных и секвенциальных исчислений при поиске натурального вывода в интуиционистской логике высказываний. Основное внимание будет сконцентрировано на системе MLJ++ – односукцедентном префиксном секвенциальном интуиционистском исчислении высказываний и системе MNJ+ – натуральном префиксном интуиционистском исчислении. Все же остальные системы носят подготовительный характер к восприятию этих систем. Будет показана также взаимосвязь этих исчислений, которая позволит сформулировать алгоритм поиска вывода в натуральной системе интуиционистской логики высказываний.

Ввиду того, что эта проблематика является центральной, имеет смысл вначале обсудить вопрос о связи рассмотренного в предыдущих главах алгоритма поиска вывода в классической логике высказываний с секвенциальным классическим пропозициональном исчислением. Оказывается, что так как в указанном алгоритме множество целей представляет собой стек, т.е. линейную последовательность формул и знака противоречия, то это позволяет на основе данного алгоритма построить односук-

цедентное классическое секвенциальное исчисление. Это в дальнейшем позволит нам осуществить обратную процедуру – по односукцедентному секвенциальному исчислению разработать алгоритм поиска вывода для натурального исчисления интуиционистской логики.

3.1. *Поиск вывода в классической логике высказываний*

Рассмотрим следующее секвенциальное односукцедентное классическое исчисление высказываний LK+, которое возникло в результате анализа алгоритма поиска доказательства для классической логики в натуральном исчислении, сформулированного в работе В.В. Макарова [1998].

Основные секвенции:

1. $\Gamma, A, \Delta \to A,$
2. $\Gamma\ A, \Theta, \neg A, \Delta \to,$
3. $\Gamma, \neg A, \Theta, A, \Delta \to.$

Схемы логических фигур заключения:

$$\frac{\Gamma, A, B, \Delta \to D}{\Gamma, A\&B, \Delta \to D}\ \&л \qquad \&п\ \frac{\Gamma \to A \mid \Gamma \to B}{\Gamma \to A\&B}$$

$$\frac{\Gamma, \Delta \to \neg A \mid \Gamma, A, \Delta \to}{\Gamma, A \vee B, \Delta \to}\ \vee л \qquad \neg\vee л\ \frac{\Gamma, \Delta, \neg A, \neg B \to}{\Gamma, \neg(A \vee B), \Delta \to}$$

$$\frac{\Gamma, \Delta \to A \mid \Gamma, B, \Delta \to}{\Gamma, A \supset B, \Delta \to}\ \supset л \qquad \supset п\ \frac{\Gamma, A \to B}{\Gamma \to A \supset B}$$

$$\frac{\Gamma, \Delta \to A}{\Gamma, \neg A, A \to,}\ \neg л \qquad \neg п\ \frac{\Gamma, A \to}{\Gamma \to \neg A,}$$

где A не пропозициональная переменная или дизъюнктивная формула

где A – переменная или дизъюнктивная формула

$$\frac{\Gamma, \neg A \to}{\Gamma \to A}\ Aп$$

Здесь A – пропозициональная переменная или дизъюнтивная формула

Понятие вывода является стандартным. Для выявления некоторых свойств данного исчисления введем следующие определения:

Определение 3.1.1. Редукционным деревом для некоторой секвенции $\Gamma \to A$ называется дерево, построенное с применением правил вывода снизу вверх. Последовательность применений левых правил определяется положением формул в антецеденте слева направо. Правило &л имеет приоритет перед всеми остальными правилами.

Лемма 3.1.2. Любой вывод секвенции в LK+ *можно перестроить в вывод, удовлетворяющий условиям редукционного дерева.*

Доказательство основывается на перестановочности правил в LK+. Доказательство же последнего является рутинным и аналогично результату С.К. Клини [1967].

Определение 3.1.3. Для каждой формулы ее ω-степень определяется следующим образом:

$\omega(p) = \omega(\neg p) = 0$, где p – пропозициональная переменная;

для всех остальных формул их ω-степень определяется обычным образом, но с учетом предыдущего пункта.

Для списка формул $B_1,..., B_n$ – $\omega(B_1,..., B_n) = \omega(Bi) + \ldots + \omega(B_n)$, а для секвенции $\Gamma \to C - \omega(\Gamma \to C) = \omega(\Gamma) + \omega(C)$;

Лемма 3.1.4. Для секвенции $\Gamma \to A$ *существует конечное редукционное дерево, каждая ветвь которого оканчивается основной секвенцией или секвенциями вида* $\Pi \to$, *где* Π *– последовательность литералов.*

Доказательство:

Заметим тот факт, что для секвенции вида $\Pi \to$ имеет место $\omega(\Pi \to) = 0$.

Доказательство будет вестись методом индукции по ω-степени секвенции. Пусть $\omega(\Gamma \to A) = 0$, тогда $\omega(\Gamma) = 0$, т.е. Γ состоит из литералов, и $\omega(A) = 0$, т.е. A есть p или $\neg$p. Если $\Gamma \to A$ основная секвенция, то описываемое в лемме *3.2.4.* редукционное дерево построено. Если нет, тогда применяется правило Ап в случае A = p или правило $\neg$п в случае A = $\neg$p, завершающее построение редукционного дерева. Последнее оканчивается секвенцией вида $\Pi \to$.

Пусть для $\omega(\Gamma \to A) \leq n$, где $n > 0$, лемма верна, докажем это для $\omega(\Gamma \to A) = n + 1$. Если $\Gamma \to A$ основная секвенция, то редукционное дерево построено. Если нет, то покажем, что при применении снизу вверх фигур заключения ω-степень секвенций уменьшается. Тем самым на основе индуктивного допущения получим доказательство теоремы.

Для правил &л, &п, $\vee$л, $\neg\vee$л, $\supset$л, $\supset$п, $\neg$л, $\neg$п ω-степень уменьшается (здесь для правила $\neg$п надо рассматривать 2 случая: когда в сукцеденте находится формула $\neg$A и когда находится литерал $\neg$p; в последнем случае ω-степень посылки правила остается равной ω-степени заключения, но последующие применения правил приведут к уменьшению ω-степени верхних секвенций по отношению к нижним).

Рассмотрим правило Ап.

1. Пусть формулой, находящейся в сукцеденте секвенции заключения правила Ап является дизъюнктивная формула вида $C \vee D$.

Тогда комбинация правила Ап с $\neg\vee$л и $\neg$л позволяет показать, что $\omega(\Gamma, \neg C \to D) \leq \omega(\Gamma \to C \vee D)$, а также $\omega(\Gamma, \neg D \to C) \leq \omega(\Gamma \to C \vee D)$. Например,

$$\dfrac{\dfrac{\dfrac{\Gamma, \neg C \to D}{\Gamma, \neg C, \neg D \to}\ \neg\vee\text{л}}{\Gamma, \neg(C \vee D) \to}\ \text{Ап}}{\Gamma \to C \vee D}\ \neg\text{л}$$

Дальнейшее применение правил редукции приведет к тому, что ω-степень верхней секвенции станет $<$, чем ω-степень секвенции $\Gamma \to C \vee D$.

2. Пусть формулой, находящейся в сукцеденте секвенции заключения правила Ап, является пропозициональная формула p. Тогда, так как ω-степень посылки правила Ап остается равной $n+1 > 0$, то в последовательности Γ находятся сложные формулы, к которым возможно применить правило, которое уменьшает ω-степень секвенции. □

Следствие *3.1.5. Исчисление* LK+ *разрешимо*. □

Определение 3.1.6. Интерпретация опровергает секвенцию $\Gamma \to A$, если при данной интерпретации каждая формула из Γ истинна, а A – ложна. Соответственно, интерпретация опровергает секвенцию вида $\Gamma \to$, если при данной интерпретации каждая формула из Γ истинна.

Следуя Г. Такеути [1978], докажем полноту исчисления LK+.

Теорема 3.1.7. (*о полноте*). *Если для некоторой секвенции* $\Gamma \to A$ *не существует доказательства, то существует опровергающая ее интерпретация*.

Доказательство.

Опровергающая интерпретация строится по редукционному дереву секвенции $\Gamma \to A$. Рассмотрим некоторую ветвь этого дерева, которая не оканчивается основной секвенцией. Так как данное редукционное дерево не является доказательством, то такая ветвь существует. Конечная секвенция данной ветви имеет вид $\Pi \to$, где Π – последовательность литералов.

Функцию приписывания значений переменным определим следующим образом:

$\varphi(p) = \boldsymbol{и} \Leftrightarrow p \in \Pi$;
$\varphi(p) = \boldsymbol{л} \Leftrightarrow \neg p \in \Pi$.

Оценка формул определяется стандартным образом.

Докажем, что данная интерпретация является опровергающей для исходной секвенции $\Gamma \to A$. Доказательство ведется методом математической индукции по длине рассматриваемой ветви дерева.

Базис. Интерпретация является опровергающей для конечной секвенции $\Pi \to$ по определению и построению интерпретации.

Пусть для k секвенций данная интерпретация является опровергающей. Докажем это для k+1 секвенции.

1. k+1 секвенция получена по следующему правилу:

$$\frac{\Gamma, A \to B}{\Gamma \to A \supset B.}$$

Формулы из Γ и A по индуктивному предположению являются истинными в данной интерпретации, а B – ложной. Тогда формула $A \supset B$ является ложной, так как $|A \supset B|_\varphi = \boldsymbol{л} \Leftrightarrow |A|_\varphi = \boldsymbol{и} \;\dot{\&}\; |B|_\varphi = \boldsymbol{л}$.

Относительно правил &л, &п, ∨л, ⊃л, ¬л, ¬п доказательство ведется аналогично. Для двупосылочных правил рассматриваются два случая в зависимости от того, какая посылка принадлежит рассматриваемой ветви.

2. k+1 секвенция получена по следующему правилу

$$\frac{\Gamma, \Delta, \neg A, \neg B \to}{\Gamma, \neg(A \vee B), \Delta \to.}$$

Формулы из Γ и Δ, а также ¬A и ¬B по индуктивному предположению являются истинными в данной интерпретации. Тогда формула $\neg(A \vee B)$ является истинной. Действительно, $|\neg(A \vee B)|_\varphi = \boldsymbol{и} \Leftrightarrow |A \vee B|_\varphi = \boldsymbol{л} \Leftrightarrow |A|_\varphi = \boldsymbol{л} \;\dot{\vee}\; |B|_\varphi = \boldsymbol{л} \Leftrightarrow |\neg A|_\varphi = \boldsymbol{и} \;\dot{\&}\; |\neg B|_\varphi = \boldsymbol{и}$. Остальные правила доказываются аналогично. □

Теорема 3.1.8. Исчисление LK+ *непротиворечиво*.

Доказательство заключается в проверке того факта, что опровергающая интерпретация для заключения секвенциальных правил будет опровергающей интерпретацией для посылок правил. □

Лемма 3.1.9. Секвенция $\Gamma, \neg A \to$ *доказуема тогда и только тогда, когда доказуема секвенция* $\Gamma \to A$.

Доказательство. Если секвенция $\Gamma, \neg A \to$ недоказуема, то ее опровергающая интерпретация будет опровергать и секвенцию $\Gamma \to A$, а следовательно, по *теореме 3.2.8.* секвенция $\Gamma \to A$ также будет недоказуемой. Доказательство в обратную сторону аналогично. □

Лемма 3.1.10. Если секвенция $\Gamma \to A$ *доказуема, то доказуема и секвенция* $\Gamma, B \to A$.

Доказательство от противного. Если секвенция Г, В $\to$ А недоказуема, то ее опровергающая интерпретация будет опровергать и секвенцию Г $\to$ А, а следовательно, по *теореме 3.2.8.* секвенция Г $\to$ А также будет недоказуемой. □

В алгоритме для классической логики высказываний используются также обычные правила $\vee$л, но они являются допустимыми в данном исчислении.

Лемма 3.1.11. Следующие правила являются допустимыми:

$$\frac{\Gamma \to A}{\Gamma \to A \vee B}\ \vee л1 \qquad \frac{\Gamma \to B}{\Gamma \to A \vee B}\ \vee л2$$

Покажем это для правила $\vee$л1.

$\Gamma \to A$	*Лемма* 3.1.9.
$\Gamma, \neg A \to$	*Лемма* 3.1.10.
$\Gamma, \neg A, \neg B \to$	$\neg\vee$л
$\Gamma, \neg(A \vee B) \to$	Ап
$\Gamma \to A \vee B$	

Правило $\vee$л2 доказывается аналогично. □

Лемма 3.1.12. Секвенция Г, А, В, $\Delta \to$ D *доказуема тогда и только тогда, когда доказуема секвенция* Г, В, А, $\Delta \to$ D.

Доказательство аналогично доказательству *леммы 3.2.9.* □

Определение 3.1.13. Алгоритмическим выводом называется натуральный вывод, построенный по правилам алгоритма.

Теорема 3.1.14. Пусть неисключенные формулы алгоритмического вывода соответствуют формулам антецедента секвенции, а последняя формула-цель – формуле сукцедента. Тогда для каждой секвенции П $\to$ А *редукционного дерева существует алгоритмический вывод этой формулы* А *из посылок* П, *где* П – *некоторые формулы*.

Доказательство теоремы ведется индукцией по длине доказательства П $\to$ А.

1. Базис: длина вывода $n = 0$.

1.1. Основной секвенции Г, А, $\Delta \to$ А соответствует следующий алгоритмический вывод.

вывод	цели
Г – посылки	А
А – посылка	
Δ – посылки	

Алгоритм находит в выводе формулу A, графически равную цели. Тем самым цель A достижима. Таким образом, получен алгоритмический вывод из Γ, A, Δ формулы A.

1.2. Основной секвенции Γ, ¬A, Θ, A, Δ → соответствует следующий алгоритмический вывод.

вывод	цели
Γ – посылки	⊥
¬A – посылка	
Θ – посылки	
A – посылка	
Δ – посылки	

Алгоритм находит в выводе противоречащие формулы ¬A и A, что означает достижение противоречия. Тем самым получен алгоритмический вывод противоречия из Γ, ¬A, Θ, A, Δ.

1.3. Аналогично для основной секвенции Γ, A, Θ, ¬A, Δ →.

2. По индуктивному допущению предполагается, что для вывода длины *n* теорема верна, т.е. для вывода длины *n* существует алгоритмический вывод. Докажем теперь это для вывода длиной *n* + 1. Пусть на *n* + 1 шаге применялись следующие правила:

$$2.1.\ \frac{\Gamma, A, B, \Delta \to D}{\Gamma, A \,\&\, B, \Delta \to D.}\ \&л$$

Доказательству секвенции Γ, A & B, Δ → D соответствует в алгоритме достижение цели D из Γ, A & B, Δ. Алгоритмически происходит применение правила исключения конъюнкции к формуле A & B и переход к достижению цели D из Γ, A, B, Δ, что соответствует выводу секвенции Γ, A, B, Δ → D. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции Γ, A, B, Δ → D. А значит, имеет место алгоритмический вывод из Γ, A & B, Δ формулы D.

	вывод	цели		вывод	цели
	Γ	D		Γ	D
	A			A & B	
Если	B		то	A	
	…			B	
	D			…	
				D	

$$2.2.\ \frac{\Gamma \to A \mid \Gamma \to B}{\Gamma \to A \,\&\, B}\ \&п$$

Доказательству секвенции Γ → A & B соответствует в алгоритме достижение цели A & B из Γ. Алгоритмически происходит переход к

достижению формулы А из Г. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma \to A$. В последовательности формул вывода появляется формула А, и алгоритм переходит к достижению формулы В из Г, А. Но так как А является логическим следствием из Г, то по сути дела – к достижению формулы В из Г. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma \to B$. Алгоритм вычеркивает формулу-цель В из последовательности формул-целей в силу ее достижимости и последней целью становится А & В. Далее алгоритм на основе существования обоих конъюнктов в последовательности формул вывода применяет правило &в и вводит в вывод формулу А & В. Тем самым цель А & В достигается из Г, А, В. Но так как А и В являются логическим следствием из Г, то по сути дела – к достижению формулы А & В из Г.

вывод	цели
Г – посылки	А & В
…	А – вычеркивается в
А	силу достижимости
…	В – вычеркивается в
В	силу достижимости
А & В	

2.3. $$\frac{\Gamma, A \to B}{\Gamma \to A \supset B} \supset п$$

Доказательству секвенции $\Gamma \to A \supset B$ соответствует в алгоритме достижение цели $A \supset B$ из Г. Алгоритмически происходит переход к достижению формулы В из Г, А, где А принимается как новая посылка. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma, A \to B$. В последовательности формул вывода появляется формула В. Алгоритм вычеркивает цель В из последовательности целей в силу ее достижимости и последней целью становится $A \supset B$. Далее алгоритм применяет правило $\supset$в и вводит в вывод формулу $A \supset B$, исключая все формулы вывода от последней посылки (т.е. А) до вновь полученной формулы $A \supset B$. Тем самым цель $A \supset B$ достигается из Г.

вывод	цели
Г – посылки	$A \supset B$
А – посылка	В – вычеркивается в
…	силу достижимости
В	
$A \supset B$	

$$2.4.\ \frac{\Gamma, \Delta \to \neg A \mid \Gamma, \Delta, B \to}{\Gamma, A \vee B, \Delta \to}\ \vee л$$

Доказательству секвенции $\Gamma, A \vee B, \Delta \to$ соответствует в алгоритме достижение противоречия из $\Gamma, A \vee B, \Delta$. Алгоритмически происходит переход к достижению формулы $\neg A$ из $\Gamma, A \vee B, \Delta$ (анализ *фпц* $A \vee B$), а на формулу $A \vee B$ накладывается метка, запрещающая ее повторный анализ. Это фактически означает, что достижение формулы $\neg A$ осуществляется только из Γ, Δ без использования $A \vee B$. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma, \Delta \to \neg A$. В последовательности формул вывода появляется формула $\neg A$. Происходит применение правила modus tollendo ponens и в выводе появляется формула B. Но так как из Γ, Δ, B по индуктивному предположению алгоритмически выводится противоречие, то тем самым алгоритмически происходит переход к достижению противоречия из $\Gamma, A \vee B, \Delta, \neg A, B$. Так как $\neg A$ и B алгоритмически следуют из $\Gamma, A \vee B, \Delta$, то тем самым противоречие следует из $\Gamma, A \vee B, \Delta$.

вывод	цели
Γ – посылки	⊥
$A \vee B$	$\neg A$ – новая цель
Δ – посылки	
…	
$\neg A$	
B	

$$2.5.\ \frac{\Gamma, \Delta, \neg A, \neg B \to}{\Gamma, \neg(A \vee B), \Delta \to}\ \neg\vee л$$

Доказательству секвенции $\Gamma, \neg(A \vee B), \Delta \to$ соответствует в алгоритме достижение противоречия из $\Gamma, \neg(A \vee B), \Delta$. Алгоритмически происходит переход к достижению целей $\neg A$ и $\neg B$ из $\Gamma, \neg(A \vee B), \Delta$. Алгоритм в данном случае работает независимо от структуры формул A и B и всегда достигает цели $\neg A$ и $\neg B$. В последовательности формул вывода появляются формулы $\neg A$ и $\neg B$.

По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma, \Delta, \neg A, \neg B \to$. Но так как $\neg A$ и $\neg B$ являются логическим следствием из $\Gamma, \neg(A \vee B), \Delta$, то существует алгоритмический вывод противоречия из $\Gamma, \neg(A \vee B), \Delta$.

вывод	цели
Γ – посылки	⊥
$\neg(A \vee B)$ – посылка	$\neg A$ – новая цель

Δ – посылки	…
…	…
¬A	¬B – новая цель
…	
¬B	

2.6. $\dfrac{\Gamma, \Delta \to A \mid \Gamma, \Delta, B \to}{\Gamma, A \supset B, \Delta \to}$ ⊃л

Доказательству секвенции Γ, A ⊃ B, Δ → соответствует в алгоритме достижение противоречия из Γ, A ⊃ B, Δ.

Алгоритмически происходит переход к достижению формулы A из Γ, A ⊃ B, Δ (анализ формулы вывода A ⊃ B), а на формулу A ⊃ B накладывается метка, запрещающая ее повторный анализ. Это фактически означает достижение формулы A только из Γ, Δ без использования формулы A ⊃ B.

По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции Γ, Δ → A. В последовательности формул вывода появляется формула A, цель A вычеркивается, применяется аналитическое правило ⊃и, и последовательность формул вывода пополняется формулой B. Алгоритм переходит к достижению противоречия из Γ, A ⊃ B, Δ, A, B. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции Γ, Δ, B →. Тем самым противоречие достигается из Γ, A ⊃ B, Δ, A, B. Но так как A и B являются логическим следствием из Γ, A ⊃ B, Δ , то существует алгоритмический вывод противоречия из Γ, A ⊃ B, Δ.

вывод	цели
Γ – посылки	⊥
A ⊃ B – посылка	A – новая цель
Δ – посылки	
…	
A	
B – по правилу ⊃в	

2.7. Доказательства для правил с отрицанием ведутся аналогичным способом □.

3.2. *Поиск вывода в интуиционистской логике высказываний*

Представленное исчисление LK+ отражает достаточно точным образом работу алгоритма поиска доказательства для классической логики высказываний. В силу того, что LK+ оказалось свободным от структурных правил, весьма удобно перестроить вывод в этом исчислении в натуральный вывод. В дальнейшем

строится аналогичное секвенциальное исчисление без структурных правил для интуиционистской логики высказываний.

Вначале приведем многосукцедентный вариант MLJ_{cf} исчисления Р. Дикхофа (Дикхоф [1992]), которое будет перестраиваться в более удобное для наших целей секвенциальное исчисление.

Основные секвенции:

1. $\Gamma, A, \Delta \to \Sigma, A, \Theta,$
2. $\Gamma, f, \Delta \to \Theta.$

Правила вывода MLJ_{cf}:

$$\frac{\Gamma, A, B, \Delta \to \Theta}{\Gamma, A \,\&\, B, \Delta \to \Theta}\ \&л \qquad \&п\ \frac{\Gamma \to \Delta, A, \Theta \mid \Gamma \to \Delta, B, \Theta}{\Gamma \to \Delta, A \,\&\, B, \Theta}$$

$$\frac{\Gamma, A, \Delta \to \Theta \mid \Gamma, B, \Delta \to \Theta}{\Gamma, A \vee B, \Delta \to \Theta}\ \vee л \qquad \vee п\ \frac{\Gamma \to \Delta, A, B, \Theta}{\Gamma \to \Delta, A \vee B, \Theta}$$

$$\frac{\Gamma, (A_1 \supset B) \,\&\, (A_2 \supset B), \Delta \to \Theta}{\Gamma, (A_1 \vee A_2) \supset B, \Delta \to \Theta}\ \vee\supset л \qquad \supset п\ \frac{\Gamma, A \to B}{\Gamma, \to \Delta, A \supset B, \Theta}$$

$$\frac{\Gamma, A_1 \supset (A_2 \supset B), \Delta \to \Theta}{\Gamma, (A_1 \,\&\, A_2) \supset B, \Delta \to \Theta}\ \&\supset л \qquad p\supset л\ \frac{\Gamma, p, B, \Delta \to \Theta}{\Gamma, p, p \supset B, \Delta \to \Theta}$$

$$\frac{\Gamma, \Delta \to \neg A_1, \Theta \quad \Gamma, B, \Delta \to \Theta}{\Gamma, \neg A_1 \supset B, \Delta \to \Theta}\ \neg\supset л \qquad \supset\supset л\ \frac{\Gamma, A_2 \supset B, \Delta \to A_1 \supset A_2, \Theta \quad \Gamma, B, \Delta \to \Theta}{\Gamma, (A_1 \supset A_2) \supset B, \Delta \to \Theta}$$

Связка "$\neg$" вводится как сокращение, т.е. $\neg A = A \supset f$. Особенностью этого исчисления является правило $\supset$п, в котором происходит вычеркивание формул сукцедента секвенции при переходе от заключения к посылке правила.

Доказываются теоремы о полноте и непротиворечивости этого исчисления.

Далее для перестройки данного исчисления в систему с префиксами введем новые объекты языка, следуя терминологии А. Ваалера:

– a, b, c,... – параметры вывода.

Термом вывода считается параметр вывода. Префиксом является последовательность термов вывода, в том числе и пустая последовательность ε. Причем, отождествляются следующие префиксы b, εb и bε, т.е. $b = \varepsilon b = b\varepsilon$.

Пусть запись "α:X" (где α – префикс, X – формула) означает утверждение формулы X относительно префикса α.

На префиксах определяется следующее отношение "$\leq$":

(a) $\alpha \leq \alpha$ для любого префикса α;

(b) если $\alpha \leq \beta$, где αиβ– префиксы, то $\alpha \leq \beta\gamma$, где γ – некоторая последовательность термов вывода.

Вводятся следующие обозначения:

1. $\max(\alpha,\beta)=\begin{cases}\beta, \text{если } \alpha\le\beta;\\ \alpha, \text{если } \beta<\alpha.\end{cases}$

2. $\neg A = A\supset f$.

Основные секвенции:

1. $\Gamma, \alpha{:}A, \Delta\to\Sigma, \beta{:}A, \Theta$, если $\alpha\le\beta$,
2. $\Gamma, a{:}f, \Delta\to\Theta$.

Схемы заключений исчисления MLJ:

$$\frac{\Gamma, \alpha{:}A, \alpha{:}B, \Delta\to\Theta}{\Gamma, \alpha{:}A\ \&\ B, \Delta\to\Theta}\ \&л \qquad \&п\ \frac{\Gamma\to\Delta, \alpha{:}A, \Theta \mid \Gamma\to\Delta, \alpha{:}B, \Theta}{\Gamma\to\Delta, \alpha{:}A\ \&\ B, \Theta}$$

$$\frac{\Gamma, \alpha{:}A, \Delta\to\Theta \mid \Gamma, \alpha{:}B, \Delta\to\Theta}{\Gamma, \alpha{:}A\vee B, \Delta\to\Theta}\ \vee л \qquad \vee п\ \frac{\Gamma\to\Delta, \alpha{:}A, \alpha{:}B, \Theta}{\Gamma\to\Delta, \alpha{:}A\vee B, \Theta}$$

$$\frac{\Gamma, \Delta\to\alpha'{:}\neg A, \Theta \mid \Gamma, \alpha'{:}B, \Delta\to\Theta}{\Gamma, \alpha{:}\neg A\supset B, \Delta\to\Theta,}\ \neg\supset л \qquad \supset п\ \frac{\Gamma, \alpha b{:}A\to\alpha b{:}B}{\Gamma\to\Delta, \alpha{:}A\supset B, \Theta,}$$

где α' удовлетворяет условию $\dot\forall\,\beta(\beta{:}X\in\{\Gamma, \alpha{:}\neg A\supset B, \Delta\}\Rightarrow\beta\le\alpha')$ и $\dot\exists\, X(\alpha'{:}X\in\{\Gamma, \alpha{:}\neg A\supset B, \Delta\})$

где b не встречается в заключении правила

$$\frac{\Gamma, \alpha{:}(A_1\supset B)\ \&\ (A_2\supset B), \Delta\to\Theta}{\Gamma, \alpha{:}(A_1\vee A_2)\supset B), \Delta\to\Theta}\ \vee\supset л \qquad \&\supset л\ \frac{\Gamma, \alpha{:}A_1\supset(A_2\supset B), \Delta\to\Theta}{\Gamma, \alpha{:}(A_1\ \&\ A_2)\supset B, \Delta\to\Theta}$$

$$\frac{\Gamma, \alpha{:}A_2\supset B, \Delta\to\alpha'{:}(A_1\supset A_2), \Theta \mid \Gamma, \alpha'{:}B, \Delta\to\Theta}{\Gamma, \alpha{:}(A_1\supset A_2)\supset B, \Delta\to\Theta,}\ \supset\supset л \qquad p\supset л\ \frac{\Gamma, \beta{:}p, \Delta, \max(\alpha,\beta){:}\ A_2\to\Theta}{\Gamma, \alpha{:}\,(p\supset A_2), \beta{:}\,p, \Delta\to\Theta}$$

где α' удовлетворяет условию $\dot\forall\,\beta(\beta{:}X\in\{\Gamma, \alpha{:}(A_1\supset A_2)\supset B, \Delta\}\Rightarrow\beta\le\alpha')$ и $\dot\exists\, X(\alpha'{:}X\in\{\Gamma, \alpha{:}(A_1\supset A_2)\supset B, \Delta\})$

Доказательством формулы A является дерево секвенций с конечной секвенцией (корнем) $\to\varepsilon{:}A$, каждая секвенция или является аксиомой, или получена по одной из схем заключений исчисления MLJ.

Относительно данного исчисления верна следующая

Лемма 3.2.1. Для любой секвенции MLJ*-выводимости множество содержащихся в ней префиксов обладает свойством линейности*: $\forall\alpha\forall\beta(\alpha\le\beta\vee\beta\le\alpha)$.

Доказательство: Ветвление множества префиксов может наступить при применении снизу вверх правил, в которых появляются новые префиксы, т.е. в правилах $\supset$п, $\neg\supset$л и $\supset\supset$л. Однако эти префиксы являются непосредственно следующими максимальному префиксу секвенции, как в правиле $\supset$п, или самим максимальным префиксом, как в правилах $\neg\supset$л и $\supset\supset$л. □

Таким образом, в правиле p$\supset$л не требуется условия сравнимости префиксов α и β, так как любые префиксы в выводе сравнимы.

Теорема 3.2.2. Пусть π *– выводимость в исчислении* MLJ_{cf} *некоторой секвенции* $\Gamma \to \Delta$, *тогда существует* δ – MLJ-*выводимость с той же структурой, что и* π, *и с конечной секвенцией* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, *такой, что если* π *– доказательство* $\Gamma \to \Delta$, *то* δ *– доказательство* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$.

Доказательство:

Тот факт, что для каждой выводимости в исчислении MLJ_{cf} некоторой секвенции $\Gamma \to \Delta$ существует MLJ-выводимость с той же структурой и с конечной секвенцией $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, является очевидным и не требует доказательства. Докажем лишь индукцией по длине выводимости, что если π – доказательство $\Gamma \to \Delta$, то δ – доказательство $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$.

Базис. Пусть π – выводимость, состоящая из одной секвенции $\Gamma \to A$, тогда δ представляет собой $\varepsilon{:}\Gamma \to \varepsilon{:}A$. Если π – доказательство $\Gamma \to \Delta$, то $\Gamma \to \Delta$ – основная секвенция и имеет вид $\Gamma_1, A, \Gamma_2 \to \Delta_1, A, \Delta_2$. Тогда δ имеет вид $\varepsilon{:}\Gamma_1, \varepsilon{:}A, \varepsilon{:}\Gamma_2 \to \varepsilon{:}\Delta_1, \varepsilon{:}A, \varepsilon{:}\Delta_2$ и также является доказательством. Аналогично для второго случая.

Пусть r – верхнее правило π и π' – выводимость, получающаяся отбрасыванием этого правила, т.е. заключение r есть верхняя секвенция в π'. Пусть δ' – MLJ-выводимость с такой же структурой, что и π' и r являлось правилом

$$\frac{\Gamma, A_2 \supset B, \Delta \to (A_1 \supset A_2), \Theta \;\big|\; \Gamma, B, \Delta \to \Theta}{\Gamma, (A_1 \supset A_2) \supset B, \Delta \to \Theta} \;\supset\supset\text{л}$$

Тогда верхней секвенцией δ' является секвенция вида Γ, $a{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta$. Добавляем к δ' правило $\supset\supset$л в MLJ и получаем выводимость δ

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha'{:}(A_1 \supset A_2), \Theta \;\big|\; \Gamma, \alpha'{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta} \;\supset\supset\text{л}$$

Если π являлось доказательством конечной секвенции, то $\Gamma, A_2 \supset B, \Delta \to A_1 \supset A_2, \Theta$ и $\Gamma, B, \Delta \to \Theta$ являются аксиомами MLJ_{cf}. Тогда аксиомами MLJ являются и секвенции $\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha'{:}(A_1 \supset A_2), \Theta$ и $\Gamma, \alpha'{:}B, \Delta \to \Theta$, т.е. δ является доказательством в силу того, что в MLJ-выводе любой префикс β формулы сукцедента секвенции больше или равен любого префикса α антецедента той же секвенции, т.е. $\alpha \le \beta$.

Аналогично доказываются случаи относительно других правил вывода. □

Как следствие только что доказанного утверждения, имеет место

Теорема 3.2.3. (*о полноте* MLJ): *для любой интуиционистски общезначимой формулы* A *имеется* MLJ*-доказательство секвенции* $\rightarrow \varepsilon$:A. □

Теорема 3.2.4. (*о непротиворечивости* MLJ): *исчисление* MLJ *непротиворечиво, т.е. если секвенция опровержима, то она не доказуема в* MLJ.

Доказательство:

Предполагается, что существует некая опровергающая интерпретация для секвенции $\Gamma \rightarrow \Delta$. Докажем, что при построении для нее MLJ-выводимости данная интерпретация будет опровергающей для верхних секвенций правил вывода (в случае с двупосылочными правилами хотя бы одной из верхних секвенций).

1. Рассмотрим следующее правило:

$$\frac{\Gamma, \alpha b{:}A \rightarrow \alpha b{:}B}{\Gamma \rightarrow \Delta, \alpha{:}A \supset B, \Theta} \supset п$$

Формулы из Γ по индуктивному предположению являются истинными в данной интерпретации. Докажем, что TA/αb и FB/αb.

	F(A ⊃ B)/αпосылка
	$\dot{\exists}\alpha'$(Rαα′ и TA/α′ и
FB/α′)	из 1 по определению
	Rα(αb)из 2
	TA/αbиз 2
	FB/αbиз 2

2. Рассмотрим следующее правило:

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \rightarrow \alpha'{:}(A_1 \supset A_2), \Theta \;\big|\; \Gamma, \alpha'{:}B, \Delta \rightarrow \Theta}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \rightarrow \Theta} \supset\supset л$$

Формулы из Γ, Δ, Θ по индуктивному предположению не меняют своего истинностного значения в данной интерпретации. Пусть в данной интерпретации TB/α′, тогда эта интерпретация является опровергающей для секвенции Γ, α′:B, Δ → Θ. Тогда имеет место FB/α′. Докажем, что при этом условии $T(A_2 \supset B)/\alpha$ и $F(A_1 \supset A_2)/\alpha'$.

$T((A_1 \supset A_2) \supset B)/\alpha$	посылка
$FB/\alpha', \alpha \leq \alpha'$	посылка
$F(A_2 \supset B)/\alpha$ или $T(A_1 \supset A_2)/\alpha'$	допущение
$F(A_2 \supset B)/\alpha$	допущение
$\dot{\exists}\beta$(Rαβ и (TA_2/β и FB/β))	из 4
Rαβ и (TA_2/β и FB/β))	из 5

$R\alpha\beta$	из 6, &и
TA_2/β	из 6, &и
FB/β	из 6, &и
$\dot{\forall}\,\beta(R\alpha\beta \Rightarrow (T(A_1 \supset A_2)/\beta \Rightarrow TB/\beta))$	из 1
$R\alpha\beta \Rightarrow (T(A_1 \supset A_2)/\beta \Rightarrow TB/\beta)$	из 10
$T(A_1 \supset A_2)/\beta \Rightarrow TB/\beta$	из 11, 7, m.p.
$F(A_1 \supset A_2)/\beta$	из 12, 9, контрапозиция
$\dot{\exists}\,\beta'(R\beta\beta'$ и $(TA_1/\beta'$ и $FA_2/\beta'))$	из 13
$R\beta\beta'$ и $(TA_1/\beta'$ и $FA_2/\beta')$	из 14
$R\beta\beta'$	из 15, &и
	FA_2/β' из 15, &и
	$\dot{\forall}\,\beta\,\dot{\forall}\,\beta'\,\dot{\forall}$ A($R\beta\beta'$
$\Rightarrow (TA/\beta \Rightarrow TA/\beta'))$	теорема
	TA_2/β' из 18, 16, 8

Значит,

$T(A_2 \supset B)/\alpha$	из 19, 17
$T(A_1 \supset A_2)/\alpha'$	из 3, 20
$\dot{\forall}\,\beta(R\alpha\beta \Rightarrow (T(A_1 \supset A_2)/\beta \Rightarrow TB/\beta))$	из 1
$R\alpha\alpha' \Rightarrow (T(A_1 \supset A_2)/\alpha' \Rightarrow TB/\alpha')$	из 22
$R\alpha\alpha'$	из 2 в силу усл.: $\alpha \leq \alpha'$
$T(A_1 \supset A_2)/\alpha' \Rightarrow TB/\alpha')$	из 23, 24, m. p.
$F(A_1 \supset A_2)/\alpha'$	из 25, 2, контрапозиция

Значит,

27. $T(A_2 \supset B)/\alpha$ и $F(A_1 \supset A_2)/\alpha'$	из 21, 26

Аналогично доказываются остальные правила. □

Доказать непротиворечивость MLJ возможно еще короче, основываясь на том факте, что для каждого доказательства в MLJ существует с той же структурой доказательство в MLJ_{cf}. Это достигается путем отбрасывания префиксов в MLJ-доказательстве. Таким образом, возможно показать 1-1 соответствие исчисления MLJ и MLJ_{cf}.

Продолжим дальнейшую перестройку исчисления и будем для формирования префиксов использовать метапеременные, которые служат обозначением некоторого кортежа термов вывода. Другими словами, поиск вывода в следующем исчислении будет разбиваться на два процесса. Первый процесс будет направлен на создание заготовки вывода с использованием метапеременных в префиксах формул. Второй процесс будет состоять в превращении заготовки в полноценный вывод, если он существует, путем замены метапеременных на определенные кортежи термов вывода.

Расширяя понятие терма вывода, введем в язык переменные по префиксам:

– U, V, W,... – переменные вывода.

Таким образом, определение терма вывода становится идентичным определению, данному А. Ваалером в [2000], т.е. термом вывода считается или параметр вывода или переменная вывода. Понятия метки и префикса остаются прежними.

Далее введем следующие определения, следуя А. Ваалеру.

Если некоторый терм вывода t входит в метку q, то p является *префиксом терма t в q*, если $pt \le q$. Множество меток Θ удовлетворяет условию единственного префикса, если для любого терма вывода существует единственный префикс во всех метках из Θ.

Определяется подстановка σ меток как функция из множества переменных вывода во множество меток. Запись $Z\sigma$ означает результат применения подстановки σ к некоторому объекту Z. Подстановка σ называется идемпотентной, если $Z\sigma\sigma = Z\sigma$ для любого Z. Подстановка σ считается связанной множеством меток Θ, если выполняется следующее условие: если t является последним термом в $U\sigma$, то существуют такие метки αU и βt в Θ, что $(\alpha U)\sigma = (\beta t)\sigma$.

Имеет место следующий факт:

Пусть Θ выполняет условие единого префикса и σ является идемпотентной подстановкой, связанной этим множеством Θ, тогда $\Theta\sigma$ тоже будет выполнять условие единого префикса.

Теперь сформулируем исчисление MLJ+, в котором используются переменные по префиксам, т.е. заново определенные термы вывода.

Список основных секвенций остается неизменным от исчисления MLJ.

Правила исчисления MLJ+:

$$\frac{\Gamma, \alpha{:}A, \alpha{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}A \,\&\, B, \Delta \to \Theta}\ \&л \qquad \&п\ \frac{\Gamma \to \Delta, \alpha{:}A, \Theta \mid \Gamma \to \Delta, \alpha{:}B, \Theta}{\Gamma \to \Delta, \alpha{:}A \,\&\, B, \Theta}$$

$$\frac{\Gamma, \alpha{:}A, \Delta \to \Theta \mid \Gamma, \alpha{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}A \vee B, \Delta \to \Theta}\ \vee л \qquad \vee п\ \frac{\Gamma \to \Delta, \alpha{:}A, \alpha{:}B, \Theta}{\Gamma \to \Delta, \alpha{:}A \vee B, \Theta}$$

$$\frac{\Gamma, \Delta \to \alpha U{:}\neg A, \Theta \mid \Gamma, \alpha U{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}\neg A \supset B, \Delta \to \Theta,}\ \neg \supset л \qquad \supset п\ \frac{\Gamma, \alpha b{:}A \to \Delta, \alpha b{:}B, \Theta}{\Gamma \to \Delta, \alpha{:}A \supset B, \Theta,}$$

где U не встречается в заключении

где b не встречается в заключении

$$\frac{\Gamma, \alpha{:}(A_1 \supset B)\ \&\ (A_2 \supset B), \Delta \to \Theta}{\Gamma, \alpha{:}((A_1 \vee A_2) \supset B), \Delta \to \Theta}\ \vee\supset\text{л} \qquad \&\supset\text{л}\ \frac{\Gamma, \alpha{:}A_1 \supset (A_2 \supset B), \Delta \to \Theta}{\Gamma, \alpha{:}(A_1\ \&\ A_2) \supset B, \Delta \to \Theta}$$

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha U{:}(A_1 \supset A_2), \Theta \quad | \quad \Gamma, \alpha U{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta,}\ \supset\supset\text{л}$$

где U не встречается в заключении

$$\frac{\Gamma, \beta{:}p, \Delta \to \alpha U{:}p, \Theta \qquad \Gamma, \beta{:}p, \alpha U{:}A_2 \to \Theta}{\Gamma, \alpha{:}(p \supset A_2), \beta{:}p, \Delta \to \Theta,}\ p\supset\text{л}$$

где $\alpha \le \beta$ или $\beta \le \alpha$, U не встречается в заключении

Вхождения формулы $\alpha{:}A_2 \supset B$ в правило $\supset\supset$л называется псевдосокращением. Вхождения формул $\alpha{:}(A_1 \supset B)\ \&\ (A_2 \supset B)$, $\alpha{:}A_1 \supset (A_2 \supset B)$, $\alpha U{:}p$ и $\alpha U{:}A_2$ в правила $\vee\supset$л, $\&\supset$л и $p\supset$п, соответственно, считаются боковыми вхождениями. Боковые (левое и правое), основные и внешние вхождения формул относительно остальных случаев определяются обычным образом (Генцен [1967]).

Следуя А. Ваалеру, вводится отношение контекстуальной эквивалентности на вхождениях формул в MLJ+-выводимости.

Условие MLJ+*-вывода*: если в правилах одного типа при переходе от заключения к посылкам вводится один и тот же терм вывода t, то основные вхождения формул соответствующих правил являются контекстуально эквивалентными, а боковые вхождения формул обладают одинаковыми префиксами.

δ есть σ-выводимость, если σ идемпотентно и σ связано относительно множества префиксов δ. Аксиомой считается одна из основных секвенций. Секвенция S есть σ-аксиома, если $S\sigma$ является аксиомой. Конечная σ-выводимость δ с самой нижней секвенцией S является доказательством S, если все листья $\delta\sigma$ (т.е. самые верхние секвенции) являются аксиомами.

Другими словами, вначале строится макет вывода – дерево секвенций, каждая их которых получена по одной из схем заключений исчисления MLJ+. Затем мы получаем вывод через применение унификации к макету вывода, префиксы формул которого лишаются переменных вывода (переменных по префиксам), а листья (верхние секвенции) становятся основными секвенциями. Формула A считается доказуемой, если существует вывод секвенции $\to \varepsilon{:}A$.

Теорема 3.2.5. Пусть π *– выводимость в исчислении* MLJ *некоторой секвенции* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, *тогда существуют* δ – MLJ+*-выводимость с той же структурой, что и* π, *с конечной секвенцией* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, *и подстановка* σ *такова, что если* π *доказательство* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, *то* $\delta\sigma$*-доказательство* $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$.

Доказательство:

Строя индуктивно δ и σ из π, докажем, что $\delta\sigma$ удовлетворяет следующим двум свойствам:

(i) для каждого антецедентного вхождения α:A в π существует соответствующее ему антецедентное вхождение β:A в $\delta\sigma$ такое, что $\beta \leq \alpha$.

(ii) для каждого сукцедентного вхождения α:A в π существует соответствующее ему сукцедентное вхождение α:A в $\delta\sigma$.

Теорема будет непосредственно следовать из этого.

Доказательство ведется индукцией по длине MLJ-вывода.

Базис. Пусть π – выводимость, состоящая из одной только конечной секвенции $\varepsilon{:}\Gamma \to \varepsilon{:}\Delta$, тогда $\delta = \pi$ и $\sigma = 0$.

Пусть r – верхнее правило π и π' – выводимость, получающаяся отбрасыванием этого правила, т.е. заключение r есть верхняя секвенция в π'. По индуктивному допущению существуют δ' и σ' такие, что $\delta'\sigma'$ удовлетворяют условиям (i) и (ii) относительно π'. Пусть r будет правилом

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha'{:}(A_1 \supset A_2), \Theta \quad | \quad \Gamma, \alpha'{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta} \supset\supset\text{л}$$

Тогда верхней секвенцией π' является секвенция вида $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta$. Пусть тогда верхней секвенцией δ' является $\Gamma', \alpha{:}(A_1 \supset A_2) \supset B, \Delta' \to \Theta'$. По (i) $\beta\sigma' \leq \alpha$. Добавляем к δ' следующее правило:

$$\frac{\Gamma', \beta{:}A_2 \supset B, \Delta' \to \beta U{:}(A_1 \supset A_2), \Theta' \quad | \quad \Gamma', \beta U{:}B, \Delta' \to \Theta'}{\Gamma', \beta{:}(A_1 \supset A_2) \supset B, \Delta' \to \Theta',} \supset\supset\text{л}$$

где U – новая переменная вывода. Так как $\beta\sigma' \leq \alpha$ и $\alpha \leq \alpha'$, то $\beta\sigma' \leq \alpha'$. Тогда должна существовать метка γ такая, что $\beta\sigma'\gamma = \alpha'$. Пусть $\sigma = \sigma' \cup \{U/\gamma\}$ и тогда $\delta\sigma$ удовлетворяет условиям (i) и (ii).

Аналогично доказываются случаи относительно других правил вывода. □

Как следствие только что доказанного утверждения имеет место:

Теорема 3.2.6. (*о полноте* MLJ+): *для любой интуиционистски общезначимой формулы* A *имеется* MLJ+-*доказательство секвенции* $\to \varepsilon{:}A$. □

Определение 3.2.7. Интерпретация при некоторой подстановке σ опровергает секвенцию $\Gamma \to A$, если при данной интерпретации каждая формула из Γ истинна, а A ложна. Соответственно, интерпретация при некоторой подстановке σ опровергает секвенцию вида $\Gamma \to$, если при данной интерпретации каждая формула из Γ истинна.

Теорема 3.2.8. (*о непротиворечивости* MLJ+). *Исчисление непротиворечиво, т.е. если секвенция опровержима, то она не доказуема.*

Доказательство:

Предполагается, что существует некая опровергающая интерпретация для секвенции $\Gamma \to A$. Докажем, что при построении для нее MLJ+-выводимости данная интерпретация будет опровергающей для верхних секвенций правил вывода (в случае с двупосылочными правилами хотя бы одной из верхних секвенций).

1. Рассмотрим следующее правило:

$$\frac{\Gamma, \alpha b{:}A \to \Delta, \alpha b{:}B, \Theta}{\Gamma \to \Delta, \alpha{:}A \supset B, \Theta.} \quad \supset\text{п}$$

Формулы из Γ, Δ, Θ по индуктивному предположению не меняют своего истинностного значения в данной интерпретации. Доказательство того, что $TA/\alpha b$ и $FB/\alpha b$, полностью совпадает с доказательством аналогичного случая при обосновании непротиворечивости системы MLJ.

2. Рассмотрим следующее правило:

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha U{:}(A_1 \supset A_2), \Theta \;\big|\; \Gamma, \alpha U{:}B, \Delta \to \Theta}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to \Theta.} \quad \supset\supset\text{л}$$

Формулы из Γ, Δ, Θ по индуктивному предположению не меняют своего истинностного значения в данной интерпретации. Интересен случай, когда $FB/\alpha U$. Доказательство того, что при этом условии $T(A_2 \supset B)/\alpha$ и $F(A_1 \supset A_2)/\alpha U$, совпадает (с заменой везде α' на αU) с доказательством аналогичного случая при обосновании непротиворечивости системы MLJ.

Аналогично доказываются остальные правила. □

Лемма 3.2.9. Следующие правила являются допустимыми в исчислении MLJ+:

$$\frac{\Gamma \to \Theta, \alpha{:}A \;\big|\; \Gamma, \alpha{:}A \to \Theta}{\Gamma \to \Theta} \;\text{Cut1} \qquad \frac{\Gamma, \alpha b{:}A, \Delta \to \Theta}{\Gamma, \alpha{:}A, \Delta \to \Theta} \;\text{LT}$$

$$\frac{\Gamma \to \Theta, \alpha{:}A \;\big|\; \Gamma \to \Theta, \alpha{:}\neg A}{\Gamma \to \Theta} \;\text{Cut2} \qquad \frac{\Gamma \to \Delta, \alpha{:}A, \Theta}{\Gamma \to \Delta, \alpha b{:}A, \Theta} \;\text{RT}$$

Доказательство:

Предполагается, что существует некая опровергающая интерпретация для нижней секвенции этих правил. Докажем, что данная интерпретация бу-

дет опровергающей для верхних секвенций (в случае с двупосылочными правилами хотя бы одной из верхних секвенций).

Правило Cut2:

$$\frac{\Gamma \to \Theta,\ \alpha{:}A \quad \big|\quad \Gamma \to \Theta,\ \alpha{:}\neg A}{\Gamma \to \Theta} \quad \text{Cut2}$$

Если в данной интерпретации имеет место FA/α, то она опровергает секвенцию $\Gamma \to \Theta,\ \alpha{:}A$. Тогда в ней должно быть TA/α. Пусть в данной интерпретации не опровергается секвенция $\Gamma \to \Theta,\ \alpha{:}\neg A$, тогда $T\neg A/\alpha$.

TA/α	посылка
$T\neg A/\alpha$	посылка
$\dot{\forall}\,\beta(R\alpha\beta \Rightarrow FA/\beta)$	из 2
FA/α	из 3

Противоречие 1 и 4.

Таким образом, или опровергается секвенция $\Gamma \to \Theta,\ \alpha{:}A$ или секвенция $\Gamma \to \Theta,\ \alpha{:}\neg A$.

Допустимость правила Cutl доказывается аналогично.

Правило LT:

$$\frac{\Gamma,\ \alpha b{:}A,\ \Delta \to \Theta}{\Gamma, \alpha{:}A, \Delta \to \Theta} \quad \text{LT}$$

$T\,A/\alpha$	посылка
$R\alpha(\alpha b)$	так как $\alpha \le \alpha b$
$\dot{\forall}\,\beta(R\alpha\beta \Rightarrow (TA/\alpha \Rightarrow TA/\beta))$	теорема
$R\alpha(\alpha b) \Rightarrow (TA/\alpha \Rightarrow TA/\alpha b)$	из 3
$TA/\alpha b$	m.p., из 4, 2, 1

Таким образом, опровергается и верхняя секвенция. Допустимость правила RT доказывается аналогично. □

Для исчисления MLJ+ (на основании теоремы о полноте MLJ+) можно использовать комбинацию таких стратегий поиска вывода, как антидублирование и супердублирование.

1. Антидублирование применяется относительно переменных вывода: в правилах $\supset\supset$л и $\neg\supset$л вводится новый уникальный терм вывода U, т.е. не входящий ни в какую ветвь MLJ+-выводимости.

2. Супердублирование применяется относительно параметров вывода: в правиле $\supset$п вводится такой терм вывода, который

уже вводился другим применением правила ⊃п в MLJ+-выводимости, если основные вхождения формул в этих двух применениях правила контекстуально эквивалентны.

Таким образом, через систему правил Р. Дикхофа снимается проблема избыточного порождения вхождений формул в секвенции и условия окончания процедуры поиска вывода (что было характерно для стратегии супердублирования А. Ваалера), а проблема перебора последовательностей применений правил вывода (что было характерно для стратегии антидублирования того же А. Ваалера) заменяется задачей поиска подходящей подстановки, т.е. задачей унификации.

Однако, не особенно усложняя процедуру поиска подходящей подстановки, можно отказаться от стратегии супердублирования и последовательно использовать идею антидублирования в построении доказательства.

Проблема построения алгоритма унификации, необходимого для данного исчисления, рассматривалась Дж. Оттеном и К. Крайтцем в [1995b], [2000]. Унификационная проблема для MLJ+ идентична проблеме, рассматриваемой этими авторами. Разница состоит лишь в использовании отношения "≤" вместо "=", относительно которого происходит унификация. Поэтому возможно применить разработанный ими алгоритм к исчислению MLJ+.

Для того чтобы сделать данное исчисление пригодным для реализации в натуральном выводе (как это было показано выше относительно классической логики высказываний), следует перестроить данное исчисление в односукцедентное исчисление MLJ++.

Для этого введем в язык новую связку "~" и новый тип правильно построенного выражения – "~-структуры".

Определение 3.2.10. ~-структура исчисления MLJ++ определяется индуктивным способом:

1. Любая формула А исчисления MLJ+ является ~-структурой исчисления MLJ++.

2. Если А является ~-структурой исчисления MLJ++, то выражение ~А также является ~-структурой исчисления MLJ++.

3. Ничто иное не является ~-структурой исчисления MLJ++.

Семантически связка "~" понимается следующим образом:

$T{\sim}A/\alpha \Leftrightarrow FA/\alpha$,
$F{\sim}A/\alpha \Leftrightarrow TA/\alpha$.

Соответственно изменяется понятие секвенции. Под секвенцией исчисления MLJ++ понимается выражение вида $\Gamma \to \alpha{:}A$, где Γ – последовательность $\sim$-структур исчисления MLJ++ с префиксами, а $\alpha{:}A$ – $\sim$-структура A исчисления MLJ++ с префиксом α.

Основными секвенциями исчисления MLJ++ будут считаться следующие:

$\Gamma, \beta{:}A, \Theta, \alpha{:}\sim A, \Delta \to$, если $\beta \le \alpha$.
$\Gamma, \alpha{:}\sim A, \Theta, \beta{:}A, \Delta \to$, если $\beta \le \alpha$.
$\Gamma, \alpha{:}f, \Delta \to$.

Схемы заключений исчисления MLJ++:

$$\frac{\Gamma, \alpha{:}A, \alpha{:}B, \Delta \to D}{\Gamma, \alpha{:}A \,\&\, B, \Delta \to D}\ \&л \qquad \frac{\Gamma \to \alpha{:}A \mid \Gamma \to \alpha{:}B}{\Gamma \to \alpha{:}A \,\&\, B}\ \&п$$

$$\frac{\Gamma, \Delta \to \alpha{:}\sim A \mid \Gamma, \Delta, \alpha{:}B \to}{\Gamma, \alpha{:}A \vee B, \Delta \to}\ \vee л \qquad \frac{\Gamma, \Delta, \alpha{:}\sim A, \alpha{:}\sim B \to}{\Gamma, \alpha{:}\sim(A \vee B), \Delta \to}\ \sim\vee л$$

$$\frac{\Gamma, \Delta \to \alpha U{:}\neg A \mid \Gamma, \Delta, \alpha U{:}B \to}{\Gamma, \alpha{:}\neg A \supset B, \Delta \to,}\ \neg\supset л \qquad \frac{\Gamma, \alpha b{:}A \to \alpha b{:}B}{\Gamma \to \alpha{:}A \supset B,}\ \supset п$$

где U не встречается в заключении — где b не встречается в заключении

$$\frac{\Gamma, \alpha{:}(A_1 \supset B) \,\&\, (A_2 \supset B), \Delta \to}{\Gamma, \alpha{:}((A_1 \vee A_2) \supset B), \Delta \to}\ \vee\supset л \qquad \frac{\Gamma, \alpha{:}A_1 \supset (A_2 \supset B), \Delta \to}{\Gamma, \alpha{:}(A_1 \,\&\, A_2) \supset B, \Delta \to}\ \&\supset л$$

$$\frac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha U{:}(A_1 \supset A_2) \mid \Gamma, \alpha U{:}B, \Delta \to}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to,}\ \supset\supset л$$

где U не встречается в заключении

$$\frac{\Gamma, \alpha{:}\sim A \to}{\Gamma \to \alpha{:}A}\ Ап \qquad \frac{\Gamma, \Delta \to \alpha{:}A}{\Gamma, \alpha{:}\sim A, \Delta \to}\ \sim л$$

$$\frac{\Gamma, \alpha{:}A \to D}{\Gamma, \alpha{:}\sim\sim A \to D}\ \sim\sim л \qquad \frac{\Gamma, \alpha{:}\sim\sim A \to}{\Gamma \to \alpha{:}\sim A}\ \sim п$$

$$\frac{\Gamma, \Delta \to \alpha U{:}p \mid \Gamma, \Delta, \alpha U{:}A_2 \to}{\Gamma, \alpha{:}(p \supset A_2), \Delta \to}\ p\supset л$$

Понятия вывода и доказательства являются аналогичными соответствующим понятиям для исчисления MLJ+.

Связка "~" в MLJ++ играет роль аналогичную структурным связкам в дисплейных логиках (Вансинг [1998]), где вхождение формулы $\sim A$ в антецедент секвенции означает сукцедентное вхождение в секвенцию формулы A.

Лемма 3.2.11. Секвенция $\Gamma, \alpha{:}A, \Delta \to \alpha{:}A$ *доказуема в исчислении* MLJ++. □

Лемма 3.2.12. Если секвенция $\Gamma \to A$ *доказуема, то доказуема и секвенция* $\Gamma, B \to A$.

Доказательство:

Для построения доказательства секвенции $\Gamma, B \to A$ применяется снизу вверх такая же последовательность правил, что и для построения доказательства секвенции $\Gamma \to A$. □

Таким образом, можно принять допустимое правило утончения:

$$\frac{\Gamma \to A}{\Gamma, B \to A}$$

Лемма 3.2.13. Следующие правила являются допустимыми:

$$\frac{\Gamma \to \alpha{:}A}{\Gamma \to \alpha{:}A \vee B}\ \vee л1 \qquad \frac{\Gamma \to \alpha{:}B}{\Gamma \to \alpha{:}A \vee B}\ \vee л2$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\Gamma \to \alpha{:}A}{\Gamma, \alpha{:}{\sim}A \to}\ \textit{Лемма 3.2.12}}{\Gamma, \alpha{:}{\sim}A, \alpha{:}{\sim}B \to}\ {\sim}\vee л}{\Gamma, \alpha{:}{\sim}(A \vee B) \to}\ Ап}{\Gamma \to \alpha{:}A \vee B}\ {\sim}л$$

Аналогично для $\vee$л2. □

Лемма 3.2.14. Секвенция $\Gamma, \alpha{:}A, \alpha{:}B, \Delta \to D$ *доказуема тогда и только тогда, когда доказуема секвенция* $\Gamma, \alpha{:}B, \alpha{:}A, \Delta \to D$.

Доказательство аналогично доказательству *леммы 3.2.12*. □

Определение 3.2.15. Пусть [A] обозначает формулу A, не содержащую "~", тогда определим следующие правила, соответствующие допустимым правилам Cutl, Cut2, LT, RT в исчислении MLJ+:

$$\frac{\Gamma \to \alpha{:}[A] \mid \Gamma, \alpha{:}{\sim}[A] \to}{\Gamma \to}\ Cut1+ \qquad LT+\ \frac{\Gamma, \alpha b{:}[A], \Delta \to D}{\Gamma, \alpha{:}[A], \Delta \to D}$$

$$\frac{\Gamma \to \alpha{:}[A] \mid \Gamma, \alpha{:}{\neg}[A] \to}{\Gamma \to}\ Cut2+ \qquad RT+\ \frac{\Gamma \to \alpha{:}[A]}{\Gamma \to \alpha b{:}[A]}$$

Лемма 3.2.16. Исчисление MLJ++ *является расширением исчисления* MLJ+, *т.е. если формула доказуема в* MLJ+, *то она доказуема и в* MLJ++.

Доказательство:

Каждой секвенции исчисления MLJ+ сопоставим секвенцию исчисления MLJ++, получающуюся в результате переноса всех формул из сукцедента в антецедент секвенции со знаком "~". Например, секвенции $\Gamma \to \alpha{:}A$ сопоставляется секвенция $\Gamma, \alpha{:}{\sim}A \to$.

Докажем индукцией по длине MLJ+-вывода, что если секвенция $\Gamma \to \Theta$ доказуема в MLJ+, то сопоставленная ей секвенция $\Gamma, \Theta\sim \to$, где $\Theta\sim = \{\alpha:\sim A \mid \alpha:A \in \Theta\}$, доказуема в MLJ++. Из доказательства этого факта будет следовать доказательство *леммы 3.2.16*.

Базис. Пусть $\Gamma \to \Theta$ является основной секвенцией в MLJ+, тогда сопоставленная ей секвенция $\Gamma, \Theta\sim \to$, где $\Theta\sim = \{\alpha:\sim A \mid \alpha:A \in \Theta\}$, также является основной секвенцией в MLJ++.

Допустим, что для секвенции, полученной на n шаге MLJ+-вывода, сопоставленная ей секвенция доказуема в MLJ++. Докажем это для секвенции, полученной на $n + 1$ шаге MLJ+-вывода.

Пусть $n + 1$ шаг MLJ+-вывода представляет собой применение правила

$$\frac{\Gamma \to \Delta, \alpha:A, \Theta \mid \Gamma \to \Delta, \alpha:B, \Theta}{\Gamma \to \Delta, \alpha:A \,\&\, B, \Theta}\ \&\text{п}$$

Тогда MLJ++-вывод будет представлять собой следующее:

$$\dfrac{\dfrac{\dfrac{\Gamma, \Delta\sim, \alpha:\sim A, \Theta\sim \to \mid \Gamma, \Delta\sim, \alpha:\sim B, \Theta\sim \to}{\Gamma, \Delta\sim, \Theta\sim \to \alpha:A} \text{Ап} \mid \Gamma, \Delta\sim, \Theta\sim \to \alpha:B}{\Gamma, \Delta\sim, \Theta\sim \to \alpha:A \,\&\, B}\ \&\text{п}}{\Gamma, \Delta\sim, \alpha:\sim(A \,\&\, B), \Theta\sim \to}\ \sim\text{л}$$

Аналогичным образом доказываются остальные случаи применения других правил исчисления MLJ+. Рассмотрим применения дополнительных правил MLJ+. Пусть $n + 1$ шаг MLJ+- вывода представляет собой применение правила

$$\frac{\Gamma \to \Theta, \alpha:A \mid \Gamma, \alpha:A \to \Theta}{\Gamma \to \Theta}\ \text{Cutl}$$

Тогда MLJ+- вывод будет представлять собой следующее:

$$\dfrac{\dfrac{\dfrac{\Gamma, \alpha:A, \Theta\sim \to}{\Gamma, \alpha:\sim A, \Theta\sim \to} \sim\sim\text{л} \mid \Gamma, \alpha:\sim\sim A, \Theta\sim \to}{\Gamma, \Theta\sim \to \alpha:A} \text{Ап, } \sim\text{п} \mid \Gamma, \Theta\sim \to \alpha:\sim A}{\Gamma, \Theta\sim \to}\ \text{Cut1+}$$

Аналогичным образом доказываются остальные случаи применения других допустимых правил исчисления MLJ+. □

Следует заметить относительно правила Ап, что если при поиске вывода (т.е. построении вывода снизу вверх) главной формулой является формула вида $\alpha:C \supset B$ или $\alpha:C \,\&\, B$, то применение этого правила становится излишним. Так как обратное правило, т.е. правило для главной формулы вида $\alpha:\sim(C \supset B)$ или $\alpha:\sim(C \,\&\, B)$ слева, дает в сукцеденте посылки опять

формулу вида α:C $\supset$ B или α:C & B. Таким образом, можно наложить на правило Ап следующее ограничение:

$$\frac{\Gamma, \alpha{:}\sim A \rightarrow}{\Gamma \rightarrow \alpha : A,} \quad \text{Ап}$$

где A – пропозициональная переменная или дизъюнктивная формула.

Лемма 3.2.17. Секвенция MLJ++ *доказывается с правилом* Ап *без ограничения тогда и только тогда, когда она доказывается с правилом* Ап *с ограничением.*

Доказательство леммы справа налево является тривиальным. Доказательство в обратную сторону ведется индукцией по степени сложности главной формулы α:A правила Ап и является рутинным. □

Аналогично обстоит дело с правилом ~л, если главной формулой является пропозициональная переменная или дизъюнктивная формула. Таким образом, правило с ограничением представляет собой следующее правило:

$$\frac{\Gamma, \Delta \rightarrow \alpha{:}A,}{\Gamma, \alpha{:}\sim A, \Delta \rightarrow,} \quad \sim\text{л}$$

где A не является пропозициональной переменной и дизъюнктивной формулой.

Лемма 3.2.18. Секвенция MLJ++ *доказывается с правилом* ~л *без ограничения тогда и только тогда, когда она доказывается с правилом* ~л *с ограничением.* □

В результате лемм *3.2.17.* и *3.2.18* исключается возможность зацикливаний при построении вывода снизу вверх.

Теорема 3.2.19. Исчисление MLJ++ *является консервативным расширением* MLJ+, *т.е. если формула* A, не содержащая знака "~", *доказуема в исчислении* MLJ++, *то она доказуема в исчислении* MLJ+.

Доказательство:

Каждую секвенцию MLJ++-вывода заменяют секвенцией исчисления MLJ+, получающейся в результате следующей процедуры:

1. Если в антецеденте секвенции находится формула вида α:~A, то она из него вычеркивается, а в сукцедент вписывается формула α:A.

2. Если в сукцеденте секвенции находится формула вида α:~A, то она из него вычеркивается, а в антецедент вписывается формула α:A.

3. Пункты 1 и 2 выполняются до тех пор, пока не останется ни одной формулы вида α:~A.

Легко видеть, что возникающее при этом дерево секвенций является MLJ+-выводом при допущении в исчислении MLJ+ правила:

$$\frac{\Gamma \to \Delta}{\Gamma \to \Delta}$$

Аналогично обстоит дело с правилами определения *3.2.15*, которые перестраиваются в допустимые правила Cutl, Cut2, LT, RT исчисления MLJ+. □

Относительно исчисления MLJ++ изменяется определение редукционного дерева, введенного ранее относительно классического секвенциального исчисления высказываний. Теперь правила &л и ~~л имеют приоритет перед всеми остальными. А определение ω-степени формулы и секвенции сохраняется в прежнем виде.

Лемма 3.2.20. Для любого MLJ++*-вывода секвенции* $\Gamma \to \alpha$:A *существует редукционное дерево в исчислении* MLJ++ *для той же секвенции* $\Gamma \to \alpha$:A.

Доказательство леммы *3.2.20* следует из перестановочности применения правил вывода исчисления MLJ++. Последнее доказывается аналогично доказательству результата С.К. Клини [1967]. □

Теорема 3.2.21. Исчисление MLJ++ *разрешимо*.

Доказательство.

Покажем, что при применении снизу вверх фигур заключения ω-степень секвенций в редукционном дереве уменьшается. Из этого факта будет следовать доказательство теоремы.

Для правил &л, &п, ¬⊃л, ⊃п, р⊃л, ~л, ~~л это выполняется очевидным образом. Для остальных правил это выполняется в комбинации с другими правилами. Так, правило ~п, которое, на первый взгляд, увеличивает ω-степень секвенции, в комбинации с правилом ~~л уменьшает ω-степень секвенции. Аналогично обстоит дело и с другими правилами. □

Следствие 3.2.22. Для конечной секвенции $\Gamma \to$ A *существует конечное редукционное дерево, каждая ветвь которого оканчивается секвенцией вида* $\Pi \to$*, являющейся либо основной секвенцией, либо секвенцией, где* П *последовательность литералов.*

Доказательство.

Для секвенции вида $\Pi \to$ имеет место $\omega(\Pi \to) = 0$.

Пусть $\omega(\Gamma \to A) = 0$, тогда $\omega(\Gamma) = 0$, т.е. Г состоит из литералов, и $\omega(A) = 0$, т.е. A = ~p или A = p. Если $\Gamma \to$ A основная секвенция, то редукционное дерево построено. Если нет, тогда применяется правило Ап или ~п и

~~л, завершающие построение редукционного дерева. Последнее оканчивается секвенцией вида $\Pi \rightarrow$.

Пусть для $\omega(\Gamma \rightarrow A) \leq n$, где $n > 0$, лемма верна. Докажем это для $\omega(\Gamma \rightarrow A) = n + 1$. Если $\Gamma \rightarrow A$ основная секвенция, то редукционное дерево построено. Если нет, то при применении снизу вверх фигур заключения ω-степень секвенций уменьшается. □

Таким образом, на основе выявленных характеристик исчисления LK+, которое является секвенциальным аналогом правил алгоритма поиска натурального вывода для классической логики высказываний, построено исчисление MLJ++, обладающее такими же свойствами, т.е. отсутствием структурных правил и односукцедентностью секвенций. Исчисление MLJ++ оказалось консервативным расширением интуиционистской логики, причиной чего является введение в язык связки "~", позволившей достичь односукцедентности секвенций исчисления MLJ++.

Кроме того, поиск вывода в исчислении MLJ++ свободен от таких недостатков, как наличие процедуры проверки циклов и перебора последовательностей применения правил вывода.

4. Алгоритм доказательства в интуиционистской логике высказываний

В данной части работы формулируется алгоритм поиска доказательств теорем для интуиционистской пропозициональной логики. Принципиальная идея создания такого алгоритма взята из работы (Болотов и др. [1998]). Однако при создании алгоритма автоматического поиска доказательств для исчисления интуиционистской логики высказываний использовалась в качестве отправной точки не система натурального вывода, а система секвенциальных правил, а именно MLJ++. С другой стороны, формулировка алгоритма привела к появлению натурального префиксного исчисления, названного автором MNJ. Исходя из удобства изложения материала, данное исчисление будет сформулировано перед описанием алгоритма.

4.1. *Натуральное исчисление* MNJ

Префикс определяется так же, как и для исчисления MLJ++. Пусть

1. α^* является непосредственным предшественником α, если и только если α^* удовлетворяет условиям:

$R\alpha^*\alpha$ и $\dot{\forall}\,\beta(R\beta\alpha \Rightarrow R\beta\alpha^*)$, причем $\alpha^* \neq \alpha$ и $\alpha \neq \varepsilon$.

2. [A] – A не содержит связки ~.

Правила вывода:

$$\frac{\alpha{:}[A],\ \beta{:}[B]}{\max(\alpha, \beta){:}A \,\&\, B,\quad \text{где } (\alpha \leq \beta) \vee (\beta \leq \alpha)}\ \&в \qquad \&и\ \frac{\alpha{:}A \,\&\, B}{\alpha{:}A}$$

$$\frac{\alpha{:}[A]}{\alpha{:}A \vee B}\ \vee\text{в} \qquad \frac{\alpha{:}A \,\&\, B}{\alpha{:}B}\ \&\text{и}$$

$$\frac{\alpha{:}[B]}{\alpha{:}A \vee B}\ \vee\text{в} \qquad \frac{\alpha{:}A \vee B,\ \beta{:}{\sim}A}{\beta{:}B,\ \text{где}\ \alpha \le \beta}\ \vee\text{и}$$

$$\frac{\Gamma, \gamma{:}{\sim}B \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}{\sim}B \vdash \beta{:}{\sim}[A]}{\Gamma, \vdash \gamma{:}{\sim\sim}B,\ \ \text{где}\ \alpha \le \beta}\ {\sim\sim}\text{в} \qquad \frac{\alpha{:}{\sim\sim}A}{\alpha{:}A}\ {\sim}\text{и}$$

$$\frac{\Gamma, \alpha{:}[B] \vdash \beta{:}[A]}{\Gamma \vdash \alpha^*{:}[B \supset A],\ \ \text{где}\ \beta \le \alpha}\ \supset\text{в} \qquad \frac{\alpha{:}B \supset A,\ \beta{:}B}{\max(\alpha, \beta){:}A,\ \text{где}\ (\alpha \le \beta) \vee (\beta \le \alpha)}\ \supset\text{и}$$

$$\frac{\Gamma, \gamma{:}[B] \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}[B] \vdash \beta{:}{\neg}[A]}{\Gamma \vdash \gamma^*{:}{\neg}[B],\ \ \text{где}\ (\alpha \le \beta) \vee (\beta \le \alpha)}\ {\neg\neg}$$

$$\frac{\Gamma, \gamma{:}[B] \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}[B] \vdash \beta{:}{\sim}[A]}{\Gamma \vdash \gamma^*{:}{\neg}[B],\ \ \text{где}\ \alpha \le \beta}\ {\sim\neg}\text{в}$$

$$\frac{\Gamma, \gamma{:}{\sim}B \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}{\sim}B \vdash \beta{:}{\neg}[A]}{\Gamma \vdash \gamma{:}{\sim\sim}B,\ \ \text{где}\ (\alpha \le \beta) \vee (\beta \le \alpha)}\ {\neg\sim}\text{в}$$

Макетом вывода формулы A называется непустая конечная последовательность формул $\alpha_1{:}A_1,\ldots, \alpha_k{:}A_k$, удовлетворяющая условиям:

– $\alpha_k{:}A_k$ графически равна формуле $\varepsilon{:}A$,

– каждая $\alpha_i{:}A_i$ есть либо посылка вида $\beta{:}[B]$, где β – новый не существующий в выводе префикс, либо $\alpha{:}{\sim}B$, где α – некоторый уже существующий префикс, либо получена из предыдущих формул по одному из правил вывода,

– если в выводе применялись правила $\supset$в, ${\sim\sim}$в, ${\sim\neg}$в, ${\neg\sim}$в, ${\neg\neg}$в, то все формулы, начиная от последней посылки и вплоть до результата применения данного правила, исключаются из дальнейших шагов построения.

Выводом называется результат применения унификации к макету вывода, префиксы формул которого лишаются переменных вывода (переменных по префиксам).

Доказательством A считается вывод из пустого множества посылок формулы $\varepsilon{:}A$.

Теорема 4.1.1. Исчисление MNJ *непротиворечиво, т.е. если формула* A *доказуема в* MNJ, *то она доказуема в* MLJ++.

Доказательство.

Правила MNJ можно обосновать с помощью правил секвенциального исчисления MLJ++. А из этого факта следует доказательство *теоремы 4.1.1.*

Рассмотрим правило $\supset$в:

$$\frac{\Gamma, \alpha{:}[B] \vdash \beta{:}[A]}{\Gamma \vdash \alpha^*{:}[B \supset A], \text{где } \beta \leq \alpha} \supset_{в}$$

Его обосновывает секвенциальное правило $\supset$п

$$\frac{\Gamma, \alpha b{:}B \to \alpha b{:}A}{\Gamma \to \alpha^*{:}B \supset A,\ \text{где b не встречается в заключении.}} \supset_{п}$$

Правило $\supset$в отражает несколько более общий случай применения правила $\supset$п, а именно его применение в комбинации с правилом RT+; т.е. пусть $\beta \leq \alpha$, тогда

$$\begin{array}{l} \dfrac{\Gamma, \alpha{:}B \to \beta{:}[A]}{\Gamma, \alpha{:}B \to \alpha{:}[A], \text{так как } (\alpha^*)b = \alpha} \ RT+ \\ \dfrac{\Gamma, (\alpha^*)b{:}B \to (\alpha^*)b{:}[A]}{\Gamma \to \alpha^*{:}[B \supset A]} \ \supset_{п} \end{array}$$

2. Правило

$$\frac{\Gamma, \gamma{:}\sim B \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}\sim B \vdash \beta{:}\sim[A]}{\Gamma, \vdash \gamma{:}\sim\sim B, \text{где } \alpha \leq \beta} \sim\sim_{в}$$

Это правило обосновывается секвенциальным правилом ~п следующим образом:

$$\begin{array}{l|l|l} \Gamma, \gamma{:}\sim B \to \alpha{:}[A] & & RT+ \\ \hline \Gamma, \gamma{:}\sim B \to \beta{:}[A] & \Gamma, \gamma{:}\sim B \to \beta{:}\sim[A] & Cut1+ \\ \hline \multicolumn{2}{c|}{\Gamma, \gamma{:}\sim B \to} & \sim\sim_{л} \\ \hline \multicolumn{2}{c|}{\Gamma, \gamma{:}\sim\sim\sim B \to} & \sim_{п} \\ \hline \multicolumn{2}{c|}{\Gamma \to \gamma{:}\sim\sim B} & \end{array}$$

3. Правило

$$\frac{\Gamma, \gamma{:}\sim B \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}\sim B \vdash \beta{:}\neg[A]}{\Gamma \vdash \gamma{:}\sim\sim B, \text{где } (\alpha \leq \beta) \vee (\beta \leq \alpha)} \neg\sim_{в}$$

обосновывается следующим образом, если $\alpha \leq \beta$

$$\begin{array}{l|l|l} \Gamma, \gamma{:}\sim B \to \alpha{:}[A] & & RT+ \\ \hline \Gamma, \gamma{:}\sim B \to \beta{:}[A] & \Gamma, \gamma{:}\sim B \to \beta{:}\neg[A] & Cut2+ \\ \hline \multicolumn{2}{c|}{\Gamma, \gamma{:}\sim B \to} & \sim\sim_{л} \\ \hline \multicolumn{2}{c|}{\Gamma, \gamma{:}\sim\sim\sim B \to} & \sim_{п} \\ \hline \multicolumn{2}{c|}{\Gamma \to \gamma{:}\sim\sim B} & \end{array}$$

и следующим образом, если $\beta \leq \alpha$

$$\begin{array}{l|l|l}
 & \Gamma, \gamma{:}\sim B \to \beta{:}\neg[A] & RT+ \\
\Gamma, \gamma{:}\sim B \to \alpha{:}[A] & \Gamma, \gamma{:}\sim B \to \alpha{:}\neg[A] & Cut2+ \\
\hline
\multicolumn{2}{c|}{\Gamma, \gamma{:}\sim B \to} & \sim\sim л \\
\hline
\multicolumn{2}{c|}{\Gamma, \gamma{:}\sim\sim\sim B \to} & \sim п \\
\hline
\multicolumn{2}{c|}{\Gamma \to \gamma{:}\sim\sim B} &
\end{array}$$

4. Правило

$$\frac{\Gamma, \gamma{:}[B] \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}[B] \vdash \beta{:}\neg[A]}{\Gamma \vdash \gamma^*{:}\neg[B], \text{ где } (\alpha \le \beta) \vee (\beta \le \alpha)} \ \neg\neg в$$

обосновывается секвенциальным правилом $\neg$п следующим образом, если $\alpha \le \beta$:

$$\begin{array}{l|l|l}
\Gamma, \gamma{:}[B] \to \alpha{:}[A] & & RT+ \\
\Gamma, \gamma{:}[B] \to \beta{:}[A] & \Gamma, \gamma{:}[B] \to \beta{:}\neg[A] & Cut2+ \\
\hline
\multicolumn{2}{c|}{\Gamma, \gamma{:}[B] \to} & (\gamma^*)b = \gamma \\
\hline
\multicolumn{2}{c|}{\Gamma, (\gamma^*)b{:}[B] \to} & \neg п \\
\hline
\multicolumn{2}{c|}{\Gamma \to \gamma^*{:}\neg[B]} &
\end{array}$$

и следующим образом, если $\beta \le \alpha$:

$$\begin{array}{l|l|l}
 & \Gamma, \gamma{:}[B] \to \beta{:}\neg[A] & RT+ \\
\Gamma, \gamma{:}[B] \to \alpha{:}[A] & \Gamma, \gamma{:}[B] \to \alpha{:}\neg[A] & Cut2+ \\
\hline
\multicolumn{2}{c|}{\Gamma, \gamma{:}[B] \to} & (\gamma^*)b = \gamma \\
\hline
\multicolumn{2}{c|}{\Gamma, (\gamma^*)b{:}[B] \to} & \neg п \\
\hline
\multicolumn{2}{c|}{\Gamma \to \gamma^*{:}\neg[B]} &
\end{array}$$

5. Правило

$$\frac{\Gamma, \gamma{:}[B] \vdash \alpha{:}[A] \quad \Gamma, \gamma{:}[B] \vdash \beta{:}\sim[A]}{\Gamma \vdash \gamma^*{:}\neg[B], \text{ где } \alpha \le \beta} \ \sim\neg в$$

обосновывается следующим образом:

$$\begin{array}{l|l|l}
\Gamma, \gamma{:}[B] \to \alpha{:}[A] & & RT+ \\
\Gamma, \gamma{:}[B] \to \beta{:}[A] & \Gamma, \gamma{:}[B] \to \beta{:}\sim[A] & Cut1+ \\
\hline
\multicolumn{2}{c|}{\Gamma, \gamma{:}[B] \to} & (\gamma^*)b = \gamma \\
\hline
\multicolumn{2}{c|}{\Gamma, (\gamma^*)b{:}[B] \to} & \neg п \\
\hline
\multicolumn{2}{c|}{\Gamma \to \gamma^*{:}\neg[B]} &
\end{array}$$

6. Правило

$$\frac{\alpha{:}A \ \& \ B}{\alpha{:}B} \ \&и$$

обосновывается следующим образом:

$$\frac{\alpha{:}A,\ \alpha{:}B \to \alpha{:}B}{\alpha{:}A\ \&\ B \to \alpha{:}B}\ \&л$$

7. Правило

$$\frac{\alpha{:}A \vee B,\ \beta{:}{\sim}A}{\beta{:}B,\ \text{где}\ \alpha \le \beta}\ \vee и$$

обосновывается следующим образом:

$$\dfrac{\dfrac{\dfrac{\beta{:}{\sim}A,\ \beta{:}{\sim}B \to \beta{:}{\sim}A \mid \beta{:}{\sim}A,\ \beta{:}{\sim}B,\ \beta{:}B \to}{\beta{:}A \vee B,\ \beta{:}{\sim}A,\ \beta{:}{\sim}B \to}\ Ап}{\beta{:}A \vee B,\ \beta{:}{\sim}A \to \beta{:}B}\ LT+}{\alpha{:}A \vee B,\ \beta{:}{\sim}A \to \beta{:}B}\ \vee л$$

Остальные правила обосновываются аналогично. □

Теорема 4.1.2. Исчисление MNJ *является консервативным расширением интуиционистской логики высказываний.*

Доказательство.

Так как исчисление MLJ++ является консервативным расширением интуиционистской логики, то на основе *теоремы 4.2.1* следует, что и исчисление MNJ является консервативным расширением интуиционистской логики высказываний. □

Однако в связи с особенностью функционирования алгоритма, о котором будет сказано ниже, жесткое разбиение поиска вывода на этап построения макета вывода и этап унификации, как это имеет место в исчислении MLJ++, невозможно. Необходимо уже в период построения макета учитывать определенный тип подстановки.

Исходя из этого, введем новую спецификацию для формулы – подстановка σ, при которой имеет место $\alpha{:}A$. Будем записывать ее в форме суффикса, т.е. $\alpha{:}A\sigma$. Относительно функции подстановки выполняется условие идемпотентности, т.е. для любого префикса Z имеет место $Z\sigma = (Z\sigma)\sigma$.

Тогда правила MNJ модифицируются в MNJ+ следующим образом:

$$\frac{\alpha{:}[A]\sigma',\quad \beta{:}[B]\sigma''}{\max(\alpha\sigma, \beta\sigma){:}A\ \&\ B\sigma,\ \text{где}\ (\alpha\sigma \le \beta\sigma) \vee (\beta\sigma \le \alpha\sigma)}\ \&в \qquad \&и\ \frac{\alpha{:}A\ \&\ B\sigma}{\alpha{:}A\sigma}\quad \frac{\alpha{:}A\ \&\ B\sigma}{\alpha{:}B\sigma}$$

$$\frac{\alpha{:}[A]\sigma}{\alpha{:}A \vee B\sigma}\quad \frac{\alpha{:}[B]\sigma}{\alpha{:}A \vee B\sigma}\ \vee в \qquad \vee и\ \frac{\alpha{:}A \vee B\sigma',\quad \beta{:}{\sim}A\sigma''}{\beta\sigma{:}B\sigma,\ \text{где}\ \alpha\sigma \le \beta\sigma}$$

$$\frac{\Gamma, \gamma{:}{\sim}B\sigma''' \vdash \alpha{:}[A]\sigma''\quad \Gamma, \gamma{:}{\sim}B\sigma''' \vdash \beta{:}{\sim}[A]\sigma'}{\Gamma, \vdash \gamma\sigma{:}{\sim}{\sim}B\sigma,\ \text{где}\ \alpha\sigma \le \beta\sigma}\ {\sim}{\sim}в \qquad {\sim}и\ \frac{\alpha{:}{\sim}{\sim}A\sigma}{\alpha{:}A\sigma}$$

$$\frac{\Gamma, \alpha{:}[B]\sigma' \vdash \beta{:}[A]\sigma''}{\Gamma \vdash (\alpha\sigma)^*{:}[B \supset A]\sigma,} \quad \text{где } \beta\sigma \le \alpha\sigma \qquad \supset_в$$

$$\supset_и \qquad \frac{\alpha{:}B \supset A\sigma', \quad \beta{:}B\sigma''}{\max(\alpha\sigma, \beta\sigma){:}A\sigma,} \quad \text{где } (\alpha\sigma \le \beta\sigma) \vee (\beta\sigma \le \alpha\sigma)$$

$$\frac{\Gamma, \gamma{:}[B]\sigma''' \vdash \alpha{:}[A]\sigma'' \quad \Gamma, \gamma{:}[B]\sigma''' \vdash \beta{:}\neg[A]\sigma'}{\Gamma \vdash (\gamma\sigma)^*{:}\neg[B]\sigma,} \quad \neg\neg_в \text{ где } (\alpha\sigma \le \beta\sigma) \vee (\beta\sigma \le \alpha\sigma)$$

$$\frac{\Gamma, \gamma{:}[B]\sigma''' \vdash \alpha{:}[A]\sigma'' \quad \Gamma, \gamma{:}[B]\sigma''' \vdash \beta{:}\sim[A]\sigma'}{\Gamma \vdash (\gamma\sigma)^*{:}\neg[B]\sigma} \quad \sim\neg_в \text{ где } \alpha\sigma \le \beta\sigma$$

$$\frac{\Gamma, \gamma{:}\sim B\sigma''' \vdash \alpha{:}[A]\sigma'' \quad \Gamma, \gamma{:}\sim B\sigma''' \vdash \beta{:}\neg[A]\sigma'}{\Gamma \vdash \gamma\sigma{:}\sim\sim B,} \quad \neg\sim_в \text{ где } (\alpha\sigma \le \beta\sigma) \vee (\beta\sigma \le \alpha\sigma)$$

В правилах $\neg\neg_в$, $\neg\sim_в$, $\sim\neg_в$, $\sim\sim_в$, $\supset_в$, $\supset_и$, $\&_в$, $\vee_и$ для подстановки σ выполняется следующее условие: для любого префикса Z имеет место $Z\sigma = (Z\sigma')\sigma = (Z\sigma'')\sigma = (Z\sigma''')\sigma$. Подстановка σ' является несовместимой с подстановкой σ'', если для любого префикса Z не существует подстановки σ такой, что $(Z\sigma')\sigma = (Z\sigma'')\sigma$. Макет вывода определяется как и в исчислении MNJ. Выводом формулы A является результат применения к макету вывода формулы A функции подстановки σ с последующим отбрасыванием формул, имеющих в суффиксе подстановку, не совместимую с σ.

Доказательство непротиворечивости правил вывода не отличается от доказательства непротиворечивости исчисления MNJ: правила вывода превращаются в правила вывода исчисления MNJ после применения к ним "унифицирующей" функции подстановки σ.

4.2. *Алгоритм поиска вывода*

В связи с особенностью правил MLJ++ алгоритм сильно отличается от своего классического предшественника. Изменения возникли не только относительно правил вывода, замена которых была необходима, поскольку сменилась лежащая в основе алгоритма логическая система, но изменились также правила анализа формул-целей и формул вывода.

Например, так как формула $\neg\neg A \supset A$ не является в интуиционистской логике теоремой, то многие правила работы алгоритма для классической логики, которые действовали на основе принципа от противного, становятся неправомочными. В классической логике при недостижении формулы-цели вида p, новой целью становилось "противоречие", а в качестве посылки бралась формула $\neg p$. Но это правило не может быть использовано в работе алгоритма для интуиционистской логики высказываний, как и само правило снятия двойного отрицания. В алгоритме, правда, не происходит полный отказ от таких правил. Они достаточно активно используются, но применитель-

но не к интуиционистскому отрицанию "¬", а введенной новой связке "~". Таким образом, изменился сам характер алгоритма: алгоритм стал более сложным и объемным, чем для классической логики.

Ниже излагается в идейном плане алгоритм поиска вывода для интуиционистской логики высказываний. Алгоритм проверялся по теоремам, приведенным в работах: Мендельсон [1971], Новиков [1977], Буль и Боровик [1992], Вессель [1989].

4.2.1. *Общая структура алгоритма*

Итак, работа алгоритма разбивается на два этапа: первый – построение макета вывода с использованием правил поиска вывода MLJ++, сформулированных в предыдущем параграфе, второй – унификация на основе построенной в последней ветви подстановки.

Этап 1. Построение макета вывода.

Идея алгоритма состоит в том, что по некоторой заданной выводимости $\Gamma \vdash A$ создаются две последовательности формул. Первая последовательность представляет собой формулы вывода, вторая является вспомогательной и представляет собой формулы-цели. Изначально последовательности формируются следующим образом:

– если необходимо осуществить вывод $\varepsilon{:}B_1\sigma i,..., \varepsilon{:}B_k\sigma \vdash \varepsilon{:}A\sigma$, то формулы $\varepsilon{:}B_1\sigma i,..., \varepsilon{:}B_k\sigma$ составляют исходный список формул вывода, а формула $\varepsilon{:}A\sigma$ помещается в список целей. При этом формула $\varepsilon{:}A\sigma$ считается главной целью, и если эта цель достигается, то вывод считается законченным;

– если необходимо осуществить вывод $\vdash \varepsilon{:}A\sigma$, то список формул вывода остается пустым, а формула $\varepsilon{:}A\sigma$ помещается в список формул-целей. Естественно, формула $\varepsilon{:}A\sigma$ является главной целью.

В алгоритме осуществляется следующий ряд процедур:

Процедура 1: по формулам вывода проверяется достижимость последней цели в списке целей.

Если последней целью является некоторая формула $\alpha{:}A\sigma'$, то она считается достигнутой, если в выводе существует формула $\beta{:}A\sigma''$, для которой существует подстановка σ такая, что $\beta\sigma \le \alpha\sigma$.

Если последней целью является противоречие, то она считается достигнутой тогда, когда среди неисключенных формул вывода имеются $\alpha{:}A\sigma'$ и $\beta{:}{\sim}A\sigma''$, где $\alpha\sigma \le \beta\sigma$ при некоторой подстановке σ, или имеются $\alpha{:}A\sigma'$ и $\beta{:}\neg A\sigma''$, где $\alpha\sigma \le \beta\sigma$ или $\beta\sigma \le \alpha\sigma$ при некоторой подстановке σ.

Достигнутая цель из списка целей устраняется, а очередной целью становится предыдущая цель.

Процедура 2: формируется последовательность формул вывода.

Находится одна, две формулы, к которым можно применить правила исключения логических связок. Если такие формулы обнаруживаются, то вывод пополняется результатом применения данного правила. Никакие другие правила вывода в этой процедуре не применяются.

Процедура 3: по формулам-целям формируются подцели, помещаемые в последовательность целей, и выбираются новые дополнительные посылки, помещаемые под очередным номером в вывод.

Процедура 4: по формулам вывода формируются новые цели, которые помещаются в последовательность целей.

Эта процедура выполняется в том случае, если все возможные указанные в *Процедуре 2* правила вывода применены, последней целью в последовательности является противоречие, но при этом эта цель не достигнута. Какие именно вводятся подцели, определяется видом содержащихся в выводе формул.

Процедура 5: сигнализируется необходимость автоматического применения в выводе соответствующих правил вывода.

Каждый случай достижения некоторой цели сигнализирует о необходимости выполнить некоторые шаги вывода. В большинстве случаев необходимо выполнить соответствующее правило введения логической связки.

Этап 2. Унификация.

Применяет к построенному макету вывода функцию подстановки, которая построена в суффиксе конечной формулы. При этом отбрасываются те формулы макета, которые имеют в суффиксе подстановку, не совместимую с окончательной. Тем самым макет вывода перестраивается в натуральный вывод исчисления MNJ.

4.2.2. *Описание алгоритма*

Определяется *главная цель* вывода. Таковой является формула, стоящая справа от знака выводимости в заданном метаутверждении о выводимости. Данная формула помещается в качестве начальной в последовательность целей. Если слева от знака выводимости стоят формулы, то все они помещаются в качестве начальных в последовательность формул вывода. Переход к 2.

Анализ множества формул вывода.

Если множество формул вывода пусто, то переход к 4.

Если множество формул вывода непусто, то переход к 3.

Умозаключения по формулам вывода.

Осуществляются все непосредственные умозаключения по аналитическим правилам; при этом

– если на формуле стоит метка "stop", то к ней невозможно применить аналитическое правило;

– на посылки правила вывода &и, ~и, ¬и ставится *метка* "parents(n)", где n порядковый номер первой посылки, указывающая на недопустимость вторичного применения к ней указанного правила, а на заключение ставится метка "child(n)", которая указывает на происхождение заключения вследствие применения аналитических правил вывода;

– на посылки правил вывода ∨и, ⊃и ставится *метка* "parents(n, m)", где n порядковый номер первой посылки, а m – порядковый номер второй посылки, указывающая на недопустимость вторичного применения к ним указанных правил, а на заключение ставится метка "child(n, m)", которая указывает на происхождение заключения вследствие применения аналитических правил вывода.

Переход к 11 (к анализу достижимости цели).

Анализируется главный знак очередной последней цели – Z.

Z – противоречие. Переход к 11.

Z = α:A ⊃ Bσ. Переход к 5.

Z = α:¬Aσ. Переход к 6.

Z = α:A & Bσ. Переход к 7.

Z = α:A ∨ Bσ. Переход к 8.

Z = α:Aσ, где A – пропозициональная переменная. Переход к 9.

Z = α:~Aσ. Переход к 10.

Формула αb:Aσ включается в последовательность формул вывода в качестве посылки, подцелью становится αb:Bσ, переход к 3.

Формула αb:Aσ включается в последовательность формул вывода в качестве посылки, подцелью становится противоречие, переход к 3.

α:Aσ и α:Bσ становятся подцелями. Начинаем всегда работать с α:Aσ.

7.1. Подцель – α:Aσ. Формула α:Aσ, а также, начиная с этого момента, все новые формулы вывода и цели помечаются меткой "andl", переход к 11. Если α:Aσ достижима, что устанавливается в 11, то со всех псевдоформул вывода и всех целей снимается метка "andl", переход к 7.2 – к достижению второго члена конъюнкции.

7.2. Подцелью становиться α:Bσ. Формула α:Bσ, а также, начиная с этого момента, все новые формулы вывода и цели помечаются меткой "and2", переход к 11. Если α:Bσ достижима, что устанавливается в 11, то со всех псевдоформул вывода и всех целей снимается метка "and2", переход к 12.

Последней целью становится противоречие, а формула α:~(A ∨ B)σ берется в качестве посылки. Переход к 3.

Формула α:~Aσ включается в последовательность формул вывода в качестве посылки, подцелью становится противоречие, переход к 3.

Формула α:~~Aσ включается в последовательность формул вывода в качестве посылки, подцелью становится противоречие, переход к 3.

11. *Проверка достижимости последней цели*

(а) Если последней целью является формула α:Aσ'', то она считается достигнутой тогда, когда среди неисключенных формул вывода имеется формула β:Aσ', для которой существует σ такая, что $\beta\sigma \leq \alpha\sigma$. В противном случае цель считается не достигнутой.

(б) Если последней целью является противоречие, то она считается достигнутой тогда, когда среди неисклюенных формул вывода имеются α:[A]σ' и β:~[A]σ'', где $\alpha\sigma \leq \beta\sigma$ при некоторой подстановке σ, или имеются α:[A]σ и β:¬[A]σ'', где $\alpha\sigma \leq \beta\sigma$ или $\beta\sigma \leq \alpha\sigma$ при некоторой подстановке σ.

Если цель достигнута, то переход к 12.

Если цель не достигнута, то переход к 13.

12. *Достигнута цель Zn* вида α:Aσ на основании того, что в последовательности вывода существует некоторая формула β:Aσ' (или, если цель – противоречие, то наличие соответствующих противоречащих формул).

12.1. Если Zn – *главная цель*, то Zn помечается как *достигнутая*, выход из алгоритма – выводимость обоснована, переход 15.

12.2. Если предыдущей к достигнутой цели Zn является цель Zn-1 = α*:B $\supset$ Aσ, то:

из последовательности целей устраняется цель Zn,

очередной целью становится ближайшая ранее не достигнутая цель Zn-1,

(c) к формуле вывода β:Aσ' применяется правило $\supset$в, т.е. формула β*:B $\supset$ Aσ' переносится в вывод вместе со всеми метками, наложенными на цель Zn-1,

(d) в последовательности формул вывода делается отметка, что все формулы, начиная с посылки α:Bσ и заканчивая формулой β:Aσ', в дальнейших шагах вывода принимать участия не могут,

(e) если в число последних формул попадает формула с меткой "child()", являющаяся заключением некоторого правила вывода, то с соответствующей формулы, являющейся посылкой данного применения правила вывода, снимается метка "parents()",

(j) если в число последних формул попадает формула с меткой типа "V*", являющаяся результатом достижения цели, взятой по некоторой формуле вывода, то с формулы вывода, породившей данную цель, снимается метка типа "V*" (см. п. 14). Переход к 3.

Если предыдущей к достигнутой цели Zn = α:Aσ' является цель Zn-1 = α:A & Bσ, то переход к 7.2.

Если предыдущей к достигнутой цели Zn = α:Bσ'' является цель Zn-1 = α:A & Bσ, то:

Zn устраняется из последовательности целей,

очередной целью становится ближайшая ранее не достигнутая цель Zn-1, применяется правило &в, т.е. формула β:A & Bσ'', где $\beta \leq \alpha\sigma''$, переносится в вывод и в последовательности формул вывода на формулу β:A & Bσ'' ставится метка "stop", переход к 3.

12.5. Если предыдущей к достигнутой цели Zn = противоречие, является цель Zn-1 = α*:$\neg$Bσ, то

из последовательности целей устраняется цель Zn,

очередной целью становится ближайшая ранее не достигнутая цель Zn-1,

применяется правило $\sim\neg$в или $\neg\neg$в, т.е. формула α*:$\neg$Bσ' переносится в вывод вместе со всеми метками, наложенными на цель Zn-1,

в последовательности формул вывода делается отметка, что все формулы, начиная с посылки α:Bσ и заканчивая предпоследней формулой вывода, в дальнейших шагах вывода принимать участия не могут,

(e) если в число последних формул попадает формула с меткой "child()", являющаяся заключением некоторого правила вывода, то с соответствующей формулы, являющейся посылкой данного применения правила вывода, снимается метка "parents()",

(j) если в число последних формул попадает формула с меткой типа "V*", являющаяся результатом достижения цели, взятой по некоторой формуле вывода, то с формулы вывода, породившей данную цель снимается метка типа "V*". Переход к 3.

12.6. Если предыдущей к достигнутой цели Zn = противоречие, является цель Zn-1 = α:$\sim$Bσ или Zn-1 = α:pσ или Zn-1 = α:A $\vee$ Bσ, то

из последовательности целей устраняется цель Zn,

очередной целью становится ближайшая ранее не достигнутая цель Zn-1,

(c) применяется правило ~~в или ¬~в, т.е. появляется в выводе формула α:~~Bσ или α:~~pσ' или α:~~(A $\vee$ B)σ' вместе со всеми метками, наложенными на цель Zn-1,

(d) в последовательности формул вывода делается отметка, что все формулы, начиная с α:~~Bσ (соответственно α:~pσ' или α:~(A $\vee$ B)σ') и заканчивая предыдущей формулой, в дальнейших шагах вывода принимать участия не могут,

(e) если в число последних формул попадает формула с меткой "child()", являющаяся заключением некоторого правила вывода, то с соответствующей формулы, являющейся посылкой данного применения правила вывода, снимается метка "parents()",

(j) если в число последних формул попадает формула с меткой типа "V*", являющаяся результатом достижения цели, взятой по некоторой формуле вывода, то с формулы вывода, породившей данную цель, снимается метка типа "V*". Переход к 3.

12.7. В противном случае из последовательности целей устраняется цель Zn, очередной целью становится предыдущая цель, переход к 11.

13. *Цель Zn не достигнута*.

Если Zn является противоречием и она не достигнута, то 14.

Если Zn является формулой и она не достигнута, то 4.

14. *Выбор новых целей по формулам вывода*.

Если при просмотре сверху вниз всех формул вывода в их последовательности имеются сложные формулы, не отмеченные метками "stop", "V1", "V2", "V3", "V5", "V6" или "V7", то:

14.1. Если α:A $\vee$ Bσ, то в качестве подцели берется α:~Aσ. Формула α:A $\vee$ Bσ и цель α:~Aσ помечаются метками "V1(n)", где n – номер формулы α:A $\vee$ Bσ. Переход к 4.

14.2. Если α:A $\supset$ Bσ, то

14.2.1. Если A = C & D, то в качестве новой цели берется формула α:C $\supset$ (D $\supset$ B)σ. Формула α:A $\supset$ Bσ и цель α:C $\supset$ (D $\supset$ B)σ помечаются метками "V2(n)", где n – номер формулы α:A $\supset$ Bσ. Переход к 4.

Если A = C $\vee$ D, то в качестве новой цели берется формула α:(C $\supset$ B) & (D $\supset$ B)σ. Формула α:A $\supset$ Bσ и цель α:(C $\supset$ B) & (D $\supset$ B)σ помечаются метками "V3(n)", где n – номер формулы α:A $\supset$ Bσ. Переход к 4.

Если A = C ⊃ D и не стоит метка "V4", то новой целью становится формула α:(D ⊃ B)σ. α:A ⊃ Bσ и цель α:(D ⊃ B)σ помечаются метками "V4(n)", где n – номер формулы α:A ⊃ Bσ. Переход к 4.

Если A – пропозициональная переменная или A = ¬C, то новой целью становится формула αU:Aσ, где U – новая переменная вывода. Формула α:A ⊃ Bσ и цель αU:Aσ помечаются метками "V5(n)", где n – номер формулы α:A ⊃ Bσ. Переход к 4.

Если A = C ⊃ D и стоит метка "V4", то в качестве новой цели берется формула αU:Aσ, где U – новая переменная вывода. Формула α:A ⊃ Bσ и цель αU:Aσ помечаются метками "V6(n)" (добавление метки не отменяет других меток, поставленных на формулу ранее), где n – номер формулы α:A ⊃ Bσ. Переход к 4.

14.3. Если α:¬Aσ, то

Если A = C & D, то в качестве новой цели берется формула α:C ⊃ ¬Dσ. Формула α:¬Aσ и цель α:C ⊃ ¬Dσ помечаются метками "V2(n)", где n – номер формулы α:¬Aσ. Переход к 4.

Если A = C ∨ D, то в качестве новой цели берется формула α:¬C & ¬Dσ. Формула α:¬Aσ и цель α:¬C & ¬Dσ помечаются метками "V3(n)", где n – номер формулы α:¬Aσ. Переход к 4.

Если A = C ⊃ D и не стоит метка "V4", то в качестве новой цели берется формула α:¬Dσ. Формула α:¬Aσ и цель α:¬Dσ помечаются метками "V4(n)", где n – номер формулы α:¬Aσ. Переход к 4.

Если A – пропозициональная переменная или A = ¬C, то новой целью становится формула αU:Aσ, где U – новая переменная вывода. Формула α:¬Aσ и цель αU:Aσ помечаются метками "V5(n)", где n – номер формулы α:¬Aσ. Переход к 4.

Если A = C ⊃ D и стоит метка "V4", то в качестве новой цели берется формула αU:Aσ, где U – новая переменная вывода. Формула α:¬Aσ и цель αU:Aσ помечаются метками "V6(n)" где n – номер формулы α:¬Aσ. Переход к 4.

14.4. Если в выводе имеется формула вида α:~Aσ, то

Если A = C & D, то в качестве новой цели берется формула α:C & Dσ. Формула α:~Aσ и цель α:Aσ помечаются метками "V7(n)", где n – номер формулы α:~Aσ. Переход к 4.

Если A = C ∨ D и не стоит метка "V8", то новой целью становится формула α:~C & ~Dσ. α:~Aσ и цель α:~C & ~Dσ помечаются метками "V8(n)", где n – номер формулы α:~Aσ. Переход к 4.

Если A = C ⊃ D, то в качестве новой цели берется формула α:C ⊃ Bσ. Формула α:~Aσ и цель α:Aσ помечаются метками "V7(n)", где n – номер формулы α:~Aσ. Переход к 4.

Если A = ¬C, то новой целью становится формула α:Cσ. Формула α:~Aσ и цель α:Aσ помечаются метками "V7(n)", где n – номер формулы α:~Aσ. Переход к 4.

14.4.5. Если A = C ∨ D и стоит метка "V8", то в качестве новой цели берется формула α:Aσ. Формула α:~Aσ и цель α:Aσ помечаются метками "V7(n)", где n – номер формулы α:~Aσ. Переход к 4.

15. *2 этап*. Происходит процесс минимизации. Он состоит в просмотре снизу вверх формул вывода. Если на данную формулу не ссылается ни один комментарий (информация о том, как получена формула) из нижестоящих формул, то она считается несущественной для вывода и исключается из него. Переход к 16.

16. Унификация на основе подстановки, построенной в суффиксе конечной формулы. Если в ней какие-то переменные вывода не определены, то они заменяются пустой последовательностью термов вывода.

Конец.

Список меток.

"stop" – недопустимость применения аналитического правила к формуле вывода,

"andl" – анализ первого члена конъюнктивной цели,

"and2" – анализ второго члена конъюнктивной цели,

"child()" – формула получена по аналитическому правилу из некоторой формулы вывода,

"parents()" – к формуле было применено аналитическое правило,

метки типа "V*" – недопустимость повторного взятия новой одной и той же цели по формуле вывода, где * варьируется от 1 до 8.

4.2.3. *Разбор примеров*

Ниже приводится доказательство ослабленного закона Пирса, которое осуществлено при помощи данного алгоритма. Требуется доказать: ¬¬(((p ⊃ q) ⊃ p) ⊃ p).

Формулы вывода	Формулы целей
1. a: ¬(((p ⊃ q) ⊃ p) ⊃ p) – пос.	1. ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p)
2. ab:p – пос.	2. ⊥
3. aUc:((p ⊃ q) ⊃p) – пос.	3. a:¬p – цель от 1
4. abV:(((p ⊃ q) ⊃ p) ⊃ p) {U/bV}	4. ⊥
5. a:¬p {U/bV, V/0}	5. aU:(((p ⊃ q) ⊃ p) ⊃ p) – достигнута 4
6. aWd:((p ⊃ q) ⊃ p) – пос.	6. aUc:p
7. aWd:~p – пос.	7. aW:(((p ⊃ q) ⊃ p) ⊃ p)
8. aWde:q – пос.	8. aWd:p
9. aWde:~p – пос.	9. ⊥
10. aWdZf:p – пос.	10. aWd:q ⊃ p

11. aWdeX:p ⊃ q {Z/eX}	11. aWde:p
12. aWdeX: p {Z/eX}	12. ⊥
13. aWde:~~p {Z/eX, X/0}	13. aWdZ:p ⊃ q
14. aWde:p {Z/eX, X/0}	14. aWdZf:q
15. aWd:q:Dp {Z/eX, X/ 0}	15. aWdQ:p ⊃ q
16.aWdQg:p – пос.	16. aWdQg:q
17. aWdQg:~q – пос.	17. ⊥
18. aWdQg:~~q {U/bV}	
19. aWdQg:q {U/bV}	
20. aWdQ:p ⊃ q {U/bV}	
21. aWdQ:p {U/bV}	
22. aWd:~~p {U/bV, Q/0}	
23. aWd:p {U/bV, Q/0}	
24. aW:(((p ⊃ q) ⊃ p) ⊃ p) {U/bV, Q/0}	
25. ε:¬¬((p ⊃ q) ⊃ p) ⊃ p) {U/bV,Q/0}	

Приведем пояснения к данному выводу. Так как необходимо построить доказательство формулы, то первоначально в макете вывода нет никаких формул, а в качестве главной цели выступает формула ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p). Так как множество формул вывода равно нулю, то происходит анализ цели: очередной целью становится противоречие ⊥ (п. 2 формул целей), а в качестве первой посылки берется a:¬(((p ⊃ q) ⊃ p) ⊃ p) (п. 1 формул вывода). Никаких правил вывода (т.е. правил исключения логических связок) применить нельзя. Текущая цель не достижима. Так как последняя графически равна ⊥, то происходит анализ формул вывода. Первой формулой вывода является формула a:¬(((p ⊃ q) ⊃ p) ⊃ p). Тогда, согласно п. 14.3.3. алгоритма, очередной целью становится формула a:¬p (п. 3 формул целей) и ставится метка "V4(l)" на соответствующие формулы. Цель a:¬p не достижима. Происходит анализ текущей цели. Очередной целью становится ⊥, а второй посылкой формула ab:p (п.2 формул вывода). Текущая цель не достижима. Правила вывода применить нельзя. Происходит анализ формул вывода. Первой формулой является a:¬(((p ⊃ q) ⊃ p) ⊃ p). Но по ней уже была взята новая цель по пункту 14.3.3. алгоритма и стоит метка "V4(l)". Можно взять цель по пункту 14.3.5., т.е. aU:(((p ⊃ q) ⊃ p) ⊃ p). Ставится на соответствующие формулы метка "V6(l)". Новая цель не достижима. Происходит анализ цели. Новой целью становится aUc:p (п. 6 формул целей), а посылкой формула aWc:((p ⊃ q) ⊃ p). Проверяется на достижимость текущая цель, она достижима при U/bV (она, конечно, достижима и при U/b, но так как вывод еще не окончен и может понадобиться подстановка вместо U более

длинной последовательности параметров вывода, чем b, то необходимо применить здесь наиболее общую подстановку). Выводится формула abV:(((p ⊃ q) ⊃ p) ⊃ p) по ⊃и, 2, 3, на нее ставится метка "V6(l)", из вывода исключается формула п.3. Текущей целью становится aU: (((p ⊃ q) ⊃ p) ⊃ p), она достижима и вычеркивается, следующей целью становится противоречие. Противоречие достижимо (пп. 1 и 4 формул вывода), предыдущей целью является a:¬p, тогда применяется правило ¬¬в. Вводится в вывод формула (ab)*:¬p, т.е. a:¬p, на нее ставится метка "V4(l)". Формулы вывода с п. 2 по п.4 исключаются из вывода. Так как исключенная формула п.4 abV:((p ⊃ q) ⊃ p) ⊃ p имеет метку "V6(l)", то с формулы вывода п. 1 снимается метка "V6(l)", но остается метка "V4(l)". Цель a:¬p вычеркивается, так как она достижима. Текущей целью становится противоречие. Происходит анализ формул вывода. С формулы a:¬(((p ⊃ q) ⊃ p) ⊃ p) уже была взята цель согласно пунктам 14.3.3 и 14.3.5 описания алгоритма. Однако с нее была снята метка "V6(l)". Таким образом, можно применить еще раз пункт 14.3.5 алгоритма. Очередной целью становится формула aW:((p ⊃ q) ⊃ p) ⊃ p. Ставится на соответствующие формулы метка "V6(1)". Цель aW:((p ⊃ q) ⊃ p) ⊃ p не достижима и в результате дальнейшей работы алгоритма следующей целью становится aWd:p, а в качестве посылки в вывод вносится aWd:((p ⊃ q) ⊃ p). Далее очередной целью становится противоречие и появляется новая посылка aWd:~p. Снова происходит анализ формул вывода. Теперь брать новые цели с формулы a:¬(((p ⊃ q) ⊃ p) ⊃ p) нельзя, (на ней стоят метки "V4(l)" и "V6(l)", а потому анализируется следующая неисключенная формула вывода, т.е. п. 6 вывода aWd:((p ⊃ q) ⊃ p). В качестве новой цели берется aWd:(q ⊃ p) (п. 9 формул целей). Ставится на соответствующие формулы метка "V4(6)". Цель aWd:(q ⊃ p) не достижима. Происходит анализ текущей цели. Очередной целью становится aWde:p, а новой посылкой формула aWde:q. Текущая цель не достижима. Тогда очередной целью становится ⊥, а новой посылкой формула aWde:~p. Правила вывода применить нельзя. Происходит анализ формул вывода. Но по формуле aWd:((p ⊃ q) ⊃ p) уже была взята новая цель согласно пункту 14.2.3 алгоритма (стоит метка "V4(6)"). Можно взять цель по пункту 14.2.5, т.е. aWdZ:(p ⊃ q). Ставится на соответствующие формулы метка "V6(6)". Новая цель не достижима. Происходит анализ цели. Новой целью становится aWdZf:q, а посылкой формула aWdZf:p. Проверяется на достижимость текущая цель, она достижима при {Z/eX}. Выводится формула aWdeX:(p ⊃ q) по ⊃в, 8, 10, на нее ставится метка "V6(6)", из вы-

вода исключается формула п. 10. Цель aWdZ:(p ⊃ q) достижима, она вычеркивается, следующей целью становится противоречие. Применяются аналитические правила. В выводе появляется формула 12: aWdeX:p {Z/eX} по ⊃и 6, 11. Противоречие достижимо (пп. 9 и 12 при {X/0}), предыдущей целью является aWde:p, тогда применяется правило ~~в и вводится формула вывода 13: aWde:~~p {Z/eX, X/0}, исключается соответствующий подвывод. Так как в число исключенных формул вывода входит формула, отмеченная "V6(6)", то с формулы вывода п. 6 снимается метка "V6(6)", но остается метка "V4(6)". Применяется правило ~в к п. 13 вывода. Текущая цель является aWde:p, она достижима, а предыдущей целью является aWd:q ⊃ p. Применяется правило ⊃в, в выводе появляется 15: aWd:q ⊃ p {Z/eX, X/O} с меткой "V4(6)". Цель aWd:q ⊃ p достижима, она вычеркивается, текущей целью становится противоречие. Оно недостижимо. Происходит анализ формул вывода. С формулы aWd:((p ⊃ q) ⊃ p) уже была взята цель согласно пунктам 14.2.3 и 14.2.5 описания алгоритма. Однако так как на ней стоит только метка "V4(6)", то можно применить еще раз пункт 14.2.5. алгоритма. Очередной целью становится формула aWdQ:(p ⊃ q). Ставится на соответствующие формулы метка "V6(6)". Формула aWdQ:(p ⊃ q) не достижима, а потому следующей целью становится aWdQg:q, а в качестве посылки в вывод вносится aWdQg:p. Цель недостижима, тогда очередной целью становится противоречие, а в качестве посылки в вывод вносится aWdQg:~q. Последняя цель достижима (противоречие пп. 5 и 16). Выводится формула п. 18: aWdQg:~~q и снимается двойное отрицание. Текущей целью становится aWdQg:q, она достижима. Выводится формула 20: aWdQ:(p ⊃ q) по ⊃в 16, 19, с меткой "V6(6)", из вывода исключаются соответствующие формулы.

Цель aWdQ:(p ⊃ q) достижима, она вычеркивается, следующей целью становится противоречие. Применяются аналитические правила. В выводе появляется формула п.21: aWdQ:p по ⊃и 6, 20. Противоречие достижимо (пп. 7 и 21 при {Q/0}), предыдущей целью является п. 9 aWd:p. Тогда применяется правило ~~в и вводится формула вывода 22: aWd:~~p, исключается соответствующий подвывод, снимаются соответствующие метки, применяется правило ~и к 22. Текущая цель является aWd:p, она достижима, а предыдущей целью является aW:(((p ⊃ q) ⊃ p) ⊃ p), применяется правило ⊃в, в выводе появляется 24: aW:(((p ⊃ q) ⊃ p) ⊃ p) с меткой "V6(l)", исключается соответствующий подвывод и снимаются соответствующие метки. Цель aW:(((p ⊃ q) ⊃ p) ⊃ p) достижима, она вычеркивается, текущей целью становится противоречие. Оно дости-

жимо (1 и 24). Предыдущей целью является ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p), применяется правило ¬¬в, в выводе появляется 25: ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p), исключаются соответствующие формулы и снимаются метки. Главная цель ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p) достижима, она вычеркивается, макет вывода построен.

Представим теперь обоснование данной формулы не в форме процедуры поиска вывода, а в виде процедуры доказательства:

Формулы вывода	
1. a: ¬(((p ⊃ q) ⊃ p) ⊃ p)	посылка
2. ab:p	посылка
3. aUc:((p ⊃ q) ⊃p)	посылка
4. abV:(((p ⊃ q) ⊃ p) ⊃ p) {U/bV}	⊃в, 2
5. a:¬p {U/bV}	¬¬в, 1, 4
6. aWd:((p ⊃ q) ⊃ p)	посылка
7. aWd:~p	посылка
8. aWde:q	посылка
9. aWde:~p	посылка
10. aWdZf:p	посылка
11. aWdeX:p ⊃ q {Z/eX}	⊃в, 8
12. aWdeX:p {Z/eX}	⊃и, 6, 11
13. aWde:~~p {Z/eX, X/0}	~~в, 9, 12
14. aWde:p {Z/eX, X/0}	~и, 13
15. aWd:q ⊃ p {Z/eX, X/ 0}	⊃в, 14
16.aWdQg:p – пос.	посылка
17. aWdQg:~q – пос.	посылка
18. aWdQg:~~q {U/bV}	¬~в, 5, 16
19. aWdQg:q {U/bV}	~и, 18
20. aWdQ:p ⊃ q {U/bV}	⊃в, 19
21. aWdQ:p {U/bV}	⊃и, 6, 20
22. aWd:~~p {U/bV, Q/0}	~~в, 7, 21
23. aWd:p {U/bV, Q/0}	~и, 22
24. aW:(((p ⊃ q) ⊃ p) ⊃ p) {U/bV, Q/0}	⊃в, 23
25. ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p) {U/bV,Q/0}	¬¬в, 1, 24

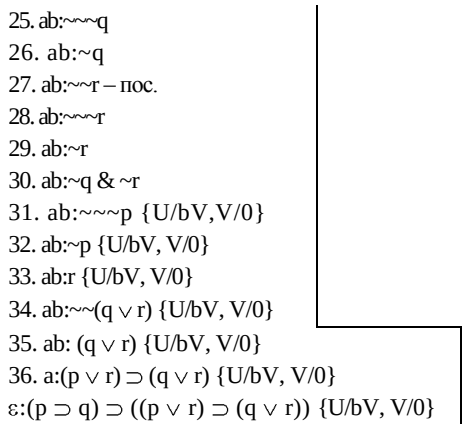
Далее, когда построен макет вывода с конечной формулой ε:¬¬((p ⊃ q) ⊃ p) ⊃ p) {U/bV, Q/0}, наступает второй этап работы алгоритма – минимизация с унификацией, в результате которой получается следующий MNJ-вывод:

Формулы вывода	
1. a:¬(((p ⊃ q) ⊃ p) ⊃ p)	посылка
2. ab:p	посылка
3. abc:((p ⊃ q) ⊃ p)	посылка

4. ab:(((p ⊃ q) ⊃ p) ⊃ p)	⊃в, 2
5. a:¬p	¬¬в, 1, 4
6. ad:((p ⊃ q) ⊃ p)	посылка
7. ad:~p	посылка
8. adg:p	посылка
9. adg:~q	посылка
10. adg:~~ q	~~в, 7, 8
11. adg:q	~и, 10
12. ad:p ⊃ q	⊃в, 11
13. ad: p	⊃и, 6, 12
14. ad: ~~p	¬~в, 13, 5
15. ad:p	~и, 14
16. a:(((p ⊃ q) ⊃ p) ⊃ p)	⊃в, 15
17. ε:¬¬(((p ⊃ q) ⊃ p) ⊃ p)	¬¬в, 1, 16

В качестве другого примера работы алгоритма по нахождению доказательств возьмем теорему монотонности (p ⊃ q) ⊃ ((p ∨ r) ⊃ (q ∨ r)). На первом этапе строится следующий макет вывода:

Формулы вывода	Формулы цели
1. a:p ⊃ q – пос.	1. ε:(p ⊃ q) ⊃ ((p ∨ r) ⊃ (q ∨ r))
2. ab:p ∨ r – пос.	2. a:(p ∨ r) ⊃ (q ∨ r)
3. ab:~(q ∨ r) – пос.	3. ab: q ∨ r
4. aU:~p – пос.	4. ⊥
5. abV:r {U/bV}	5. aU:p
6. ab:~~q – пос.	6. ⊥
7. ab:q	7. ab:~q & ~r
8. ab: q ∨ r	8. ab:~q
9. ab:~~~q	9. ⊥
10. ab:~q	11. ab:q ∨ r
11. ab:~~r – пос.	12. ab:q
12. ab:r	13. ab:~r
13. ab: q ∨ r	14. ⊥
14. ab:~~~r	15. ab:q ∨ r
15. ab:~r	16. ab:r
16. ab:~q & ~r	17. ab:~p
17. ab:~~p {U/bV, V/0}	18. ⊥
18. ab:p {U/bV, V/0}	19. ab~q & ~r
19. ab:q {U/bV,V/0}	20. ab~q
20. ab:~~p – пос.	21.
21. ab:p	22. ab: q ∨ r
22. ab:~~q – пос.	23. ab:q
23. ab:q	24. ab:~r
24. ab:q ∨ r	25. ⊥



Приведем комментарии к работе алгоритма в сокращенном виде. Вначале происходит анализ формул целей, в результате чего в выводе появляются посылки 1-3, а последней целью становится противоречие. Происходит анализ первой формулы вывода a:p ⊃ q, новой целью становится aU:p. Она недостижима. Снова цель – противоречие, в выводе новая посылка aU:~p. Тут возможно применить правило ∨и к формулам 2 и 4 при подстановке {U/bV}.

Цель – противоречие – не достижима, поэтому происходит анализ формул вывода. Формулы 1 и 2 анализу не подлежат, так как первая была уже проанализирована, а ко второй было применено правило ∨и. В результате анализа третьей формулы вывода новой целью становится ab:~q & ~r. Далее алгоритм пытается достичь эту цель, что происходит на 16-ом шаге вывода. Целью становится "старая" цель пункта 6 – противоречие. Противоречие в выводе имеется: это выражения 5: abV:r и 15: ab:~r при подстановке {U/bV, V/0}. Применяется правило ~~в, из вывода исключаются формулы 4-16, вывод пополняется формулой 17: ab:~~p. Далее достигается цель 5: aU:p. Следует заметить, что в результате исключения формул 4-16 из вывода, с формулы 2 ab:p ∨ r была снята метка применения правила ∨и, а с формулы 3 ab:~(q ∨ r) была снята метка "V7(3)", и теперь эти формула могут стать источником новых целей, что и происходит. Вначале анализируется формула 2: ab:p ∨ r, новой целью становится 17: ab:~p, а затем опять формула 3: ab:~(q ∨ r). Новой целью становится 19: ab: ~p & ~q. Последняя из этих целей достигается на 30-м шаге вывода, а первая – на 32-м. Текущей целью на данном этапе работы алгоритма является цель 4: ⊥. Противоречие в выводе имеется между формулами 18: – p и 32: – ~p. Работа алгоритма продолжается дальше по знакомой из первого примера "заключительной" схеме, последовательно применяя правила ⊃в.

На втором этапе после минимизации и унификации возникает следующий MNJ-вывод:

Формулы вывода	
l. a:p ⊃ q	посылка
2. ab:p ∨ r	посылка
3. ab:~(q ∨ r)	посылка
4. a:~p	посылка
5. ab:r	∨и, 2, 4
6. ab:~~r	посылка
7 ab:r	~и, 6
8. ab:q ∨ r	∨в, 7
9. ab:~~~r	~~в, 8, 3
10. ab:~r	~и, 9
11: ab:~~p	~в, 10, 5
12. ab: p	~и, 11
13. ab: q	⊃и 1, 12
14. ab:~~q	посылка
15. ab:q	~и, 14
16. ab:q ∨ r	∨в, 15
17. ab:~~~q	~~в, 16, 3
18. ab:~q	~и, 17
19. ab:~~(q ∨ r)	~~в, 13, 18
20. ab:(q ∨ r)	~и, 19
21. a:(p ∨ r) ⊃ (q ∨ r)	⊃в, 20
22. ε:(p ⊃ q) ⊃ ((p ∨ r) ⊃ (q ∨ r))	⊃в, 21

4.3. *Адекватность алгоритма*

Докажем соответствие данного алгоритма MLJ++-дереву.

Теорема 4.3.1. Пусть неисключенные формулы алгоритмического вывода соответствуют формулам антецедента секвенции исчисления MLJ++, *а последняя формула-цель – формуле сукцедента этого исчисления. Тогда для каждого редукционного дерева секвенции* Φ → ε:A *существует алгоритмический вывод этой формулы* ε:A из Φ.

Доказательство теоремы ведется индукцией по длине вывода Φ → ε:A.

1. Базис: длина вывода $n = 0$.

1.1. Основной секвенции Γ, α:A, Θ, β:~A, Δ →, где $\alpha \leq \beta$, соответствует следующий алгоритмический вывод.

вывод	цели
Γ – посылки	⊥

α:A – посылка
Θ – посылки
β:~A – посылка
Δ – посылки

Цель $\perp$ достижима. Таким образом, алгоритмически получено противоречие из Γ, α:A, Θ, β:~A, Δ (при доказательстве теоремы не учитывается подстановка, записываемая в суффиксе формулы, так как на результат доказательства это не влияет).

1.2. Основной секвенции Γ, β:~A, Θ, α:A, $\Delta \rightarrow$, где $\alpha \leq \beta$, соответствует следующий алгоритмический вывод.

вывод	цели
Γ – посылки	$\perp$
β:~A – посылка	
Θ – посылки	
α:A – посылка	
Δ – посылки	

Тем самым алгоритмически получено противоречие из Γ, β:~A, Θ, α:A, Δ.

2. По индуктивному допущению предполагается, что для вывода длины n теорема верна, т.е. вывод длины n может быть перестроен в алгоритмический вывод. Докажем теперь, что это можно сделать и для вывода длиной n + 1. Пусть на n + 1 шаге применялись следующие правила:

$$2.1.\quad \frac{\Gamma, \alpha{:}A, \alpha{:}B, \Delta \rightarrow D}{\Gamma, \alpha{:}A \,\&\, B, \Delta \rightarrow D}\ \&л$$

Доказательству секвенции Γ, α:A & B, $\Delta \rightarrow$ D соответствует в алгоритме достижение формулы-цели D из Γ, α:A & B, Δ. Алгоритмически происходит применение правила исключения конъюнкции к формуле α:A & B и переход к достижению формулы-цели D из Γ, α:A, α:B, Δ, что соответствует выводу секвенции Γ, α:A, α:B, $\Delta \rightarrow$ D. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции Γ, α:A, α:B, $\Delta \rightarrow$ D. А значит, существует алгоритмический вывод D из Γ, α:A & B, Δ.

	вывод	цели		вывода	цели
	Γ	D		Γ	D
	α:A			α:A & B	
Если	α:B		то	α:A	
	Δ			α:B	
	…			Δ	
	D			…	
				D	

$$2.2.\quad \frac{\Gamma \to \alpha{:}A \mid \Gamma \to \alpha{:}B}{\Gamma \to \alpha{:}A \,\&\, B}\quad \&п$$

Доказательству секвенции $\Gamma \to \alpha{:}A \,\&\, B$ соответствует в алгоритме достижение формулы-цели $\alpha{:}A \,\&\, B$ из Γ. Алгоритмически происходит переход к достижению формулы $\alpha{:}A$ из Γ. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma \to \alpha{:}A$. В последовательности формул вывода появляется формула A, и алгоритм переходит к достижению формулы B из Γ, A. Но так как A является логическим следствием из Γ, то – к достижению формулы B из Γ. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma \to \alpha{:}B$. Алгоритм вычеркивает формулу-цель B из последовательности формул-целей в силу ее достижимости и последней целью становится $A \,\&\, B$. Далее алгоритм на основе существования обоих конъюнктов в последовательности формул вывода применяет правило &в и вводит в вывод формулу $A \,\&\, B$. Тем самым формула-цель $A \,\&\, B$ достигается из Γ, A, B. Но так как A и B являются логическим следствием из Γ, то достигается и формула $A \,\&\, B$ из Γ.

вывод	цели
Γ – посылки	$\alpha{:}A \,\&\, B$
…	$\alpha{:}A$ – При ее
$\alpha{:}A$	достижении –
…	вычеркивается
$\alpha{:}B$	$\alpha{:}B$ – При ее
$A \,\&\, B$	достижении –
	вычеркивается

$$2.3.\quad \frac{\Gamma, \alpha b{:}A \to \alpha b{:}B}{\Gamma \to \alpha{:}A \supset B}\quad \supset п$$

Доказательству секвенции $\Gamma \to \alpha{:}A \supset B$ соответствует в алгоритме достижение формулы-цели $\alpha{:}A \supset B$ из Γ. Алгоритмически происходит переход к достижению формулы $\alpha b{:}B$ из Γ, $\alpha b{:}A$, где $\alpha b{:}A$ принимается как новая посылка. По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции Γ, $\alpha b{:}A \to \alpha b{:}B$. В последовательности формул вывода появляется формула $\alpha b{:}B$, алгоритм вычеркивает формулу-цель $\alpha b{:}B$ из последовательности формул-целей в силу ее достижимости и последней целью становится $\alpha{:}A \supset B$. Далее, алгоритм применяет правило $\supset$в и вводит в вывод формулу $\alpha{:}A \supset B$, исключая все формулы вывода от

последней посылки (т.е. $\alpha b{:}A$) до вновь полученной формулы $\alpha{:}A \supset B$. Тем самым формула-цель $\alpha{:}A \supset B$ достигается из Г.

вывод	цели
Г – посылки	$\alpha{:}A \supset B$
$\alpha b{:}A$ – посылка	$\alpha b{:}B$ – вычеркивается в
…	силу достижимости
$\alpha b{:}B$	
$\alpha{:}A \supset B$	

2.4. $\dfrac{\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha U{:}(A_1 \supset A_2) \mid \Gamma, \alpha U{:}B, \Delta \to}{\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to}$ $\supset\supset$л

Доказательству секвенции $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \to$ соответствует в алгоритме достижение противоречия из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta$. Алгоритмически, в соответствии с пунктом 14.2.3 алгоритма, происходит переход к достижению формулы $\alpha{:}A_2 \supset B$ из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta$, и на формулу $\alpha{:}(A_1 \supset A_2) \supset B$ накладывается метка V4, запрещающая ее повторный аналогичный анализ. Формула-цель $\alpha{:}A_2 \supset B$ алгоритмически достигается, так как при взятии с формулы $\alpha{:}(A_1 \supset A_2) \supset B$, помечаемой теперь новой меткой V6, согласно пункту 14.2.5 алгоритма, новой цели $\alpha W{:}(A_1 \supset A_2)$ происходит достижение ранее введенной цели по формуле $aWb{:}A_2$. Таким образом, в выводе появляется формула $\alpha{:}A_2 \supset B$. Далее, так как согласно пунктам 12.2.(j) и 14 с формулы $\alpha{:}(A_1 \supset A_2) \supset B$ снята метка V6, алгоритм вновь берет новую цель $\alpha U{:}(A_1 \supset A_2)$, а на формулу $\alpha{:}(A_1 \supset A_2) \supset B$ накладывается метка, запрещающая ее повторный аналогичный анализ.

По индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma, \alpha{:}A_2 \supset B, \Delta \to \alpha U{:}(A_1 \supset A_2)$, т.е. $\Gamma, \alpha{:}A_2 \supset B, \Delta \vdash \alpha U{:}(A_1 \supset A_2)$. Тогда в выводе появляется формула $\alpha U{:}(A_1 \supset A_2)$, которая, согласно нашему предположению, является достижимой, а текущей целью становится предыдущая цель – противоречие. Так как выражение вида $\alpha{:}A_2 \supset B$ выводимо из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta$, о чем шла речь выше, то и выражение $\alpha U{:}(A_1 \supset A_2)$ выводимо из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta$. Теперь применяется правило $\supset$и и в последовательности формул вывода записывается формула $\alpha U{:}B$. Таким образом, из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta \vdash \alpha U{:}B$. Но по индуктивному допущению существует алгоритмический вывод, соответствующий выводу секвенции $\Gamma, \alpha U{:}B, \Delta \to$, т.е. из $\Gamma, \alpha U{:}B, \Delta$ достижимо противоречие. Следовательно, противоречие достижимо и из $\Gamma, \alpha{:}(A_1 \supset A_2) \supset B, \Delta$, что и обосновывает нижнюю секвенцию.

Остальные правила исчисления MLJ++ разбираются аналогично уже рассмотренным. □

Следствие 4.3.2. Алгоритм полон относительно исчисления MLJ++. □

Теорема 4.3.3. Алгоритм непротиворечив.

Доказательство тривиально, так как алгоритм использует непротиворечивую систему MNJ. □

Теорема 4.3.4. Алгоритм конечен.

Доказательство.

Алгоритм не использует никакие другие правила, кроме сформулированных в исчислении MLJ++. Таким образом, на основе *теоремы 3.2.20* (о конечности редукционного дерева) следует конечность алгоритма. □

Таким образом, построен алгоритм, который точно воспроизводит построение доказательства в исчислении MLJ++. На основе этого исчисления сформулировано натуральное исчисление MNJ, которое не является "простым переложением" секвенциального исчисления, как это сделано у Р. Дикхофа (Дикхоф [1992]), а использует более минимальные средства при построении вывода, чем его секвенциальный аналог. Так, например, секвенциальное правило $\vee\supset$л явно не имеет аналога в правилах исчисления MNJ, а зафиксировано только как эвристика поиска вывода в правилах алгоритма.

ЛИТЕРАТУРА

Андервуд (Underwood J.)

[1990] A constructive completeness proof for intuitionistic propositional calculus // Technical Report 90-1179, Cornell University

Байн, Клаланд (Baain D., Klarlund N.)

[1995] Hardware Verification using Monadic Second-Order Logic // BRICS Technical Report Series, RS-95-7

Барателла, Масини (Baratella S., Masini A.)

[2003] A proof-theoretic investigation of a logic of positions // Annals of Pure and Applied Logic. Vol. 123, Issues 1-3. p. 135-162

Бачмейр, Ганзингер (Bachmair L., Ganzinger H.)

[2001] A Theory of Resolution // Handbook of Automated Reasoning, Elsevier. J.A. Robinson and A. Voronkov (eds.)

Бентли (Bentley B.)

[2001] Validating The Intel Pentium 4 Processor. Intel Technology Journal, Issue 1, (on-line: `http://www`. intel.com/technogy/-itj/q12001.htm)

Бернс (Byrnes J.)

[1999] Proof search and normal forms in natural deduction // PhD thesis, Pittsburgh

Бибель (Bibel W.)

[1992] Deduktion: Automatisierung der Logik. Munchen, Wien, Oldenburg

[1998] Let's plan it deductively! // Artificial Intelligence Journal. Vol. 103, Issues 1-2. p. 183-208

Болотов А.Е., Бочаров В.А., Горчаков А.Е.

[1996] Алгоритм поиска вывода в классической пропозициональной логике // Труды научно-исследовательского семинара логического центра Института философии РАН. М.: ИФРАН

[1998] Алгоритм поиска вывода в классической логике предикатов // Логические исследования. Вып. 5. М.: Наука

Болотов, Фишер (Bolotov A., Fisher M.)
[1997] A Resolution Method for Computation Tree Branching Time Temporal Logic // IV International Workshop on Temporal Representation and Reasoning (TIME97). Published by the IEEE, Florida

Бохеньский (Bocheński J.M.)
[1978] Formale Logik. Freiburg [im Breisgau], Munchen: Alber

Бочаров В., Болотов А., Горчаков А, Шангин В. (Bocharov V., Bolotov A., Gorchakov A., Shangin V.)
[2003] Proof-searching algorithm in first order classical natural deduction calculus // 12th International Congress of Logic, Methodology and Philosophy of Science. Oviedo (Spain), August 7-13.

Бочаров В.А., Маркин В.И.
[1994] Основы логики. М.: Космополис

Бохманн (Bochmann G.)
[1982] Hardware Specification with Temporal Logic: An Example // IEEE Transactions on Computers. Vol. C-31. № 3

Братко И.
[1990] Программирование на языке Пролог для искусственного интеллекта. Пер. с англ. А.И. Лупенко, А.М. Степанова. Под ред. А.М. Степанова. М.

Брода, Фингер, Руссо (Broda K., Finger M., Russo A.)
[1999] Labelled Natural Deduction for Substructural Logics // Logic Journal of the IGPL. Vol. 7(3). p. 283-318

Брукс (Brooks R.)
[1985] A Layered Intelligent Control System for a Mobile Robot // Proceedings of the Third International Symposium of Robotics Research. Gouvieux, France, MIT Press
[1986] A Robust Layered Control System for a Mobile Robot // IEEE Journal of Robotics and Automation. Vol. 2, № 1
[1991] Intelligence Without Representation // Artificial Intelligence Journal. Vol. 47. p. 139-159

Буль, Боровик (Bole L., Borowik P.)
[1992] Many-valued logics: Theoretical foundations. Vol. 1. Berlin

Бэйзин, Мэттьюс, Вигано (Basin D., Matthews S., Viganò L.)
[1998] Natural Deduction for Non-Classical Logics // Studia Logica. Vol. 60, № 1. p. 119-160

Ваалер (Waaler A.)
[2000] Position paper: Termination in intuitionistic connection-driven search // Automated Reasoning with Analytic Tableaux and Related Methods. International Conference, Tableaux 2000. Position Papers and Tutorials

Вайх (Weich K.)
[1998] Decision Procedures for Intuitionistic Prepositional Logic by Program Extraction // Proceedings of the TABLEAUX' 98 Conference. LNCS 1397. Springer-Verlag. p. 292-308

Вансинг (Wansing H.)
[1998] Displaying Modal Logic. Trends in Logic Series. Kluwer Academic Publishers, Dordrecht

Вессель (Wessel H.)
[1989] Logik. VEB Deutscher Verlag der Wissenschaften, Berlin

Вишняков В.А., Буланже Д.Ю., Герман О.В.
[1991] Аппаратно-программные средства процессоров логического вывода. М.: Радио и связь

Войшвилло Е.К.
[1967] Понятие. М.: Изд-во МГУ

Воробьев Н.Н. (Vorob'ev N.N.)
[1958] Новый алгоритм выводимости в конструктивном исчислении высказываний // Труды математического института им. В.А. Стеклова. LII, Проблемы конструктивного направления в математике. Сб. М.-Л.
[1970] A new algorithm for derivability in the constructive propositional calculus // Amer. Math. Soc. Transl. Vol. 94(2). p. 37-71

Вулдридж (Wooldridge M.)
[1999] Intelligent Agents // G. Weiss (ed.). Multiagent Systems. The MIT Press

Габбай (Gabbay D.)
[1996] Labelled Deductive Systems. Vol. 1. Foundations. Oxford University Press

Генцен Г.
[1967] Исследования логических выводов // Математическая теория логического вывода. М.: Наука

Городецкий В.И
[1997] Свойства моделей координации поведения агентов и модель планирования на основе аукциона // DIAMAS'97, СПб.

Городецкий В.И., Грушинский М.С., Хабалов А.В.
[1999] Многоагентные системы. Обзор // Новости искусственного интеллекта. М. Вып. 3

Городецкий, Котенко (Gorodetski V., Kotenko I.)
[2002] The Multi-agent Systems for Computer Network Security Assurance: frameworks and case studies // IEEE ICAIS-02. IEEE International Conference Artificial Intelligence Systems. Proceedings

Горчаков А.Е., Макаров В.В., Спирин С.В.
[1998] Некоторые проблемы компьютерной реализации автоматической процедуры доказательства для натурального вывода (on-line: http://www.logic.ru (Логические исследования))

Д'Агостино, Мондадори (D'Agostino M., Mondadori M.)
[1994] The Taming of the Cut Classical Refutation with Analytic Cut // Journal of Logic and Computation. Vol. 4(3). p. 285-319

Девис, Патнэм (Davis M., Putnam H.)
[1960] A computing Procedure for Quantification Theory // Journal of Association for Computing Machinery, Vol. 7, Issue 3. ACM Press, N.Y.

Де Лима, Лингенфельдер (de Lima F., Lingenfelder C.)
[2000] Presentation of proofs in modal natural deduction // Journal of Logic and Computation. Vol. 10(4). p. 527-572

Дикхоф (Dyckhoff R.)
[1992] Contraction-free Sequent Calculi for Intuitionistic Logic // The Journal of Symbolic Logic. Vol. 57(3), p. 795-807.

Джорджеф, Пэлл, Поллак, Тамбе, Вулдридж (Georgeff M., Pell B., Pollack M., Tambe M., Wooldridge M.)

[1998] The Belief-Desire-Intention Model of Agency // Proceedings of 5th International Workshop Agent Theories Architectures, and Languages" (ATAL), Paris, France, July 4-7, Lecture Notes in Computer Science, Vol. 1555, Springer-Verlag

Изабель
[1996] Izabelle. User's guide. University of Cambridge

Кларк, Эмерсон (Clarke E.M., Emerson E.A.)
[1982] Using branching time logic to synthesize synchronization skeletons // Science of Computer Programming, Vol. 2, Issue 3, p. 241-266, Elsevier

Кларк, Джа, Маррего (Clarke E., Jha S., Marrero W.)
[1998] Using state space exploration and a natural deduction style message derivation engine to verify security protocols // Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET), 8-12 June, Shelter Island, N.Y., IFIP Conferens Proceedings 125, p. 87-106, Chapman & Hall

Кларк, Грамберг, Пелед (Clarke E., Grumberg O., Peled D.)
[1999] Model checking. Cambridge, MA; London, MIT Press

Клини С.К.
[1967] Перестановочность применений правил в генценовских исчислениях LK и LJ // Математическая теория логического вывода. М.: Наука
[1973] Математическая логика. М.: Мир

Корн (Korn D.)
[1999] Konstruktiv adäquate Beweisautomatisierung für intuitionistische Logik. Sankt Augustin: Infix (Dissertationen zur künstlichen Intelligenz; Bd. 198)

Корн, Крейтц (Korn D., Kreitz C.)
[1996] A Constructively Adequate Refutation System for Intuitionistic Logic // Technischer Bericht AIDA-96-14, TU Darmstadt
[1997] A Constructively Adequate Refutation System for Intuitionistic Logic // International Conference TABLEAUX' 97 (Position Papers), Technischer Bericht CRIN 97-R-030, Centre de Recherche en Informatique de Nancy

Куайн (Quine W.)

[1950] On natural deduction // The Journal of Symbolic Logic. Vol. 15(2)

Кудинов В., Цуканов М.

[2003] Принципы Создания Системы Дистанционного Образования на Основе Мультиагентных Технологий // Материалы Конференции ИТО 2003 – Информационные Технологии в Образовании, М.

Ли (Li D.)

[1997] Unification algorithms for eliminating and introducing quantifiers in natural deduction automated theorem proving // Journal of Automated Reasoning. Vol. 18, № 1

Макаров В.В.

[1998] Автоматическое доказательство теорем интуиционистской пропозициональной логики // Современная логика: проблемы теории, истории и применения в науке. Материалы 5-ой всероссийской научной конференции. СПб.

[2001] Односукцедентное секвенциальное классическое пропозициальное исчисление // Материалы международной конференции "Третьи Смирновские чтения", М.

[2002] Поиск доказательства теорем в интуиционистской логике высказываний // Человек – культура – общество. Актуальные проблемы философских, политологических и религиоведческих исследований. Материалы Международной конференции, посвященной 60-летию воссоздания философского факультета в структуре МГУ им. М.В. Ломоносова. М.

Мейер (Meier A.)

[2000] System Description: TRAMP Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level // Conference on Automated Deduction (CADE 2000). p. 460-464

Мендельсон Э.

[1976] Введение в математическую логику. 2-е изд., испр. М.: Наука

Миглиоли, Москато, Орнахи (Miglioli P., Moskato U., Ornaghi M.)

[1994] An Improved Refutation System for Intuitionistic Predicate Logic // Journal of Automated Reasoning. Vol. 13, p. 361-373.

Минский (Minsky M.)
[1963] Steps toward Artificial Intelligence // Computer and Thoughts (eds. E. Feigenbann and D. Feldmann). McGraw Hill
[1988] The Society of Mind. New York; London: Simon & Schuster

Минц Г.Е.
[1967] Теорема Эрбрана // Математическая теория логического вывода. М.: Наука
[1985] Исчисления резолюций для неклассических логик // Семиотика и информатика. Вып. 25. М.

Непейвода Н.Н.
[2000] Прикладная логика. Учебное пособие. 2-е изд., испр. и доп. Новосибирск: Изд-во Новосиб. ун-та

Новиков П.С.
[1977] Конструктивная математическая логика с точки зрения классической. М.: Наука

Оттен (Otten J.)
[1997] IleanTAP: An Intuitionistic Theorem Prover. In Automated Reasoning with Analytic Tableaux and Related Methods // Inter. Conference, TABLEAUX'97, LNAI, Springer-Verlag

Оттен и Крайтц (Otten J., Kreitz C.)
[1995a] A Connection Based Proof Method for Intuitionistic Logic // Theorem Proving with Analytic Tableaux and Related Methods, TABLEAUX'95, Springer-Verlag
[1995b] T-string unification: Unifying prefixes in non-classical proof methods // Proc. 4-th TABLEAUX'95. Vol. 918 of LNAI, p. 244-260, Springer-Verlag, Berlin
[2000] Towards efficient prefix unification in non-classical theorem proving // Technical Report A1DA-00-03, Intellectics Group, Darmstadt University of Technology. May 2000

Пеллетье (Pelletier F.J.)
[1986] Seventy-five graduated problems for testing automatic theorem provers // Journal of Automated Reasoning, Vol. 2, № 2, p. 191-216. Kluwer Academic Publishers
[1988] Errata for 75 problems // Journal of Automated Reasoning. № 4
[1998] Automated natural deduction in THINKER // Studia Logica. Vol. 60, № 1

Пнуэли (Pnueli A.)
[1977] The Temporal Logic of Programs // Proc. of the Eighteenth Symposium on the Foundations of Computer Science. Springer-Verlag
[1984] Now you can compose temporal logic specifications // Proceedings of the 16-th ACM Sumposium on Theory of Computing, p. 51-63, Washington, D.C., ACM Press

Поллок (Pollock J.)
Skolemization and unification in natural deduction (неопубликованная версия статьи доступна по адресу: http://oscarhome.socsci.arizona.edu/ftp/publications.html)

Порториро (Portoraro F.)
[1998] Strategic constructions of Fitch-style proofs // Studia Logica. Vol. 60, № 1

Расмуссен (Rasmussen T.)
[2001] Labelled Natural Deduction for Interval Logics // Computer Science Logic, 15th International Workshop, Paris, France, September 10-13, Lecture Notes in Computer Science, Vol. 2142, p. 308-323, Springer-Verlag

Рао, Джорджеф (Rao A., Georgeff M.)
[1998] Decision Procedures for BDI Logics // Journal of Logic and Computation. Vol. 8(3). p. 293-342

Рассел, Норвиг [Russell S., Norvig P.)
[2003] Artificial Intelligence: A Modern Approach (Second Edition). Prentice Hall

Салин, Френзен, Хариди (Sahlin D., Franzen T., Haridi S.)
[1992] An Intuitionistic Predicate Logic Theorem Prover // Journal of Logic and Computation. Vol. 2(5). p. 619-656

Сиг (Sieg W.)
[1992] Mechanism and search: aspects of proof theory. Pittsburgh
[1994] Intercalation calculi for classical logic // Carnegie Mellon Technical Report PHIL-46. Philosophy, Methodology and Logic

Сиг, Бернс (Sieg W., Byrnes J.)
[1998] Normal natural deduction proofs (in classical logic) // Studia Logica. Vol. 60, № 1

Смирнов А.В., Новодворский А.Е.
[1995] Язык описания логических систем // Логические исследования. Вып. 3. М.: Наука

Смирнов В.А.
[1972] Формальный вывод и логические исчисления. М.
[1995] Поиск доказательства в натуральном интуиционистском исчислении предикатов с ε-символом и предикатом существования // Логические исследования. Вып. 3. М.: Наука
[2000] Теория логического вывода. М.: РОССПЭН

Смирнов В.А., Маркин В.И., Новодворский А.Е., Смирнов А.В.
[1996] Доказательство и его поиск (курс логики и компьютерный практикум) // Логика и компьютер. Вып. 3. М.: Наука (Серия "Кибернетика – неограниченные возможности и возможные ограничения")

Стёрлинг, Брэдфилд (Stirling C., Bradffield J.)
[2001] Modal logics and mu-calculi: an introduction // Handbook of Process Algebra. Preprint. Elsevier

Суханов К.Н.
[1973] Критический очерк гносеологии интуиционизма. Челябинск: Юж.-Ур. кн. изд-во

Такеути Г.
[1978] Теория доказательств. М.: Мир

Таммет (Tammet T. A.)
[1996] A Resolution Theorem Prover for Intuitionistic Logic // Proceedings of CADE-13, LNAI, Springer-Verlag

Тимофеев А.
[2000] Мультиагентное управление коллективом роботов // Проблемы информатизации. Вып. 1. М.

Успенский В. А.
[1960] Абстракция актуальной бесконечности // Философская энциклопедия: В 5 т. Т. 1. М.

Фиттинг (Fitting M.)
[1969] Intuitionistic Logic Model Theory and Forcing // Studies in Logic and the Foundations of Mathematics. North-Holland publishing company. Amsterdam, London

[1983] Proof Methods for Modal and Intuitionistic Logic. D. Reidel, Dodrecht.

[1990] First-Order Logic and Automated Theorem Proving. Springer-Verlag, New York, Berlin

Фреге (Frege G.)

[1883] Uber den Zweck der Begriffsschrift, 1882 // Translstion by T. Bynum, "On the Aim of the Conceptual Notation", in Conceptual Notation and Related Articles, (ed) T. Bunym, Oxford, 1972, p. 90-100

Ханнебауэр, Вендлер, Пагелло (Hannebauer M., Wendler J., Pagello E.)

[2001] Balancing Reactivity and Social Deliberation in MAS - From RoboCup to Real-World Applications // Lecture notes in Artificial Intelligence. Vol. 2103. Springer-Verlag

Хинтикка (Hintikka J.)

[1953] A new approach to sentential logic // Societas Scientarium Fennica Commentationes Physico-Mathematicae. Vol. 17, №3. The Finnish Society of Sciences and Letters, Helsinki

[1955] Notes on the quantification theory // Societas Scientarium Fennica Commentationes Physico-Mathematicae. Vol. 17, № 12. The Finnish Society of Sciences and Letters, Helsinki

Худельмайер (Hudelmaier J.)

[1988] A PROLOG program for intuitionistic logic // SNS-Bericht, Universität Tübingen

[1993] An O(n log n)-space decision procedure for intuitionistic propositional logic and Kuroda logic // Rapporto interno 99-93, Dipartimento di Scienze dell'Informazione. Università degli Studi di Milano

Чень Ч., Ли Р.

[1983] Математическая логика и автоматическое доказательство теорем. М.: Наука

Черч (Church A.)

[1936a] A note on the Entscheidungsproblem // The Journal of Symbolic Logic. Vol. 1(1)

[1936b] Correction to A note on the Entscheidungsproblem // The Journal of Symbolic Logic. Vol. 1(3)

[1960] Введение в математическую логику. М.: Иностранная литература

Шангин В.О.

[2000] Теорема корректности для алгоритма поиска вывода в классической пропозициональной логике // Материалы VI Международной научной конференции "Современная логика: проблемы теории, истории и применения в науке". СПб., СПбГУ.

[2002] Автоматический поиск натурального вывода в интуиционистской логике и проблема дубликации // Материалы VII Международной научной конференции "Современная логика: проблемы теории, истории и применения в науке". СПб., СПбГУ.

[2003] Метатеоретические свойства натурального вывода // Материалы IV Международной конференции "Смирновские чтения". М.: ИФРАН.

Шанин Н.А., Давыдов Г.В., Маслов С.Ю., Минц Г.Е., Оревков В.П., Слисенко А.О.

[1964] Алгорифм машинного поиска естественного логического вывода в исчислении высказываний. Л.: Наука

Шмит, Крайтц (Schmitt S., Kreitz C.)

[1996] Converting Non-Classical Matrix Proofs into Sequent-Style Systems // 13-th International Conference on Automated Deduction. LNAI 1104. p. 418-432, Springer-Verlag

Шрамко Я.В.

[1997] Логическое следование и интуиционизм (проблема релевантизации интуиционистской логики). Киев: ВИПОЛ

Шталмарк (Stalmarck G.)

[1991] Normalization theorems for full first order classical deduction // The Journal of Symbolic Logic. Vol. 56. p. 129-149

Чекинов С.

[2001] Интеллектуальные программные исполнительные устройства (агенты) в системах связи // Информационные Технологии. № 12

Эмерсон, Кларк (Emerson E., Clarke E.)

[1980] Characterizing correctness properties of parallel programs using fixpoints // Proceedings of Automata Languages and Pro-

gramming, 7th Colloquium, Noordwijkerhout, The Netherland, July 14-18, Lecture Notes in Computer Science. Vol. 85. p. 169-181, Springer-Verlag

Эндрюс (Andrews P.)

[1980] Transforming matings into natural deduction proofs // Proceedings of 5th Conference on Automated Deduction, Les Arcs, France, July 8-11, Lecture Notes in Computer Science, Vol. 87, p. 281-292, Springer-Verlag

Янсен (Jansen G.)

[1990] Hardware Verification using Temporal Logic: A Practical View. Formal VLSI Correctness Verifiation. VLSI Design Methods-II. Elsevier Science Publishers

ОГЛАВЛЕНИЕ