Московский государственный университет им. М.В. Ломоносова
Философский факультет
Кафедра логики

На правах рукописи

ШАНГИН ВАСИЛИЙ ОЛЕГОВИЧ

АВТОМАТИЧЕСКИЙ ПОИСК НАТУРАЛЬНОГО ВЫВОДА В КЛАССИЧЕСКОЙ ЛОГИКЕ ПРЕДИКАТОВ

Диссертация на соискание ученой степени кандидата философских наук
Специальность 09.00.07 – Логика

Научный руководитель:
проф. Бочаров В.А.

Москва
2004

ОГЛАВЛЕНИЕ

Введение

Актуальность темы исследования. Проблема поиска логического вывода традиционно считается одной из центральной тем логики. Бурное развитие данной проблематики в XX веке стимулировали, с одной стороны, фундаментальные работы Г. Генцена и Ж. Эрбрана и, с другой, появление ЭВМ. Возможность использования ЭВМ в процессе поиска логического вывода привела к появлению проблематики *автоматического* (машинного) поиска логического вывода.

В настоящее время определяющим фактором при предпочтении одной логической системы перед другой становится наличие (автоматической) процедуры поиска вывода. Такие процедуры существенным образом облегчают нахождение логического вывода и активно используются в педагогической работе.

В свою очередь, эти процедуры являются объектом исследования и постоянно сравниваются между собою по степени сложности (вычислительные затраты на поиск вывода), гибкости (возможность адаптации к нескольким логическим системам), удобства (понятный интерфейс, возможность поиска вывода как от посылок к заключению, так и от заключения к посылкам) и т.д.

В диссертационном исследовании тема автоматического поиска логического вывода ограничивается поиском вывода в натуральном исчислении 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 всегда имеет место отношение логического следования.

Основное внимание авторы программ автоматического поиска натурального вывода обычно уделяют исчислениям типа Генцена. *Непрямое* правило удаления квантора существования в таких исчислениях предполагает построение дополнительного подвывода, гарантирующего наличие отношение логического следования между посылками и заключением. Поскольку построение дополнительного подвывода приводит к усложнению вывода, удобнее, по нашему мнению, пользоваться прямым правилом удаления квантора существования, т.е. искать вывод в исчислениях типа Куайна.

Степень разработанности проблемы. Долгое время исследования в области автоматического поиска логического вывода были сосредоточены на поиске вывода с помощью метода резолюции, секвенциальных и аналитико-табличных типов логического вывода.

Наличие *свойства подформульности* (в выводе формулы используются только подформулы или отрицания подформул этой формулы), которое следует из теоремы Генцена об устранении сечения, существенно облегчает поиск вывода в данных исчислениях [Генцен].

С нашей точки зрения, перечисленные логические методы являются не более, чем методами проверки формул на общезначимость и выполнимость. В то же время, традиционно под логическим выводом подразумевается возможность выведения некоторой формулы из некоторого (возможно, пустого) множества посылок, что достигается только лишь в аксиоматических и натуральных исчислениях.

Исчисления последнего вида особенно интенсивно исследуются на предмет автоматического поиска в них вывода в конце 80-х – начале 90-х гг. XX века.

Так, Дж. Поллок [Pollock] предложил программу поиска натурального вывода OSCAR в классической логике предикатов (а также в некоторых неклассических логиках) с использованием сколемовских термов. Он показал, что OSCAR работает в 40 раз эффективнее программы OTTER [Pollock], основанной на методе резолюций. С другой стороны, круг логических проблем, которые решает OSCAR, шире, чем аналогичный круг для OTTER. Дж. Поллоком была выдвинута также гипотеза, что OSCAR обладает свойством семантической полноты, т.е. что OSCAR может найти вывод любой общезначимой формулы классической логики предикатов.

Д. Пеллетье [Pelletier] предложил программу поиска натурального вывода Thinker в классической логике предикатов (а также в некоторых неклассических логиках) с предикатом равенства. Показывается, что Thinker решает 75 тестовых проблем для произвольного алгоритма поиска вывода в классической логике предикатов с предикатом равенства. Thinker не обладает свойством семантической полноты, поскольку количество переменных, которые используются в выводе, заранее ограничено.

У. Сиг вместе с Дж. Бернсом [Sieg], [Sieg & Byrnes] предложили программу автоматического поиска натурального вывода CMU PT в классической логике (авторы

также рассматривают возможность обобщения программы на неклассические логики). Специфика данного алгоритма состоит в том, что натуральный вывод строится не *прямым*, а *косвенным* образом. Сначала строится вывод в т.н. *промежуточном* исчислении, а затем показывается, каким образом можно преобразовать вывод в промежуточном исчислении в натуральный вывод. Авторы показывают, что CMU PT обладает свойством семантической полноты.

Д. Ли [Li] предложил программу поиска натурального вывода ANDP в классической логике. Особенно подчеркивая прикладное значение ANDP, Д. Ли дает машинные доказательства некоторых известных проблем математической логики: проблемы остановки машины Тьюринга, проблемы зависимости некоторых аксиом в формализации проективной геометрии и др. Вопрос, обладает ли ANDP свойством семантической полноты, остается открытым.

В.А. Бочаров, А.Е. Болотов и А.Е. Горчаков [Болотов и др.] предложили алгоритм поиска натурального вывода Prover для классической логики предикатов. Спецификой Prover является поиск вывода в натуральных исчислениях типа Куайна с использованием абсолютно и относительно ограниченных переменных. В процессе поиска вывода Prover использует также сколемовские термы. Касаясь вопроса о семантической полноте для Prover, авторы предлагают пути решения данной проблемы. Однако доказательства данного факта для Prover предложено не было.

Группа исследователей под руководством Н.А. Шанина [Шанин и др.] предложила процедуру поиска натурального вывода типа Генцена в классической логике высказываний. Отличительной особенностью данной процедуры является поиск вывода в секвенциальном исчислении. Затем полученный вывод в секвенциальном исчислении перестраивается в натуральный вывод типа Генцена. Отмечая пионерский характер данной работы (она вышла в 1964 году), подчеркнем, что вопрос о семантической полноте процедуры авторами не ставился, поскольку в формулах, для которых требуется найти натуральный вывод, разрешается использовать не более трех пропозициональных переменных. В значительной степени на работы группы под руководством Н.А. Шанина опирается У. Сиг.

Цель и задачи исследования.

Целью диссертационного исследования является пересмотр алгоритма поиска натурального вывода типа Куайна в классической логике предикатов первого порядка,

предложенного В.А. Бочаровым, А.Е. Болотовым и А.Е. Горчаковым, и доказательство для этого алгоритма теорем о семантической непротиворечивости и семантической полноте.

Для достижения данной цели ставятся и решаются следующие задачи:

– Предложить доказательство теоремы о семантической непротиворечивости для натурального исчисления типа Куайна с механизмом использования в выводе абсолютно и относительно ограниченных переменных.

– Представить содержательное описание алгоритма поиска вывода в виде формальных правил поиска вывода.

– Опираясь на вышеупомянутый результат, доказать теорему о семантической непротиворечивости алгоритма поиска вывода в данном исчислении.

– Разработать представление линейного алгоритмического вывода в натуральных исчислениях типа Куайна в виде древовидной структуры, узлами которой являются не формулы вывода, а особенные конечные последовательности формул вывода (*блоки*), переход между которыми осуществляется с помощью формальных правил поиска вывода алгоритма.

– Показать с помощью представления алгоритмического вывода в виде древовидной структуры конечность ветвления в произвольном блоке.

– Обосновать возможность *прямого* (т.е. не с помощью промежуточных исчислений) доказательства теоремы о семантической полноте алгоритма поиска натурального вывода.

Методологические основы и источники исследования.

При решении поставленных задач автор опирался на современный аппарат символической логики. В диссертационной работе использовались формулировки систем натурального вывода, предложенные В.А. Бочаровым, Е.К. Войшвилло, Г. Генценом, Ф. Пеллетье, Дж. Поллоком, В.А. Смирновым и др.

В основе описанного в диссертационном исследовании алгоритма поиска вывода лежит алгоритм поиска вывода в классической логике предикатов, предложенный А.Е. Болотовым, В.А. Бочаровым и А.Е. Горчаковым. В процессе использования написанной этими авторами программы возникла необходимость модифицировать данный алгоритм, определенным образом упростить процедуру поиска натурального вывода и четко сформулировать процедуру унификации.

Научная новизна исследования.

В диссертационном исследовании предложен метод доказательства теоремы о семантической непротиворечивости для натурального исчисления типа Куайна. Показана гибкость данного метода, позволяющая применять его и к другим натуральным исчислениям этого типа, отличительными свойствами которых являются наличие *прямого* правила удаления квантора существования и использование в выводе абсолютно и относительно ограниченных переменных.

В процессе исследования получены следующие новые результаты, выносимые на защиту:

- Предложено оригинальное доказательство теоремы о семантической непротиворечивости для натурального исчисления типа Куайна с абсолютно и относительно ограниченными переменными.
- Модифицирован стандартный алгоритм унификации для временных переменных и сколемовских функций с целью работы с абсолютно и относительно ограниченными переменными.
- Предложено оригинальное представление алгоритмического вывода в виде древовидной структуры (*поисковое дерево*), узлами которого являются непустые, конечные последовательности формул (*блоки*).
- Обоснован *прямой* метод доказательства теоремы о семантической полноте алгоритма поиска натурального вывода. С помощью данного метода предлагается оригинальное доказательство теоремы о семантической полноте для алгоритма поиска натурального вывода в исчислениях типа Куайна.
- Предложено оригинальное доказательство теоремы о семантической полноте для системы натурального вывода типа Куайна, следующее из теоремы о семантической полноте для алгоритма поиска вывода в данной системе.

Практическая значимость. Разработанный алгоритм поиска натурального вывода в исчислениях типа Куайна может служить основой для создания компьютерных реализаций, которые, в свою очередь, могут использоваться в педагогической практике, облегчая усвоение основ дедукции.

Содержание диссертационного исследования может быть использован для разработки специального курса по автоматическому поиску логического вывода.

Апробация работы. Основные положения и результаты диссертационного исследования докладывались на VI и VII Международных научных конференциях «Современная логика: проблемы теории, истории и применения в науке» (Санкт-Петербург, 2000 и 2002), IV Международной конференции «Смирновские чтения» (Москва, 2003) и XII Международном конгрессе по логике, философии и методологии науки (Овьедо, 2003).

Структура диссертации. Диссертация состоит из Введения, 4-х глав: «Автоматический поиск натурального вывода: история вопроса», «Анализ системы натурального вывода BMV», «Алгоритм поиска вывода в системе BMV» и «Анализ алгоритма поиска вывода в системе BMV», Заключения и Литературы.

Глава 1. Автоматический поиск натурального вывода: история вопроса

§ 1.1. Натуральный вывод как тип логического вывода

Моментом появления *натурального вывода* (НВ) как одного из типов логического вывода традиционно считается выход в 1934 году статьи Г. Генцена «Исследования логических выводов» [Генцен] и статьи С. Яськовского «Правила введения посылок в формальной логике» [Jaskowski].

В англоязычной литературе системы НВ называются «natural deduction». В отечественной литературе системы НВ иногда называются «естественным выводом».

Название таких систем указывает на ту черту, которая отличает НВ от других типов логического вывода: исчислений гильбертовского типа (аксиоматик) и исчислений секвенций. Системы НВ создавались с целью (насколько это возможно) имитировать рассуждения, которые характерны для человеческого мышления, решающего (прежде всего) математическую задачу.

Г. Генцен писал: «Я хотел прежде всего построить такой формализм, который был бы как можно ближе к применяющимся в действительности рассуждениям» [Генцен, с. 10]. Далее он определяет свою задачу более точно: «Мы хотим построить формализм, по возможности точно передающий логические заключения, которые в действительности встречаются в математических доказательствах» [Генцен, с. 17].

Созданию С. Яськовским систем НВ способствовало замечание Я. Лукасевича, что «математики строят свои доказательства, используя не аксиоматический метод, а иные способы рассуждения; главным образом, математики берут «произвольные посылки» и смотрят, что из этих посылок можно вывести» [Pelletier, с. 4].

Данная черта отличает НВ как от аксиоматического метода (вывод в гильбертовских исчислениях страдает громоздкостью и определенной искусственностью, не характерной для естественных рассуждений), так и от исчисления секвенций и непосредственно идущих от метода секвенций методов семантических и аналитических таблиц (для последних двух методов характерен поиск опровержения, а не доказательства).

Приведем цитаты, показывающие достоинства НВ перед другими типами логического вывода. «Естественный вывод гораздо более тонкий инструмент, чем семантические таблицы, обладающий большой аналитической силой и легко

модифицируемый» [Непейвода, с. 297]. «Наиболее удобным способом ее (логики предикатов – В.Ш.) задания является система натурального вывода» [Ивлев, с. 95]. «Для применения в качестве логического аппарата наиболее удобными (системами исчисления предикатов – В.Ш.) являются натуральные системы (системы натурального вывода)» [Войшвилло, с. 85].

Размышляя о преимуществах натурального вывода перед методом резолюции, Д. Пеллетье пишет: «Я предположил, что представление вывода в виде натурального на самом деле обладает значительными преимуществами с познавательной точкой зрения: такое максимально обобщенное представление позволяет студентам глубже проникнуть в структуру логических проблем» [Pelletier, с. 6].

Характеризуя гильбертовские исчисления, авторы [Смирнов и др., с. 21] пишут: «Сами способы построения и выводов в этих системах в значительной степени не соответствуют естественным способам рассуждений». Характеризуя секвенциальные и аналитико-табличные исчисления, эти же авторы пишут: «Сам стиль осуществления этих процедур во многом перестает носить «естественный» характер, т.е. перестает соответствовать обычным способам рассуждений» [Смирнов и др., с. 21].

«Логическая система может быть задана различными способами: аксиоматически, в виде табличного исчисления и т.д. Не все из них одинаково удобны для анализа выводов. Самые широкие возможности в этом плане открывают секвенциальные исчисления и системы натурального вывода» [Быстров, с. 139].

«Во всех этих приемах (секвенциальные исчисления и метод резолюции – В.Ш.) 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. И секвенции, и метод резолюции – 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» [Болотов и др., с. 172].

Из перечисленного можно сделать вывод, что НВ обладает определенными преимуществами перед другими типами логического вывода: удобство и простота, конгениальность естественным рассуждениям и др.

Что касается дедуктивной силы НВ, то, представляя НВ, Г. Генцен конструктивно показал, что в классической логике предикатов НВ дедуктивно эквивалентен исчислению секвенций и гильбертовскому исчислению, т.е. вывод некоторой формулы в одном исчислении можно перестроить в вывод этой же формулы в другом исчислении [Генцен].

Также отметим, что в нашем исследовании используются только системы НВ для классической логики. Поэтому рассмотрение НВ для неклассических логик выходит за рамки нашего исследования. Подробнее о системах НВ для неклассических логик см. [Basin et al].

Возможны различные классификации систем НВ [Pelletier], [Смирнов].

Например, Д. Пеллетье выделяет девять основных пунктов («nine choice points»), по которым можно классифицировать системы НВ. В то же время он признает, что «многообразие систем НВ делает весьма затруднительным выделение необходимых и достаточных признаков, которые позволили бы однозначно назвать некоторую логическую систему натуральным выводом» [Pelletier, с. 11].

Поскольку Д. Пеллетье не упоминает системы НВ, предложенные отечественными учеными, мы будем добавлять их в качестве примеров в соответствующие разделы классификации.

Первым основным пунктом является тип представления натурального вывода:

1) в виде дерева (Г. Генцен, Н.А. Шанин, Н.Н. Непейвода);

2) в виде линейного вывода (С. Яськовский, Ф. Фитч, В.А. Бочаров и В.И. Маркин), где подвыводы обозначаются некоторыми графическими объектами (скобками, квадратными скобками, линиями и др.);

3) в виде линейного вывода (С. Яськовский, У. Куайн), где подвыводы обозначаются различными (числовыми) префиксами;

4) в виде линейного вывода с множеством зависимостей (П. Суппес, Е.К. Войшвилло).

Вторым основным пунктом является присутствие (Г. Генцен, У. Куайн, Е.К. Войшвилло) или отсутствие (С. Яськовский) в системе НВ некоторых аксиом наряду с правилами вывода.

Назовем систему НВ *симметричной*, если в ней для любого логического символа (связки или квантора) содержатся хотя бы одно правило введения и хотя бы

одно правило удаления (исключения). Тогда наличие (Г. Генцен, Н.Н. Непейвода, В.А. Бочаров и В.И. Маркин) или отсутствие (Д. Пеллетье, Е.К. Войшвилло, Дж. Поллок, А.М. Анисов) симметричности в системе НВ – это третий основной пункт.

Четвертый основной пункт (для пропозиционального исчисления) – количество непрямых правил вывода в системе НВ, где непрямое правило – это правило, требующее построения подвывода.

Обычным непрямым правилом является $\supset_в$: если из посылки С выводима формула В, значит, можно построить вывод формулы $C \supset B$.

Например, Г. Генцен предложил следующее непрямое правило $\vee_и$: из формулы $A \vee B$, подвывода С из посылки А и подвывода С из посылки В выводима формула С. Однако можно предложить прямое правило $\vee_и$: из $A \vee B$, $A \supset C$, $B \supset C$ выводима формула С, или из $A \vee B$, $\neg A$ выводима формула В.

Д. Пеллетье также отмечает, что прямые и непрямые правила можно вводить для других пропозициональных связок (например, $\neg$, $\equiv$).

Следующие основные пункты 5-9 касаются работы с переменными в системах НВ. Д. Пеллетье отмечает нетривиальность проблемы работы с переменными в системах НВ в сравнении с другими типами логического вывода.

Количество кванторов, которые используются в системах НВ, – это пятый основной пункт. Подавляющее большинство авторов используют и квантор общности, и квантор существования при формулировке своих систем НВ. Исключение составляет, например, С. Яськовский.

Шестой основной пункт – наличие прямого (У. Куайн, В.А. Бочаров и В.И. Маркин) или непрямого (Г. Генцен, Н.Н. Непейвода) правила удаления квантора существования.

Важнейшей характеристикой систем НВ с прямым правилом удаления $\exists$ является тот факт, что в общем случае заключение такого натурального вывода логически не следует из посылок этого вывода.

В таких системах НВ появляется наряду с понятием вывода понятие *завершенного* вывода, т.е. вывода, в котором заключение логически следует из всех неисключенных посылок этого вывода. Далее мы подробнее остановимся именно на данном основном пункте.

В системах НВ формулировка правила $\forall_в$ предполагает наличие произвольной или новой, ранее не встречающейся в выводе переменной. Седьмой основной пункт – это различные способы, которые гарантируют, что переменная в формулировке правила $\forall_в$ является произвольной.

Например, в системе НВ, предложенной Н.Н. Непейводой, переменная считается произвольной только в том подвыводе, в котором к ней применено $\forall_в$, и в этом подвыводе запрещено пользоваться формулами из других подвыводов, в которые данная переменная входит свободно.

В то же время, в системе НВ, предложенной В.А. Бочаровым и В.И. Маркиным, наличие произвольной переменной в формулировке правила $\forall_в$ задается неявным образом.

С одной стороны, данная переменная не обязательно новая, ранее не встречающаяся в выводе, с другой стороны, согласно понятию вывода в этой системе НВ, ни к одной переменной правило $\forall_в$ не может быть применено более одного раза, а значит, формулировка $\forall_в$ предполагает новую переменную.

Восьмым основным пунктом является разделение всех систем НВ на те, которые допускают вхождение свободных переменных в посылки и заключение выводов (У. Куайн, В.А. Бочаров и В.И. Маркин), и на те, которые не допускают посылок и заключений такого вида, т.е. посылки и заключения в таких системах могут быть только предложениями (В.А. Смирнов).

Дополнительно отметим, что возможна условная (В.А. Бочаров и В.И. Маркин) и универсальная (Д. Пеллетье) интерпретация свободных переменных.

Наконец, девятым основным пунктом является наличие (Дж. Поллок, В.А. Смирнов) или отсутствие (У. Куайн, В.А. Бочаров, В.И. Маркин) особых термов в формулировках кванторных правил.

Например, В.А. Смирнов формулирует кванторные правила $\forall_в$, $\exists_и$ с помощью ε-термов, содержательно трактующихся как неопределенные дескрипции.

Особое внимание мы обратим на деление всех систем НВ в зависимости от того, принимается ли в них прямое или непрямое правило удаления (исключения) квантора существования, и отметим классификацию систем НВ, предложенную В.А. Смирновым.

В данной классификации система НВ, названная NC, с непрямым правилом удаления ∃ называется системой НВ *первого* типа. *Второй* тип систем НВ (в качестве примера предлагается система NεC) содержит в множестве своих правил прямое правило удаления ∃. Ниже мы подробно проанализируем систему NεC.

В оригинальной системе, предложенной Г. Генценом, имеется непрямое правило удаления ∃: из формулы ∃xA(x) выводима формула C, если формула C выводима из A(a), 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) выводима формула C.

По-видимому, первым, кто обратил внимание на неудобство применения этого правила и кто предложил альтернативный вариант удаления ∃, был У. Куайн [Quine]. В статье «On natural deduction» он сформулировал прямое правило удаления ∃: из формулы ∃xA(x) выводима формула A(y), где y – переменная, которая в алфавитном порядке не предшествует ни одной переменной, свободно входящей в ∃xA(x).

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. У. Куайн показывает, что именно такой порядок гарантирует непротиворечивость и полноту предложенной им системы НВ.

Однако подход У. Куайна не является единственным. Например, В.А. Смирнов в системе NεC 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 «некоторые x, обладающие свойством A» [Смирнов].

Поскольку ε-термы не являются термами классической логики предикатов, то система NεC не эквивалентна системе NC (системе НВ 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 А и Г, А не содержат ε-термы, то А следует из Г в NC [Смирнов, с. 228], [Мендельсон, с. 111-112].

В.А. Бочаров и В.И. Маркин предлагают вводить абсолютно ограниченные и относительно ограниченные переменных в выводе. Тогда правило удаления ∃ запишется следующим образом: из формулы $\exists x A(x, z_1,\ldots, z_n)$ выводима формула $A(y, z_1,\ldots, z_n)$, при этом переменная y становится абсолютно ограниченной в выводе переменной, а все переменные $z_1,\ldots, z_n$ – относительно ограниченными в выводе переменными ($z_1,\ldots, z_n$ 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 (непосредственно, т.е. y не входит в $z_1,\ldots, z_n$, или по транзитивности, т.е. если x ограничивает y и y ограничивает x, то x ограничивает x). Во второй главе нашей работы показывается, что такой подход гарантирует непротиворечивость этой системы НВ.

Мы подробно останавливаемся на системах НВ с прямым правилом удаления ∃, поскольку система BMV (Bocharov, Markin, Voishvillo), алгоритм поиска вывода в которой является предметом нашей работы, – система НВ именно такого типа.

С другой стороны, мы останавливаемся подробно на работе с переменными в системах НВ с непрямым правилом удаления ∃, т.к. существуют примеры систем НВ, не корректно работающие с переменными.

В англоязычной литературе известна серия публикаций 60-70-х гг. XX века в «The Journal of Symbolic Logic» по поводу системы НВ, предложенной И. Копи (I. Copi). И. Копи публиковал не являющиеся семантически непротиворечивыми системы НВ с прямым правилом удаления ∃ [Copi]. Семантическая противоречивость предлагаемых И. Копи систем НВ была установлена Д. Калишем (D. Kalish) [Kalish].

Не является семантически непротиворечивой система НВ, предложенная Е.К. Войшвилло [Войшвилло]. В этой системе доказывается, например, формула $\exists y(\forall x A(x, x) \supset \forall z A(y, z))$. Данная формула ложна, если в качестве А взять отношение равенства на множестве натуральных чисел.

§ 1.2. История создания систем автоматического поиска вывода

В данном параграфе мы даем краткий обзор истории создания систем автоматического поиска вывода. Поскольку неотъемлемой частью создания таких систем является наличие программируемых электронно-вычислительных устройств, в начале данного параграфа приводятся некоторые факты из истории создания ЭВМ.

Отметим, что системы автоматического поиска натурального вывода в данном параграфе не рассматриваются. Такие системы будут подробно рассмотрены в следующем параграфе данной главы.

Пионером создания вычислительных логических машин (именно логических, т.е. работающих с текстами, а не с цифрами) считается средневековый мыслитель Р. Луллий.

Создавая такую машину, Р. Луллий опирался на средневековое 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 исходных понятий и представлений возникает знание. В обладании этими комбинациями и тем, что из них вытекает, заключается подлинная мудрость. Таким образом, логическая машина должна была служить инструментом для порождения таких комбинаций.

Построенные Р. Луллием многочисленные модели такой машины не заменяли деятельность человека. Человек был необходим для интерпретации понятийных комбинаций и получения, таким образом, окончательного знания. Отметим, что в своих машинах Р. Луллию удалось реализовать одну из важнейших функций вычислительных машин – перебор вариантов.

В 1834 году Ч. Бэббидж сконструировал цифровое счетное устройство. Особенностью этой машины была способность выполнять инструкции, считываемые с перфокарт. В 1842 году А. Лавлейс сделала описание работы аналитической машины Бэббиджа и составила первую программу для нее.

В дальнейшем стали создаваться машины, совершенствующие модель Бэббиджа. В 1855 году Дж. и Э. Шутц, базируясь на работах Ч. Бэббиджа, построили свою механическую вычислительную машину. В 1869 году У.С. Джевонс, используя результаты Дж. Буля, построил усовершенствованную модель этой машины. В 1896

году Г. Холлерит создал первую электромеханическую вычислительную машину и основал фирму, которая впоследствии превратилась в корпорацию IBM.

Промежуток времени между двумя мировыми войнами считается сегодня периодом рождения первого компьютера. В 1927 году в Массачусетском технологическом институте был изобретен аналоговый компьютер. В 1937 году Дж. Стибитц построил первую вычислительную машину на основе двоичной системы счисления.

В 1938-41 гг. К. Цузе предложил несколько моделей (Z1, Z2 и Z3) своей механической программируемой цифровой машины. Модель Z1 иногда называют первым в мире компьютером. В 1942 году в Университете штата Айова Дж. Атанасов и К. Берри создают первый в США электронный цифровой компьютер.

В России созданием «умных» машин занимались П.Д. Хрущов и А.Н. Щукарев. Создание вычислительных машин активно велось в советское время.

Появление первого компьютера открыло путь для развития механизируемых исчислений. В 50-е годы XX века были созданы две компьютерные программы, которые заложили два основных направления в области автоматического поиска доказательства.

Первая программа была создана М. Дэвисом на ламповом компьютере "Johniac". Эта программа могла доказать, 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 А.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.

В 1956 году результаты этой работы были доложены на конференции в Дартмаунте, которую считают местом рождения нового раздела компьютеристики – исследований по искусственному интеллекту. На этой конференции разгорелась дискуссия: необходимо ли пытаться реализовывать на вычислительных машинах процедуры вывода, формулируемые математической логикой, или, следуя А. Невелу, Дж. К. Шоу и Г. Саймону, симулировать человеческие эвристики.

В 1961 году М. Минский подвел итог этой дискуссии: «Кажется ясно, что программа решения реальных математических проблем должна комбинировать математические рассуждения... с эвристическими рассуждениями Невела, Шоу и Саймона».

В 1958-63 гг. Х. Ван предложил несколько версий своей первой логикоориентированной программы фирмы IBM по автоматическому доказательству. Это был большой шаг вперед: его программа могла доказывать около 350 предложений «Principia Mathematica» для чистого исчисления предикатов с равенством.

В 1954 году А. Робинсон предложил рассматривать точки, линии и 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. Патнем. Она состояла из двух частей: первая часть программы генерировала Эрбрановский универсум и делала соответствующие подстановки в формулы, вторая – оценивала эти формулы. Основная проблема, которая возникла перед авторами, заключалась в несистематичности подстановок.

В 1960 году эту проблему решил Д. Правиц посредством механизма, названного унификацией. Позже был предложен унифицирующий алгоритм.

Независимо от этого над доказательствами математических утверждений в национальной лаборатории Аргонны работали Д. Карсон, Дж. Робинсон и Л. Уос. Ученые поставили перед собой задачу разработать такой принцип, в котором бы объединились различные методы в одно общее логическое правило.

В 1963-1965 гг. такое правило было найдено, и Дж. Робинсон публикует свою работу "A machine oriented logic, based on the resolution principle" [Robinson]. 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 направление, самое большое по числу представителей, продолжало отстаивать и совершенствовать метод резолюции, второе направление видело своей задачей имитировать эвристические методы применяемые «человеческими» математиками.

К первому лагерю принадлежали ученые национальной лаборатории Аргонны под руководством Л. Уоса и группа под руководством Б. Мелтцера в университете Эдинбурга. В 1965-75 гг. они предложили десятки усовершенствованных стратегий поиска вывода.

Другим предметом исследований этого направления было само правило резолюции. Вскоре оказалось, что оно не всегда является удобным, и были предложены другие правила вывода, например, гиперрезолюция. Исследовались также возможности представления метода резолюций в виде графоориентированной процедуры. Так, основная структура данных – множество дизъюнктов – пополнялась дополнительной информацией, например, представлением возможных шагов в виде графа.

Следующим результатом этого направления оказался матричный метод, который разрабатывался независимо друг от друга П. Андреусом 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 машинноориентированной. Поиск доказательства протекает в определенной структуре данных, так называемой матрице, и найденное доказательство не имеет ничего общего с обычным пониманием этой процедуры.

Представители второго лагеря критически относились к такой механизации дедуктивных систем, в особенности М. Минский из Массачусетского института технологий. 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 каждого из них в частности. Особенный импульс был придан второму направлению.

В 60-е годы XX века Л.М. Нортон предложил эвристический прувер для теории групп, а А. Невинс – алгоритм, разрабатываемый на базе «человекоориентированной

логики». Результаты данных программ оказались не достижимыми для резолюционных процедур того времени. Здесь можно отметить работу исследователей университета Техаса, развивавших специализированные методы доказательства для конкретных областей математики.

На сегодняшний день наиболее эффективными остаются пока 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, работающие в национальной лаборатории Аргонны, Остине, Карлсруе и Кайзерслаутерна. 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 (лаборатория Аргонны).

Системы автоматического поиска доказательства долгое время считались эзотерической областью знания. Однако в последнее время появился ряд сфер применения результатов, достигнутых в этой области: развитие программного и аппаратного обеспечения; становление дедуктивных баз данных.

Другими областями, диктующими необходимость совершенствования методов автоматического доказательства, являются компьютерная алгебра, а также некоторые аспекты тестирования программ. В этот ряд можно включить экспертные системы, робототехнику и другие направления.

Системы доказательства теорем разбиваются на два больших лагеря. Первый лагерь – это системы интерактивного доказательства теорем (proof checker), другое название этих систем – редакторы доказательств.

Это системы, которые под своим контролем дают возможность пользователю строить доказательство. Они в большинстве своем основываются на теории типов (часто с зависимыми типами) и на системах правил заключения. При этом правила вывода обычно охватывают систему натурального вывода типа Генцена и зависят от желаемого поля применения. Шаги доказательства проверяются системой на

применимость и корректность. Таким образом достигается уверенность, что в системе могут быть построены только корректные доказательства.

Примером 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 AUTOMATH, как правило, в 10-20 раз длиннее, чем в естественном языке.

Из отечественных работ можно привести программу DEDUCTIO, разработанную А.В. Смирновым при участии А.Е. Новодворского [Смирнов-мл]. 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 (например, система NuPRL). Высказывается предположение, что с развитием более сильных тактик возможно достижение высокого уровня автоматизации.

Ко второму лагерю 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 найдено доказательство или не выполнится критерий, обрывающий работу системы. В качестве такого критерия стараются привести построение контрмодели для доказываемого утверждения.

Разработки программ в области автоматической дедукции проводятся в целом ряде крупных мировых научно-исследовательских центров. В национальной лаборатории Аргонны создана программа OTTER, основанная на теории резолюций. OTTER является последним продуктом в цепи систем (AURA, ITP), которые были созданы под руководством Л. Уоса. Программа создана для поиска доказательств для классической логики предикатов первого порядка с равенством, написана на языке C,

занимает примерно 180 Кб и способна оперировать большим количеством дизъюнктов (миллиардами), что показывает ее высокую эффективность.

В Кембридже создана программа автоматической дедукции Isabelle. В Карнеги-Меллон университете создана программа для автоматического доказательства теорем для логики высоких порядков. В лаборатории KAKEHI и Университете WASEDA создана программа автоматического доказательства для секвенциального исчисления

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

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 не случайна и обусловлена, в первую очередь, практическими соображениями. 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 составной частью Пролога [Братко].

Еще одной причиной такой ситуации является факт достаточно успешного практического применения неклассических логик в рамках различных проектов тестирования аппаратных средств [Bochmann], нередко спонсирующихся ведущими производителями таких средств (например, Intel).

На сегодняшний день самым распространенным и эффективным методом в данной области является метод тестирования моделей. В данном методе для описания моделей аппаратных средств используется дедуктивный аппарат, чаще всего основанный на модальной или временной логике, семантика для которых чаще всего строится в форме аналитических таблиц [Bochmann], [Bolotov & Fisher].

§ 1.3. Автоматический поиск вывода в натуральном исчислении

В данном параграфе мы анализируем алгоритмы поиска натурального вывода, выявляем сильные и слабые стороны данных программ.

В силу огромного количества предложенных систем натурального вывода, данная выборка не претендует на полноту и всесторонний охват всех систем поиска натурального вывода (даже в классической логике высказываний). Поэтому отсутствие какой-либо работы по данной теме в нашем обзоре ни в коем случае не характеризует эту работу отрицательным образом.

Анализ работ будет зафиксирован в следующей таблице, в верхней строке которой будут перечислены следующие алгоритмы поиска натурального вывода в классической первопорядковой логике:

1) программа THINKER, разработанная Д. Пеллетье (J. Pelletier) [Pelletier];
2) программа OSCAR, разработанная Дж. Поллоком (J. Pollock) [Pollock];
3) программа ANDP (Automated Natural Deduction Prover), разработанная Д. Ли (D. Li) [Li];
4) программа CMU PT (Carnegie Mellon University Proof Tutor), разработанная У. Сигом (W. Sieg), Р. Шейнсом (R. Scheines) и Дж. Бернсом (J. Byrnes) [Sieg], [Sieg & Byrnes], [Byrnes];
5) программа Symlog (Symbolic Logic), разработанная Ф. Порторарo (F. Portoraro) [Portoraro];
6) программа Prover, разработанная В.А. Бочаровым, А.Е. Болотовым и А.Е. Горчаковым и модифицированная В.О. Шангиным [Болотов и др.], [Болотов и др.1], [Шангин], [Bocharov et al].

В первом столбце таблицы находятся те самые признаки, по отсутствию или наличию которых мы анализируем данные алгоритмы поиска вывода:

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

7) алгоритм имеет программную реализацию.

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

		Thinker	Oscar	ANDP	CMU PT	Symlog	Prover
1	=	+	–	–	–	–	–
2	≡	+	–	+	–	+	–
3	Неклассика	+	+	–	+	–	+
4	Ограничение на переменные	+	–	–	–	–	–
5	Свойства	–	+–	+–	+	–	+
6	Явная сколемизация	–	+	–	+	+	–
7	Реализация	+	+	+	+	+	+–

Комментарий.

Из перечисленных программ только Thinker работает с предикатом равенства. Соответственно, среди предложенных программ только Thinker может построить вывод для всех 75 проблем, предложенных в [Pelletier2], [Pelletier3]. Данный факт неудивителен, поскольку у Thinker и у списка 75 тестовых задач для алгоритма поиска вывода один и тот же автор – Д. Пеллетье.

Около половины программ не работают с формулами, содержащими знак эквивалентности. С другой стороны, отметим тривиальность задачи введения в алгоритм дополнительных правил для работы с такими формулами.

Безусловным преимуществом алгоритма является возможность работы не только в классической, но и в неклассических логиках. Программа Thinker умеет работать в модальной логике [Pelletier]; программа Oscar – в логике «отменяемых» (defeasible) рассуждений; программа ANDP – в интуиционистской логике, и, наконец, можно рассматривать работы [Макаров], [Шангин1], [Шангин3] о поиске вывода в интуиционистской логике высказываний как перенесение идей Prover 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, например, квантор общности снимается бесконечное количество раз.

Строка «Свойства» отмечает разработанность метатеоретических проблем для данных алгоритмов. Например, «+–» 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.

По разным причинам, метатеоретическая проблематика отсутствует для программ Thinker и Symlog. По словам Ф. Порторaро, автора программы Symlog, «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 – предоставлять 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» [Portoraro, с. 59].

Мы не можем согласиться с таким подходом, поскольку предоставлять действительно *всестороннюю помощь при построении вывода* может только алгоритм, для которого доказана теорема о семантической полноте.

Что касается Thinker, то данный алгоритм неполон, если учесть вышеуказанное ограничение на количество используемых в выводе переменных: «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» [Pelletier, 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. также [Sieg], [Sieg & Byrnes], [Byrnes]) заключается в том, что по дереву вывода, которое не является доказательством, можно построить контрмодель для данной формулы или для данной выводимости. Т.е. 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. О различиях в указанных подходах подробно говорится в конце данного параграфа.

Не все алгоритмы используют в своей работе сколемизацию. Под сколемизацией здесь имеется ввиду 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 вывод не обязательно предполагает явную сколемизацию. Существуют различные формы неявной сколемизации. Например, Thinker предполагает в процессе поиска вывода использовать т.н. «шаблоны» (templates), которые при окончательном построении вывода превращаются в формулы. Например, снятие $\forall$ с формулы $\forall xA(x, y)$ 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. Что касается кванторного фрагмента Prover, то он находится в стадии разработки.

Из данного анализа заметно, что проблема исследований метатеоретических свойств алгоритмов поиска натурального вывода или вообще не ставится, или не решается. Фундаментальными в этом отношении являются работы У. Сига. Именно на них во многом базируется наше исследование.

Однако здесь мы отметим те существенные различия, которые лежат между подходом У. Сига и нашим подходом.

Во-первых, в диссертационном исследовании исследуется система НВ с *прямым* правилом удаления $\exists$ (в таких системах НВ заключение в общем случае не является логическим следствием из множества всех неисключенных посылок.) У. Сиг, в свою очередь, работает с системой НВ с *непрямым* правилом удаления $\exists$.

Во-вторых, в диссертационном исследовании поиск вывода осуществляется непосредственно в системе НВ, а предложенный У. Сигом алгоритм CMU PT использует для поиска вывода промежуточные, вставочные исчисления (intercalation calculi). По внешнему виду такие исчисления напоминают секвенциальные исчисления. Вывод в таких исчислениях затем перестраивается в натуральный вывод.

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 [Andrews]. Т.е. натуральный вывод в CMU PT строится не в системе натурального вывода, а с помощью промежуточного исчисления.

Что касается Prover, то программа строит натуральный вывод именно в системе натурального вывода. Таким образом, наша работа заполняет своеобразную лакуну, которая образовалась в области автоматического поиска НВ.

Глава 2. Анализ системы натурального вывода BMV

§ 2.1. 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 Φ, Φ_1, Φ_2,… обозначают произвольные функторы и буквами Π, Π_1, Π_2,… обозначаются произвольные предикаторы.

Определение 2.1.1. *Термом* является объект, обладающий следующими свойствами:

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

Определение 2.1.2. *Формулой* является объект, обладающий следующими свойствами:

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

Формулы, определяемые пунктом 1, называются *элементарными формулами*. Все остальные формулы называются *сложными* формулами. Буквами A, B, C,… (с индексами и без них) обозначаются произвольные формулы.

Зададим несколько стандартных определений, необходимых для формулировки правил исчисления.

В выражениях $\forall\alpha A$ и $\exists\alpha A$ формула A называется *областью действия* кванторов $\forall$ и $\exists$, взятых по индивидной переменной 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 A, если она имеет хотя бы одно свободное (связанное) вхождение в формуле A.

Вхождение квантора в выражениях $\forall\alpha A$ и $\exists\alpha A$ называется *вырожденным*, если ни одно вхождение переменной α не находится в A. Например, вхождение $\forall$ в формулу $\forall x\exists y P(y)$ вырождено. Если некоторое вхождение переменной α в A связано двумя или более вхождениями кванторов по α, то все эти вхождения кванторов по α, кроме самого правого, являются вырожденными [Непейвода, с. 184]. Например, вхождение $\forall$ в формулу $\forall x\exists x P(x)$ вырождено.

Пусть t – терм, α – индивидная переменная. Тогда выражение вида $A(\alpha/t)$ обозначает результат подстановки в формулу $A(\alpha)$ вместо всех свободных вхождений α терма t. Подстановка $A(\alpha/t)$ считается *правильной*, если никакое свободное вхождение α не лежит в области действия никакого квантора по каждой переменной, входящей в терм t.

Частным случаем выражения $A(\alpha/t)$ является выражение $A(\alpha/\beta)$, обозначающее правильную подстановку в A переменной β вместо переменной α. При этом фиксируется множество всех свободных переменных $\gamma_1,\ldots,\gamma_n$ в A и $\beta \notin \{\gamma_1,\ldots,\gamma_n\}$. Данное выражение обозначается как $A(\alpha/\beta, \gamma_1,\ldots,\gamma_n)$.

В нижеследующих правилах вывода формула(ы) сверху является необходимым условием применения соответствующего правила (*посылками*), а формула(ы) снизу – *заключением* этого правила. Правила введения и исключения пропозициональных связок называются *пропозициональными* правилами; правила

введения и исключения кванторов называются *кванторными* правилами. Полужирным начертанием в формулировке некоторых правил выделена *большая* посылка.

Задаются следующие правила системы BMV (BMV-правила):

$$\&_{в}\ \frac{A,\ B}{A\ \&\ B} \qquad \&_{и}\ \frac{A\ \&\ B}{A} \qquad \frac{A\ \&\ B}{B}$$

$$\vee_{в}\ \frac{A}{A \vee B} \qquad \frac{B}{A \vee B} \qquad \vee_{и}\ \frac{\mathbf{A \vee B},\ \neg A}{B}$$

$$\supset_{в}\ \frac{B}{C \supset B} \qquad \supset_{и}\ \frac{\mathbf{A \supset B},\ A}{B}$$

$$\neg_{в}\ \frac{B,\ \neg B}{\neg C} \qquad \neg_{и}\ \frac{\neg\neg A}{A}$$

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

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

В правилах $\supset_{в}$ и $\neg_{в}$ формула C является *последней неисключенной* посылкой (это понятие будет разъяснено ниже). $A(\alpha/t)$ и $A(\alpha/\beta)$ – результаты правильной подстановки.

В правилах $\forall_{в}$ и $\exists_{и}$ запись «β – абс.; $\gamma_1,\ldots,\ \gamma_n$ – огр.» означает, что β абсолютно ограничена в выводе и переменная β относительно ограничивает в выводе каждую переменную из множества $\gamma_1,\ldots,\ \gamma_n$, где $\{\gamma_1,\ldots,\ \gamma_n\}$ – это множество всех свободных переменных, входящих в посылки данных правил.

В дальнейшем мы иногда вместо «β относительно ограничена в выводе» будем писать «β ограничена», не указывая на ту переменную, которая относительно ограничивает β, 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. [Бочаров и Маркин, с. 133-134].

Определение 2.1.3. *Выводом* в системе BMV (BMV-выводом) называется непустая конечная последовательность формул C_1, C_2,…, C_n ($n \geq 1$), удовлетворяющая следующим условиям:

1. каждая формула есть либо посылка, либо получена из предыдущих по одному из BMV-правил;
2. при применении правил $\supset_в$ и $\neg_в$ все формулы, начиная с последней неисключенной посылки C_i ($1 \geq i \geq n$) и вплоть до результата применения данного правила $C_i \supset C_j$ ($1 \geq j \geq n$) или $\neg C_i$, исключаются из дальнейших шагов вывода в том смысле, что к ним запрещено применять правила вывода;
3. ни одна индивидная переменная в выводе не ограничивается абсолютно более одного раза;[1]
4. ни одна переменная не ограничивает в выводе сама себя.

Выводом формулы В из непустого множества посылок Г называется BMV-вывод, в котором множество неисключенных посылок есть Г и последняя формула есть формула В.

Доказательством формулы В называется BMV-вывод формулы В из пустого множества неисключенных посылок.

Теоремой называется формула, для которой имеется доказательство.

Определение 2.1.4. *Завершенным* BMV-выводом называется BMV-вывод, в котором никакая переменная, абсолютно ограниченная в выводе, не встречается свободно ни в неисключенных посылках, ни в заключении. *Завершенное* доказательство есть завершенный BMV-вывод из пустого множества неисключенных посылок.

Проанализируем правила системы.

Правила для конъюнкции, а также правило введения дизъюнкции, обычны для систем натурального вывода: их можно обнаружить в [Генцен].

Что касается правила для исключения дизъюнкции, то иногда его формулируют в виде правила *разбора случаев* (р.с.): из $A \vee B$, $A \supset C$, $B \supset C$ следует C. Известно, что р.с. сильнее, чем $\vee_и$ в том смысле, что, например, в интуиционистской логике можно

показать с помощью р.с. производность $\vee_{и}$, но невозможно показать с помощью $\vee_{и}$ производность р.с.. Однако существуют определенные трудности с построением процедуры поиска натурального вывода типа Куайна с правилом р.с. [Макаров].

Что касается правила введения импликации, то специфика этого правила в том, что формула С является *последней неисключенной посылкой*. Данное требование необходимо для корректности системы. При применении этого правила выводима формула $C \supset B$ и из вывода *исключаются* все формулы, начиная с формулы С и заканчивая формулой, непосредственно предшествующей формуле $C \supset B$. Формула *исключается* из вывода не в том смысле, что она стирается и удаляется, а в том смысле, что к данной формуле не может быть применено ни одно правило вывода.

Наличие правила $\supset_{и}$ – чуть ли не обязательное свойство всех систем натурального вывода.

Правило $\neg_{и}$ (снятие двойного отрицания) стандартно.

Что касается правила введения отрицания, то иногда используют два правила: из B, $\neg$B следует $\perp$ и из $\perp$ следует C, где $\perp$ – знак противоречия (тождественно ложной формулы) и C – произвольная формула. В нашей системе формула C не произвольна: она является *последней неисключенной посылкой*, что, опять же, обусловлено требованием корректности системы. Применение этого правила также влечет *исключение* из вывода всех формул, начиная с формулы С и заканчивая формулой, непосредственно предшествующей формуле $\neg$C.

Кванторные правила исключения квантора общности и введения квантора существования обычны. Специфика нашей системы – в формулировке правила введения квантора общности и правила исключения квантора существования.

Пусть $\exists y P(y)$ – формула, причем P – одноместный предикатор. Можно, например, не нарушая требования правильной подстановки, получить из $\exists y P(y)$ по $\exists_{и}$ формулу P(x), затем применить $\forall_{в}$ и получить $\forall y P(y)$. Из $\forall y P(y)$ с помощью правил $\forall_{и}$, $\exists_{в}$ можно снова получить $\exists y P(y)$ и т.д.

Отметим, что в системе BMV такие бесконечные преобразования формул ограничиваются понятием вывода. Однако нам важен тот факт, что такие

---

[1] Иногда пишут «не может абсолютно ограничиваться *дважды*». При этом Е.К. Войшвилло отмечал, что данное требование можно нарушить без ущерба для корректности, «когда правило $\forall_{в}$ или $\exists_{и}$ вторично применяется для вывода из той же формулы» [Войшвилло, с. 90-91].

преобразования возможны, если не следовать понятию вывода, а следовать только формулировкам $\forall_в$, $\exists_и$ и требованию правильной подстановки.

Иногда формулировки данных правил не позволяют такие взаимопревращения формул [Войшвилло]. Более того, Дж. Пеллетье вообще не формулирует правила введения квантора общности явным образом. Аналог этого правила имплицитно содержится в формулировке достижимости некоторого подвывода [Pelletier].

Необходимость учета абсолютных и ограниченных переменных обусловлена, как мы далее увидим, требованием, чтобы ни одна переменная не ограничивала в выводе сама себя.

У. Куайн, например, явно указывает на наличие абсолютно ограниченных (flagged) переменных и не указывает явно на наличие относительно ограниченных переменных. Однако он задает свободные переменные в алфавитном порядке и требует, чтобы при применении правил $\exists_и$, $\forall_в$ новая свободная переменная не предшествовала в алфавитном порядке имеющимся в формуле свободным переменным [Quine]. Именно таким образом У. Куайн избегает самоограничения переменной.

Касаясь правил вывода нашего исчисления в целом, отметим их *симметричность*: каждой логической связке соответствуют *хотя бы* одно правило исключения и *хотя бы* одно правило введения. Свойство симметричности правил натурального вывода идет от работ Г. Генцена [Генцен]. См. также [Fitch]. В то же время симметричность правил не является необходимым свойством натурального вывода. Можно указать на системы натурального вывода, где данное свойство не выполняется [Pelletier].

Проанализируем понятие вывода в системе.

Требование, чтобы в правилах $\supset_в$, $\neg_в$ формула C была последней неисключенной посылкой, обосновывается следующим образом:

⎡	1	A	– пос.
⎢	2	B	– пос.
⎢		….	
⎣	n	C	– BMV-правило
	n+1	$A \supset C$	$– \supset_в$: n

В данном выводе мы от вывода формулы C из посылок A, B по правилу $\supset_в$ перешли к доказательству формулы $A \supset C$. Однако такой переход семантически неверен, т.е. из A, B |= C не следует |= $A \supset C$.

Аналогично обосновывается данное требование в формулировке правила $\neg_в$.

Обоснование требования не ограничивать абсолютно некоторую переменную более одного раза, а также требований наличия завершенного вывода, можно найти в [Quine, стр. 97-99].

Что касается запрета на самоограничение переменной, которого нет в [Quine], то он обосновывается, например, невозможностью построения завершенного вывода $\exists x \forall y A(x, y)$ из $\forall y \exists x A(x, y)$ [Бочаров и Маркин, с. 136].

Необходимость введения понятия завершенного вывода (доказательства), наряду с понятием вывода (доказательства), является характерным свойством выводов в первопорядковых натуральных исчислениях типа Куайна, где в общем случае заключение не следует из своих посылок: «При незавершенности вывода не обеспечена истинность заключения при истинности посылок» [Войшвилло, с. 94].

Натуральные исчисления типа Куайна [Quine], [Войшвилло], [Бочаров и Маркин], [Ивлев] противопоставляются натуральным исчислениям типа Генцена [Генцен], [Fitch], [Li], не требующим введения понятия завершенного вывода.

В качестве примеров приведем завершенный BMV-вывод $\neg\exists x P(x) \vdash \forall x \neg P(x)$.

	1	$\neg\exists x P(x)$		
⌈	2	$P(x)$	– пос.	
⌊	3	$\exists x P(x)$	– $\exists_в$: 2	
	4	$\neg P(x)$	– $\neg_в$: 1, 3	
	5	$\forall x \neg P(x)$	– $\forall_в$: 4	x – абс. огр

Комментарий.

Помещаем формулу $\neg\exists x P(x)$ в качестве посылки. Берем другую посылку $P(x)$. Применяя к этой формуле правило $\exists_в$, выводим формулу $\exists x P(x)$. Поскольку формула $\neg\exists x P(x)$ (шаг 1) противоречит формуле $\exists x P(x)$ (шаг 3), на шаге 4 выводим формулу $\neg P(x)$, применяя правило $\neg_в$. Формула $\neg P(x)$ является отрицанием последней неисключенной посылки $P(x)$. При этом все формулы, начиная с формулы $P(x)$ и заканчивая формулой $\exists x P(x)$, исключаются из дальнейших шагов вывода. Применяя к формуле $\neg P(x)$ правило $\forall_в$, выводим формулу $\forall x \neg P(x)$. При этом переменная x является абсолютно ограниченной в выводе.

Данная последовательность формул является *завершенным* выводом формулы $\forall x\neg P(x)$ из посылки $\neg\exists xP(x)$, поскольку она является выводом формулы $\forall x\neg P(x)$ из посылки $\neg\exists xP(x)$ и абсолютно ограниченная переменная x не входит свободно ни в $\forall x\neg P(x)$, ни в $\neg\exists xP(x)$.

§ 2.2. Семантическая непротиворечивость системы BMV

Определение 2.2.1. Переменная α называется *активной (пассивной)* в BMV-выводе, если α абсолютно ограничена в этом выводе и (не) существует абсолютно ограниченная в этом выводе переменная β, которую относительно ограничивает переменная α.

При применении правила $\forall_в$ к формуле $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ выводится формула $\forall\alpha A(\alpha)$, переменная β становится абсолютно ограниченной и β относительно ограничивает все переменные из $\gamma_1,\ldots, \gamma_n$.

При применении правила $\exists_и$ к формуле $\exists\alpha A(\alpha)$ выводится формула $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$, переменная β становится абсолютно ограниченной и β относительно ограничивает все переменные из $\gamma_1,\ldots, \gamma_n$.

Построим следующий завершенный вывод $\exists x\forall yA(x, y) \vdash \exists x\forall zA(x, z)$.

1. $\exists x\forall yA(x, y)$	– пос.
2. $\forall yA(x, y)$	– $\exists_и$: 1; x – абс.
3. $A(x, y)$	– $\forall_и$: 2
4. $\forall zA(x, z)$	– $\forall_в$: 3; y – абс., x – отн.
5. $\exists x\forall zA(x, z)$	– $\exists_в$: 4

В этом выводе переменная x пассивна и переменная y активна, поскольку абсолютно ограниченная переменная y относительно ограничивает другую абсолютно ограниченную переменную x на шаге 4.

Лемма 2.2.2. В произвольном BMV-выводе D с непустым множеством M всех абсолютно ограниченных переменных в D найдется переменная α: $\alpha \in M$ и α не ограничивает относительно в D ни одну переменную из M.[2]

Доказательство: от противного.

Допустим, что в D каждая переменная из M ограничивает относительно некоторую переменную из M.

[2] Техникой доказательства этой леммы мы обязаны В.М. Попову.

Для всякой переменной α, пусть $[\alpha]$ – это множество всех переменных из M, каждую из которых α ограничивает относительно в D, и пусть для всякого непустого множества X переменных $(X)_{min}$ есть переменная из X, которая нестрого предшествует в алфавитном порядке переменных любой переменной из X.

Поскольку произвольный BMV-вывод конечен (определение 2.1.3), найдется такое натуральное число n: M есть n-элементное множество. Пусть для определенности n_0 есть натуральное число и M есть n_0-элементное множество.

Рассмотрим последовательность множеств M1, M2, …, Mn_0, Mn_0+1, определяемую следующим образом: $M_1 = [(M)_{min}]$, $Mk+1 = [(Mk)_{min}]$, $1 < k \leq n_0$.

Легко видеть, что все множества M1, M2, …, Mn_0, Mn_0+1 непусты, а $(M1)_{min}$, $(M2)_{min}$, …, $(Mn_0)_{min}$, $(Mn_0+1)_{min}$ – это переменные, принадлежащие множеству M.

Ясно, что для любого i и для любого j, $1 \leq i < j \leq n_0 + 1$, переменная $(Mn_i)_{min}$ ограничивает относительно в D переменную $(Mn_j)_{min}$.

Отсюда из антирефлексивности и транзитивности отношения «переменная … ограничивает относительно в D переменную …» получаем, что $(M1)_{min}$, $(M2)_{min}$, …, $(Mn_0)_{min}$, $(Mn_0+1)_{min}$ попарно различны.[3]

Но все они принадлежат множеству M. Следовательно, множеству M принадлежит не менее n_0+1 элементов, что противоречит тому, что M есть n_0-элементное множество.

Значит, допущение неверно, и в произвольном BMV-выводе D некоторая переменная из непустого множества M абсолютно ограниченных переменных в D не ограничивает относительно ни одну переменную из M.

Доказано.

Лемма 2.2.2 доказана для произвольного BMV-вывода. Поскольку всякий завершенный BMV-вывод есть BMV-вывод, лемма 2.2.2 верна для произвольного завершенного BMV-вывода с непустым множеством абсолютных переменных.

По определению вывода в системе BMV, вывод – это непустая конечная последовательность формул, каждая из которых является либо посылкой, либо получена из предыдущих формул по одному из правил вывода системы BMV.

[3] Подробный логический анализ данного отношения проводится в параграфе 3.1 нашего исследования.

Таким образом, важным параметром любого завершенного вывода является число применений в нем правил вывода. Будем обозначать это число посредством q. Из определения вывода следует, что $q \geq 0$.

Специфическим свойством вывода в данном исчислении является наличие в выводе (возможно, пустого) множества абсолютно ограниченных переменных. Будем называть *рангом* вывода число абсолютно ограниченных переменных данного вывода и обозначать ранг вывода посредством r. Из определения вывода следует, что $r \geq 0$.

Пусть Г – произвольное непустое множество формул языка логики предикатов, тогда из Г логически следует формула В (обозначается записью «Г |= В»), если и только если не существует модели и приписывания значений предметным переменным, при которых каждая формула из Г принимает значение «истина», а формула В – значение «ложь».

Лемма 2.2.3. В произвольном завершенном выводе, число применений правил вывода в котором есть q и ранг которого есть r, последняя формула этого вывода семантически следует из множества всех неисключенных в этом выводе посылок.

Доказательство.

Данная лемма будет доказана методом двойной математической индукции по числу применений правил вывода и по рангу произвольного завершенного вывода. В обосновании данного метода мы следуем [Смирнов, с. 122-123].

Запишем сокращенно данную лемму в виде $\forall q \forall r Л(q, r)$.

Для доказательства этой леммы достаточно доказать три утверждения:

(А) $Л(0, 0)$

(Б) $\forall m(m \leq n \supset \forall r Л(m, r)) \supset Л(n+1, 0)$

(С) $\forall r_1(r_1 \leq r \supset Л(n, r_1)) \supset Л(n, r+1)$

Принцип полной математической индукции имеет вид

$\{A(0) \& \forall n[\forall m(m \leq n \supset A(m)) \supset A(n+1)]\} \supset \forall n A(n)$

Пусть A(n) равняется $\forall r Л(n, r)$, тогда

(I) $\{\forall r Л(0, r) \& \forall n[\forall m(m \leq n \supset \forall r Л(m, r)) \supset \forall r Л(n+1, r)]\} \supset \forall n \forall r Л(n, r)$

Пусть A(r) равняется Л(n, r), тогда

(II) $\{Л(n, 0) \& \forall r[\forall r_1(r_1 \leq r \supset Л(n, r_1)) \supset Л(n, r+1)]\} \supset \forall r Л(n, r)$

покажем, что из (А), (Б), (С), (I) и (II) следует $\forall q \forall r Л(q, r)$.

1. $\{Л(0, 0) \& \forall r[\forall r_1(r_1 \le r \supset Л(0, r_1)) \supset Л(0, r+1)]\} \supset \forall r Л(0, r)$	(II), n = 0
2. $Л(0, 0)$	(A)
3. $\forall r_1(r_1 \le r \supset Л(0, r_1)) \supset Л(0, r+1)$	(C), n = 0
4. $\forall r(\forall r_1(r_1 \le r \supset Л(0, r_1)) \supset Л(0, r+1))$	$\forall_в$: 3
5. $Л(0, 0) \& \forall r(\forall r_1(r_1 \le r \supset Л(0, r_1)) \supset Л(0, r+1))$	$\&_в$: 2, 4
6. $\forall r Л(0, r)$	$\supset_и$: 1, 5
7. $\forall m(m \le n \supset \forall r Л(m, r)) \supset Л(n+1, 0)$	(Б)
8. $\{Л(n+1, 0) \& \forall r[\forall r_1(r_1 \le r \supset Л(n+1, r_1)) \supset Л(n+1, r+1)]\} \supset \forall r Л(n+1, r)$	(II), n = n+1
9. $\forall r_1(r_1 \le r \supset Л(n+1, r_1)) \supset Л(n+1, r+1)$	(C), n = n+1
10. $\forall r(\forall r_1(r_1 \le r \supset Л(n+1, r_1)) \supset Л(n+1, r+1))$	$\forall_в$: 9
11. $\forall m(m \le n \supset \forall r Л(m, r)) \supset \forall r Л(n+1, r)$	$K \supset M, L, (M \& L) \supset D \vdash K \supset D$: 7, 8, 10
12. $\forall n(\forall m(m \le n \supset \forall r Л(m, r)) \supset \forall r Л(n+1, r))$	$\forall_в$: 11
13. $\{\forall r Л(0, r) \& \forall n[\forall m(m \le n \supset \forall r Л(m, r)) \supset \forall r Л(n+1, r)]\} \supset \forall q \forall r Л(q, r)$	(I)
14. $\forall r Л(0, r) \& \forall n(\forall m(m \le n \supset \forall r Л(m, r)) \supset \forall r Л(n+1, r))$	$\&_в$: 6, 12
15. $\forall q \forall r Л(q, r)$	$\supset_и$: 13, 14

Поэтому доказательство леммы сводится к доказательству утверждений (А), (Б) и (С).

(А) Л(0, 0): лемма верна для всех завершенных выводов, ранг и число применений правил вывода в которых равны 0.

Пусть последовательность формул $C_1, C_2, \ldots, C_t$ ($t \ge 1$) есть завершенный вывод, ранг и число применений правил вывода в котором равны 0. Для доказательства Л(0, 0) достаточно показать, что формула C_t семантически следует из множества всех неисключенных посылок в выводе $C_1, C_2, \ldots, C_t$.

По определению семантического следования, получаем, что $\{C_1, C_2, \ldots, C_t\} \models C_t$. Но множество $\{C_1, C_2, \ldots, C_t\}$ есть множество всех неисключенных посылок в выводе $C_1, C_2, \ldots, C_t$. Поэтому C_t семантически следует из множества всех неисключенных посылок в выводе $C_1, C_2, \ldots, C_t$.

Таким образом, имеет место Л(0, 0).

(Б) лемма верна для всех завершенных выводов, ранг которых равен 0 и число применений правил вывода в которых равно n+1, в предположении, что лемма верна для всех завершенных выводов любого ранга, число применений правил вывода в которых меньше или равно n.

Ранг вывода равен 0, значит, в выводе не применялись правила $\forall_в$, $\exists_и$.

Таким образом, (Б) распадается на 10 случаев по количеству BMV-правил (все BMV-правила, кроме $\forall_в$, $\exists_и$).

– Последняя формула завершенного вывода получена по правилу $\&_и$.

Тогда вывод имеет следующий вид: $C_1, C_2,\ldots, C_j,\ldots, C_t$, где $t > 0$, $t > j$, C_j есть $A \& C_t$ и C_t есть B.

Вывод $C_1, C_2,\ldots, C_j$ является завершенным, т.к. в выводе $C_1, C_2,\ldots, C_j,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

По индуктивному предположению, $\Gamma_1 \models C_j$, где Γ_1 ($\Gamma_1 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_j$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_j,\ldots, C_t$. Значит, $\Gamma_1 \models A \& B$.

Учитывая $\Gamma_1 \models A \& B$ и $A \& B \models B$, получаем $\Gamma_1 \models B$. Отсюда, используя $\Gamma_1 \subseteq \Gamma$, получаем $\Gamma \models B$.

Подслучай, когда C_j есть $C_t \& A$, рассматривается симметрично.

Аналогичным образом рассматриваются еще 4 случая однопосылочных правил: $\vee_в$, $\neg_и$, $\forall_и$ и $\exists_в$.

– Последняя формула завершенного вывода получена по правилу $\&_в$.

Тогда вывод имеет следующий вид: $C_1, C_2,\ldots, C_j,\ldots, C_i,\ldots, C_t$, где $t > 1$, $t > j$, $t > i$, C_j есть A, C_i есть B и C_t есть $A \& B$.

Вывод $C_1, C_2,\ldots, C_j$ является завершенным, т.к. в выводе $C_1, C_2,\ldots, C_j,\ldots, C_i,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

Вывод $C_1, C_2,\ldots, C_i$ является завершенным, т.к. в выводе $C_1, C_2,\ldots, C_j,\ldots, C_i,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

По индуктивному предположению, $\Gamma_1 \models C_j$, где Γ_1 ($\Gamma_1 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_j$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_j,\ldots, C_i,\ldots, C_t$. Значит, $\Gamma_1 \models A$.

По индуктивному предположению, $\Gamma_2 \models C_i$, где Γ_2 ($\Gamma_2 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_i$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_j,\ldots, C_i,\ldots, C_t$. Значит, $\Gamma_2 \models B$.

Учитывая $\Gamma_1 \models A$, $\Gamma_2 \models B$ и $A, B \models A \& B$, получаем $\Gamma_1, \Gamma_2 \models A \& B$. Отсюда, используя $\Gamma_1 \subseteq \Gamma$ и $\Gamma_2 \subseteq \Gamma$, получаем $\Gamma \models A \& B$.

Аналогичным образом рассматриваются еще 2 случая однопосылочных правил: $\vee_и$ и $\supset_и$.

– Последняя формула завершенного вывода получена по правилу $\neg_в$.

Тогда вывод имеет следующий вид: $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_i,\ldots, C_t$, где $t > 2$, $t > j$, $t > i$, $t > h$, C_j есть D, C_i есть $\neg D$ и C_t есть $\neg C_h$, где C_h есть последняя неисключенная посылка в выводе.

Вывод $C_1, C_2,\ldots, C_j$ является завершенным, т.к. в $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_i,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

Вывод $C_1, C_2,\ldots, C_i$ является завершенным, т.к. в $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_i,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

По индуктивному предположению, $\Gamma_1 \models C_j$, где Γ_1 ($\Gamma_1 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_j$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_i,\ldots, C_t$. Значит, $\Gamma_1 \models D$. Отсюда $\Gamma_1, C_h \models D$.

По индуктивному предположению, $\Gamma_2 \models C_i$, где Γ_2 ($\Gamma_2 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_i$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_i,\ldots, C_t$. Значит, $\Gamma_2 \models \neg D$. Отсюда $\Gamma_2, C_h \models \neg D$.

Учитывая $\Gamma_1, C_h \models D$ и $\Gamma_2, C_h \models \neg D$, получаем $\Gamma_1, \Gamma_2 \models \neg C_h$. Отсюда, используя $\Gamma_1 \subseteq \Gamma$ и $\Gamma_2 \subseteq \Gamma$, получаем $\Gamma \models C_t$.

– Последняя формула завершенного вывода получена по правилу $\supset_в$.

Тогда вывод имеет следующий вид: $C_1, C_2,\ldots, C_h,\ldots, C_j,\ldots, C_t$, где $t > 2$, $t > h$, $t > j$, C_t есть $C_h \supset C_j$, где C_h есть последняя неисключенная посылка в выводе.

Вывод $C_1, C_2,\ldots, C_j$ является завершенным, т.к. в выводе $C_1, C_2,\ldots, C_j,\ldots, C_t$ не применялись правила $\forall_в$, $\exists_и$.

По индуктивному предположению, $\Gamma_1 \models C_j$, где Γ_1 ($\Gamma_1 \subseteq \Gamma$) – множество неисключенных посылок из $C_1, C_2,\ldots, C_j$ и Γ – множество неисключенных посылок из $C_1, C_2,\ldots, C_j,\ldots, C_t$. Значит, $\Gamma_1 \models C_j$. Отсюда $\Gamma_1, C_h \models C_j$.

Из $\Gamma_1, C_h \models C_j$ получаем $\Gamma_1 \models C_h \supset C_j$. Отсюда, используя $\Gamma_1 \subseteq \Gamma$, получаем $\Gamma \models C_h \supset C_j$.

Таким образом, имеет место $\forall m(m \le n \supset \forall r Л(m, r)) \supset Л(n+1, 0)$.

(С) лемма верна для всех завершенных выводов, ранг которых равен r+1 и число применений правил вывода в которых равно n, в предположении, что лемма

верна для всех завершенных выводов, ранг которых меньше или равен r и число применений правил вывода в которых равно n.

Т.е. докажем $\forall r Л(n, r+1)$ в предположении, что $\forall r_1(r_1 \leq r \supset Л(n, r_1))$.

Пусть имеется завершенный вывод Д с n+1 количеством абсолютно ограниченных переменных и $\{\alpha_1, \alpha_2, \ldots, \alpha_{n+1}\}$ – множество ($n \geq 1$) абсолютно ограниченных переменных в этом выводе.

По лемме 2.2.2, среди $\{\alpha_1, \alpha_2, \ldots, \alpha_{n+1}\}$ найдется пассивная переменная α_i, $1 \leq i \leq n+1$. Если в Д несколько пассивных переменных, то мы выбираем первую (сверху вниз) из них.

Тогда (С) разбивается на 2 случая:

С1. Пассивная переменная α_i абсолютно ограничена по правилу $\forall_в$.

С2. Пассивная переменная α_i абсолютно ограничена по правилу $\exists_и$.

Тогда вывод Д может принять один из двух видов:

Г				Г	
…				…	
h. $C(\alpha_i)$			или	h. $\exists\beta C(\beta)$	
…				…	
k. $\forall\beta C(\beta)$	$-\forall_в$: h; α_i – абс. огр.			k. $C(\alpha_i)$	$-\exists_и$: h; α_i – абс. огр.
…				…	
В				В	

Построим следующий вывод Д*, который будет отличаться от предыдущего только наличием посылки 0 вида $К \supset М$, где К – формула, находящаяся на шаге h в выводе Д, и М – формула, находящаяся на шаге k в выводе Д.

Тогда вывод Д* может принять один из двух видов:

0. $C(\alpha_i) \supset \forall\beta C(\beta)$	– пос.			0. $\exists\beta C(\beta) \supset C(\alpha_i)$	– пос.
Г				Г	
…				…	
h. $C(\alpha_i)$			или	h. $\exists\beta C(\beta)$	
…				…	
k. $\forall\beta C(\beta)$	$-\supset_и$: 0, h			k. $C(\alpha_i)$	$-\supset_и$: 0, h
…				…	
В				В	

В Д* α_i не является абсолютно ограниченной переменной, т.к. формула $\forall\beta C(\beta)$ или $C(\alpha_i)$, которая находится на шаге k, получена по правилу $\supset_и$.

Д* является завершенным, т.к.

- ни одна переменная из $\{\alpha_1, \alpha_2, \ldots, \alpha_{n+1}\}$ не входит свободно ни в Г, ни в формулу В (поскольку Д – это завершенный вывод);
- из $\{\alpha_1, \alpha_2, \ldots, \alpha_{n+1}\}$ только α_i входит свободно в посылку, находящуюся на шаге 0 (поскольку α_i пассивна).

Поскольку Д* завершен, в Д* n абсолютно ограниченных переменных (α_i не является абсолютно ограниченной переменной в Д*) и число применений правил вывода в Д* равно числу применений правил вывода в Д, по индуктивному предположению, 0, $\Gamma \models B$, т.е. $C(\alpha_i) \supset \forall\beta C(\beta), \Gamma \models B$ или $\exists\beta C(\beta) \supset C(\alpha_i), \Gamma \models B$.

Из $\Gamma, C(\alpha_i) \supset \forall\beta C(\beta) \models B$ следует, что $\Gamma, \exists\alpha_i(C(\alpha_i) \supset \forall\beta C(\beta)) \models B$, или

Из $\Gamma, \exists\beta C(\beta) \supset C(\alpha_i) \models B$ следует, что $\Gamma, \exists\alpha_i(\exists\beta C(\beta) \supset C(\alpha_i)) \models B$.

Согласно формулировке правил $\forall_в$, $\exists_и$, переменная α_i не входит свободно ни в $\forall\beta C(\beta)$, ни в $\exists\beta C(\beta)$, значит, $\models \exists\alpha_i(C(\alpha_i) \supset \forall\beta C(\beta))$ или $\models \exists\alpha_i(\exists\beta C(\beta) \supset C(\alpha_i))$.

Из $\Gamma, \exists\alpha_i(C(\alpha_i) \supset \forall\beta C(\beta)) \models B$ и $\models \exists\alpha_i(C(\alpha_i) \supset \forall\beta C(\beta))$ следует $\Gamma \models B$ или

Из $\Gamma, \exists\alpha_i(\exists\beta C(\beta) \supset C(\alpha_i)) \models B$ и $\models \exists\alpha_i(\exists\beta C(\beta) \supset C(\alpha_i))$ следует $\Gamma \models B$.

Таким образом, имеет место $\forall r_1(r_1 \leq r \supset Л(n, r_1)) \supset Л(n, r+1)$.

Доказано.

Теорема 2.2.4. Система BMV семантически непротиворечива.

Доказательство. Следует из леммы 2.2.3.

Доказано.

Глава 3. Алгоритм поиска вывода в системе BMV

§ 3.1. Изменение формулировки системы BMV

В целях эффективности автоматического поиска вывода добавим к BMV-правилам следующие правила исключения логических связок:

$$\neg\exists_{и}\ \frac{\neg\exists\alpha A(\alpha)}{\forall\alpha\neg A(\alpha)} \qquad \neg\vee_{и}\ \frac{\neg(A \vee B)}{\neg A} \quad \frac{\neg(A \vee B)}{\neg B}$$

Несложно показать, что данные правила производны: их добавление не влияет на множество теорем исчисления.

Теперь объясним необходимость введения именно таких правил.

Отсутствие этих правил приводит к следующим затруднениям [Болотов и др.].

Пусть (ситуация «а») в алгоритме необходимо доказать формулу $A \vee B$, но в выводе нет ни A, ни B.

Тогда (ситуация «б») в качестве последней посылки в последовательность вывода помещается $\neg(A \vee B)$ и теперь необходимо получить не формулу $A \vee B$, а 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.

Во избежании цикла 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 последней 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 выводимость формул $\neg A$, $\neg B$ из $\neg(A \vee B)$.

Т.к. 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 $A \vee B$. Т.е. от ситуации «$в_1$» алгоритм не перейдет к ситуации «а».

Таким образом, цикл в работе алгоритма ликвидирован. Нетрудно видеть, что выводимость формул $\neg A$, $\neg B$ из $\neg(A \vee B)$ в ситуации «$в_1$» есть ничто иное, как вводимое сейчас в системы 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, что *может быть построен* алгоритм поиска вывода в BMV без правил $\neg\vee_и$, $\neg\exists_и$ [Болотов и др.], [Болотов и др.1].

Однако введение этих правил облегчает 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 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.

$$\forall_в \quad \frac{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)}{\forall\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)} \quad \begin{matrix}\beta - абс.;\\ \gamma_1,\ldots, \gamma_n - огр.\end{matrix} \qquad \exists_и \quad \frac{\exists\alpha A(\alpha, \gamma_1,\ldots, \gamma_n)}{A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)} \quad \begin{matrix}\beta - абс.;\\ \gamma_1,\ldots, \gamma_n - огр.\end{matrix}$$

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 $\{\gamma_1,\ldots, \gamma_n\}$, т.к., согласно определению BMV-вывода, ни одна переменная не может ограничивать сама себя (Определение 2.1.3).

Покажем, что система BMV *неявно* использует сколемовские функции в правилах $\forall_в$, $\exists_и$. (Явно сколемовские термы используются при формулировке правила $\exists_и$ в системе натурального вывода, предложенной Е.К. Войшвилло [Войшвилло], т.е. в той системе, модификацией которой является BMV.)

Приведем вариант записи данных правил с использованием сколемовских функций σ, σ_1 и т.д. [Мендельсон, с. 111]: $\forall_в$: из $A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$ выводима $\forall\alpha A$; $\exists_и$: из $\exists\alpha A$ выводима $A(\alpha/\sigma(\gamma_1,\ldots, \gamma_n))$, 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, результатом сколемизации формулы $\exists_v\forall_w\exists_x\forall_y\exists_z(Q(v, w, y, z) \supset P(f(x, y), g(z)))$ будет $Q(a, w, y, \sigma_1(w, y)) \supset P(f(\sigma(w), y), g(\sigma_1(w, y)))$.

Теперь сравним запись 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 множество $\{\gamma_1,\ldots,\gamma_n\}$ совпадает. 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 константы, 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 отметить содержательную трактовку (полностью верную для правила $\exists_и$ и частично верную для правила $\forall_в$) абсолютной переменной как константы, обозначающей некий объект, для которого выполняется 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» [Бочаров и Маркин, 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 тоже: 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 требование в варианте BMV, что в BMV-выводе ни одна переменная не может быть абсолютно ограничена дважды (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 процедуре унификации, которая будет описана ниже, это означает, что абсолютные переменные наряду с константами не подлежат замене на другие термы, или, другими 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 [Бочаров и Маркин, с. 135]: именно данный запрет, а не требование правильной подстановки не позволяет вывести в системе из A(y, y) формулу $\forall x A(x, y)$ и из $\exists x B(x, y)$ формулу B(y, y). Отмечается, что «при таком переходе переменная y ограничивает сама себя».

Далее авторы пишут: «Надо только учитывать, что ситуация, когда переменная ограничивает сама себя, может возникнуть не только прямым образом, как это было в приведенных примерах, но и косвенно. Здесь имеется в виду то обстоятельство, что отношение «x ограничивает y» является транзитивным, то есть для него верно соотношение: «если α ограничивает β, а β ограничивает γ, то α ограничивает γ». В силу этого в системе не проходит доказательство формулы $\forall x \exists y P(x, y) \supset \exists y \forall x P(x, y)$.

Пусть запись $\beta < \gamma$ означает, что переменная β абсолютно ограничена и переменная γ ограничена относительно переменной β. 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 $\exists \alpha A(\alpha)$.

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 (частичного) порядка.

Заметим, что теория строгого порядка разрешима.

Поскольку в конечном выводе (доказательстве) мы имеем дело только с конечным числом абсолютно и относительно ограниченных переменных, *разрешимость теории строго порядка позволяет за конечное число шагов определить, ограничивает ли произвольная переменная в произвольном BMV-выводе сама себя*.

§ 3.2. Унификация

Введем основные определения, следуя [Чень и Ли].

Особенность нижеследующего алгоритма унификации – работа с абсолютными и неабсолютными переменными, а также работа с множествами ограниченных переменных. Поскольку семантически абсолютные переменные сходны с константами, алгоритм, например, запрещает подстановку одной абсолютной переменной вместо другой абсолютной переменной.

Определение 3.2.1. 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 и логических связок и кванторов.

Свойство 2 исключает подстановку неабсолютной переменной вместо ее самой.

Свойство 3 исключает различные варианты подстановки для одной и той же неабсолютной переменной.

Пусть $\theta = \{\alpha_1/t_1, \ldots, \alpha_n/t_n\}$ – подстановка и A – формула. Тогда $A\theta$ – формула, полученная 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.

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.

Подстановка θ *накрывает* подстановку θ_1 (обозначается $\theta \geq \theta_1$), если существует подстановка θ_2: $\theta_1 = \theta^\circ\theta_2$. Если $\theta \geq \theta_1$ и $\theta_1 \geq \theta$, то $\theta = \theta_1$.

Определение 3.2.2. Подстановка θ называется *унификатором* для множества формул $\{A_1\theta, \ldots, A_k\theta\}$ т.т.т., когда $A_1\theta = A_2\theta = \ldots = A_k\theta$. Множество формул $\{A_1, \ldots, A_k\}$ *унифицируемо*, если для этого множества 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 $A(x, v)\theta = A(y, v) = A(y, z)\theta$. Значит, множество формул {A(x, v) и A(y, z)} унифицируемо.

Определение 3.2.3. Унификатор σ для множества формул $\{A_1, \ldots, A_k\}$ 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.

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. $k = 0$, $A_k = A$, $B_k = B$, $\sigma_k = \varepsilon$ (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 $A_k = B_k$. Значит, последняя подстановка является наиболее общим унификатором для формул A_0 и B_0./

Иначе найдем множество рассогласований D_k для формул 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: формулы 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 формулы $A_0(B_0)$ с помощью подстановки $\sigma_{k+1} = ((\sigma_1 \circ \sigma_2) \circ \sigma_3) \ldots \circ \sigma_k))$, т.е. с помощью композиции k подстановок./

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

Теорема 3.2.4. Вышеприведенный алгоритм конечен и всегда находит наиболее общий унификатор для унифицируемого множества A.

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

Данная проверка конечна в силу разрешимости теории строго порядка, которая описывает абсолютно и относительно ограниченные переменные в выводе.

С другой стороны, данная проверка сужает множество подстановок, допустимых согласно алгоритму унификации в [Чень и Ли].

Поскольку алгоритм унификации в [Чень и Ли] конечен и всегда находит наиболее общий унификатор, наш алгоритм унификации также конечен и всегда находит наиболее общий унификатор.

Доказано.

Поясним данный алгоритм на примерах.

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, т.к. подстановка $\theta = \{x/y\}$ невозможна, поскольку 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 абсолютна 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 $B_2 = P(y_2, y_2)$. 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.

В заключении отметим, что алгоритм унификации можно задать таким образом, чтобы он унифицировал формулы, отличающиеся между собой только *вырожденными* вхождениями кванторов, например, ∀xP(x) и ∀y∀xP(x). Такой алгоритм предполагает упорядоченное переименование всех связанных переменных [Sieg & Byrnes].

Унификация класса формул, отличающихся вхождениями различных связанных переменных (например, ∀xP(x) и ∀yP(y)), требует введения в исчисление BMV *правила правильной подстановки*. Этого мы не будем делать. См. такой алгоритм унификации в [Sieg & Byrnes], [Li].

§ 3.3. Правила поиска вывода в системе BMV

Основная идея нижеследующего алгоритма – построение двух непустых последовательностей. Интуитивно говоря, поиск вывода осуществляется в двух (*синтетическом* и *аналитическом*) направлениях.[4] С одной стороны, мы стараемся получить необходимую формулу из уже имеющихся в выводе формул.[5] С другой стороны, задача поиска вывода необходимой формулы сводится к подзадаче поиска вывода другой формулы таким образом, что успешное решение подзадачи позволяет успешно решить саму задачу.[6]

Соответственно, те преобразования, с помощью которых алгоритм осуществляет поиск вывода в синтетическом (аналитическом) направлении, мы будем называть *синтетическими (аналитическими) правилами поиска вывода*.

Особо отметим, что в данном контексте «правило» не понимается как преобразование формул в рамках некоторого исчисления (в таком смысле «правило» понимается в Главе 2 нашей работы, где речь идет о формулировке правил системы BMV).

Здесь и далее понятие «синтетическое (аналитическое) правило» понимается как некоторая эвристика, позволяющая в процессе поиска вывода выводить новые

---

[4] В англоязычной литературе синтетическое направление называется forward reasoning, а аналитическое направление – backward reasoning, соответственно.

[5] «Задачу устранения «дырки» в строящемся выводе можно решать, как говорят, «в лоб», не сводя ее к подзадачам, а просто заполняя пробел, вставляя сверху от знака |– по определенным правилам новые и новые формулы, пока в конце концов не получится искомое заключение. Такой метод поиска вывода можно назвать синтетическим» [Смирнов и др., с. 40].

[6] Однако в процессе поиска вывода наряду с синтетическими могут осуществляться и аналитические шаги. В этом случае мы сводим задачу по поиску данного вывода к одной или нескольким подзадачам по поиску вспомогательных выводов [Смирнов и др., с. 42].

формулы из уже имеющихся (синтетические правила) или сводить задачу поиска некоторой формулы к подзадачам поиска других формул (аналитические правила).

Отметим, что такое (неформальное) понимание понятия «правило» согласуется с одной из задач нашего исследования, а именно, обоснование возможности *прямого* (т.е. не с помощью промежуточных исчислений) доказательства теоремы о семантической полноте алгоритма поиска натурального вывода.[7]

Все правила вывода системы BMV рассматриваются как синтетические правила поиска вывода в этой системе. Действительно, правила введения и правила исключения логических связок обладают общей чертой: с их помощью в **последовательность вывода** (ПВ) добавляются новые формулы. Таким образом, в процессе поиска вывода эти правила используются для выведения новых формул в ПВ.

С помощью аналитических правил поиска вывода алгоритм строит **последовательность целей** (ПЦ), которые алгоритм стремится *достигнуть* с помощью формул из ПВ. ПЦ не является BMV-выводом, т.к. целью может быть достижение противоречия в ПВ, т.е. целью необязательно является формула, а также потому, что цели получаются из предыдущих целей не по правилам системы BMV.

Остановимся более подробно на последовательности целей.

Основными понятиями для ПЦ являются понятия «главная цель» и «текущая цель». *Главная* цель – это первая цель в ПЦ, т.е. формула, BMV-вывод которой строится алгоритмом. *Текущая* цель – это последняя цель в ПЦ на данный момент поиска вывода, 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 (достигнута)*, если она унифицируется с некоторой неисключенной формулой из ПВ.

Если текущей целью является противоречие, то текущая цель *достижима (достигнута)*, если в выводе содержатся две неисключенные формулы вида A и ¬B,

[7] В противном случае, множество аналитических и синтетических правил, заданных в смысле Главы 2 нашей работы, в совокупности с некоторым понятием вывода образовывало бы некоторое промежуточное исчисление, позволяющее искать вывод в системе BMV. См. данный подход в [Sieg], [Sieg & Byrnes].

причем для A и B существует унификатор, который, будучи примененным к A и B, дает две противоречащие друг другу формулы.

В противном случае цель считается *недостижимой (недостигнутой)*.

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.

Теперь остановимся подробнее на аналитических правилах поиска вывода.

Характерной чертой этих правил является сведение «задачи поиска некоторого вывода к одной или нескольким подзадачам по поиску вспомогательных выводов» [Смирнов и др., с. 4]. Используя предложенную в нашей работе терминологию, можно сказать, что 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 достигнуть цель A & B. 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, из которых A становится текущей целью, а B – предыдущей целью по отношению к A (и подцелью по отношению к A & B) в ПЦ. Если подцели A и B достижимы (а значит, A и B находятся в ПВ), то, применяя $\&_в$, можно достигнуть цель A & B.

Если необходимо достигнуть текущую цель $A \supset B$, то можно взять формулу A в качестве посылки в ПВ и стараться достигнуть подцель B. Если подцель B достижима (а значит, B находится в ПВ), то, применяя $\supset_в$, можно достигнуть цель $A \supset B$.

Если необходимо достигнуть текущую цель $\neg A$, то лучше взять в качестве посылки формулу A в ПВ и стараться достигнуть подцель противоречие. Если подцель противоречие достижима, то, применяя $\neg_в$, можно достигнуть цель $\neg A$.

Такие советы по поиску вывода иногда называются *эвристиками*. Общее для вышеперечисленных 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.[8]

В нашей работе мы называем такие аналитические правила поиска вывода для целей *ц-правилами*.

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

$$\&_{ц}\ \frac{A \,\&\, B \in ПЦ}{A^{M3} \in ПЦ,\ B \in ПЦ} \qquad \supset_{ц}\ \frac{A \supset B \in ПЦ}{A - пос. \in ПВ,\ B \in ПЦ}$$

$$\neg_{ц}\ \frac{A \in ПЦ}{\neg A - пос. \in ПВ,\ \bot \in ПЦ} \qquad \vee_{ц}\ \frac{A \vee B \in ПЦ}{A^{M4} \in ПЦ} \qquad \frac{A \vee B \in ПЦ}{B^{M5} \in ПЦ}$$

$$\exists_{ц}\ \frac{\exists \alpha A(\alpha) \in ПЦ}{A(\alpha/\beta)^{M6} \in ПЦ,\ \beta - \text{новая неабсолютная переменная}} \qquad \forall_{ц}\ \frac{\forall \alpha A(\alpha, \gamma_1, \ldots, \gamma_n) \in ПЦ}{A(\alpha/\beta, \gamma_1, \ldots, \gamma_n) \in ПЦ}\ \beta - абс.;\ \gamma_1, \ldots, \gamma_n - огр.$$

В правиле $\neg_{ц}$ формула А может быть либо элементарной формулой $P(\alpha)$, либо $\neg C$, либо $C \vee D$, либо $\exists \alpha C$.

Проанализируем ц-правила. Эти правила строятся таким образом, что достижение подцели заключения с необходимостью влечет достижение предыдущей цели.

Правило $\forall_{ц}$ 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 – 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 В: В и $A(\alpha/\beta, \gamma_1, \ldots, \gamma_n)$ унифицируемы с помощью подстановки Ω), то к $B\Omega$ можно применить правило $\forall_{в}$ по переменной β, вывести формулу $\forall \alpha A(\alpha, \gamma_1, \ldots, \gamma_n)\Omega$ и достигнуть цель $\forall \alpha A(\alpha, \gamma_1, \ldots, \gamma_n)\Omega$.

Ц-правила представляют собой переход от цели большей степени к цели меньшей степени из ПЦ вплоть до получения цели противоречие, к которой никакое ц-правило неприменимо. Поскольку алгоритм работает с формулами конечной длины, количество применений ц-правил к произвольной цели конечно. С другой стороны, ц-

[8] *Степенью* формулы А, обозначается g(A), является количество вхождений логических символов в формулу А: $g(P(\alpha)) = 0$, где $P(\alpha)$ обозначает *элементарную* формулу; $g(\neg A) = g(\forall \alpha A(\alpha)) = g(\exists \alpha A(\alpha)) = g(A) + 1$; $g(A \,\&\, B) = g(A \vee B) = g(A \supset B) = g(A) + g(B) + 1$.

правила не работают с целью противоречие. Значит, сформулируем простое свойство ц-правил: процесс сведения произвольной цели из ПЦ с помощью ц-правил к цели противоречие конечен. Содержательно это означает, что процесс сведения некоторой цели к цели противоречие с помощью ц-правил конечен.

Помимо ц-правил, существуют группа аналитических правил поиска вывода, которые применяются не к целям из ПЦ, а к формулам из ПВ.

Например, пусть текущей целью является противоречие, 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 противоречие?

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 противоречие. 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 следующей главе нашей работы).

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 в нашей работе мы называем правилами взятия цели по формуле вывода или, кратко, *вц-правила*. (Отметим, что вц-правила применяются только, если текущей целью является противоречие.) В дальнейшем формулы, к которым могут применяться вц-правила, мы будем называть *вц-формулы*.

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

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

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

В правиле $\neg_{\text{вц}}$ формула A не имеет вида элементарной формулы $P(\alpha)$, или $C \vee D$, или $\exists \alpha C$.

В правиле $\forall_{\text{вц}}$ терм t не может быть неабсолютной переменной. Более того, мы подставляем только такой терм t, что неисключенная формула $A(\alpha/t)$ *не находится выше* в ПВ.[9]

Отметим, что правило $\forall_{\text{вц}}$ *отлично* от правила $\forall_{\text{и}}$. Правило $\forall_{\text{вц}}$ разрешает снимать квантор общности по *некоторым определенным* термам из ПВ. Таким образом, применение правила $\forall_{\text{вц}}$ представляет собой конечное число применений правила $\forall_{\text{и}}$.

ВЦ-правилам соответствуют аналитические правила удаления в [Смирнов и др.], [Смирнов-мл.] и [Смирнов-мл.1]. Сравнительный анализ данных групп правил обнаруживает значительные различия. Например, в нашей нотации аналитическое правило поиска вывода $\vee$_ae в [Смирнов-мл., с. 99] запишется следующим образом:

$$\vee\text{\_ae}\ \frac{A \vee B \in \text{ПВ}, C \in \text{ПЦ}}{\begin{array}{c} A - \text{пос.} \in \text{ПВ}, C \in \text{ПЦ} \\ B - \text{пос.} \in \text{ПВ}, C \in \text{ПЦ} \end{array}}$$

Если мы хотим достичь текущей цели C из формулы $A \vee B$, то мы помещаем A в ПВ в качестве посылки и в качестве текущей цели берем формулу C. Если C достижима, то формула B помещается в ПВ в качестве посылки и текущей целью становится формула C. Если и в этом случае C достижима, то в ПВ применяется правило разбора случаев.

В системе BMV нет правила разбора случаев. Поэтому предлагаемый алгоритм в такой ситуации берет в качестве текущей цели $\neg A$. Если она достижима, то к формулам $A \vee B$, $\neg A$ из ПВ применяется правило $\vee_{\text{и}}$ и в ПВ помещается формула B.

Отметим, что формульным коррелятом правила $\vee$_ae является отношение между формулами $(A \vee B) \supset C$ и $(A \supset C)$ & $(B \supset C)$. Данное отношение является отношением логического следования в обе стороны, т.е. эти формулы эквивалентны в

[9] Таким образом предотвращаются избыточные применения правила $\forall_{\text{и}}$ в ПВ.

классической пропозициональной логике. Таким образом, правило $\vee$_ае представляет собой эквивалентный переход.

В свою очередь, формульным коррелятом правила $\vee_{вц}$ является отношение между формулами $(A \vee B) \supset C$ и $(\neg A \supset B) \supset C$, которое также является отношением логического следования в обе стороны, т.е. эти формулы эквивалентны. Таким образом, правило $\vee_{вц}$ представляет собой *иной* эквивалентный переход.

Аналогичные замечания относятся к аналитическому правилу поиска вывода $\supset$_ае в [Смирнов-мл., с. 99], которое в нашей нотации запишется следующим образом:

$$\textbf{⊃_ае} \quad \frac{A \supset B \in ПВ,\ C \in ПЦ}{\begin{array}{c}\neg C - пос. \in ПВ,\ A \in ПЦ \\ B \in ПВ,\ C \in ПЦ\end{array}}$$

Если мы хотим достичь текущей цели С из формулы $A \supset B$, то мы помещаем $\neg C$ в ПВ в качестве посылки и в качестве текущей цели берем формулу А. Если она достижима, т.е. формула А появилась в ПВ, то к формулам $A \supset B$, А применяется правило $\supset_{и}$ и в ПВ помещается формула В. При этом текущей целью становится формула С.

Предлагаемый алгоритм в такой ситуации устанавливает текущую цель А. Если она достижима, то к формулам $A \supset B$, А применяется правило $\supset_{и}$ и в ПВ помещается формула В.

Формульным коррелятом правила $\supset$_ае является отношение между формулами $(A \supset B) \supset C$ и $(\neg C \supset A)$ & $(B \supset C)$. Данное отношение является отношением логического следования в обе стороны, т.е. эти формулы эквивалентны в классической пропозициональной логике. Таким образом, правило $\supset$_ае представляет собой эквивалентный переход. Однако наш алгоритм в такой ситуации вводит новую текущую цель А, т.е. он не использует эквивалентные переходы.

Итак, алгоритм осуществляет поиск вывода в двух направлениях.

Первое направление – от формул вывода к доказываемой формуле (главной цели) путем применения синтетических правил поиска (правил введения и исключения логических связок).

Второе направление – от доказываемой формулы (главной цели) к формулам вывода путем применения аналитических правил поиска вывода (ц-правил и вц-правил): к доказываемой формуле (главной цели) применяются ц-правила и к формулам из ПВ применяются вц-правила.

§ 3.4. 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.

Пусть ПВ и ПЦ служат сокращениями для понятий «последовательность вывода» и «последовательность целей», соответственно, тогда к формулам из ПВ применяются следующие процедуры, которые мы делим на три группы: процедуры 1*-4*, которые применяются только к формулам из ПВ; процедура 5*, которая применяется только к целям из ПЦ, и процедуры 6*-7*, которые 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 $\exists_и$.

Отметим, что заданный порядок призван сделать поиск вывода более эффективным, позволяя не загромождать дерево поиска вывода. Более того, данный порядок не является единственным и, поэтому, не учитывается при нижеследующем доказательстве теоремы о семантической полноте алгоритма.

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 M0 (смысл данной метки см. далее).[10] Если среди всех неисключенных формул в ПВ найдется формула A_1: $A_1\theta = A_2\theta$, где θ – унификатор, и ни одна переменная не ограничивает сама себя, то данный унификатор применяется к $A_2 \supset B$, A_1 и из формул $A_1\theta$, $A_2 \supset B\theta$ по правилу $\supset_и$ выводима формула $B\theta$.

$\vee_и$. 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.

Синтетические правила введения применяются следующим образом:

$\&_в$. Пусть формула A & B является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдутся формулы A и B, то к ним применяется правило $\&_в$, формула A & B добавляется в ПВ, а цель A & B помечается как *достигнутая*.

$\vee_в$. Пусть формула $A \vee B$ является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдется формула A или формула B, то к ней применяется правило $\vee_в$, формула $A \vee B$ добавляется в ПВ, а цель $A \vee B$ помечается как *достигнутая*.

$\supset_в$. Пусть формула $A \supset B$ является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдется формула A, которая является последней посылкой, и формула B, то к формуле B применяется правило $\supset_в$, формула $A \supset B$ добавляется в ПВ, а цель $A \supset B$ помечается как *достигнутая*.

$\neg_в$. Пусть цель противоречие является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдутся формулы A и $\neg B$: $A\theta = B\theta$, где θ – унификатор, и ни одна переменная не ограничивает сама себя, то к последней неисключенной посылке C из ПВ применяется правило $\neg_в$, формула $\neg C$ добавляется в ПВ, все неисключенные формулы из ПВ, начиная с формулы C и заканчивая формулой, предшествующей формуле $\neg C$, исключаются из ПВ, а цель противоречие помечается как *достигнутая*.

$\exists_в$. Пусть формула $\exists\alpha A(\alpha)$ является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдется формула A(t), где t – произвольный терм, то к формуле A(t) применяется правило $\exists_в$, формула $\exists\alpha A(\alpha)$ добавляется в ПВ, а цель $\exists\alpha A(\alpha)$ помечается как *достигнутая*.

$\forall_в$. Пусть формула $\forall\alpha A(\alpha)$ является *текущей* целью в ПЦ. Если среди неисключенных формул из ПВ найдется формула $A(\alpha_i)$, где α_i не является абсолютно

---

[10] Грубо говоря, к формуле $A_2 \supset B$ не применялось правило $\supset_и$.

ограниченной переменной, то, если ни одна переменная не ограничивает сама себя при абсолютно ограниченной α_i, к формуле $A(\alpha_i)$ применяется правило $\forall_в$, формула $\forall\alpha A(\alpha)$ добавляется в ПВ, делается отметка, что α_i относительно ограничивает все остальные свободные переменные в $A(\alpha_i)$, а цель $\forall\alpha A(\alpha)$ помечается как *достигнутая*.

*Процедура 3** определяет применение вц-правил, кроме $\forall_{вц}$.[11] Алгоритм ищет первую сверху вниз сложную неисключенную формулу из ПВ, которая не отмечена метками М0 или М5 (смысл этих меток см. далее). Эта формула является *вц-формулой*. Вид вц-формулы определяет, какое именно вц-правило применимо.

*Процедура 4** применяется после всех остальных процедур, если в ПВ содержатся неисключенные формулы вида $\forall\alpha A(\alpha)$, к которым уже применялось правило $\forall_и$ (т.е. они отмечены меткой М1). В таком случае метка М1 снимается со всех таких формул квантор общности снимается по *некоторым определенным термам из ПВ*.[12]

Напомним, что в правиле $\forall_{вц}$ терм t не может быть неабсолютной переменной. Более того, мы подставляем только такой терм t, что неисключенная формула $A(\alpha/t)$ *не находится выше* в ПВ.

К формулам из ПЦ применяются следующая процедура:

*Процедура 5** определяет применение ц-правил.

Ц-правила применяются к текущей цели следующим образом:

$\&_ц$. Пусть формула A & B является *текущей* целью в ПЦ. Если она не достижима, то формула A становится текущей целью и отмечается меткой М3. Если цель A достижима, то текущей целью становится формула B. Если цель B достижима, то текущей целью вновь становится формула A & B.

$\vee_ц$. Пусть формула $A \vee B$ является *текущей* целью в ПЦ. Если она не достижима, то формула A становится текущей целью и отмечается меткой М4. Если цель A недостижима, то цель A удаляется из ПЦ и формула B становится текущей целью и отмечается меткой М5. Если цель B недостижима, то цель B удаляется из ПЦ, формула $\neg(A \& B)$ помещается в ПВ в качестве последней посылки и цель противоречие становится текущей целью.

[11] Применение $\forall_{вц}$ определяется Процедурой 4*.

[12] Отметим, что метка М1 снимается с таких формул, а значит, впоследствии к ним применимо правило $\forall_и$ по новой неабсолютной переменной из ПВ.

$\supset_{ц}$. Пусть формула $A \supset B$ является *текущей* целью в ПЦ. Если она не достижима, то формула A помещается в ПВ в качестве последней посылки и формула B становится текущей целью.

$\neg_{ц}$. Пусть формула A является *текущей* целью в ПЦ, где A может быть либо элементарной формулой $P(\alpha)$, либо $\neg C$, либо $C \vee D$, либо $\exists\alpha C$. Если она не достижима, то формула $\neg A$ помещается в ПВ в качестве последней посылки и цель противоречие становится текущей целью.

$\exists_{ц}$. Пусть формула $\exists\alpha A$ является *текущей* целью в ПЦ. Если она не достижима, то формула $A(\alpha/\beta)$, где β – новая неабсолютная переменная, становится текущей целью.

$\forall_{ц}$. Пусть формула $\forall\alpha A$ является *текущей* целью в ПЦ. Если она не достижима, то формула $A(\alpha/\beta, \gamma_1,\ldots, \gamma_n)$ становится текущей целью, где β – новая абсолютная переменная, которая относительно ограничивает все свободные переменные $\gamma_1,\ldots, \gamma_n$ из $\forall\alpha A$.

По всему дереву поиска вывода применяется следующие процедуры:

*Процедура 6** осуществляет глобальную подстановку Ω в ПВ и ПЦ, если Ω – наиболее общий унификатор. Если в процессе поиска вывода произошло достижение некоторой цели с помощью Ω и ни одна переменная не ограничивает сама себя, то алгоритм осуществляет данную подстановку Ω во всех (исключенных и неисключенных) формулах и целях из ПВ и ПЦ.

*Процедура 7** осуществляет с помощью алгоритма унификации проверку достижимости текущей цели.

Если текущей целью является формула, то цель *достижима (достигнута)*, если цель унифицируется с некоторой неисключенной формулой вывода и ни одна переменная не ограничивает сама себя.

Если текущей целью является цель противоречие, то цель *достижима (достигнута)*, если в ПВ содержатся две неисключенные формулы вида A и $\neg B$, причем для A и B существует унификатор, и ни одна переменная не ограничивает сама себя. В противном случае цель считается *недостижимой (недостигнутой)*.

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) правила исключения после применения данного правила.

М1 – ставится на формулы вида $\forall\alpha A$ после применение к ним правила $\forall_и$.

М2 – ставится на формулы, полученные в результате применения правил исключения.

✓ Следующие метки ставятся на цели из ПЦ:

М3 – ставится на левый конъюнкт A_i, полученный из цели $A_i \,\&\, A_j$.

М4 – ставится на левый дизъюнкт A_i, полученный из цели $A_i \vee A_j$.

М5 – ставится на правый дизъюнкт A_j, полученный из цели $A_i \vee A_j$.

М6 – ставится на формулу A(t), полученную из цели $\exists\alpha A(\alpha)$.

✓ М7 – ставится на формулы из ПВ и ПЦ при применении вц-правил.

Описание алгоритма

Символами /…/ обозначены комментарии.

1. Определяется *главная цель* вывода. Таковой является формула, стоящая справа от знака выводимости в заданном утверждении о выводимости. 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 текущей цели и начальных посылок помечаются как *абсолютно ограниченные*. Переход к 10. /Алгоритм проверяет, не является ли рассматриваемый вывод *тривиальным*, т.е. главная цель вывода унифицируется с одной из начальных посылок вывода./

2. Применение правил исключения к формулам из ПВ.

2.1. Применяется процедура 1*. /Применяются правила исключения./

При этом на посылки правил вывода вида $A \,\&\, B$, $A \vee B$, $A \supset B$, $\neg\neg A$, $\neg(A \vee B)$, $\neg\exists\alpha A$, $\forall\alpha A$ и $\exists\alpha A$, к которым были применены, соответственно, правила $\&_и$, $\vee_и$, $\supset_и$, $\neg_и$, $\neg\vee_и$, $\neg\exists_и$, $\forall_и$ и $\exists_и$, ставится М0, указывающая на два обстоятельства: (а) недопустимость вторичного применения к ним указанных правил, (б) невозможность их использования в качестве вц-формул. На заключения правил вывода ставится М2.

Правила $\exists_и$, $\forall_и$ применяются следующим образом:

$\forall_и$.

a) Квантор общности снимается по правилу $\forall_и$ с формулы $\forall\alpha A$ по *новой* неабсолютной переменной. На формулу $\forall\alpha A$ ставится метка М1.

b) Если вхождение квантора $\forall$ по переменной α в формуле А *вырождено*, то результатом снятия квантора $\forall$ будет А.

$\exists_и$.

a) Квантор $\exists$ снимается по правилу $\exists_и$ с формулы $\exists\alpha A$ по *новой* абсолютной переменной.

b) Рядом с так полученной формулой $A(\alpha/\beta)$ пишется "β – абс., $\gamma_1, \gamma_2,..., \gamma_n$ – огр.", где $\gamma_1, \gamma_2,..., \gamma_n$ – все свободные переменные, входящие в $\exists\alpha A$.

c) Если первое вхождение квантора $\exists$ по переменной α в формуле $\exists\alpha A$ *вырождено*, то результатом снятия квантора $\exists$ будет А.

Переход к 10.

2.2. Если ни одно правило исключения не применимо, то переход к 3.

3. Применяется процедура 5* к текущей цели А.

- А – элементарная формула $P(\alpha)$. Переход к 4.
- Главный знак А – $\neg$. Переход к 4.
- $A = A_i \supset A_j$. Переход к 5.
- $A = A_i \,\&\, A_j$. Переход к 6.
- $A = A_i \vee A_j$. Переход к 7.
- $A = \exists\alpha A$. Переход к 8.
- $A = \forall\alpha A$. Переход к 9.
- А есть $\perp$. Переход к 12.

4. Формула $\neg A$ помещается в ПВ в качестве последней посылки. Текущей целью становится получение $\perp$. Переход к 2. /Алгоритм начинает доказывать формулы от противного./

5. Формула A_i помещается в ПВ в качестве последней посылки. Текущей целью становится A_j. Переход к 2. /Алгоритм, доказывая импликативную формулу, берет в качестве посылки ее антецедент и пытается вывести ее консеквент./

6. Текущею целью становится формула A_i. На формулу A_i ставится метка М3. Переход к 10.

6.1. С формулы A_i из ПЦ снимается метка М3. Текущей целью становиться формула A_j. Переход к 10.

7. Текущею целью становится формула A_i. На формулу A_i из ПЦ, ставится метка М4. Переход к 10.

7.1. Удаляем цель A_i, отмеченную М4. Текущей целью становиться формула A_j. На формулу A_j ставится метка М5. Переход к 10.

7.2. Удаляем цель A_j, отмеченную М5. Текущей целью становиться $\perp$. Формула $\neg(A_i \vee A_j)$ помещается в ПВ в качестве последней посылки. Переход к 2.

8. Текущей целью становиться формула $A(\alpha/v)$, где v – новая неабсолютная переменная. На формулу $A(\alpha/v)$ ставится метка М6. Переход к 10.

8.1. Удаляем цель $A(\alpha/v)$, отмеченную М6. Текущей целью становиться $\perp$. Формула $\neg\exists\alpha A(\alpha)$ помещается в ПВ в качестве последней посылки. Переход к 2.

9. Текущей целью становиться формула $A(\alpha/\beta)$, где β – новая абсолютная переменная.

a) Отмечается, что «β – абс., γ_1, γ_2,..., γ_n – огр.», где γ_1, γ_2,..., γ_n – все свободные переменные, входящие в $\forall\alpha A$.

b) Если первое вхождение квантора $\forall$ по переменной α в формуле $\forall\alpha A(\alpha)$ вырождено, то результатом снятия квантора $\forall$ будет сама формула А.

Переход к 10.

10. Применяется процедура 7*.

10.1. Текущая цель достигнута. Применяется процедура 6* (глобальная подстановка):

- Если текущая цель отмечена М3, то переход к 6.1. /Начинаем доказывать правый конъюнкт./
- Если текущая цель не отмечена М3, то переход к 11.

10.2. Текущая цель не достигнута:

- Если текущая цель отмечена М4, то переход к 7.1. /Начинаем доказывать правый дизъюнкт./
- Если текущая цель отмечена М5, то переход к 7.2. /Начинаем доказывать дизъюнкцию от противного./

- Если текущая цель отмечена М6, то переход к 8.1. /Начинаем доказывать цель $\exists\alpha A(\alpha)$ от противного./
- Если текущая цель не отмечена М4-М6, то переход к 3.

11. Достигнута текущая цель A_n.

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

11.2. Если A_n не является *главной целью*, то

Если A_n есть $\perp$, то:

a) A_n устраняется из ПЦ.

b) Текущей целью становиться предыдущая цель и к формулам из ПВ применяется правило $\neg_в$, результатом применения которого является формула $\neg C$, где C – последняя неисключенная посылка в ПВ.

c) Делается отметка, что все формулы, начиная с формулы С и заканчивая формулой, предшествующей $\neg C$, исключаются из дальнейших шагов вывода.

d) Если в число этих формул попадает формула с меткой М2, то с соответствующей формулы снимается метка М0. Переход к 2.

Если A_n не есть $\perp$, то рассматриваем подцель A_{n-1} из ПЦ. В зависимости от вида A_{n-1} рассматриваются следующие случаи:

– $A_{n-1} = A_n \vee A_k$ или $A_k \vee A_n$, то:

a) A_n и A_{n-1} устраняются из ПЦ.

b) Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\vee_в$ к формуле A_n (A_k).

c) На формулу $A_n \vee A_k$ или $A_k \vee A_n$ из ПВ ставится метка М0.

d) Если на A_{n-1} стояла метка М7, то с *соответствующей вц-формулы* снимается метка М7. Переход к 2.

– $A_{n-1} = A_k \supset A_n$, то:

a) A_n и A_{n-1} устраняются из ПЦ.

b) 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.

c) Делается отметка, что все формулы, начиная с A_k и заканчивая формулой A_n, исключаются из дальнейших шагов вывода.

d) Если в число этих формул попадает формула с меткой M2, то с соответствующей формулы снимается метка M0.

e) Если на цели A_{n-1} стояла метка M7, то с *соответствующей вц-формулы* снимается метка M7. Переход к 2.

– $A_{n-1} = A_k \,\&\, A_n$, то:

a) A_n и A_{n-1} устраняются из ПЦ.

b) 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.

c) На формулу $A_k \,\&\, A_n$ из ПВ ставится метка M0.

d) Если на цели A_{n-1} стояла метка M7, то с *соответствующей вц-формулы* снимается метка M7. Переход к 2.

– $A_{n-1} = \forall\alpha A$, то:

a) A_n и A_{n-1} устраняются из ПЦ.

b) Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\forall_в$ к формуле $A(\alpha/\beta, \gamma_1, \gamma_2,..., \gamma_n)$.

c) Делается отметка, что β абсолютно ограничена и $\gamma_1, \gamma_2,..., \gamma_n$ относительно ограничены.

d) На формулу $\forall\alpha A$ ставятся метки M0 и M1.

e) Если на цели A_{n-1} стояла метка M7, то с *соответствующей вц-формулы* снимается метка M7. Переход к 2.

– $A_{n-1} = \exists\alpha A$, то:

a) A_n и A_{n-1} устраняются из ПЦ.

b) Текущей целью становится предыдущая цель и в ПВ включается формула A_{n-1}, которая является результатом применения правила $\exists_в$ к формуле $A(\alpha/\beta)$.

c) На формулу $\exists\alpha A$ ставится метка M0.

d) Если на цели A_{n-1} стояла метка M7, то с *соответствующей вц-формулы* снимается метка M7. Переход к 2.

– $A_{n-1} = \bot$, то:

a) A_n устраняется из ПЦ.

b) Текущей целью становится предыдущая цель A_{n-1}.

c) Если на цели A_{n-1} стояла метка M7, то с *соответствующей вц-формулы* снимается метка M7. Переход к 2.

12. Применяем процедуру 3*.

12.1. Если в ПВ имеются сложные формулы, не отмеченные метками M0 или M7, то к ним применяются следующие процедуры:

a) 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 M7. Переход к 3.

/Имея в выводе формулу вида $A_i \supset A_j$, алгоритм пытается вывести A_i для того, чтобы в случае успеха применить $\supset_и$./

b) 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 M7. Переход к 3. /Имея в выводе формулу вида $A_i \vee A_j$, алгоритм пытается вывести $\neg A_i$ для того, чтобы в случае успеха применить $\vee_и$./

c) Если в выводе встречается формула $\neg A_i$, где A_i не имеет вида $A_n \vee A_k$, $\exists\alpha A$, то в качестве текущей цели берется A_i. Формула $\neg A_i$ и цель A_i помечаются меткой M7. Переход к 3. /К формулам вида $\neg(A_n \vee A_k)$, $\neg\exists\alpha A$ применяются, соответственно, правила исключения $\neg\vee_и$, $\neg\exists_и$. К остальным формулам вида $\neg A_i$ алгоритм применяет вц-правила для того, чтобы получить цель противоречие./

12.2. Если все сложные формулы в ПВ отмечены M0 или M7, то 13.

13. Завершение алгоритма.

13.1. Если ПВ не содержит формул, отмеченных M1, то выход из алгоритма: ПВ содержит *конечный контрпример*. /Если в выводе нет формул вида $\forall\alpha A(\alpha)$, то ПВ содержит конечный контрпример (опровержение) для данной выводимости./

13.2. В противном случае переход к 14.

14. Применение процедуры 4*. /Применение $\forall_{вц}$./

Квантор общности снимается по правилу $\forall_и$ с каждой неисключенной формулы вида $\forall\alpha A$ из ПВ по *некоторым определенным* термам из ПВ. Со всех формул вида $\forall\alpha A$ снимается метка М1. Переход к 10.

Конец описания.

Приведем пример работы алгоритма. (*Курсивом* мы будем обозначать текущую цель.)

Пусть необходимо обосновать в алгоритме вывод формулы $\forall xP(x)\ \&\ \forall xQ(x)$ из формулы $\forall x(P(x)\ \&\ Q(x))$.

Согласно пункту 1 описания алгоритма, формула $\forall xP(x)\ \&\ \forall xQ(x)$ помещается в ПЦ в качестве *главной* цели, а последняя формула $\forall x(P(x)\ \&\ Q(x))$ помещается в ПВ в качестве *начальной* посылки. Алгоритм переходит к проверке достижимости текущей цели.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)\ \&\ Q(x))$ – посылка	$\forall xP(x)\ \&\ \forall xQ(x)$

Т.к. текущая цель не достижима, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Согласно пункту 3, текущей целью становится левый конъюнкт, на который ставится метка М3.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)\ \&\ Q(x))$ – посылка	$\forall xP(x)\ \&\ \forall xQ(x)$
	$\forall xP(x)^{M3}$

Т.к. текущая цель не достигнута, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Согласно пункту 3, текущей целью становится формула $P(x/y_1)$, где y_1 – это новая абсолютная переменная в ПВ и ПЦ.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)\ \&\ Q(x))$ – посылка	$\forall xP(x)\ \&\ \forall xQ(x)$
	$\forall xP(x)^{M3}$
	$P(y_1)$, y_1 новая абсолютная переменная

Т.к. текущая цель не достигнута, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Поскольку текущая цель – это элементарная формула, то,

согласно пункту 3, формула $\neg P(y_1)$ помещается в ПВ в качестве последней посылки, а текущей целью становится $\perp$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
	$P(y_1)$, y_1 новая абсолютная переменная
	$\perp$

Т.к. текущая цель не достигнута, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Поскольку текущая цель – это $\perp$, то, согласно пункту 2, алгоритм ищет возможность применения правил исключения в ПВ. К формуле $\forall x(P(x) \,\&\, Q(x))$ применяется правило $\forall_и$ по новой неабсолютной переменной из ПВ и ПЦ, т.е. по переменной x_1. На формулу $\forall x(P(x) \,\&\, Q(x))$ ставится метка M1. На формулу $P(x_1) \,\&\, Q(x_1)$ ставится метка M0.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
3. $P(x_1) \,\&\, Q(x_1)^{M0}$ – $\forall_и$: 1	$P(y_1)$, y_1 новая абсолютная переменная
	$\perp$

Согласно пункту 2, алгоритм делает все возможные заключения по правилам исключения и применяет правило $\&_и$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
3. $P(x_1) \,\&\, Q(x_1)^{M0}$ – $\forall_и$: 1	$P(y_1)$, y_1 новая абсолютная переменная
4. $P(x_1)^{M2}$ – $\&_и$: 3	$\perp$
5. $Q(x_1)^{M2}$ – $\&_и$: 3	

Т.к. никакие правила исключения в ПВ неприменимы, алгоритм проверяет достижимости текущей цели. Поскольку она достигнута (в ПВ имеются формулы

$\neg P(y_1)$ и $P(x_1)$, для которых существует унификатор $\{x_1/y_1\}$), то, согласно пункту 10.1, применяется глобальная подстановка $\{x_1/y_1\}$ в ПВ и ПЦ. Цель $\perp$ помечается как достигнутая и устраняется из ПЦ, а текущей целью становится $P(y_1)$. В ПВ применяется правило $\neg_в$ к формулам 2, 4 и из ПВ исключаются формулы 2-5.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⌈2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
| 3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$P(y_1)$, y_1 новая абсолютная переменная
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)$ – $\neg_в$: 2, 4	

После применения $\neg_и$ к формуле 6 вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⌈2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
| 3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$P(y_1)$, y_1 новая абсолютная переменная
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	

Поскольку текущая цель достигнута, то, согласно пункту 10.1, цель $P(y_1)$ помечается как достигнутая и устраняется из ПЦ, а текущей целью становится $\forall x P(x)$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⌈2. $\neg P(y_1)$ – пос.	$\forall x P(x)^{M3}$
| 3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	

6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4

7. $P(y_1)^{M2}$ – $\neg_и$: 6

Поскольку текущая цель является подцелью достигнутой цели, то она также является достигнутой, согласно пункту 11.2. К формуле $P(y_1)$ из ПВ применяется правило $\forall_в$, в ПВ помещается формула $\forall xP(x)$, цель $\forall xP(x)$ помечается как достигнутая и устраняется из ПЦ, метка М3 снимается, а текущей целью становится $\forall xP(x)$ & $\forall xQ(x)$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \& Q(x))^{M1}$ – посылка	$\forall xP(x) \& \forall xQ(x)$
⌈2. $\neg P(y_1)$ – пос.	
| 3. $P(y_1) \& Q(y_1)^{M0}$ – $\forall_и$: 1	
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.	

Поскольку текущая цель является подцелью достигнутой цели, отмеченной меткой М3, то текущей целью становится $\forall xQ(x)$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \& Q(x))^{M1}$ – посылка	$\forall xP(x) \& \forall xQ(x)$
⌈2. $\neg P(y_1)$ – пос.	$\forall xQ(x)$
| 3. $P(y_1) \& Q(y_1)^{M0}$ – $\forall_и$: 1	
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.	

Т.к. в ПВ нет формулы, которая унифицируется с $\forall xQ(x)$, то, согласно пункту 3, текущей целью становится $Q(y_2)$, где y_2 *новая* абсолютная переменная в выводе.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall x Q(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall x P(x)$ – $\forall_в$: 4; y_1 – абс. огр.	

Т.к. текущая цель не достигнута, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Поскольку текущая цель – это элементарная формула, то, согласно пункту 3, формула $\neg Q(y_2)$ помещается в ПВ в качестве последней посылки, а текущей целью становится $\perp$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall x Q(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	$\perp$
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall x P(x)$ – $\forall_в$: 4; y_1 – абс. огр.	
9. $\neg Q(y_2)$ – пос.	

Т.к. текущая цель не достигнута, алгоритм, согласно пункту 10.2, переходит к анализу текущей цели. Поскольку текущая цель – это $\perp$, то, согласно пункту 2, алгоритм ищет возможность применения правил исключения в ПВ. К формуле $\forall x(P(x) \,\&\, Q(x))$ не применяется правило $\forall_и$, т.к. эта формула отмечена меткой М1. Правило $\forall_и$ применяется к формуле $\forall x P(x)$, в ПВ помещается формула $A(x_2)$, где x_2 – новая неабсолютная переменная в ПВ, и на формулу $\forall x P(x)$ ставится метка М1.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))^{M1}$ – посылка	$\forall xP(x) \,\&\, \forall xQ(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall xQ(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	$\perp$
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)^{M1}$ – $\forall_в$: 4; y_1 – абс. огр.	
9. $\neg Q(y_2)$ – пос.	
10. $P(x_2)$ – $\forall_и$: 8	

Поскольку ни одно правило исключения не применимо, то Процедура 4* применяется следующим образом:

С формулы $\forall x(P(x) \,\&\, Q(x))$ квантор общности снимается по всем абсолютным переменным из ПВ: y_1 и y_2, т.к. ни формула $P(y_1) \,\&\, Q(y_1)$, ни формула $P(y_2) \,\&\, Q(y_2)$ не находятся неисключенными в ПВ. На формулы $P(y_1) \,\&\, Q(y_1)$ и $P(y_2) \,\&\, Q(y_2)$ ставятся метки M0 и M7. С формулы $\forall x(P(x) \,\&\, Q(x))$ снимается метка M1.

С формулы $\forall xP(x)$ квантор общности снимается только по абсолютной переменной y_2, т.к. формула $P(y_1)$ находится неисключенной в ПВ. На формулу $P(y_2)$ ставятся метки M0 и M7. С формулы $\forall xP(x)$ снимается метка M1.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))$ – посылка	$\forall xP(x) \,\&\, \forall xQ(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall xQ(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	$\perp$
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.	

9. $\neg Q(y_2)$ – пос.

10. $P(x_2)$ – $\forall_и$: 8

11. $P(y_1)$ & $Q(y_1)^{M0, M7}$ – $\forall_и$: 1

12. $P(y_2)$ & $Q(y_2)^{M0, M7}$ – $\forall_и$: 1

13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8

Согласно пункту 2, алгоритм делает все возможные заключения по правилам исключения и применяет правило $\&_и$ к формулам 11 и 12.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)$ & $Q(x))$ – посылка	$\forall xP(x)$ & $\forall xQ(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall xQ(x)$
⎢3. $P(y_1)$ & $Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	$\perp$
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.	
9. $\neg Q(y_2)$ – пос.	
10. $P(x_2)$ – $\forall_и$: 8	
11. $P(y_1)$ & $Q(y_1)^{M0, M7}$ – $\forall_и$: 1	
12. $P(y_2)$ & $Q(y_2)^{M0, M7}$ – $\forall_и$: 1	
13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8	
14. $P(y_1)^{M2}$ – $\&_и$: 11	
15. $B(y_1)^{M2}$ – $\&_и$: 11	
16. $P(y_2)^{M2}$– $\&_и$: 12	
17. $B(y_2)^{M2}$ – $\&_и$: 12	

Алгоритм проверяет достижимость текущей цели. Поскольку она достигнута (в ПВ имеются формулы $\neg Q(y_2)$ и $Q(y_2)$), то цель $\perp$ помечается как достигнутая и устраняется из ПЦ, а текущей целью становится $Q(y_2)$. В ПВ применяется правило $\neg_в$ к формулам 9, 17 и из ПВ исключаются формулы 9-17.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall x Q(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall x P(x)$ – $\forall_в$: 4; y_1 – абс. огр.	
⎡9. $\neg Q(y_2)$ – пос.	
⎢10. $P(x_2)$ – $\forall_и$: 8	
⎢11. $P(y_1) \,\&\, Q(y_1)^{M0, M7}$ – $\forall_и$: 1	
⎢12. $P(y_2) \,\&\, Q(y_2)^{M0, M7}$ – $\forall_и$: 1	
⎢13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8	
⎢14. $P(y_1)^{M2}$ – $\&_и$: 11	
⎢15. $B(y_1)^{M2}$ – $\&_и$: 11	
⎢16. $P(y_2)^{M2}$– $\&_и$: 12	
⎣17. $B(y_2)^{M2}$ – $\&_и$: 12	
18. $\neg\neg B(y_2)$ – $\neg_в$: 9, 12	

После применения $\neg_и$ к формуле 18 вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x) \,\&\, Q(x))$ – посылка	$\forall x P(x) \,\&\, \forall x Q(x)$
⎡2. $\neg P(y_1)$ – пос.	$\forall x Q(x)$
⎢3. $P(y_1) \,\&\, Q(y_1)^{M0}$ – $\forall_и$: 1	$Q(y_2)$, y_2 – новая абсолютная переменная
⎢4. $P(y_1)^{M2}$ – $\&_и$: 3	
⎣5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall x P(x)$ – $\forall_в$: 4; y_1 – абс. огр.	
⎡9. $\neg Q(y_2)$ – пос.	

| 10. $P(x_2)$ – $\forall_и$: 8

| 11. $P(y_1)$ & $Q(y_1)^{M0, M7}$ – $\forall_и$: 1

| 12. $P(y_2)$ & $Q(y_2)^{M0, M7}$ – $\forall_и$: 1

| 13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8

| 14. $P(y_1)^{M2}$ – $\&_и$: 11

| 15. $B(y_1)^{M2}$ – $\&_и$: 11

| 16. $P(y_2)^{M2}$– $\&_и$: 12

⌊17. $B(y_2)^{M2}$ – $\&_и$: 12

18. $\neg\neg B(y_2)^{M0}$ – $\neg_в$: 9, 12

19. $B(y_2)^{M2}$ – $\neg_и$: 18

Поскольку текущая цель достигнута, то, согласно пункту 10.1, цель $Q(y_2)$ помечается как достигнутая и устраняется из ПЦ, а текущей целью становится $\forall xQ(x)$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)$ & $Q(x))$ – посылка	$\forall xP(x)$ & $\forall xQ(x)$
⌈2. $\neg P(y_1)$ – пос.	*$\forall xQ(x)$*
| 3. $P(y_1)$ & $Q(y_1)^{M0}$ – $\forall_и$: 1	
| 4. $P(y_1)^{M2}$ – $\&_и$: 3	
⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3	
6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4	
7. $P(y_1)^{M2}$ – $\neg_и$: 6	
8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.	
⌈9. $\neg Q(y_2)$ – пос.	
| 10. $P(x_2)$ – $\forall_и$: 8	
| 11. $P(y_1)$ & $Q(y_1)^{M0, M7}$ – $\forall_и$: 1	
| 12. $P(y_2)$ & $Q(y_2)^{M0, M7}$ – $\forall_и$: 1	
| 13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8	
| 14. $P(y_1)^{M2}$ – $\&_и$: 11	
| 15. $B(y_1)^{M2}$ – $\&_и$: 11	
| 16. $P(y_2)^{M2}$– $\&_и$: 12	

⌊17. $B(y_2)^{M2}$ – $\&_и$: 12

18. $\neg\neg B(y_2)^{M0}$ – $\neg_в$: 9, 12

19. $B(y_2)^{M2}$ – $\neg_и$: 18

Поскольку текущая цель является подцелью достигнутой цели, то она также является достигнутой, согласно пункту 11.2. К формуле $Q(y_2)$ из ПВ применяется правило $\forall_в$, в ПВ помещается формула $\forall xQ(x)$, цель $\forall xQ(x)$ помечается как достигнутая и устраняется из ПЦ, а текущей целью становится $\forall xP(x)$ & $\forall xQ(x)$.

Теперь вывод выглядит следующим образом:

ПВ	ПЦ
1. $\forall x(P(x)$ & $Q(x))$ – посылка	$\forall xP(x)$ & $\forall xQ(x)$

⌈2. $\neg P(y_1)$ – пос.

⎢3. $P(y_1)$ & $Q(y_1)^{M0}$ – $\forall_и$: 1

⎢4. $P(y_1)^{M2}$ – $\&_и$: 3

⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3

6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4

7. $P(y_1)^{M2}$ – $\neg_и$: 6

8. $\forall xP(x)$ – $\forall_в$: 4; y_1 – абс. огр.

⌈9. $\neg Q(y_2)$ – пос.

⎢10. $P(x_2)$ – $\forall_и$: 8

⎢11. $P(y_1)$ & $Q(y_1)^{M0, M7}$ – $\forall_и$: 1

⎢12. $P(y_2)$ & $Q(y_2)^{M0, M7}$ – $\forall_и$: 1

⎢13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8

⎢14. $P(y_1)^{M2}$ – $\&_и$: 11

⎢15. $B(y_1)^{M2}$ – $\&_и$: 11

⎢16. $P(y_2)^{M2}$– $\&_и$: 12

⌊17. $B(y_2)^{M2}$ – $\&_и$: 12

18. $\neg\neg B(y_2)^{M0}$ – $\neg_в$: 9, 12

19. $B(y_2)^{M2}$ – $\neg_и$: 18

20. $\forall xB(x)$ – $\forall_в$: 19; y_2 – абс. огр.

Поскольку текущая цель является подцелью достигнутой цели, то она также является достигнутой, согласно пункту 11.2. К формулам 8 и 20 применяется правило

$\&_в$, в ПВ помещается формула $\forall x P(x) \ \& \ \forall x Q(x)$, цель $\forall x P(x) \ \& \ \forall x Q(x)$ помечается как достигнутая и устраняется из ПЦ.

Теперь вывод выглядит следующим образом:

ПВ ПЦ

1. $\forall x(P(x) \ \& \ Q(x))$ – посылка

⌈2. $\neg P(y_1)$ – пос.

| 3. $P(y_1) \ \& \ Q(y_1)^{M0}$ – $\forall_и$: 1

| 4. $P(y_1)^{M2}$ – $\&_и$: 3

⌊5. $Q(y_1)^{M2}$ – $\&_и$: 3

6. $\neg\neg P(y_1)^{M0}$ – $\neg_в$: 2, 4

7. $P(y_1)^{M2}$ – $\neg_и$: 6

8. $\forall x P(x)$ – $\forall_в$: 4; y_1 – абс. огр.

⌈9. $\neg Q(y_2)$ – пос.

| 10. $P(x_2)$ – $\forall_и$: 8

| 11. $P(y_1) \ \& \ Q(y_1)^{M0, M7}$ – $\forall_и$: 1

| 12. $P(y_2) \ \& \ Q(y_2)^{M0, M7}$ – $\forall_и$: 1

| 13. $P(y_2)^{M0, M7}$ – $\forall_и$: 8

| 14. $P(y_1)^{M2}$ – $\&_и$: 11

| 15. $B(y_1)^{M2}$ – $\&_и$: 11

| 16. $P(y_2)^{M2}$– $\&_и$: 12

⌊17. $B(y_2)^{M2}$ – $\&_и$: 12

18. $\neg\neg B(y_2)^{M0}$ – $\neg_в$: 9, 12

19. $B(y_2)^{M2}$ – $\neg_и$: 18

20. $\forall x B(x)$ – $\forall_в$: 19; y_2 – абс. огр.

21. $\forall x P(x) \ \& \ \forall x Q(x)$ – $\&_в$: 8, 20

Т.к. достигнутая цель является главной, то вывод построен.

Глава 4. Анализ алгоритма поиска вывода в системе BMV

§ 4.1. Семантическая непротиворечивость алгоритма

В предыдущей главе детально описан алгоритм поиска натурального вывода в системе BMV. Возникает вопрос о семантической непротиворечивости алгоритма, т.е. не возникнет ли ситуация, что алгоритм докажет формулу, которая не является общезначимой?

Есть несколько способов доказательства теоремы о семантической непротиворечивости алгоритма. Например, Дж. Поллок использует исчисление предикатов второго порядка [Pollock].

Другой способ состоит в доказательстве, что все выводы, которые строит алгоритм, суть выводы в системе BMV. Учитывая, что система BMV семантически непротиворечива, получаем, что алгоритм также семантически непротиворечив.

Для доказательства такого утверждения необходимо ввести ряд новых понятий, центральным из которых является понятие алгоритмического BMV-вывода.

Алгоритмическим BMV-выводом называется пара <A, B>, где

(A) непустая последовательность формул вывода (ПВ), для которой верно:

(i) Каждая формула в ПВ является или начальной посылкой, или посылкой, полученной по ц-правилам, или формулой, полученной из предыдущих по одному из BMV-правил, причем, правила введения применяются следующим образом:

- Правило $\&_в$ применяется к формулам D и B, если в ПВ есть D, формула B есть текущая достигнутая цель и предыдущей целью по отношению к D и B является формула D & B;[13]
- 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 $C \supset B$;

[13] Понятие «достигнутая цель», «предыдущая цель», «главная цель» подробно рассматриваются в описании алгоритма.

- 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 противоречие является C;
- Правило $\exists_в$ применяется к формуле D(β), если D(β), где β – новая неабсолютная переменная, является текущей достигнутой целью и предыдущей целью по отношению к D(β) является формула $\exists\alpha$D(α);
- Правило $\forall_в$ применяется к формуле D(β), где β – новая абсолютно ограниченная переменная, если D(β) является текущей достигнутой целью и предыдущей целью по отношению к D(β) является формула $\forall\alpha$D(α).

(ii) 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;

(iii) Ни одна переменная не ограничивается более одного раза;

(iv) Ни одна переменная в результате унификации не ограничивает сама себя;

и (B) непустая последовательность формул целей (ПЦ), в которой каждая цель получена или по вц-правилам или по одному из ц-правил из предыдущих целей.

В дальнейшем мы будем называть алгоритмический BMV-вывод *алго-выводом*.

Алго-выводом формулы A из (возможно, пустого) множества посылок Г называется *конечный* алго-вывод, в котором

- Последовательность формул вывода есть BMV-вывод формулы A из Г;
- Формула A является *главной* целью и A достигнута.

Согласно вышеприведенному определению, последовательность формул вывода в алго-выводе A из Г есть BMV-вывод формулы A из Г. Возникает вопрос, является ли этот BMV-вывод завершенным или нет? Положительный ответ на этот вопрос дает следующая

Лемма 4.1.1. В алго-выводе формулы A из (возможно, пустого) множества посылок Г последовательность формул вывода есть *завершенный* BMV-вывод A из Г.

Доказательство. Напомним, что BMV-вывод является завершенным, если ни одна переменная, абсолютно ограниченная в этом выводе, не входит свободно ни в множество Г, ни в формулу A.

Согласно описанию алгоритма (пункт 1), все свободные переменные из Г, А помечаются как абсолютно ограниченные.

Поскольку ни одна переменная в алго-выводе А из Г не ограничивается абсолютно более одного раза, ни одна свободная переменная из Г, А не ограничиваются абсолютно в данном алго-выводе, а значит, ни одна переменная, абсолютно ограниченная в этом выводе, не входит свободно ни в Г, ни в А.

Доказано.

В главе 2 нашей работы показана семантическая непротиворечивость исчисления BMV. Значит, семантическая непротиворечивость алгоритма устанавливается следующей

Теоремой 4.1.2. В алго-выводе формулы А из (возможно, пустого) множества Г формула А семантически следует из Г.

Доказательство: из леммы 4.1.1 и теоремы 2.2.4.

Доказано.

В процессе построения алго-вывода зафиксируем каждое применение вц-правил. Интуитивно представим процесс построения алго-вывода как ряд применений вц-правил. Такие последовательности формул и целей в алго-выводе, полученные в результате применения одного вц-правила вплоть до другого применения вц-правила, будем называть *блоками*.

В общем случае блок может состоять только из последовательности формул вывода (последовательность целей пуста, если блок получен по вц-правилу $\forall_{вц}$). Т.к. понятие блока фундаментально для нашего исследования, дадим строгое его определение.

Определение 4.1.3. *Блоком* α_i, $i \in N$, является непустая последовательность формул вывода $\{A_1, A_2, \ldots, A_n\}$ и (возможно, пустая) последовательность целей $\{\partial_1, \partial_2, \ldots, \partial_k\}$:[14]

(i) Если ∂_1 есть *главная цель алго-вывода*, то в этом случае $i = 1$; Если ∂_1 получена по одному из вц-правил, то в этом случае $i > 1$;

(ii) Каждая цель блока, кроме ∂_1, получена по ц-правилам;

(iii) Последней целью блока является либо противоречие, либо формула из ПЦ, которая унифицируется с некоторой формулой из ПВ.

(iv) Каждая формула из последовательности вывода в α_i либо является *начальной* посылкой, либо получена по ц-правилу из цели, содержащейся в α_i (в этом случае формула является посылкой), либо есть результат применения BMV-правила.

Введенное понятие блока в совокупности с вц-правилами определяет в дальнейшем понятие дерева поиска вывода в нашем исчислении.

Процесс поиска вывода представляется в виде дерева, вершинами которого являются блоки, а ребрами – вц-правила, по которым из одного блока получается другой блок.

В качестве примера приведем алго-вывод $((p \supset q) \supset p) \supset p$ (вертикальная черта обозначает границу между блоком №1 и блоком №2).

⌜1.	$(p \supset q) \supset p$	– пос.	$((p \supset q) \supset p) \supset p$
⎢⌜2.	$\neg p$	– пос.	P
⎢⎢			$\perp$

⎢⎢⌜3.	p	– пос.	$p \supset q$
⎢⎢⎢⌞4.	$\neg q$	– пос.	Q
⎢⎢⎢ 5.	$\neg\neg q$	– $\neg_в$: 2, 3	$\perp$
⎢⎢⌞6.	q	– $\neg_и$: 5	
⎢⎢ 7.	$p \supset q$	– $\supset_в$: 6	
⎢⌞8.	p	– $\supset_и$: 1, 7	
⎢ 9.	$\neg\neg p$	– $\neg_в$: 2, 8	
⌞10.	p	– $\neg_и$: 9	
11.	$((p \supset q) \supset p) \supset p$	– $\supset_в$: 10	

Комментарий. Помещаем исходную формулу в качестве *главной* цели в последовательность целей (ПЦ). При этом последовательность вывода (ПВ) пуста. Применяя к главной цели ц-правило $\supset_ц$, получаем подцель p и добавляем в ПВ формулу $(p \supset q) \supset p$ в качестве последней посылки. Поскольку ни одно правило исключения в ПВ неприменимо, продолжаем работать с *текущей* целью p. Применяем к ней правило $\neg_ц$, получаем новую подцель $\perp$ (противоречие) и добавляем в ПВ формулу $\neg p$ в качестве последней посылки. Поскольку ни одно правило исключения в ПВ

[14] Напомним, что целью может быть цель противоречие, которая не является формулой.

неприменимо, алгоритм осуществляет поиск формул из ПВ, к которым можно применить вц-правила. 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-правило $\supset_{вц}$ и начинаем строить новый блок №2. При этом формулы $(p \supset q) \supset p$ и $\neg p$ из ПВ, а также цели $((p \supset q) \supset p) \supset p$, p и $\perp$ из ПЦ образуют блок №1.

Итак, по вц-правилу $\supset_{вц}$, примененному к формуле $(p \supset q) \supset p$, мы получаем первую цель блока №2 – формулу $(p \supset q)$. Применяя ц-правило $\supset_{ц}$ к ней, получаем подцель q и добавляем в ПВ формулу p в качестве последней посылки. Т.к. текущая цель – формула q – недостижима, применяем к ней ц-правило $\neg_{ц}$, получаем подцель $\perp$ и добавляем в ПВ формулу $\neg q$ в качестве последней посылки. Далее действуем согласно правилам системы BMV и целям алгоритма. При этом формулы 3-11, а также цели $p \supset q$, q и $\perp$, находящиеся ниже вертикальной черты, образуют блок №2.

§ 4.2. Свойства алгоритма

Конечность длины произвольного блока устанавливается

Леммой 4.2.1. В произвольном алго-выводе каждый блок конечен.

Доказательство: полная индукция по порядковому номеру блока 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 ц-правил). 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 подцелей блока α_1 достигнута, то обязательно достигается главная цель A. Тем самым данный блок конечен, завершаясь выводом A.

Если же ни одна из 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 неисключённых формул в алго-выводе для формулы A, где α_{n-1} – последний блок с последней целью противоречие. 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-правила, по которому был получен блок α_n: 1) блок α_n получен по вц-правилам $\vee_{вц}$, $\supset_{вц}$ и $\neg_{вц}$; 2) блок α_n получен по вц-правилу $\forall_{вц}$.

Случай 1. Пусть формула Y является первой целью в блоке α_n и формула Y получена по вц-правилам $\vee_{вц}$, $\supset_{вц}$ и $\neg_{вц}$. Последовательность целей в α_n конечна (свойство ц-правил). 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 (по индуктивному предположению, 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 α_n и формула Y получена по вц-правилу $\forall_{вц}$. Последовательность целей в α_n пуста, т.к. по вц-правилу $\forall_{вц}$ в последовательность целей не добавляются новые цели. Значит, в последовательность вывода не вводятся новые посылки. Количество применений правил исключения в блоке α_n конечно (по индуктивному предположению, 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 $\forall_{вц}$, 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 заканчивая формулой $\neg Z$, 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 $\perp$.

Доказано.

Разбив алго-вывод на последовательность блоков, перейдем к анализу блоков.

Определение 4.2.2. Если блок α получен в результате применения $\neg_{вц}$, $\supset_{вц}$ и $\vee_{вц}$, то α является *достигнутым* (д-блоком), если α – последний блок алго-вывода и последняя цель в α достигнута. Если блок α получен в результате применения $\forall_{вц}$, то α является *достигнутым* (д-блоком), если α – последний блок алго-вывода и первая формула в α исключена.

В противном случае блок является *недостигнутым*.

Определение 4.2.3. 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 $\neg_{вц}$, $\forall_{вц,}$ то α_n *сильно* непосредственно следует за α_k. Если блок α_n получен по $\supset_{вц}$, $\vee_{вц}$, 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 $\neg_{вц}$, означает, что в последовательности формул вывода имеются формулы A (первая цель α_n) и $\neg A$ (посылка правила $\neg_{вц}$), а это автоматически ведет к достижению последней цели $\perp$ в блоке α_k. (Правило $\neg_{вц}$ применяется только, если последней целью в α_k является $\perp$.) При этом достижение последней цели в α_k влечет достижение первой цели в α_k. Значит, α_k – д-блок.

Что касается $\forall_{вц}$, то, по определению д-блока, α_n достижим, если из последовательности вывода исключена первая формула A из α_n. Исключение формулы A возможно только при применении $\neg_в$, $\supset_в$; причем, формула C (последняя

неисключенная посылка в последовательности вывода), исключаемая в результате применения правил $\neg_в$, $\supset_в$, находится выше, чем формула А. Т.к. 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 не стать д-блоком.

Зададим формальное понятие поискового дерева (П-дерева) для алго-вывода.

Определение 4.2.4. *Поисковым деревом* является частично упорядоченное множество $\{\alpha_1, \alpha_2,\ldots\}$ блоков, распределенных по непересекающимся «уровням» следующим образом:

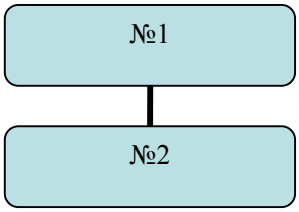
1. Нулевой уровень состоит из единственного блока α_1, называемого *начальным* блоком;
2. Каждый блок i+1-го уровня соединён отрезком в точности с одним блоком i-го уровня;
3. Каждый блок α i-го уровня либо не соединён ни с одним блоком i+1-го уровня, либо соединён отрезками с несколькими блоками i+1-го уровня (эти блоки i+1-го уровня называются *непосредственно следующими* за блоком α);
4. Блок i-го уровня, не соединённый ни с какими блоками i+1-го уровня, называется *заключительным* блоком.

Структура алго-вывода представляет собою *поисковое* дерево (П-дерево), вершинами которого являются блоки, а ребрами – правила взятия цели по формуле вывода (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. Таким образом, понятие «П-дерево» является *обобщением* понятия «алго-вывод».

Приведем некоторые П-деревья и соответствующие им алго-выводы.

Пример 1.

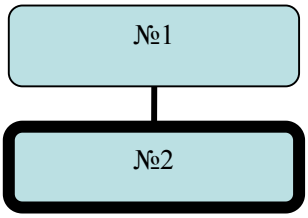


В данном П-дереве блок №2 непосредственно следует из блока №1.

Такому П-дереву соответствует, например, следующий алго-вывод (вертикальная черта обозначает границу между блоком №1 и блоком №2):

1.	$p \vee q^{M7}$	– пос.	$q \supset r$
2.	q	– пос.	r
3.	$\neg r$	– пос.	$\perp$
4.	p	– пос.	$\neg p^{M7}$ $\perp$

Пример 2.



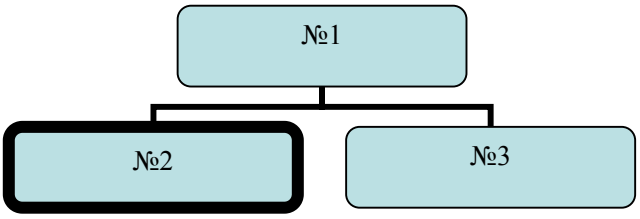
В данном П-дереве: (а) блок №2 непосредственно следует из блока №1; (б) блок №2 является д-блоком (обозначается жирным контуром) и, значит, (с) алгоритм посредством бэктрэкинга вернулся на блок №1, который не является д-блоком.

Такому П-дереву соответствует, например, следующий алго-вывод (вертикальная черта обозначает границу между блоком №1 и блоком №2):

1.	$(p \vee q) \supset q^{M0, M7}$	– пос.	$q \supset r$
2.	q	– пос.	r
3.	$\neg r$	– пос.	$\perp$
4.	$p \vee q^{M0}$	– $\vee_в$: 2.	$p \vee q^{M7}$
5.	q^{M2}	– $\supset_и$: 1, 4	q^{M4}

Отметим, что формула $p \vee q$ отмечена меткой M0, сигнализирующей, что эта формула получена по $\vee_в$ из формулы q. Значит, к этой формуле нельзя применять вц-правило $\vee_{вц}$ и помещать новую цель $\neg p$ в последовательность целей.

Пример 3.



В данном П-дереве после того, как алгоритм посредством бэктрэкинга вернулся на блок №1 (блок №2 обозначен жирным контуром), был порожден блок №3, который непосредственно следует из блока №1.

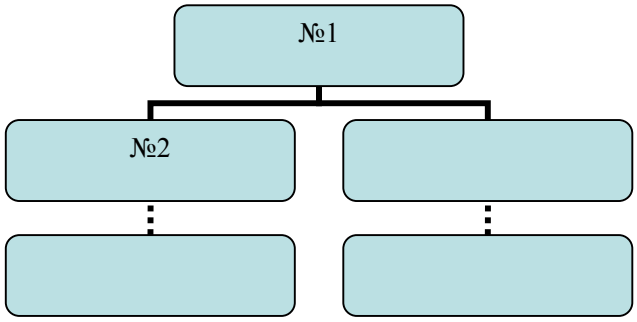
Такому П-дереву соответствует следующий алго-вывод (вертикальная черта обозначает границу между блоком №1 и блоком №2 и между блоком №2 и блоком №3):

1.	$(p \vee q) \supset q^{M0, M7}$	– пос.	$q \supset \neg(r \supset p)$
2.	q	– пос.	$\neg(r \supset p)$
3.	$r \supset p^{M7}$	– пос.	$\perp$
4.	$p \vee q^{M0}$	$-\vee_в$: 2.	$p \vee q^{M7}$
5.	q^{M2}	$-\supset_и$: 1, 4	q^{M4}
6.	$\neg r$	– пос.	r^{M7}
			$\perp$

Отметим, что, как и в предыдущем примере, формула $p \vee q$ отмечена меткой M0, сигнализирующей, что эта формула получена по $\vee_в$ из формулы q. Значит, к этой формуле нельзя применять вц-правило $\vee_{вц}$ и помещать новую цель $\neg p$ в последовательность целей.

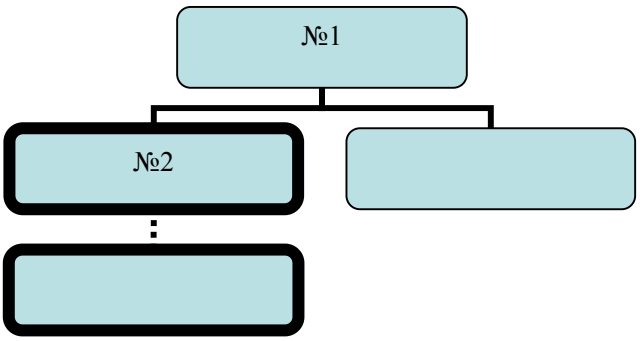
Приведем некоторые примеры деревьев, которые не могут быть П-деревьями.

Пример 4.



Данное дерево содержит блок, из которого непосредственно следует две бесконечные (обозначается пунктирной линией) нити, состоящие из недостигнутых блоков.

Пример 5.



Данная схема означает дерево, содержащее нить, расположенную *правее* бесконечной нити.

В классической и интуиционистской пропозициональных логиках [Шангин], [Шангин1], [Шангин3] *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 множество неисключенных *литералов* (пропозициональные переменные и их отрицания в формуле А) образует *контрпример*. Таким образом, данные процедуры являются разрешающими.

Ситуация в первопорядковой логике сложнее. В силу результата А. Черча 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 [Church], [Church1].

С другой стороны, Ж. Эрбран показал, что любая общезначимая формула логики предикатов доказуема за конечное число шагов [Минц], [Li]. В такой ситуации встает проблема выбора стратегии, позволяющей за конечное число шагов вывести произвольную общезначимую формулу классической логики предикатов. Такие стратегии известны: метод резолюции, метод аналитических таблиц, секвенциальные исчисления, метод семантических таблиц и др.

Анализируя данную ситуацию, Ч. Чень и Р. Ли пишут: «Тем не менее существуют алгоритмы поиска доказательства, которые могут подтвердить, что формула общезначима, если она на самом деле общезначима. Для необщезначимых формул эти алгоритмы, вообще говоря, не заканчивают свою работу. Принимая во внимание результат Черча и Тьюринга, это лучшее, что мы можем ожидать от алгоритма поиска доказательства» [Чень и Ли, с. 52]. Обратной стороной таких техник является «уход в бесконечность» в некоторых случаях, когда формула необщезначима.

Вообще, с точки зрения прикладной логики, доказательство полноты прувера – дело «неблагодарное». Распространена точка зрения, что пруверы (в том числе для поиска натурального вывода) создаются под решение каких-то отдельных прикладных проблем.[15] Поэтому проблема полноты для них не ставится.

В Главе 2 мы уже касались этого вопроса. Здесь подчеркнем, что процедура поиска вывода должна задаваться таким образом, чтобы гарантировать построение контрпримера (множества Хинтикки) даже в случае «ухода в бесконечность» [Hintikka1], [Sieg].

Определение 4.2.5. Линейно упорядоченная последовательность блоков α_1, α_2, ... называется *нитью* данного П-дерева, если: 1) α_1 – *начальный* блок и 2) $\forall\alpha_n$ ($n > 1$ и α_n *непосредственно следует* за α_{n-1}).

Напомним, что интуитивно алго-вывод представляется как последовательность блоков, полученных по одному из вц-правил. В такой ситуации большую роль играют формулы вывода, к которым применимы вц-правила (вц-формулы). Такими формулами являются формулы вида $\neg A$, $A \supset B$, $A \vee B$, не

[15] С помощью программы ANDP Д. Ли дал первое машинное доказательство проблемы остановки машины Тьюринга [Li]. Д. Пеллетье с помощью программы THINKER предложил доказательства теорем теории множеств [Pelletier]. Предложенная Д. Поллоком с целью продемонстрировать преимущества натурального вывода перед другими логическими системами программа OSCAR оказалась в 40 раз быстрее, чем программа OTTER, основанная на методе резолюции [Pollock].

отмеченные меткой M0, т.е. к ним не применялись правила исключения, и формулы вида $\forall\alpha A(\alpha)$. К этим и только этим формулам применяются вц-правила.

В процессе поиска вывода вц-правила применяются только к вц-формулам. Поиск вывода осуществляется путем выведения в ПВ подформул имеющихся в ПВ формул. С одной стороны, выведение подформул осуществляется с помощью правил исключения. Однако не ко всем формулам вывода применимы правила исключения. В такой ситуации, с другой стороны, при поиске вывода необходимо применять вц-правила, которые представляют собой переходы от формулы к ее подформулам.[16]

Поэтому нижеследующее определение выделяет роль таких формул в процессе поиска вывода.

Определение 4.2.6. 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 $G_1, G_2,\ldots, G_s, H_1, H_2,\ldots, H_t$ – все такие вц-формулы; причем, $G_1, G_2,\ldots, G_s$ – формулы вида $\neg A$, $A \supset B$, $A \vee B$, к которым не применялись правила исключения, и $H_1, H_2,\ldots, H_t$ – формулы вида $\forall\alpha A(\alpha)$, к которым применимо правило $\forall_{вц}$.

Множество E^j_n конечно для любых конечных j, n, т.к. все блоки П-дерева конечны (лемма 4.2.1).

В свою очередь, конечность множества E^j_n позволяет эффективно определять формулу (формулы, если их несколько) наибольшей степени из E^j_n.

Тогда *степень* множества E^j_n (обозначается «$g(E^j_n)$») есть степень формулы наибольшей степени из E^j_n.

Используя понятие П-дерева, опишем процесс поиска алго-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:

[16] Исключение: переход от $A \vee B$ к $\neg A$ по $\vee_{вц}$, если B – элементарная формула. Тогда $g(A \vee B) = g(\neg A)$.

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 формуле H_1 из множества E^0_1 строится новый блок α_2; переход к 5.

5. Проверяется, является ли α_2 д-блоком.

Если да, то возвращаемся к α_1.

Если нет, то рассматривается множество $E^0_2 = \{G'_1, G'_2,\ldots, G'_s, H'_1, H'_2,\ldots, H'_t\}$. Множество E^0_2 отличается от множества $E^0_1 = \{G_1, G_2,\ldots, G_s, H_1, H_2,\ldots, H_t\}$ тем, что, с одной стороны, некоторые формулы, которые входили в E^0_1 и были вц-формулами, таковыми не являются, а, с другой стороны, в последовательности вывода могут появиться новые вц-формулы, которые и войдут в E^0_2.

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 «backtracking») 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, т.к., 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 $\forall_{вц}$), 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.

С помощью понятия *порождающего* множества (Определение 4.2.6) мы исследуем взаимоотношения между формулами, которые входят в блоки, (непосредственно) следующие друг за другом в одной нити П-дерева. Тот факт, что при увеличении числа блоков в нити степень новых порождающих множеств не увеличивается, устанавливется

Леммой 4.2.7. Пусть α_n непосредственно следует за α_k, $k < n$, и E_n, E_k – соответственно, порождающие множества из α_n и α_k, тогда $g(E_n) \leq g(E_k)$.

Доказательство: возвратная индукция по количеству n блоков, непосредственно следующих за α_k.

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 вц-правило при порождении α_n и M – первая цель (формула) в α_n. Значит, по свойству вц-правил, $g(M) < g(G)$. По свойству ц-правил, $g(M') < g(M)$, где M' – произвольная (не 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 $X \in E_k$.[17]

Итак, добавленные к E_k 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.

Доказано.

§ 4.3. Семантическая полнота алгоритма

Доказательство теоремы о семантической полноте алгоритма состоит из следующих этапов.

– Показываем конечность ветвления в произвольном П-дереве, т.е. из некоторого блока в П-дереве непосредственно следует конечное число блоков.

– Конечность ветвления в произвольном П-дереве позволяет применить к П-дереву лемму Кенига [König] для определения бесконечной нити в П-дереве.

– Данная бесконечная нить создается канонически: в этой нити возможно построение модельного множества [Hintikka1].

Рассматривая П-дерево отметим, прежде всего, что в общем случае для логики предикатов оно не будет конечным, как для логики высказываний [Шангин], [Шангин1].

Лемма 4.3.1. 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.

Мы допускаем, что за блоком α_k следует бесконечное число блоков.

Базис: $g = 0$.

Если $g = 0$, то порождающее множество в α_k пусто и построение новых блоков, непосредственно следующих за α_k, невозможно.

Допущение неверно. Значит, за блоком α_k следует конечное число блоков.

Индуктивное предположение: лемма верна для произвольного $g < n$.

[17] Если правило исключения двухпосылочно, то X – большая посылка.

Индуктивный шаг: лемма верна для g = n.

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;[18]

(2) все следующие в П-дереве за α_k блоки суть д-блоки, но сам α_k не является д-блоком.[19]

Из (1) и индуктивного предположения следует, что все нити в таком дереве конечны. Т.к. бесконечность длины нити в П-дереве – это результат потенциально неограниченного количества применений $\forall_{вц}$, то все непосредственно следующие за α_k блоки не были получены по правилу $\forall_{вц}$.

Более того, т.к. 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,[20] 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 имеется, 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 3.3.7, $g(E_{k+1}) \leq g(E_k)$, если в E_k входят формулы вида $\forall\alpha A(\alpha)$ или некоторые формулы максимальной степени из E_k перешли в E_{k+1}. Т.к. 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.

---

[18] Если допустить наличие бесконечного ветвления в одном блоке П-дерева, то механизм перехода от блока к блоку с помощью бэктрэкинга запрещает возникновение второго блока с бесконечным ветвлением.

[19] Автомат 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.

[20] В противном случае достижимость этих блоков привела бы к достижимости α_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.

Доказано.

Прежде чем перейти к доказательству теоремы о семантической полноте алгоритма, зададим некоторые определения [Непейвода].

Определение 4.3.2. П-дерево является *достигнутым*, если все блоки этого П-дерева суть д-блоки.

Если все блоки П-дерева суть д-блоки, значит, П-дерево конечно. Таким образом, некоторому достигнутому П-дереву соответствует счетное множество конечных алго-выводов.

Поскольку все конечные алго-выводы суть *завершенные* BMV-выводы (теорема 4.1.2), каждому *достигнутому* П-дереву соответствует бесконечное множество *завершенных* BMV-выводов.

Теперь нужно ответить на вопрос, множество каких алго-выводов соответствует недостигнутому П-дереву.

Определение 4.3.3. Назовем нить φ *незапертой*, если или φ бесконечна или заключительный блок α_k из φ не является д-блоком и порождающее множество формул блока α_k пусто.

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 4.3.4. П-дерево является *недостигнутым*, если оно содержит незапертую нить.

После того, как мы установили конечность ветвления в произвольном П-дереве (лемма 4.3.1), можно применить *лемму Кенига*, согласно которой бесконечное П-дерево с конечным ветвлением содержит бесконечную нить [König], [Непейвода].

Напомним, что в П-дереве бесконечная нить, если она существует, единственна. Поэтому зададим построение этой нити так, чтобы она всегда содержала опровергающую модель. Т.е. множество формул, принадлежащих этой нити, должно образовывать множество Хинтикки [Hintikka], [Hintikka1].

Пусть φ – незапертая (конечная или бесконечная) нить в П-дереве, Σ – множество неисключенных формул вывода, принадлежащих φ, и T(Σ) – множество термов из формул в Σ. Тогда справедлива

Лемма 4.3.5. Для любых формул A и B из Σ верно:

1. (а) $A \in \Sigma \Rightarrow \neg A \notin \Sigma$,

   (б) $\neg A \in \Sigma \Rightarrow A \notin \Sigma$,
2. $\neg\neg A \in \Sigma \Rightarrow A \in \Sigma$,
3. $A \,\&\, B \in \Sigma \Rightarrow A \in \Sigma$ и $B \in \Sigma$,
4. $\neg(A \,\&\, B) \in \Sigma \Rightarrow \neg A \in \Sigma$ или $\neg B \in \Sigma$,
5. $A \vee B \in \Sigma \Rightarrow A \in \Sigma$ или $B \in \Sigma$,
6. $\neg(A \vee B) \in \Sigma \Rightarrow \neg A \in \Sigma$ и $\neg B \in \Sigma$,
7. $A \supset B \in \Sigma \Rightarrow \neg A \in \Sigma$ или $B \in \Sigma$,
8. $\neg(A \supset B) \in \Sigma \Rightarrow A \in \Sigma$ и $\neg B \in \Sigma$,
9. $\forall\alpha A(\alpha) \in \Sigma \Rightarrow A(\alpha/t) \in \Sigma$, для *всех* t из T(Σ),
10. $\neg\forall\alpha A(\alpha) \in \Sigma \Rightarrow \neg A(\alpha/t) \in \Sigma$, для *некоторого* t из T(Σ),
11. $\exists\alpha A(\alpha) \in \Sigma \Rightarrow A(\alpha/t) \in \Sigma$, для *некоторого* t из T(Σ),
12. $\neg\exists\alpha A(\alpha) \in \Sigma \Rightarrow \neg A(\alpha/t) \in \Sigma$, для *всех* t из T(Σ).

Доказательство.

(1а) По условию, φ незаперта, значит, она содержит только недостигнутые блоки. Причем, последней целью в этих блоках является цель противоречие. Значит, эта цель не достигнута, т.е. в множестве вывода нет противоречащих формул.

(1б) Аналогично (1а).

(2) По BMV-правилу $\neg_{и}$.

(3) По BMV-правилу $\&_{и}$.

(4) Если $\neg(A \,\&\, B) \in \Sigma$, то по правилу $\neg_{вц}$ 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, т.к. A & B недостижима и φ незаперта.)

(5) Если $A \vee B \in \Sigma$ и $\neg A \in \Sigma$, то формула B выводима по $\vee_{и}$. Если $A \vee B \in \Sigma$ и $\neg A \notin \Sigma$, то возможно два подслучая. (а) Либо к формуле $A \vee B$ применяется вц-правило

$\vee_{вц}$ и текущей целью становится формула $\neg A$. Из нее по правилу $\neg_{ц}$ вводим формулу $\neg\neg A$ в качестве посылки в множество Σ. Из $\neg\neg A$ по $\neg_{и}$ получаем A. (б) Либо формула $A \vee B$ получена по $\vee_{в}$ из формулы A (B).

(6) По BMV-правилу $\neg\vee_{и}$.

(7) Если $A \supset B \in \Sigma$ и $A \in \Sigma$, то формула B выводима по $\supset_{и}$. Если $A \supset B \in \Sigma$ и $A \notin \Sigma$, то к формуле $A \supset B$ применяется вц-правило $\supset_{вц}$ и текущей целью становится формула A. Из нее по правилу $\neg_{ц}$ вводим формулу $\neg A$ в качестве посылки в Σ.

(8) Аналогично (4).

(9) Если $\forall\alpha A(\alpha) \in \Sigma$, то квантор общности снимается по новой неабсолютной переменной x; в алго-выводе появляется формула A(x) и на формулу $\forall\alpha A(\alpha)$ ставится метка M1, запрещающая применение к этой формуле правила $\forall_{и}$. Данная метка в дальнейшем снимается, если: (а) все формулы вида $\forall\alpha A(\alpha)$ 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 $\forall\alpha A(\alpha)$ из множества Σ применяется правило $\forall_{вц}$. Согласно формулировке этого правила, квантор общности снимается по некоторым определенным термам из множества Σ, т.е. по всем термам из Л(Σ), по которым квантор общности еще не снимался.

(10) Если $\neg\forall\alpha A(\alpha) \in \Sigma$, то по правилу $\neg_{вц}$ текущей целью становится формула $\forall\alpha A(\alpha)$. Применяем к этой цели правило $\forall_{ц}$ и получаем формулу A(y), где y – новая абсолютная переменная. Т.к. A(y) недостижима, то применяем к ней правило $\neg_{ц}$ и в множество Σ вводится $\neg A(y)$ в качестве посылки. При этом на формулу $\neg\forall\alpha A(\alpha)$ ставится метка M5, не позволяющая еще раз применить к ней правило $\neg_{вц}$.

(11) Если $\exists\alpha A(\alpha) \in \Sigma$, то по правилу $\exists_{и}$ получаем формулу A(y), где y – новая абсолютная переменная. При этом на формулу $\exists\alpha A(\alpha)$ ставится метка M0, не позволяющая еще раз применить к ней правило $\exists_{и}$. Если на формуле $\exists\alpha A(\alpha)$ уже стоит метка M0, таким образом, $\exists\alpha A(\alpha)$ получена по $\exists_{в}$ из формулы A(t), для некоторого терма t.

(12) Аналогично (9): к формуле $\neg\exists\alpha A(\alpha)$ применяется правило $\neg\exists_{и}$ и выводится формула $\forall\alpha\neg A(\alpha)$.

Доказано.

Определение 4.3.6. *Литералами* для формулы А называется множество элементарных формул или их отрицаний в А.

Теорема 4.3.7. Если формула А семантически следует из (возможно, пустого) множества посылок Г, то существует алго-вывод А из Г.

Доказательство. Покажем, что справедлива контрапозиция данного утверждения, т.е. если не существует алго-вывода А из Г, то А семантически не следует из Г.

Если не существует алго-вывода А из Г, то имеется *недостигнутое* П-дерево (связь алго-вывода и П-дерева).

По определению 4.3.4, недостигнутое П-дерево содержит *незапертую* нить φ. Множество формул вывода Σ из φ является множеством Хинтикки (лемма 4.3.5).

Значит, множество *литералов* из Σ представляет собой такую интерпретацию, при которой все формулы из Г истинны, а формула А ложна [Hintikka1]. Таким образом, если не существует алго-вывода А из Г, то А семантически не следует из Г.

Доказано.

Теорема 4.3.8. Система BMV семантически полна.

Доказательство: из Теоремы 4.3.7 и того факта, что всякий алгоритмический вывод есть завершенный вывод в системе BMV.

Доказано.

Заключение

Диссертационное исследование посвящено автоматическому поиску натурального вывода типа Куайна в классической логике предикатов. Специфика данной системы натуральной вывода – наличие *прямого* правила удаления квантора существования и наличие абсолютно и относительно ограниченных переменных.

Отсюда следует, что в общем случае между посылками и заключением вывода не имеет место отношение логического следования, поскольку формулировка прямого правила удаления квантора существования позволяет от общезначимых посылок переходить к необщезначимым заключениям.

Для обеспечения корректности системы наряду с понятием вывода (доказательства) в системе (Определение 2.1.3) вводится понятие *завершенного* вывода (*завершенного* доказательства), т.е. такого вывода (доказательства), в неисключенные посылки и заключение которого не входит ни одна абсолютно ограниченная переменная данного вывода (доказательства).[21]

Относительно завершенного вывода (завершенного доказательства) в системе натурального вывода предлагается доказательство утверждения о семантической непротиворечивости (Теорема 2.2.4), т.е. в произвольном завершенном выводе (доказательстве) между посылками и заключением имеет место отношение логического следования. Таким образом, всякая формула, доказуемая в системе, общезначима.

Доказательство теоремы о семантической непротиворечивости системы опирается на предложенное У. Куайном доказательство теоремы о семантической непротиворечивости.

Отметим, что в системе У.Куайна (а значит, в предложенном им доказательстве теоремы о семантической непротиворечивости) существенным образом используется алфавитный порядок, заданный на множестве используемых в языке переменных.

Система BMV не предполагает наличие алфавитного порядка на множестве используемых в выводе переменных. Поэтому доказательство теоремы о семантической непротиворечивости системы натурального вывода, предложенное У. Куайном, не обобщается на систему BMV.

[21] Определение 2.1.4.

В связи с этим вводится понятие *пассивной* переменной в BMV-выводе (доказательстве), т.е. такой абсолютно ограниченной переменной в BMV-выводе (доказательстве), которая не ограничивает относительно ни одну абсолютно ограниченную переменную данного BMV-вывода (доказательства).[22]

Показывается, что в произвольном алго-выводе всегда найдется пассивная переменная (Лемма 2.2.4).

Далее предлагается алгоритм поиска вывода в данном исчислении, который является модификацией алгоритма поиска натурального вывода, разработанного В.А. Бочаровым, А.Е. Болотовым и А.Е. Горчаковым.

С использованием теоремы о семантической непротиворечивости системы натурального вывода BMV показывается, что данный алгоритм обладает свойством семантической непротиворечивости, поскольку каждый вывод (доказательство), полученный алгоритмом, является выводом (доказательством) в системе BMV (Теорема 4.1.2).

Понятие вывода (доказательства) в системе BMV предполагает, что в выводе (доказательстве) ни одна переменная не ограничивает сама себя. Переменная ограничивает другую переменную согласно формулировкам правил $\forall_в$, $\exists_и$.

Экспликация отношения ограничения показывает, что данное отношение, заданное на множестве переменных вывода (доказательства), обладает свойствами *иррефлексивности* (ни одна переменная не ограничивает сама себя) и *транзитивности* (если переменная x ограничивает переменную y и переменная y ограничивает z, то переменная x ограничивает переменную z).

Таким образом, отношение ограничения, заданное на множестве переменных вывода (доказательства), является отношением строгого (частичного) порядка.

В силу того, что теория строгого порядка разрешима, процедура проверки, ограничивает ли произвольная переменная сама себя, *конечна* для произвольного завершенного вывода (доказательства).

Встроенный в алгоритм поиска вывода стандартный алгоритм унификации адаптирован для работы с абсолютно и относительно ограниченными переменными и содержит вышеупомянутую процедуру поиска в выводе (доказательстве) переменной, которая ограничивает сама себя.

[22] Определение 2.2.1.

Минимальной единицей алгоритмического вывода является *блок* – непустая, конечная последовательность формул. Последовательность блоков образует собой древовидную структуру – дерево поиска вывода, в котором переход от одного блока к другому осуществляется с помощью правил поиска вывода.

Показывается конечность ветвления для произвольного блока в произвольном дереве поиска вывода (Лемма 4.3.1).

Опираясь на представление алгоритмического натурального вывода в виде древовидной структуры, выделяется некоторая нить данного дерева, множество формул в которой образует множество Хинтикки (*модельное* множество).

Таким образом, если для некоторой выводимости формулы из (возможно, пустого) множества посылок невозможно построить алгоритмический вывод, то данная формула логически не следует из данного множества посылок и алгоритмический вывод содержит (возможно, бесконечную) контрмодель, т.е. такую интерпретацию, при которой все формулы из данного множества посылок принимают значение «истина», а данная формула принимает значение «ложь».

Отсюда следует по контрапозиции, что предложенный алгоритм поиска натурального вывода типа Куайна в классической логике предикатов первого порядка обладает свойством семантической полноты, т.е. для любой общезначимой формулы классической логики предикатов можно построить вывод в предложенном алгоритме (Теорема 4.3.7).

Поскольку всякий алгоритмический вывод есть вывод в системе BMV, из утверждения о семантической полноте алгоритма следует утверждение о семантической полноте системы BMV (Теорема 4.3.8).

В ходе диссертационного исследования обнаружены следующие проблемы, требующие дальнейшей разработки:

a. теорема о семантической полноте системы BMV тривиально следует из теоремы о семантической полноте алгоритма поиска вывода в системе BMV. Однако остается невыясненной возможность прямого, а не косвенного доказательства теоремы о семантической полноте системы BMV (например, установлением факта, что все, что доказуемо в стандартном гильбертовском исчислении, имеет завершенное доказательство в системе BMV).

b. обобщение изложенного алгоритма и прямого метода доказательства теоремы о семантической полноте на неклассические логики предикатов (интуиционистскую, релевантную и др.). Учитывая, что пропозициональный вариант алгоритма для классической логики обобщается на интуиционистскую логику высказываний, выдвигается гипотеза, что указанные в работе методы применимы к неклассическим логикам предикатов.

c. решение для предложенного алгоритма проблемы поиска *минимальных* контрмоделей: если некоторая формула имеет как конечную, так и бесконечную контрмодель, то в процессе поиска вывода алгоритм не всегда предлагает конечную контрмодель. При этом предполагается, что построение алгоритмом бесконечной контрмодели в случае, если имеется возможность построения конечной контрмодели, неэффективно с точки зрения вычислимости.

d. создание машинной реализации для данного алгоритма в виде компьютерной программы, которая позволит облегчить усвоение и запоминание основ дедукции.

Литература

- [Andrews] Andrews, P. Transforming matings into natural deduction proofs // 5th Conference on Automated Deduction, 1980.
- [Basin et al] Basin, D., Matthews, S. and L. Vigano. Natural deduction for non-classical logics // *Studia Logica*, vol. 60, №1, 1998.
- [Bocharov et al] Bocharov, V., Bolotov, A., Gorchakov, A. and V. Shangin. 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, 2003.
- [Bochmann] Bochmann, G. Hardware specification with temporal logic: an example // *IEEE Transactions on computers*, vol. C-31, №3, 1982.
- [Bolotov & Fisher] Bolotov, A. and M. Fisher. A resolution method for computational tree branching time temporal logic // IV International workshop on temporal representation and reasoning (TIME'97). Florida, 1997.
- **[**Byrnes] Byrnes, J. Proof search and normal forms in natural deduction. PhD thesis, Pittsburgh, 1999.
- [Church] Church, A. A note on the Entscheidungsproblem // *The Journal of Symbolic Logic*, vol. 1, №1, 1936.
- [Church1] Church, A. Correction to A note on the Entscheidungsproblem // *The Journal of Symbolic Logic*, vol. 1, №3, 1936.
- **[**Copi] Copi, I. Symbolic logic. 3rd ed. New-York, London, 1967.
- **[**Fitch] Fitch, F. Symbolic logic. New York, 1952.
- [Hintikka] Hintikka, J. A new approach to sentential logic // *Societas Scientarium Fennica Commentationes Physico-Mathematicae* XVII, 2, 1957.
- [Hintikka1] Hintikka, J. Notes on the quantification theory // *Societas Scientarium Fennica Commentationes Physico-Mathematicae* XVII**,** 11, 1957.
- **[**Jaskowski] Jaskowski, S. On the rules of suppositions in formal logic // *Studia Logica*, №1, 1934.
- [Kalish] Kalish, D. Review of Copi: Symbolic logic. 3rd ed. New-York, London, 1967 // *The Journal of Symbolic Logic*, vol. 39, №1, 1974.

- [König] König, D. Theorie der endlichen und unendlichen graphen, Akademische Verlagsgesellschaft M.B.H., Leipzig, 1936.
- [Li] Li, D. Unification algorithms for eliminating and introducing quantifiers in natural deduction automated theorem proving // *Journal of Automated Reasoning*, vol. 18, №1, 1997.
- [Pelletier] Pelletier, F.J. Automated natural deduction in THINKER // *Studia Logica*, vol. 60, №1, 1998, доступна по адресу: http://www.cs.ualberta.ca/~jeffp/.
- [Pelletier1] Pelletier, F.J. A brief history of natural deduction // *History and Philosophy of Logic*, vol. 20, 1999, доступна по адресу: http://www.cs.ualberta.ca/~jeffp/.
- [Pelletier2] Pelletier, F.J. Seventy-five graduated problems for testing automatic theorem provers // *Journal of Automated Reasoning*, 1986.
- [Pelletier3] Pelletier, F.J. Errata for 75 problems // *Journal of Automated Reasoning*, № 4, 1988.
- [Pollock] Pollock, J. Skolemization and unification in natural deduction, неопубликованная версия статьи доступна по адресу: http://oscarhome.soc-sci.arizona.edu/ftp/publications.html.
- [Portoraro] Portoraro, F. Strategic constructions of Fitch-style proofs // *Studia Logica*, vol. 60, №1, 1998.
- [Quine] Quine, W. On natural deduction // *The Journal of Symbolic Logic*, vol. 15, №2, 1950.
- [Robinson] Robinson, J. A machine-oriented logic based on the resolution principle // *Journal of the ACM*, vol. 12, №1, 1965.
- [Sieg] Sieg, W. Mechanism and search: aspects of proof theory. Pittsburgh, 1992.
- [Sieg & Byrnes] Sieg, W. and J. Byrnes. Normal natural deduction proofs (in classical logic) // *Studia Logica*, vol. 60, №1, 1998.
- [Анисов] Анисов А.М. Современная логика. М., ИФРАН, 2003.
- [Болотов и др.] Болотов А.Е., Бочаров В.А., Горчаков А.Е. Алгоритм поиска вывода в классической логике предикатов // *Логические исследования*. Вып. 5. М., Наука, 1998.

- [Болотов и др.1] Болотов А.Е., Бочаров В.А., Горчаков А.Е. Алгоритм поиска вывода в классической пропозициональной логике // *Труды научно-исследовательского семинара логического центра Института философии РАН*. М., ИФРАН, 1996.
- [Болотов и др.2] Болотов А.Е., Бочаров В.А., Горчаков А.Е., Макаров В.В., Шангин В.О. Пусть докажет компьютер // *Логика и компьютер*. Вып. 5. М., Наука, 2004. (Серия «Кибернетика – неограниченные возможности и возможные ограничения».)
- [Бочаров и Маркин] Бочаров В.А., Маркин В.И. Основы логики. М., Космополис, 1994.
- [Бочаров] Бочаров В.А. Исчисление предикатов с универсалиями (II. Семантика) // *Логические методы в компьютерных науках. (Труды научно-исслед. семинара по логике Ин-та философии АН СССР*); Сб. ст. / Редкол.: Смирнов В.А. (отв. ред.) и др. М., ИФАН, 1991.
- [Братко] Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. А.И. Лупенко, А.М. Степанова / под ред. А.М. Степанова. М., Мир, 1990.
- [Войшвилло] Войшвилло Е.К. Понятие. М., Изд-во МГУ, 1967.
- [Войшвилло1] Войшвилло Е.К. Процедура поиска доказательства для формул системы Е // Войшвилло Е.К. Философско-методологические аспекты релевантной логики. М., МГУ, 1989.
- [Генцен] Генцен Г. Исследования логических выводов // *Математическая теория логического вывода*: Пер. с англ. А.В. Идельсона / Под ред. А.В. Идельсона и Г.Е. Минца. М., Наука, 1967.
- [Ивлев] Ивлев Ю.В. Логика: Учебник для высших учебных заведений. 2-е изд., перераб. и доп. М., Логос, 1997.
- [Макаров] Макаров В.В. Алгоритм поиска натурального вывода для интуиционистской логики высказываний // Автореферат диссертации на соиск. учен. степ. канд. филос. наук. М., Соцветие красок, 2002.
- [Мендельсон] Мендельсон Э. Введение в математическую логику: Пер. с англ. Ф.А. Кабакова / Под ред. С.И. Адяна. 2-е изд., испр. М., Наука, 1976.
- [Минц] Минц Г.Е. Теорема Эрбрана // *Математическая теория логического вывода*: Под ред. А.В. Идельсона и Г.Е. Минца. М., Наука, 1967.

- [Непейвода] Непейвода Н.Н. Прикладная логика: Учебное пособие. 2-е изд., испр. и доп. Новосибирск, Изд-во Новосиб. ун-та, 2000.
- [Смирнов] Смирнов В.А. Теория логического вывода. М., РОССПЭН, 2000.
- [Смирнов и др.] Смирнов В.А., Маркин В.И., Новодворский А.Е., Смирнов А.В. Доказательство и его поиск (курс логики и компьютерный практикум) // *Логика и компьютер*. Вып. 3. М., Наука, 1996. (Серия «Кибернетика – неограниченные возможности и возможные ограничения».)
- [Смирнов-мл.] Смирнов А.В. Система интерактивного доказательства теорем // *Логические исследования*. Вып. 2. М., Наука, 1993.
- [Смирнов-мл.1] Смирнов А.В. Язык описания логических систем для автоматического поиска доказательства // Автореферат диссертации в виде научного доклада на соиск. учен. степ. канд. филос. наук. М., 1998.
- [Чень и Ли] Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. Пер. с англ. / Под ред. С.Ю. Маслова. М., Наука, 1983.
- [Шангин] Шангин В.О. 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, 2000.
- [Шангин1] Шангин В.О. 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, 2002.
- [Шангин2] Шангин В.О. Метатеоретические свойства натурального вывода // *Материалы IV Международной конференции «Смирновские чтения»*. М., ИФРАН, 2003.
- [Шангин3] Шангин В.О. Автоматический поиск натурального вывода в интуиционистской логике и проблема дубликации // *Аспекты*, Том 2. М., Современные тетради, 2003.
- [Шанин] Шанин Н.А., Давыдов Г.В., Маслов С.Ю., Минц Г.Е., Оревков В.П., Слисенко А.О. Алгорифм машинного поиска естественного логического вывода в исчислении высказываний. Л., Наука, 1964.