Поиск натурального вывода в системах негативной силлогистики

А. В. Красненкова

abstract. The paper presents an algorithm, dealing with process of natural deduction in the systems of negative syllogistic of different styles, i.e. Aristotelian, fundamental, traditional, etc. The rules of inference in a system under investigation are defined by the user. Thus the algorithm can be considered as a synthesis of so called proof checkers and automatic systems of natural deduction. Such an approach allows using genetic method to make the proposed algorithm applicable to a wide class of formal syllogistics.

В данной работе дается описание алгоритма, осуществляющего автоматический поиск выводов и доказательств в натуральных системах негативной силлогистики (НС), базирующихся на аксиоматических силлогистических системах, построенных В.А. Бочаровым и В.И. Маркиным, и вместе с тем приводится генетическое описание процесса автоматического поиска доказательств.

Построенный алгоритм является своего рода синтезом алгоритмов, приведенных в [1] и [2].

Если обобщить все используемые алгоритмом системы, то язык, с которым он работает, можно сформулировать следующим образом:

1. S, P, M, Q, S_1, P_1, M_1, Q_1, $S_2\ldots$ — список индивидных силлогистических переменных общего типа.
2. a, e, i, o — логические(силлогистические) константы.
3. $\&$, $\vee$, $\supset$, $\neg$ — логические(пропозициональные) константы.
4. $\sim$ — логическая (силлогистическая) константа.
5. $(,)$ — технические символы.

Будем использовать в качестве метапеременных, пробегающих по множеству индивидных силлогистических переменных общего типа, бесконечный список вида X, Y, Z, X_1, Y_1, Z_1, $X_2 \ldots$

Будем использовать в качестве метапеременных, пробегающих по множеству силлогистических формул, бесконечный список вида A, B, C, A_1, B_1, C_1, $A_2 \ldots$

Эту же символику предполагается использовать для записи правил вывода той или иной силлогистической системы, предназначенной для нашего алгоритма.

Теперь введем понятие *силлогистического терма* и *силлогистической формулы* для допустимых силлогистических систем.

1. Любая индивидная силлогистическая переменная общего типа есть силлогистический терм.

2. Если X — силлогистический терм, то $\sim X$ — силлогистический терм.

3. Ничто иное не есть силлогистический терм.

Понятие *силлогистической формулы.*

1. Если X — терм и Y — терм, то XaY, XeY, XiY, XoY — силлогистические формулы.

2. Если A — силлогистическая формула, то $\neg A$ — силлогистическая формула.

3. Если A,B — силлогистические формулы, то $A\&B$, $A \vee B$, $A \supset B$ — силлогистические формулы.

4. Ничто иное не является силлогистической формулой.

Формулы, задаваемые пунктом 1, называются простыми силлогистическими формулами; формулы, определяемые пунктами 2–3, договоримся называть сложными, или пропозициональными, формулами.

Внешние формульные скобки договоримся опускать.

На настоящий момент алгоритм может работать в таких системах негативной силлогистики, как традиционная негативная силлогистика, фундаментальная негативная силлогистика, негативная силлогистика Аристотеля, Больцано и Кэрролла. Также

ведутся исследования по работе алгоритма в сингулярных негативных силлогистиках оккамовского и аристотелевского типов.

1 Общая характеристика алгоритма

На мой взгляд, значимым фактом является то, что правила поиска вывода практически не отличаются от правил вывода в исследуемых натуральных силлогистических исчислениях. Это экономящее время работы алгоритма и делающее его более компактным отличие основано на значительном сокращении множества меток. Метки, введенные и используемые в [2], не применяются. Таким образом, у нас в качестве меток используется конечное подмножество множества целых чисел, а также метка «-», которая применяется при «вычерпывании» из множества формул вывода только тех формул, которые необходимы и достаточны для вывода или доказательства.

Алгоритм принципиально строится как открытая система, к которой пользователь может приложить потенциально бесконечное множество текстовых пакетов, содержащих иные силлогистические правила вывода. К данным пакетам предъявляется только одно требование: все они должны быть написаны на указанном выше стандартном формализованном языке, чтобы алгоритм смог их прочитать и начать корректную работу в предложенных ему дедуктивных системах. Помимо открытого уровня дедуктивных правил в нашем алгоритме существует еще один открытый уровень — уровень сложности используемых программой эвристик.

Эвристики позволяют «просеивать» формулы, создающие пространства состояний, и «высаживать» в поисковое поле программы только полезные, «творческие» формулы, отбрасывая те, которые «прорастут сорняками», т.е. формулы лишние и тормозящие работу компьютера.

Разъясним данное утверждение. Если у нас в произвольной системе есть правило, которое позволяет наращивать термную сложность заключения по сравнению с термной сложностью посылок, то такое правило позволяет строить потенциально бесконечные объекты, если его ничем не ограничить сверху. Например, возьмем правила $XaY \vdash Xe \sim Y$ и $XY \vdash X \sim Y$ и посылку SaM. С помощью этих правил получаем такой ряд

конструктивных объектов: $Se \sim M$, $S \sim\sim M$, $Se \sim\sim\sim M$, $Sa \sim\sim\sim\sim M$ и т. п. Очевидно, что данный ряд стремится к бесконечности и делает невозможной работу любого алгоритма. Помня о том, что не следует умножать сущности без необходимости, мы можем ввести следующее аналитическое ограничение на объекты такого вида: глубина данных объектов не должна превышать некоторого заданного значения специальной переменной. Обозначим указанную специальную переменную как n. Она может принимать значения от 0 до k, $k \in N$. Тогда нулевое значение переменной n будет означать самое простое, минимальное пространство состояний, на котором проводится поиск вывода или доказательства (или соответственно, опровержения некорректной формулы); на правила построения конструктивных объектов указанного типа налагается запрет. При $n = 1$ данное пространство состояний расширяется и в него включаются формулы первого уровня повышенной термной сложности (т.е. формулы, полученные в результате однократного применения таких конструктивных правил и содержащие не более 1 символа термного отрицания). При $n = 2$ поисковое пространство состояний расширяется еще больше, и теперь допускается использование формул второго уровня термной сложности. Вообще, при $n = k$ в зону поиска включаются формулы k-ого уровня сложности, построенные с помощью правил описанного выше типа. В программной реализации алгоритма пользователю автоматически предлагаются эвристики уровня 0–2 в виде флажков выбора. Если данных эвристик недостаточно, пользователь может сам задать эвристику необходимого термного уровня сложности. В принципе, можно индуктивно доказать лемму о том, что максимальная термная сложность, необходимая и достаточная для вывода, доказательства или опровержения формулы, будет равна $\mathbf{m} + k$, где $\mathbf{m}$ — максимальное количество термных отрицаний у некоторого терма, входящего в состав исследуемой формулы, а k — соответственно количество термных отрицаний в заключении того из правил вывода, повышающих термную сложность формулы, которое имеет наибольшее число термных отрицаний.

Использование данных эвристических приемов позволяет существенно уменьшить перебор на поисковом пространстве со-

стояний и тем самым значительно повысить эффективность программы. Обработка правил вывода системы, введенных пользователем в текстовый редактор — Блокнот или MS Word, — происходит согласно концепции В.А. Смирнова. Это означает, что правила вывода той или иной системы делятся на *аналитические* и *синтетические*. Напомним, что аналитические правила вывода конструируют вспомогательные подвыводы на базе основного вывода, а синтетические правила позволяют добавлять в основной вывод новые формулы, и таким образом данные правила осуществляют синтез. Данное деление происходит по следующей схеме: в группу аналитических правил автоматически заносятся правила, в которых *пропозициональная* сложность посылок выше или равна пропозициональной сложности заключения, а в группу синтетических правил, — наоборот, те правила, где пропозициональная сложность заключения выше пропозициональной сложности посылок.

Раскроем понятие пропозициональной сложности формулы, так как оно несколько отличается от общепринятого. Похожая трактовка приведена в [3].

Обозначим функцию оценки пропозициональной сложности произвольной силлогистической формулы буквой φ. Пусть ее аргументами будут произвольные формулы, а значениями — элементы множества рациональных чисел.

Определим данную функцию индуктивно:

1. $\varphi(A) = 0$, где A — произвольная простая силлогистическая формула любой степени силлогистической сложности.
2. $\varphi(\neg A) = \varphi(A) + 0,5$;
3. $\varphi(A \supset B) = \varphi(A\&B) = \varphi(A) + \varphi(B) + 1$;
4. $\varphi(A \vee B) = \varphi(A) + \varphi(B) + 1,5$.

Определим отношение порядка $<$ на множестве формул. Для этого введем функцию Trunc (от англ. truncate — «срезать верхушку», «усекать»), находящую целую часть от аргумента, которая от любого числа с дробным «хвостом» оставляет только целую часть.

Итак, $A < B \Leftrightarrow Trunc(\varphi(A)) < Trunc(\varphi(B))$, где A и B — произвольные формулы.

Например, оценим при помощи введенных функций пропозициональное правило вывода $\neg(A \vee B) \vdash A\&B$. $[Trunc(\varphi(\neg(A \vee B))) = Trunc(\varphi(A \vee B)+0,5) = Trunc(\varphi(A)+\varphi(B)+1,5+0,5) = Trunc(0+0+1,5+0,5) = Trunc(2) = 2] = [Trunc(\varphi(\neg A\&\neg B)) = Trunc(\varphi(\neg A)+\varphi(\neg B)+1) = Trunc(\varphi(A)+\varphi(B)+0,5+0,5+1) = Trunc(0+0+0,5+0,5+1) = Trunc(2) = 2]$. Следовательно, согласно введенному нами критерию, это правило алгоритма включается в группу аналитических правил вывода. Рассмотрим еще один пример — правило $XaY \vdash \neg XoY$. $[Trunc(\varphi(XaY)) = Trunc(0) = 0] = [Trunc(\varphi(\neg XoY)) = Trunc(\varphi(XoY)+0,5) = Trunc(0+0,5) = Trunc(0,5) = 0]$. Таким образом, данное правило тоже оценивается алгоритмом как аналитическое, что позволяет сохранить логику «естественного» вывода, а также избежать введения лишних целей.

Теперь введем функцию оценки термной сложности произвольной силлогистической формулы. Данная функция понадобится нам для использования специальных эвристик, о которых будет сказано ниже. Обозначим указанную функцию через ψ.

Определим ее индуктивно:

1. $\psi(X \circ Y) = 0$, где X, Y — простые силлогистические термы, а $\circ$ — одна из допустимых силлогистических констант;
2. $\psi(\alpha \circ \sim \beta) = \psi(\alpha \circ \beta) + 1$, где α, β — произвольные силлогистические термы;
3. $\psi(\neg\alpha \circ \beta) = \psi(\alpha \circ \beta) + 1$;
4. $\psi(\neg A) = \psi(A)$, где A — произвольная силлогистическая формула;
5. $\psi(A\&B) = \psi(A \supset B) = \psi(A \vee B) = \psi(A) + \psi(B)$, где A и B — произвольные силлогистические формулы.

Для примера оценим при помощи функции ψ формулу $Sa \sim\!\!\sim P$. $\psi(Sa \sim\!\!\sim P) = \psi(Sa \sim P)+1 = \psi(SaP)+1+1 = 0+1+1 = 2$.

После разделения правил на аналитические и синтетические с помощью функции φ, оценивающей пропозициональную сложность формулы, алгоритм проводит дополнительный анализ правил внутри группы синтетических и аналитических правил вывода (далее сокращенно — п.в.). Данный анализ необходим для корректного поиска выводов и доказательств.

Суть его для синтетических п.в. состоит в следующем: синтетические правила вывода дополнительно подразделяются на прямые и непрямые. Делается это чисто формальным способом: алгоритм «смотрит», входит ли в заключение правила подформула C, которая заранее, до начала работы алгоритма, является условным обозначением подформулы, входящей только в непрямые правила вывода. Пользователь может по своему желанию переопределить символические обозначения для выражений подобного рода.

Все правила вывода, не имеющие подформулы C, автоматически классифицируются как прямые правила вывода.

Аналитические правила вывода, в свою очередь, делятся на правила вывода с посылками и беспосылочные правила вывода (беспосылочным, к примеру, является правило вывода $\vdash XaX$).

Затем алгоритм формирует две последовательности формул — *стек целей* и *стек вывода*, на которых базируется вся дальнейшая работа.

Стек вывода (далее СВ) представляет собой натуральный вывод в той или иной допустимой силлогистической системе. Здесь следует указать, что полученный таким образом вывод является избыточным, т.е. ряд применений правил НС (в особенности это касается так называемых силлогистических правил, т.е. правил, работающих с внутренней структурой элементарных, или простых, формул) является лишним, порождая неиспользуемые в процессе дедукции формулы. Такая ситуация возникает в силу того, что алгоритм должен рассмотреть все допустимые варианты для получения максимально точного и корректного результата.

Контроль над построением вывода и связанных с ним подвыводов осуществляется стеком целей. Стек целей (далее СЦ) строится благодаря синтетическим правилам поиска вывода. Цель, стоящая первой в стеке целей, называется главной целью, а остальные — вспомогательными. Главная цель представляет собой формулу, которую требуется вывести или доказать. В последнем случае формула называется теоремой рассматриваемой системы. Цель, обрабатываемая алгоритмом в некоторый момент программного

времени, называется текущей. Главная цель тоже может быть текущей.

2 Алгоритмические процедуры

Процедура 1, или процедура достижимости

Проверяет достижимость текущей цели. Если текущая цель является формулой, то для того чтобы считать ее достигнутой, необходимо, чтобы некоторая неисключенная формула из стека вывода графически совпадала с ней. Если текущая цель является противоречием, то с ней работает процедура 2, или процедура унификации.

Процедура 2, или процедура унификации

Работает с текущей целью, содержащей метапеременные. Процедура 2 пытается найти формулу или кортеж формул, являющийся результатом правильной подстановки в выражение, содержащее метапеременные. Например, когда текущей целью является противоречие, записываемое как «$\neg B, B$», процедура 2 ищет в СВ пары формул, являющиеся результатом правильной подстановки в «$\neg B, B$», например $< \neg SaP, SaP >$. В случае неудачи процедура 2 возвращает пустой кортеж.

Процедура 3, или процедура применения беспосылочных аналитических правил вывода

Процедура 3 применяет беспосылочные правила вывода, последовательно подставляя все простые термы обосновываемой выводимости или теоремы в беспосылочные правила и добавляя полученные формулы в СВ. Например, применяя правило $\vdash a$ к теореме $\vdash SaP \supset Se \sim P$ в системе традиционной негативной силлогистики, мы добавляем в СВ формулы SaS и PaP.

Чтобы избежать вхождения в цикл, каждый терм используется ровно один раз. Контроль за использованием термов осуществляется с помощью специально введенной строки термов, из которой термы вычеркиваются по мере их обработки. Когда строка становится пустой, действие процедуры 3 заканчивается.

Процедура 4, или процедура применения аналитических правил с посылками

Заключается в последовательном применении всех аналитических правил, имеющих посылки, ко всем неисключенным формулам СВ. Чтобы избежать образования циклов, вводится спе-

циальный счетчик, который запрещает добавлять более одного раза ту же самую формулу в СВ. После каждой серии применения данной группы аналитических правил вычисляется приращение числа формул СВ. Когда это приращение становится равным 0, работа процедуры 4 завершается.

Здесь также действуют описанные выше эвристики допустимого уровня сложности формулы. Когда при помощи функции ψ алгоритм обнаружил правило, попадающее в зону действия этих эвристик, то он оценивает результат применения данного правила посредством функции ψ, и если ее значение превышает допустимое целочисленное значение для заданного пользователем типа эвристик, то полученная формула удаляется. Если формула, полученная по произвольному аналитическому п.в., добавляется в СВ, то номер этого правила записывается в специальную аналитическую строку *Static*, с помощью которой идет построение результирующего вывода.

Процедура 5, или процедура порождения новых целей с помощью синтетических правил вывода

Данная процедура строит новые цели меньшей степени пропозициональной сложности, чем формулы-цели, предшествующие ей. Процедура 5 начинает свою работу, когда главная цель — теорема или формула, которую необходимо вывести из посылок, — не достижима непосредственно, при помощи процедур 3 и 4. Процедура 5 меняет местами посылки и заключение в синтетических п.в. и затем пытается последовательно применить данные «обращенные» правила к текущей цели. Если некоторое синтетическое п.в. оказалось применимо к текущей цели, то результат его применения, являющийся либо формулой, либо кортежем формул, либо записью с метатермами, добавляется в СЦ, а на рассматриваемую цель ставится метка с номером примененного синтетического п.в.

Использование меток — целых чисел — необходимо, чтобы вести контроль за полным и последовательным применением данной группы п.в.

Если алгоритм идентифицировал применяемое правило как непрямое, то дополнительно в СВ добавляется подформула текущей цели, которая соответствует метатерму C непрямого правила (о данном метатерме говорилось выше). С добавленной в

СВ подформулой ассоциируется некоторый объект (термин из языка программирования), чтобы обеспечить корректную субординацию главного вывода и всех его подвыводов.

После каждого применения процедуры 5 применяется процедура 1 для проверки достижимости новой текущей цели. Если она оказывается достигнутой, то происходит обрыв работы процедуры 5; в противном случае процедура продолжает анализ СЦ. В случае применения непрямого п.в., когда в СВ, по сути, начинается построение нового подвывода, в действие включается также процедура 4.

Анализ СЦ, в случае недостижимости текущей цели, осуществляется процедурой 5 до тех пор, пока новой текущей целью не станет выражение, содержащее метатермы.

Процедура 6, или процедура построения новых целей по формулам СВ

Процедура 6 формирует новые вспомогательные цели на базе формул СВ и аналитических правил вывода с количеством посылок, превосходящим или равным 2 (обычно это правила исключения импликации и дизъюнкции). Процедура 6 включается в работу, когда процедура 5 естественным образом остановилась. Последняя цель, полученная данной процедурой, фиксируется алгоритмом. Сам алгоритм не ставит на нее никакой метки, но мы для удобства изложения обозначим ее знаком $\oplus$. Значок $\oplus$ разделяет цели, полученные по «обращенным» синтетическим правилам вывода, и цели, добавленные в СЦ с помощью аналитических правил вывода. Данное разделение необходимо для корректной работы алгоритма.

Процедура 6 просматривает аналитические правила вывода и, как я уже упоминала, отбирает те из них, где число посылок равно или больше 2. Затем при помощи функции φ данная процедура выбирает посылку наименьшей пропозициональной сложности и делает ее новой текущей целью (например, при работе с правилом исключения дизъюнкции вида $A \vee B$, $\neg A \vdash B$ такой целью становится формула $\neg A$).

Если на основе исследуемого аналитического правила можно сформировать несколько новых целей, то данные цели рассматриваются последовательно, по мере необходимости (например, если с помощью первой цели мы не достигли желаемого резуль-

тата, то мы добавляем в СЦ вторую по счету цель и т.п.). Каждая добавляемая в СЦ цель тестируется специальными счетчиками: она не должна находиться в СЦ ниже $\oplus$, а также не должна находиться в СВ, даже будучи исключенной. Указанные ограничения помогают избежать лишних циклов при работе алгоритма. Данные действия производятся алгоритмом до тех пор, пока приращение формул СВ не станет нулевым.

Если в процедуру 6 поступил сигнал о том, что текущая цель достигнута, происходит остановка ее работы.

Если же в результате работы процедуры 6 мы получили пустой кортеж формул или перебор новых целей, полученных с помощью аналитических правил вывода, не дал результата, то процедура 6 естественным образом заканчивает свою работу.

Процедура 7, или процедура работы с достигнутыми целями

Процедура 7 оперирует с достигнутыми целями СЦ.

Если текущая цель является формулой, то алгоритм анализирует крайнюю правую метку, стоящую на данной цели. Если эта метка является номером прямого синтетического правила вывода, то формула освобождается от меток и добавляется в СВ; параллельно она удаляется из СЦ, и алгоритм переходит к следующей цели. Если же указанная метка является номером непрямого синтетического п.в., то к текущей цели применяется данное п.в. так, как мы привыкли при «естественном» построении вывода. Здесь играют важную роль *объекты*, ассоциированные с посылками, полученными по непрямым п.в. Когда в программу поступает сигнал о необходимости применения непрямого правила вывода, то процедура 7 находит в СВ среди формул, которым сопоставлен объект, формулу с максимальным номером, разрушает связанный с ней объект, отмечает вертикальной чертой как заблокированные все формулы, начиная с указанной и заканчивая последней в СВ, а затем добавляет в СВ результат применения данного непрямого правила. Текущая цель удаляется из СЦ, и алгоритм обрабатывает новую текущую цель.

Если текущая цель является кортежем формул, то из данного списка удаляется первый элемент, а следующий за ним элемент объявляется новой текущей целью, к которой применяется процедура достижимости. Если она выдала положительный резуль-

тат, то дальше процедура 7 работает согласно описанной выше схеме. В противном случае происходит выход из процедуры 7.

Если текущая цель является противоречием, то применяется процедура унификации. Если данная процедура на выходе дала непустой кортеж формул, то цель считается достигнутой. После чего данная цель удаляется из СЦ, и алгоритм приступает к работе со следующей целью.

Во всех случаях, когда происходит пополнение СВ новыми формулами, в специальную строку *Static* записываются номера примененных синтетических п.в., для того чтобы обеспечить корректное построение результирующего вывода.

Если алгоритм добавил в СВ главную цель и СЦ представляет собой пустой список, то теорема объявляется доказанной (выводимость обоснованной), и программа переходит к построению результирующего вывода.

Процедура 8, или процедура работы с недостигнутыми целями

Процедура 8 начинает свою работу, когда текущая цель считается недостигнутой алгоритмом. Если текущая цель является противоречием, то она сразу же удаляется из СЦ. При этом происходит удаление из СВ связанного с ней подвывода, т.е. всех формул, начиная с первой с конца формулы СВ, которой сопоставлен некоторый объект (т.е. начиная с последнего допущения в выводе), и заканчивая последней формулой СВ. При этом связанный с указанным допущением объект разрушается. Из *Static* вычеркиваются номера всех удаленных формул. Происходит переход к следующей цели.

Если текущая цель является кортежем формул, то она удаляется из СЦ. Алгоритм переходит к следующей цели.

Если текущая цель является формулой, то алгоритм работает следующим образом. Если данная формула была получена по непрямому п.в., то из СВ удаляется связанный с ним подвывод по схеме, описанной выше. Если нет, то алгоритм с помощью меток, стоящих на формуле-цели, проверяет, все ли синтетические п.в. были применены к ней. Если не все, то происходит обрыв работы процедуры 8 (алгоритм попытается получить данную цель по другим синтетическим п.в.). Если же к данной цели были применены все допустимые синтетические правила вывода, то

текущая цель удаляется из СЦ. Происходит переход к следующей цели.

В общем, процедура 8 тестирует все возможные варианты получения некоторой цели (обычно при работе со сложными дизъюнктивными целями), строя, таким образом, древовидную структуру целей. Тупиковые ветви удаляются из СЦ и СВ.

Если алгоритм удалил главную цель, то формула объявляется опровергнутой. Происходит выход из алгоритма.

Процедура 9, или процедура построения результирующего вывода

Процедура 9 строит результирующий вывод из исходного по методу, напоминающему работу в секвенциальных исчислениях. Вначале процедура 9 помечает меткой «-» последнюю формулу СЦ. Затем она выбирает из *Static* соответствующее ей число (т.е. номер правила, по которому она была получена; заметим, что между элементами *Static* и формулами СЦ существует взаимнооднозначное соответствие), находит по нему синтетическое п.в., «обращает» его и применяет к отмеченной формуле.

На выходе мы получаем либо кортеж формул, либо кортеж выражений с метапеременными.

В первом случае алгоритм поэлементно анализирует кортеж и отмечает знаком «-» формулы СВ, эквиморфные исследуемому элементу.

Во втором случае алгоритм применяет процедуру унификации. Если процедура унификации, в свою очередь, на выходе предложила несколько вариантов кортежей формул (т.е. данный переход в выводе может быть осуществлен несколькими способами), то алгоритм выбирает из них самый экономный, т.е. такой, элементы которого имеют самые маленькие номера и наименьшую структурную сложность.

Далее выбранный кортеж (или, если полученный кортеж изначально единственный, то он не подвергается процедуре отбора) обрабатывается описанным выше способом.

Процедура 9 прорабатывает все «-»-отмеченные формулы до тех пор, пока они не закончатся.

Затем отмеченные формулы отбираются, с них снимаются метки «-», а полученной последовательности формул сопостав-

ляется анализ, и данная формульная последовательность с анализом объявляется результирующим выводом.

Заметим, что у нас не может возникнуть такой ситуации, когда «обращенное» п.в. или процедура унификации дала на выходе пустую строку, или какой-либо элемент кортежа не был найден в СВ, так как в *Static* были записаны только номера реально примененных к формулам СВ правил, заключения которых были добавлены в СВ (сам смыл записи в *Static* говорит о том, что некоторая формула, полученная по некоторому п.в., была добавлена в СВ). Посылки же и допущения записываются в *Static* под нулевым номером.

Справедливость указанного факта можно доказать методом возвратной математической индукции.

3 Общее описание алгоритма

Сначала алгоритм сортирует, как уже было сказано выше, выбранные или введенные пользователем правила вывода. Затем он формирует **стек вывода** и **стек целей** и формулу, стоящую справа от знака выводимости $\vdash$, добавляет в СЦ. Если слева от знака выводимости $\vdash$ находится последовательность формул (т.е. у нас имеется не доказательство теоремы, а выводимость), то алгоритм добавляет каждую посылку в СВ.

Также алгоритм фиксирует выбранную пользователем эвристику допустимого уровня термной сложности формул. По умолчанию используется эвристика минимального уровня сложности.

I. Затем применяется процедура 1, или процедура достижимости. Если она дала положительный результат, то переход к VIII.

II. Если результат применения процедуры 1 отрицательный, то применяется процедура 3. После применения данной процедуры начинает вновь действовать процедура 1. Переход к I. Процедура 3 работает до тех пор, либо пока текущая цель не будет достигнута, либо пока она естественным образом не остановится (т.е. мы больше не можем применять данную процедуру). В последнем случае переход к III.

III. Применяется процедура 4. После естественного окончания ее работы осуществляется переход к I. Как уже было сказано выше, процедура 4 в случае недостигнутости текущей цели работает до тех пор, пока приращение числа формул СВ не станет равным нулю. В этом случае осуществляется переход к IV.

IV. В действие вступает процедура 5. Как мы уже упоминали, после каждого ее применения осуществляется переход к I, а в случае, когда она добавляет в СВ посылку нового подвывода по непрямому п.в., алгоритм переходит к III. В случае недостижимости текущей цели процедура 5 работает до тех пор, пока текущей целью не станет противоречие. Тогда алгоритм переходит к V.

V. Применяется процедура 2. Если процедура 2 обнаружила в СВ противоречие, то переход к VIII. В противном случае переход к VI.

VI. Начинает свою работу процедура 6 так, как было описано выше. После каждого пополнения СЦ алгоритм переходит к IV. Если текущая цель не достигнута, то процедура 6 применяется до тех пор, пока все возможные варианты новых целей не будут исчерпаны. Тогда осуществляется переход к VII.

VII. В действие включается процедура 8. Как мы уже говорили, если к цели, имеющей вид формулы, были применены не все допустимые правила, то алгоритм переходит к IV. В противном случае она работает до тех пор, пока либо не найдет конструктивную ветвь построения вывода (доказательства), либо пока не удалит главную цель из СЦ. Тогда процедура 8 прекращает свою работу, а алгоритм объявляет теорему не доказуемой или выводимость необоснованной.

VIII. В данном пункте описывается действие процедуры 7. Когда текущая цель является формулой, то она автоматически считается достигнутой, и она либо просто добавляется

в СВ, либо сначала применяется соответствующее п.в., после чего происходит добавление данной формулы в СВ (об условиях данного разделения см. выше).

Напомним, что после каждого пополнения СВ осуществляется переход к III.

В случае наличия цели-противоречия алгоритм переходит к V. Когда цель является кортежем формул, то новой целью становится его первый элемент и осуществляется переход к I.

Если применение процедуры 7 зашло в тупик (она не может больше применяться, а главная цель не достигнута), то алгоритм переходит к IV. Когда алгоритм обнаружил, что главная цель является достигнутой, то он переходит к IX.

IX. Применяется процедура 9 так, как мы описали выше.

На этом краткое описание алгоритма завершено.

4 Пример работы алгоритма

Докажем теорему $\vdash ((SaM \vee \neg SiS)\&(MaP \vee \neg MiM)) > (Se \sim P \vee \neg SiS)$ в системе негативной силлогистики Больцано.

Ниже курсивом выделены номера правил, которые алгоритм отсортировал как *синтетические*, а полужирным курсивом — как *синтетические непрямые* правила. Остальные правила являются *аналитическими*.

Вместо импликации будем использовать знак $>$.

Итак, пусть правилами вывода в этой системе будут:

1.	$B \vdash C > B$,	10.	$\neg\neg A \vdash A$,
2.	$A \vdash A \vee B$,	11.	$XeY \vdash \neg XiY$,
3.	$B \vdash A \vee B$,	12.	$XeY \vdash XiX$,
4.	$A\&B \vdash A$,	13.	$XoY \vdash \neg XaY$,
5.	$A\&B \vdash B$,	14.	$XoY \vdash XiX$,
6.	$A, B \vdash A\&B$,	15.	$XiY \vdash XaX$,
7.	$A \vee B, \neg A \vdash B$,	16.	$XaY \vdash XiY$,
8.	$A > B, A \vdash B$,	17.	$XiY \vdash YiX$,
9.	$\neg(A \vee B) \vdash \neg A\&\neg B$,	18.	$XeY \vdash YeX$,

19. $XaY \vdash Xe \sim Y$,
20. $Xe \sim Y \vdash XaY$,
21. $Xa \sim\sim Y \vdash XaY$,
22. $Xo \sim\sim Y \vdash XoY$,
23. $Xe \sim\sim Y \vdash XeY$,
24. $Xi \sim\sim Y \vdash XiY$,
25. $\neg B, B \vdash \neg C$,
26. $\neg XeY, XiX \vdash XiY$,
27. $\neg XoY, XiX \vdash XaY$,
28. $\neg XiY, XiX \vdash XeY$,
29. $\neg XaY, XiX \vdash XoY$,
30. $XaZ, ZaY \vdash XaY$,
31. $XaZ, ZeY \vdash XeY$,
32. $Xi \sim Y \vdash XoY$.

Выберем эвристику нулевого уровня сложности (от выбранной эвристики зависит количество шагов вывода и время работы программы).

Вот что на выходе нам выдала программа:

1.	$(SaM \vee \neg SiS)\&$ $(MaP \vee \neg MiM)$	Assump.
2.	$SaM \vee \neg SiS$	$A\&B \vdash A$, 1
3.	$MaP \vee \neg MiM$	$A\&B \vdash B$, 1
4.	$\neg(Se \sim P \vdash \neg SiS)$	Assump.
5.	$\neg Se \sim P\&\neg\neg SiS$	$\neg(A \vee B) \vdash \neg A\&\neg B$, 4
6.	$\neg Se \sim P$	$A\&B \vdash A$, 5
7.	$\neg\neg SiS$	$A\&B \vdash B$, 5
8.	SiS	$\neg\neg A \vdash A$, 7
9.	$Si \sim P$	$\neg XeY, XiX \vdash XiY$, 6, 8
10.	SoP	$Xi \sim Y \vdash XoY$, 9
11.	$\neg SaP$	$XoY \vdash \neg XaY$, 10
12.	$\neg\neg SaM$	Assump.
13.	SaM	$\neg\neg A \vdash A$, 12
14.	SiM	$XaY \vdash XiY$, 13
15.	MiS	$XiY \vdash YiX$, 14
16.	MaM	$XiY \vdash XaX$, 15
17.	MiM	$XaY \vdash XiY$, 16
18.	$\neg\neg MaP$	Assump.
19.	MaP	$\neg\neg A \vdash A$, 18
20.	SaP	$XaZ, ZaY \vdash XaY$, 13, 19
21.	$\neg\neg\neg MaP$	$\neg B, B \vdash \neg C$, 11, 20
22.	$\neg MaP$	$\neg\neg A \vdash A$, 21
23.	$\neg MiM$	$A \vee B, \neg A \vdash B$, 3, 22
24.	$\neg\neg\neg SaM$	$\neg B, B \vdash \neg C$, 23, 17

	25.	$\neg SaM$	$\neg\neg A \vdash A$, 24
	26.	$\neg SiS$	$A \vee B, \neg A \vdash B$, 2, 25
	27.	$\neg\neg(Se \sim P \vee \neg SiS)$	$\neg B, B \vdash \neg C$, 26, 8
	28.	$Se \sim P \vee \neg SiS$	$\neg\neg A \vdash A$, 27
	29.	$((SaM \vee \neg SiS)\&$ $(MaP \vee \neg MiM)) >$ $(Se \sim P \vee \neg SiS)$	$B \vdash C > B$, 28

Сокращение «Assump.» обозначает допущение.

Как видно из анализа вывода, программа построила 4 дополнительных подвывода. Первый подвывод образован посредством «обращенного» правила введения импликации (правило ***1*** из вышеуказанного списка), соответственно, допущение — это антецедент доказываемой теоремы. Текущей же целью стал ее консеквент. Анализируя вывод, мы замечаем, что алгоритм сумел применить к данному первому допущению только аналитические п.в., отмеченные в списке правил номерами 4 и 5. Затем он попытался доказать первый дизъюнкт консеквента теоремы. Однако, двигаясь в этом направлении, алгоритм зашел в тупик, и поэтому все вспомогательные подцели и подвыводы были удалены из СЦ и СВ. То же самое произошло и при попытке доказать второй дизъюнкт. Поэтому программа взяла в качестве допущения отрицание всей дизъюнкции (в построенном выводе оно — второе допущение). В силу того, что у нас есть правило 9, СВ пополнился формулой, пронумерованной в выводе 5. Из данной конъюнкции мы получили ряд полезных для дальнейшей дедукции формул. Так как без вспомогательных целей текущая цель не могла быть обоснована, алгоритм, используя перестроенное правило 7, ввел две дополнительные цели и, соответственно, две дополнительные посылки — $\neg\neg SaM$ и $\neg\neg MaP$ (третий и четвертый подвыводы). Последовательно применяя аналитические правила 10 и 33, мы получили противоречие — 11-й и 20-й шаги в СВ. Затем алгоритм применил правило введения отрицания (правило 28) и с помощью формул, полученных по аналитическим п.в., в том числе и на более ранних этапах дедукции, завершил и третий подвывод (получив $\neg\neg\neg SaM$). Но у нас в СВ осталось еще две противоречащих друг другу формулы — SiS и $\neg SiS$. Таким образом, мы исключаем второе на-

ше допущение и вскоре переходим к непосредственно введению импликации и достижению главной цели вывода. Теорема объявляется доказанной. Строится результирующий вывод, из СВ удаляются все лишние формулы, а оставшиеся формулы образуют замыкание по отношению зависимости, определенному в [2]. К данным формулам строится анализ, и результат выводится на экран. Время, потраченное на доказательство, — 23,5 сек. Эта длительность работы алгоритма объясняется, во-первых, тем, что алгоритм строится на метауровне, позволяющем описывать разные силлогистики, а потому каждый раз необходимо время на приспособление к выбранной системе, а во-вторых, тем, что в силлогистике имеется большое число правил, которые выдают излишние формулы.

5 О программной реализации

Программа написана на языке Object Pascal (система программирования Delphi). Данная система обладает богатыми программистскими возможностями и обширными динамическими библиотеками программных средств, позволяющими создавать приложения, работающие не только под ОС Windows, но и под ОС Linux, что не может не рассматриваться как положительный момент данной системы программирования.

Программная реализация активно опирается на понятие *генетического алгоритма.*

Под генетическим алгоритмом обычно подразумевается такой алгоритм, который моделирует своеобразный «эволюционный процесс». Делается это следующим образом: сначала вычисляется желательный результат (пусть в нашем случае таким результатом будет текущая цель) и общий критерий останова алгоритма; затем берется некоторое количество исходных формул — «начальная популяция» (в нашем случае это формулы СВ или СЦ, находящиеся в стеке на момент применения «генетической» процедуры). Следует отметить, что формулы СВ и СЦ рассматриваются как знаки некоторых организмов, а цели-противоречия — как знаки отношений, в которых находится определенная группа организмов (например, выражение $\neg B$, B может рассматриваться как кодировка отношения конкуренции).

Из «особей» начальной популяции выбираются только репродуктивные, т.е. способные давать потомство организмы, — в нашем случае алгоритм не задействует заблокированные, или пострепродуктивные формулы.

Репродуктивные «особи» исходной популяции подвергаются оценке с использованием так называемой «функции приспособленности». Числовое значение, возвращенное данной функцией, рассматривается как вероятность выживания организма, представленного некоторой формулой. «Особи», чья «приспособленность» ниже определенного вычисляемого программой критерия (организмы не «дозрели» до репродукции), удаляются из популяции. В нашем алгоритме критерием полезности служит пропозициональная либо термная оценка правила вывода, применяемая к «популяции формул» на текущий момент программного времени. Формулы с пропозициональной или термной (в зависимости от типа используемого правила) сложностью ниже указанного критерия временно изымаются из «генетического фонда».

Описанные механизмы выбора продуктивных индивидов обычно называют селекцией. У нас это будет первичная селекция.

Под вторичной селекцией будем подразумевать отбор среди репродуктивных организмов, т.е. мы отбираем «особей» для «скрещивания» — применения правил вывода.

Сначала алгоритм тестирует «особей» на предмет «генетической совместимости» с заданным критерием (критерий — применяемое правило вывода; «генетическая совместимость» — возможность применить подстановку в заключение данного правила). Данное тестирование происходит в два этапа. На первом отбираются организмы, обладающие строением того же типа, что и критерий скрещивания, т.е. их логическая структура — пропозициональная или термная — должна быть гомоморфна логической структуре посылок правила. Так мы установили «внешнее» соответствие. На втором этапе мы проверяем «внутреннее» соответствие критерия отбора из выбранного кортежа «особей»-формул: они должны обладать гомоморфной структурой входящих в их состав терминов. Проще говоря, алгоритм проверяет, имеют ли выбранные

«особи» и посылки правила-критерия одинаковую логическую форму.

Прошедшие селекцию организмы допускаются к скрещиванию. Под «скрещиванием» мы подразумеваем подстановку в заключение применяемого правила вывода. Подстановка здесь понимается как «генетический оператор», т.е. как оператор, порождающий «потомство».

Заметим, что в отличие от большинства генетических алгоритмов, где используется только некоторое количество репродуктивных «особей», но *не все* (так называемый вероятностный подход), мы отбираем абсолютно все годящиеся для «скрещивания» «организмы», так как в силу поставленной задачи нам важна абсолютная точность результата.

Полученное «потомство» (результат применения того или иного правила вывода), в свою очередь, оценивается на «жизнеспособность» (в ряде случаев, например при работе процедуры 4, «жизнеспособные организмы» должны соответствовать по термной сложности уровню выбранной пользователем эвристики или не содержать метатермов любого типа).

«Жизнеспособное потомство» создает следующее «поколение». Особи следующего поколения также, в свою очередь, оцениваются, селекционируются, скрещиваются между собой и с особями старших поколений, порождают поколение следующего уровня и т.д. Таким образом, моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет достигнут желаемый результат эволюции (достигнута текущая цель) или не выполнится общий критерий останова алгоритма.

Данный критерий таков:

1. Исчерпание числа поколений, отпущенных на эволюцию (например, приращение формул СЦ равно нулю при применении процедуры 6 — число участвующих в эволюции поколений исчерпано и т.п.).

2. Возникла «кризисная ситуация» — т.е. «особи» перестали размножаться и начали «выяснять отношения» (получение цели с метатермами при работе процедуры 5).

В последнем случае, когда эволюция временно прекратилась,

необходимо «антикризисное» решение — например, нахождение и исключение из числа репродуктивных организмов особей и их потомства, вступающих в конфликт с основным генетическим фондом популяции (применение процедуры 6 — добавление новых перспективных особей для скрещивания и, как следствие, правил введения отрицания и блокировка соответствующих формул СВ; тут же следует отметить, что целая группа поколений *принципиально* объявляется пострепродуктивной в случае применения *любого* непрямого п.в.). Если путь выхода из кризиса так и не будет найден, то придется объявить популяцию непродуктивной и не годящейся для полноценной эволюции, а теорему или выводимость — недоказуемой или необосновываемой. В частном случае (когда тупиковой является только одна ветвь доказательства) непродуктивным объявляется одно из поколений, входящих в популяцию.

Аналогичным образом работает механизм построения результирующего вывода из исходного. Программа строит «родословную» теоремы или выводимости, отбирая только те «организмы», которые действительно участвовали в процессе ее порождения. В случае, когда образуется ветвление, т.е. у формулы обнаруживается несколько потенциальных «предков», алгоритм выбирает те из них, которые обладают по возможности самыми экономными характеристиками, другими словами, сами имеют наименьшее число «предков» («жадный алгоритм»).

Таким образом, у нас задан эффективный критерий выбраковки «организмов» и отбора действительно полезных и продуктивных «особей», ведущих к вершине эволюционного процесса — доказательству теоремы или обоснованию выводимости.

Также мы достигаем по возможности оптимальной и эффективной работы нашего алгоритма, облегчая поиск решений на очень больших и сложных пространствах поиска.

6 Метатеоретические свойства алгоритма и перспективы его развития

В дальнейшем предполагается дать доказательство главных метатеоретических свойств построенного алгоритма: синтаксической и семантической непротиворечивости и полноты. При доказательстве данных фактов планируется опираться на

методы, используемые В.О Шангиным в [8] и [3] и Е.Д. Смирновой в [7].

Но уже сейчас можно утверждать, что данный алгоритм является *абсолютно непротиворечивым* в том смысле, что не все формулы произвольной силлогистической системы являются теоремами. Подробнее об этом см. [6].

Предполагается доказательство дедуктивной эквивалентности натуральных силлогистических исчислений, прилагающихся к программе в качестве текстовых пакетов, и соответствующих им аксиоматических систем. Также мы планируем улучшить эффективность работы алгоритма и устранить возможные неполадки в его работе. Дело в том, что при очень объемных пространствах поиска (например, при попытке обосновать заключение сорита из двенадцати посылок), при значительном количестве правил вывода (более 30), наличия правила или правил, увеличивающих термную сложность формул, и при необходимом выборе эвристики выше первого уровня термной сложности «биосистема» не выдерживает такого энергетического подъема и «зависает». Поэтому хотелось бы повысить порог терпимости алгоритма к скачкам в численности «особей»-формул. Однако «терпимость» системы во многом зависит от формулировки ее правил вывода, в особенности от того, является ли данная формулировка не только полной, но и *минимальной*. Например, когда у нас имеется избыточное количество правил построения формул, вызывающих «зависание» программы, то достаточно часто ситуацию можно исправить, переформулировав правила вывода таким образом, что количество правил построения формулы уменьшится, а полученная система будет дедуктивно эквивалентна исходной.

Правила вывода той или иной системы пользователь по желанию может легко изменять, а также задавать собственные системы негативной силлогистики.

Построенную программу предполагается использовать для обучающих целей, а также в качестве вспомогательного средства при работе с различными силлогистическими системами.

Литература

[1] *Буркова И.А., Деменкова Т.А., Семенов Н.В., Харитонов А.Ш., Чумакова Е.И.* Страшны ли роботам фреймы? Философия и будущее цивилиза-

ции // Тезисы докладов и выступлений IV Российского философского конгресса (Москва 24-28 мая 2005 г.). Том 1. М.: «Современные тетради», 2005.

[2] *Глушков В.М.* Введение в кибернетику. Киев: Издательство Академии Наук Украинской ССР, 1964.

[3] *Горчаков А.Е., Макаров В.В., Спирин С.В.* О некоторых особенностях построения компьютерной реализации алгоритма автоматического доказательства теорем в натуральном исчислении. http: // www.logic.ru (Логические исследования). М., 1998.

[4] Логика и компьютер. Вып. 3. Доказательство и его поиск (курс логики и компьютерный практикум). М.: Наука, 1996.

[5] Логика и компьютер. Вып. 5. Пусть докажет компьютер. М.: Наука, 2004.

[6] *Маркин В.И.* Силлогистические теории в современной логике. М.: Издательство Московского университета, 1991.

[7] *Смирнова Е.Д.* Логика и философия // Сер. «Научная философия». М.: РОССПЭН, 1996.

[8] *Чёрч А.* Введение в математическую логику. Том 1. Глава I. § 17. М.: Издательство иностранной литературы, 1960.

[9] *Шангин В.О.* Автоматических поиск натурального вывода в классической логике предикатов. Диссертация на соискание ученой степени кандидата философских наук. Специальность 09.00.07 — Логика. М., 2004.