Применение метода абстракций для поиска логического вывода в системах искусственного интеллекта

И. Л. Братчиков

Санкт-Петербургский государственный университет
199034, Санкт-Петербург, Университетская набережная, д.7/9
Igor.Bratchikov@pobox.spbu.ru

УДК 681.3
И. Л. Братчиков. **Применение метода абстракций для поиска логического вывода в системах искусственного интеллекта** // Труды СПИИРАН. Вып. 1, т. 2 — СПб: СПИИРАН, 2002.
Аннотация. *Рассматривается метод поиска логического вывода с предварительной настройкой на конкретные базы знаний. Для настройки используется абстракция и формально-грамматическая интерпретация проблемы дедукции. Метод позволяет улучшить эффективность поиска логического вывода.* — Библ. 14 назв.

UDC 681.3
I. L. Bratchikov. **Application of Abstraction Method for Search of Logical Inference in Systems of Artificial Intelligence** // SPIIRAS Proceedings. Issue 1, v. 2. — SPb: SPIIRAS, 2002.
Abstract. *A method of logical inference search with preliminary adjustment to a concrete knowledge base is considered. For the adjustment an abstraction and formal- grammar interpretation of deduction problem is used. The method allows to improve the efficiency of logical inference search.* — Bibl. 14 items.

В настоящее время наблюдается тенденция к более широкому применению средств и методов искусственного интеллекта (ИИ) в программных системах различного назначения. Существенное влияние на более широкое применение ИИ имеет объединение средств логического и объектно-ориентированного программирования, что сокращает время разработки интеллектуальных систем и значительно расширяет их выразительные возможности.

Одним из главных направлений в этой области является разработка систем ИИ, опирающихся на логическую форму представления знаний в виде баз знаний (БЗ) и использующих методы поиска логического вывода вопросов пользователей, часто называемых целями [8–10]. Наиболее широкое применение имеют резолюционные методы поиска логического вывода. К числу таких методов относится, например, входная линейная резолюция, на основе которой был разработан широко известный язык ИИ Пролог. Однако эти методы зачастую недостаточно эффективны. Они осуществляют перебор большого числа вариантов вывода, в том числе и тупиковых, а в некоторых случаях могут привести к зацикливанию. Из сказанного ясно, что исследования, направленные на сокращение перебора, устранение зацикливаний и тем самым повышение эффективности этих методов представляются весьма актуальными.

Для повышения эффективности резолюционных методов часто используются различные сокращающие перебор (суживающие) стратегии [11–13]. Большинство суживающих стратегий реализуется путем включения в инструментальные средства создания систем ИИ специфических конструкций, управляющих перебором. Как правило, такие стратегии требуют ручной разработки высококвалифицированными специалистами. Поэтому важной представляется задача создания суживающих стратегий, которые могли бы использоваться в автоматическом режиме.

В настоящей статье представлены результаты исследований, выполненных автором совместно с рядом аспирантов и студентов Санкт-Петербургского

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

государственного университета, которые направлены на создание автоматизируемой суживающей стратегии для резолюционных алгоритмов поиска логического вывода [1–7]. Основой для этой стратегии явился метод абстракций, впервые предложенный и теоретически обоснованный в [14]. Коротко этот метод можно сформулировать следующим образом. По исходной задаче **A** строится более простая задача **B** таким образом, чтобы для каждого решения задачи **A** имелось решение задачи **B**, структура которого подобна решению задачи **A**. Таким образом, найденные решения простой задачи **B** могут использоваться как некие схемы для поиска возможных решений первоначальной задачи **A** (хотя некоторые решения задачи **B** могут не привести к цели). Задача **B** строится с помощью так называемых абстракционных отображений, формально определенных в [14], и называется абстракционной задачей или абстракцией задачи **A**. Очевидно, для успешного применения данного метода необходимо, чтобы, во-первых, абстракционная задача решалась существенно проще исходной и, во-вторых, чтобы решения абстракционной задачи позволяли сократить перебор, необходимый для поиска решений исходной задачи. Ниже мы покажем, что предлагаемая суживающая стратегия удовлетворяет обоим требованиям.

Теоретической основой представления знаний в системах ИИ является исчисление предикатов первого порядка, в рамках которого решается известная задача математической логики — проблема дедукции. Хотя проблема дедукции достаточно хорошо известна, для полноты изложения приведем ее краткую формулировку. Пусть имеется конечное множество логических формул $\mathbf{E = \{H_1, H_2, \dots, H_n\}}$, которые мы назовем гипотезами, а также еще одна формула **C**. Проблема дедукции заключается в проверке того, является ли **C** логическим следствием множества **E**. Логическим следствием называется формула, принимающая значение «истина» при любой интерпретации, при которой все гипотезы множества **E** принимают значение «истина». С прикладной точки зрения наиболее интересен случай, когда гипотезы множества **E** представлены в виде хорновских дизъюнктов, т.е. таких дизъюнктов, которые имеют не более одной позитивной литеры (под позитивной литерой понимается предикат, а под негативной — отрицание предиката).

Обозначив позитивные литеры через **l**, а негативные — через $\neg$**l**, определим три группы хорновских дизъюнктов:

1) унитарные позитивные или единичные дизъюнкты, состоящие из одной позитивной литеры и имеющие вид **(l)**;

2) точные дизъюнкты, состоящие из позитивной и хотя бы одной негативной литер и имеющие вид $\mathbf{(l, \neg l_1, \dots, \neg l_n)}$;

3) негативные дизъюнкты, состоящие лишь из негативных литер и имеющие вид $\mathbf{(\neg l_1, \dots, \neg l_n)}$.

Множества гипотез **E** состоят из дизъюнктов первых двух групп, а **C** представляется в виде конъюнкции позитивных литер: $\mathbf{l_1 \,\&\, l_2 \,\&\dots\&\, l_n}$. Для решения проблемы дедукции достаточно образовать множество $\mathbf{S = E \cup \{ \neg C \}}$ и доказать его невыполнимость (тогда **C** является логическим следствием **E**) или выполнимость (в этом случае **C** не является логическим следствием). Именно так действуют резолюционные методы. Существенно то, что $\neg$**C** — это негативный дизъюнкт $\mathbf{(\neg l_1, \dots, \neg l_n)}$ и, следовательно, **S** состоит из хорновских дизъюнктов. Таким образом, логический вывод понимается как вывод из множества **S** пустого дизъюнкта Λ и при наличии хотя бы одного вывода проблема дедукции решается положительно. В терминах ИИ **E** — это БЗ, а **C** — цель, т.е. вопрос, по-

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

зволяющий пользователю извлечь из системы нужные ему сведения. В этом случае вывод в **S** пустого дизъюнкта будем называть выводом **C** в **E**.

Отметим, что проблема дедукции формулируется как для исчисления предикатов (ИП), так и для исчисления высказываний (ИВ). В последнем случае под литерами понимаются высказывания или их отрицания. Применяя метод абстракций, мы можем заменить все предикаты БЗ и цели независимо от значений их аргументов высказываниями. Такая абстракция носит название пропозициональной. Ее определение таково [14].

Пусть **D** — дизъюнкт вида $(L_1, L_2, \dots, L_k)$. Тогда пропозициональная абстракция определяется абстракционным отображением $f(D) = D'$, где D' — дизъюнкт вида $(L'_1, L'_2, \dots, L'_k)$, $L'_i = P$, если $L_i = P(l_1, \dots, l_n)$ и $L'_i = \neg P$, если $L_i = \neg P(l_1, \dots, l_n)$, $1 \le i \le k$.

В построенной таким образом простой модели проблема дедукции решается значительно проще. В ИП задача поиска логического вывода в общем случае является NP-полной, а для получения полиномиальной оценки по времени необходимы существенные ограничения на хорновские дизъюнкты (например, использование лишь одноместных предикатов). В то же время в ИВ большинство резолюционных методов имеют оценку трудоемкости Cn^2, где **n** — число литер, присутствующих в **S**, а предложенный ниже метод имеет оценку **Cn.**

Суживающая стратегия, предлагаемая в данной статье, построена на основе пропозициональной абстракции. В противоположность большинству известных суживающих стратегий она предусматривает настройку алгоритма поиска логических выводов на работу с конкретной БЗ. Ее целесообразно применять при работе со стационарными БЗ, т.е. такими, которые в процессе эксплуатации изменяются относительно редко. На практике стационарные и даже статические (не модифицируемые) БЗ встречаются достаточно часто. Так, например, такие БЗ используются в обучающих системах с элементами ИИ, часто называемых экспертно-обучающими системами.

Описываемый метод поиска логического вывода состоит из двух основных частей. Во-первых, в него включен эффективный алгоритм решения абстракционной задачи, полученной применением пропозициональной абстракции. Это алгоритм решения проблемы дедукции в ИВ. Он позволяет построить выводы дизъюнкта **C**, если последний является логическим следствием (ниже будут указаны некоторые особенности таких выводов, связанные с понятием мультидизюнкта). Алгоритм использует формально-грамматическую интерпретацию проблемы дедукции и предусматривает предварительную настройку на работу с конкретным множеством гипотез. Следует подчеркнуть, что предварительная настройка выполняется до задания целей, т.е. один раз для БЗ. Во-вторых, в предлагаемый метод включен алгоритм построения выводов в ИП по выводам в ИВ, полученным при решении абстракционной задачи. Рассмотрим сначала первый алгоритм, остановившись прежде всего на необходимой для него предварительной настройке.

Пусть имеется БЗ **E** в ИВ, состоящая из дизъюнктов $\{D_1, \dots, D_m\}$, в которой используются высказывания $P_1, \dots, P_n$. Построим по **E** КС-грамматику **G(E)**, без терминальных символов и начального нетерминала: $G(E) = \{V_n, Q\}$, где V_n — множество нетерминальных символов, а **Q** — множество правил вывода. Множество нетерминальных символов образуем из высказываний в **E**: $V_N = \{P_1, \dots, P_n\}$. Правила вывода грамматики построим по входящим в **E** дизъюнктам. Пусть $D_i = (P_j, \neg P_{l1}, \dots, \neg P_{lk})$. Построим по нему правило вывода $P_j \rightarrow P_{l1} \dots P_{lk}$. Если $D_i = (P_j)$, то получим укорачивающее правило $P_j \rightarrow \varepsilon$. По цели, представленной в

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

виде $C = (\neg P_{i1}, \ldots, \neg P_{iq})$, образуем цепочку соответствующих нетерминальных символов $\omega = P_{i1} \ldots P_{iq}$. Теперь логический вывод во множестве **S** можно интерпретировать как вывод пустой цепочки из ω: $\omega \Rightarrow^{+} \varepsilon$. Формально-грамматическая интерпретация позволит нам использовать методы теории формальных грамматик для настройки на конкретные БЗ, что, в свою очередь, значительно повысит эффективность алгоритма поиска логических выводов.

Введем понятие укорачивающего символа грамматики. Нетерминальный символ **A** называется укорачивающим, если в данной грамматике из него возможен вывод пустой цепочки: $A \Rightarrow^{+} \varepsilon$. Очевидно, вывод пустой цепочки из некоторой ω возможен лишь в том случае, если все входящие в ω символы являются укорачивающими. Это утверждение дает возможность выполнить предварительную настройку алгоритма на работу с заданной БЗ. Настройка включает в себя выделение подмножества укорачивающих символов, которое удобно представить в виде двоичной шкалы. Разряды шкалы соответствуют нетерминалам грамматики и принимают разные значения, в зависимости от того, является ли данный нетерминал укорачивающим или нет. После построения шкалы укорачивающих символов алгоритм, распознающий свойство выводимости целей совершенно элементарен.

Нашей задачей является не только распознавание выводимости целей, но и построение выводов (по возможности всех) в случае выводимости. Поэтому предварительная настройка на работу с конкретной БЗ осуществляется алгоритмом, не только выделяющим укорачивающие символы, но и формирующим для каждого укорачивающего символа списки правил, с которых могут начинаться выводы пустой цепочки. Рассмотрим этот алгоритм.

Пусть имеется некоторая грамматика **G(E)**. Обозначим через **W** формируемое подмножество укорачивающих символов грамматики, через **S(A) —** формируемые для каждого символа из **W** списки правил вывода, с которых могут начинаться выводы пустой цепочки из **A**. Алгоритм выделения укорачивающих символов и соответствующих им правил вывода представим в виде следующего итеративного процесса:

$$W_0 = \{ A \mid A \rightarrow \varepsilon \in Q \}; \; S_0(A) = \{ R \mid R = A \rightarrow \varepsilon, R \in Q \},$$

т.е. включаем в W_0 те символы грамматики, для которых в ней имеются укорачивающие правила вывода, а в соответствующие S_0 **—** сами укорачивающие правила. Пусть уже получены W_i и $S_i(A)$ для всех $A \in W_i$.

Выполним очередной шаг итеративного процесса следующим образом:

$$W_{i+1} = W_i \cup \{ A \mid A \rightarrow A_1 A_2 \ldots A_n \in Q; A_i \in W_i, i = 1, \ldots, n \};$$

$$S_{i+1}(A) = S_i(A) \cup \{ R \mid R = A \rightarrow A_1 A_2 \ldots A_n ; A_i \in W_i, i = 1, \ldots, n; R \in Q \setminus S_i(A) \},$$

т.е. включаем в W_{i+1} все символы из W_i и те символы грамматики, для которых найдутся правила вывода, все символы правых частей которых включены в W_i. В соответствующие S_{i+1} включим правила из S_i, а также правила, обладающие указанным выше свойством и не входящие в S_i. Отметим, что S_{i+1} – это упорядоченные множества (списки) и вновь включаемые в них правила вывода попадают в концы списков.

Алгоритм закончит работу, когда на некотором шаге **j+1** окажется, что имеют место равенства $S_{j+1}(A) = S_j(A)$ для всех $A \in V_N$. Примем $W = W_j$ и $S(A) = S_j(A)$ для каждого $A \in W$. Для оценки числа шагов алгоритма докажем следующие две леммы.

Лемма 1. *Пусть на j-ом шаге алгоритма получено* $W_j = W_{j-1}$. *Тогда* $S_{j+1}(A) = S_j(A)$ *для любого* $A \in W_j$.

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Доказательство. *Доказывая утверждение от противного, предположим, что для некоторого* A *найдется правило* $R = A \to \eta$ *такое, что* $R \notin S_j(A)$ *и* $R \in S_{j+1}(A)$. *Так как правило* R *не попало в* $S_j(A)$, *найдется символ* A_i, *который входит в цепочку* η *и* $A_i \notin W_{j-1}$. *Но* R *включено в* $S_{j+1}(A)$ *и, следовательно,* $A_i \in W_j$. *Поэтому* $W_j \neq W_{j-1}$. *Получили противоречие, доказывающее наше утверждение.*

Лемма 2. *Число шагов алгоритма настройки не превышает числа нетерминальных символов грамматики.*

Доказательство. *Так как для продолжения работы алгоритма необходимо, чтобы на каждом его шаге число выделенных укорачивающих символов увеличивалось, число шагов не может превышать числа нетерминальных символов грамматики.*

Важное свойство формируемых алгоритмом настройки подмножеств **S** сформулировано в следующей теореме.

Теорема 1. *На всех шагах любого вывода* $A \Rightarrow^+ \varepsilon$ *могут применяться лишь правила из подмножеств* S.

Доказательство. *Докажем это утверждение от противного. Не умаляя общности, мы можем рассматривать левосторонние выводы. Пусть на некотором шаге вывода к цепочке* $\omega_j = B\xi$, $B \in V_N$, $\xi \in V_N^*$, *было применено правило* $R = B \to \eta$ *такое, что* $R \notin S(B)$. *Так как мы исследуем вывод пустой цепочки, в нем должна найтись цепочка* $\omega_k = \xi$, $k > j$. *Поэтому существует вывод* $\eta \Rightarrow^+ \varepsilon$, *и все символы цепочки* η *укорачивающие, а тогда найдется такое* i, *что в* W_i, *входят все символы цепочки* η *и* $R \in S_i(B)$. *Так как* $S_i(B) \subseteq S(B)$, $R \in S(B)$. *Получили противоречие, доказывающее наше утверждение.*

Подмножество **W** позволяет сформировать двоичную шкалу нетерминальных символов, которая вместе со списками **S** является результатом работы алгоритма предварительной настройки.

Теперь можно описать алгоритм решения абстракционной задачи. Выводимость цели **C** проверяется построением соответствующей цепочки нетерминальных символов грамматики и просмотром значений шкалы для символов этой цепочки. **C** выводима в том и только том случае, если шкала покажет, что все символы цепочки являются укорачивающими. Для перебора всех вариантов ее вывода нужно в построенной по данной БЗ КС-грамматике перебрать выводы пустой цепочки из цепочки, соответствующей **C**. Такой перебор легко осуществить, пользуясь списками **S**. Отметим, что при этом не будут перебираться тупиковые варианты. Это выгодно отличает предлагаемый метод от других алгоритмов, построенных на основе метода резолюций.

Проиллюстрируем описанный алгоритм, включая предварительную настройку, следующим примером.

Пример 1. Пусть задана следующая БЗ в ИВ:

E = { (1) (p, ¬r, ¬t,), (2) (p, ¬q, ¬r,), (3) (p, ¬q, ¬s,), (4) (q, ¬s, ¬t, ¬u), (5) (q, ¬u), (6) (r), (7) (t, ¬q, ¬u), (8) (u) }.

Соответствующая этой БЗ грамматика выглядит так:

V_N = { p, q, r, s, t, u }.

В **Q** войдут следующие правила вывода:

1) **p → rt;** 2) **p → qr;** 3) **p → qs;** 4) **q → stu;** 5) **q → u;** 6) **r → ε;** 7) **t → qu;** 8) **u → ε.**

Применим итерационный алгоритм настройки. Правила вывода будем представлять своими номерами.

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

W_i	Символы		S_i	p	q	r	s	t	u
W_0	r, u		S_0			6			8
W_1	q, r, u		S_1		5	6			8
W_2	p, q, r, t, u		S_2	2	5	6		7	8
W_3	p, q, r, t, u		S_3	2, 1	5	6		7	8

Так как $W_2 = W_3$, работа алгоритма закончена. Настройка на заданную БЗ произведена: $W = W_2$, $S(\alpha) = S_3(\alpha)$, $\alpha \in W$. Имеем шкалу (1 — символ укорачивающий, 0 — символ неукорачивающий):

p	q	r	s	t	u
1	1	1	0	1	1

Пусть цель **C = p&q&s**. Ее отрицание ⌝**C = (**⌝**p,** ⌝**q,** ⌝**s)**. Этому дизъюнкту соответствует цепочка ω **= pqs**. Находим в шкале значения символов ω: 110. Так как **s** – неукорачивающий, **C** невыводим.

Пусть **C = p&q**. Его отрицание ⌝**C = (**⌝**p,** ⌝**q)**. Этому дизъюнкту соответствует цепочка ω **= pq**. Значения символов ω в шкале 11. Следовательно, **C** выводим. Найдем все его выводы, перебирая варианты вывода из ω пустой цепочки с учетом подмножеств **S**:

1) **pq, qrq, urq, rq, q, u,** ε (при использовании на первом шаге правила 2).
2) **pq, rtq, tq, quq, uuq, uq, q, u,** ε (при использовании на первом шаге правила 1).

Мы видим, что наш алгоритм не применял в процессе перебора правило 3, так как оно не входит в **S(p)**. По полученным выводам в грамматике легко построить выводы **C** в исходной БЗ (предоставим это читателю). Отметим лишь, что в каждой цепочке выводов можно оставить лишь по одному вхождению каждого символа и сократить совпадающие цепочки. Такое преобразование следует выполнять, если нас интересуют выводы в ИВ. Если же поиск выводов в ИВ рассматривается как метод решения абстракционной задачи, то выводы нужно оставить в их первоначальном виде.

Перейдем к описанию второго алгоритма предлагаемого метода — алгоритма построения выводов в ИП по выводам в ИВ, полученным применением пропозициональной абстракции. Этот алгоритм должен учитывать особенность, связанную с тем, что в дизъюнктах ИП литерами могут оказаться одни и те же предикаты с различными значениями аргументов. При применении пропозициональной абстракции таким предикатам соответствуют одни и те же высказывания. Сокращение повторяющихся литер в дизъюнктах ИВ недопустимо, так как приводит к несоответствию между выводами в ИП и ИВ. Во избежание подобного обстоятельства в рамках пропозициональной абстракции используется понятие *мультидизъюнкта* — это дизъюнкт ИП, который может содержать несколько экземпляров одного и того же высказывания (в том числе и литеры вида **p** и ⌝**p**, хотя дизъюнкты, содержащие подобные литеры, являются тавтологиями и обычно исключаются из рассмотрения). Мультидизъюнкты являются объектами и результатами действия *m-абстракции* и *m-резолюции*, которые определяются аналогично понятиям абстракции и резолюции. Выводы в множестве мультидизъюнктов ИВ строятся с помощью m-резолюции и поэтому будут соответствовать по своей структуре выводам в исходном множестве дизъюнктов ИП.

Описываемый алгоритм для построения выводов целей в ИП использует выводы в ИВ, полученные в результате решения абстракционной задачи, как

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

схемы выводов в ИП. Построение выводов в ИП производится последовательным просмотром всех схем с одновременным восстановлением аргументов предикатов и выполнением унификации. Если на каком-либо этапе просмотра очередной схемы унификация окажется невозможной, то данная схема является непродуктивной, т.е. выводы в ИП, соответствующие этой схеме отсутствуют. Если все схемы окажутся непродуктивными, то выводы в ИП отсутствуют и соответствующая цель невыводима.

Наиболее эффективная стратегия построения выводов в ИП — это параллельное построение выводов в грамматике, схем в ИВ и выводов в ИП. Алгоритм, реализующий эту стратегию, назовем параллельным. Каждый шаг его работы заключается в применении некоторого правила вывода для построения вывода в грамматике, использовании соответствующего правилу мультидизъюнкта в абстракционной задаче для резолюции и в попытке использования соответствующего дизъюнкта для резолюции в исходной задаче. Эта попытка будет успешной, если найдется соответствующий унификатор. Параллельный алгоритм оперативно выявляет непродуктивные схемы еще в процессе их построения, если на некотором шаге унификация окажется невозможной. Предусматривается также удаление алгоритмом "лишних" литер. Если одинаковым литерам в мультидизъюнктах схемы соответствуют одинаковые литеры в дизъюнктах вывода в ИП, то повторяющиеся литеры исключаются в схеме, и в выводе, а из цепочки вывода удаляются соответствующие им символы. Формальное описание параллельного алгоритма достаточно громоздко. Поэтому проиллюстрируем этот алгоритм следующими двумя примерами.

Пример 2. Пусть в ИП задана следующая БЗ:

E = { (1) (q(c), ¬p(a), ¬p(b)), (2) (p(a)), (3) (p(b)), (4) r(d)) }.

Пропозициональные m-абстракции, построенные по множеству **E**:

f(E) = { (1) (q, ¬p, ¬p), (2) (p), (3) (p), (4) (r) }.

Грамматика, построенная по множеству **f(E)**:

G(f(E)) = { V_N, Q }; **V_N = { q, p, r }; Q = { (1) q → pp, (2) p → ε, (3) p → ε, (4) r → ε }.**

Двоичная шкала символов:

q	p	r
1	1	1

Списки правил, выделенных для укорачивающих символов грамматики:
S(q) = (1), **S(p) = (2, 3)**, **S(r) = (4)**.

Пусть задана цель **C = q(c)&r(d)**. Ее отрицание **¬C = (¬q(c), ¬r(d))**. M-абстракция **f(¬C) = (¬q, ¬r)**. Соответствующая цепочка **ω = qr**. Имеем из шкалы значения 11. Следовательно, m-абстракция цели выводима. Используя параллельный алгоритм, построим вывод исходной цели, формируя вывод в грамматике с указанием применяемых правил и соответствующие им схемы.

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
qr (1)	**(¬q, ¬r)**	**(¬q(c), ¬r(d))**	**Λ**
ppr (2)	**(¬p, ¬p, ¬r)**	**(¬p(a), ¬p(b), ¬r(d))**	**Λ**
pr (2)	**(¬p, ¬r)**	**(¬p(b), ¬r(d))**	Унификатор отсутствует

Так как провести унификацию для продолжения вывода в **E** оказалось невозможным, возник тупик, означающий, что текущая схема непродуктивна. Алгоритм прекращает работу с ней и, осуществляя перебор, переходит к работе

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

со следующей схемой. Эта схема строится применением на последнем шаге вместо правила 2 в **G(f(E))** правила 3:

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
pr (3)	**(¬p, ¬r)**	**(¬p(b), ¬r(d))**	**Λ**
r (4)	**(¬r)**	**(¬r(d)**	**Λ**
ε	**Λ**	**Λ**	

Один вывод найден, но работа алгоритма не закончена, так как имеется еще одна схема, которая строится применением на втором шаге вместо правила 2 правила 3:

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
ppr (3)	**(¬p, ¬p, ¬r)**	**(¬p(a), ¬p(b), ¬r(d))**	Унификатор отсутствует

Возник тупик, означающий, что текущая схема непродуктивна. Так как перебор схем исчерпан, параллельный алгоритм заканчивает работу, построив единственный вывод цели. Данный пример иллюстрирует необходимость использования m-абстракции. Если бы в мультидизъюнкте **(¬p, ¬p, ¬r)** вывода в **f(E)** была удалена одна из литер **¬p**, выводы в грамматике, и в **f(E)** оказались бы неадекватными выводам в **E**.

Пример 3 [1]. Пусть задана следующая БЗ в ИП (**a** и **b** — константы, **x** — переменная):

E = { (1) (p(a), ¬q(x), ¬r(b)), (2) (p(x), ¬t(x)), (3) (q(b), ¬p(a), ¬t(a)), (4) (t(b)), (5) (t(a)) }.

Пропозициональные m-абстракции, построенные по множеству **E**:

f(E) = { (1) (p, ¬q, ¬r), (2) (p, ¬t), (3) (q, ¬p, ¬t), (4) (t), (5) (t) }.

Грамматика, построенная по множеству **f(E)**:

G(f(E)) = { V_N, Q }; **V_N = { p, q, r, t }; Q = { (1) p → qr, (2) p → t, (3) q → pt, (4) t → ε, (5) t → ε }.**

Двоичная шкала символов:

p	**q**	**r**	**t**
1	**1**	**0**	**1**

Списки правил, выделенных для укорачивающих символов грамматики:

S(p) = (2), **S(q) = (3)**, **S(t) = (4, 5)**.

Пусть задана цель **C = q(x)&p(x)**. Ее отрицание **¬C = (¬q(x), ¬p(x))**. M-абстракция **f(¬C) = (¬q, ¬p)**. Соответствующая цепочка **ω = qp**. Имеем из шкалы значения 11. Следовательно, m-абстракция цели выводима. Используя параллельный алгоритм, построим вывод исходной цели, формируя вывод в грамматике с указанием применяемых правил и соответствующие им схемы.

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
pq (3)	**(¬q, ¬p)**	**(¬q(x), ¬p(x))**	**(x = b)**
ptp (2)	**(¬p, ¬t, ¬p)**	**(¬p(a), ¬t(a), ¬p(b))**	**(x = a)**
ttp	**(¬t, ¬t, ¬p)**	**(¬t(a), ¬t(a), ¬p(b))**	

В дизъюнкте вывода в **E** литера **¬t(a)** оказалась представленной двумя копиями. Поэтому параллельный алгоритм удаляет одну из них, а также соответ-

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ствующие копию в мультидизъюнкте вывода в **f(E)** и символ в цепочке вывода в **G(f(E))**:

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
tp (4)	**(¬t, ¬p)**	**(¬t(a), ¬p(b))**	Унификатор отсутствует

Возник тупик, означающий, что текущая схема непродуктивна. Алгоритм прекращает работу с ней и, осуществляя перебор, переходит к работе со следующей схемой. Эта схема строится применением на последнем шаге вместо правила 4 в **G(f(E))** правила 5:

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
tp (5)	**(¬t, ¬p)**	**(¬t(a), ¬p(b))**	**Λ**
p (2)	**(¬p)**	**(¬p(b))**	**(x = b)**
t (4)	**(¬t)**	**(¬t(b))**	**Λ**
Λ	**Λ**	**Λ**	

Один вывод найден, но работа алгоритма еще не закончена, так как имеется еще одна схема:

Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
t (5)	**(¬t)**	**(¬t(b))**	Унификатор отсутствует

Здесь вновь возник тупик, и так как перебор схем исчерпан, алгоритм заканчивает работу, построив единственный вывод цели. Отметим, что параллельный алгоритм не пытался строить вывод путем резольвирования литеры **¬p(a)** с дизъюнктом 1: **(p(a), ¬q(x), ¬r(b))**. Это вызвано тем, что правило 1 грамматики **p → qr** не попало в **S(p)**. Из обоих примеров также хорошо видно, что непродуктивные схемы удаляются из рассмотрения уже в процессе их построения. Если бы алгоритм строил схемы и выводы последовательно, непродуктивные схемы формировались бы полностью.

В [1] доказана полнота рассмотренного метода поиска логического вывода с предварительной настройкой, а также идентичность выводов, полученных данным методом, с выводами, полученными входной линейной резолюцией. В то же время, очевидно, что метод с предварительной настройкой во многих случаях (хотя и не всегда) эффективнее входной линейной резолюции. Было произведено сравнение этих двух методов, для чего они оба были реализованы в рамках экспериментальной программной системы. Получены статистически значимые результаты, показывающие предпочтительность метода с предварительной настройкой.

Абстракционные методы поиска логического вывода в системах ИИ не обязательно должны строиться на основе пропозициональной абстракции. Могут представлять интерес и другие типы абстракций, в частности такие, в которых экземплярам одного и того же предиката с неидентичными аргументами при некоторых условиях соответствуют различные высказывания. Для разработки практически применимых методов на основе подобных абстракций требуются дальнейшие исследования.

Труды СПИИРАН. 2002. Вып. 1. Т. 2. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2002. Issue 1. V. 2. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Литература

[1] *Анисимова И. Н.* Формально-грамматическая модель логического вывода в системапх искусственного интеллекта. Диссертация на соискание ученой степени кандидата физ.-мат. наук. — СПб., 1999. — 143 с.

[2] *Анисимова И. Н., Братчиков И. Л.* Формально-грамматическая модель метода резолюций для исчисления высказываний // Вестник СПбГУ. Сер. 1. Математика. Механика. Астрономия. Вып. 3 (№ 15), 1996 — с. 3–7.

[3] *Анисимова И. Н., Братчиков И. Л.* Двухэтапный алгоритм логического вывода в ЭОС // Труды Международного семинара "Искусственный интеллект в образовании" — Казань, 1996. — с. 16–19.

[4] *Анисимова И. Н., Братчиков И. Л.* "Эффективный метод построения логических выводов в стационарных базах знаний // Ученые записки. Ленинградский областной гос. Ун-т. Сер. "Математика и информатика". т. 1. — СПб., 1998. — с. 12–16.

[5] *Братчиков И. Л.* Формально-грамматическая интерпретация метода резолюций // Процессы управления и устойчивость. Труды XXIX научной конференции. СПб.: СПбГУ, Факультет прикладной математики – процессов управления, 1998 — с. 197–202.

[6] *Братчиков И. Л., Калинина Т. В.* Абстракционные модели построения логического вывода // Процессы управления и устойчивость. Труды XXXI научной конференции. СПб.: СПбГУ, Факультет прикладной математики – процессов управления, 2000 — с. 281–286.

[7] *Буланов А. А., Калинина Т. В.* Реализация метода поиска вывода в исчислении высказываний с предварительной настройкой // Процессы управления и устойчивость. Труды XXIX научной конференции. СПб.: СПбГУ, Факультет прикладной математики – процессов управления, 1998 — с. 203–208.

[8] *ред.Д.А.Поспелов* Искусственный интеллект. Справочник. — Кн.2. Модели и методы // М.: Радио и связь, 1990. — 303 с.

[9] *Тейз А., Грибомон П. и др.* Логический подход к искусственному интеллекту: от классической логики к логическому программированию. — М.: Мир, 1990. — 432 с.

[10] *Лорьер Ж.-Л.* Системы искусственного интеллекта. — М.: Мир, 1991. — 568 с.

[11] *Клещев А. С.* Реализация экспертных систем на основе декларативных моделей представления знаний.: Препринт ДВО АН СССР. — Владивосток, 1988. — 45 с.

[12] *Eisinger N., Ohlbach H. J., Präcklein A.* Reduction Rules for Resolution Based Systems // Artificial Intelligence, vol. 50, no. 2, 1991. p. 141–181.

[13] *Kerber M., Präcklein A.* Using Tactics to Reformulate Formulae for Resolution Theorem Proving // Annals of Mathematics and Artificial Intelligence, vol. 18, 1996. p. 221–241.

[14] *Plaisted D. A.* Theorem Proving with Abstraction // Artificial Intelligence. vol. 16, no. 1. 1981. — p. 47–108.