УДК 681.3

Братчиков Игорь Леонидович,
Сазонова Наталья Викторовна

ФОРМАЛЬНО-ГРАММАТИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ ЛОГИЧЕСКОГО ВЫВОДА

Аннотация

В статье рассматриваются суживающие (сокращающие перебор) стратегии для резолюционных алгоритмов логического вывода в базах знаний, построенных на основе логической модели знаний. Особенностью этих стратегий является предварительная настройка на работу с конкретными базами знаний. Для настройки используется метод абстракций и формально-грамматическая интерпретация логического вывода. Абстракция – это функция, которая отображает класс задач $\mathbf{K_1}$ в более простой класс $\mathbf{K_2}$. При решении задачи $\mathbf{A} \in \mathbf{K_1}$ ее отображают в задачу $\mathbf{B} \in \mathbf{K_2}$. Затем задача **B** решается, и ее решения используются для поиска решений исходной задачи **A**. В настоящей работе класс задач $\mathbf{K_1}$ – это проблема дедукции в исчислении предикатов первого порядка. Для $\mathbf{K_1}$ рассматриваются две абстракции – пропозициональная, при которой $\mathbf{K_2}$ – это проблема дедукции в исчислении высказываний, и абстракция, при которой проблема дедукции решается для конструкций, названных псевдо-предикатами. Для обеих абстракций предлагается формально-грамматическая интерпретация, позволяющая выполнить предварительную настройку, которая гарантирует перебор лишь успешных вариантов выводов. Приведенные примеры иллюстрируют достоинства предложенной методики.

Ключевые слова: база знаний, логическая модель представления знаний, проблема дедукции, суживающая стратегия логического вывода, входная линейная резолюция, метод абстракций, формальная грамматика.

В настоящее время наблюдается тенденция к применению в программных системах различного назначения средств и методов искусственного интеллекта (ИИ). Не являются исключением и системы компьютерного обучения, в которых реализуются такие средства как индивидуализация процесса обучения, генерация контрольных вопросов, анализ ответов на них и пр.

В значительной части систем ИИ используется логическая модель представления знаний в виде баз знаний (БЗ) и резолюционные методы поиска логического вывода. К числу таких методов относится входная линейная резолюция, на основе которой был разработан широко известный язык ИИ Пролог. Недостатком метода входной линейной резолюции, как и других резолюционных методов, является его недостаточная эффективность, связанная с необходимостью перебора большого числа вариантов вывода, в том числе и тупиковых, причем в некоторых случаях возможно даже зацикливание.

Для повышения эффективности резолюционных методов обычно используются различные сокращающие перебор (суживающие) стратегии, большинство из которых требует ручной разработки высококвалифицированными специалистами. Поэтому задача создания суживающих стратегий, которые могут применяться в автоматическом режиме, является важной и актуальной. В настоящей статье представлены результаты исследований, направленных на создание автоматизируемой суживающей стратегии для резолюционных алгоритмов поиска логического вывода, отличительной особенностью ко-

© И.Л. Братчиков, Н.В. Сазонова, 2010

торой является возможность предварительной настройки на работу с заданными БЗ [2]. Подобная настройка особенно важна в случае работы со стационарными (редко модифицируемыми) или статическими БЗ, к числу которых относятся и БЗ учебного назначения. Основой для предварительной настройки явилась формально-грамматическая модель знаний [3], которая оказалась особенно эффективной в сочетании с методом абстракций [1]. В рамках этого подхода были разработаны два алгоритма предварительной настройки, представленные ниже.

Метод абстракций формулируется следующим образом. Пусть имеется некоторый класс задач $\mathbf{K_1}$, алгоритмы решения которых недостаточно эффективны. По классу $\mathbf{K_1}$ строится класс более простых задач $\mathbf{K_2}$, решение которых не представляет значительных трудностей. При этом требуется, чтобы для любой задачи $\mathbf{A} \in \mathbf{K_1}$ нашлась такая задача $\mathbf{B} \in \mathbf{K_2}$, решения которой имели бы структуру, сходную со структурой решений задачи **A**, и могли бы использоваться как некие схемы для поиска решений первоначальной задачи. При этом важно, чтобы задачи из $\mathbf{K_2}$ обладали свойством полноты в том смысле, что решения задачи **B** всегда позволяли бы найти все решения соответствующей ей задачи **A** (хотя некоторые решения задачи **B** могут не привести к цели).

В [1] класс $\mathbf{K_2}$ строится из так называемых абстракционных отображений задач класса **A**, а задача $\mathbf{B} \in \mathbf{K_2}$ называется абстракционной задачей или абстракцией задачи $\mathbf{A} \in \mathbf{K_1}$. 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.

Логическая модель представления знаний в системах ИИ основана на логике предикатов первого порядка, в рамках которой решается известная задача математической логики – проблема дедукции. Для полноты изложения приведем ее краткую формулировку. Пусть имеется конечное множество логических формул $\mathbf{E} = \{\mathbf{H_1}, \mathbf{H_2}, ..., \mathbf{H_n}\}$, которые обычно называются гипотезами, а также еще одна формула **C**. **C** называется логическим следствием множества **E**, если она принимает значение «истина» при любой интерпретации, при которой все гипотезы множества **E** принимают значение «истина». Проблема дедукции заключается в проверке того, является ли **C** логическим следствием множества **E**. Для решения проблемы дедукции используются два метода – прямой и обратной дедукции. Первый из них заключается в проверке на невыполнимость формулы

$$\mathbf{H_1 \ \& \ ... \ \& \ H_n \ \& \ \neg C},$$

а второй – в проверке общезначимости формулы

$$\mathbf{\neg H_1 \vee ... \vee \neg H_n \vee C}.$$

В системах ИИ, как правило, применяется метод прямой дедукции, причем гипотезы множества **E** и отрицание **C** представляются в виде хорновских дизъюнктов, то есть 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 $\mathbf{\neg l}$, определим три группы хорновских дизъюнктов:

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

2) точные дизъюнкты (ТД), состоящие из позитивной и хотя бы одной негативной литер и имеющие вид $\mathbf{(l, \neg l_1, ..., \neg l_n)}$, $\mathbf{n \geq 1}$;

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

В рамках логической модели базы знаний представляются в виде совокупности хорновских дизъюнктов первых двух групп:

1. УПД применяются для указания свойств и отношений, установленных для

выбранной предметной области, которые обычно называются фактами.

2. ТД используются для представления правил, с помощью которых выводятся новые факты, явно не представленные в виде УПД. Семантика этих дизъюнктов такова: предикат, представленный литерой **l**, принимает значение «истина», если установлено, что предикаты, представленные литерами $\mathbf{l_1}$, ..., $\mathbf{l_n}$, истинны.

Вопросы пользователей удобно формулировать в виде конъюнкции позитивных литер: $\mathbf{(l_1 \,\&\, ... \,\&\, l_n)}$, отрицание которой является негативным дизъюнктом: $\mathbf{(\neg l_1 \vee ... \vee \neg l_n)}$.

Метод входной линейной резолюции (ВЛР), как и другие резолюционные методы, основан на правиле резолюции, согласно которому к двум хорновским дизъюнктам, содержащим контрарную пару, то есть позитивную и негативную литеры, представленные одним и тем же предикатом с идентичными аргументами, может быть применена процедура резолюции, в результате которой формируется хорновский дизъюнкт, называемый резольвентой:

$$\mathbf{d_1 = (l, l_1, ..., l_m),\ d_2 = (\neg l, l'_1, ..., l'_n),}$$

$$\mathbf{r = (l_1, ..., l_m, l'_1, ..., l'_n).}$$

Здесь $\mathbf{d_1}$ и $\mathbf{d_2}$ – дизъюнкты, содержащие контрарную пару **(l, ¬l)**, $\mathbf{l_i}$ и $\mathbf{l'_i}$ – литеры (позитивные или негативные), **r** – резольвента.

Метод ВЛР применяется к множеству $\mathbf{T = E \cup \{Ng\}}$, где **E** – база знаний, состоящая из хорновских дизъюнктов первых двух групп, а **Ng** – НД, представляющий вопрос пользователя: $\mathbf{Ng = \neg C}$. Целью является получение в качестве резольвенты пустого дизъюнкта **F**, и, если эта цель достигнута, проблема дедукции решается положительно. Это означает, что пользователю может быть предоставлена интересующая его информация. Особенность данного метода заключается в том, что процедура резолюции применяется лишь к парам, у которых один из элементов является негативным дизъюнктом. На первом шаге – это **Ng**, на последующих – резольвенты, полученные на предыдущих шагах. Последовательность шагов, в результате которых выводится **F**, называется логическим выводом **C** из **E**.

Известно, что для хорновских дизъюнктов задача поиска логического вывода в логике предикатов является NP-полной, и поэтому применение метода абстракций представляется вполне оправданным.

Сначала рассмотрим так называемую пропозициональную абстракцию [1], которая определяется следующим образом. Класс задач $\mathbf{K_1}$ – это проблема дедукции для множеств **T** в логике предикатов (ЛП), класс $\mathbf{K_2}$ – проблема дедукции для множеств **T′** в логике высказываний (ЛВ). Абстракция задачи из $\mathbf{K_1}$ строится следующим образом. Пусть дизъюнкт $\mathbf{d = (l_1, ..., l_n)}$ и $\mathbf{d \in T}$. Тогда его абстракционное отображение $\mathbf{d' = (l'_1, ..., l'_n)}$, где $\mathbf{l'_i = p}$, если $\mathbf{l_i = p(t_1, ..., t_m)}$ и $\mathbf{l'_i = \neg p}$, если $\mathbf{l_i = \neg p(t_1, ..., t_m)}$. Иначе говоря, абстракция задачи из $\mathbf{K_1}$ строится путем замены предикатов, входящих в дизъюнкты, соответствующими высказываниями. В логике высказываний резолюционные методы имеют полиномиальную оценку трудоемкости $\mathbf{Cn^2}$, а в некоторых случаях и **Cn**. Существенное различие в трудоемкости алгоритмов решения этих задач послужило основанием для разработки данного метода.

Особенностью описанного отображения является то, что одним и тем же предикатам с различными значениями аргументов будут соответствовать одинаковые высказывания. Например, абстракционное отображение дизъюнкта **((p(x,y), ¬q(x,a), ¬q(b,y))** таково: **(p, ¬q, ¬q)**, а для дизъюнкта **((p(x,y), ¬p(x,a), ¬q(b,y))** получим **(p, ¬p, ¬q)**. Сокращение одинаковых литер и исключение дизъюнктов, содержащих пары литер вида **p** и **¬p**, недопустимо, так как приводит к несоответствию выводов в ЛВ и ЛП. Поэтому нами используется понятие *мультидизъюнкта*, то есть дизъюнкта, который может содержать несколько литер, представленных одним предикатом. Применение процедуры резолюции к мультидизъюнктам называется *т-резолюцией.* Кроме того, абстракцион-

ные отображения различных дизъюнктов исходной задачи могут совпадать. Например, дизъюнктам **((p(x,y), ¬q(b,y))** и **((p(x,z), ¬q(a,b))** соответствует одно и то же отображение **(p, ¬q)**. Для правильной работы описываемого нами алгоритма необходимо включать в множества **T′** все экземпляры таких дизъюнктов.

Построив по исходному **T** множество **T′**, нетрудно получить для него все выводы, которые могут быть использованы как схемы для формирования выводов в исходном множестве. Естественно, что некоторые выводы в **T′** могут оказаться непродуктивными, то есть не соответствующими выводам в **T**.

В [3] предложена формально-грамматическая модель множеств хорновских дизъюнктов в исчислении высказываний. С ее помощью осуществляется предварительная настройка модуля ВЛР для работы с конкретными множествами **T′**, что позволяет повысить эффективность модуля, а в некоторых случаях вообще избежать его применения. Эта модель представлена в виде КС-грамматики без терминальных символов и начального нетерминала. Приведем ее определение.

Пусть $\mathbf{E' = (d_1, ..., d_n)}$ и в дизъюнктах множества используются высказывания $\mathbf{p_1, ..., p_m}$. Образуем грамматику $\mathbf{G(E') = (N, Q)}$, где **N** – множество нетерминальных символов, **Q** – множество правил вывода. Нетерминальные символы суть высказывания: $\mathbf{N = \{p_1, ..., p_m\}}$, а правила вывода строятся по дизъюнктам из **E′** следующим образом:

1. По точному дизъюнкту $\mathbf{d_i = (p, \neg p_1, ..., \neg p_k)}$, $\mathbf{k \geq 1}$ строится правило $\mathbf{p \rightarrow p_1 ... p_k}$.

2. По унитарному позитивному дизъюнкту $\mathbf{d_i = (p)}$ строится укорачивающее правило $\mathbf{p \rightarrow \varepsilon}$, где **ε** – пустая цепочка.

НД $\mathbf{(\neg p_1, ..., \neg p_k)}$ представляется цепочкой $\mathbf{p_1 ... p_k}$, а метод ВЛР интерпретируется как поиск вывода в грамматике **G(E′)** пустой цепочки: $\mathbf{p_1 ... p_k \Rightarrow^+ \varepsilon}$.

Настройка модуля ВЛР выполняется алгоритмом, выделяющим укорачивающие символы и формирующим для каждого из них списки правил, с которых могут начинаться выводы пустой цепочки. Отметим, что правила в списках располагаются в порядке увеличения длин выводов пустой цепочки, начинающихся с этих правил. Рассмотрим этот алгоритм.

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

$$\mathbf{W_0 = \{ p \mid p \rightarrow \varepsilon \in Q \};}$$

$$\mathbf{S_0(p) = \{ R \mid R = p \rightarrow \varepsilon, R \in Q \},}$$

то есть включаем в $\mathbf{W_0}$ те символы грамматики, для которых в ней имеются укорачивающие правила вывода, а в соответствующие $\mathbf{S_0}$ – сами укорачивающие правила. Пусть уже получены $\mathbf{W_i}$ и $\mathbf{S_i(p_j)}$ для всех $\mathbf{p_j \in W_i}$. Выполним очередной шаг итеративного процесса следующим образом:

$$\mathbf{W_{i+1} = W_i \cup \{ p \mid p \rightarrow p_1 p_2 ... p_n \in Q;\ p_i \in W_i,\ i = 1, ..., n \};}$$

$$\mathbf{S_{i+1}(p) = S_i(p) \cup \{ R \mid R = p \rightarrow p_1 p_2 ... p_n;\ p_i \in W_i,\ i = 1, ..., n;\ R \in Q \setminus S_i(p)\},}$$

то есть включаем в $\mathbf{W_{i+1}}$ все символы из $\mathbf{W_i}$ и те символы грамматики, для которых найдутся правила вывода, все символы правых частей которых включены в $\mathbf{W_i}$. В соответствующие $\mathbf{S_{i+1}}$ включим правила из $\mathbf{S_i}$, а также правила, обладающие указанным выше свойством и не входящие в $\mathbf{S_i}$. Вновь включаемые в них правила вывода помещаются в концы списков.

Алгоритм заканчивает работу, когда на некотором шаге **j+1** окажется, что имеют место равенства $\mathbf{S_{j+1}(p_k) = S_j(p_k)}$ для всех $\mathbf{p_k \in N}$. Примем $\mathbf{W = W_j}$ и $\mathbf{S(p_k) = S_j(p_k)}$ для каждого $\mathbf{p_k \in W}$. Очевидно, число шагов алгоритма не превышает числа нетерминальных символов грамматики **G**.

Легко показать, что в любом выводе $\mathbf{p_k \Rightarrow^+ \varepsilon}$ могут применяться только прави-

ла из списков **S**. Подмножество **W** и списки **S** являются результатом работы алгоритма предварительной настройки.

Теперь можно перейти к описанию первой модификации алгоритма ВЛР в исчислении предикатов для множеств, представленных хорновскими дизъюнктами. Алгоритм состоит из двух основных модулей:

1. Модуль предварительной настройки. Состоит из двух шагов. На первом шаге строится абстракционное отображение **T′** множества **T**, на втором – формируется грамматика **G(E′)** и выполняется описанный выше алгоритм выделения укорачивающих символов.

2. Модуль, реализующий модифицированный метод ВЛР. Его исходными данными при заданной БЗ являются НД в ЛП, представляющий вопрос пользователя, его абстракционное отображение в ЛВ и соответствующая цепочка **ω** нетерминалов в **G(E′)**.

Соответствующий алгоритм выполняет следующие действия:

1. Осуществляет перебор вариантов вывода $\omega \Rightarrow^{+} \varepsilon$ в грамматике **G(E′)**. При этом используются списки **S** правил грамматики, что гарантирует вывод **ε** для каждого варианта, реализуемого при переборе.

2. По каждому варианту вывода в грамматике строит решение абстракционной задачи, то есть вывод **F** в ЛВ.

3. Используя выводы в ЛВ в качестве схем, строит выводы в ЛП, при этом применяются две процедуры – унификации и резолюции. Процедура унификации ищет для пары контрарных литер, то есть позитивной и негативной, представленных одним и тем же предикатом, наиболее общий унификатор (НОУ), то есть состоящую из наименьшего возможного числа пар подстановку, позволяющую сделать идентичными аргументы-термы контрарных литер. При наличии НОУ выполняется подстановка, после чего применяется процедура резолюции. В противном случае возникает ситуация *унификационного отказа*, свидетельствующая о том, что данный вывод в ЛВ оказался непродуктивным.

Для повышения эффективности был применен параллельный вариант работы алгоритма, при котором на каждом шаге работы выполняются действия, указанные в **a**, **b**, и **c**. Это позволяет прекращать непродуктивные выводы в грамматике и в ЛВ, если соответствующие шаги в ЛП окажутся невозможными из-за унификационного отказа.

Приведем пример применения описанного алгоритма.

Пример 1

Пусть задана следующая БЗ в ЛП в виде множества **E** (**a** и **b** – константы, **x** – переменная):

1) (p(x), ¬q(x), ¬u(a))
2) (p(a), ¬q(x), ¬r(b))
3) (p(x), ¬t(x))
4) (r(x), ¬ q(x), ¬ t(a))
5) (q(b), ¬t(a), ¬t(b))
6) (t(b))
7) (t(a))

Пропозициональные абстракции (множество **E′**):

1) (p, ¬q, ¬u)
2) (p, ¬q, ¬r)
3) (p, ¬t)
4) (r, ¬ q, ¬ t)
5) (q, ¬t, ¬t)
6) (t)
7) (t)

Грамматика
f(E′) = { N, Q }: N= { p, q, r, t }; Q:

1) p → qu
2) p → qr
3) p → t
4) r → qt
5) q → tt
6) t → ε
7) t → ε

Подмножество укорачивающих символов **W = {p, q, r, t}**. Отметим, что в него не попал символ **u**.

Списки правил, выделенных для символов грамматики:

S(p) = (3,2), S(q) = (5), S(r) = (4), S(t) = (6,7).

Пусть задана цель **C = p(x)&r(x)**. Ее отрицание **¬C = (¬p(x), ¬r(x))**. Абстрак-

ция **¬C = (¬p, ¬r)**. Соответствующая цепочка **ω = pr**. Оба символа **ω** являются укорачивающими. Следовательно, абстракция цели выводима. Используем параллельный алгоритм, для построения вывода исходной цели, формируя вывод в грамматике с указанием применяемых правил и соответствующие им схемы (табл. 1).

Пара литер, для которых потребовалось найти унификатор такова: **¬t(a)** и **t(b)**. Так как **a** и **b** константы, унификатора для них нет. Следовательно, данный вариант вывода оказался непродуктивным. Осуществляя перебор, алгоритм применит на последнем шаге правило 7, что приведет к успеху. Затем будут подобраны правила 6 и 7, получено следующее продолжение вывода (см. табл. 2).

Итак, успешный вариант вывода получен. Однако работа алгоритма продолжается, так как требуется найти все возможные варианты вывода. Осуществляя перебор, алгоритм применит на шаге 2 правило 7 (см. табл. 3).

Возникла ситуация, аналогичная рассмотренной выше: для пары **¬q(a)** и **q(b)** не существует унификатор и, следовательно, данный вариант вывода снова оказался непродуктивным. Читатель легко убедится, что применение на первом шаге правила (2) также не приведет к успеху. Отметим, что алгоритм не применял правило 1 – оно не попало в список правил, с которых может начинаться вывод **ε** из **p** (так как **u** не является укорачивающим).

Приведенный пример, несмотря на его модельный характер, иллюстрирует преимущества использования формально-грамматической модели. Как уже упоминалось, она позволяет осуществить предварительную настройку алгоритма на работу с конкретной базой знаний, главным достоинством которой являются последовательности правил вывода **S**, позволяю-

Табл. 1

№ шага	Вывод в **G(E′)** и номера правил	Вывод в **E′**	Вывод в **E**	Унификатор
1	**pr (3)**	**(¬p, ¬r)**	**(¬p(x), ¬r(x))**	**не нужен**
2	**tr (6)**	**(¬t, ¬r)**	**(¬t(x), ¬r(x))**	**{(x.b)}**
3	**r (4)**	**(¬r)**	**(¬r(b))**	**{(x,b)}**
4	**qt (5)**	**(¬q, ¬t)**	**(¬q(b), ¬t(a))**	**не нужен**
5	**ttt (6)**	**(¬t, ¬t, ¬t)**	**(¬t(a), ¬t(b), ¬t(a))**	**униф. отказ**

Табл. 2

№ шага	Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
5	**ttt (7)**	**(¬t, ¬t, ¬t)**	**(¬t(a), ¬t(b), ¬t(a))**	**не нужен**
6	**tt (6)**	**(¬t, ¬t)**	**(¬t(b), ¬t(a))**	**не нужен**
7	**t (7)**	**(¬t)**	**(¬t(a))**	**не нужен**
8	**ε**	**F**	**F**	–

Табл. 3

№ шага	Вывод в **G(f(E))** и номера правил	Вывод в **f(E)**	Вывод в **E**	Унификатор
2	**tr (7)**	**(¬t, ¬r)**	**(¬t(x), ¬r(x))**	**{(x.a)}**
3	**r (4)**	**(¬r)**	**(¬r(a))**	**{(x,a)}**
4	**qt (5)**	**(¬q, ¬t)**	**(¬q(a), ¬t(a))**	**униф. отказ**

щие осуществить перебор только успешных выводов в порядке увеличения их длины. Кстати, порядок следования предложений в базе знаний несущественен в противоположность Прологу, где для правильной работы интерпретатора требуется соблюдение определенных требований (ср. декларативную и процедурную семантики).

Несмотря на очевидные достоинства пропозициональной абстракции в качестве суживающей стратегии, у нее имеется и недостаток, заключающийся в большом числе выводов в абстракционной задаче. Причина в том, что в ЛВ полностью игнорируются аргументы предикатов исходной задачи. Поэтому нами были разработаны некоторые другие виды абстракций, в большей степени, чем пропозициональная, учитывающие особенности исходных БЗ. Один из них представлен ниже.

Так называемая абстракция второго уровня [4, 5] определяется следующим образом. Класс задач $\mathbf{K_1}$ – это по-прежнему проблема дедукции для множеств **T** в логике предикатов. Класс $\mathbf{K_2}$ строится следующим образом. Пусть дизъюнкт $\mathbf{d = (l_1, ..., l_n)}$, где каждая литера $\mathbf{l_i}$ есть предикат $\mathbf{p_i(t_1, ..., t_n)}$ или его отрицание. Тогда его абстракционное отображение $\mathbf{d' = (l'_1, ..., l'_n)}$, где

a) если все термы $\mathbf{t_j}$ $\mathbf{(1 \le j \le n)}$ предиката $\mathbf{p_i}$ являются константами, то $\mathbf{l'_i = l_i}$;

b) если среди $\mathbf{t_j}$ есть хотя бы одна переменная, то $\mathbf{l'_i = p_i(t'_1, ..., t'_n)}$ или $\mathbf{l'_i = \neg p_i(t'_1, ..., t'_n)}$ соответственно, причем если $\mathbf{t_j}$ – константа, то $\mathbf{t'_j = t_j}$, в противном случае $\mathbf{t'_j}$ = «_».

Как видно из определения, абстракционные отображения сохраняют число аргументов-термов исходных предикатов, а также термы, являющиеся константами. Однако термы-переменные заменяются нейтральным символом «_». В [5] эти отображения названы *псевдо-предикатами*.

Алгоритм решения абстракционной задачи в данном случае состоит из двух процедур – псевдо-унификации и резолюции. Рассмотрим первую из них. Псевдо-унификация не предполагает формирования унификатора и выполнения подстановки. Вместо этого проверяется отношение *соответствия* контрарных литер, в связи с чем псевдо-унификатор может рассматриваться как предикат, принимающий значение «истина» (true), если для контрарных литер указанное отношение выполнено, и «ложь» (false) в противном случае.

Процедура псевдо-унификации достаточно проста. Отношение соответствия для контрарных литер выполнено, если:

a) они представлены совпадающими псевдо-предикатами;

b) они представлены псевдо-предикатами $\mathbf{P(t'_1, ..., t'_n)}$ и $\mathbf{P(t''_1, ..., t''_n)}$ и для каждой пары термов выполнено одно из следующих двух условий: либо они совпадают, либо один из термов представлен знаком «_», а второй – константой.

Обозначим процедуру псевдо-унификации через $\mathbf{pu(l_1, l_2)}$. Процедура резолюции не имеет существенных отличий от стандартной.

Так же как в случае пропозициональной абстракции, для БЗ может быть выполнена предварительная настройка с помощью ее формально-грамматической интерпретации, которая определяется следующим образом.

Пусть $\mathbf{E' = (d_1, ..., d_n)}$ и в дизъюнктах множества используются псевдо-предикаты $\mathbf{p_1, ..., p_m}$ (для простоты мы опускаем аргументы). Образуем грамматику $\mathbf{G(E') = (N, Q)}$, где **N** – множество нетерминальных символов, **Q** – множество правил вывода. Нетерминальные символы суть псевдо-предикаты: $\mathbf{N = \{p_1, ..., p_m\}}$, а правила вывода представляются схемами, которые строятся так:

1. По точному дизъюнкту $\mathbf{d_i = (p, \neg p_1, ..., \neg p_k)}$, $\mathbf{k \ge 1}$ строится схема $\mathbf{p \rightarrow p_1 ... p_k}$.

2. По унитарному позитивному дизъюнкту $\mathbf{d_i = (p)}$ строится схема укорачивающих правил $\mathbf{p \rightarrow \varepsilon}$.

Каждая схема $\mathbf{p \rightarrow \omega}$ задает множество правил вывода следующим образом. Пусть **P** – множество псевдо-предикатов, для которых выполнено отношение соответствия с **p**: $\mathbf{pu(p, p') = true}$, если $\mathbf{p' \in P}$.

Тогда $\mathbf{p' \to \omega}$ является правилом, заданным указанной схемой.

НД $(\neg p_1, ..., \neg p_k)$ представляется цепочкой $p_1...p_k$, а метод ВЛР интерпретируется как поиск вывода в грамматике $G(E')$ пустой цепочки: $p_1...p_k \Rightarrow^+ \varepsilon$.

Аналогично случаю пропозициональной абстракции, настройка модуля ВЛР выполняется алгоритмом, выделяющим укорачивающие символы и формирующим для каждого из них списки схем правил, с которых могут начинаться выводы пустой цепочки. Однако алгоритм выделения укорачивающих символов сложнее вышеописанного.

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

$$W_0 = \{ p \mid p \to \varepsilon \in Q \};$$

$$S_0(p) = \{ R \mid R = p \to \varepsilon, R \in Q \},$$

то есть включаем в W_0 те символы грамматики, для которых в ней имеются схемы укорачивающих правил вывода, а в соответствующие S_0 – сами схемы.

Пусть уже получены W_i и $S_i(p)$ для всех $p \in W_i$. Выполним очередной шаг итеративного процесса следующим образом:

$W_{i+1} = W_i \cup \{ p \mid p \to p_1p_2 ... p_n \in Q;$ **для** p_i **найдется такой** $p_i' \in W_i$, **что** $pu(p_i, p_i') = true$, $i = 1, ..., n \};$

$S_{i+1}(A) = S_i(A) \cup \{ R \mid R = p \to p_1p_2 ... p_n$; $R \in Q \setminus S_i(p)$; **R удовлетворяет условию, сформулированному строкой выше}.**

Таким образом, в W_{i+1} включаются все символы из W_i и те символы грамматики, для которых найдутся схемы, все символы правых частей которых либо включены в W_i, либо для них найдутся символы из W_i, удовлетворяющие отношению соответствия. В соответствующие S_{i+1} включим схемы из S_i, а также схемы, обладающие указанным выше свойством и не входящие в S_i. Вновь включаемые в них схемы помещаются в концы списков.

Алгоритм заканчивает работу, когда на некотором шаге $j+1$ окажется, что имеют место равенства $S_{j+1}(p) = S_j(p)$ для всех $p \in N$. Примем $W = W_j$ и $S(p) = S_j(p)$ для каждого $p \in W$. Очевидно, число шагов алгоритма не превышает числа нетерминальных символов грамматики $G(E')$.

Подмножества W и S обладают свойствами, несколько отличными от свойств одноименных подмножеств при использовании пропозициональной абстракции. Нетерминал $p \in W$ определяет семейство укорачивающих символов, в которое входит он сам, а также все символы p' такие, что $pu(p, p') = true$. В процессе вывода могут применяться правила, определенные схемами из S. Эти особенности позволяют, с одной стороны, проверить выводимость ε из нетерминала, воспользовавшись подмножеством W, а с другой стороны, выполнять процедуру псевдоунификации уже на стадии формирования вывода в грамматике. Обе эти возможности очевидным образом повышают эффективность ВЛР. Отметим также, что аналогично пропозициональной абстракции предварительная настройка гарантирует перебор лишь успешных вариантов выводов в грамматике и в абстракционной задаче.

Вторая модификация алгоритма ВЛР, разработанная на основе абстракции второго уровня, по своей структуре схожа с модификацией на основе пропозициональной абстракции. На каждом шаге в параллельном режиме выполняется по одному шагу вывода в грамматике, в абстракции и в исчислении предикатов. Проиллюстрируем работу алгоритма примером, взятым из [5]. В этой публикации предварительная настройка алгоритма не применялась. Поэтому с помощью этого примера можно продемонстрировать преимущества, достигаемые при выполнении предварительной настройки.

Пример 2

Пусть задана следующая БЗ в ЛП в виде множества **E** (**a, b, c, d, e** – константы, **w, x, y, z** – переменные):

1. $p_1(a)$
2. $p_2(b,a,w)$
3. $p_3(c,a)$
4. $(p_1(w), \neg p_4(a))$
5. $(p_1(x), \neg p_3(c,y))$
6. $(p_2(x,z,a), \neg p_3(d,z))$
7. $(p_5(x,e), \neg p_1(b))$
8. $(p_5(x,a), \neg p_3(x,y), \neg p_2(b,a,e))$

Применяя абстракцию второго уровня, получим **E′**:

1. $p_1(a)$
2. $p_2(b,a,\_)$
3. $p_3(c,a)$
4. $(p_1(\_), \neg p_4(a))$
5. $(p_1(\_), \neg p_3(c,\_))$
6. $(p_2(\_,\_,a), \neg p_3(d,\_))$
7. $(p_5(\_,e), \neg p_1(b))$
8. $(p_5(\_,a), \neg p_3(\_,\_), \neg p_2(b,a,e))$

Осуществим предварительную настройку алгоритма, для чего сформируем грамматику:

1. $p_1(a) \rightarrow \varepsilon$
2. $p_2(b,a,\_) \rightarrow \varepsilon$
3. $p_3(c,a) \rightarrow \varepsilon$
4. $p_1(\_) \rightarrow p_4(a)$
5. $p_1(\_) \rightarrow p_3(c,\_)$
6. $p_2(\_,\_,a) \rightarrow p_3(d,\_)$
7. $p_5(\_,e) \rightarrow p_1(b)$
8. $p_5(\_,a) \rightarrow p_3(\_,\_)p_2(b,a,e)$

Вычислим множества **W** и **S**:

$W = \{ p_1(a), p_2(b,a,\_), p_3(c,a), p_1(\_), p_5(\_,a), p_5(\_,e)\}.$

$S(p_1(a)) = (1), S(p_2(b,a,\_)) = (2), S(p_3(c,a)) = (3), S(p_1(\_)) = (5), S(p_5(\_,a)) = (8), S(p_5(\_,e)) = (7).$

Пусть задана цель

$C = p_5(x,y)\&p_2(b,z,a).$

Ее отрицание

$\neg C = (\neg p_5(x,y), \neg p_2(b,z,a)).$

Абстракция второго уровня:

$(\neg p_5(\_,\_) , \neg p_2(b,\_,a)).$

Соответствующая цепочка

$\omega = p_5(\_,\_)p_2(b,\_,a).$

Проверим выводимость ε из ω:

$pu(p_5(\_,\_),p_5(\_,a)) = true;$
$pu(p_2(b,\_,a), p_2(b,a,\_)) = true.$

Следовательно, абстракция цели выводима. Используя параллельный алгоритм, построим вывод исходной цели, формируя вывод в грамматике с указанием применяемых правил и соответствующих им схем (табл. 4).

Итак, найден первый вариант вывода. Но работа алгоритма продолжается, так как имеется еще один вариант вывода (табл. 5).

Табл. 4

№ шага	Вывод в **G(E′)** и номера правил	Вывод в **E′**	Вывод в **E**	Унификатор
1	$p_5(\_,\_)p_2(b,\_,a)$ (8)	$(\neg p_5(\_,\_), \neg\ p_2(b,\_,a))$	$(\neg p_5(x,y), \neg p_2(b,z,a))$	{(y,a)}
2	$p_3(\_,\_)p_2(b,a,e)p_2(b,\_,a)$ (3)	$(\neg p_3(\_,\_), \neg p_2(b,a,e), \neg p(b,\_,a))$	$((\neg p_3(x,y), \neg p_2(b,a,e), \neg p_2\ (b,z,a))$	{(x.c), (y,a)}
3	$p_2(b,a,e)p_2(b,\_,a)$ (2)	$(\neg p_2(b,a,e), \neg p(b,\_,a))$	$(\neg p_2(b,a,e), \neg p(b,z,a))$	{(w,e)}
4	$p_2(b,\_,a)$ (2)	$(\neg p(b,\_,a))$	$(\neg p(b,z,a))$	{(z,a), (w,a)}
5	ε	**F**	**F**	–

Табл. 5

№ шага	Вывод в **G(E′)** и номера правил	Вывод в **E′**	Вывод в **E**	Унификатор
1	$p_5(\_,\_)p_2(b,\_,a)$ (7)	$(\neg p_5(\_,\_), \neg\ p_2(b,\_,a))$	$(\neg p_5(x,y), \neg p_2(b,z,a))$	{(y,e)}
2	$p_1(b)\ p_2(b,\_,a)$ (5)	$(\neg p_1(b), \neg p_2(b,\_,a))$	$((\neg p_1(b), \neg p_2(b,z,a))$	{(x.b)}
3	$p_2(b,\_,a)$ (2)	$(\ \neg p_2(b,\_,a))$	$(\neg p_2(b,z,a))$	{(z,a), (w,a)}
4	ε	**F**	**F**	–

Работа алгоритма закончена. Мы видим, что были найдены два варианта вывода в грамматике и в абстракции. Оба вывода оказались продуктивными и позволили сформировать выводы в ЛП. При переборе не использовалась схема 4, так как она не вошла в $\mathbf{S(p_1(\_))}$. В [5] указывалось, что при решении данной задачи (без предварительной настройки) были получены четыре псевдо-унификационных отказа и один непродуктивный вывод. Таким образом, предварительная настройка значительно повысила эффективность ВЛР.

Литература

1. *Plaisted D. A.* Theorem Proving with Abstraction // Artificial Intelligence. Vol. 16, №. 1. 1981. P. 47–108.

2. *Анисимова И.Н., Братчиков И.Л.* Двухэтапный алгоритм логического вывода в ЭОС // Труды Международного семинара «Искусственный интеллект в образовании». Казань, 1996. С. 16–19.

3. *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.

4. *Bratchikov I.L.* Application of Abstraction Method for Search of Logical Inference in Knowledge Bases // Proceedings of FDPW'2001–2002: Advances in Methods of Modern Information Technology, Vol. 4. Petrozavodsk, 2003. P. 167–178.

5. *Братчиков И.Л.* Суживающие стратегии поиска логического вывода на основе метода абстракций // Вестник Санкт-Петербургского университета. Сер. 10. Вып. 1–2. 2005. С. 56–63.

Abstract

In this paper restricting (reducing search) strategies for resolution algorithms of logical inference in knowledge bases formed in terms of logical model of knowledge are considered. Peculiarity of these strategies is preliminary adjustment to a concrete knowledge bases. For the adjustment an abstraction and formal-grammar interpretation of the deduction problem is used. An abstraction is a function that maps certain class of problems $\mathbf{K_1}$ onto a simpler class $\mathbf{K_2}$. In order to solve a problem $\mathbf{A} \in \mathbf{K_1}$ it is mapped onto a problem $\mathbf{B} \in \mathbf{K_2}$. Then **B** is solved and its solutions are used for searching solutions of the initial problem **A**. In the paper class $\mathbf{K_1}$ is the deduction problem in the first-order predicate calculus. Two abstractions are considered for $\mathbf{K_1}$. The first one is propositional abstraction in which $\mathbf{K_2}$ is the deduction problem in the propositional calculus. In the second one the deduction problem is solved for structures named pseudo-predicates. For both abstractions formal-grammar interpretations are proposed that allow to fulfill the preliminary adjustment which guarantees searching only successful variants of inference . Given examples illustrate virtues of suggested methods.

Keywords: knowle3dge base, logical model of knowledge, deduction problem, restricting strategy of logical inference, input linear resolution, abstraction method, formal grammar.

Братчиков Игорь Леонидович, доктор физико-математических наук, профессор кафедры математической теории микропроцессорных систем управления факультета прикладной математики – процессов управления СПбГУ, braigor@yandex.ru,

Сазонова Наталья Викторовна, кандидат педагогических наук, доцент кафедры информатики математического факультета РГПУ им. А.И. Герцена, natvic1959@mail.ru.

© Наши авторы, 2010.
Our authors, 2010.