Моделирование логических знаний и знаниевого вывода средствами СУБД

Т. А. Шиян

abstract. It is considered in the paper abilities of automatic conclusions of knowledge about formal theories, these abilities base on Relational Database Management System and offered by author theoretical model of representation of knowledge about formal theories. It is considered some abilities of pseudo-deduction based on the model and the structure of representation of knowledge. It is described some ways of production of new theories and knowledge about them.

1 Введение

Идея моделирования баз знаний (БЗ) и знаниевого вывода на них средствами СУБД не столь нелепа, как представляют апологеты «знаниевой» инженерии. Во-первых, у такого подхода есть экономическая подоплека: право доступа к некоторой реляционной СУБД и к интерпретатору некоторого серверного языка программирования входит обычно в минимальный пакет услуг хост-провайдеров, тогда как создание и поддержка в сети СУБЗ может потребовать значительных финансовых вложений. Во-вторых, различие между БД и БЗ в значительной степени принадлежит области идеологии. Основное различие (принимаемое в ИИ) между данными и знаниями — связанность. Данное — это относительно элементарный текст, фиксирующий некоторую информацию о предметной области. Простые данные можно объединять в более сложные. Знания — это структура, состоящая из элементарных (относительно данной структуры) данных и многочисленных и разнородных связей между ними. На основании этих связей мы можем из одних данных «выводить» другие. Но современная методология проектирования реляционных БД подразумевает разбиение данных на

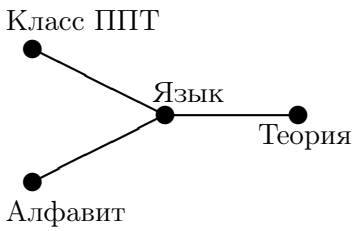
достаточно простые фрагменты, объединяемые средствами таблиц и межтабличных связей. В каждой реляционной СУБД реализована полнофункциональная реляционная алгебра, благодаря чему мы имеем возможность манипулировать данными и осуществлять необходимые выводы. Таким образом, единственными реальными (но не принципиальными) различиями между СУБД и СУБЗ являются, во-первых, необязательность автоматического поддержания связности БД (реализовано в некоторых коммерческих СУБД) и, во-вторых, хранение в БЗ не только «хорошо структурированных данных» о предметной области, но и сведений о вычислительных процедурах, посредством которых из одних данных можно получать другие. Второе отличие также не принципиально, поскольку в СУБД часто предусматриваются средства определения собственных процедур и функций, что позволяет свести роль внешнего интерфейса только к операции приложения некоторой функции к соответствующему набору данных-аргументов.

Описав предметную область как структуру взаимосвязанных объектов и представив ее в виде реляционной БД, мы получаем не только положительный ответ на вопрос о разрешимости рассуждений о данной предметной области, но и машинно реализованный механизм осуществления этих рассуждений. Ниже описывается теоретическая модель представления логических знаний о формальных теориях, положенная в основу информационной системы по логике *Theo.ru*. Из сказанного выше следует, что все рассматриваемые выводы эффективно вычисляемы средствами реализованной в СУБД реляционной алгебры. При этом акцент делается не на вопросах технической реализации теоретической модели средствами реляционной СУБД, а на вычислительных возможностях вывода знаний о предметной области в рамках данной теоретической модели и некоторых ее уточнений.

2 Классы базовых объектов и их взаимосвязи

Базовая модель содержит четыре класса объектов: «формальные теории», «формальные языки, или классы формул (ППФ)», «классы термов (ППТ)», «формальные алфавиты». Обозначу эти классы объектов через ***Theor***, ***Lang***, ***Term*** и ***Alph***. Меж-

ду объектами разных классов заданы перекрестные отношения: «теория — язык», «язык — класс ППТ», «язык — алфавит», так что элементарная структура предметной области имеет следующий вид:



Класс ППТ и алфавит понимаются как атрибуты (в терминологии объектно-ориентированного подхода) языка, а язык — как атрибут теории. Поскольку это отношения типа много — один, то можно представить их в виде функций:

- $\boldsymbol{L}(T)$ — язык теории T;
- $\boldsymbol{A}(L)$ — алфавит языка L;
- $\boldsymbol{Tr}(L)$ — класс термов языка L.

Такое представление данных отношений вполне соответствует способу их задания (в описании каждого объекта явно указывается значение соответствующего атрибута). Суперпозицией трех базовых функций ($\boldsymbol{A}$, $\boldsymbol{Tr}$ и $\boldsymbol{L}$) можно определить еще две:

- $\boldsymbol{A}_t(T) =_{df} \boldsymbol{A}(\boldsymbol{L}(T))$ — алфавит теории T;
- $\boldsymbol{Tr}_t(T) =_{df} \boldsymbol{Tr}(\boldsymbol{L}(T))$ — класс термов теории T;

Через описанные функции (отношения) можно задать ряд классификаций объектов модели. Для языков это — классификации по множеству ППТ и по алфавиту, а для теорий — классификации по языку, по множеству ППТ и по алфавиту. Знания о классификации объектов можно представить различными способами. В *Theo.ru* реализованы три стратегии представления таких знаний. Базовый подход основан на вычислении следующих функций:

- $C_{lang-a}(A) =_{df} \{x \ / \boldsymbol{A}(x) = A\}$ — множество языков в алфавите A;
- $C_{lang-t}(Tr) =_{df} \{x \ / \boldsymbol{Tr}(x) = Tr\}$ — множество языков с классом термов Tr;
- $C_{theor-l}(L) =_{df} \{x \ / \boldsymbol{L}(x) = L\}$ — множество теорий в языке L;
- $C_{theor-a}(A) =_{df} \{x \ / \boldsymbol{A}_t(x) = A\}$ — множество теорий в алфавите A;
- $C_{theor-t}(Tr) =_{df} \{x \ / \boldsymbol{Tr}_t(x) = Tr\}$ — множество теорий с классом термов Tr.

Второй подход задается следующими функциями:

- $K_{lang-a}(L) =_{df} C_{lang-a}(\boldsymbol{A}(L)) = \{x \ / \boldsymbol{A}(x) = \boldsymbol{A}(L)\}$ — множество языков в том же алфавите, что и язык L;
- $K_{lang-t}(L) =_{df} C_{lang-t}(\boldsymbol{Tr}(L)) = \{x \ / \boldsymbol{Tr}(x) = \boldsymbol{Tr}(L)\}$ — множество языков с тем же классом термов, что и язык L;
- $K_{theor-l}(T) =_{df} C_{theor-l}(\boldsymbol{L}(T)) = \{x \ / \boldsymbol{L}(x) = \boldsymbol{L}(T)\}$ — множество теорий в том же языке, что и теория T;
- $K_{theor-a}(T) =_{df} C_{theor-a}(\boldsymbol{A}_t(T)) = \{x \ / \boldsymbol{A}_t(x) = \boldsymbol{A}_t(T)\}$ — множество теорий в том же алфавите, что и теория T;
- $K_{theor-t}(T) =_{df} C_{theor-t}(\boldsymbol{Tr}_t(T)) = \{x \ / \boldsymbol{Tr}_t(x) = \boldsymbol{Tr}_t(T)\}$ — множество теорий с тем же классом термов, что и теория T.

Третий способ представления знаний о классификации можно получить введением отношения эквивалентности и последующей факторизацией классифицируемого множества.

- $L_1 \sim_a L_2 \Leftrightarrow_{df} \boldsymbol{A}(L_1) = \boldsymbol{A}(L_2)$ — отношение эквивалентности языков по алфавиту;
- $L_1 \sim_t L_2 \Leftrightarrow_{df} \boldsymbol{Tr}(L_1) = \boldsymbol{Tr}(L_2)$ — отношение эквивалентности языков по классу термов;

- $T_1 \sim_l T_2 \Leftrightarrow_{df} \boldsymbol{L}(T_1) = \boldsymbol{L}(T_2)$ — отношение эквивалентности теорий по языку;
- $T_1 \sim_{ta} T_2 \Leftrightarrow_{df} \boldsymbol{A}_t(T_1) = \boldsymbol{A}_t(T_2)$ — отношение эквивалентности теорий по алфавиту;
- $T_1 \sim_{tt} T_2 \Leftrightarrow_{df} \boldsymbol{Tr}_t(T_1) = \boldsymbol{Tr}_t(T_2)$ — отношение эквивалентности теорий по классу термов.

В результате факторизации по этим отношениям получаем следующие разбиения (классификации) множеств ***Lang*** и ***Theor***: $[\boldsymbol{Lang}]_{\sim a}$, $[\boldsymbol{Lang}]_{\sim t}$, $[\boldsymbol{Theor}]_{\sim l}$, $[\boldsymbol{Theor}]_{\sim ta}$, $[\boldsymbol{Theor}]_{\sim tt}$.

В заключение хочу еще раз отметить, что все выводы, рассмотренные в данном параграфе, основаны (кроме реализованной в СУБД реляционной алгебры) всего лишь на трех базовых функциях (отношениях) $\boldsymbol{A}$, $\boldsymbol{Tr}$ и $\boldsymbol{L}$, задаваемых в таблицах БД при определении языков и теорий.

3 Отношения порядка на классах базовых объектов

Первоначальной основной целью разработки *Theo.ru* было создание системы автоматической $3D$-визуализации соотношений формальных теорий по дедуктивной силе. Поэтому в модель исходно было введено отношение частичного строгого порядка, соответствующее соотношению формальных теорий по дедуктивной силе. Поскольку формальная теория понимается как множество формул, доказуемых в данной теории, то отношение по дедуктивной силе (для теорий с непустым множеством теорем) совпадает с отношением «быть подмножеством». Это отношение представлено в БД своей диаграммой Хассе (транзитивной редукцией), т.е. перечислением всех пар только ближайших объектов (отношение $\boldsymbol{R}_{Theor}$). Поэтому первая задача — установление по диаграмме Хассе полного отношения порядка. Транзитивное замыкание отношения R, произвольно упорядочивающего некоторое множество, можно получить, используя операцию умножения отношений, определяемую следующим образом [1]:

- $R_1 \circ R_2 = \{< x, y > / \exists z(< x, z > \epsilon R_1 \ \& < z, y > \epsilon R_2)\}$.

Если некоторое отношение R умножается само на себя, то говорят о возведении этого отношения в степень:

1. $R^1 = R$;
2. $R^{n+1} = R^n \circ R$.

Операция транзитивного замыкания R определяется как объединение всех степеней отношения R:

- $Trans(R) = \bigcup\limits_{i=1} R^i$.

Поскольку рассматриваемые множества конечны и R — редукция отношения строгого порядка (т.е. не содержит петель и циклов), то транзитивное замыкание R получается конечным объединением первых n степеней отношения R, причем, n — меньше мощности самого множества — носителя R:

- $Trans(R) = \bigcup\limits_{i=1}^{n} R^i$,

где n — степень последнего не пустого R^i. Для получения нестрогого частичного порядка $Trans(R)$ необходимо объединить с соответствующим диагональным отношением.

Наоборот, из отношения строгого (полного и некоторых случаев неполного) порядка R можно получить его транзитивную редукцию (диаграмму Хассе) по формуле:

- $R \backslash R^2$.

Основным типом задач, который приходится решать системе *Theo.ru* в процессе ее работы, является вывод из знаний о порядке на некотором классе объектов знаний о порядке на некоторой произвольной выборке объектов из этого класса. Рассмотрим подробней проводимые при этом вычисления. Пусть на множестве M задано отношение строгого порядка R и имеется описание упорядоченного множества $< M, R >$ в виде системы $< M, R\prime >$, где $R\prime$ — транзитивная редукция R. Пусть также задана произвольная выборка из M, обозначим ее — M_1. Задача — построить описание $< M_1, R_1\prime >$ упорядоченного множества, которое получается из $< M, R >$ за счет сокращения M до M_1. Поскольку множество M_1 уже задано, то остается найти отношение R_1. Эта задача решается следующим алгоритмом.

1. $R = Trans(R\prime)$ — восстанавливаем $R\prime$ до R;
2. $R_1 = (M_1 \times M_1) \cap R$ — получаем ограничение порядка R на множество M_1;
3. $R_1\prime = R_1 \backslash (R_1 \circ R_1)$ — получаем искомую транзитивную редукцию порядка R_1.

Поскольку объекты остальных классов также понимаются как множества (множество букв, множество термов и множество формул), то и на них определяется отношение порядка, совпадающее с отношением «быть подмножеством» (соответствующие транзитивные редукции, хранимые в БД: $\boldsymbol{R}_{Lang}$, $\boldsymbol{R}_{Term}$, $\boldsymbol{R}_{Alph}$). Причем, эти отношения таковы, что функции $\boldsymbol{A}$, $\boldsymbol{Tr}$, $\boldsymbol{L}$, $\boldsymbol{A}_t$, $\boldsymbol{Tr}_t$ задают гомоморфизмы относительно соответствующих нестрогих порядков:

- $T_1 \subseteq T_2 \Rightarrow \boldsymbol{L}(T_1) \subseteq \boldsymbol{L}(T_2)$;
- $L_1 \subseteq L_2 \Rightarrow \boldsymbol{A}(L_1) \subseteq \boldsymbol{A}(L_2)$;
- $L_1 \subseteq L_2 \Rightarrow \boldsymbol{Tr}(L_1) \subseteq \boldsymbol{Tr}(L_2)$;
- $T_1 \subseteq T_2 \Rightarrow \boldsymbol{A}_t(T_1) \subseteq \boldsymbol{A}_t(T_2)$;
- $T_1 \subseteq T_2 \Rightarrow \boldsymbol{Tr}_t(T_1) \subseteq \boldsymbol{Tr}_t(T_2)$.

Таким образом, имея описание порядка на $\boldsymbol{Theor}$ и функции $\boldsymbol{A}$, $\boldsymbol{Tr}$, $\boldsymbol{L}$ ($\boldsymbol{A}_t$ и $\boldsymbol{Tr}_t$ — производны), можно вывести некоторые знания о порядках на $\boldsymbol{Lang}$, $\boldsymbol{Term}$ и $\boldsymbol{Alph}$:

- $T_1 \subseteq T_2 \ \& \ \boldsymbol{L}(T_1) \neq \boldsymbol{L}(T_2) \Rightarrow \boldsymbol{L}(T_1) \subset \boldsymbol{L}(T_2)$;
- $L_1 \subseteq L_2 \ \& \ \boldsymbol{A}(L_1) \neq \boldsymbol{A}(L_2) \Rightarrow \boldsymbol{A}(L_1) \subset \boldsymbol{A}(L_2)$;
- $L_1 \subseteq L_2 \ \& \ \boldsymbol{Tr}(L_1) \neq \boldsymbol{Tr}(L_2) \Rightarrow \boldsymbol{Tr}(L_1) \subset \boldsymbol{Tr}(L_2)$;
- $T_1 \subseteq T_2 \ \& \ \boldsymbol{A}_t(T_1) \neq \boldsymbol{A}_t(T_2) \Rightarrow \boldsymbol{A}_t(T_1) \subset \boldsymbol{A}_t(T_2)$;
- $T_1 \subseteq T_2 \ \& \ \boldsymbol{Tr}_t(T_1) \neq \boldsymbol{Tr}_t(T_2) \Rightarrow \boldsymbol{Tr}_t(T_1) \subset \boldsymbol{Tr}_t(T_2)$.

Правда, эти знания могут оказаться недостаточными, поэтому в *Theo.ru* реализована частично избыточная модель со всеми четырьмя изначально заданными отношениями порядка.

Все знания о порядке на одном классе объектов можно выводить из знаний о порядке на другом классе, если гомоморфизм (из второго класса в первый) обладает полнотой относительно этих порядков. Пусть имеются два произвольно упорядоченных множества $< M_1, R_1 >$ и $< M_2, R_2 >$ и функция φ из M_1 в M_2, являющаяся гомоморфизмом относительно R_1 и R_2, тогда φ — сильный гомоморфизм, если выполнено условие (1):

- $(\forall x_1, x_2 \epsilon M_2)(\exists y_1, y_2 \epsilon M_1)((x_1 R_2 x_2 \ \& \ \varphi(y_1) = x_1 \ \& \ \varphi(y_2) = x_2) \Rightarrow y_1 R_1 y_2)$.

Из этого условия следует, что любой изоморфизм будет сильным гомоморфизмом, но не наоборот. Если отношение R_2 транзитивно, то для вычисления R_2 по R_1 достаточно соблюдения «более слабого» условия (2):

- Для всех x_1 и x_2 из M_2, ближайших относительно R_2, $(\exists y_1, y_2 \epsilon M_1)((x_1 R_2 x_2 \ \& \ \varphi(y_1) = x_1 \ \& \ \varphi(y_2) = x_2) \Rightarrow y_1 R_1 y_2)$.

- Гомоморфизм буду называть полным, если для него выполнено хотя бы одно из условий (1) и (2).

Для вполне упорядоченных относительно R_2 множеств (в том числе и для всех конечных) из условия (1) следует условие (2), но это неверно для более общего случая (например, для естественных порядков на множествах рациональных и действительных чисел). Поскольку в нашей модели R_1 и R_2 — транзитивные редукции отношений порядка, то условия (1) и (2) совпадают.

Гомоморфизм ***L*** из ***Theor*** в ***Lang*** можно сделать полным, вводя в ***Theor*** тривиальные (совпадающие с множеством ППФ) теории для каждого языка из ***Lang***. Гомоморфизмы ***A*** (из ***Lang*** в ***Alph***) и ***Tr*** (из ***Lang*** в ***Term***) не являются полными ни логически, ни фактически, поскольку в БД *Theo.ru* имеются такие языки, алфавиты и классы термов, что:

- $\boldsymbol{A}(L_1) \subset \boldsymbol{A}(L_2) \ \& \ \neg(L_1 \subset L_2)$;

- $\boldsymbol{Tr}(L_1) \subset \boldsymbol{Tr}(L_2)$ & $\neg(L_1 \subset L_2)$.

Иллюстрацией для обоих случаев могут быть языки негативных сингулярных силлогистик оккамовского и аристотелевского типа, описанные в [2]. Фактической полноты гомоморфизмов $\boldsymbol{A}$ и $\boldsymbol{Tr}$ можно добиться, подбирая *ad hoc* и вводя в БД новые языки, отвечающие условиям полноты для $\boldsymbol{A}$ и $\boldsymbol{Tr}$.

4 Возможности псевдодедукции

Каждый из объектов рассмотренных классов является точкой соотнесения некоторого набора текстов, фиксирующих наши сведения о данном объекте. Дальнейшего развития базовой модели и появления новых возможностей знаниевого вывода можно добиться путем структурирования сведений, представленных в этих текстах.

Например, для каждой формальной теории кроме языка указывается множество схем аксиом и правил вывода, посредством которых можно задать данную теорию. Пусть $\boldsymbol{Rules}$ — класс множеств правил вывода и $\boldsymbol{R}$ — функция, сопоставляющая теории T множество правил вывода Rl, фигурирующих в определении теории и относительно которых она замкнута. Пусть $\boldsymbol{Axiom}$ — множество схем формул, фигурирующих в БД в качестве схем аксиом при определении теорий; $\boldsymbol{R}_{ax}$ — бинарное отношение между элементами $\boldsymbol{Theor}$ и $\boldsymbol{Axiom}$. Тогда аксиоматика теории T есть $\boldsymbol{Ax}(T) = \{x \,/\, \boldsymbol{R}_{ax}(T, Ax)\}$. Такое представление знаний об аксиоматике теорий предоставляет целый ряд интересных дополнительных возможностей знаниевого вывода. В частности, у нас появляется возможность вывода знаний о том, в какой теории доказуема некоторая формула и какие формулы доказуемы в некоторой теории. Возможность этих выводов основана на знании:

- $F \epsilon T_1$ & $T_1 \subseteq T_2 \Rightarrow F \epsilon T_2$,

что аналогично:

- $T_1 \vdash F$ & $T_1 \subseteq T_2 \Rightarrow T_2 \vdash F$.

На основании этого общего знания имеем более частное:

- если F используется в качестве аксиомы T_1 и $T_1 \subseteq T2$, то $T_2 \vdash F$.

Соответственно, снимая квантор всеобщности либо по F, либо по T_2, получаем знания:

- формула доказуема во всех теориях, в формулировках которых она используется в качестве аксиомы, и в расширениях этих теорий;

- в теории доказуемы все формулы, которые используются в качестве аксиом в ее формулировке или в формулировках ее подтеорий.

Определение множества теорий, в которых доказуема некоторая формула, и определение множества формул, которые доказуемы в некоторой теории, названы здесь псевдодедукцией, поскольку они осуществляются не на основе дедуктивных рассуждений внутри исследуемых теорий, а на основе некоторых внешних знаний о теориях и их соотношениях. Соответственно, поскольку исходные знания касаются соотношений объектов из ***Theor*** и ***Axiom***, то и новые выведенные знания касаются соотношений объектов этих же классов.

5 Алгоритмы порождения новых теорий

Анализируемый способ представления знаний о формальных теориях позволяет осуществлять еще ряд интересных выводов, пока не реализованных в *Theo.ru*. Автоматическое замыкание БД относительно некоторых из этих алгоритмов привело бы к нарушению полноты системы относительно порядка на ***Theor***. Это связано с тем, что добавление каждой новой теории к БД требует выяснения ее отношений по дедуктивной силе с теориями, уже находящимися в БД. Но процедуры сравнения теорий по дедуктивной силе не автоматизированы и в общем виде автоматизированы быть не могут в силу проблемы разрешимости (иначе путем сравнения теорий T и $T + A$ решался бы вопрос о доказуемости или не доказуемости в T произвольной формулы A из языка этой теории). Тем не менее, сама возможность осуществления таких выводов важна с теоретической точки зрения

и может найти применение в дальнейшем. Рассмотрим несколько возможных групп алгоритмов порождения знаний о новых формальных теориях на основе уже имеющихся в БД знаний.

Порождение тривиальных теорий. Как было указано выше, можно добиться полноты гомоморфизма $\boldsymbol{L}$ из ***Theor*** в ***Lang***, введя в ***Theor*** тривиальные теории (совпадающие с классами ППФ) для каждого языка из ***Lang***. Такие теории можно получать автоматически, приняв в качестве правила вывода: $A \vdash B$. Для языков с импликацией этого же можно добиться, приняв в качестве аксиомы схему формул $(A \supset B)$, а в качестве правила вывода — *modus ponens*. Возможны и некоторые другие частные алгоритмы, учитывающие возможность наличия различных правил вывода (поскольку множество формул является формальной теорией не само по себе, а относительно некоторого дедуктивного замыкания). Эта группа алгоритмов позволяет отслеживать место порожденных теорий относительно $\boldsymbol{R}_{Theor}$, поскольку созданная так теория будет максимальной для данного языка, а отношения между максимальными (тривиальными) теориями совпадают с отношениями между их классами ППФ.

Порождение теорий путем пропозиционального замыкания. При формулировке пропозициональных теорий мы не учитываем структуру простых предложений. Но если мы пропозициональные параметры заменим на элементарные предикатные формулы, то получим формулировку исходной пропозициональной теории в новом языке. Таким образом, если в ***Lang*** есть пропозициональный язык L_1 и предикатный язык L_2 такие, что множества логических (не дескриптивных) знаков в них совпадают (способ описания алфавитов позволяет это проверять), то для каждой теории T_i в L_1 можно получить ее формулировку в L_2: значения функций $\boldsymbol{Ax}$ и $\boldsymbol{R}$ для новой теории остаются как у теории T_i, а значение $\boldsymbol{L}$ меняется на L_2. Возможны модификации алгоритма, учитывающие варианты дедуктивных замыканий (множеств правил вывода) теорий в L_1 и в L_2, уже имеющихся в ***Theor***. Алгоритмы этой группы в основном позволяют отслеживать место порожденных теорий относительно $\boldsymbol{R}_{Theor}$, но это требует учета множества частных случаев. Введение некоторых дополнительных ограничений «на целостность» БД позволяет уменьшить число возможных случаев и обеспе-

чить возможность автоматического определения места каждой вновь порожденной теории относительно $\boldsymbol{R}_{Theor}$.

Объединение теорий. Объединение двух формальных теорий (относительно замыкания Cn) в одном и том же языке на базе КЛВ определяется формулой [3]:

- $T_1 + T_2 = Cn(\{T_1\} \cup \{T_2\})$.

При задании теорий некоторыми исчислениями (с одним и тем же множеством правил вывода) мы получаем $(T_1 + T_2)$ простым объединением аксиом (схем аксиом) этих исчислений. Таким образом, основным алгоритмом порождения этого частного вида объединений теорий будет:

- $\boldsymbol{L}(T_1) = \boldsymbol{L}(T_2)$ & $\boldsymbol{R}(T_1) = \boldsymbol{R}(T_2) \Rightarrow \boldsymbol{L}(T_1 + T_2) = \boldsymbol{L}(T_1)$ & $\boldsymbol{R}(T_1 + T_2) = \boldsymbol{R}(T_1)$ & $\boldsymbol{Ax}(T_1 + T_2) = \boldsymbol{Ax}(T_1) \cup \boldsymbol{Ax}(T_2)$.

Этот алгоритм не позволяет для всех случаев точно определять местоположения новой теории относительно $\boldsymbol{R}_{Theor}$. Выводимые о местоположении новой теории знания таковы:

- Если в ***Theor*** не существует такой теории T_3, чтобы $(< T_1, T_3 >, < T_2, T_3 > \epsilon \boldsymbol{R}_{Theor})$, то можно добавить в ***Theor*** теорию T_3 такую, что $T_3 = (T_1 + T_2)$, а в $\boldsymbol{R}_{Theor}$ — пары теорий $< T_1, T_3 >$ и $< T_2, T_3 >$.
- $\exists T_3(< T_1, T_3 >, < T_2, T_3 > \epsilon \boldsymbol{R}_{Theor}) \Rightarrow$
  1. $(T_1 + T_2) \subseteq T_3$,
  2. $\boldsymbol{L}(T_1) \neq \boldsymbol{L}(T_3) \Rightarrow (T_1 + T_2) \subset T_3$.

Пересечение теорий. Этот алгоритм не реализуем средствами только реляционной алгебры и требует использования символьных операций (осуществляются средствами организации интерфейса между БД и пользователем). Пересечение двух формальных теорий (относительно замыкания Cn) в одном и том же языке на базе КЛВ совпадает с их пересечением как множеств. Но получить из формулировок T_1 и T_2 формулировку $(T_1 \cap T_2)$ сложней. Если язык этих теорий содержит конъюнкцию и дизъюнкцию, то для конечно аксиоматизируемых теорий (других теорий в БД нет) мы имеем [3]:

- $(T_1 \cap T_2) = Cn(K_{T_1} \vee K_{T_2})$, где K_{T_1} и K_{T_2} — конъюнкции аксиом теорий T_1 и T_2.

Поскольку $((A \wedge B) \vee (A \wedge C)) = A \wedge (B \vee C)$, то получаем следующий алгоритм порождения пересечений теорий:

- $\boldsymbol{L}(T_1) = \boldsymbol{L}(T_2)$ & $\boldsymbol{R}(T_1) = \boldsymbol{R}(T_2) \Rightarrow \boldsymbol{L}(T_1 \cap T_2) = \boldsymbol{L}(T_1)$ & $\boldsymbol{R}(T_1 \cap T_2) = \boldsymbol{R}(T_1)$ & $\boldsymbol{Ax}(T_1 \cap T_2) = \{K_{T_1} \vee K_{T_2}\} = (\boldsymbol{Ax}(T_1) \cap \boldsymbol{Ax}(T_2)) \cup \{C_{T_1} \vee C_{T_2}\}$, где C_{T_1} и C_{T_2} — конъюнкции аксиом теорий T_1 и T_2, не вошедших в $\boldsymbol{Ax}(T_1) \cap \boldsymbol{Ax}(T_2)$.

Этот алгоритм также не позволяет для всех случаев точно определять местоположения новой теории относительно $\boldsymbol{R}_{Theor}$. Выводимые о местоположении новой теории знания таковы:

- Если в ***Theor*** не существует такой теории T_3, чтобы $(< T_3, T_1 >, < T_3, T_2 > \epsilon \boldsymbol{R}_{Theor})$, то можно добавить в ***Theor*** теорию T_3 такую, что $T_3 = (T_1 \cap T_2)$, а в $\boldsymbol{R}_{Theor}$ — пары теорий $< T_3, T_1 >$ и $< T_3, T_2 >$;
- $\exists T_3 (< T_3, T_1 >, < T_3, T_2 > \epsilon \boldsymbol{R}_{Theor}) \Rightarrow$
  1. $T_3 \subseteq (T_1 \cap T_2)$,
  2. $\boldsymbol{L}(T_1) \neq \boldsymbol{L}(T_3) \Rightarrow T_3 \subset (T_1 \cap T_2)$.

Существуют и другие способы порождения новых теорий. При этом встают две группы задач. Во-первых, задача выбора алгоритмов, порождающих хотя бы условно значимые теории. Во-вторых, определение места каждой вновь порожденной теории в уже существующем на ***Theor*** порядке $\boldsymbol{R}_{Theor}$.

6 Заключение

В первой части статьи описана базовая модель $< O, \Re >$ ($O = \{$***Theor***, ***Lang***, ***Term***, ***Alph***$\}$, $\Re = \{=$ $\boldsymbol{L}, =$ $\boldsymbol{A}, =$ $\boldsymbol{Tr}; \boldsymbol{R}_{Theor}$, $\boldsymbol{R}_{Lang}$, $\boldsymbol{R}_{Term}$, $\boldsymbol{R}_{Alph}\}$), служащая в качестве онтологии для синтеза знаний о формальных теориях, рассмотрены выводы, осуществимые в рамках базовой модели. Во второй части — описан один из возможных (реализованный в *Theo.ru*) способов представления знаний об аксиоматике формальных теорий и возможности знаниевого вывода, открываемые таким способом

представления. Поскольку описание формальной теории включает и другие блоки данных (в том числе описание языка, алфавита и класса термов), то, используя структурное, а не чисто текстовое представление других блоков, можно получить в рамках системы и новые возможности вывода знаний о формальных теориях. Кроме обычного отношения по множеству теорем между формальными теориями существуют и многие другие отношения. Учет этой информации также может существенно расширить информационные возможности системы.

Литература

[1] *Лисовский К.Ю., Марков А.С.* Базы данных. Введение в теорию и методологию. М., 2004.
[2] *Маркин В.И.* Силлогистические теории в современной логике. М., 1991.
[3] *Смирнов В.А.* Логические методы анализа научного знания. М., 2002.