АЛГЕБРАИЧЕСКИЕ БАЙЕСОВСКИЕ СЕТИ: РЕАЛИЗАЦИЯ ЛОГИКО-ВЕРОЯТНОСТНОГО ВЫВОДА В КОМПЛЕКСЕ JAVA-ПРОГРАММ

ТУЛУПЬЕВ А.Л.

УДК 004.8

Тулупьев А.Л. **Алгебраические байесовские сети: реализация логико-вероятностного вывода в комплексе java-программ.**

Аннотация. Алгебраические байесовские сети (АБС) — это логико-вероятностная модель баз фрагментов знаний с вероятностной неопределенностью. Математической моделью фрагмента знаний (ФЗ) в теории АБС выступает идеал конъюнктов с оценками вероятности их истинности, причем оценки могут быть как скалярные, так и интервальные. Алгебраическая байесовская сеть состоит из набора фрагментов званий, который рассматривается как ее первичная структура; связи между фрагментами знаний — вторичная структура АБС — представляются виде графа смежности и его подвидов (дерева смежности и цепи смежности). В статье описаны как структуры данных, которые позволяют представить в СУБД и коде программы на java фрагменты знаний, а также первичную и вторичную структуру АБС, так и реализация основных алгоритмы логико-вероятностного вывода в этих сетях.

Ключевые слова: представление неопределенности, алгебраические байесовские сети, вероятностные графические модели, фрагмент знаний, знания с неопределенностью, логико-вероятностный вывод.

Tulupyev A.L. **Algebraic Bayesian networks: an implementation of probabilistic-logic inference in a system of Java-programs.**

Abstract. Algebraic Bayesian Networks (ABN) is a probabilistic-logic model for knowledge patterns bases with uncertainty. A knowledge pattern mathematical model is a conjuncts ideal with their estimates of probability. The estimates can be scalar as well as interval. An ABN consists of a set of knowledge patterns; this set is referred to as the primary structure of the ABN. The set of links among knowledge patterns is referred to as the secondary structure of the ABN; this structure is represented with a join graph (or with its subtypes — join tree or join chain). The paper offers data structures that can represent knowledge patterns as well as the primary and secondary structures of ABNs in RDBMSs or java code as well as an implementation of probabilistic-logic inference in ABNs.

Keywords: uncertainty representation, algebraic Bayesian networks, probabilistic graphical models, knowledge pattern, knowledge with uncertainty, probabilistic-logic inference.

1. Введение. Подводя итоги одного из этапов многолетнего изучения моделей знаний с неопределенностью, В.И. Городецкий в 1993 году ввел алгебраические байесовские сети (АБС) в область исследований искусственного интеллекта (ИИ) как новую парадигму экспертных систем [2].

Активному изучению моделей знаний с неопределенностью способствуют нереализовавшиеся ожидания середины 1950-х – 1980-х годов быстрого прогресса в области масштабных промышленных вне-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

дрений экспертных систем. Источником особого научного интереса к указанным моделям послужил анализ причин, обусловивших возникшую в отрасли стагнацию. Он выявил, в частности, два вида дефицита: дефицит математических моделей для представления знаний с неопределенностью (uncertain knowledge representation bottleneck) и просто дефицит знаний (knowledge bottleneck) [17–19, 24, 34] Интенсивно развивающиеся в рамках ИИ теории вероятностных графических моделей(ВГМ) вносят существенный вклад в поиск и разработку путей преодоления дефицита двух указанных видов.

Теории ВГМ предлагают как новые модели для представления систем знаний с неопределенностью, т.е. позволяют смягчить влияние дефицита первого вида, так и новые подходы к алгоритмизации машинного обучения (machine learning, синоним~--- автоматическое обучение) таких моделей на основе накопления и обработки разнородных исходных данных, сведений или знаний с неопределенностью (в частности, отличающихся неполнотой, неточностью и имеющих нечисловой характер). Машинное обучение в свою очередь позволяет обойти некоторые препятствия, создаваемые дефицитом второго вида. Помимо этого, в теориях ВГМ с помощью методов математики и информатики исследуются предложенные модели данных и знаний с неопределенностью, принципы создания и функционирования программных средств, позволяющих представить эти модели в памяти компьютера, а также автоматизировать процессы их формирования, хранения, обработки и обучения.

К ВГМ, нацеленным в первую очередь на решение очерченных и ряда смежных проблем, в рамках искусственного интеллекта можно причислить байесовские сети доверия (введены J. Pearl'ом) [16, 21, 24, 27, 31, 34, 35, 39] в некоторой степени — марковские сети (были заимствованы из статистической физики, где классическим примером их применения является модель Изинга в изучении магнетизма) [16, 32], а также алгебраические байесовские сети (введены В.И. Городецким и формализованы автором настоящей работы [2, 6, 7, 13, 14, 16, 21, 24]).

Теория алгебраических байесовских сетей занимает свое место среди теорий ВГМ и решает общие с ними задачи, а сами алгебраические байесовские сети в первую очередь могут быть охарактеризованы как логико-вероятностные графические модели (ЛВГМ) баз фрагментов знаний (БФЗ) с неопределенностью [1, 3–25].

Основной принцип, лежащий в основе ВГМ, хорошо известен — это принцип декомпозиции. Он применяется к совокупности знаний о предметной области. Считается, что эксперт может достаточно деталь-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

но охарактеризовать связи между двумя–тремя–четырьмя утверждениями о предметной области [39] — в каком-то смысле получается «фрагмент знаний» (ФЗ). Таких фрагментов знаний много, они образуют базу фрагментов знаний (БФЗ). Однако фрагменты знаний и их базы нельзя напрямую заложить в интеллектуальную информационную систему или комплекс программ. Сначала требуется рассмотреть математические модели ФЗ и БФЗ, разработать соответствующие структуры данных и снабдить их алгоритмами обработки. Получающиеся модели должны обладать определенной «регулярностью» структуры, общностью принципов построения своих элементов, чтобы можно было использовать одни и те же алгоритмы вывода как на локальном уровне (т.е. на уровне фрагментов знаний), так и на глобальном (т.е. на уровне всей сети).

С математической точки зрения возникающие объекты могут быть рассмотрены как система случайных элементов, которая, как правило, организована в виде графа со специфическими свойствами или решетки (отсюда — графическая модель). Случайные элементы в ней [в системе] могут быть связаны друг с другом, оказывать влияние на означивания других случайных элементов; однако такие связи предполагаются достаточно редкими, немногочисленными, разряжёнными (sparse) [27, 31, 39].

Декомпозиция системы знаний дает преимущества и на психологическом (экспертном или пользовательском) уровне, поскольку получающаяся математическая модель структурирована и обозрима, и на технологическом, поскольку уменьшаются необходимые для хранения модели объемы памяти и сложность алгоритмов ее обработки как на локальном, так и на глобальном уровне. Структурированность и обозримость вероятностной графической модели также видятся ее преимуществом и при представлении сложных связей, выявленных классическим статистическим анализом или интеллектуальным анализом данных (data mining) [3, 20, 24, 34, 35].

Особый интерес с точки зрения моделирования процесса размышлений (reasoning) [33, 30, 38] специалиста-эксперта представляет логико-вероятностный подход, в котором моделью утверждения являются пропозициональные формулы (заданные над определенным алфавитом), что традиционно для формальной логики, а степень уверенности в истинности (или стохастическая неопределенность истинности) этих пропозициональных формул и сила связей между ними характеризуется с помощью оценок вероятностей: как скалярных (точечных), так и интервальных.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

В своем современном виде логико-вероятностный подход был достаточно последовательным образом внесен в область исследований искусственного интеллекта Н.~Нильссоном [36]. (В исследовании мы отталкиваемся от интерпретации и формализации его идей, предпринятой группой исследователей, наиболее активными участниками которой представляются Фейгин, Хальперн и Меггидо [28, 29].) После публикации основополагающей статьи [36] приоритет Н. Нильссона, как он сам отмечает [37], был неоднократно оспорен. В частности, указывалось, что аналогичные идеи были ранее отражены в работах де Финетти — представителя итальянской школы исследователей в области теории вероятностей. В ответ на критику Н.~Нильссон опубликовал статью [37], где в какой-то степени признавал приоритеты других исследователей; однако, на наш взгляд, остается неоспоримым, что о применении логико-вероятностного подхода к современным проблемам искусственного интеллекта достаточно детально и конструктивно первым высказался именно он.

Вместе с тем нельзя упускать из виду того, что еще в 1854 г. Дж. Буль [26] пытался применить вероятность как меру истинности (или как степень доверия к истинности) пропозициональных формул. Хотя с точки зрения языка современной математики выкладки Дж. Буля показались бы некорректными, при знакомстве с его работой обнаруживается, что он сформулировал ряд проблем (и подошел к их решению), которыми занимаются исследователи в области вероятностной логики и неопределенности в искусственном интеллекте по сей день. В частности, Дж. Буль открыл, что в ряде случаев даже при скалярных исходных данных в процессе логико-вероятностного вывода возникают задачи линейного программирования. В его время, правда, не было ни самого термина «задача линейного программирования», ни симплекс-метода для ее решения.

Хотя логико-вероятностный подход имеет длительную историю, по указанным выше причинам остается актуальным решение комплекса задач, нацеленных на его применение в интеллектуальных информационных системах или комплексах программ с интеллектуальными компонентами. В рамках этого подхода требуется разработать математические модели (допускающие интервальные оценки истинности) баз фрагментов знаний с неопределенностью с тем, чтобы по ним в свою очередь можно было бы построить структуры данных для представления накопленных знаний. Затем автоматизировать (с учетом получившихся моделей и структур) ряд операций логико-вероятностного вывода: проверка и поддержание непротиворечивости

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ФЗ и БФЗ, априорный вывод во фрагменте знаний, апостериорный вывод (распространение влияние свидетельства) в ФЗ и БФЗ. Наконец, исследовать сформированные модели и структуры данных, а также обосновать корректность и изучить некоторые свойства разработанных алгоритмов и выдаваемых ими результатов.

Цель работы — описать структуры данных, использующихся для представления объектов из теории АБС в коде программ на языке java и в реляционных базах данных, а также комплекс программ, предназначенный для проведения вычислительных экспериментов, связанных с различными видами логико-вероятностного вывода в АБС. Положения работы уточняют, дополняют и развивают ранее опубликованные [1, 4, 5, 15, 25].

2. Определения. Приведем необходимые для дальнейшего изложения сведения об индексации конъюнктов и определение графа смежности и его подвидов [13–16, 24, 25].

Алфавит — упорядоченное множество атомарных пропозициональных формул (атомов) $A=(x_1,\ldots,x_n)$. *Литерал (аргументное место)* $\tilde{x}$, где $x\in A$, может принимать одно из двух значений $\tilde{x}\in\{x,\overline{x}\}$, где $\overline{x}$ — логическое отрицание x.

$F^{\circ}=F^{\circ}(A)$ — набор всех возможных пропозициональных формул над A. $F=F(A)=F^{\circ}(A)/\equiv$ — набор классов эквивалентных пропозициональных формул над A. *Множество квантов* $Q=Q(S)=\left\{\bigwedge\limits_{x\in S}\tilde{x}\right\}$, где $S\subseteq A$ и каждое $\tilde{x}\in\{x,\overline{x}\}$ (пробегает два своих возможных значения). Например, $Q=Q(A)=\{\tilde{x}_1\tilde{x}_2\ldots\tilde{x}_n\}$.

Идеалом цепочек конъюнкций (идеалом *конъюнктов*) над заданным *множеством* $S\subseteq A$ атомарных формул назовем все положительно-означенные цепочки конъюнкций атомарных пропозициональных формул, которые можно образовать над S. Такой идеал обозначим как $S^{\Diamond}$. Он содержит пустой конъюнкт.

Мы рассматриваем на самом деле классы цепочек *конъюнкций* и не различаем цепочки конъюнкций с одинаковым набором аргументных мест и/или атомарных пропозициональных формул.

Над *пропозициональными* формулами можно определить вероятность истинности.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Математическая модель фрагмента знаний со *скалярными* (точечными) оценками вероятностей КР — это упорядоченная пара $\text{КР} = \left\langle S^{\Diamond}, p \right\rangle$, где $S \subseteq A$, а p — скалярные (точечные) оценки вероятностей, заданные на элементах идеала $p : S^{\Diamond} \to [0;1]$. *Математическая модель фрагмента знаний* с *интервальными* оценками вероятностей КР — это упорядоченная пара $\text{КР} = \left\langle S^{\Diamond}, \mathbf{p} \right\rangle$, где $S \subseteq A$, а $\mathbf{p}$ — интервальные оценки вероятностей, заданные на элементах идеала $\mathbf{p} : S^{\Diamond} \to \left\{ [a;b] \middle| a,b \in [0;1], a \le b \right\}$.

Пусть задано k-элементное подмножество атомов из алфавита $S = \left\{ x_{i_0}, x_{i_1}, \ldots, x_{i_{k-1}} \middle| x_{i_j} \in A,\ 0 \le j \le k-1,\ i_0 < i_1 < \ldots < i_{k-1} \right\}$; сопоставим ему упорядоченную последовательность его элементов $\overleftarrow{S}$, содержащую элементы множества S в обратном лексикографическом порядке $\overleftarrow{S} = \left(x_{i_{k-1}}, \ldots, x_{i_1}, x_{i_0} \right)$. Элементы $\overleftarrow{S}$ индексируются справа налево, счет позиций начинается с нуля; j-й элемент справа записывается как $\overleftarrow{S}[j]$; в наших обозначениях $\overleftarrow{S}[j] = x_{i_j}$.

Заметим, что единица в двоичной записи степени двойки 2^j стоит на той же позиции, что и элемент $\overleftarrow{S}[j]$ в последовательности $\overleftarrow{S}$. Это несложное наблюдение позволит достаточно просто решить вопрос о построении структур данных для алфавита, его подмножеств, квантов и конъюнктов.

И н д е к с а ц и я к в а н т о в. Проиндексируем [13, 21] кванты из множества квантов $Q = Q(S)$. Будем считать, что атомы в записи квантов упорядочены в обратном лексикографическом порядке. Сопоставим кванту неотрицательное целое число, в двоичной записи которого на месте отрицательных означиваний атомов стоят нули, а на месте положительных означиваний — единицы. И наоборот, неотрицательному числу в двоичной записи сопоставим квант, у которого на месте нулей стоят отрицательно означенные атомы, а на мести единиц — положительно означенные атомы. Такая индексация задает би-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

екцию между множеством квантов и целыми неотрицательными числами от нуля до $\left(2^k - 1\right)$.

Например, для $\bar{S} = \left(y_6, y_4, y_2, y_1\right)$ двоичной записи 1011_2 будет соответствовать квант $y_6\bar{y}_4 y_2 y_1$, а 1100_2 — $y_6 y_4 \bar{y}_2 \bar{y}_1$.

Индексация конъюнктов. Проиндексируем [13, 21] конъюнкты из идеала конъюнктов $C = C(S)$. Будем рассматривать конъюнкт как подпоследовательность $\vec{S}$. Тогда каждому конъюнкту можно биективно сопоставить двоичную запись числа, содержащую k позиций (на каждой из которых может находиться ноль или единица), как характеристический вектор соответствующей подпоследовательности. Такая индексация задает биекцию между идеалом конъюнктов и целыми неотрицательными числами от нуля до $\left(2^k - 1\right)$.

Например, для $\bar{S} = \left(y_6, y_4, y_2, y_1\right)$ двоичной записи 1011_2 будет соответствовать конъюнкт $y_6 y_2 y_1$, а 1100_2 — $y_6 y_4$.

Векторы вероятностей квантов и конъюнктов. Упорядоченной последовательности квантов сопоставим вектор $\mathbf{Q}^{(k)}$, в котором сверху вниз идут скалярные оценки вероятностей соответствующих квантов. Аналогично упорядоченной последовательности конъюнктов сопоставим вектор $\mathbf{P}^{(k)}$, в котором сверху вниз идут скалярные оценки вероятностей соответствующих конъюнктов [13, 21].

Индексация конъюнктов и квантов позволяет хранить сведения об оценках вероятности их истинности в массивах, что существенно облегчает разработку структур данных для представления фрагментов знаний и алгоритмов логико-вероятностного вывода в них.

Структура и состав идеала-носителя фрагмента знаний однозначно определяется тем подмножеством атомов $V \subseteq A$, из которого сформирован главный конъюнкт самого идеала. Следовательно, для задания первичной структуры АБС достаточно перечислить все такие подмножества алфавита A, из которых сформированы главные конъюнкты фрагментов знаний, вошедших в АБС.

Множество подмножеств алфавита $\mathbf{V} = \{V_i \subseteq A\}_{i=1}^{i=m}$ перечисляет наборы атомов, вошедших в главные конъюнкты максимальных ФЗ заданной алгебраической байесовской сети.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Мы исключаем из рассмотрения несобственные подмножества алфавита $V_i = A$ и $V_i = \varnothing$. В первом случае окажется, что все остальные ФЗ будут входить в ФЗ, образованный над идеалом $A^{\Diamond}$; во втором случае получится ФЗ над $\varnothing^{\Diamond}$, содержащий лишь пустую конъюнкцию, что неинформативно.

Кроме того, для любых $i \neq j$ справедливо, что $V_i \not\subset V_j$ и $V_j \not\subset V_i$, т.е. ни один элемент $\mathbf{V}$ не входит в другой. Иначе у нас в АБС были бы фрагменты знаний, которые бы полностью входили бы в другие фрагменты знаний.

Множеством вершин графа смежности является ранее определенное множество $\mathbf{V}$. Обозначения V_i при $1 \le i \le m$ рассматриваются как метки вершин графа смежности. *Весом* $W(V_i)$ вершины V_i графа смежности является множество атомов из алфавита, вошедших в V_i. Фактически $W(V_i) = V_i$, но мы сохраним различие в обозначениях для большей выразительности, хотя достаточно понимать, что метками и весами вершин графа смежности будут одни и те же множества из $\mathbf{V}$.

Граф смежности G — это *ненаправленный* граф, заданный парой множеств $(\mathbf{V}, \mathbf{E})$, где $\mathbf{V}$ — *множество вершин* (Vertex), определенное ранее, а $\mathbf{E}$ — *множество ребер* (Edge). При этом должны выполняться дополнительные условия:

1) между каждой парой несовпадающих узлов V_i и V_j, веса которых содержат общие элементы $W(V_i) \cap W(V_j) \neq \varnothing$, существует путь;
2) в веса каждого из узлов пути, указанного в предыдущем пункте, входят все элементы, общие для весов начального и конечного узлов;
3) вес одного узла графа не входит полностью в вес никакого другого узла графа [5].

Вес ребра $W\left(\{V_i, V_j\}\right)$, $\{V_i, V_j\} \in \mathbf{E}$, графа смежности G определяется как пересечение весов тех вершин, которые соединены этим ребром $W\left(\{V_i, V_j\}\right) = W(V_i) \cap W(V_j)$. Ребра с пустым весом в граф смежности не входят.

Дерево смежности T — это граф смежности без циклов (ациклический граф смежности). *Линейная цепь* (или просто *цепь*) фрагментов

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

знаний — это дерево смежности, в котором можно выделить два узла V_b и V_e, между которыми существует путь (без самопересечений), причем этот путь содержит все оставшиеся узлы дерева смежности. Следует обратить внимание на то, что узлы V_b и V_e могут и не содержать общих элементов.

Далее, как правило, будем все графы смежности, заданные над одним и тем же множеством вершин $\mathbf{V}$, считать частично упорядоченными по отношению вхождения множеств их ребер $\mathbf{E}$: $G' \prec G''$, если соответствующие множества ребер $\mathbf{E}' \subset \mathbf{E}''$. Поскольку множество вершин для всех графов смежности совпадает по нашему соглашению, графы смежности на заданном множестве вершин равны тогда и только тогда, когда совпадают множества их ребер.

3. Представление АБС в базе данных. Один из вариантов хранения первичной структуры алгебраических байесовских сетей предполагает разделение их на типы по использующимся оценкам вероятностей истинности конъюнктов. В теории АБС рассматриваются бинарные, скалярные и интервальные оценки вероятности. Соответственно, для каждого типа потребовалось бы определить отдельные сущности — для хранения только бинарных оценок (рис. 3.1а), только стохастических оценок (рис. 3.1б) или только интервальных оценок (рис. 3.1в). Общим для всех трех таблиц является глобальный индекс конъюнкта в сети (CONJ_GLOBAL_INDEX) и сеть, к которой принадлежит данный конъюнкт (ABN_ID). В случае сугубо бинарных оценок, определяется индекс положительно означенных атомов в конъюнкте (CONJ_BINARY_INDEX), сугубо скалярных — вероятность (PROBABILITY), сугубо интервальных — интервальные оценки вероятности конъюнктов (PROBABILITY_LB, PROBABILITY_UB), то есть нижняя и верхняя оценка вероятности соответственно.

Аналогично представлению отдельных типов фрагментов знаний и сетей, можно предложить представление свидетельств, диверсифицированное по их типу.

BINARY_ABN_CONJ	
ABN_ID	INTEGER
CONJ_GLOBAL_INDEX	LONG
CONJ_BINARY_INDEX	LONG

а

SCALAR_ABN_CONJ	
ABN_ID	INTEGER
CONJ_GLOBAL_INDEX	LONG
PROBABILITY	DOUBLE

б

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

INTERVAL_ABN_CONJ	
ABN_ID	INTEGER
CONJ_GLOBAL_INDEX	LONG
PROBABILITY_LB	DOUBLE
PROBABILITY_UB	DOUBLE

в

Рис. 3.1. Представление конъюнкта в реляционной таблице.

Структуру таблиц для представления детерминированных свидетельств можно увидеть на рис. 3.2а, стохастических — на рис. 3.2б, неопределенных — на рис. 3.2в. Детерминированное свидетельство представляет собой идеал конъюнктов с бинарными оценками истинности (на самом деле, достаточно задать лишь означивания атомов, вошедших во фрагмент знаний), стохастическое — со скалярными, неопределенное — с интервальными. Для хранения детерминированного свидетельства достаточно определить структуру одной таблицы; а для хранения стохастических и неточных свидетельств потребуется определение пары таблиц, находящихся в отношении 1:m, поскольку каждому такому свидетельству сопоставлен список оценок.

Более точно, для стохастических и неопределенных свидетельств используются отдельные сущности для хранения оценок вероятностей: таблица STOCHASTIC_EVIDENCE_CONJ и таблица IMPRECISE_EVIDENCE_CONJ.

Однако, допуская некоторые обобщения и компромиссы с требованиями нормальных форм, для хранения произвольного вида свидетельств можно использовать пару таблиц, представленных на рис. 3.3.

Поскольку бинарные сети представляют лишь теоретический интерес и использование их на практике не ожидается, в рамках данной работы они исключены из рассмотрения. Напротив, представление двух типов сетей, вызывающих непосредственный интерес, то есть алгебраических байесовских сетей со скалярными и интервальными оценками, заслуживает особого внимания. Снова за счет определенных компромиссов, обе сущности могут быть объединены в одну — ABN (рис. 3.4). Каждому фрагменту знаний — ABN_KP — соответствует глобальный индекс (KP_GLOBAL_INDEX) в контексте сети, к которой он принадлежит. А сущность ABN_CONJ хранит оценки вероятностей конъюнктов фрагментов знаний, по глобальному индексу конъюнкта (CONJ_GLOBAL_INDEX) и алгебраической байесовской сети, которой он принадлежит (ABN_ID).

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

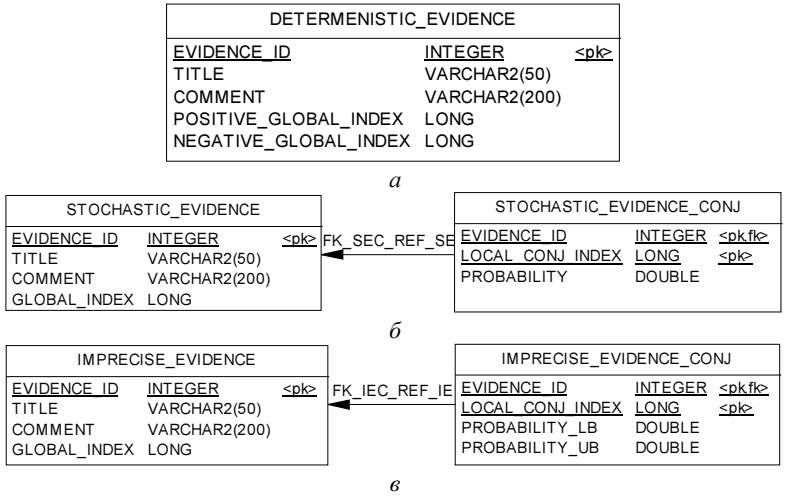

Рис. 3.2. Представление свидетельств в реляционной базе данных.

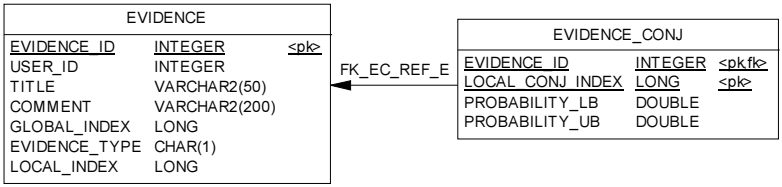

Рис. 3.3. Обобщенное представление свидетельств.

Не следует упускать из виду, что между элементами таблицы ABN_KP (которую в контексте поднятого вопроса можно рассмотреть как главную) и элементами таблицы ABN_CONJ (зависимая) фактически существует функциональная зависимость. Каждому элементу главной таблицы сопоставляются элементы зависимой, которые удовлетворяют двум условиям:

1. ABN_KP.ABN_ID == ABN_CONJ.ABN_ID;
2. ABN_CONJ.CONJ_GLOBAL_INDEX ==
   ABN_CONJ.CONJ_GLOBAL_INDEX
   && ABN_KP.KP_GLOBAL_INDEX,

где операция && — это операция побитовой конъюнкции. Последнее условие позволяет выделить все конъюнкты, вошедшие в идеал, образованный главным конъюнктом, индекс которого хранится в поле ABN_KP.KP_GLOBAL_INDEX. (Можно также сказать, что это поле

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

хранит индекс идеала конъюнктов или характеристический вектор, указывающий на то, над какими атомами построен идеал, лежащий в основе фрагмента знаний.)

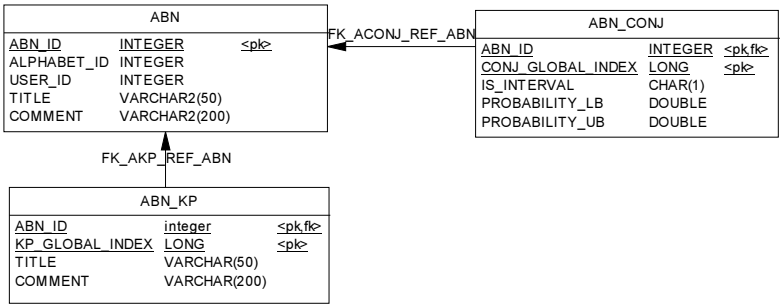

Рис. 3.4. Представление алгебраической байесовской сети.

Объединение структур для фрагментов знаний и свидетельств позволяет значительно снизить количество сходных по строению сущностей в базе данных. Это упрощает как восприятие собственно структуры базы данных, так и построение ее моделей в коде приложений, которые будут с базой данных взаимодействовать.

В частности, для задачи по пропагации конкретного свидетельства в конкретной сети потребуется использовать только одну таблицу ABN_EVIDENCE (рис. 3.5), которая хранит пользовательские сессии, сформированные из двух ключей-ссылок: на сеть (ABN_ID) и на свидетельство (EVIDENCE_ID). В случае же разделения сетей на два вида, а свидетельств — на три, неизбежно пришлось бы использовать шесть очень схожих между собой по структуре таблиц для заданий сессий над различными комбинациями типов алгебраических байесовских сетей и свидетельств.

Связи между фрагментами знаний задают вторичную структуру алгебраических байесовских сетей, которую можно, в частности, представить как цепь, дерево или граф смежности. Для представления вторичной структуры АБС организована сущность JOIN_GRAPH (рис. 6), в которой содержится указание, какого типа структура (цепь, дерево или граф) формируется из связей между фрагментами знаний, а также указывается глобальный индекс начального элемента – в атрибуте ROOT_GLOBAL_INDEX. Эти данные важны в случае представления вторичной структуры в виде цепочки. В случае графа или дерева, можно выбрать любой из элементов сети.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Более точно, в случае графа смежности по соглашению предполагается, что в поле ROOT_GLOBAL_INDEX хранится особое значение NULL, в случае дерева смежности — его корень (на роль которого, правда, действительно может претендовать любой узел), а в случае цепи смежности — один из двух ее листьев (концов).

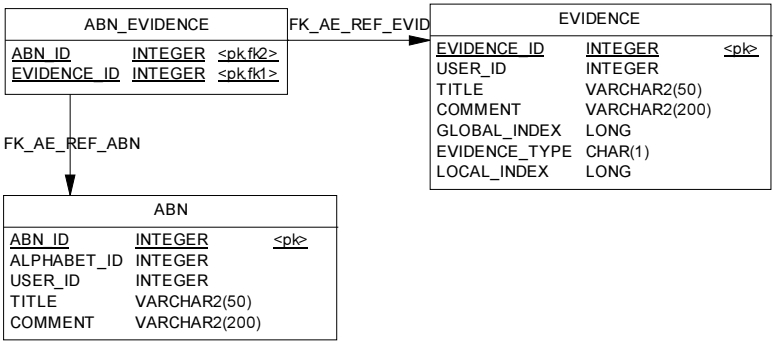

Рис. 3.5. Сессия, задающая исходные данные для апостериорного вывода.

Порядок соединения фрагментов знаний для каждой из определенных структур данной сети хранится в таблице KP_NEIGHBOR, в которой указываются связи соседних фрагментов знаний в сети. При написании программного кода рекомендуется договориться, что в случае дерева и цепи смежности рассматриваются их направленные версии, чтобы не удваивать напрасно число ребер, сведения о которых как раз и хранит таблица KP_NEIGHBOR.

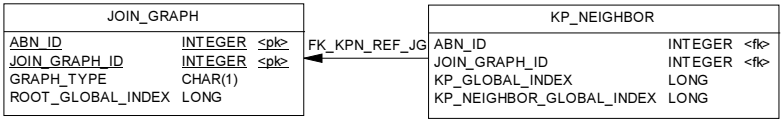

Рис. 3.6. Представление графа смежности и его разновидностей в базе данных.

Общая структура представления алгебраических байесовских сетей в базе данных, учитывающая изложенные принципы, указана на рис. 3.7.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

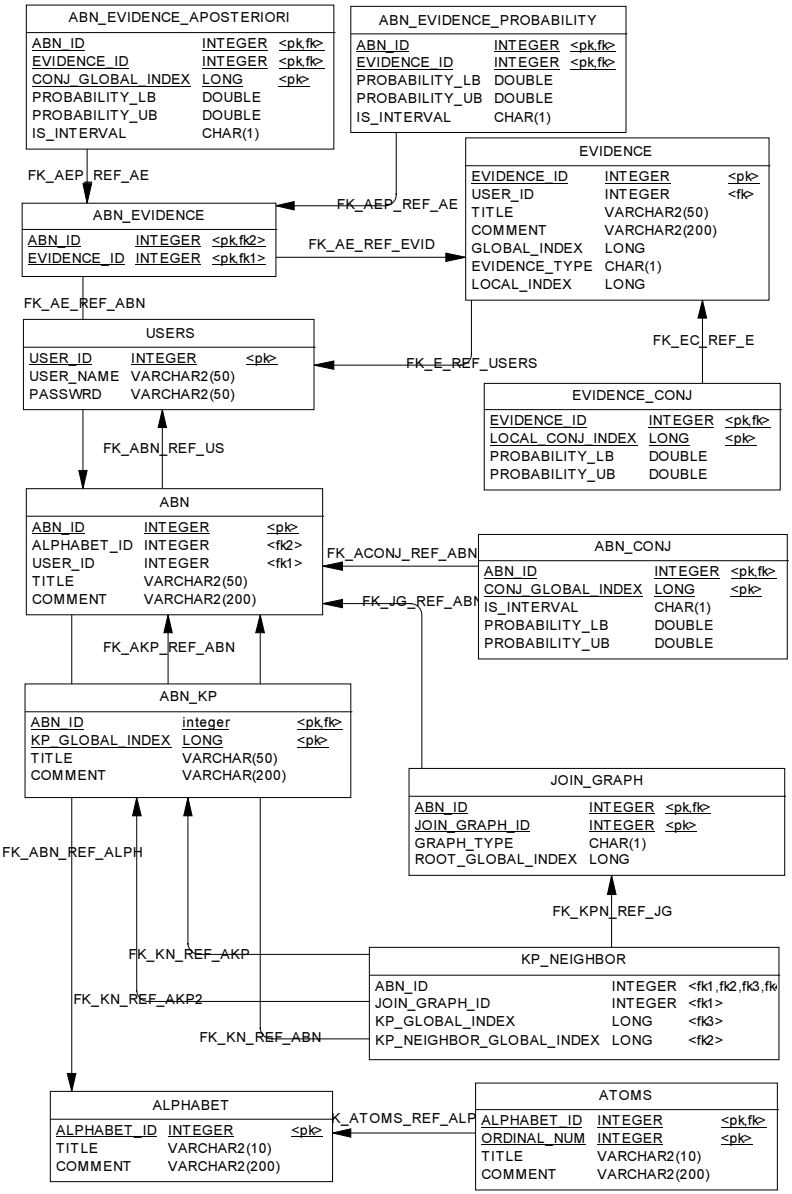

Рис. 3.7. Общая структура базы данных для представления АБС.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Она имеет «конкурентов», в частности, можно договориться о переносе признака IS_INTERVAL, указывающего на тип оценки (интервальная или скалярная) из таблицы ABN_CONJ в таблицу ABN, из таблицы EVIDENCE_CONJ в таблицу EVIDENCE, из таблиц с результатами сессий ABN_EVIDENCE_APOSTERIORI и ABN_EVIDENCE_PROBABILITY в таблицу-сессию ABN_EVIDENCE.

У каждого из вариантов есть свои преимущества, но в целом они взаимозаменяемы. Выбор зависит от того контекста, в котором будут разрабатываться и применяться базы данных и библиотеки логико-вероятностного вывода.

Следует отметить, что за пределами рассмотрения в настоящей работе остались вопросы, связанные с заданием сессий по проверке и/или поддержанию непротиворечивости (локальной, экстернальной, интернальной или глобальной) алгебраических байесовских сетей. Не вдаваясь в подробности, отметим, что принципы описания таких сессий и представления результатов логико-вероятностного вывода, с ними связанных, не сильно отличаются от описания сессий, связанных с апостериорным выводом.

4. Библиотека логико-вероятностного вывода в АБС.

4.1. Структура комплекса программ. Основной частью комплекса программ является библиотека, написанная на языке java. Согласно объектно-ориентированной парадигме программирования, каждая задача или тип данных реализованы в виде отдельного класса. Все классы разбиты на пакеты, каждый из которых является достаточно автономным модулем (рис. 4.1). В структуру пакетов входят следующие:

- algbn.data.local содержит классы и интерфейсы, определяющие и реализующие структуры данных для хранения ФЗ и свидетельств;
- algbn.data.global — классы и интерфейсы первичной структуры сети, то есть базы знаний;
- algbn.inferrers.local — классы и интерфейсы локального априорного вывода, которые работают с отдельными ФЗ;
- algbn.inferrers.global — классы и интерфейсы глобального априорного вывода, которые работают с базами знаний, опираясь на локальный вывод;
- algbn.propagators.local — классы и интерфейсы локального апостериорного вывода;

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

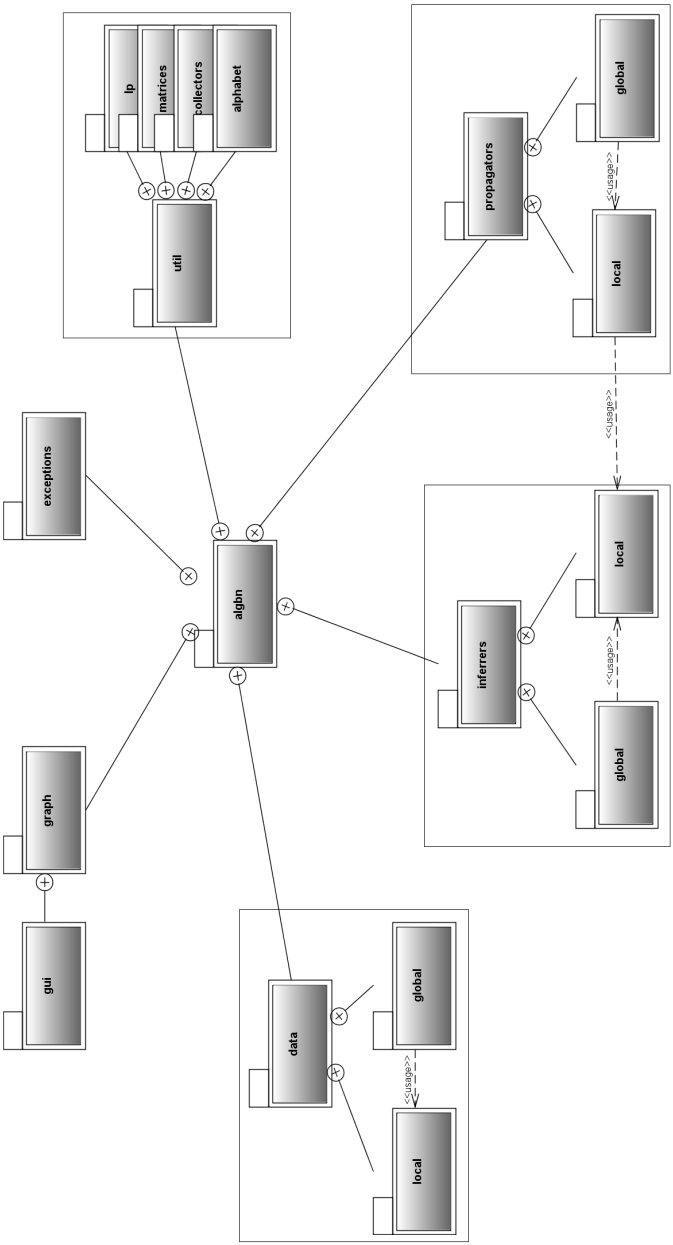

Рис 4.1. Диаграмма основных пакетов библиотеки algbn.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

- algbn.propagators.global — классы и интерфейсы глобального апостериорного вывода, которые работают с базами знаний и их вторичной структурой, опираясь на локальный вывод;
- algbn.graph — классы и интерфейсы вторичной структуры сети и алгоритм ее построения на основе первичной, а также отображения вторичной структуры сети на стандартной swing компоненте JPanel (в подпакете gui);
- algbn.utils — утилитарные классы для работы с алфавитами, матрицами, индексами;
- algbn.exceptions — классы ошибок, используемые в программе.

Помимо стандартных библиотек java, используются четыре сторонние библиотеки: OR-Objects — библиотека для решения задач линейного программирования; JGraph — библиотека для визуализации графов; JGraphLayout — библиотека для автоматической компоновки графов; JUnit-4.5 — библиотека для автоматизированного тестирования.

4.2. Локальные структуры данных. В основе реализации структур данных ее лежит разделение фрагментов знаний по типам носителя и по типам оценок вероятностей; интерфейсы {*Conjuncts, Quants, Disjuncts}KnowledgePatternI*, наследующие общий абстрактный интерфейс фрагмента знаний *KnowledgePatternBasisI*, определяют тип носителя. Установлена договоренность, что классы, реализующие фрагменты знаний, должны, во-первых, реализовывать один из этих интерфейсов, во-вторых, в методе *getKPBasisStructure()* должны возвращать соответствующее значение. Тип оценок вероятностей определяется с помощью интерфейсов чтения и записи оценок вероятностей ФЗ, приведенные на рис. 4.2.

Такая иерархия интерфейсов возникает потому, что бинарный фрагмент знаний является частным случаем скалярного, который в свою очередь является частным случаем интервального. Между интерфейсами чтения и записи существует семантическая связь, которая представлена в таблице 3.1. Но эту связь не удалось выразить путем наследования, поэтому существует договоренность о том, что в программе каждый конкретный класс обязан реализовывать оба соответствующих интерфейса чтения и записи, хотя его интерфейс содержит лишь интерфейс чтения.

В основе работы с фрагментами знаний и их оценками вероятностей лежит глобальная и локальная индексация элементов, подробно описанная в [13]. Для хранения глобального индекса конъюнкта используется тип long (точнее, его объектный аналог Long).

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

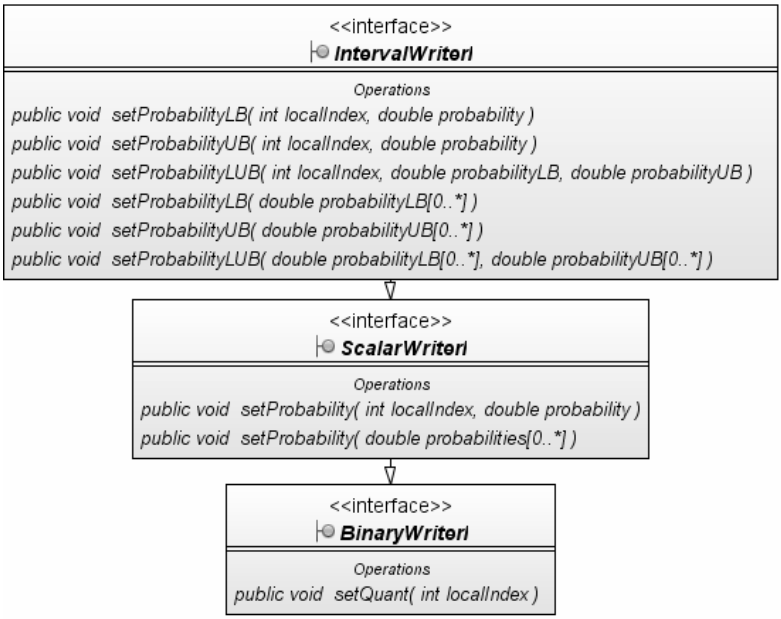

Рис. 4.2. Интерфейсы чтения и записи оценок вероятностей во ФЗ.

Таблица 4.1. **Связь между интерфейсами чтения и записи**

	Binary Knowledg PatternI	Scalar Knowledge PatternI	Interval Knowledge PatternI
BinaryReaderI	+		
ScalarReaderI	+	+	
IntervalReaderI	+	+	+
BinaryWriterI	+	+	+
ScalarWriterI		+	+
IntervalWriterI			+

Установлено ограничение на максимальное число атомов во фрагменте знаний, так как расход памяти растет экспоненциально. При 10 атомах во ФЗ интервальные оценки занимают 16 Кб памяти. Этого достаточно для эксперта, так как он едва ли сможет может задать столь большое число связей (1024).

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

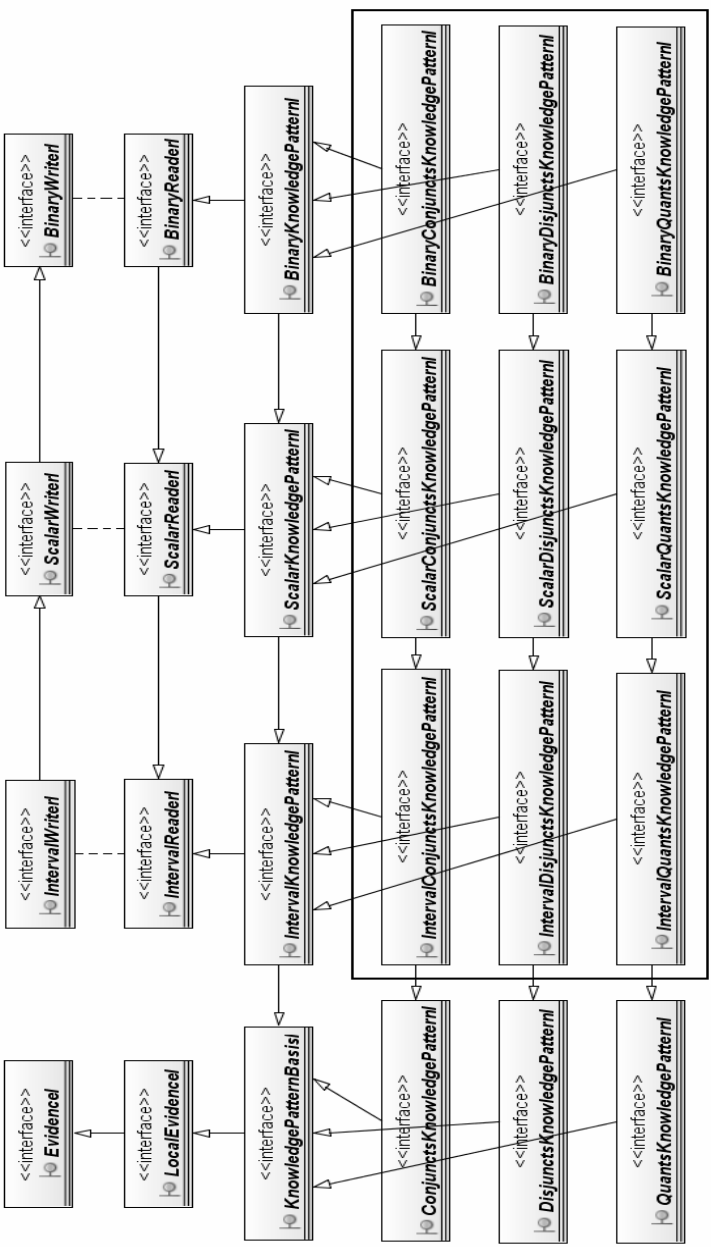

Рис. 4.3. Полная иерархия интерфейсов пакета algbn.data.local.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Из множества классов ФЗ были выделены абстрактные классы *BinaryKnowledgePattern, ScalarKnowledgePattern* и *IntervalKnowledgePattern*, реализующие общую функциональность. На рис. 4.3 изображена полная диаграмма интерфейсов локальных структур данных.

Представленная детализированная иерархия локальных структур данных позволила в других частях программы использовать параметризацию интерфейсов. Это значительно уменьшило число интерфейсов для машин логико-вероятностноговывода, работающих с разными типами ФЗ.

4.3. Первичная структура АБС. База фрагментов знаний задается своей первичной структурой, то есть набором фрагментов знаний. Реализованы параметризованный интерфейс базы знаний KnowledgeBaseI<E extends KnowledgePatternBasisI>, классы ScalarConjunctsKnowledgeBase и IntervalConjunctsKnowledgeBase для базы знаний со скалярными и интервальными оценками конъюнктов соответственно. Общая для обоих классов функциональность была выделена в абстрактный класс ConjunctsKnowledgeBase.

Первичная структура имеет интерфейс, представленный в листинге 4.1. Этот интерфейс представляет две коллекции: фрагментов знаний и конъюнктов. В качестве параметра интерфейсу передается интерфейс фрагмента знаний — ScalarConjunctsKnowledgePatternI или IntervalConjunctsKnowledgePatternI.

```
public interface KnowledgeBaseI<E extends KnowledgePatternBasisI>
{
   AlphabetI getAlphabet();
   int getSize();
   Set<Long> getPatternIndices();
   Set<Long> getElementIndices();
   E getKnowledgePattern(Long globalIndex);
   BaseElement getElement(Long globalIndex);
}
```

Листинг 4.1. Интерфейс первичной структуры базы знаний

База знаний хранит оценки конъюнктов в единственном экземпляре в виде объекта ScalarProbability или IntervalProbability, содержащих глобальный индекс и оценку вероятности конъюнкта.

База знаний является неизменяемой, доступной только для чтения. Фрагменты знаний передаются в конструкторе при создании.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

База знаний должна иметь связный максимальный граф смежности и согласованные на пересечениях оценки вероятностей. Оба этих свойства проверяются один раз при создании и если они не выполнены, создается исключение IllegalArgumentException, означающее неправильно заданные параметры. Проверить связность максимального графа смежности перед создание базы знаний можно с помощью класса algbn.graph.MaxJoinGraph. Можно проверить согласованность оценок вероятностей и пересечь их с помощью классов-утилит ScalarConjunctsIntersector и IntervalConjunctsIntersector.

Так как одной первичной структуре может соответствовать целое семейство вторичных структур, разработан особый способ их хранения. Он реализован в классе SSDescription:

```
public class SSDescription {
    protected final Long[] vertices;
    protected final IntEdge[] edges;
    …
}
```

Массив вершин создается в единственном экземпляре и никогда не меняется, поэтому используется всеми вторичными структурами, построенными над одним и тем же набором вершин. Ребра также создаются один раз (при создании максимального графа смежности), причем ребра ссылаются не на сами вершины, а на их индексы в массиве. Иногда возникает задача по глобальному индексу вершины получить его индекс в массиве. Для этих целей массив один раз сортируется при создании и в дальнейшем используется бинарный поиск.

SSDescription неудобно использовать для обхода графа, для этих целей создан класс SecondaryStructure, дополнительно строящий список смежности вершин, так же использующий индексы вершин в массиве вместо самих вершин.

```
public class SecondaryStructure implements SecondaryStructureI {
    private SSDescription description;
    private List<Integer>[] adjacencyList;
    private Type type;
    private Boolean connected;
    …
}
```

Такое разделение необходимо для экономии памяти и достаточно обоснованно. Например, для сети, состоящей из семи попарно пересекающихся ФЗ $\{x_1x_2x_7, x_1x_3x_7, x_1x_4x_7, x_4x_6x_7, x_4x_5x_7, x_6x_7, x_7x_9\}^{\Delta}$, имеется 507904 различных вариантов вторичной структуры. Используя раздельно классы SSDescription для хранения и SecondaryStructure для

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

работы с конкретной вторичной структурой, приложение визуализации потребовало выделения не менее 64 Mb памяти. Используя SecondaryStructure как для хранения, так и для работы, потребовалось выделить не менее 400 Mb памяти.

Для обхода графа в глубину, представленного в виде SecondaryStructure, реализован итератор FirstDepthIterator. Он используется для нескольких целей:

1. Проверка связности графа. При обходе помечаются уже посещенные вершины, и метод hasNext() возвращает true, если еще есть непосещенные вершины. Однако если посетить никакую другую вершину больше нельзя, то метод next() создаст исключение NoSuchElementException.
2. Распространение свидетельства по сети. В случае АБС без циклов обход в глубину работает корректно, в случае наличия циклов обход в глубину работает, но полученный результат некорректен с точки зрения теории и зависит от того, в каком порядке посещались вершины.
3. Распознавание типа вторичной структуры (граф, цикл или цепь).

Было предложено несколько алгоритмов построения семейства вторичных структур, в частности, в [4] предложен алгоритм, оптимизированный по количеству используемой памяти

4.4. Локальный синтез согласованных оценок истинности. Задача проверки непротиворечивости ФЗ и априорного вывода тесно связаны между собой и реализованы в пакете algbn.inferrers.local. Иерархия интерфейсов машин вывода представлена на рис. 4.4. За счет параметризации интерфейсов и унификации имен методов одинаковые методы были перенесены в интерфейс LocalInferrerI. Но с методом, который преобразует конъюнкты в кванты, этого сделать не удалось.

BinaryLocalInferrerI представляет интерес лишь с научной точки зрения как обобщение, так как бинарные фрагменты знаний используются только для задания детерминированных свидетельств и не используются в базе знаний.

Классы ScalarConjunctsLocalInferrer и IntervalConjunctsLocalInferrer реализуют представленные выше интерфейсы. С их помощью можно:

1. Провести проверку непротиворечивости ФЗ. После вызова метода processKP(pattern) метод isConsistent() вернет true или false;

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

2. В случае интервальных оценок знаний можно получить уточненные оценки вероятностей. Метод isNarrowed() вернет true, метод getResult() вернет ФЗ с уточненными оценками;

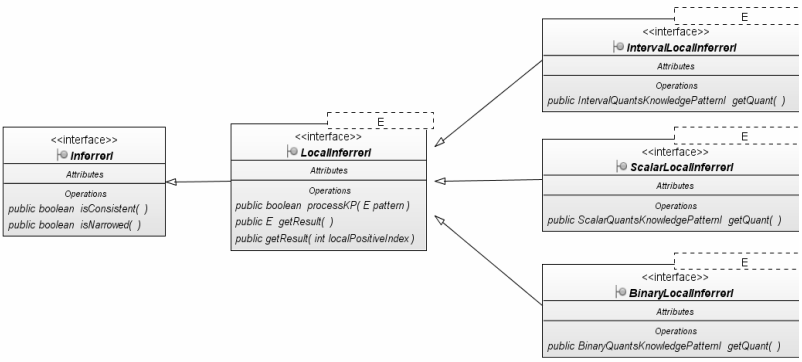

Рис. 4.4. Иерархия интерфейсов проверки непротиворечивости ФЗ.

3. Для непротиворечивого ФЗ провести переозначивание атомов. Метод getResult(localPositiveIndex) вернет ФЗ, в котором все атомы из указанного в параметрах конъюнкта остались положительно означенными, а остальные атомы переозначены;
4. Преобразовать непротиворечивый ФЗ в набор квантов;
5. Провести априорный вывод во ФЗ.

Для проведения априорного вывода во фрагмент знаний нужно добавить одну или несколько линейных форм с вещественными коэффициентами, задающими одну или несколько пропозициональных формул (листинг 4.4). При этом ФЗ становится расширенным. Эта функциональность реализована на уровне класса KnowledgePatternBase и ею обладают все имеющиеся фрагменты знаний, сейчас она используется только для проведения априорного вывода. После проведения вывода в линейной форме будут указаны границы, в которые она заведомо попадает. В случае скалярного ФЗ эти границы будут совпадать, в случае интервального — нет. Точно также можно учесть знания эксперта, заданные качественно, в виде линейного соотношения, если перед выводом задать линейной форме границы, указанные экспертом.

Перечисленная функциональность расположена в машине априорного вывода, потому что основана на одном и том же алгоритме проверки непротиворечивости. Сначала оценки вероятностей над идеалом конъюнктов переводятся в оценки вероятностей над набором

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

квантов, потом в скалярном случае просто проверяется выполнение системы неравенств, в интервальном случае — решается серия ЗЛП.

Стоит отметить, что во всех машинах вывода принято соглашение относительно входных и выходных данных. Сначала, если необходимо, устанавливаются все входные данные. Они хранятся в виде ссылок на объект и подразумевается, что они не меняются во время работы алгоритма извне (другим потоком, например). Также гарантируется, что они не будут изменены в результате работы алгоритма изнутри. Это сделано для того, чтобы избежать лишнего копирования данных. Выходные данные алгоритма всегда является новым объектом.

```
public interface LinearFormI {
    String getName();
    int getNumberOfElements();
    double[] getWeights();
    double getWeight(int i);
    void setWeight(int i, double w);
    void setWeights(double[] w);
    double getLB();
    double getUB();
    void setLB(double lowerBound);
    void setUB(double upperBound);
    void setLUB(double lowerBound, double upperBound);
    void removeBounds();
    void removeLB();
    void removeUB();
}
```

Листинг 4.4. Интерфейс линейной формы

4.5. Локальный апостериорный вывод. Алгоритм локального апостериорного вывода зависит как от типа оценок вероятностей фрагмента знаний, куда поступает свидетельство, так и от вида свидетельства. Результатом вывода является ФЗ с новыми, апостериорными оценками вероятностей, а также оценка вероятности такого свидетельства. Тип оценок вероятностей результата совпадает с наиболее общим из типов оценок в свидетельстве и во фрагменте знаний.

Каждый алгоритм инкапсулирован в отдельном классе (машине вывода), но работает по общей схеме:

1. Сначала с помощью метода setPattern(pattern) задается ФЗ. Можно использовать один и тот же класс множество раз, с

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

разными ФЗ. ФЗ не копируется и предполагается, что он не меняется во время работы алгоритма ни извне, ни самим алгоритмом;

2. Для пропагации свидетельства вызывается метод propagate(evidence), который вернет true, если пропагация прошла успешно и false, если на каком-то этапе произошла ошибка. Для одного и того же ФЗ можно провести несколько выводов с разными свидетельствами, не устанавливая ФЗ повторно;
3. Если propagate(…) вернул false, то методы, предназначенные для получения результата, содержат все что угодно и их использовать нельзя. При этом ошибку, из-за которой пропагация не прошла, можно получить с помощью метода getError() и проанализировать. В частности, существует набор предопределенных ошибок, расположенных в пакете algbn.exceptions;
4. Случай, когда вероятность свидетельства оказалась равна нулю, рассматривается как ошибка, getError() возвращает ImprobableException;
5. После пропагации свидетельства его вероятность можно получить с помощью метода getEvidenceProbability() или методов getEvidenceProbabilityLB() и getEvidenceProbabilityUB() , в зависимости от типа оценки вероятности;
6. ФЗ с апостериорными оценками вероятностей можно получить с помощью метода getResult();
7. Если одна машина вывода использует при пропагации другую, то ошибка, возникающая в используемой машине вывода, прекращает работу алгоритма и возвращается методом getError().

DeterministicScalarConjunctsLocalPropagator пропагирует детерминированное свидетельство, поступившее во ФЗ с точечными оценками. Этот класс использует машину априорного вывода ScalarConjunctsLocalInferrer для проверки непротиворечивости исходного ФЗ и переозначивания атомов. Переозначивание необходимо для того, чтобы поступившее свидетельство можно было бы считать положительно означенным и применить указанные формулы.

DeterministicIntervalConjunctsLocalPropagator пропагирует детерминированное свидетельство, поступившее во ФЗ с интервальными оценками. Этот класс использует машину априорного вывода IntervalConjunctsLocalInferrer для проверки непротиворечивости ФЗ и переозначивания атомов во ФЗ.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

StochasticScalarConjunctsLocalPropagator пропагирует стохастическое свидетельство, поступившее во ФЗ с точечными оценками вероятностей. Этот класс использует машину априорного вывода ScalarConjunctsLocalInferrer для проверки непротиворечивости ФЗ, проверки непротиворечивости свидетельства и преобразования свидетельства из ФЗ над идеалом конъюнктов во ФЗ над набором квантов; DeterministicScalarConjunctsLocalPropagator для проведения апостериорного вывода детерминированных свидетельств, класс ScalarSummator для подсчета математического ожидания.

StochasticIntervalConjunctsLocalPropagator пропагирует стохастическое свидетельство, поступившее во ФЗ с интервальными оценками вероятностей. Этот класс использует машину априорного вывода IntervalConjunctsLocalInferrer для проверки непротиворечивости ФЗ; ScalarConjunctsLocalInferrer для проверки непротиворечивости свидетельства и преобразования свидетельства из ФЗ над идеалом конъюнктов во ФЗ над набором квантов; DeterministicIntervalConjunctsLocalPropagator для проведения апостериорного вывода детерминированных свидетельств, класс IntervalSummator для подсчета математического ожидания.

ImpreciseScalarConjunctsLocalPropagator пропагирует неточное свидетельство, поступившее во ФЗ с точечными оценками вероятностей. Этот класс использует ScalarConjunctsLocalInferrer для проверки непротиворечивости ФЗ, IntervalConjunctsLocalInferrer для проверки непротиворечивости свидетельства и преобразования его из ФЗ над идеалом конъюнктов во ФЗ над набором квантов; DeterministicScalarConjunctsLocalPropagator для проведения апостериорного вывода детерминированных свидетельств.

ImpreciseIntervalConjunctsLocalPropagator пропагирует неточное свидетельство, поступившее во ФЗ с интервальными оценками вероятностей. Этот класс использует машину априорного IntervalConjunctsLocalInferrer для проверки непротиворечивости ФЗ, проверки непротиворечивости свидетельства и преобразования свидетельства из ФЗ над идеалом конъюнктов во ФЗ над набором квантов; DeterministicIntervalConjunctsLocalPropagator для проведения апостериорного вывода детерминированных свидетельств.

Перечисленные выше алгоритмы имеют общие детали реализации, которые были выделены в 3 абстрактных класса. Получившаяся иерархия классов изображена на рис. 4.5.

Обычно иерархия выстраивается по одинаковым входным и выходным данным. Но в данном случае так сделать не удается, потому

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

что пропагация неточного свидетельства в скалярном ФЗ приводит к изменению типа оценок вероятностей в этом фрагменте на интервальные. Кроме того, между пропагацией неточного свидетельства в скалярном и интервальном ФЗ значительно больше общего, чем, например, между пропагацией стохастического и неточного свидетельств в интервальном ФЗ, несмотря на разные входные данные в первом случае. Такая особенность не позволяет четко выделить общие методы в интерфейсах. По сути, число требуемых интерфейсов совпадает с числом алгоритмов вывода. Похожая ситуация возникает и с глобальным апостериорным выводом.

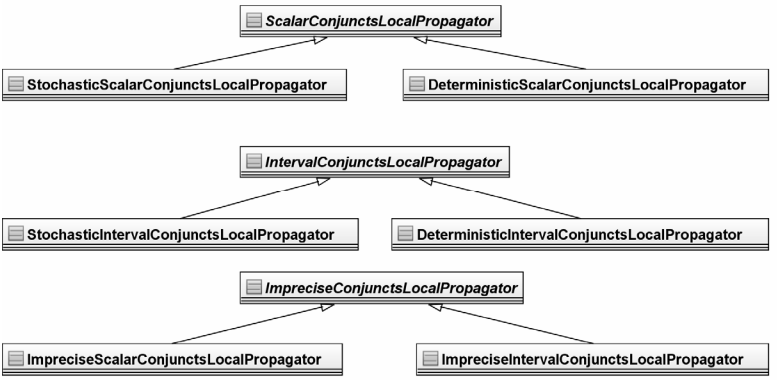

Рис. 4.5. Иерархия классов локального апостериорного вывода.

4.6. Непротиворечивость АБС. В теории АБС рассматривается четыре вида непротиворечивости алгебраической байесовской сети (локальная, интернальная, экстернальная и глобальная). Каждый последующий вид включает в себя предыдущий, но имеет значительно большую алгоритмическую сложность. Алгоритм реализации проверки непротиворечивости, как и в локальном случае, зависит от типа оценок вероятностей ФЗ.

Проверку глобальной непротиворечивости базы фрагментов знаний со скалярными оценками выполняет класс ScalarConjunctsGlobalInferrer. Из-за выбранной структуры данных для хранения базы знаний локальная непротиворечивость совпала с экстернальной. Из-за скалярного типа оценок вероятностей экстернальная непротиворечивость совпала с интернальной. Глобальная непротиворечивость проверяется тем же способом, что и глобальная непротиворечивость базы знаний с интервальными оценками вероятностей, по-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

этому она выделена в отдельный класс. Таким образом, в этом классе реализована только локальная непротиворечивость АБС, которая сводится к проверке локальной непротиворечивости каждого ФЗ с помощью класса ScalarConjunctsLocalInferrer.

Проверку глобальной непротиворечивости базы фрагментов знаний с интервальными оценками выполняет класс IntervalConjunctsGlobalInferrer.

Интернально непротиворечивая сеть получается с помощью решения серии ЗЛП по максимизации и минимизации оценки вероятности каждого конъюнкта в сети. При этом для того, чтобы проверить, противоречива ли АБС, достаточно решить ровно одну ЗЛП. Ограничениями в ЗЛП выступают условия локальной непротиворечивости каждого из ФЗ в совокупности, с учетом того, что на пересечениях ФЗ должны быть одни и те же оценки вероятностей.

Глобальная непротиворечивость реализована в отдельном классе GlobalConjunctsInferrer. Алгоритм для скалярных и интервальных оценок в сети совпадает. Необходимо построить объемлющий ФЗ, куда попадут все фрагменты сети. Этот ФЗ будет интервальным, так как всем конъюнктам, не содержащимся в сети, присваивается максимально широкая оценка вероятности $[0,1]$. Далее проводится проверка локальной непротиворечивости этого объемлющего ФЗ с помощью класса IntervalConjunctsLocalInferrer. Если ФЗ оказался непротиворечив, то исходная сеть глобально непротиворечива. Кроме того, для интервальной сети получены уточненные оценки. Это довольно простой для понимания и реализации, но самый вычислительно сложный алгоритм проверки непротиворечивости, так как порядок объемлющего ФЗ равен числу атомов в алфавите.

4.7. Глобальный апостериорный вывод. Глобальный апостериорный вывод — это распространение влияния свидетельства, поступившего в один фрагмент знаний, на другие. Для этого необходимо использовать вторичную структуру сети, задающую связи между фрагментами. В библиотеке реализован алгоритм, корректно работающий для ациклических АБС (у которых вторичная структура представлена в виде дерева смежности или цепи). Алгоритм распространения един для всех видов свидетельств, поэтому он выделен в общий абстрактный класс ConjunctsGlobalPropagator. Этот класс использует FirstDepthIterator для обхода вторичной структуры в глубину, каждый раз при посещении еще непосещенной вершины вызывая метод propagateNode(evidence, index). Метод propagateNode, опираясь на тип пере-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

данного ему свидетельства, вызывает один из трех абстрактных методов:

- propagateDeterministic(evidence, Long patternIndex),
- propagateStochastic(evidence, Long patternIndex),
- propagateImprecise(evidence, Long patternIndex).

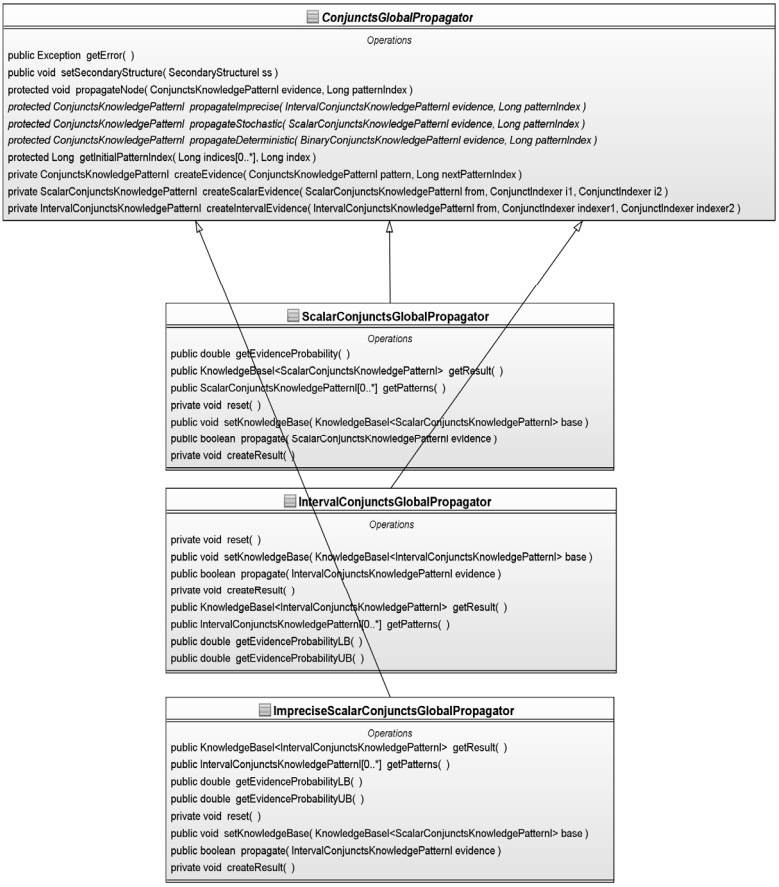

Рис. 4.6. Классы глобального апостериорного вывода.

В этом же классе реализован метод createEvidence(pattern, nextIndex), который создает виртуальное свидетельство подходящего типа.

Потомки класса ConjunctsGlobalPropagator реализуют перечисленные абстрактные методы, основываясь на локальном апостериор-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ном выводе соответствующего типа (для скалярной базы знаний — скалярные алгоритмы, для интервальной — интервальные). Всего было выделено 3 конкретных класса, изображенные на рис. 3.10. Класс ImpreciseScalarConjunctsGlobalPropagator пришлось отделить от класса ScalarConjunctsGlobalPropagator из-за того, что в случае неточного свидетельства возвращается сеть с интервальными, а не скалярными оценками.

Результатом глобального апостериорного вывода является новая АБС, с новыми оценками фрагментов знаний. Исходная сеть, согласно принципам работы алгоритмов вывода, не модифицируется.

5. Примеры использования библиотеки. В библиотеке реализовано 9 видов фрагментов знаний — {Interval | Scalar | Binary}{Conjuncts | Disjuncts | Quants}KnowledgePattern:

При создании ФЗ необходимо обязательно указать его глобальный индекс, который изменить впоследствии нельзя. Создать экземпляр класса, соответствующего ФЗ, можно следующим образом (на примере скалярного ФЗ):

```
ScalarConjunctsKnowledgePatternI pattern =
new ScalarConjunctsKnowledgePattern(5L);
```

То же самое, но в более понятной форме:

```
ScalarConjunctsKnowledgePatternI pattern =
new ScalarConjunctsKnowledgePattern (
KnowledgePatternBasis.globalIndex("101"));
```

Задать оценки вероятностей можно по отдельности, используя локальный индекс массива вероятностей (от 0 до pattern.getNumberOfElements() – 1):

```
((ScalarWriterI)pattern).setProbability(2, 0.38);
```

Также можно задать целиком весь массив оценок вероятностей (при этом значения массива будут скопированы):

```
double[] p = new double[] {1.0, 0.5, 0.4, 0.3};
((ScalarWriterI)pattern).setProbability(p);
```

Прочитать оценки вероятности можно аналогичным образом, только без необходимости приводить к соответствующему интерфейсу чтения:

```
pattern.getProbability(2); // вернет оценку вероятности по локальному индексу.
pattern.getProbability(); // вернет копию массива оценок.
```

В случае интервального ФЗ используются методы getProbabilityLB() и getProbabilityUB().

Бинарный ФЗ отличается от других тем, что не хранит оценки вероятностей в массиве. Он реализован таким образом, что не содержать противоречивые оценки вероятности. Этот факт используется при вы-

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

воде. Чтобы записать оценки, необходимо задать локальный индекс положительно означенного конъюнкта (по умолчанию при создании все атомы положительно означенные):

```
BinaryQuantsKnowledgePatternI quant = new BinaryQuantsKnowledgePattern(4L);
((BinaryWriterI)quant).setQuant(2);
```

Типичный способ использования бинарного ФЗ — перебор всех возможных означиваний:

```
for (int i = 0; i < quant.getNumberOfElements(); i++) {
        ((BinaryWriterI)quant).setQuant(i);
        ...
}
```

Стоит отметить, что при использовании массивов все оценки вероятностей каждый раз копируются, так как ФЗ не должен разрешать доступ к своему внутреннему состоянию. Исключением является задание массива вероятностей при создании ФЗ. В этом случае массив не копируется, а сохраняется ссылка на него. Пример:

```
double[] pLB = new double[] {1.0, 0.5};
double[] pUB = new double[] {1.0, 0.6};
IntervalConjunctsKnowledgePatternI pattern =
new IntervalConjunctsKnowledgePattern(
KnowledgePatternBasis.globalIndex("11L"), pLB, pUB);
```

Это можно использовать для полного копирования ФЗ:

```
IntervalConjunctsKnowledgePatternI p2  =
new IntervalConjunctsKnowledgePattern(p1.getGlobalIndex(),
p1.getProbabilityLB(), p1.getProbabilityUB());
```

Оценки вероятностей нельзя сравнивать обычным способом, так как в результате вычислений возникает ошибка вычислений. Для этих целей в классе *KnowledgePatternBasis* определены 2 статических метода и задана погрешность:

```
public static boolean isEqual(double p1, double p2);
public static int compare(double p1, double p2);
```

5.1. Использование локальных машин вывода. В библиотеке реализовано 2 класса машин локального априорного вывода (для интервальных и скалярных ФЗ):

- IntervalConjunctsLocalInferrer,
- ScalarConjunctsLocalInferrer.

Они реализуют различные алгоритмы, но работают по одинаковому принципу. Все машины вывода являются «многоразовыми». Создание (на примере интервального случая):

```
IntervalLocalInferrerI<IntervalConjunctsKnowledgePatternI>  inferrer =
new IntervalConjunctsLocalInferrer();
```

Запуск алгоритма проверки непротиворечивости ФЗ:

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

inferrer.processKP(pattern);

Если метод вернул false, то это значит, что входные данные оказались не верны (то есть, в метод мог быть передан null). Если true, то результаты проверки можно получить с помощью методов isConsistent(), isNarrowed(), getResult(). Если ФЗ оказался противоречив, метод isConsistent() возвращает false, isNarrowed() возвращает false, getResult() возвращает null. Дальнейшая работа с таким ФЗ не имеет смысла. Если ФЗ непротиворечив, то в скалярном случае, так как оценки не могут быть уточнены, метод getResult() возвращает копию исходного ФЗ, в интервальном случае — копию ФЗ с уточненными оценками.

После успешного вызова processKP(pattern) можно:

- при помощи метода getResult(localPositiveIndex) получить ФЗ с переозначенными атомами;
- при помощи метода getQuant() преобразовать исходный ФЗ над идеалом конъюнктов во ФЗ над набором квантов.

В обоих случаях, если исходный ФЗ оказался противоречив, возвращается null. Это надо учитывать при работе и перед вызовом методов проверять результат с помощью метода isConsistent().

Помимо проверки непротиворечивости и уточнения оценок вероятностей inferrers используются для проведения априорного вывода. Для этого необходимо сделать следующее:

```
pattern.addLinearForm("p(y or z)", 0, 1, 1, -1); // добавить в ФЗ линейную форму
inferrer.processKP(pattern); // провести априорный вывод
if (inferrer.isConsistent()) { // получить результат
LinearFormI f = inferrer.getResult().getLinearForm("p(y or z)");
System.out.println(f.getLB() + "<= p(y or z) <= " + f.getUB);
}
```

В результате изначально неопределенные границы линейной формы будут уточнены.

Коэффициенты линейной формы могут быть заданы сразу при создании (как сделано выше), тогда необходимо указывать и все нулевые коэффициенты. Либо после, тогда достаточно указать только ненулевые:

```
LinearFormI f  = pattern.addLinearForm("p(y or z)");
f.setWeight(1, 1);
f.setWeight(2, 1);
f.setWeight(3, 1);
```

В библиотеке реализовано 6 классов машин локального апостериорного вывода {Deterministic | Stochastic | Imprecise}{Scalar | Interval}LocalPropagator.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Каждый из них инкапсулирует алгоритм вывода для соответствующих типов оценок вероятностей фрагмента знаний и свидетельства, построенных над идеалом конъюнктов.

Рассмотрим типичный пример для скалярного ФЗ и неточного свидетельства:

```
ScalarConjunctsKnowledgePatternI pattern = new ScalarConjunctsKnowledgePat-
tern(      Long.parseLong("1101", 2), new double[] {1, 0.8, 0.5, 0.4, 0.2, 0.15, 0.1,
0.05});
IntervalConjunctsKnowledgePatternI evidence = new IntervalConjunctsKnowl-
edgePattern(      Long.parseLong("100", 2), new double[] {1.0, 0.3}, new dou-
ble[] {1.0, 0.7});
ImpreciseScalarConjunctsLocalPropagatorI propagator =
        new ImpreciseScalarConjunctsLocalPropagator();
propagator.setPattern(pattern);
if (propagator.propagate(evidence)) {
        double pLB = propagator.getEvidenceProbabilityLB();
        double pUB = propagator.getEvidenceProbabilityUB();
        IntervalConjunctsKnowledgePatternI result = propagator.getResult();
        System.out.println(String.format("pLB=%1.3f, pUB=%1.3f", pLB, pUB));
        System.out.println(result);
} else {
        System.out.println(propagator.getError());
}
```

Если на каком-то этапе пропагации свидетельства возникает ошибка, то метод вернет false, выходные данные неопределены и их значения обрабатывать нельзя, а ошибку можно проанализировать с помощью метода getError(). Пропагация свидетельства, имеющего нулевую вероятность, рассматривается как ошибка ImprobableException. Противоречивость ФЗ и свидетельства — как ошибки InconsistentPatternException и InconsistentEvidenceException соответственно.

5.2. Модели базы фрагментов знаний. В библиотеке реализовано 2 класса для хранения базы знаний (для скалярных и интервальных ФЗ):

- ScalarConjunctsKnowledgeBase,
- IntervalConjunctsKnowledgeBase.

Для создания базы знаний необходимо сначала создать алфавит и множество ФЗ над этим алфавитом. В библиотеке реализовано 2 вида алфавитов:

```
AlphabetI alphabet = new PrefixAlphabet(3, "x");// основанный на  префиксе
AlphabetI alphabet2 = new ArrayAlphabet("x", "y", "z");// основанный на пере-
числении
```


Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Для именования конъюнктов с помощью алфавита можно использовать класс:

```
ConjunctNamer namer = ConjunctNamer.get(alphabet);
String name = namer.getName(globalIndex, localIndex);
```

Множество ФЗ удобнее всего хранить следующим образом:

```
Map<Long, IntervalConjunctsKnowledgePatternI> patterns =
        new HashMap<Long, IntervalConjunctsKnowledgePatternI>();
// patterns.put(…)
```

Перед тем, как создать базу знаний, необходимо проверить два условия.

Множество ФЗ должно образовывать связный граф. Проверить это можно следующим образом:

```
MaxJoinGraph graph = MaxJoinGraph.create(patterns.keySet());
boolean connected = graph.getSecondaryStructure().isConnected();
```

Оценки вероятностей на пересечении должны совпадать. Для этого ФЗ можно пересечь с помощью класса IntervalConjunctsIntersector (или ScalarConjunctsIntersector):

```
IntervalConjunctsIntersector i = new IntervalConjunctsIntersector();
i.intersect(new HashSet(patterns.values()), true);
if (i.isConsistent()) {          … }
if (i.isNarrowed()) { … i.getResult() }
```

Intersector имеет два режима работы: при пересечении модифицировать исходные ФЗ или же создавать копии. Пользователь может выбрать, какой режим подходит лучше.

Создание базы знаний:

```
KnowledgeBaseI<ScalarConjunctsKnowledgePatternI> sbase =
new ScalarConjunctsKnowledgeBase(alphabet, new HashSet(patterns.values()));
KnowledgeBaseI<IntervalConjunctsKnowledgePatternI> ibase =
new IntervalConjunctsKnowledgeBase(alphabet,new HashSet(patterns.values()));
```

При создании опять проверяются вышеназванные условия и если они не выполнены, создается исключение IllegalArgumentException. После создания база знаний является неизменяемой. С помощью методов чтения по глобальному индексу можно получить копию ФЗ или оценку вероятности отдельного конъюнкта:

```
ScalarConjunctsKnowledgePatternI pattern =
sbase.getKnowledgePattern(globalIndex);
BaseElement conjunct = sbase. getElement(globalIndex);
```

Вторичная структура сети отделена от первичной. В библиотеке используется два класса для представления вторичной структуры:

- SSDescription (сокращенная),
- SecondaryStructure (расширенная).

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Первая хранит массив вершин и массив ребер, ссылающихся на индексы вершин. Вторая дополнительно хранит список смежности вершин, также ссылающийся на индексы. Создание:

```
Long[] vertices = …;          IntEdge[] edges = …;
SSDescription ss = new SSDescription(vertices, edges);
SecondaryStructureI secondaryStructure = ss.getSecondaryStructure();
```

Для корректной работы с SSDescription необходимо выполнять следующие условия:

- массив вершин vertices отсортирован по возрастанию индексов,
- массив вершин и массив ребер не меняются после своего создания,
- ребра ссылаются на существующие вершины.

При нарушении одного из этих условий поведение становится непредсказуемым. Предполагается, что SSDescription создается с помощью алгоритма, перебирающего вторичные структуры. В библиотеке реализован такой алгоритм, инкапсулированный в классе JoinGraphEnumerator. Получить все возможные вторичные структуры данных по первичной структуре можно, например, следующим образом:

```
KnowledgeBaseI<ScalarConjunctsKnowledgePatternI> base = …;
Set<Long> vertices = base.getPatternIndices();
JoinGraphEnumeratorI enumerator = new JoinGraphEnumerator(vertices);
enumerator.generateSecondaryStructures();
List<SSDescription> result = enumerator.getSecondaryStructures();
```

Также в программе реализован класс MaxJoinGraph, создающий максимальный граф смежности по набору вершин. Этот класс может быть полезен, если надо проверить связность сети, получить набор всех возможных ребер, получить только максимальный граф смежности:

```
SecondaryStructure gmax = MaxJoinGraph.create(vertices); // создает граф
// для того, чтобы максимальный граф смежности был графом смежности,
// достаточно проверить его на связность
boolean isConnected = gmax.isConnected(); // проверяет на связность
SecondaryStructureI.Type type = gmax.getType(); // возвращает тип графа
boolean isJoinGraph = (type != SecondaryStructureI.Type.UNDEFINED);
SSDescription ss = gmax.getSSDescription(); // возвращает сокращенную структуру (минимальную по числу ребер)
```

Метод getType() возвращает тип графа смежности (JOIN_GRAPH, JOIN_TREE, JOIN_CHAIN), основываясь на предположении, что ребра задают именно граф смежности, но не проверяя это предположение. Если же ребра образуют связный граф, не удовлетворяющий условию

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

смежности, то этот метод будет работать неверно. Для максимального графа смежности он работает всегда верно в силу его особенностей.

Основная операция, которая проводится с вторичной структурой, — обход графа. В библиотеке реализован итератор, перебирающий вершины графа с помощью обхода в глубину. Используя его, эта операция становится очевидной:

```
SecondaryStructure structure = …;
Long startFrom = …;
FirstDepthIterator it = new FirstDepthIterator(structure, startFrom);
while (it.hasNext) {
        Long v = it.next();
        doWhatYouNeed();
}
```

Кроме этого, в классе SecondaryStructure доступны следующие стандартные для списка смежности операции:

- *degreeOf(vertexIndex),*
- *List<Integer> neighbours(vertexIndex),*
- *List<Long> vertices(),*
- *List<IntEdge> edges().*

5.3. Обработка базы фрагментов знаний. В библиотеке реализовано два класса для проверки непротиворечивости сети:

- ScalarConjunctsGlobalInferrer (скалярной),
- IntervalConjunctsGlobalInferrer (и интервальной).

Они позволяют проверять различные степени непротиворечивости сети. Проверка основывается на некоторых особенностях реализации глобальных структур данных, что не является самым удачным решением, так как необходимо, чтобы проверка опиралась лишь на разработанный интерфейс структур данных. В частности, на основании того, что оценки вероятностей конъюнктов в сети хранятся в единственном экземпляре и всегда совпадают между собой, локальная и экстернальная непротиворечивость совпадает. Приведем пример.

Сначала необходимо создать машину вывода.

```
KnowledgeBaseI<ScalarConjunctsKnowledgePatternI> kb = ...;
ScalarGlobalInferrerI<ScalarConjunctsKnowledgePatternI> inferrer =
                new ScalarConjunctsGlobalInferrer();
```

Аналогично локальной машине априорного вывода, метод processKB(base, type) запускает алгоритм проверки непротиворечивости. Он возвращает false, если исходные данные не могут быть обработаны алгоритмом. Если true, то результат проверки можно получить с помощью методов isConsistent(), isNarrowed(), getResult().

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

inferrer.processKB(kb, type); // запускает алгоритм проверки непротиворечивости

Параметр type может принимать одно из 4-х значений ConsistencyType: LOCAL, EXTERNAL, INTERNAL, GLOBAL. При этом в скалярном случае алгоритм проверки не различает типы LOCAL, EXTERNAL, INTERNAL из-за особенностей структуры данных. В интервальном случае не различаются типы LOCAL и EXTERNAL.

В интервальном случае для проверки экстернальной непротиворечивости необходимо дополнительно задать вторичную структуру сети. Алгоритм может корректно работать только для ациклических сетей. Для циклических сетей алгоритма пока не разработано.

```
KnowledgeBaseI<IntervalConjunctsKnowledgePatternI> base = …;
SecondaryStructureI ss = …;
inferrer.setSecondaryStructure(ss);
if (inferrer.processKB(base, ConsistencyType.EXTENRNAL)) {
boolean isConsistent = inferrer.isConsistent();
boolean isNarrowed = inferrer.isNarrowed();
KnowledgeBaseI< IntervalConjunctsKnowledgePatternI> result =
                    inferrer.getResult();
```

Если метод processKB(base, type) вернул true, то можно обрабатывать результат. Метод isConsistent() возвращает true, если сеть непротиворечива, false, если противоречива. Метод isNarrowed() возвращает true, если оценки вероятности в сети были уточнены. Для скалярной сети результатом будет всегда false.

Если сеть непротиворечива, метод getResult() для скалярной сети вернет копию исходной сети, для интервальной — копию сети с уточненными оценками. Если сеть противоречива, метод вернет null.

В библиотеке реализовано три класса для проведения глобального апостериорного вывода:

- ScalarConjunctsGlobalPropagator (пропагирует детерминированные и стохастические свидетельства в скалярной сети);
- ImpreciseScalarConjunctsGlobalPropagator (пропагирует неточные свидетельства в скалярной сети и возвращает интервальную сеть),
- IntervalConjunctsGlobalPropagator (пропагирует все 3 вида свидетельств в интервальной сети).

Для пропагации свидетельства сначала необходимо создать машину вывода.

```
IntervalConjuctsGlobalPropagatorI propagator = new IntervalConjunctsGlobal-
Propagator();
```


Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Затем задать исходные данные — первичную и вторичную структуры сети.

```
KnowledgeBaseI<IntervalConjunctsKnowledgePatternI> base = …;
SecondaryStructureI ss = …;
propagator. setKnowledgeBase(base);
propagator.setSecondaryStructure(SecondaryStructureI ss);
```

После этого необходимо запустить алгоритм пропагации.

```
ScalarConjunctsKnowledgePatternI evidence = …;
boolean noErrors = propagator.propagate(evidence);
```

В зависимости от типа свидетельства, необходимый алгоритм пропагации будет выбран автоматически.

Аналогично машине локального апостериорного вывода, в случае возникновения при пропагации какой-либо ошибки, метод propagate(evidence) вернет false, а метод getError() вернет ошибку. Результат можно использовать только тогда, когда метод propagate(evidence) вернул true:

```
if (noErrors) {
        double probabilityLB = propagator.getEvidenceProbabilityLB();
        double probabilityUB = propagator.getEvidenceProbabilityUB();
        KnowledgeBaseI<IntervalConjunctsKnowledgePatternI> result =
                propagator.getResult();
        Set< IntervalConjunctsKnowledgePatternI > patterns = propaga-
tor.getPatterns();
} else {
        System.out.println(propagator.getError());
}
```

6. Выводы. Наиболее важными аспектами представления алгебраических байесовских сетей в реляционных базах данных являются решения, касающиеся формализации фрагмента знаний, свидетельства, первичной и вторичной структуры АБС. Хотя при проведении операций логико-вероятностного вывода различия между типами объектов (в частности, по оценкам — бинарным, скалярным и интервальным) существенно, в целях хранения достаточно определить «обобщенную» структуру, которая позволяет учесть все описанные варианты. По этой структуре для вычислений уже можно восстановить специализированные более узко.

Перечень операций, которые необходимо выполнить, задаются сессиями, по содержанию которых понятно, какой вид вывода с какими объектами должен быть произведен. Результаты логико-вероятностного вывода, таким образом, ассоциируются именно с сессией.

Задача представления АБС в java-коде распадается на три подзадачи по представлению *локальной структуры* — глобальный индекс

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

главного конъюнкта и вектор вероятностей всех конъюнктов во фрагменте знаний, *первичной глобальной структуры* — перечень глобальных индексов главных конъюнктов, а также перечень пар глобальных индексов и оценок вероятностей конъюнктов, *вторичной глобальной структуры* — перечень пар глобальных индексов, задающих ребра в графе смежности.

Рассмотренные структуры для представления алгебраических байесовских сетей в базах данных и java-коде хорошо согласованы. На их основе был разработан комплекс java-программ, предназначенный для проведения вычислительных экспериментов. Указанный комплекс поддерживает ключевые алгоритмы логико-вероятностного вывода: локальные проверку и поддержание непротиворечивости, априорный и апостериорный вывод. Следует особо отметить две возможности: апостериорный вывод реализован для всех трех видов свидетельств, рассматриваемых в теории АБС — детерминированного, стохастического и неточного; в синтезе согласованных оценок истинности допускается обработка расширенных фрагментов знаний.

Литература

1. *Абрамян А.К.* Комплекс программ для локального логико-вероятностного вывода в алгебраических байесовских сетях: разработка прототипа ядра (Java- и ILOG-технологии). Дипломная записка / Под рук. А. Л. Тулупьева. СПб., 2007. 262 с. (СПбГУ. Математико-механический факультет. Каф. информатики.)
2. *Городецкий В.И.* Алгебраические байесовские сети — новая парадигма экспертных систем // Юбилейный сборник трудов институтов Отделения информатики, вычислительной техники и автоматизации РАН. Т. 2. М.: РАН, 1993. С. 120–141.
3. *Казакова О.С., Сироткин А.В., Тулупьева Т.В., Тулупьев А.Л., Пащенко А.Е.* Выделение главных компонент для ядра системы «Личность--Деятельность--Эффективность» на основе байесовских сетей // Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте. Научно-практическая конференция студентов, аспирантов, молодых ученых и специалистов (Коломна, 26–27 мая 2009 г.). Научные доклады. В 2-х т. Т. 2. М.: Физматлит, 2009. С. 123–131.
4. *Павельчук А.В.* Реинжиниринг библиотеки логико-вероятностного вывода и визуализация АБС. Дипломная записка / Под рук. А. Л. Тулупьева. СПб., 2009. 119 с. (СПбГУ. Математико-механический факультет. Каф. информатики.)
5. *Тотмянина С.А.* Реинжиниринг структуры базы данных для представления алгебраических байесовских сетей. Дипломная записка / Под рук. А. Л. Тулупьева. СПб., 2009. 77 с. (СПбГУ. Математико-механический факультет. Каф. информатики.)
6. *Тулупьев А.Л.* Алгебраические байесовские сети: теоретические основы и непротиворечивость. СПб.: СПИИРАН, 1995. 76 с.
7. *Тулупьев А.Л.* Алгебраические байесовские сети: логико-вероятностный подход к моделированию баз знаний с неопределенностью. СПб.: СПИИРАН, 2000. 282 с.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

8. *Тулупьев А.Л.* Структурированные сети фрагментов знаний с неопределенностью // Интегрированные модели и мягкие вычисления в искусственном интеллекте: Сборник трудов международного научно-практического семинара. Коломна, 17–18 мая 2001. М.: Наука, Физматлит, 2001. С. 178–183.
9. *Тулупьев А.Л.* Метод построения и исследования баз фрагментов знаний с неопределенностью // Труды СПИИРАН. 2002. Вып. 1, т. 1. СПб.: Наука, 2002. С. 258–271.
10. *Тулупьев А.Л.* Алгебраические байесовские сети: логико-вероятностная модель баз фрагментов знаний с неопределенностью // Всероссийская научная конференция по нечетким системам и мягким вычислениям НСМВ-2006 (20–22 сентября 2006 г.). Труды. Тверь, 2006. С 31–47.
11. *Тулупьев А.Л.* Дерево смежности с идеалами конъюнктов как ациклическая алгебраическая байесовская сеть // Труды СПИИРАН. Вып. 3, т. 1. СПб.: Наука, 2006. С. 198–227.
12. *Тулупьев А.Л.* Ациклические алгебраические байесовские сети: логико-вероятностный вывод // Нечеткие системы и мягкие вычисления: Научный журнал Российской ассоциации нечетких систем и мягких вычислений. 2006. Том 1, № 1. С. 57–93.
13. *Тулупьев А.Л.* Алгебраические байесовские сети: локальный логико-вероятностный вывод: Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 80 с. (Сер. Элементы мягких вычислений).
14. *Тулупьев А.Л.* Алгебраические байесовские сети: глобальный логико-вероятностный вывод в деревьях смежности: Учеб. пособие. СПб.: СПбГУ; ООО Издательство «Анатолия», 2007. 40 с. (Сер. Элементы мягких вычислений).
15. *Тулупьев А.Л., Абрамян А.К.* Логико-вероятностный вывод в направленном БСД-цикле // Труды СПИИРАН. 2007. Вып. 4. СПб.: Наука, 2007. С. 87–118.
16. *Тулупьев А.Л.* Байесовские сети: логико-вероятностный вывод в циклах. СПб.: Изд-во С.-Петерб. ун-та, 2008. 140 с.
17. *Тулупьев А.Л.* Вероятностная логика и вероятностные графические модели в базах фрагментов знаний с неопределенностью // Интегрированные модели, мягкие вычисления, вероятностные системы и комплексы программ в искусственном интеллекте. Научно-практическая конференция студентов, аспирантов, молодых ученых и специалистов (Коломна, 26–27 мая 2009 г.). Научные доклады. В 2-х т. Т. 1. М.: Физматлит, 2009. С. 26--46.
18. *Тулупьев А.Л.* Автоматическое обучение фрагментов знаний в алгебраических байесовских сетях // Интегрированные модели и мягкие вычисления в искусственном интеллекте. V-я Международная научно-практическая конференция. Сборник научных трудов. В 2-х т. Т. 1. С. 163–176.
19. *Тулупьев А.Л.* Обработка дополнительной нечисловой информации в локальном обучении алгебраических байесовских сетей по выборкам с пропусками // Международная конференция по мягким вычислениям и измерениям. Сборник докладов. 2009. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2009.
20. *Тулупьев А.Л., Горшков А.С., Сироткин А.В., Тулупьева Т.В., Пащенко А.Е., Чжен Жичан.* Моделирование систем «Личность–Деятельность–Эффективность» на основе байесовских сетей: постановка проблемы // Труды СПИИРАН. 2008. Вып. 6. СПб.: Наука, 2008. С. 198–202.
21. *Тулупьев А.Л., Николенко С.И., Сироткин А.В.* Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. 607 с.
22. *Тулупьев А.Л., Сироткин А.В.* Алгебраические байесовские сети: принцип декомпозиции и логико-вероятностный вывод в условиях неопределенности // Информационно-измерительные и управляющие системы. 2008. № 10, т. 6. С. 85–87.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

23. *Тулупьев А.Л., Сироткин А.В.* Байесовские и марковские сети: логико-вероятностный вывод в базах фрагментов знаний с неопределенностью // Научн. конф. Информационные технологии и системы, Геленджик, сентябрь 29–октябрь 03, 2008 г.: Труды конференции. М.: ИППИ РАН, 2008. С.~440–456.
24. *Тулупьев А.Л., Сироткин А.В., Николенко С.И.* Байесовские сети доверия: логико-вероятностный вывод в ациклических направленных графах. СПб.: Изд-во С.-Петерб. ун-та, 2009. 400 с.
25. *Тулупьев А.Л., Столяров Д.М., Ментюков М.В.* Представление локальной и глобальной структуры алгебраической байесовской сети в Java-приложениях // Труды СПИИРАН. 2007. Вып. 5. СПб.: Наука, 2007. С. 71–99.
26. *Bool G.* An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities. Cambridge: Macmillan / London: Walton & Maberly, 1854. (Reprinted in 1951, Dover Publications, New York.)
27. *Cowell R.G., Dawid A.Ph., Lauritzen S.L., Spiegelhalter D.J.* Probabilistic Networks and Expert Systems. Berlin: Springer, 2003. 321 p.
28. *Fagin R., Halpern J.Y., Megiddo N.* A Logic for Reasoning about Probabilities. Report RJ 6190 (60900) 4/12/88. pp. 1–41.
29. *Fagin R., Halpern J.Y.* Uncertainty, Belief, and Probability–2 // Proc. of the IEEE Symposium on Logic and Computer Science. 1991. Vol. 7. P. 160–173.
30. *Halpern J.Y.* Reasoning about uncertainty. Cambridge, MS: The MIT Press, 2003. 483 p.
31. *Jensen F.V.* Bayesian Networks and Decision Graphs. NY.: Springer-Verlag, 2001. 268 p.
32. *Kindermann R., Snell J.L.* Markov Random Fields and Their Applications. Providence, RI: Amer. Math. Soc., 1980. 142 p.
33. *Kyburg H.E., Teng C.M.* Uncertain inference. Cambridge: Cambridge University Press, 2001. 298 p.
34. *Korb K.B., Nicholson A.E.* Bayesian Artificial Intelligence. NY.: Chapman and Hall/CRC, 2004. 364 p.
35. *Neapolitan R.E.* Learning Bayesian Networks. Pearson Prentice Hall, 2003. 674 p.
36. *Nilsson N.J.* Probabilistic Logic // Artificial Intelligence. 1986. Vol. 47. Amsterdam: Elsevier Science Publishers B.V., 1986. P. 71–87.
37. Nilsson N.J. Probabilistic Logic Revisited // Artificial Intelligence. 1993. Vol. 59. Amsterdam: Elsevier Science Publishers B.V., 1993. P. 31–36.
38. Parsons S. Qualitative methods for reasoning under uncertainty. Cambridge, MS: The MIT Press, 2001. 506 p.
39. Perl J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. NY etc.: Morgan Kaufmann Publ., 1994. P. 552.

Тулупьев Александр Львович — к.ф.-м.н., доцент; ведущий научный сотрудник научно-исследовательской группы междисциплинарных проблем информатики СПИИРАН, доцент кафедры информатики математико-механического факультета С.-Петербургского государственного университета (СПбГУ). Область научных интересов: представление и обработка данных и знаний с неопределенностью, применение методов математики и информатики в социокультурных исследованиях, применение методов биостатистики и математического моделирования в эпидемиологии, технология разработки программных комплексов с СУБД. Число научных публикаций — 150. ALT@iias.spb.su, www.tulupyev.spb.ru; СПИИРАН, 14-я линия В.О., д. 39, г. Санкт-Петербург, 199178, РФ; р.т. +7(812)328-3337, факс +7(812)328-4450.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ, проект № 09-01-00861-а.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Рекомендовано ЛПИ СПИИРАН, зав. лаб. Юсупов Р.М., член-корреспондент РАН.
Статья поступила в редакцию 25.06.2009.

Труды СПИИРАН. 2009. Вып. 8. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2009. Issue 8. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru