

ЛОГИКО-ВЕРОЯТНОСТНЫЙ ВЫВОД В НАПРАВЛЕННОМ БСД-ЦИКЛЕ

А. Л. ТУЛУПЬЕВ[♦], А. К. АБРАМЯН

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178

<alt@iias.spb.su>

УДК 004.8

Тулупьев А. Л., Абрамян А. К. **Логико-вероятностный вывод в направленном БСД-цикле** // Труды СПИИРАН. Вып. 4. — СПб.: Наука, 2007.

Аннотация. *Обработка направленных циклов остается открытым вопросом в теории байесовских сетей доверия (БСД). Нами предлагается алгоритм первичной пропации, который основан на традиционном для БСД принципе — передаче сообщений между узлами. Получаемые результаты вычислений совпадают с результатами алгоритма, предложенного ранее, но основанного на другом подходе. Кроме того, в результате указанных вычислений формируется семантически эквивалентный образ направленного БСД-цикла. При переходе к этому образу становится возможным использование ряда алгоритмов логико-вероятностного вывода (ЛВВ): поддержание непротиворечивости, априорный вывод и апостериорный вывод.* — Библ. 14 назв.

UDC 004.8

Tulupuyev A. L., Abramyan A. K. **Probabilistic Logic Inference for a Directed Cycle in Bayesian Belief Networks** // SPIIRAS Proceedings. Issue 4. — SPb.: Nauka, 2007.

Abstract. *Processing of a directed cycle remains an open question of the Bayesian belief network theory. We propose a message-passing algorithm for initial propagation in the cycle. The results of this algorithms are fully compatible with the results of the algorithms we proposed earlier that is based on another principle. Besides, as result of this computations is formed a semantically equal image of the directed cycle in a Bayesian belief network. For this image we could perform probabilistic logic inference: reconciliation, a priori inference and a posteriori inference.* — Bibl. 14 items.

1. Введение

Запрет на наличие направленного цикла в байесовских сетях доверия (БСД) является жестким ограничением на их структуру [14]. Направленные БСД-циклы изменяют вероятностную семантику байесовской сети доверия: вместо одного распределения вероятностей она задает их семейство, причем последнее может быть пустым при наличии противоречия в исходных данных. Известные алгоритмы первичной пропации, вычисляющие маргинальные вероятности атомарных пропозициональных переменных, стоящих в узлах БСД, оказываются неработоспособным в случае семейств распределений (и в случае противоречий). Также становится невозможным использование алгоритмов ЛВВ, разработанных для ациклических БСД.

Ранее [5–9, 11, 13] был предложен подход к алгоритмизации первичной пропации в БСД-цикле как с атомарными логическими, так и с многозначными переменными в узлах. Разработанный алгоритм расчета маргинальных вероятностей атомов и конъюнкций атомов из смежных узлов сводился к решению яв-

[♦] Исследования, результаты которых представлены в настоящей работе, были частично поддержаны грантом Фонда содействия отечественным учёным за 2006 и 2007 гг., а также госконтрактом № 02.442.11.7289 от 28.06.2006 на выполнение НИР «Направленные циклы в байесовских сетях доверия: вероятностная семантика и алгоритмы логико-вероятностного вывода для программных комплексов с байесовской интеллектуальной компонентой».

но заданного линейного матричного уравнения. *Целью* настоящей работы будет построение алгоритма первичной пропагации в направленном БСД-цикле на иной основе — с помощью передачи сообщений между узлами. Алгоритмы на основе передачи сообщений более традиционны для расчета вероятностных оценок в БСД. Несмотря на указанное отличие, новый алгоритм будет приводить к тем же самым оценкам маргинальных вероятностей. Также будет показано, как на основе семантически эквивалентного образа, получающегося в результате первичной пропагации в БСД-цикле, можно осуществить виды ЛВВ, недоступные для БСД с направленным циклом.

Мы придерживаемся системы терминов и обозначений, развитой в [10]. Кроме того, в «Трудах СПИИРАН» неоднократно [7, 11] обсуждались вопросы введения вероятности на пропозициональных формулах. Чтобы сократить объем материала, мы не станем повторно затрагивать указанную тематику.

2. Направленный БСД-цикл

Действуя в рамках логико-вероятностного подхода, мы будем полагать, что направленный цикл (примеры направленных БСД-циклов — на рис. 1) в байесовской сети доверия является, в первую очередь, направленным графом. Этот граф состоит из узлов, в которых расположено по одному литералу, и ребер, соединяющих узлы так, что все они входят в один направленный цикл. Кроме того, каждому узлу приписаны тензоры условных вероятностей. Речь идет о вероятностях литерала принять то или иное истинностное означивание в зависимости от означивания литерала в узле-родителе.

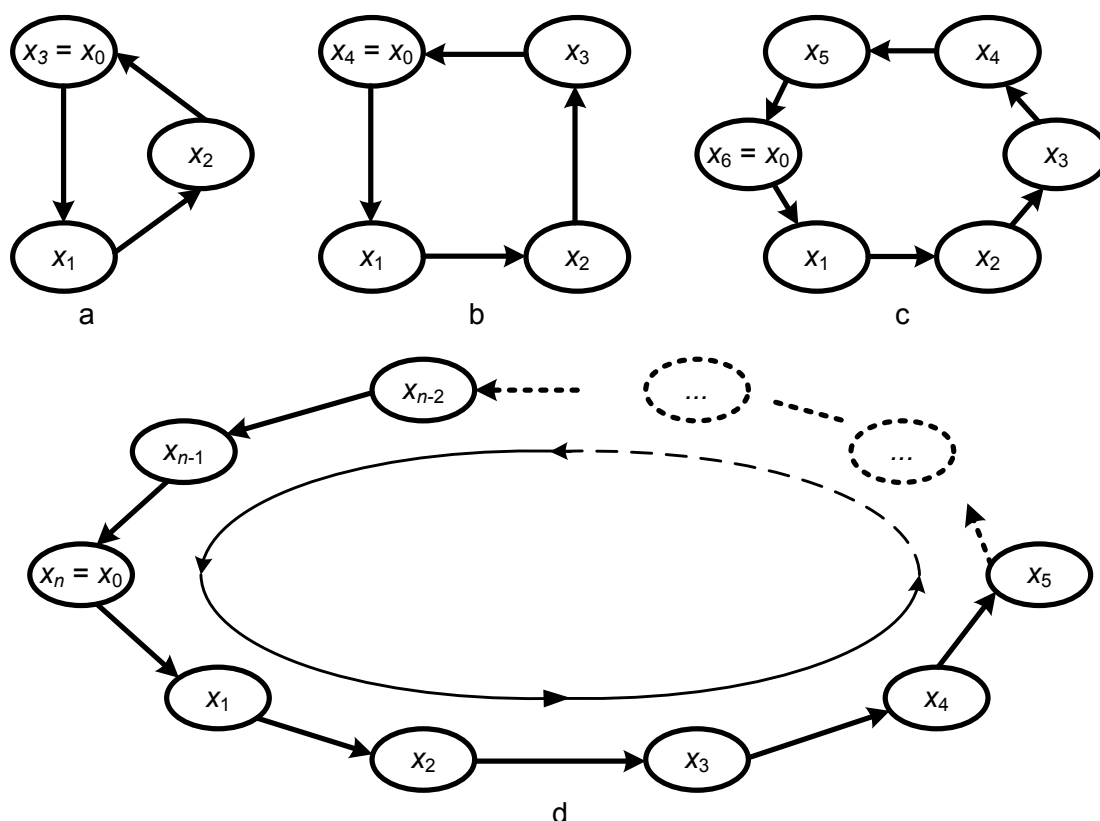


Рис. 1. Направленные циклы в байесовских сетях доверия. а, b, с — циклы над тремя, четырьмя и шестью узлами соответственно. d — общий вид цикла. В узлах цикла указаны атомы (атомарные пропозициональные формулы).

Литералы, стоящие в узлах, обозначим \tilde{x}_i (\tilde{x}_i может принимать значения x_i и \bar{x}_i), где $i \in \{1, \dots, n\}$, а n — число узлов в цикле. На узлы цикла будем ссылаться либо с помощью указания соответствующего литерала \tilde{x}_i , либо (особенно на рисунках) с помощью указания на связанный с ним атом x_i .

Тензор условных вероятностей, стоящий в узле, имеет вид $p(\tilde{x}_i | \tilde{x}_j)$, где индекс j предшествует i . Как правило, $j = i - 1$, но для $i = 1$ предшествующим индексом будет $j = n$. Для удобства будем считать $j = n$ и $j = 0$ одним и тем же значением индекса, что влечет, например, совпадение $x_0 = x_n$. Более того, поскольку цикл замыкается, удобно рассматривать индексы (номера) узлов, которые больше n ; будем считать, что они ссылаются на тот узел, «исходный» номер которого совпадает с остатком от целого деления индекса на n .

Тензор условных вероятностей $p(\tilde{x}_i | \tilde{x}_j)$ состоит из четырех величин: $p(x_i | x_j)$ и $p(\bar{x}_i | x_j)$, $p(x_i | \bar{x}_j)$ и $p(\bar{x}_i | \bar{x}_j)$. Набор указанных четырех значений однозначно определяется всего двумя величинами $p(x_i | x_j)$ и $p(x_i | \bar{x}_j)$, поскольку имеют место соотношения вероятностей: $p(\bar{x}_i | x_j) = 1 - p(x_i | x_j)$ и $p(\bar{x}_i | \bar{x}_j) = 1 - p(x_i | \bar{x}_j)$.

Для использования в дальнейшем определим величины r_i как разность условных вероятностей $r_i = p(x_i | \bar{x}_j) - p(x_i | x_j)$.

3. Вероятностные соотношения между узлами цикла

По формуле полной вероятности мы можем рассчитать вероятность истинности $p(x_i)$ атома x_i , если знаем вероятность истинности $p(x_{i-1})$ атома x_{i-1} из предыдущего узла: $p(x_i) = p(x_i | x_{i-1})p(x_{i-1}) + p(x_i | \bar{x}_{i-1})p(\bar{x}_{i-1})$, или что то же:

$$\begin{aligned} p(x_i) &= p(x_i | x_{i-1})p(x_{i-1}) + p(x_i | \bar{x}_{i-1})(1 - p(x_{i-1})), \\ p(x_i) &= p(x_i | x_{i-1})p(x_{i-1}) + p(x_i | \bar{x}_{i-1}) - p(x_{i-1})p(x_i | \bar{x}_{i-1}), \\ p(x_i) &= p(x_i | \bar{x}_{i-1}) + (p(x_i | x_{i-1}) - p(x_i | \bar{x}_{i-1}))p(x_{i-1}), \\ p(x_i) &= p(x_i | \bar{x}_{i-1}) - r_i p(x_{i-1}). \end{aligned}$$

Определив величину $p(x_i)$, можно аналогичным образом вычислить $p(x_{i+1})$, а затем $p(x_{i+2})$, $p(x_{i+3})$ и так далее, пока цикл не замкнется на $p(x_{i-1})$. Зная величину $p(x_{i-1})$, по определению условной вероятности можно рассчитать маргинальные вероятности конъюнкции атомов из смежных узлов:

$$p(x_{i-1}x_i) = p(x_i | x_{i-1})p(x_{i-1}).$$

При известных вероятностях $p(x_{i-1})$, $p(x_i)$ и $p(x_{i-1}x_i)$ по формуле включений–исключений можно рассчитать вероятность любого означивания конъюнкции литералов из смежных узлов $p(\tilde{x}_{i-1}\tilde{x}_i)$:

$$\begin{aligned} p(x_{i-1}\bar{x}_i) &= p(x_{i-1}) - p(x_{i-1}x_i), \\ p(\bar{x}_{i-1}x_i) &= p(x_i) - p(x_{i-1}x_i), \\ p(\bar{x}_{i-1}\bar{x}_i) &= 1 - p(x_{i-1}) - p(x_i) + p(x_{i-1}x_i). \end{aligned}$$

Таким образом, если бы стала известна маргинальная вероятность хотя бы одного атома из узла цикла, мы смогли бы восстановить все остальные маргинальные вероятности вида $p(x_{i-1})$, $p(x_{i-1}x_i)$ и вообще $p(\tilde{x}_{i-1}\tilde{x}_i)$.

Обозначим неизвестную величину $p(x_0)$ как π . Тогда согласно формулам, приведенным выше, через неизвестную π можно выразить последовательно маргинальные вероятности атомов $p(x_i) = p_i(\pi)$. Величину неизвестной π можно будет определить, решив уравнение

$$\pi = p_n(\pi) \quad (1)$$

Узнав численное значение π , можно будет в свою очередь вычислить численные значения интересующих нас маргинальных вероятностей вида $p(x_{i-1})$, $p(x_{i-1}x_i)$ и вообще $p(\tilde{x}_{i-1}\tilde{x}_i)$. Если же уравнение относительно π не будет иметь решения, то тогда исходные данные придется признать противоречивыми.

Утверждение 1. $p_i(\pi) = a_i\pi + b_i$, где $a_0 = 1$, $b_0 = 0$, а a_i и $b_i \in \mathbb{R}$.

Доказательство. Воспользуемся методом математической индукции. При $i = 0$ база индукции очевидна. Выполним индукционный переход. Пусть справедлива формула $p_{i-1}(\pi) = a_{i-1}\pi + b_{i-1}$, рассчитаем величину $p_i(\pi)$:

$$\begin{aligned} p_i(\pi) &= p(x_i | x_{i-1}) - r_i p_{i-1}(\pi), \\ p_i(\pi) &= p(x_i | x_{i-1}) - r_i(a_{i-1}\pi + b_{i-1}), \\ p_i(\pi) &= p(x_i | x_{i-1}) - r_i a_{i-1}\pi - r_i b_{i-1}. \end{aligned}$$

Индукционный переход завершен; $p_i(\pi)$ линейно выражается через π , причем

$$\begin{aligned} a_i &= -r_i a_{i-1}, \\ b_i &= p(x_i | x_{i-1}) - r_i b_{i-1}. \end{aligned}$$

Замечание 1. Методом математической индукции можно доказать, что

$$\forall (i \geq 1) \quad a_i = (-1)^i r_1 r_2 \dots r_i.$$

Уравнение (1) теперь можно преобразовать следующим образом:

$$\begin{aligned} a_n \pi + b_n &= \pi, \quad \pi - a_n \pi = b_n, \\ (1 - a_n) \pi &= b_n, \quad (1 - (-1)^n r_1 r_2 \dots r_n) \pi = b_n. \end{aligned}$$

Если $(1 - a_n) \neq 0$, уравнение будет иметь единственное решение

$$\pi = \frac{b_n}{1 - a_n} = \frac{b_n}{1 - (-1)^n r_1 r_2 \dots r_n},$$

которое можно распространить по циклу и получить численные значения искомых вероятностей. Если коэффициент и свободный член оба равны нулю, то π может принимать любое значение; но, учитывая, что π — это вероятность, возможные значения могут лежать только в промежутке $[0; 1]$.

Заметим, что более глубокий анализ показал бы [7, 10, 11], что все остальные вероятности атомов тоже стали бы неопределенными, но все они должны были бы совпадать с любым выбранным допустимым точечным значением π или $1 - \pi$. Наконец, если коэффициент перед π равен нулю, а свободный член — нет, то тогда исходные данные содержат противоречие. В таком случае маргинальные вероятности не могут быть вычислены. Более глубокий анализ показал бы, что при $(1 - a_n) = 0$ атомы в узлах тождественны некоторому атому x или его отрицанию \bar{x} [7, 10, 11]

4. Структура сообщений и алгоритмы их обработки

Полученные результаты позволяют предложить новый алгоритм первичной пропагации. Задачей этого алгоритма является расчет маргинальных вероятностей $p(x_i)$, где $i \in \{1, \dots, n\}$, но также дополнительно могут быть вычислены $p(x_{i-1}x_i)$. В случае обнаружения противоречия в исходных данных или особой ситуации с произвольным допустимым значением π алгоритм должен об этом сообщить (изменить значение соответствующей переменной).

Для удобства рассмотрим параллельно два описания алгоритма: без матриц и с ними. Пусть сообщение состоит из двух полей и имеет векторный вид:

$$\mathbf{M}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{M}_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}.$$

Сообщение \mathbf{M}_{i-1} поступает в узел x_i , в начале работы алгоритма генерируется сообщение \mathbf{M}_0 и считается, что оно отправлено из узла $x_0 = x_n$. В уравнениях преобразования сообщения-вектора участвует матрица \mathbf{R}_i и вектор \mathbf{v}_i :

$$\mathbf{R}_i = \begin{bmatrix} -r_i & 0 \\ 0 & r_i \end{bmatrix}, \quad \mathbf{v}_i = \begin{pmatrix} 0 \\ p(x_i | x_{i-1}) \end{pmatrix}.$$

Поступившее сообщение \mathbf{M}_{i-1} индуцирует новое, исходящее сообщение \mathbf{M}_i с помощью преобразования $\mathbf{M}_i = \mathbf{R}_i \mathbf{M}_{i-1} + \mathbf{v}_i$, что в скалярном виде будет выглядеть как

$$\begin{aligned} a_i &= -r_i a_{i-1}, \\ b_i &= r_i b_{i-1} + p(x_i | x_{i-1}). \end{aligned}$$

Сообщение \mathbf{M}_i пересылается дальше в узел-ребенок (рис. 2).

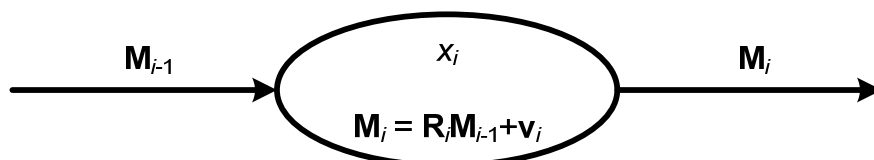


Рис. 2. Обработка сообщения в узле БСД-цикла.

Процесс останавливается, когда сообщение поступает в узел, с которого началась рассылка сообщений. Заметим, что процесс рассылки обязательно остановится, поскольку число узлов в цикле конечно. В расчетах при обработке сообщений внутри узлов особых ситуаций, связанных с некорректными арифметическими операциями, возникнуть не должно, поскольку используются лишь операции сложения, вычитания и умножения, а операция деления — нет. Распространение сообщений \mathbf{M}_i назовем *первой фазой* алгоритма.

После того как сообщение поступило в узел $x_0 = x_n$, из которого ушло начальное сообщение, формируется уравнение (1), производится его анализ, и в случае существования единственного решения определяется величина π . Она отправляется как вероятность $p(x_0)$ (или, что то же, как величина $p(x_n)$) из узла $x_0 = x_n$ в следующий узел.

Когда в узел x_i поступает сообщение с вероятностью $p(x_{i-1})$, вычисляются вероятности $p(x_{i-1}x_i) = p(x_i | x_{i-1})p(x_{i-1})$ и $p(x_i) = p(x_i | \bar{x}_{i-1}) - r_i p(x_{i-1})$. Если узел

еще не отправлял сообщение с вероятностью своего атома, рассчитанная вероятность $p(x_i)$ передается как сообщение узлу-сыну. Поскольку цикл конечный, то алгоритм распространения сообщений завершит свою работу. Распространение вероятностей $p(x_i)$ назовем *второй фазой* алгоритма.

Когда решения уравнения (1) не существует или решений бесконечно много, изменяется переменная, свидетельствующая о статусе цикла. Вероятность как сообщение распространять не требуется, но в некоторых реализациях разумно распространить неопределенное значение NULL, чтобы сбросить содержимое соответствующих переменных, содержащих сведения о маргинальных вероятностях.

5. Результаты двух алгоритмов пропагации

Предлагавшийся ранее алгоритм пропагации [5–9, 10, 11, 13] основывается на решении системы линейных уравнений вида

$$\mathbf{A}\mathbf{p} = \mathbf{q}, \text{ где} \quad (2)$$

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & r_1 \\ r_2 & 1 & 0 & \dots & 0 & 0 \\ 0 & r_3 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & r_n & 1 \end{pmatrix}, \mathbf{p} = \begin{pmatrix} p(x_1) \\ p(x_2) \\ p(x_3) \\ \vdots \\ p(x_n) \end{pmatrix}, \mathbf{q} = \begin{pmatrix} p(x_1 | \bar{x}_n) \\ p(x_2 | \bar{x}_1) \\ p(x_3 | \bar{x}_2) \\ \vdots \\ p(x_n | \bar{x}_{n-1}) \end{pmatrix}.$$

Система (2) получается при замене обозначений в записанной в исходных обозначениях системе уравнений, вытекающей из аксиоматики вероятностной логики и определения условной вероятности:

$$\begin{cases} p(x_1) = p(x_1 | x_n)p(x_n) + p(x_1 | \bar{x}_n)(1 - p(x_n)), \\ p(x_2) = p(x_2 | x_1)p(x_1) + p(x_2 | \bar{x}_1)(1 - p(x_1)), \\ \vdots \\ p(x_n) = p(x_n | x_{n-1})p(x_{n-1}) + p(x_n | \bar{x}_{n-1})(1 - p(x_{n-1})). \end{cases}$$

В решении системы уравнений (2) участвовала также матрица $\tilde{\mathbf{A}}$, получающаяся из \mathbf{A} заменой последнего столбца на столбец свободных членов \mathbf{q} :

$$\tilde{\mathbf{A}} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & p(x_1 | \bar{x}_n) \\ r_2 & 1 & 0 & \dots & 0 & p(x_2 | \bar{x}_1) \\ 0 & r_3 & 1 & \dots & 0 & p(x_3 | \bar{x}_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & r_n & p(x_n | \bar{x}_{n-1}) \end{pmatrix}.$$

Для анализа системы исследовался определитель $|\mathbf{A}|$, при разложении которого по первой строке получается $|\mathbf{A}| = 1 - (-1)^n r_1 r_2 \dots r_n$.

Эта величина совпадает с коэффициентом перед π в линейном уравнении (1), поэтому особые случаи решения системы (2) (при $|\mathbf{A}| = 0$) и уравнения (1) случаются при одних и тех же исходных данных.

Если $|\mathbf{A}| \neq 0$, тогда для нахождения $p(x_n)$ используется формула Крамера:

$$\rho(x_n) = \frac{|\tilde{\mathbf{A}}|}{|\mathbf{A}|},$$

при этом определитель $|\tilde{\mathbf{A}}|$ может быть вычислен рекурсивно:

$$\begin{aligned}\Delta_0 &= 0, \\ \Delta_1 &= -r_1\Delta_0 + \rho(x_1 | \bar{x}_n), \\ \Delta_j &= -r_j\Delta_{j-1} + \rho(x_j | \bar{x}_{j-1}), \\ |\tilde{\mathbf{A}}| &= \Delta_n.\end{aligned}$$

Заметим, что Δ_j вычисляется по тем же самым формулам, что и свободный член b_j , причем $b_0 = \Delta_0 = 0$. Таким образом, удалось показать, что величины $\rho(x_n)$, вычисленные с помощью двух разных подходов, будут совпадать. Расчеты величин $\rho(x_{j-1})$, $\rho(x_{j-1}x_j)$ ведутся по одинаковым формулам в обоих подходах, что позволяет сделать заключение о совпадении результатов первичной пропагации, выполненной двумя рассмотренными алгоритмами.

6. Непротиворечивость направленного БСД-цикла

Рассмотрим сначала мотивирующий пример, который позволит продемонстрировать, что цикл может задавать семейство распределений вероятностей.

Пример 1. Цикл из трех вершин [5, 10, 11].

Пусть заданы условные вероятности в узлах направленного БСД-цикла из трех вершин x_1 , x_2 и x_3 . Нашей задачей является описание семейства распределений вероятностей $\rho(\tilde{x}_1\tilde{x}_2\tilde{x}_3)$ над квантами $\tilde{x}_1\tilde{x}_2\tilde{x}_3$ (множество которых образует в вероятностной логике пространство элементарных событий).

В результате любого из двух описанных выше алгоритмов пропагации мы сможем определить величины $\rho(x_1)$, $\rho(x_2)$, $\rho(x_3)$, $\rho(x_1x_2)$, $\rho(x_1x_3)$ и $\rho(x_2x_3)$. Это вероятности почти всех элементов идеала конъюнктов, образованного над атомами x_1 , x_2 и x_3 . Неопределенной остается единственная величина $\rho(x_1x_2x_3)$. Заметим, что если мы знаем распределение вероятностей над идеалом конъюнктов, мы можем также оценить вероятности квантов.

Примем $\rho(x_1x_2x_3)$ за неизвестную величину и запишем систему ограничений, вытекающую из аксиоматики вероятностной логики

$$\begin{cases} \rho(x_1x_2x_3) \geq 0, \\ \rho(x_1x_2) - \rho(x_1x_2x_3) \geq 0, \\ \rho(x_1x_3) - \rho(x_1x_2x_3) \geq 0, \\ \rho(x_2x_3) - \rho(x_1x_2x_3) \geq 0, \\ \rho(x_1) - \rho(x_1x_2) - \rho(x_1x_3) + \rho(x_1x_2x_3) \geq 0, \\ \rho(x_2) - \rho(x_1x_2) - \rho(x_2x_3) + \rho(x_1x_2x_3) \geq 0, \\ \rho(x_3) - \rho(x_1x_3) - \rho(x_2x_3) + \rho(x_1x_2x_3) \geq 0, \\ 1 - \rho(x_1) - \rho(x_2) - \rho(x_3) + \rho(x_1x_2) + \rho(x_2x_3) + \rho(x_1x_3) - \rho(x_1x_2x_3) \geq 0. \end{cases}$$

Это — тривиальный одномерный случай задачи линейного программирования (ЗЛП). В явном виде относительно единственной переменной ее решение запишется следующим образом:

$$\max \left\{ \begin{array}{l} 0, \\ p(x_1x_2) + p(x_1x_3) - p(x_1), \\ p(x_1x_2) + p(x_2x_3) - p(x_2), \\ p(x_1x_3) + p(x_2x_3) - p(x_3) \end{array} \right\} \leq p(x_1x_2x_3) \leq \min \left\{ \begin{array}{l} p(x_1x_2), \\ p(x_1x_3), \\ p(x_2x_3), \\ 1 - p(x_1) - p(x_2) - p(x_3) + \\ + p(x_1x_2) + p(x_1x_3) + p(x_2x_3) \end{array} \right\}.$$

Это неравенство может оказаться неразрешимым, что будет говорить о противоречии в исходных данных (или, иными словами, о несовместности/противоречивости исходных оценок вероятности).

Если же неравенство разрешимо, то можно будет выразить через любое допустимое значение $p(x_1x_2x_3)$ из промежутка, заданного неравенством, все остальные вероятности, формирующие совокупное распределение:

$$p(\bar{x}_1x_2x_3) = p(x_2x_3) - p(x_1x_2x_3),$$

$$p(x_1\bar{x}_2x_3) = p(x_1x_3) - p(x_1x_2x_3),$$

$$p(x_1x_2\bar{x}_3) = p(x_1x_2) - p(x_1x_2x_3),$$

$$p(\bar{x}_1\bar{x}_2x_3) = p(x_3) - p(x_2x_3) - p(x_1x_3) + p(x_1x_2x_3),$$

$$p(\bar{x}_1x_2\bar{x}_3) = p(x_2) - p(x_1x_2) - p(x_2x_3) + p(x_1x_2x_3),$$

$$p(x_1\bar{x}_2\bar{x}_3) = p(x_1) - p(x_1x_2) - p(x_1x_3) + p(x_1x_2x_3),$$

$$p(\bar{x}_1\bar{x}_2\bar{x}_3) = 1 - p(x_1) - p(x_2) - p(x_3) + p(x_1x_2) + p(x_1x_3) + p(x_2x_3) - p(x_1x_2x_3).$$

Случай цикла из трех вершин вносит элементы неопределенности в получаемый ответ, однако эта неопределенность «одномерна» и ЗЛП становится тривиальной. Стоит отдельно отметить, что задача эта может, несмотря на свою тривиальность, не решиться [10]; а цикл может оказаться противоречив не только в описанной выше почти вырожденной ситуации.

Пример 1 дал достаточно полное представление того, как проверяется непротиворечивость направленного БСД-цикла. В общем случае для проверки непротиворечивости требуется выполнить следующие шаги:

1. Рассчитать величины a_n и b_n с помощью алгоритма, основывающегося на передаче сообщений.
2. Если уравнение $a_n\pi + b_n = \pi$ не имеет решений, то алгоритм прекращает работу с выводом, что исходные данные противоречивы.
3. Если уравнение $a_n\pi + b_n = \pi$ имеет бесконечно много решений, то алгоритм прекращает работу с выводом, что исходные данные непротиворечивы (совместны).
4. Если уравнение $a_n\pi + b_n = \pi$ имеет единственное решение, то следует выполнить вторую фазу алгоритма распространения сообщений, в которой рассчитываются величины вида $p(x_{i-1})$, $p(x_{i-1}x_i)$.
5. Сформировать идеал цепочек конъюнкций над атомами, вошедшими в цикл.
6. Элементом идеала вида x_{i-1} , $x_{i-1}x_i$ приписать рассчитанные оценки вероятностей $p(x_{i-1})$, $p(x_{i-1}x_i)$.
7. Остальным элементам идеала приписать оценки [0;1].

8. Поддерживать непротиворечивость сформированного фрагмента знаний (идеала конъюнктов с оценками их вероятностей) с помощью решения серии соответствующих ЗЛП [10].
9. Если ЗЛП не имеют решения, алгоритм прекращает работу с выводом о том, что исходные данные противоречивы, в противном случае алгоритм прекращает работу с выводом, что исходные данные непротиворечивы (совместны).

Вопрос о непротиворечивости направленного БСД-цикла был сведен к уже решенному вопросу о непротиворечивости фрагмента знаний (ФЗ) алгебраической байесовской сети (АБС). Более того, направленный БСД-цикл оказался погруженным в этот фрагмент знаний, а это открывает возможность не только проверить непротиворечивость исходных данных, но также выполнить апостериорный вывод на основе поступающих свидетельств или апостериорный вывод вероятности пропозициональных формул, построенных над теми же атомами, что и цикл [10].

7. Снижение сложности проверки непротиворечивости

Погружение направленного цикла целиком во фрагмент знаний потребует формирования задачи линейного программирования с 2^n ограничений и с $2^n - 1$ переменных. Экспоненциальный рост числа ограничений и переменных вычислительно нежелателен. Предлагается рассмотреть менее требовательный к объему данных алгоритм.

Заметим, что направленный БСД-цикл можно преобразовать в цикл фрагментов знаний, причем каждый фрагмент будет построен над двумя атомами. Он будет пересекаться с соседними фрагментами знаний по одному атому с каждым. На рис. 4 в правой колонке изображены циклы, а в центральной колонке — соответствующие циклические графы смежности [10] с идеалами конъюнктов.

Если в циклическом графе смежности содержится четное число ФЗ (строка 2 рис. 4), тогда можно цикл расположить в две параллельные цепи и объединить ФЗ, которые окажутся друг напротив друга. В случае нечетного числа ФЗ (строка 1 рис. 4) один раз придется объединить три фрагмента знаний. В результате преобразований получится цепь фрагментов знаний, каждый из которых построен над тремя или четырьмя атомами.

В силу того что цепь ФЗ ациклична, ее глобальная непротиворечивость следует из ее интернальной непротиворечивости, которая с вычислительной точки зрения существенно проще. Число фрагментов знаний в цикле: два ФЗ над тремя атомами, и $(n-4)/2$ ФЗ над четырьмя атомами в случае четного n или один ФЗ над тремя атомами и $(n-3)/2$ ФЗ над четырьмя атомами в случае нечетного n . Следовательно, число ограничений, которое предстоит сформировать: $2 \cdot 8 + ((n-4)/2) \cdot 16$ в первом случае и $1 \cdot 8 + ((n-3)/2) \cdot 16$ во втором случае. Число переменных ЗЛП будет $2 \cdot 7 + ((n-3)/2) \cdot 15 - (1 + (n-4)/2) \cdot 3$ в первом случае, $1 \cdot 7 + ((n-3)/2) \cdot 15 - ((n-3)/2) \cdot 3$. Таким образом, и число ограничений, и число переменных ЗЛП, которую предстоит решать для определения непротиворечивости направленного БСД-цикла в случае его погружения в цепь фрагментов знаний АБС, линейно зависят от числа узлов цикла, что существенно сокращает вычислительную сложность.

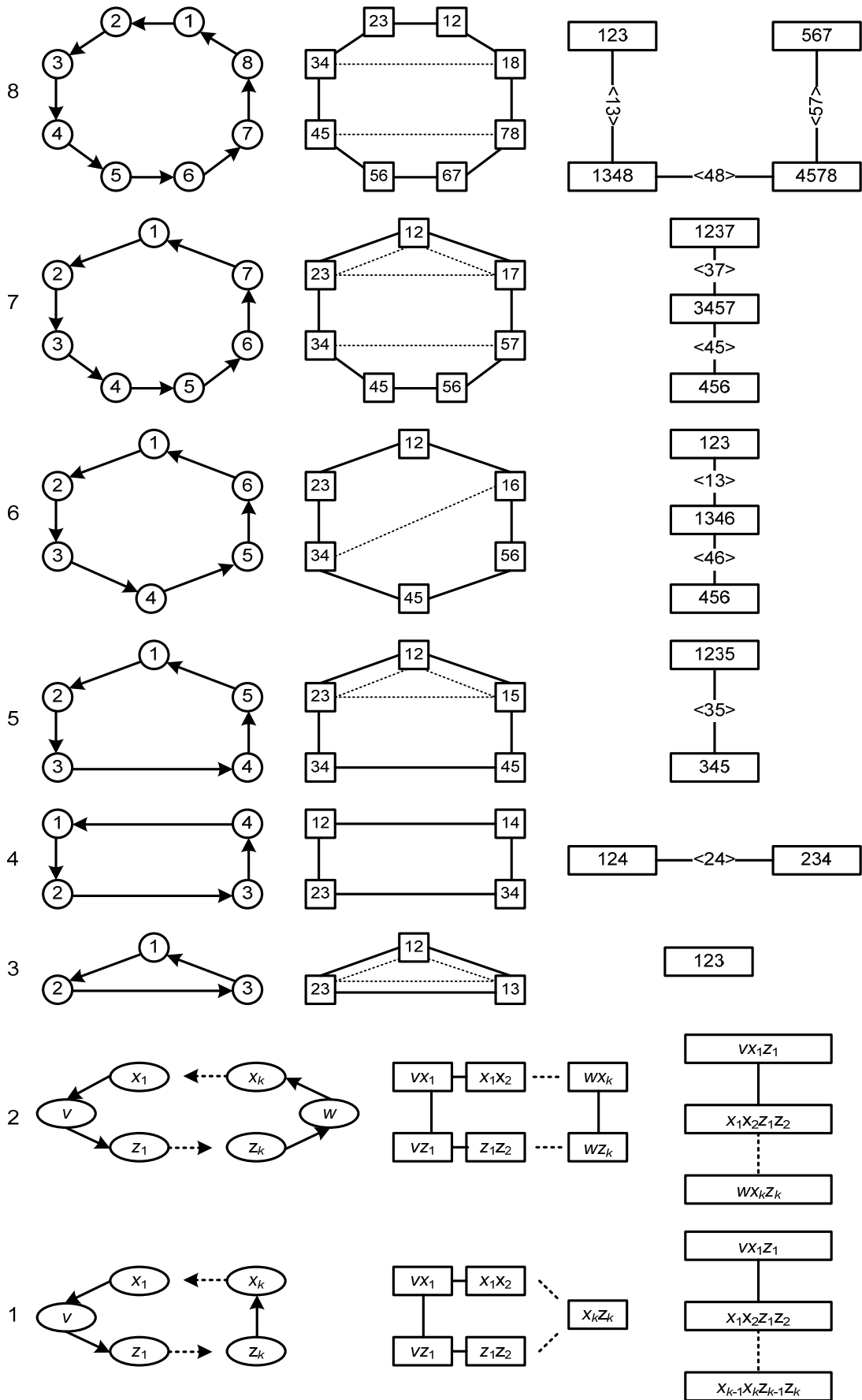


Рис. 3. Преобразование направленного БСД-цикла в цепь фрагментов знаний АБС.

8. Стохастическое моделирование означиваний узлов

В этом разделе будет изложен известный подход Гиббса к стохастическому моделированию означиваний квантов $\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$, образованных над атомами из узлов цикла. Задача состоит в том, чтобы генерировать означивания указанных квантов $\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$ так, что накапливающиеся частоты $\varphi_k(x_{i-1})$, $\varphi_k(x_{i-1}x_i)$ появления x_{i-1} и $x_{i-1}x_i$ за серию из k испытаний удовлетворяли бы условиям

$$\hat{p}_k(x_i | x_{i-1}) = \frac{\varphi_k(x_{i-1}x_i)}{\varphi_k(x_{i-1})} \xrightarrow{k \rightarrow \infty} p(x_i | x_{i-1}),$$

$$\hat{p}_k(x_{i-1}x_i) = \frac{\varphi_k(x_{i-1}x_i)}{k} \xrightarrow{k \rightarrow \infty} p(x_{i-1}x_i),$$

$$\hat{p}_k(x_i) = \frac{\varphi_k(x_i)}{k} \xrightarrow{k \rightarrow \infty} p(x_i).$$

Иными словами, требуется, чтобы эмпирические соотношения частот сходились бы к исходно заданным условным вероятностям и рассчитанным маргинальным вероятностям.



Рис.4. Байесовская сеть доверия, состоящая из цепи узлов.

Сначала рассмотрим не направленный цикл, а более простой случай — цепь, получающуюся при «размыкании цикла» (рис. 4). Исходные данные о вероятностях в цепи почти такие же, как и в направленном БСД-цикле, за одним существенным исключением. В цепи изначально задана маргинальная вероятность $p(x_0)$, что позволяет распространить ее сразу, не прибегая к первой фазе распространения сообщений, далее по цепи, чтобы вычислить $p(x_{i-1})$, $p(x_{i-1}x_i)$. Более того, сразу можно указать распределение вероятностей над квантами $\tilde{x}_0\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$, из которого можно получить вероятности, исходно заданные в цепи:

пи: $p(\tilde{x}_0\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n) = p(\tilde{x}_0) \prod_{i=1}^{i=n} p(\tilde{x}_i | \tilde{x}_{i-1})$. Существование такого распределения говорит нам о непротиворечивости исходных данных. Заметим, что предложенное распределение с указанными свойствами не единственное; однако если мы потребуем выполнения условной независимости узлов относительно разделяющих их узлов, получившееся распределение будет единственным.

Сгенерируем означивание элемента цепи \tilde{x}_0 : x_0 выпадает с вероятностью $p(x_0)$, а \bar{x}_0 — с вероятностью $p(\bar{x}_0) = 1 - p(x_0)$. Далее требуется сгенерировать одно из означиваний \tilde{x}_1 . Вероятность их выпадения зависит от того, какое означивание приобрел \tilde{x}_0 на предыдущем шаге. Если реализовалось x_0 , то тогда

$$p(x_1) = p(x_1 | x_0),$$

$$p(\bar{x}_1) = p(\bar{x}_1 | x_0);$$

если же реализовалось \bar{x}_0 , то тогда

$$p(x_1) = p(x_1 | \bar{x}_0),$$

$$p(\bar{x}_1) = p(\bar{x}_1 | \bar{x}_0).$$

Далее генерация означиваний происходит аналогично. Если известна реализация означивания предшественника, вероятность реализации означивания текущего узла выбирается по формуле:

$$p(\tilde{x}_i) = p(\tilde{x}_i | \tilde{x}_{i-1}).$$

После прохода из начала цепи в конец будет получена реализация означивания $\tilde{x}_0\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$. Затем процесс снова повторяется, начиная с генерации означивания \tilde{x}_0 .

Предложенный алгоритм можно адаптировать для генерации означиваний узлов в цикле $\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$.

По исходным данным цикла рассчитывается, скажем, вероятность $p(x_1)$. Начиная с узла x_1 и включая узел x_n , генерация означиваний производится так же, как это делалось в случае цепи узлов. После того как реализация означиваний $\tilde{x}_1\tilde{x}_2\dots\tilde{x}_n$ сформирована, требуется начать новый цикл генерации. Именно здесь возникают различия между алгоритмами генерации в цепи и в цикле. В случае цепи мы бы снова взяли для генерации означивания x_1 рассчитанную (или заданную) маргинальную вероятность $p(x_1)$. В случае цикла для начала следующего цикла генерации означиваний требуется взять

$$p(\tilde{x}_1) = p(\tilde{x}_1 | \tilde{x}_n).$$

Если исходные данные непротиворечивы, то вышеуказанные отношения частот будут сходиться к исходным данным.

9. Логико-вероятностный вывод

Как было показано в разделе 6, для проверки непротиворечивости оценок необходимо погрузить цикл БСД в ФЗ АБС, т.е. перейти к рассмотрению семантически эквивалентного образа, полученного при первичной пропагации БСД-цикла. В общем случае для осуществления ЛВВ, необходимо рассматривать семантически эквивалентный ФЗ АБС и в нем проводить ЛВВ, а именно проверку на непротиворечивость, априорный вывод и апостериорный вывод.

Проверка ФЗ на *непротиворечивость (согласованность)* оценок вероятностей — это проверка на соответствие их аксиомам вероятности и ограничениям, вытекающим из этих аксиом. *Поддержание непротиворечивости* ФЗ — это уточнение интервальных оценок в ФЗ так, чтобы их набор был непротиворечив. Поддержание непротиворечивости может закончиться либо получением непротиворечивого ФЗ, либо заключением, что исходный ФЗ противоречив.

Априорный вывод — вывод вероятности произвольной пропозициональной формулы, исходя из заданных оценок вероятностей конъюнктов.

Апостериорный вывод — вывод по свидетельству. Выделяются три вида свидетельства:

- *детерминированное свидетельство* — сведение о том, что какое-то утверждение, соответствующее атомарной пропозиции или цепи атомарных пропозиций, оказалось либо истинным, либо ложным;
- *стохастическое свидетельство*, характеризуемое распределением вероятностей своих элементов;
- *неопределенное свидетельство*, характеризуемое семейством распределений вероятностей своих элементов;

При поддержании непротиворечивости и априорном выводе в ФЗ с интервальными оценками необходимо решать задачи линейного программирования (ЗЛП); при апостериорном — задачи гиперболического программирования, но их удается свести к ЗЛП.

Удобной библиотекой для численного решения ЗЛП является ILOG CPLEX с набором программных интерфейсов ILOG Concert. Ее можно использовать, в том числе, в программах на языке Java. Реализация ЛВВ в ФЗ АБС, описанная ниже, выполнена на указанном языке и включает в себя поддержание непротиворечивости, апостериорный вывод с детерминированным и стохастическим свидетельством, а также небольшую часть априорного вывода — перевод оценок вероятностей конъюнктов в оценки вероятностей квантов и обратно.

Реализация погружения БСД в ФЗ АБС выполнена на языке Delphi. Обмен данными между этими двумя приложениями может осуществляться через базу данных.

10. Программная реализация погружения БСД-цикла в ФЗ

Пусть задан направленный БСД-цикл. Нашей задачей будет реализация алгоритма, основанного на распространении сообщений, вычисляющего величины a_i и b_i , $p(x_{i-1})$ и $p(x_{i-1}x_i)$, а также повторно рассчитанные для проверки результатов значения условных вероятностей $p(x_i | x_{i-1})$.

В качестве представления исходных и основных расчетных данных используем таблицы в реляционной базе данных MS Access 2003, а в качестве среды и языка разработки — Borland Delphi 7. Структура таблиц представлена на рис. 5, а описание используемых в алгоритме полей — в табл. 1–3. Пользовательский интерфейс получившегося приложения представлен на рис. 6.

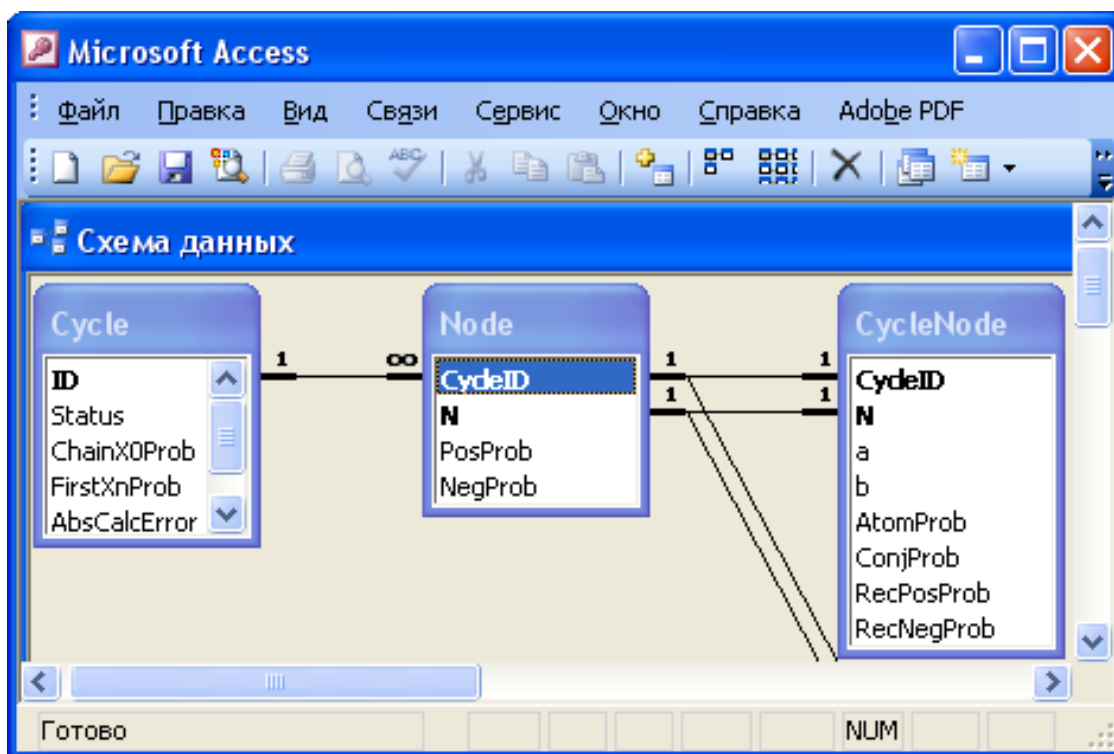


Рис. 5. Структура таблиц и связей в базе данных.

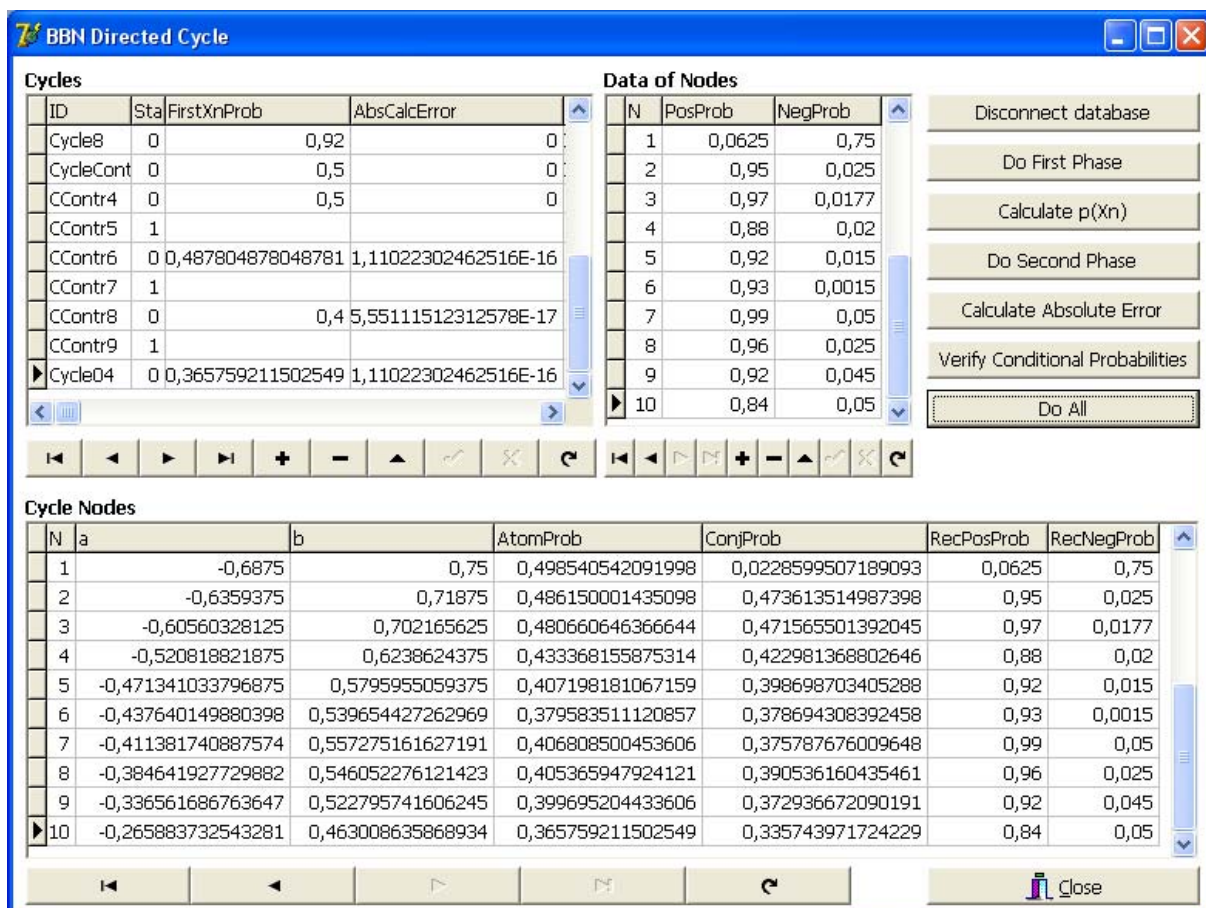


Рис. 6. Окно приложения для расчета маргинальных вероятностей атомов и конъюнкций двух соседних атомов на основе алгоритма передачи сообщений.

В программном коде (но не в базе данных) детализирующие таблицы `adotbNode` и `adotbCycleNode` связаны с главной таблицей `adotbCycle` по `CycleID-ID`, а таблицы `adotbCycleNode` и `adotbNode2` (клон `adotbNode`) 1:1-связаны по полям `CycleID-CycleID` и `N-N`. В базе данных 1:m-связь между таблицами `Cycle-Node` идет по `ID-CycleID`, а 1:1-связь `Node-CycleNode` идет по `CycleID-CycleID` и `N-N`.

Таблица 1
Структура таблицы базы данных `Cycle` (в коде программы `adotbCycle`)

Ключ	Имя поля	Тип	Описание
	ID	Text(16)	Идентификатор и одновременно название цикла.
	Status	Integer	Статус цикла. NULL — еще не обработан. 0 — непротиворечив, 1 — бесконечно много решений, непротиворечив, -1 — нет решений, противоречив, -2 — при погружении во фрагмент знаний АБС и последующем решении ЗЛП обнаружены противоречия.
	FirstXnProb	Double	Вероятность $p(x_n)$, вычисленная в результате решения уравнения (1). Рассчитывается перед второй фазой алгоритма и используется в ней.
	AbsCalcError	Double	Абсолютная величина разности между $p(x_n)$, рассчитанной по уравнению (1), и $p(x_n)$, получившейся в конце второй фазы работы алгоритма.

Таблица 2

Структура таблицы базы данных Node (в коде программы adotbNode)



Ключ	Имя поля	Тип	Описание
	CycleID	Text(16)	Идентификатор и одновременно название цикла
	N	Integer	Номер узла
	PosProb	Double	Исходно заданная вероятность $p(x_N x_{N-1})$
	NegProb	Double	Исходно заданная вероятность $p(x_N \bar{x}_{N-1})$

Таблица 3

Структура таблицы базы данных CycleNode (в коде программы adotbCycleNode)



Ключ	Имя поля	Тип	Описание
	CycleID	Text(16)	Идентификатор и одновременно название цикла
	N	Integer	Номер узла
	A	Double	Коэффициент a_N
	B	Double	Коэффициент b_N
	AtomProb	Double	Рассчитанная маргинальная вероятность $p(x_N)$
	ConjProb	Double	Рассчитанная маргинальная вероятность $p(x_{N-1}x_N)$
	RecPosProb	Double	Перерасчитанная вероятность $p(x_N x_{N-1})$
	RecNegProb	Double	Перерасчитанная вероятность $p(x_N \bar{x}_{N-1})$

Таблица 4

Реализация фаз и шагов алгоритма

Идентификатор процедуры	Шаг алгоритма
DoFirstPhase	Первая фаза алгоритма — распространяются сообщения M_i
CalculatePXn	Рассчитывается $p(x_n)$ согласно уравнению (1)
DoSecondPhase	Вторая фаза алгоритма — вычисление и распространение маргинальных вероятностей
CalculateAbsoluteError	Рассчитывается абсолютная величина разности между $p(x_n)$, рассчитанной по уравнению (1), и $p(x_n)$, получившейся в конце второй фазы работы алгоритма. [В листинге не приводится]
VerifyConditionalProbabilities	Перерасчитываются условные вероятности $p(x_i x_{i-1})$ и $p(x_i \bar{x}_{i-1})$ для сравнения с исходными. [В листинге не приводится]

В табл. 4 указаны основные шаги алгоритма и идентификаторы реализующих их процедур. Код процедур дан в листинге ниже.

```

procedure TdmDB.DoFirstPhase;
var
  a, b, r : double;
  PosProb, NegProb : double;
begin
  {Проверить открыты и не пусты ли таблицы --- код пропущен}

  CleanUpTable(adotbCycleNode);

  with adotbNode do
  begin
    a := 1;
    b := 0;
  
```

```

First;
while not EOF do
begin

    PosProb := FieldByName('PosProb'). AsFloat;
    NegProb := FieldByName('NegProb'). AsFloat;

    r := NegProb - PosProb;

    a := -r*a;
    b := NegProb - r*b;

    adotbCycleNode.Insert;
    adotbCycleNode.FieldByName('N').AsInteger :=
        FieldByName('N'). AsInteger;
    adotbCycleNode.FieldByName('a').AsFloat := a;
    adotbCycleNode.FieldByName('b').AsFloat := b;
    adotbCycleNode.Post;

    Next
end
end;
end;

procedure TdmDB.CalculatePXn;
var
    a, b : Double;
    PosProb, NegProb : Double;
begin
    with adotbCycle do
    begin
        if not Active then exit;
        if isEmpty then exit;

        if not adotbCycleNode.Active then exit;
        if adotbCycleNode.isEmpty then exit;
        adotbCycleNode.Last;

        a := adotbCycleNode.FieldByName('a').AsFloat;
        b := adotbCycleNode.FieldByName('b').AsFloat;

        if abs(1-a)>0.0000001 then
        begin
            Edit;
            FieldByName('FirstXnProb').AsFloat := b/(1-a);
            FieldByName('Status').AsInteger := 0;
            FieldByName('AbsCalcError').Value := Null;
            Post
        end
        else if abs(b)>0.0000001 then
        begin
            Edit;
            FieldByName('FirstXnProb').Value := Null;
            FieldByName('Status').AsInteger := -1;
            FieldByName('AbsCalcError').Value := Null;
            Post
        end
        else
        begin
            Edit;
            FieldByName('FirstXnProb').Value := Null;
            FieldByName('Status').AsInteger := 1;
            FieldByName('AbsCalcError').Value := Null;
            Post
        end
    end;
end;
end;
end;

```



```

procedure TdmDB.doSecondPhase;
var
    Status : Integer;
    PrevAtomP, CurrAtomP : Double;
    ConjP : Double;
    PosProb, NegProb : Double;
    FirstXnProb : Double;
begin
    with adotbCycleNode do
    begin
        {Проверить открыты и не пусты ли таблицы --- код пропущен}

        Status := adotbCycle.FieldByName('Status').AsInteger;

        if Status <> 0 then
        begin
            First;
            while not EOF do
            begin
                Edit;
                FieldByName('AtomProb').Value := Null;
                FieldByName('ConjProb').Value := Null;
                Post;
                Next
            end;
            exit
        end;

        FirstXnProb := adotbCycle.FieldByName('FirstXnProb').AsFloat;
        PrevAtomP := FirstXnProb;

        First;
        while not EOF do
        begin
            PosProb := adotbNode2.FieldByName('PosProb'). AsFloat;
            NegProb := adotbNode2.FieldByName('NegProb'). AsFloat;

            ConjP := PosProb*PrevAtomP;
            CurrAtomP := ConjP + NegProb*(1.0-PrevAtomP);

            Edit;
            FieldByName('AtomProb').Value := CurrAtomP;
            FieldByName('ConjProb').Value := ConjP;
            Post;

            PrevAtomP := CurrAtomP ;

            Next
        end
    end
end;

```

11. Структура данных для реализации ЛВВ в ФЗ

Для упрощения работы с ФЗ введена специальная система индексации. Каждый элемент ФЗ — конъюнкт над некоторым конечным набором атомов S . Если представить цепочку конъюнкций в виде последовательности 0 и 1, где 1, находящаяся в i -й позиции этой последовательности, будет означать наличие атома с порядковым номером i в данной цепочке, 0 — его отсутствие, а длина последовательности будет равна числу атомов в наборе S , то мы получим двоичное представление некоторого целого числа. Таким образом, каждо-

му элементу ФЗ однозначно сопоставляется некоторое целое число, которое будем называть *индексом* данного элемента. Такая система индексации позволяет задать оценки вероятностей элементов ФЗ в виде массива, причем каждый индекс этого массива осмыслен. Он является индексом элемента ФЗ.

Главным индексом ФЗ будем называть элемент, содержащий все атомы из набора S.

Фрагмент знаний в программном коде представлен классом KnowledgePattern, который позволяет задать структуру ФЗ, если известен глобальный индекс главного конъюнкта. Этот класс содержит методы, позволяющие ввести оценки вероятностей элементов, при этом по умолчанию оценка вероятности пустого элемента является точечной и равна 1.0, а оценки вероятности остальных элементов — интервальные и равны [0.0;1.0].

Логико-вероятностный вывод (ЛВВ) для ФЗ осуществляется методами двух классов: LocalScalarInferer для случая точечных оценок, и LocalIntervallInferer для интервального случая. Оба класса реализуют один и тот же интерфейс LocalInferer.

Эти классы получают на вход ФЗ и осуществляют в нем логико-вероятностный вывод, который включает в себя поддержание непротиворечивости и апостериорный вывод для трех видов свидетельств — детерминированного, стохастического и неопределенного. Проверка на непротиворечивость выполняется в два этапа: сначала оценки проверяются на согласуемость, а затем, в случае если они согласуемы, производится их уточнение. Полученные уточненные и апостериорные оценки сохраняются в этом же ФЗ.

Поля и методы классов, реализующих интерфейс LocalInferer, позволяют получить сведения о ходе проведения ЛВВ, а именно:

- проведена ли проверка оценок на согласуемость (isConsistency Checked),
- уточнены ли оценки (isReconciliationmade),
- проведен ли апостериорный вывод(isEvidencePropagated);

и сведения о результатах ЛВВ:

- согласуемы ли оценки (isConsistent),
- изменились ли оценки (isEstimatesArrayChanged),
- равна ли вероятность появления свидетельства нулю (isEvidenceOfZeroProbability),
- чему равна нижняя граница оценки вероятности появления свидетельства (evidencProbabilityLB),
- чему равна верхняя граница оценки вероятности появления свидетельства (evidenceProbabilityUB),
- интервальная ли оценка вероятности появления свидетельства (isEvidenceIntervallyEstimated),
- удалось ли осуществить пропагацию свидетельства (isPropagationOK).

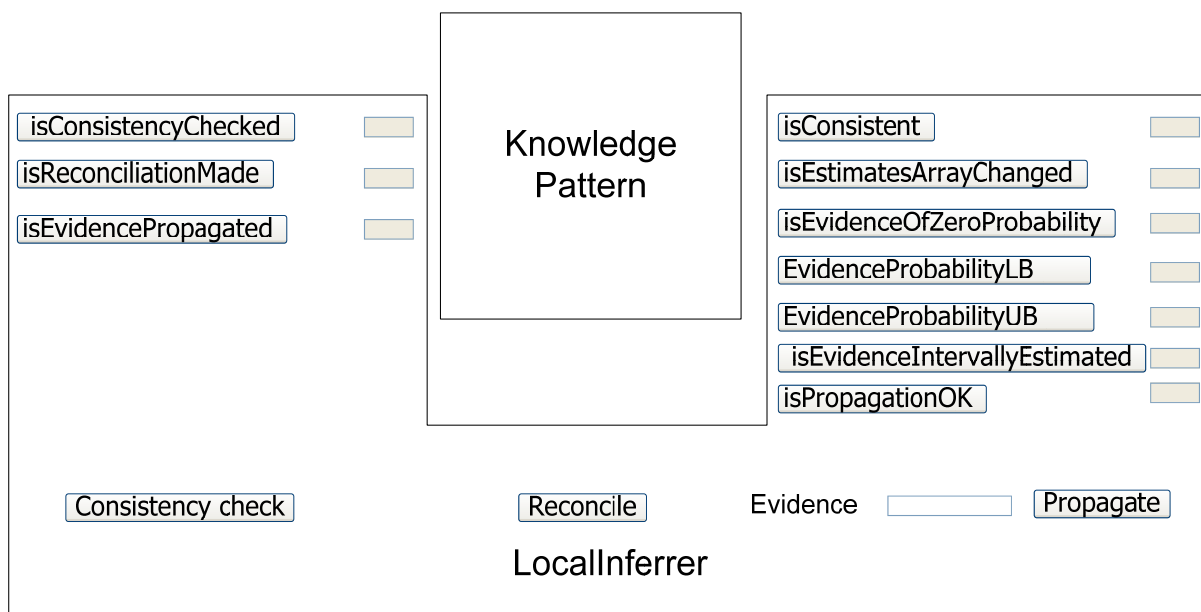


Рис. 7. Интерфейс LocalInferer.

Диаграмма классов для LocalInferer приведена на рис.8. Для задания свидетельства используются два класса: DeterministicEvidence для задания детерминированного свидетельства и IndetministicEvidence для задания стохастического свидетельства и свидетельства с неопределенностью (эти два свидетельства задаются с помощью фрагмента знаний, но оценки элементов стохастического свидетельства точечные, а неопределенного свидетельства — интервальные). Оба класса реализуют один и тот же интерфейс LocalInferer.

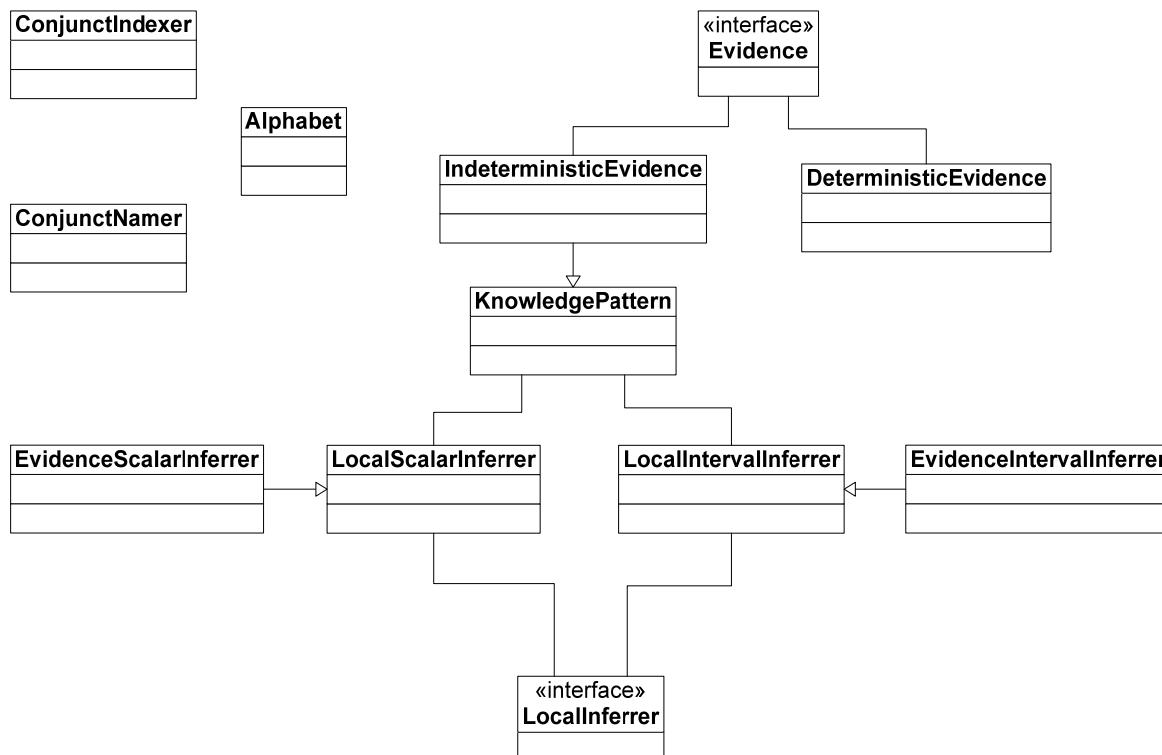


Рис. 8. Диаграмма классов.

Классы EvidenceScalarInferer и EvidenceIntervallInferer осуществляют перевод вероятностей конъюнктов в вероятности квантов и обратно в точечном и интервальном случаях соответственно. Это требуется, в частности, при пропагации свидетельства в апостериорном выводе.

Класс ConjunctIndexer нужен для работы с индексами конъюнктов. Например, для перевода индекса конъюкта из глобального (индекс конъюкта в рамках ФЗ) в локальный (индекс конъюкта в массиве) и наоборот, и для получения по локальному индексу конъюкта в ФЗ и самому ФЗ. Класс ConjunctNameer предназначен для представления конъюнктов не в виде числа, а в виде строки в привычной форме, т.е. в виде цепочки атомарных пропозиций. Для задания названия и количества этих атомарных пропозиций используется класс Alphabet.

12. Поддержание непротиворечивости в ФЗ

Пусть $\mathbf{P}^{(n)}$ — вектор вероятностей конъюнктов, $\mathbf{Q}^{(n)}$ — вектор вероятностей квантов, \mathbf{I}_n — матрица перевода вектора вероятностей конъюнктов в вектор вероятностей квантов, $\mathbf{0}^{[n]}$ — нулевой вектор, имеющий такой же порядок, что и матрица \mathbf{I}_n :

$$\mathbf{I}_1 = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{I}_n = \begin{bmatrix} \mathbf{I}_{n-1} & -\mathbf{I}_{n-1} \\ \mathbf{0} & \mathbf{I}_{n-1} \end{bmatrix}; \quad \mathbf{I}_n \times \mathbf{P}^{(n)} = \mathbf{Q}^{(n)}.$$

В программном коде матрица \mathbf{I}_n задается специальным классом IJMatrix и строится следующим образом:

```
private void IMatrix(){
    for(int i=0;i<dimension;i++){
        for(int j=0;j<dimension;j++){
            if((i&j)!=i) matrix[i][j]=0;
            else matrix[i][j]=((Integer.bitCount(j&i)==1)?-1:1);
        }
    }
}
```

Этот класс также позволяет задать матрицу \mathbf{J}_n — матрицу перевода вектора вероятностей квантов в вектор вероятностей конъюнктов. Эта матрица обратная матрице \mathbf{I}_n :

$$\mathbf{J}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{J}_n = \begin{bmatrix} \mathbf{J}_{n-1} & \mathbf{J}_{n-1} \\ \mathbf{0} & \mathbf{J}_{n-1} \end{bmatrix}; \quad \mathbf{J}_n \times \mathbf{Q}^{(n)} = \mathbf{P}^{(n)}.$$

Методы класса IJMatrix позволяют осуществлять умножение этой матрицы \mathbf{I}_n на вектор переменных типа double и умножение матрицы на вектор переменных ЗЛП (тип IloNumExpr) (реализация такого умножения представлена ниже):

```
public IloNumExpr [] multiplyOnIloVector(IloCplex cplex, IloNumVar [] iloVector){
    IloNumExpr [] result= new IloNumExpr[dimension];
    try{
        for(int i=0;i<dimension;i++){
            result[i] = cplex.scalProd(iloVector, matrix[i], 0, dimension);
        }
    }
    ...
    return result;
}
```

Случай точечных оценок. Цель — проверка соответствия точечных оценок вероятностей элементов ФЗ множеству ограничений, вытекающих из вероятностной аксиоматики, т.е. проверка на согласуемость. Второго этапа поддержания непротиворечивости, т.е. уточнения оценок, в точечном случае не производится.

Исходные данные — вектор точечных оценок вероятностей \mathbf{P}_0 . Если выполнено $\mathbf{I}_n \times \mathbf{P}_0 \geq \mathbf{0}^{[n]}$, то оценки непротиворечивы, в противном случае — противоречивы. В коде программы проверка выполняется методом consistencyCheck() класса LocalScalarInferer.

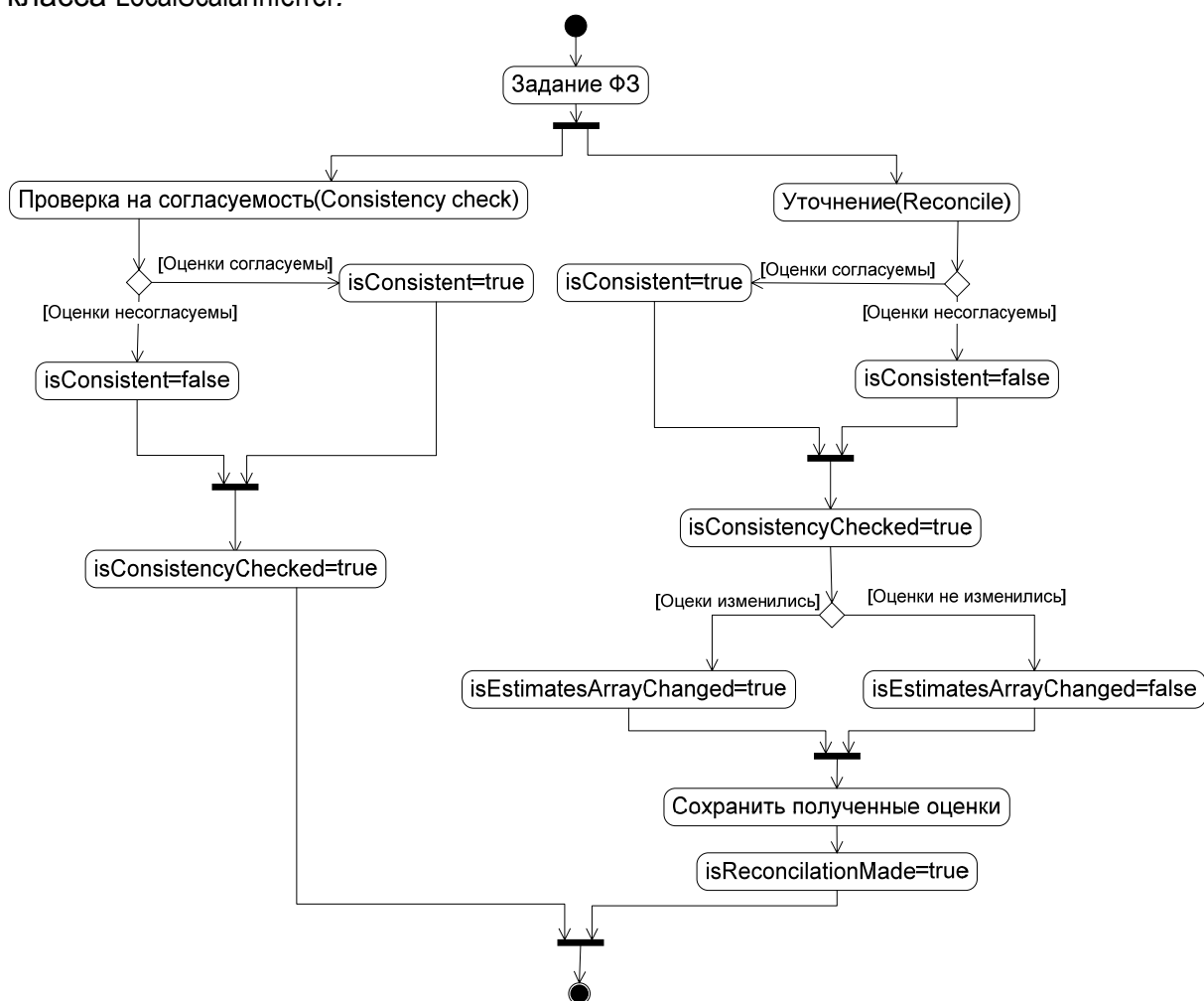


Рис. 9. Алгоритм поддержания непротиворечивости.

```

public boolean consistencyCheck(){
    // Probability estimate of the empty conjunct is scalar and equal to 1.0
    if((KPPProbability[0]!=1.0)) {
        isConsistent=false;
        return false;
    }
    IJMatrix im=new IJMatrix(numOfAtoms,true);
    double [] res=im.multiplyOnVector(KPPProbability);
    // Checks probabilistic logic acsioms constraints
    for(int i=1;<res.length;i++){
        if(res[i]<0) {
            isConsistent=false;
            isConsistencyChecked=true;
            return false;
        }
    }
}
  
```

```

}
isConsistent=true;
isConsistencyChecked=true;
return true;
}

```

Случай интервальных оценок. На основе исходных данных — $P_o^-,^{(n)}$ и $P_o^+,^{(n)}$, т.е. нижних и верхних границ вероятностей $P^{(n)}$ соответственно, — формируются ЗЛП с ограничениями:

$$I_n \times P^{(n)} \geq \mathbf{0}^{(n)} \quad (\text{ограничения } E^{(n)}),$$

$$P_o^-,^{(n)} \geq P^{(n)} \geq P_o^+,^{(n)} \quad (\text{ограничения } D^{(n)}).$$

Набор интервальных оценок непротиворечив (согласуем), если для произвольного элемента ФЗ при выборе произвольной точки из его интервальной оценки в интервальных оценках остальных элементов можно выбрать точки так, что получившийся набор точечных оценок непротиворечив.

Последовательно формируются новые (уточненные) нижние и верхние границы:

$$P^{+,^{(n)}} = \max_{R^{(n)}} \{P^{(n)}\}; \quad P^{-,^{(n)}} = \min_{R^{(n)}} \{P^{(n)}\}; \quad R^{(n)} = D^{(n)} \cup E^{(n)}.$$

В коде программы уточненные оценки сохраняются как новые оценки фрагмента знаний. Проверка на согласуемость в ФЗ осуществляется методом consistencyCheck() класса LocalIntervallInferer. А поддержание непротиворечивости (т.е. проверка на согласуемость и уточнение оценок) производится методом Reconcile() того же класса и сводится к решению ЗЛП. Стоит отметить, что для того чтобы узнать согласуемы ли интервальные оценки, достаточно проверить существует ли хотя бы один непротиворечивый набор точечных оценок, который является распределением вероятностей элементов ФЗ. Т.е. достаточно решить ЗЛП для одной границы вероятности одного элемента.

Приведем код метода, осуществляющего и проверку на непротиворечивость и уточнение оценок:

```

public void reconcile(){
    boolean solve=true;
    try{
        IloCplex cplex = new IloCplex();
        // Sets domains constraints
        IloNumVar [] pc=cplex.numVarArray(numOfElements,
                                         KPElementProbabilityLB,KPElementProbabilityUB);

        IJMatrix im=new IJMatrix(numOfAtoms,true);
        IloNumExpr [] exprs=im.multiplyOnIloVector(cplex,pc);
        // Checks probabilistic logic axioms constraints
        for(int i=0;i<numOfElements;i++){
            cplex.addLe(0.0,exprs[i]);
        }
        // Makes reconciliation for estimates lower bound
        for(int i=0;i<numOfElements;i++){
            IloObjective minObj=cplex.addMinimize(pc[i]);
            if(cplex.solve()){
                double val=cplex.getValue(pc[i]);
                if(Math.abs(val-KPElementProbabilityLB[i])>kp.precision){
                    isEstimatesArrayChanged=true;
                }
            }
            KPElementProbabilityLB[i]=val;
            cplex.remove(minObj);
        }
    }
}

```

```

else{
    solve=false;
    return;
}
}
// Makes reconciliation for estimates lower bound
...
cplex.end();
}
...
isConsistent=solve;
isReconciliationMade=true;
}

```

13. Апостериорный вывод в ФЗ

Пусть дан фрагмент знаний C . Мы что-то узнали об оценках вероятностей — поступило некоторое свидетельство $\langle \diamond \rangle$. Как оно повлияет на оценки вероятностей элементов заданного ФЗ?

Первой задачей апостериорного вывода является оценка вероятности или ожидаемой вероятности появления свидетельства, или кортежа свидетельств над заданным ФЗ C . Эта вероятность обозначается $p(\langle \diamond \rangle | C)$.

Второй задачей апостериорного вывода является оценка апостериорной вероятности или ожидаемой апостериорной вероятности цепочек конъюнкций ранее определенного вида Z_Y , но не имеющих общих атомарных пропозиций с поступившим свидетельством или кортежем свидетельств $p_a(Z_Y | \langle \diamond \rangle)$.

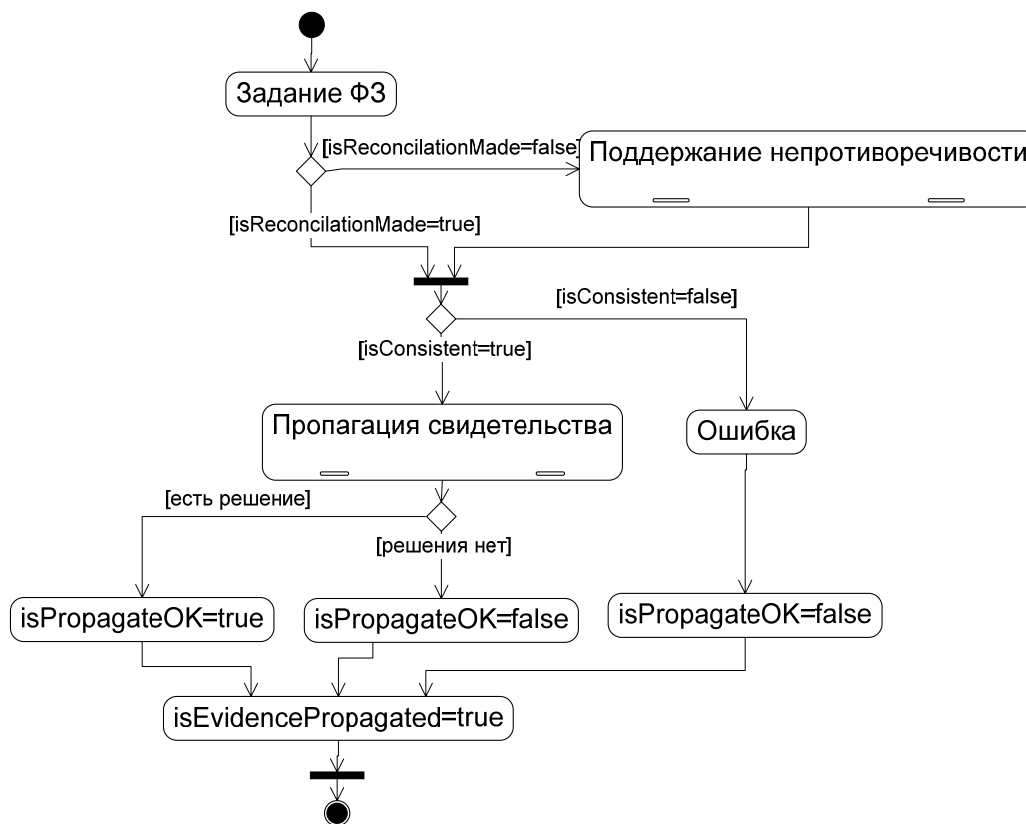


Рис. 10. Алгоритм апостериорного вывода.

В коде программы апостериорный вывод осуществляется методом `propagate(Evidence evidence)` класса `LocalScalarInferer` в точечном случае и класса `LocalInter-`

valInferer в интервальном случае. Этот метод в зависимости от типа свидетельства осуществляет пропагацию детерминированного свидетельства, стохастического свидетельства или свидетельства с неопределенностью.

Стоит отметить, что пропагировать свидетельство можно только в случае, если и фрагмент знаний, и свидетельство непротиворечивы.

14. Пропагация детерминированного свидетельства

Детерминированное свидетельство может быть атомарным, тогда оно записывается $\langle x \rangle$ (что означает « x истинно») или $\langle \bar{x} \rangle$ (что означает « x ложно»). Кортеж детерминированных свидетельств — цепь атомарных свидетельств. Например, $\langle \bar{x}_1 x_2 \rangle$. Коротко можно записывать $\langle X \rangle = \langle x_1 x_2 \dots x_m \rangle$ или $\langle \tilde{X} \rangle = \langle \tilde{x}_1 \tilde{x}_2 \dots \tilde{x}_m \rangle$, последнее означает, что нас интересует некоторое означивание цепочки конъюнкций X .

На первом этапе пропагации непротиворечивого детерминированного свидетельства проверяется, содержит ли оно отрицательно означенные элементы. Если содержит, то переходим к рассмотрению переозначенного ФЗ, у которого в элементах атом, который отрицательно означен в свидетельстве, также отрицательно означен в новом ФЗ. Это нужно, поскольку за счет процедуры переозначивания атомов и пересчета вероятностей можно считать, что поступают лишь свидетельства, являющиеся элементами ФЗ. Далее распространяем свидетельство в новом ФЗ. Получаем апостериорные оценки вероятностей элементов и производим обратное переозначивание. Случай, когда вероятность появления свидетельства равна 0, является важным частным случаем и рассматривается отдельно, поскольку в этом случае про апостериорные вероятности можно только сказать, что их оценки равны $[0;1]$.

Точечные оценки: Решение *первой задачи* апостериорного вывода рассчитывается по формуле

$$p(\langle X \rangle | C) := p(X),$$

Благодаря переозначиванию атомов всегда можно считать, что поступившее свидетельство является элементом ФЗ. Поэтому вероятность появления свидетельства $\langle X \rangle$ над фрагментом знаний C просто равна точечной вероятности элемента X этого ФЗ $p(X)$.

Решение *второй задачи* рассчитывается по формуле

$$p_a(Z_Y | \langle X \rangle) := \frac{p(XZ_Y)}{p(X)}.$$

Здесь Z_Y — цепочка конъюнкций, не имеющая общих атомов со свидетельством $\langle X \rangle$.

В коде программы пропагация детерминированного свидетельства в ФЗ с точечными оценками осуществляется методом `propagateDeterministicEvidence(Evidence evidence)` класса `LocalScalarInferer`.

```
private void propagateDeterministicEvidence(Evidence evidence){
    int negativeIndex;
    try{
        negativeIndex=evidence.getNegativeIndex();
        if(negativeIndex!=0) makeSignSubstitution(kp,evidence);
        //solves first task of aposterior inference
        evidenceProbabilityLB=kp.getProbabilityLB(evidence.getGlobalIndex());
    }
```



```

evidenceProbabilityUB= evidenceProbabilityLB;
//the zero probability case
if(Math.abs(evidenceProbabilityLB)<0){
  isEvidenceOfZeroProbability=true;
  for(int i=0;i<numOfElements;i++){
    kp.initializeIntervalEstimation(i,0,1);
  }
}
else{
  for(int i=1;j<numOfElements;i++){
    KPPProbability[i]=kp.getProbabilityLB()[i][evidence.getGlobalIndex()]
    /kp.getProbabilityLB()[evidence.getGlobalIndex()];
  }
}
if(negativeIndex!=0) makeSignSubstitution(kp,evidence);
isPropagationOK=true;
...
}

```

Интервальные оценки: Решение *первой задачи* рассчитывается исходя из формул:

$$p(\langle X \rangle | C) := p(X);$$

$$p^-(X) \leq p(\langle X \rangle | C) \leq p^+(X).$$

Так же как и в точечном случае, можно считать, что поступившее свидетельство является элементом ФЗ. Поэтому вероятность появления свидетельства $\langle X \rangle$ над фрагментом знаний C просто равна интервальной вероятности элемента X этого ФЗ $p(X)$. А по ограничениям предметной области D верно, что

$$p^-(X) \leq p(\langle X \rangle) \leq p^+(X).$$

Вторая задача рассчитывается исходя из формулы:

$$p_a(Z_Y | \langle X \rangle) := \left[\min_R \frac{p(XZ_Y)}{p(X)}; \max_R \frac{p(XZ_Y)}{p(X)} \right], \text{ где } R = D \cup E,$$

E — набор ограничений из вероятностной логики :

$$I_n \times \mathbf{P}^{(n)} \geq \mathbf{0}^{[n]}.$$

Решение второй задачи апостериорного вывода при пропагации детерминированного свидетельства в фрагменте знаний с интервальными оценками сводится к решению задач гиперболического программирования. Но эти задачи можно свести к задачам линейного программирования. Алгоритм предложен ниже.

1. Предполагаем, что свидетельство — элемент ФЗ. Иначе переходим к рассмотрению переозначенного ФЗ. Пусть ζ — индекс элемента, являющегося свидетельством.
2. Положим:

$$\xi := \frac{1}{p(X_{[\zeta]})}; \xi \geq 1;$$

$$d(X_{[i]}) := \xi p(X_{[i]}).$$

Здесь $X_{[i]}$ — элемент ФЗ с индексом i .

Выполнив все преобразования, получим ЗЛП относительно d со следующими ограничениями:

$$\begin{cases} d(e_\lambda) = \xi \geq 1, \\ \xi p^-(X_{[i]}) \leq d(X_{[i]}), \\ d(X_{[i]}) \leq \xi p^+(X_{[i]}), \\ d(X_{[i]}) \geq 0. \end{cases}$$

В коде программы пропация детерминированного свидетельства осуществляется методом `propagateDeterministicEvidence(Evidence evidence)` класса `LocalIntervalInferer`.

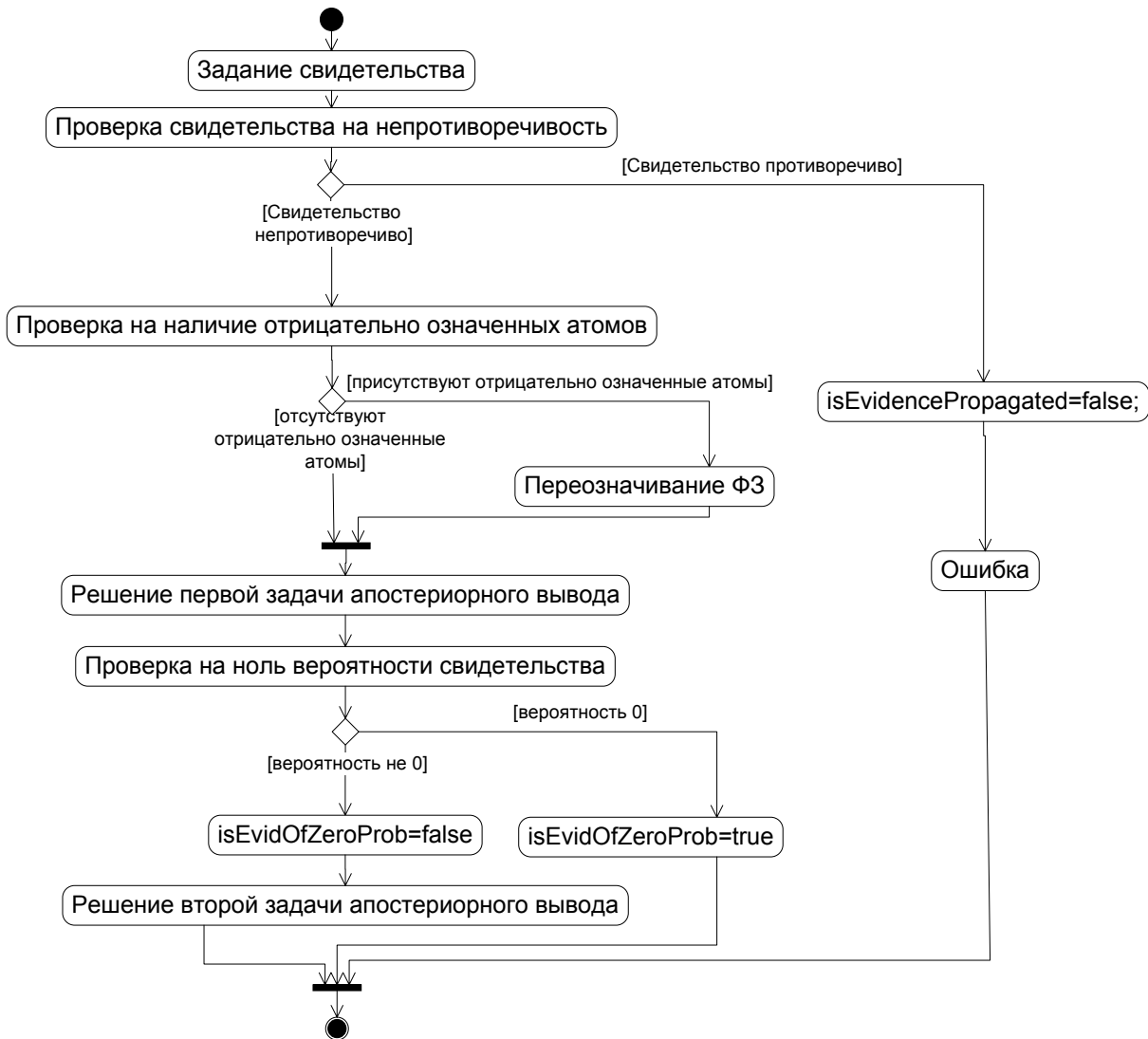


Рис. 11. Пропация детерминированного свидетельства.

```

private void propagateDeterministicEvidence(Evidence evidence){
    int negativeIndex;
    try{
        negativeIndex=evidence.getNegativeIndex();
        if(negativeIndex!=0) makeSignSubstitution(kp,evidence);
        // solves first task of aposterior inference
        evidenceProbabilityLB=kp.getProbabilityLB(evidence.getGlobalIndex());
        evidenceProbabilityUB=kp.getProbabilityUB(evidence.getGlobalIndex());
    }
}

```

```

int evidenceGlobalIndex=evidence.getGlobalIndex();
double [][] posteriorProbLB=new double[2][numOfElements];
double [][] posteriorProbUB=new double[2][numOfElements];
//the zero probability case
if((Math.abs(evidenceProbabilityLB)<kp.precision)&&(Math.abs(evidenceProbabilityUB)<0)){
    isEvidenceOfZeroProbability =true;
    for(int i=0;i<numOfElements;i++){
        kp.initializeIntervalEstimation(i,0,1);
    }
}
else{
    boolean solve=true;
    try{
        IloCplex cplex = new IloCplex();
        double [] ub=new double [numOfElements];
        double [] lb=new double [numOfElements];
        for(int i=0;i<numOfElements;i++){
            ub[i]=Double.MAX_VALUE;
            lb[i]=-Double.MAX_VALUE;
            if(i==evidenceGlobalIndex){
                ub[i]=1.0;
                lb[i]=1.0;
            }
        }
        IloNumVar [] d=cplex.numVarArray(numOfElements,lb,ub);
        IJMatrix im=new IJMatrix(numOfAtoms,true);

        IloNumExpr [] dExprs=im.multiplyOnIloVector(cplex,d);
        cplex.addGe(d[0],1.0);
        //Checks probabilistic logic axioms constraints
        for(int j=0;j<numOfElements;j++){
            cplex.addGe(dExprs[j],0.0);
        }
        for(int j=1;j<numOfElements;j++){
            cplex.addLe(cplex.prod(d[0],KPElementProbabilityLB[j]),d[j]);
            cplex.addGe(cplex.prod(d[0],KPElementProbabilityUB[j]),d[j]);
        }
        //calculates lower bounds
        for(int i=0;i<numOfElements;i++){
            if(posteriorProbLB[1][i]!=1){
                IloObjective minObj=cplex.addMinimize(d[i|evidenceGlobalIndex]);
                if( cplex.solve()){
                    double val=cplex.getValue(d[i|evidenceGlobalIndex]);
                    posteriorProbLB[0][i]=val;
                    posteriorProbLB[0][i|evidenceGlobalIndex]=val;
                    posteriorProbLB[1][i]=1;
                    posteriorProbLB[1][i|evidenceGlobalIndex]=1;
                    cplex.remove(minObj);
                }
            }
        }
        else{
            solve=false;
            return;
        }
    }
    //calculates upper bounds
    ...
}
isEvidencePropagated=solve;
}
kp.setProbabilityLB(posteriorProbLB[0]);
kp.setProbabilityUB(posteriorProbUB[0]);
makeSignSubstitution(kp,evidence);
isPropagationOK=true;
...
}

```

15. Пропагация стохастического свидетельства в ФЗ

Стохастическое свидетельство характеризуется точечной оценкой апостериорной вероятности своей истинности $p_{[a]}(\tilde{X})$. Его будем записывать как $\langle p_{[a]}(\tilde{X}) \rangle$. Т.е. мы знаем апостериорную оценку вероятностей всех означиваний $\langle \tilde{X} \rangle = \langle \tilde{x}_1 \tilde{x}_2 \dots \tilde{x}_m \rangle$. Кортеж стохастических свидетельств представляется в виде фрагмента знаний.

На первом этапе пропагации непротиворечивого стохастического свидетельства, представленного в виде ФЗ, производится перевод оценок вероятностей конъюнктов свидетельства в оценки вероятностей квантов. Затем каждый элемент свидетельства, являющийся уже квантом, пропагируется как детерминированное свидетельство в копии ФЗ; полученные апостериорные оценки умножаются на вероятность кванта свидетельства. Затем полученные ФЗ складываются и получившийся в итоге ФЗ является ФЗ с апостериорными оценками при пропагации стохастического свидетельства.

Точечные оценки. Решение *первой задачи* вычисляются по формуле

$$p(\langle p_{[a]}(\tilde{X}) \rangle | C) = \sum_{\tilde{X}} p(\tilde{X} | C) p_{[a]}(\tilde{X}).$$

Здесь $\langle p_{[a]}(\tilde{X}) \rangle$ — стохастическое свидетельство, C — фрагмент знаний, $p(\tilde{X} | C)$ — вероятность появления одного из означиваний \tilde{X} стохастического свидетельства как детерминированное свидетельство над ФЗ.

Решение *второй задачи*:

$$p(Z_Y | \langle p_{[a]}(\tilde{X}) \rangle) = \sum_{\tilde{X}} p_a(Z_Y | \tilde{X}) p_{[a]}(\tilde{X}).$$

Здесь Z_Y , так же как и в детерминированном случае, — цепочка конъюнкций, не имеющая общих атомов с X , а $p_a(Z_Y | \tilde{X})$ — апостериорные вероятности элементов ФЗ после пропагации одного из означиваний \tilde{X} стохастического свидетельства как детерминированное свидетельство над ФЗ.

В программном коде перевод точечных оценок конъюнктов в оценки квантов производится методом `ConjunctProbability2QuantProbability(int index)` класса `EvidenceScalarInferer`, а пропагация стохастического свидетельства в ФЗ с точечными оценками — методом `propagateStochasticEvidence(Evidence evidence)` класса `LocalScalarInferer`. Для сложения фрагментов знаний и умножения ФЗ на скаляр предусмотрены специальные методы класса `KnowledgePattern`, а именно `multiplyOnScalar(int scalar)` и `add(KnowledgePattern kp)` соответственно.

```
private void propagateStochasticEvidence(Evidence evidence){
    int evidenceNumOfElements=evidence.getNumOfElements();
    KnowledgePattern resultKP=kp;
    EvidenceScalarInferer evidenceInferer=new EvidenceScalarInferer(evidence);
    //converts conjunct probabilities to quant probability
    double [] quantsProbability=evidenceInferer.conjunctLB2QuantLB();
    try{
        int i=1;
        //propagates each element of stochastic evidence as deterministic evidence
        while(i<evidenceNumOfElements){
            KnowledgePattern initKP=kp;
            LocalScalarInferer KPIInferer=new LocalScalarInferer(initKP);
```

```

DeterministicEvidence de=new DeterministicEvidence();
de.initialize(evidence.getGlobalIndex(),i);
KPInferer.propagateDeterministicEvidence(de);
if(Math.abs(quantsProbability[i])<kp.precision){
    i++;
}
else{
    if(Math.abs(KPInferer.evidenceProbabilityLB)<kp.precision){
        isEvidenceOfZeroProbability=true;
    }
    evidenceProbabilityLB+=KPInferer.evidenceProbabilityLB*quantsProbability[i];
    evidenceProbabilityUB=evidenceProbabilityLB;
    initKP.multiplyOnScalar(quantsProbability[i]);
    resultKP.add(initKP);
    i++;
}
...
}
isPropagationOK=true;
isEvidencePropagated=true;
}

```

Интервальный случай. Решение *первой задачи* вычисляется по формуле:

$$p(< p_{[a]}(\tilde{X}) > | C) = \left[\sum_{\tilde{X}} p^{-}(< \tilde{X} > | C) p_{[a]}(\tilde{X}); \sum_{\tilde{X}} p^{+}(< \tilde{X} > | C) p_{[a]}(\tilde{X}) \right].$$

Здесь $p^{-}(\tilde{X} | C)$, $p^{+}(\tilde{X} | C)$ — нижняя и верхняя границы вероятности появления одного из означиваний \tilde{X} стохастического свидетельства как детерминированное свидетельство над ФЗ.

Вторая задача решается исходя из формулы

$$p_a(Z_Y | < p_{[a]}(\tilde{X}) >) = \left[\sum_{\tilde{X}} p_a^{-}(Z_Y | < \tilde{X} >) p_{[a]}(\tilde{X}); \sum_{\tilde{X}} p_a^{+}(Z_Y | < \tilde{X} >) p_{[a]}(\tilde{X}) \right].$$

Здесь $p_a^{-}(Z_Y | < \tilde{X} >)$, $p_a^{+}(Z_Y | < \tilde{X} >)$ — нижняя и верхняя границы апостериорной вероятности элементов ФЗ после пропагации одного из означиваний \tilde{X} стохастического свидетельства как детерминированное свидетельство над ФЗ.

В программном коде перевод интервальных оценок конъюнктов в оценки квантов производится методом `ConjunctProbability2QuantProbability(int index)` класса `EvidenceIntervallInferer` для интервального случая, а пропагация стохастического свидетельства — методом `propagateStochasticEvidence(Evidence evidence)` класса `LocalIntervallInferer`.

```

private void propagateStochasticEvidence(Evidence evidence){
    int evidenceNumOfElements=evidence.getNumOfElements();
    KnowledgePattern resultKP=kp;
    EvidenceIntervallInferer evidenceInferer=new EvidenceIntervallInferer(evidence);
    double [] quantsProbability=evidenceInferer.conjunctLB2QuantLB();
    try{
        int i=1;
        while(i<evidenceNumOfElements){
            KnowledgePattern initKP=kp;
            LocalIntervallInferer KPInferer=new LocalIntervallInferer(initKP);
            DeterministicEvidence de=new DeterministicEvidence();
            de.initialize(evidence.getGlobalIndex(),i);
            KPInferer.propagateDeterministicEvidence(de);
            if(Math.abs(quantsProbability[i])<kp.precision){
                i++;
            }
        }
    }
    else{

```

```

if(Math.abs(KPInferer.evidenceProbabilityLB)<kp.precision){
  isEvidenceOfZeroProbability=true;
}
evidenceProbabilityLB+=KPInferer.evidenceProbabilityLB*quantsProbability[i];
evidenceProbabilityUB+=KPInferer.evidenceProbabilityUB*quantsProbability[i];
initKP.multiplyOnScalar(quantsProbability[i]);
resultKP.add(initKP);
i++;
...
isPropagationOK=true;
isEvidencePropagated=true;
}

```

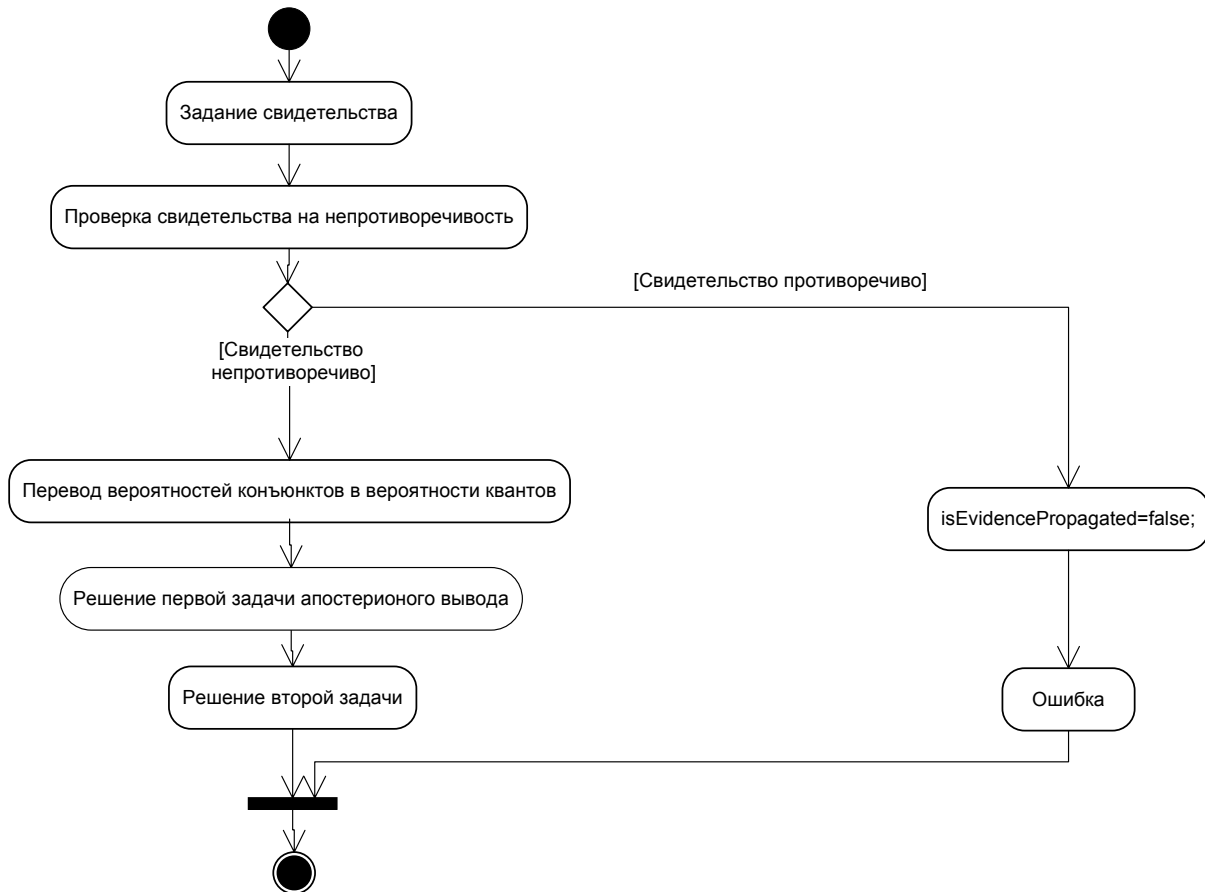


Рис. 12. Пропагация стохастического свидетельства.

9. Заключение

В работе в рамках логико-вероятностного подхода был рассмотрен направленный цикл в БСД, который изначально задается тензорами $p(x_i | x_{i-1})$.

На основе формулы полной вероятности был предложен алгоритм, основывающийся на распространении сообщений в цикле от узла к узлу, который позволяет вычислить маргинальные вероятности атомов $p(x_{i-1})$, стоящих в узлах цикла, и конъюнкций атомов $p(x_{i-1}x_i)$, стоящих в двух смежных узлах цикла.

Алгоритм содержит две основные фазы и три дополнительных шага. На первой фазе распространяются сообщения, которые позволяют сформировать уравнение $a_n\pi + b_n = \pi$. Следующий шаг — это анализ полученного уравне-

ния и поиск его решения, если оно единственное. Если такое решение найдено, то происходит вторая фаза алгоритма: в цикле от узла к узлу предаются и вычисляются маргинальные вероятности $p(x_{i-1})$ и $p(x_{i-1}x_i)$. После завершения второй фазы вычисляется абсолютная величина разности между $p(x_n)$, рассчитанной по уравнению (1), и $p(x_n)$, получившейся в конце второй фазы работы алгоритма. Наконец, для проверки полученных численных результатов пересчитываются условные вероятности $p(x_i | x_{i-1})$ и $p(x_i | \bar{x}_{i-1})$.

Основная часть алгоритма (без двух последних проверочных шагов) линейна по числу сообщений и по числу операций сложения, вычитания и умножения. Деление используется только один раз между первой и второй фазой.

Показано, что алгоритм, основанный на передаче сообщений, позволяет получить те же результаты, что и алгоритм, предложенный ранее, опирающийся на решение линейного матричного уравнения.

С учетом возможности вычислить маргинальные вероятности атомов был предложен алгоритм стохастического моделирования означиваний узлов направленного БСД-цикла. Этот алгоритм основывается на адаптированном подходе Гиббса, широко используемом в стохастическом моделировании на основе вероятностных сетей.

Также показано, что для выполнения логико-вероятностного вывода в байесовской сети доверия, содержащей направленный цикл, достаточно погрузить БСД-цикл во фрагмент знаний алгебраической байесовской сети и выполнить соответствующий вид логико-вероятностного вывода в ФЗ АБС.

Литература

1. Городецкий В. И., Тулупьев А. Л. Формирование непротиворечивых баз знаний с неопределенностью // Изв. РАН. Сер. Теория и системы управления. 1997. Т. 5. С. 33–42.
2. Городецкий В. И. Алгоритмизация приближенных рассуждений на основе байесовского вывода // Труды 2-й Всесоюзной конференции «Искусственный интеллект-90». Т. 1. Минск, 1990. С. 86–92.
3. Городецкий В. И. Байесовский вывод. Препринт №149. Л.: ЛИИАН, 1991. 38 с.
4. 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.
5. Николенко С. И., Сироткин А. В., Тулупьев А. Л. Направленный цикл и его влияние на соседние узлы в байесовских сетях доверия // Всероссийская научная конференция по нечетким системам и мягким вычислениям НСМВ-2006 (20–22 сентября 2006 г., Тверь): Труды конференции. М.: Физматлит, 2006. С. 150–166.
6. Николенко С. И., Тулупьев А. Л. Учет направленных циклов в байесовских сетях доверия: семантика и вопросы сложности // Сб. научных трудов III Международного научно-практического семинара «Интегрированные модели и мягкие вычисления в искусственном интеллекте». М.: Физматлит, 2005. С. 376–382.
7. Николенко С. И., Тулупьев А. Л. Простейшие циклы в байесовских сетях доверия: распределение вероятностей и возможность его непротиворечивого задания // Труды СПИИРАН. 2004. Вып. 2, т. 1. СПб.: Наука, 2004. С. 119–126.
8. Николенко С. И., Тулупьев А. Л. Разворот ребер как метод работы с направленными циклами в байесовских сетях // Научная сессия МИФИ-2005. Сборник научных трудов (в 15 томах). Том 3. Интеллектуальные системы и технологии. М.: МИФИ, 2005. С. 176–178.
9. Тулупьев А. Л., Николенко С. И. Циклы обратной связи узлов с одним предшественником в байесовских сетях доверия // Труды IX конференции «Региональная информатика», Санкт-Петербург, 2004. С. 65–66.
10. Тулупьев А. Л., Николенко С. И., Сироткин А. В. Байесовские сети: логико-вероятностный подход. СПб.: Наука, 2006. 608 с.

11. Тулупьев А. Л., Николенко С. И., Сироткин А. В. Циклы в байесовских сетях: вероятностная семантика и отношения с соединными узлами // Труды СПИИРАН. 2006. Вып. 3, т. 1. СПб.: Наука, 2004. С. 240–263.
12. Gorodetsky V. I., Drozdgin V. V., Jusupov R. M. Application of Attributed Grammar and Algorithmic Sensitivity Model for Knowledge Representation and Estimation // Artificial Intelligence and Information, Control System of ROBOTSA. Amsterdam: Elsevier Science Publishers B. V., 1984. С. 232–237.
13. Jensen F. V. Bayesian Networks and Decision Graphs. NY.: Springer-Verlag, 2001. 268 p.
14. Tulupyeu A. L., Nikolenko S. I. Directed Cycles in Bayesian Belief Networks: Probabilistic Semantics and Consistency Checking Complexity // MICAI 2005: Advances in Artificial Intelligence. Proceedings Series: Lecture Notes in Computer Science; Subseries: Lecture Notes in Artificial Intelligence, Vol. 3789 / Gelbukh, Alexander; Terashima, Hugo (Eds.) 2005. XXVI. P. 214–223.