Знания-Онтологии-Теории (ЗОНТ-09)

LP-структуры представления знаний и принципы их реализации

Махортов С.Д. [1]

[1]*Воронежский госуниверситет, Университетская пл., д. 1, г. Воронеж, 394006, Россия.*

sd@expert.vrn.ru

Аннотация. *Вводятся LP-структуры – основанные на решетках алгебраические системы, содержащие семантику продукционно-логического вывода на иерархических знаниях (Lattice-Production Structures). Основная идея состоит в моделировании связей (совокупности правил) дополнительным бинарным отношением с заданными свойствами. При этом определяющее решетку исходное отношение частичного порядка отражает универсальные тавтологии и является фиксированным. Второе отношение порождается логическими связями конкретной предметной области и может подвергаться эквивалентным преобразованиям. Рассматриваются принципы компьютерной реализации LP-структур. Описанные результаты могут быть применены для практических исследований и автоматической оптимизации баз знаний продукционного типа.*

Ключевые слова: продукционная система, алгебраическая модель, решетка, компьютерная реализация

1 Введение

Математическая структура – это множество, на элементах которого аксиоматически заданы некоторые отношения и операторы. Один из видов структур образуют решетки, предоставляющие эффективный способ формального представления знаний [1]. В частности, решетка множеств может быть образована наборами фактов базы знаний продукционной системы.

В работах [2-3] предложен математический аппарат, позволяющий рассматривать задачи формализации продукционно-логического вывода с точки зрения теории решеток и отношений. Основная идея состоит в моделировании продукционных связей (совокупности правил) дополнительным бинарным отношением с заданными свойствами (рефлексивность, транзитивность и некоторые другие свойства, зависящие от конкретной модели). При этом определяющее решетку исходное отношение частичного порядка отражает универсальные тавтологии и является фиксированным. Второе (дополнительное) отношение порождается логическими связями конкретной предметной области и может подвергаться эквивалентным преобразованиям. Решетка с заданным на ней логическим отношением была названа LP-структурой (Lattice-Production Structure). Изложенные в [2-3] результаты могут быть применены для автоматизации исследований баз знаний и логических программ на основе формальных моделей, включая их верификацию и оптимизацию.

В настоящей работе приведены определения и основные свойства LP-структур, а также представлены принципы их компьютерной реализации. Программная реализация LP-структур опирается на методы представления решеток, бинарных отношений, а также учитывает специфику архитектуры рассматриваемых алгебраических систем.

2 Основные понятия и обозначения

Напомним некоторые понятия из общей теории.

Бинарное отношение R на произвольном множестве F называется рефлексивным, если для любого $a \in F$ справедливо $(a,a) \in R$; транзитивным, если для любых $a,b,c \in F$ из $(a,b),(b,c) \in R$ следует $(a,c) \in R$. Известно, что существует замыкание R^* произвольного отношения R относительно свойств рефлексивности и транзитивности (рефлексивно-транзитивное замыкание).

Обратная задача – нахождение транзитивной редукции: по данному R ищется *минимальное* отношение R' такое, что его транзитивное замыкание совпадает с транзитивным замыканием R. Напомним также, что для частично упорядоченных множеств различаются понятия минимального элемента (для него нет меньшего элемента) и наименьшего элемента (он меньше всех). В [4] приведен алгоритм построения транзитивной редукции ориентированных графов. Показано, что эта задача вычислительно эквивалентна построению транзитивного замыкания, и доказана единственность транзитивной редукции ациклического графа.

Решеткой называется множество с частичным порядком $\leq$ («не больше», «содержится»), на котором для любой пары элементов определены операции $\wedge$ («пересечение») и $\vee$ («объединение»). Элемент $c = a \wedge b$ – это точная нижняя грань элементов a,b, то есть наибольший элемент решетки, удовлетворяющий неравенствам $c \leq a$ и $c \leq b$. Соответственно $d = a \vee b$ – точная верхняя грань a,b, то есть наименьший элемент решетки, для которого выполнено $a \leq d$ и $b \leq d$.

Решетка $\mathbb{F}$ называется ограниченной, если она содержит общие нижнюю и верхнюю грани – такие два элемента O,I, что $O \leq a \leq I$ для любого $a \in \mathbb{F}$.

В настоящей работе в качестве основы LP-структур используется булеан – ограниченная решетка с семантикой подмножеств. В этой связи вместо символов $\leq$, ^{3}, $\wedge$ и $\vee$ используются знаки теоретико-множественных операций $\in$, $\subseteq$, $\cap$ и $\cup$, а элементы решетки обозначаются большими буквами.

Точкой ограниченной (снизу) решетки $\mathbb{F}$ называется любой минимальный элемент ее подмножества $\mathbb{F} \setminus O$. Точки будем обозначать малыми буквами. Решетка называется точечной, если каждый ее элемент может быть представлен объединением точек. Например, в булеане точками являются все подмножества, состоящие ровно из одного элемента универсума. Для точки a элемента A будем использовать обозначение $a \in A$ (наряду с обозначением $a \in A$).

3 LP-структуры и их свойства

В данном разделе вводятся понятия LP-структуры, уравнения в LP-структуре, а также формулируются связанные с ними результаты, которые ранее были доказаны в работах [2-3].

Бинарное отношение R на решетке $\mathbb{F}$ называется дистрибутивным, если из $(A,B_1),(A,B_2) \in R$ следует $(A,B_1 \cup B_2) \in R$ ($A,B_1,B_2 \in \mathbb{F}$). Отношение называется логическим, если оно содержит $\subseteq$, дистрибутивно и транзитивно. Логическим замыканием произвольного бинарного отношения R называется наименьшее логическое отношение, содержащее R.

По LP-структурой будем подразумевать решетку с определенным на ней логическим отношением.

Пусть задано некоторое отношение R на решетке $\mathbb{F}$ и выбраны два элемента $A,B \in \mathbb{F}$. Пусть существует конечное множество пар $\{(A_i,B_i) \mid i = 1,\ldots,p\}$, где для любого i справедливо $A_i = B_i$, $A_i \in \mathbb{F}$ либо $(A_i,B_i) \in R$. Если при этом $A \subseteq A_i, 1 \leq i \leq p$ и $\bigcup_i B_i \subseteq B$, то будем говорить, что упорядоченная пара элементов (A,B) дистрибутивно связана отношением R.

Определение 1. Пусть R – произвольное отношение на $\mathbb{F}$ и выбраны два элемента $A,B \in \mathbb{F}$. Пусть также существует упорядоченный набор элементов $\vec{r}_{AB} = (B_1,\ldots,B_m)$

($B_1,\dots,B_m \in \mathbb{F}$, $0 < m < \infty$), такой, что в последовательности $(B_0,B_1),(B_1,B_2),\dots,(B_m,B_{m+1})$, где $B_0 = A$, $B_{m+1} = B$, каждая пара дистрибутивно связана отношением R (если дистрибутивно связана сама пара (A,B), полагаем $m = 0$). Тогда указанный набор $\vec{r}_{AB}$ будем называть логической цепочкой (длины m), соединяющей A и B. Пару (A,B) при этом будем называть логически связанной отношением R и обозначать этот факт $A \,R\, B$.

Теорема 1. Для произвольного отношения R на решетке $\mathbb{F}$ логическое замыкание существует и представляет собой множество R всех упорядоченных пар $A,B \in \mathbb{F}$, логически связанных отношением R.

Два отношения R_1 и R_2, заданные на общей решетке $\mathbb{F}$, называются эквивалентными, если их логические замыкания совпадают. Для таких отношений используется обозначение $R_1 \sim R_2$.

Теорема 2. Пусть R_1,R_2,R – отношения на $\mathbb{F}$. Если при этом $R_1 \sim R_2$, то $R_1 \cup R \sim R_2 \cup R$.

Теорема 2 обосновывает принцип локальности эквивалентных преобразований: заменяя некоторое подмножество данного отношения на эквивалентное множество, получаем общее эквивалентное отношение. Этот результат может быть применен для приведения отношений к некоторому каноническому виду.

Теорема 3. Если в отношении R на решетке $\mathbb{F}$ каждую пару вида (A,B), где $B = \bigcup_i B_i$ – конечное объединение элементов $B_i \in \mathbb{F}$ $(1 \le i \le n)$, заменить совокупностью пар $\{(A,B_1),\dots,(A,B_n)\}$, то полученное отношение R' будет эквивалентно исходному R.

Отношение R на точечной решетке $\mathbb{F}$ называется каноническим, если оно задано множеством пар вида (A,a), где $A \in \mathbb{F}$, a – точка в $\mathbb{F}$. Согласно теореме 3, для любого отношения на точечной решетке существует эквивалентное ему каноническое отношение.

Также имеются результаты о существовании и способе построения логической редукции отношений в LP-структурах. Логической редукцией отношения R на решетке $\mathbb{F}$ называется любое минимальное отношение, эквивалентное R.

Для произвольного бинарного отношения R на решетке $\mathbb{F}$ рассмотрим отношение $\tilde{R}$, построенное по R последовательным выполнением следующих шагов:

- добавить к R все пары вида (A,A), где $A \in \mathbb{F}$ (рефлексивные пары), и обозначить новое отношение R_1;
- добавить к R_1 всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i,B_i) $(i = 1,\dots,n)$ принадлежат R_1;
- объединить полученное отношение с отношением включения $\subseteq$.

Для произвольного отношения R рассмотрим отношение $\underset{\sim}{R}$, построенное по данному R последовательным выполнением шагов, обратных построению $\tilde{R}$, а именно:

- исключить из R содержащиеся в нем пары вида $A \supseteq B$ и обозначить новое отношение R_{-1};
- исключить из R_{-1} всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i,B_i) $(i = 1,\dots,n)$ принадлежат R_{-1} и не совпадают с (A,B);
- исключить из полученного отношения все рефлексивные пары.

Можно показать, что оба отношения $\tilde{R}$ и $\underset{\sim}{R}$ эквивалентны R.

Теорема 4. Пусть для заданного на решетке бинарного отношения R построено соответствующее отношение $\tilde{R}$. Тогда, если для $\tilde{R}$ существует транзитивная редукция R^0, то соответствующее ей отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию исходного отношения R.

Теорема 4 может быть применена для минимизации баз знаний путем устранения избыточности.

Далее вводятся связанные с LP-структурами уравнения.

Пусть дано некоторое бинарное отношение R на решетке $\mathbb{F}$ и имеет место $A\,R\,B$. Тогда в смысле общей теории B называется образом A, а A – прообразом B при отношении R. Поскольку отношение представляет собой произвольное соответствие, то каждый элемент из $\mathbb{F}$ может иметь много образов и прообразов. Более того, в нашем случае в силу определения логического отношения любой $B_1 \supseteq B$ является образом A и каждый $A_1 \supseteq A$ является прообразом B. Поэтому при изучении образов и прообразов логических отношений необходимо уточнение рассматриваемых понятий.

Для элемента $B \in \mathbb{F}$ минимальным прообразом при отношении R называется такой элемент $A \in \mathbb{F}$, что $A\,R\,B$ и A является минимальным, то есть не содержит никакого другого $A_1 \in \mathbb{F}$, для которого $A_1\,R\,B$. В данном контексте будем использовать обозначение $R(A) = B$.

Определение 2. Точка x решетки $\mathbb{F}$ называется начальной при отношении R, если в R нет ни одной такой пары (A,B), что x содержится в B и не содержится в A. Элемент X точечной решетки $\mathbb{F}$ называется начальным, если все его точки являются начальными (при отношении R). Подмножество $\mathbb{F}_0(R)$ (будем также обозначать $\mathbb{F}_0$, если это не вызовет неоднозначностей) точечной решетки $\mathbb{F}$, состоящее из всех начальных элементов $\mathbb{F}$, называется начальным множеством решетки $\mathbb{F}$ (при отношении R).

Рассмотрим уравнение

$$R(X) = B, \qquad (1)$$

где $B \in \mathbb{F}$ – заданный элемент, $X \in \mathbb{F}$ – неизвестный.

Определение 3. Частным решением X уравнения (1) называется любой минимальный прообраз элемента B, содержащийся в $\mathbb{F}_0$. Приближенным (частным) решением X уравнения (1) называется любой прообраз элемента B, содержащийся в $\mathbb{F}_0$. Общим решением уравнения (1) называется совокупность всех его частных решений $\{X_s\}, s \in S$.

Уравнения вида (1) будем называть продукционно-логическими уравнениями на решетках. Рассмотрим продукционно-логическое уравнение следующего вида:

$$R(X) = B_1 \cup B_2 \qquad (2)$$

Теорема 5. Пусть $\{X_s\}, s \in S_1$ – общее решение уравнения вида (1) с правой частью B_1, а $\{Y_p\}, p \in S_2$ – общее решение уравнения того же вида с правой частью B_2. Тогда общее решение уравнения (2) представляет собой множество всех элементов вида $X_s \cup Y_p$, из которого исключены элементы, содержащие другие элементы этого же множества.

Предположим, что R является каноническим отношением на точечной решетке $\mathbb{F}$, а правая часть B уравнения (1) представляет собой конечное объединение точек. Тогда, в силу теоремы 5, вместо уравнения (1) достаточно рассмотреть уравнения с точками в правой части:

$$R(X) = b \qquad (3)$$

В работе [3] сформулированы и доказаны условия разрешимости уравнения (3), а также приведена оценка количества его решений.

Разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же точечным элементом x_p в качестве правой части. Такое разбиение имеет смысл благодаря тому, что R является каноническим. Обозначим эти подмножества R^p соответственно их элементу $x_p, p \in P$.

Определение 4. Слоем R_t в отношении R называется подмножество R, образованное упорядоченными парами, взятыми по одной из каждого непустого R^p, $p \in P$. Два слоя, отличающиеся хотя бы одной парой, считаются различными.

Теорема 6. Для нахождения общего решения уравнения (3) достаточно найти частное решение X_t в каждом слое R_t, если оно существует. Далее из полученного множества решений необходимо исключить элементы, содержащие другие элементы этого же множества.

В отдельном слое R_t нахождение решения уравнения вида (3) эквивалентно перечислению множества входных вершин некоторого (соответствующего слою) графа $G_{R_t b}$. Построим такой

граф. Каждой точке, участвующей в отношении R, сопоставим вершину графа. Далее для каждой пары (A,a) рассматриваемого слоя R_t построим дуги, ведущие из всех вершин, соответствующих точкам A, в вершину, соответствующую данной a. В полученном графе G_{R_t} выберем вершину b, сопоставленную правой части уравнения (3). Рассмотрим подграф $G_{R_t,b} \subseteq G_{R_t}$, содержащий все вершины, из которых достижима вершина b (включая саму b) и все дуги, соединяющие такие вершины.

Теорема 7. Уравнение (3) имеет не более одного решения в каждом слое $R_t, t \in T$. Если граф $G_{R_t,b}$ не имеет циклов, то единственное решение уравнения состоит из всех точек, соответствующих входным вершинам графа. Если $G_{R_t,b}$ имеет хотя бы один цикл, то в данном слое уравнение решений не имеет.

4 Общие принципы реализации

При реализации LP-структур одним из важных вопросов является способ представления бинарных отношений на решетках. Сразу заметим, что, например, для булеана при количестве его точек n общее число элементов составит 2^n. Этот факт означает, что в общем случае способ хранения отношения на решетке в виде статической матрицы (смежности или инциденций) не является эффективным. Такой подход сопряжен с использованием разреженных матриц очень больших размеров, стандартное хранение которых в памяти компьютера не только неэффективно, но и вряд ли реализуемо на практике. Таким образом, бинарное отношение на решетке предлагается представлять в виде множества (динамического) хранимых в памяти пар ее элементов. Последующее изложение также подчеркивает основную особенность реализации LP-структур – потребность снижения расхода памяти, иногда в ущерб быстродействию.

Далее с позиций практической реализации рассмотрим задачу нахождения логической редукции LP-структуры. В приложении к экспертным продукционным системам это соответствует получению минимальной базы знаний, эквивалентной исходной базе.

Согласно п.3, для бинарного отношения R на решетке $\mathbb{F}$ логическая редукция $\tilde{R}^0$ может быть получена последовательным выполнением следующих шагов:

1) добавить к R все пары вида (A,A), где $A \in \mathbb{F}$ (рефлексивные пары), и обозначить новое отношение R_1;
2) добавить к R_1 всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i,B_i) ($i = 1,\ldots,n$) принадлежат R_1;
3) объединить полученное отношение с отношением включения $\subseteq$ и обозначить новое отношение $\tilde{R}$;
4) построить транзитивную редукцию R^0 отношения $\tilde{R}$;
5) исключить из $\tilde{R}$ содержащиеся в нем пары вида $A \subseteq B$ и обозначить новое отношение R_{-1};
6) исключить из R_{-1} всевозможные пары (A,B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i,B_i) ($i = 1,\ldots,n$) принадлежат R_{-1} и не совпадают с (A,B);
7) исключить из полученного отношения все рефлексивные пары.

В силу тех же соображений экономии памяти условимся, что физическое добавление к множеству новых пар следует производить лишь в случае необходимости. В частности, нет смысла хранить рефлексивные (шаг 1) или «подчиненные» (шаг 3) пары, особенно если способ кодирования решетки будет допускать эффективное вычисление частичного порядка.

Непосредственная реализация шага 2 также потребовала бы чрезмерного расхода памяти, соизмеримого с построением булеана на универсуме R. Кроме того, в дальнейших вычислениях из огромного количества добавленных таким образом пар практически использовалась бы лишь

незначительная часть. Итак, мы приходим к необходимости динамического построения востребованных в дальнейшем пар множества $\tilde{R}$.

Обсудим также вопрос построения транзитивной редукции (шаг 4) отношения $\tilde{R}$. Как показано в [4], эта задача вычислительно эквивалентна задаче нахождения транзитивного замыкания. Её решение, например, при применении известного алгоритма Уоршала, составило бы $O(N^3)$ операций. Однако в рассматриваемом случае величина N заменяется выражением 2^N и, таким образом, нахождение транзитивной редукции посредством замыкания не представляется эффективным. В качестве практически реализуемого варианта остается исследование на транзитивную избыточность множества пар как такового. Таким образом, в рассматриваемой реализации предлагается строить транзитивную редукцию отношения $\tilde{R}$ путем исключения пар, связанных транзитивными цепочками.

Теперь с точки зрения практической реализации рассмотрим вопросы решения продукционно-логических уравнений вида (3). Как показывают теоретические оценки, применение данного аппарата позволяет уменьшить количество медленных запросов, направляемых базе данных или интерактивному пользователю при обратном выводе.

Итак, согласно п.3, для решения уравнения исходное каноническое отношение R представляется в виде слоев, каждый из которых позволяет получить не более одного решения. Слой содержит максимально возможный набор пар исходного отношения с уникальными правыми частями. Два слоя различаются хотя бы одной парой. Реализация всех слоев привела бы к избыточному хранению пар, так как слои могут иметь большие пересечения. Для разрешения этого вопроса предлагается поступить следующим образом.

Как и в п.3, вначале разобьем R на непересекающиеся подмножества, каждое из которых образовано парами вида (A, x_p) с одним и тем же точечным элементом x_p в качестве правой части. Обозначим эти подмножества R^p соответственно их элементу $x_p, p \in P$. При реализации канонического отношения R для каждого x_p достаточно хранить лишь совокупность левых частей пар с правой частью x_p. Каждому из подмножеств R^p сопоставим итератор – индексную переменную j_p, способную перебирать собственное подмножество, останавливаясь на каждой паре ровно один раз. Таким образом, любой слой R_t в отношении R получается соответствующим набором значений итераторов $\{j_p\}$.

Согласно теореме 7, нахождение решения в отдельном слое сводится к получению списка начальных вершин графа, из которых достижима данная вершина (она соответствует точке правой части уравнения), при фиксированном наборе значений итераторов.

Работа с различными слоями может быть организована независимо и даже параллельно. Такой подход в определенном смысле максимально экономит память, но может приводить к многократному повторению вычислений. Причина в том, что, как уже отмечалось ранее, два слоя могут иметь много общих пар. Другой вариант – запоминать результаты работы в слое, чтобы ими при необходимости можно было воспользоваться и в другом слое, пересекающемся с данным. Этот способ несколько противоречит выбранной стратегической линии на экономию памяти, однако не приводит к её перегрузкам.

5 Кодирование LP-структур

Очевидно, предложенный выше способ представления в памяти LP-структур непосредственно связан со способами кодирования решеток. Последнему вопросу посвящены работы [5-7]. Из замечаний предыдущего пункта следует, что реализация LP-структур должна максимально экономить память и предоставлять быстрые «решеточные» операции (объединение, пересечение, частичный порядок). Этому требованию во многом удовлетворяет способ кодирования решеток битовыми векторами.

Как указано, например, в [6], конечная булева решетка может быть представлена множеством битовых векторов (то есть векторов с двоичными значениями компонент) одинаковой размерности N, где N – число точек решетки. Каждому элементу a булевой решетки сопоставляется единственный битовый вектор $BitVect(a)$. Каждой точке при этом соответствует

вектор с одной единицей в соответствующей позиции и остальными нулями. Вычисление «решеточных» операций сводится к вычислению побитовых логических операций сложения (|) и умножения (&) над компонентами векторов, а именно:

$$a \dot{\cup} b = BitVect(a) \mid BitVect(b);$$

$$a \dot{\cap} b = BitVect(a) \,\&\, BitVect(b);$$

$$a \leq b \Leftrightarrow BitVect(a) \,\&\, BitVect(b) = BitVect(a);$$

$$a \leq b \Leftrightarrow BitVect(a) \mid BitVect(b) = BitVect(b).$$

Как известно, побитовые логические операции относятся к низкоуровневым и, соответственно, наиболее быстрым двуместным операциям в современных компьютерах. Конечно, теоретически с точки зрения анализа сложности алгоритмов нельзя считать, что операция над двумя векторами будут выполняться за константное время, поскольку разрядность компьютера (например, 32 или 64) вполне может оказаться недостаточной для представления битового вектора единственным словом памяти. Однако величина сложности операции $N / 32$ или $N / 64$ представляется приемлемой при общем количестве элементов решетки 2^N .

Поскольку описываемая здесь идеология реализации LP-структур использует конечные булевы решетки, то наши потребности кодирования решеток исчерпываются указанным способом. Однако при необходимости реализации общих видов решеток можно обратиться, например, к исследованию [5], где описываются возможности кодирования битовыми векторами для произвольных решеток.

Еще один вопрос реализации LP-структур связан со способом хранения вторичных бинарных отношений на решетках. В соответствии с замечаниями п.4 было принято представление бинарного отношения в виде множества пар. Здесь следует уточнить, что эффективность операций доступа к множеству (поиск, включение и исключение элементов) на практике обеспечивается, например, возможностью его хранения в виде сбалансированного бинарного дерева (так, в частности, реализуются множества в STL – стандартной библиотеке C++ [8]). Такой подход требует сопоставления элементам хранимого в памяти множества уникальных вполне упорядоченных ключей. Ключ по возможности должен помещаться в слове (или двойном слове) компьютера, тогда при доступе к множеству операция сравнения ключей будет выполняться за константное время. Таким образом, приходим к механизму хранения пары элементов решетки как пары ссылок на соответствующие им битовые векторы. Эти ссылки могут содержаться в смежных частях двойного целого числа, а данное число при этом будет использоваться в качестве упомянутого выше ключа.

Для хранения канонических отношений на решетках при реализации ряда алгоритмов предлагается еще одно представление – в виде вектора множеств. Для канонического отношения можно построить массив, компонентами которого являются подмножества элементов решетки. Индексом этого массива служит номер точки решетки (например, номер единственной ненулевой компоненты в соответствующей точке битового вектора), а содержимым компоненты массива – множество элементов решетки, составляющих с данной точкой пары в качестве их левых частей.

Указанное представление весьма эффективно как с позиции использования памяти, так как хранятся лишь левые части пар, так и быстродействия – доступ к ним осуществляется по индексу в массиве. Этот способ может применяться для реализации в LP-структуре алгоритмов обратного вывода при неизменном множества правил.

Представление отношения на решетке в виде указанного вектора множеств в определенной мере можно считать аналогом «константной» сети Rete [9]. Такая сеть используется для ускорения прямого (или смешанного) вывода в продукционных экспертных системах. При прямом выводе на основе содержимого рабочей памяти системы осуществляется «вход в продукции» со стороны их левых частей. По этой причине для элементов текущей рабочей памяти Rete содержит ссылки на левые части продукций, которым эти элементы удовлетворяют. Для каждой левой части продукции также требуется хранение и соответствующей ей правой части. В результате сеть Rete представляет довольно объемную надстройку над имеющимся множеством продукционных правил, хотя и весьма эффективно организованную.

Предлагаемая конструкция (вектор множеств) предназначена для повышения эффективности исключительно обратного вывода. Она более экономична по памяти, поскольку не предусматривает дублирования информации. Если сеть Rete реализует дополнительную надстройку над множеством продукций, то вектор множеств *заменяет* множество продукций. Отмеченный результат объясняется особенностью архитектуры стандартных продукционных систем – «неравноправием» левых и правых частей продукций: согласно [2] для отношения на точечной решетке всегда существует эквивалентное каноническое отношение (с элементарными правыми частями пар), в то время как левые части продукций в общем случае невозможно эквивалентным образом «урезать» до отдельных точек.

Рассмотрим далее некоторые практические соображения по непосредственной реализации LP-структур. Для хранения элементов решетки предлагается использовать гибкую конструкцию битового вектора, при которой его длина может настраиваться «клиентской» программой. Размерность такого вектора задается двумя параметрами – длиной в байтах одного кластера (*eItemSize*) и количеством этих кластеров (*eLengthItems*).

Величина *eItemSize* определяет размер кластера – части вектора, которая обрабатывается единственной логической операцией компьютерного процессора. Эта величина может храниться, например, как статическое константное поле объектно-ориентированного класса LPStructure, и меняться с его перекомпиляцией. Выбором *eItemSize* = 1 можно структурировать битовый вектор в виде последовательности байтов, что приведет к экономии памяти и сделает структуру вектора более прозрачной с точки зрения понимания алгоритмов. Если же положить, например, *eItemSize* = 4, то можно повысить быстродействие операций над векторами, так как многие циклы в программе станут в 4 раза короче, а 32-разрядные операции окажутся более эффективными для ЭВМ с соответствующей длиной слова памяти.

Количество кластеров битового вектора может храниться в целочисленном поле *eLengthItems*. Конструктор класса LPStructure получит в качестве параметра соответствующую целую величину для настройки данного поля. Это обстоятельство предоставит возможность «клиентской» программе при создании LP-структуры динамически настраивать длину битовых векторов, исходя из размера решетки и, соответственно, объема имеющейся базы знаний.

В классе LPStructure могут быть определены типы *Elem* и *Pair* путем переименования соответственно типов обычного и двойного целых чисел. Тип *Elem* предполагает хранение адреса элемента решетки, а именно – адреса соответствующего элементу битового вектора. Тип *Pair* при этом содержит смежную пару величин *Elem*, то есть предназначается для хранения пары элементов (точнее, их адресов) из некоторого бинарного отношения на решетке. Конечно, типы *Elem* и *Pair* можно реализовать в виде объектно-ориентированных классов с перегруженными операторами (|, &, <=), соответствующими операциям на решетке. Однако решение использовать простейшие типы окажется более последовательным в плане принятой выше стратегии экономии памяти.

При использовании для реализации LP-структур библиотеки STL могут широко применяться ее эффективные параметризованные контейнерные классы *vector* и *set*, а также их комбинации. В частности, для представления множеств элементов решетки полезным окажется тип *ElemContainer*, определяемый как *set*<*Elem*>. Для хранения задаваемых на решетке бинарных отношений общего вида можно определить тип *PairContainer* как *set*<*Pair*>. Некоторыми алгоритмами в качестве промежуточных данных могут использоваться векторы пар – *PairVector* (*vector*<*Pair*>). Каноническое бинарное отношение, в соответствии с изложенными выше соображениями, представимо вектором множеств *ElemContainerVector* (*vector*<*ElemContainer*>).

Итак, рассмотрены основные принципы устройства и компьютерной реализации LP-структур. Эти принципы легли в основу выполненной автором разработки объектно-ориентированного класса LPStructure, который может быть применен для поддержки средств автоматизированного управления знаниями продукционного типа.

Литература

[1] Sowa J. F. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA (1999).

[2] Махортов С.Д. Логические отношения на решетках // Вестник ВГУ. Серия физика, математика. – Воронеж, 2003, 2. – С. 203-209.

[3] Махортов С.Д. Логические уравнения на решетках // Вестник ВГУ. Серия физика, математика. – Воронеж, 2004, 2. – С. 170-178.

[4] Aho A.V., Garey M.R., Ulman J.D. The transitive reduction of a directed graph. SIAM J. Computing 1:2, pp. 131-137 (1972).

[5] Halib M., Nourine L. Bit-vector encoding for partially ordered sets // Lect. Notes Comput. Sci. – 1994, v. 831. – Pp. 1-12.

[6] Sowa J.F. Conceptual Structures: Information Processing in Mind and Machine. Reading, MA: Addison-Wesley, 1984.

[7] Кузнецов С.О. Быстрый алгоритм построения всех пересечений объектов из конечной полурешетки // НТИ. Сер.2. – 1993, № 1. – С. 17-20.

[8] Джосьютис Н. С++ Стандартная библиотека. Для профессионалов. / Пер. с англ. – СПб.: Питер, 2004. – 730с.

[9] Forgy C. L. Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, 19(1):17-37, 1982.