УДК 004.021, 004.272.43

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ОПЕРАЦИЙ НАД ПРИБЛИЖЁННЫМИ МНОЖЕСТВАМИ ДЛЯ АНАЛИЗА ФОРМАЛЬНЫХ ПОНЯТИЙ В ОДНОРОДНЫХ СЕМАНТИЧЕСКИХ СЕТЯХ

В.П. Ивашенко (*ivashenko@bsuir.by*)
С.В. Синцов (*ssivikt@gmail.com*)
Белорусский государственный университет
информатики и радиоэлектроники

В статье рассматривается разработка программных средств автоматизации построения онтологий, представляемых в виде однородных семантических сетей с базовой теоретико-множественной интерпретацией. Приводятся алгоритмы базовых операций обработки множеств, которые допускают реализацию на параллельной вычислительной архитектуре и описывается метод построения графа онтологии по формальному контексту, заданному на приближённых множествах.

Ключевые слова: алгоритм, приближённые множества, анализ формальных понятий, онтология, однородная семантическая сеть, параллельная обработка знаний.

Введение

Наличие данных большого объёма в информационных системах требует эффективных механизмов их обработки, применение которых в сочетании с тенденциями автоматизации задач анализа данных, ориентированных на применение моделей и методов искусственного интеллекта, поддерживающих обработку семантически структурированных данных, позволяет сократить затраты на подготовку и интерпретацию результатов анализа пользователем. В работе рассматриваются подход и алгоритмы, способные обеспечить автоматизацию процесса построения онтологических структур и поддержку создания компонентов [Голенков и др., 2015] при проектировании баз знаний, использующих однородные семантические сети [Голенков и др., 2012] для представления информации. Актуальность работы обусловлена потребностью в интеллектуальной обработ-

ке больших объёмов данных и знаний, возрастающими требованиями ко временным затратам обработки данных и знаний, а также развитием семантических моделей представления знаний. Особенностью рассматриваемой задачи является поддержка работы в условиях неполноты информации, что достигается за счёт применения модели унифицированного семантического представления знаний [Ивашенко, 2015] и сочетания средств анализа формальных понятий (АФП) и аппарата приближённых множеств (ПМ) [Poelmans et al., 2014]. Результат решения задачи АФП не только позволяет выявлять порядок организации знаний (отношения между понятиями, визуализируя эти отношения в виде диаграмм Хассе), но и может служить основой для построения гипотез и формулировки теорем о предметной области. Особенностью работы является также то, что алгоритмы в рамках рассматриваемого подхода предлагаются как элемент технологии, ориентированной на создание интеллектуальных систем, и выстраиваются в согласии с соответствующими реализационными и архитектурными принципами [Ивашенко и др., 2016].

Целью работы является разработка программных средств автоматизации построения онтологий, представляемых в виде однородных семантических сетей с базовой теоретико-множественной интерпретацией [Голенков и др., 2012]. Для этого 1) разрабатываются алгоритмы базовых операций обработки множеств, которые допускают параллельную реализацию на гетерогенной вычислительной архитектуре; 2) описывается метод построения графа родовидовой онтологии (таксономии) по формальному контексту, заданному на приближённых множествах.

Система, основанная на знаниях [Гаврилова и др., 2000], состоит из базы знаний (БЗ), машины обработки знаний (МОЗ) и пользовательского интерфейса (ПИ). Компоненты БЗ и МОЗ относятся ко внутренней части системы, основанной на знаниях, которая скрыта от пользователя; ПИ относится ко внешней части, доступной пользователю.

В системе, основанной на знаниях, можно выделить три уровня (рис. 1): 1) уровень управления устройствами; 2) уровень управления данными; 3) уровень управления знаниями. Управление знаниями, представленными в виде однородных семантических сетей с базовой теоретико-множественной интерпретацией, требует решения задачи представления и обработки множеств, что включает в себя реализацию различных структур данных: динамические массивы, списки, деревья и т.д. [Кнут, 2014], а это, в свою очередь, требует реализации базовых операций динамического управления памятью [Кнут, 2014] на первом уровне: выделение, высвобождение и перевыделение участков памяти.

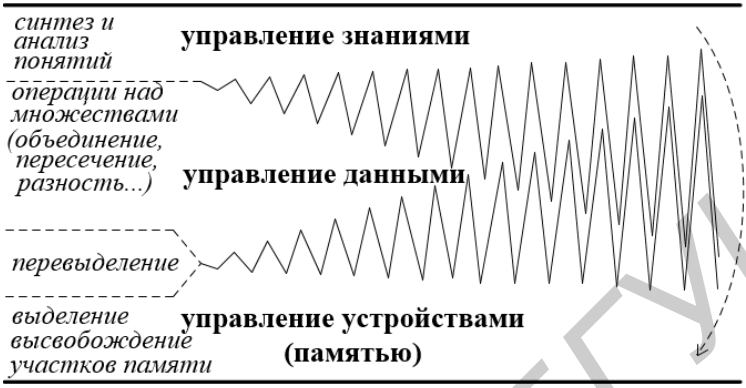

Рис. 1. Схема уровней системы, основанной на знаниях

Решение задачи АФП для ПМ в однородных семантических сетях поддерживается на нижних уровнях и полностью обеспечивается на уровне три.

Производительность и масштабирование использующихся для АФП алгоритмов зависят, в частности, от вычислительной программно-аппаратной платформы, которая выбрана основой для их реализации. При обработке данных большого объёма в качестве таких платформ используются высокопроизводительные параллельные и распределённые гетерогенные вычислительные системы классов ОКМД или МКМД.

1. Описание алгоритмов

В соответствии со схемой (рис. 1), на первом уровне системы, основанной на знаниях, реализована система динамического управления памятью, гарантирующая время порядка $\ln^2(m)$ (m – размер памяти) для операций выделения и высвобождения участков памяти [Ивашенко, 2012], а также – константное среднее амортизационное время для операции перевыделения, в случае, если число добавленных в связный участок элементов данных превышает фактическое число произведённых перевыделений исходного связного участка памяти.

На рис. 2 приведён график логарифмированного отношения экспериментальных результатов времени (t_{em}) работы операций перевыделения нижнего уровня ко времени (t_{tc}) работы операции, реализованных в tcmalloc [TCMalloc, 2016].

На втором уровне реализуются используемые при построении графа родовидовой онтологии операции пересечения и объединения (мульти)множеств.

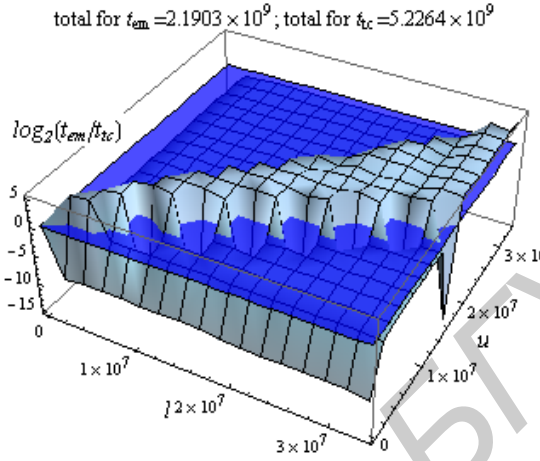

Рис. 2. График логарифма отношения времен исполнения операций перевыделения участка размером *u* ячеек из участка исходного размера *l* ячеек с размером 32 бита

Алгоритм 1. Операция *IntersectMultisets*(⟨*A*, *B*⟩) пересечения мультимножеств.

1. Выполнить поиск минимального и максимального индексов вхождения каждого элемента множества *A* во множество *B*:
   ⟨*L*, *U*⟩ ← *RangeBinarySearch*(⟨*A*, *B*⟩).
2. Отметить в массиве *C* минимальные индексы равных элементов массива *A*:
   $$C[i] \leftarrow \begin{cases} i \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] < A[i] \\ 0 \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] = A[i] \end{cases}.$$
3. Вычислить максимумы *D* ← *MaxScatter*(*C*):
   k ← ((~ 0) >> (*clz*(*Length*(*A*))+1))+1
   пока (*k* > 0):
   если (*i* ≥ *k*), *то* *D*[*i*] ← max({*D*[*i*]}∪{*D*[*i*–*k*]})
   k >>= 1
4. Вычислить
   $$E[i] \leftarrow \begin{cases} 1 \,|\, i - D[i] < U[i] - L[i] \\ 0 \,|\, i - D[i] \geq U[i] - L[i] \end{cases}.$$
5. Вычислить массив *F* префиксных сумм массива *E*: *F*[*i*] ← *E*[*i*]+*PrefixSum*(*E*)[*i*].

6. Вычислить: *если* ($E[i]$=1), *то* $R[F[i]-1] \leftarrow A[i]$.
7. Возвратить $\langle R, F[Length(A)-1]\rangle$.

Алгоритм 2. Операция *UniteMultisets*($\langle A, B\rangle$) объединения мультимножеств.

1. Выполнить поиск минимального и максимального индексов вхождения каждого элемента множества A во множество B:
   $\langle L, U\rangle \leftarrow RangeBinarySearch(\langle A, B\rangle)$.
2. Отметить в массиве C минимальные индексы равных элементов массива A:
$$C[i] \leftarrow \begin{cases} i \,\middle|\, A[\max(\{0\} \cup \{i-1\})] < A[i] \\ 0 \,\middle|\, A[\max(\{0\} \cup \{i-1\})] = A[i] \end{cases}.$$
3. Вычислить максимумы $D \leftarrow MaxScatter(C)$:
   $k \leftarrow ((\sim 0) >> (clz(Length(A))+1))+1$
   пока ($k > 0$):
    если ($i \geq k$), *то* $D[i] \leftarrow \max(\{D[i]\} \cup \{D[i-k]\})$
    $k >>= 1$
4. Вычислить
$$G[i] \leftarrow \begin{cases} 0 \,\middle|\, i - D[i] < U[i] - L[i] \\ 1 \,\middle|\, i - D[i] \geq U[i] - L[i] \end{cases}.$$
5. Выполнить для каждого элемента массива D:
   $E[i] \leftarrow i - D[i] - U[i] + L[i]$.
6. Вычислить массив H префиксных сумм массива G:
   $H \leftarrow PrefixSum(G)$.
7. Вычислить:
   для i от 0 до Length(A) + Length(B) − 1:
    $R[i] \leftarrow 1$
8. Вычислить для каждого элемента массива A:
   если ($A[i] <> B[L[i] + E[i]]$), *то* $R[L[i] + E[i] + H[i]] \leftarrow 0$
9. Вычислить массив T префиксных сумм массива R:
   $T[i] \leftarrow R[i]+PrefixSum(R)[i]$; $T[-1] \leftarrow 0$.
10. Выполнить для каждого элемента ($i \geq 0$) массива T:
    если (($T[i] \leq Length(B)$),($T[i-1] <> T[i]$)), *то* $R[i] \leftarrow B[T[i]-1]$.
    После этого массив R содержит все элементы массива B.
11. Выполнить для каждого элемента ($i \geq 0$) массива R:
    если ($T[i-1]=T[i]$), *то* $R[L[i] + E[i] + H[i]] \leftarrow A[i]$.
12. После этого массив R содержит и элементы массива A.

Возвратить $\langle R, H[Length(A)-1] + T[Length(A) + Length(B)-1]\rangle$.

Для операций уровня управления данными по алгоритмам 1 и 2 временная сложность в соответствии с теоретической оценкой ожидается не более чем $O(\ln(n)*n/p+\ln^2(m))$ (n – суммарная мощность множеств, p – количество процессоров). Для операций третьего уровня – теоретическая оценка даёт выражение $O(n^2*\ln^2(n)*2^{2n}/p+\ln^2(m))$.

Модель АФП для ПМ основана на модели многозначного анализа формальных понятий [Stumme et al., 2001] и является её частным случаем. Пусть G – множество объектов предметной области, M – множество свойств, которыми могут обладать объекты G, а $V=\{0, ½, 1\}$ – множество из трёх рациональных чисел. Приближённый формальный контекст F_R определяется как четвёрка $\langle G, M, I_R, V\rangle$, где $I_R=V^{G\times M}$ – характеристический предикат. Таким образом, для любой пары $\langle x, y\rangle$, где $x\in G$ и $y\in M$, справедливо одно из следующих утверждений:

1. Если $I_R(\langle x, y\rangle)=0$, то объект x не обладает свойством y.
2. Если $I_R(\langle x, y\rangle)=1$, то объект x обладает свойством y.
3. Если $I_R(\langle x, y\rangle)=½$, то неизвестно обладает ли объект x свойством y.

Пусть $X\subseteq G$ ($Y\subseteq M$), тогда нижняя $L(X)$ ($\bar{L}(Y)$) и верхняя $U(X)$ ($\bar{U}(Y)$) аппроксимации множества атрибутов (объектов) для множества объектов (атрибутов) задаются как:

$L(X)=\{y|\forall x((x\in X)\rightarrow(I_R(\langle x,y\rangle)>0))\}$, $U(X)=\{y|\forall x((x\in X)\rightarrow(I_R(\langle x,y\rangle)=1))\}$,

$\bar{L}(Y)=\{x|\forall y((y\in Y)\rightarrow(I_R(\langle x,y\rangle)>0))\}$, $\bar{U}(Y)=\{x|\forall y((y\in Y)\rightarrow(I_R(\langle x,y\rangle)=1))\}$.

Построение графа родовидовой онтологии (таксономии) по заданному F_R сводится к трём следующим шагам:

1. Сформировать наборы экстенсионалов (extent) [Poelmans et al., 2014].
2. Сформировать наборы интенсионалов (intent) [Poelmans et al., 2014].
3. Построить граф родовидовой онтологии (таксономии).

Наборы экстенсионалов и интенсионалов рассчитываются с помощью операций объединения и пересечения множеств, исходя из того, что для каждого объекта (атрибута) известно множество атрибутов (объектов), которыми он обладает, и множество атрибутов (объектов), которыми он не обладает. Множество атрибутов (объектов), которыми обладает объединение экстенсионалов (интенсионалов) рассчитывается как пересечение множеств атрибутов (объектов), которыми обладают объекты (атрибуты) каждого экстенсионала (интенсионала), а множество объектов (атрибутов), которыми объединение экстенсионалов (интенсионалов) не обладает, рассчитывается как объединение множеств атрибутов (объектов), объекты (атрибуты) каждого экстенсионала (интенсионала) которыми не обладают.

Правило 1. Если для любого атрибута (объекта) известно, что все объекты (атрибуты) экстенсионала (интенсионала) совместно обладают им либо – нет, то среди равных по интенсионалу (экстенсионалу) экстенсио-

налов (интенсионалов) выбирается единственный наиболее мощный экстенсионал (интенсионал).

Правило 2. Если для каких-либо атрибутов (объектов) какого-либо экстенсионала (интенсионала) ровно для одного из его объектов (атрибутов) неизвестно обладает ли он ими, тогда как остальные объекты (атрибуты) совместно все в каждом из любых более мощных расширяющих исходный экстенсионалов (интенсионалов) обладают этими какими-либо атрибутами (объектами), то последние, более мощные, экстенсионалы (интенсионалы) оставляются, а первый исключается из рассмотрения.

наборы	{a}	{b}	{c}	{d}	{a,b,d}	{a,d}	{b,c}	{b,d}	{c,d}	{a,c,d}	{b,c,d}	{a,b,c,d}
{1}	0	0	1	0	0	0	0	0	0	0	0	0
{2}	½	0	½	1	0	½	0	0	½	½	0	0
{3}	0	½	1	½	0	0	½	½	½	0	½	0
{4}	1	1	0	1	1	1	0	1	0	0	0	0
{1,2}	0	0	½	0								
{1,3}	0	0	1	0								
{1,2,3,4}	0	0	0	0								
{2,3}	0	0	½	½								
{2,4}	½	0	0	1								
{3,4}	0	½	0	½								
{1,2,3}	0	0	½	0								
{2,3,4}	0	0	0	½								

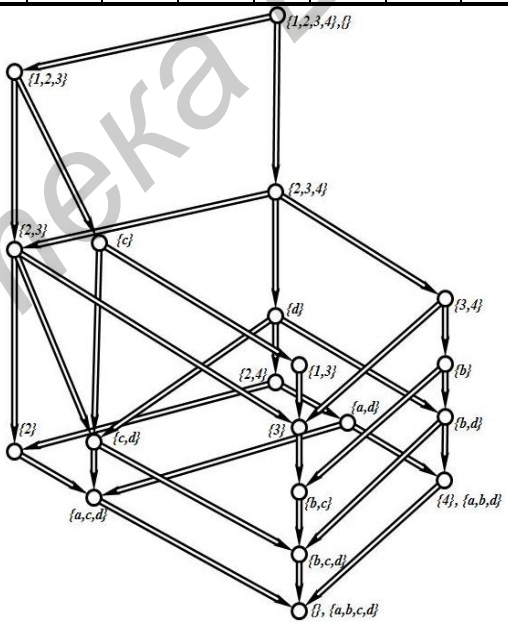

Рис. 3. Наборы экстенсионалов (интенсионалов) и граф онтологии

Правило 3. Никаких других исключений экстенсионалов (интенсионалов), кроме как по первым двум правилам, не осуществляется и рассмат-

риваются все остальные экстенсионалы (интенсионалы), которые можно получить путём объединения уже рассмотренных.

После того, как сформированы наборы экстенсионалов (интенсионалов), осуществляется построение онтологии, которая включает в себя онтологию экстенсионалов и интенсионалов (решётка). Граф онтологии формируется путём сопоставления каждому экстенсионалу (интенсионалу) из сформированных наборов вершины (SC-элемента) и указания связок отношений их непосредственного включения.

На третьем шаге устанавливаются связки отношения подмножества на экстенсионалах и интенсионалах и устанавливаются связки между $\langle \hat{L}(X), X\rangle$, $\langle X, \hat{U}(X)\rangle$, $\langle \hat{\bar{L}}(Y), Y\rangle$, $\langle Y, \hat{\bar{U}}(Y)\rangle$ в соответствии с выражениями ниже.

$$\hat{L}(X) = P_L\left(L,\bar{U},\bar{L},X\right),\ \hat{U}(X) = P_U\left(U,\bar{L},\bar{U},X\right),$$

$$\hat{\bar{L}}(Y) = P_L\left(\bar{L},U,L,Y\right),\ \hat{\bar{U}}(Y) = P_U\left(\bar{U},L,U,Y\right),$$

$$P_L(\alpha,\beta,\gamma,X) = \begin{cases} \alpha(X) \,\middle|\, \left(\beta(\alpha(X)) = X\right) \vee \left(\gamma(\alpha(X)) \supseteq \beta(\alpha(X))\right) \\ \beta(\alpha(X)) \,\middle|\, \left(\beta(\alpha(X)) \subset X\right) \wedge \left(\gamma(\alpha(X)) \subset \beta(\alpha(X))\right) \end{cases},$$

$$P_U(\alpha,\beta,\gamma,Y) = \begin{cases} \alpha(Y) \,\middle|\, \left(\beta(\alpha(Y)) = Y\right) \vee \left(\gamma(\alpha(Y)) \subseteq \beta(\alpha(Y))\right) \\ \beta(\alpha(Y)) \,\middle|\, \left(\beta(\alpha(Y)) \supset Y\right) \wedge \left(\gamma(\alpha(Y)) \supset \beta(\alpha(Y))\right) \end{cases}.$$

Исключаются из онтологии все построенные связки отношения вида $\langle X, Z\rangle$ такие, что существует Y, что $\langle X, Y\rangle$ и $\langle Y, Z\rangle$ принадлежат отношению. Пример построенной онтологии изображён на рис. 3, где G={1, 2, 3, 4}, M={a, b, c, d}.

Заключение

Разработанные алгоритмы предназначены для обработки знаний в интеллектуальных системах, включая поддержку процессов проектирований баз знаний и построения онтологий, совместимых с технологией OSTIS [Голенков и др., 2015].

Список литературы

[Гаврилова и др., 2000] Гаврилова Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский – СПб.: Питер, 2000.

[Голенков и др., 2012] Голенков В.В., Гулякина Н.А. Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования // OSTIS-2012. 2012.

[Голенков и др., 2015] Голенков, В.В. Н.А. Гулякина Семантическая технология компонентного проектирования систем, управляемых знаниями // OSTIS-2015. 2015.

[Кнут, 2014] Кнут, Д. Искусство программирования: в 3х т. – М.: Вильямс, 2014.

[Ивашенко, 2015] Ивашенко В.П. Модели и алгоритмы интеграции знаний на основе однородных семантических сетей // OSTIS-2015. 2015.

[Ивашенко, 2012] Ивашенко В.П. Алгоритмы полилогарифмической временной и логарифмической пространственной сложности для системы динамического распределения линейно адресуемой памяти с однородным доступом к данным // Карповские научные чтения/ Сб. науч. статей. 2012. Вып. 6. Ч. 1. – Минск, 2012.

[Ивашенко и др., 2016] Ивашенко В.П., Татур М.М. Принципы платформенной независимости и платформенной реализации OSTIS // OSTIS-2016. 2016.

[Poelmans et al., 2014] Poelmans J., Ignatov D., Kuznetsov S. et al. Fuzzy and rough formal concept analysis: a survey // Int J Gen Syst, 2014. Vol. 43. Iss. 2.

[Stumme et al., 2001] Stumme G., Maedche A. FCA-MERGE: bottom-up merging of ontologies // Proc. of the 17th International joint conference on AI: – Seattle, 2001.

[TCMalloc, 2016] TCMalloc: Thread-Caching Malloc. – https://google-perftools. goog-lecode.com/svn/trunk/doc/tcmalloc.html.