

УДК 004.6

Подход к разработке метаграфового исчисления

The approach for metagraph calculus development

Гапанюк Ю.Е. (Gapanuk Yu.E.), доцент кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана, gapyu@bmstu.ru

Аннотация. В статье рассмотрено метаграфовое исчисление, реализующее возможность формального описания и преобразования элементов метаграфовой модели. Приведены основные положения метаграфовой модели. Рассмотрено представление метаграфа как иерархически организованного множества предикатов. Предложены операторы метаграфового исчисления. Рассмотрены примеры описания метаграфов с использованием предложенного исчисления.

Abstract. The article discusses the metagraph calculus, which implements the possibility of formal description and transformation of the elements of the metagraph model. The basic provisions of the metagraph model are given. The representation of the metagraph as a hierarchically organized set of predicates is considered. The operators of metagraph calculus are proposed. The examples of description of metagraphs using the proposed calculus are given.

Ключевые слова: метаграф, метавершина, информационный элемент метаграфа (ИЭМ), предикат, метаграфовое исчисление

Keywords: metagraph, metavertex, metagraph information element (MIE), predicate, metagraph calculus.

Реферат. Предлагаемые ранее подходы позволяли отчасти формализовать метаграфовую модель, но не позволяли полностью формально оперировать с элементами

метаграфовой модели. Основная проблема описания элементов метаграфа на основе ИЭМ состоит в том, что ИЭМ является недостаточно «атомарной» структурой. Предикатное описание позволяет представить метаграфовую модель в текстовом виде, и составные части предикатного описания достаточно «атомарны». Но предикатное описание не дает информации о том, какие действия можно выполнять над элементами метаграфовой модели.

Структурой данных предлагаемого исчисления является вершина-предикат. Все конструкции исчисления представляют собой строки, разделенные на левую и правую части оператором присваивания, в качестве которого используется знак равенства. Оператор конструирования предназначен для создания новых вершин-предикатов на основе существующих. Оператор удаления предназначен для удаления вершин-предикатов нижнего уровня из вершин-предикатов верхнего уровня. Оператор транзитивного удаления также удаляет все элементы, теряющие логическую целостность в результате удаления. Для изменения метаграфовой структуры используется оператор замены.

Таким образом, предложенное метаграфовое исчисление позволяет конструировать и модифицировать элементы метаграфовой модели.

Summary. The previously proposed approaches allowed to formalize the metagraph model in part, but did not allow to operate completely formally with the elements of the metagraph model. The main problem of describing the elements of a metagraph based on MIE is that MIE is not enough “atomic” structure. Predicate description allows one to present a metagraph model in text form, and the constituent parts of the predicate description are sufficiently “atomic”. However, the predicate description does not provide information about what actions can be performed on the elements of the metagraph model.

The data structure of the proposed calculus is a vertex-predicate. All calculus constructs are strings divided into left and right parts by an assignment operator that uses the equal sign. The construction operator is used to create new predicate vertices based on existing ones. The deletion

operator is designed to remove the lower-level predicates from the higher-level predicates. The transitive deletion operator also removes all elements that lose logical integrity as a result of deletion. The replacement operator is used to change the metagraph structure.

Thus, the proposed metagraph calculus allows to construct and modify the elements of the metagraph model.

1. Введение

В настоящее время модели на основе сложных сетей находят все более широкое применение в различных областях технических и естественных наук. Сложные сети рассматриваются в работах И.А. Евина [1], О.П. Кузнецова и Л.Ю. Жиликовой [2], К.В. Анохина [3] и других исследователей.

На кафедре «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана в рамках данного направления предложена метаграфовая модель. Данную модель предлагается применять как средство для описания сложных сетей [4], как средство для описания семантики и прагматики информационных систем [5], как средство для описания гибридных интеллектуальных информационных систем [6].

Расширение области применения метаграфового подхода требует уточнения теоретических положений метаграфовой модели.

В работе [5] была предложена формализованная модель метаграфа, а также рассмотрены основные операции над метаграфами на основе «информационных элементов метаграфа» (ИЭМ). В дальнейшем в работе [7] в качестве внутренней модели представления метаграфа вместо ИЭМ было предложено использование предикатного описания.

Хотя предложенные подходы и позволили отчасти формализовать метаграфовую модель, но результаты работ [5, 7] не позволяют полностью формально оперировать с элементами метаграфовой модели.

Для реализации возможности формального описания и преобразования элементов метаграфовой модели в данной статье предлагается метаграфовое исчисление.

В соответствии с классическим определением [8]: «Исчисление – это основанный на четких правилах формальный аппарат оперирования со знаниями определенного вида, позволяющий дать точное описание некоторого класса задач, а для отдельных подклассов этого класса – и алгоритм решения». Под «знаниями определенного вида» понимается метаграфовая модель данных, а «основанный на четких правилах формальный аппарат» реализован в виде операторов.

2. Основные положения метаграфовой модели

В целях ясности изложения предлагаемого подхода в данном разделе рассматриваются основные положения метаграфовой модели и ее предикатное описание на основании материалов статей [5, 6, 7]. Рассмотрим формализованное представление метаграфовой модели:

$$MG = \langle V, MV, E \rangle,$$

где MG – метаграф; V – множество вершин метаграфа; MV – множество метавершин метаграфа; E – множество ребер метаграфа.

Вершина метаграфа характеризуется множеством атрибутов:

$$v_i = \{atr_k\}, v_i \in V,$$

где v_i – вершина метаграфа; atr_k – атрибут.

Ребро метаграфа характеризуется множеством атрибутов, исходной и конечной вершиной и признаком направленности:

$$e_i = \langle v_s, v_E, eo, \{atr_k\} \rangle, e_i \in E, eo = true \mid false,$$

где e_i – ребро метаграфа; v_s – исходная вершина (метавершина) ребра; v_E – конечная вершина (метавершина) ребра; eo – признак направленности ребра ($eo=true$ – направленное ребро, $eo=false$ – ненаправленное ребро); atr_k – атрибут.

Фрагмент метаграфа:

$$MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV),$$

где MG_i – фрагмент метаграфа; ev_j – элемент, принадлежащий объединению множеств вершин, метавершин и ребер метаграфа.

Метавершина метаграфа:

$$mv_i = \langle \{atr_k\}, MG_j \rangle, mv_i \in MV,$$

где mv_i – метавершина метаграфа, принадлежащая множеству вершин MV ; atr_k – атрибут, MG_j – фрагмент метаграфа.

Таким образом, метавершина в дополнение к свойствам вершины включает вложенный фрагмент метаграфа, который может также содержать вложенные вершины, метавершины, ребра.

Наличие у метавершин собственных атрибутов и связей с другими вершинами является важной особенностью метаграфов. Это соответствует принципу эмерджентности, то есть приданию понятию нового качества, несводимости понятия к сумме его составных частей. Фактически, как только вводится новое понятие в виде метавершины, оно «получает право» на собственные свойства, связи и т.д., так как в соответствии с принципом эмерджентности новое понятие обладает новым качеством и не может быть сведено к подграфу базовых понятий. Таким образом, метаграф можно охарактеризовать как «граф с эмерджентностью», то есть фрагмент графа, состоящий из вершин и связей, может выступать как отдельное целое.

Необходимо отметить, что модель метаграфа также включает более сложные элементы, такие как метаребра и метаграфовые агенты, но в данной статье они не рассматриваются.

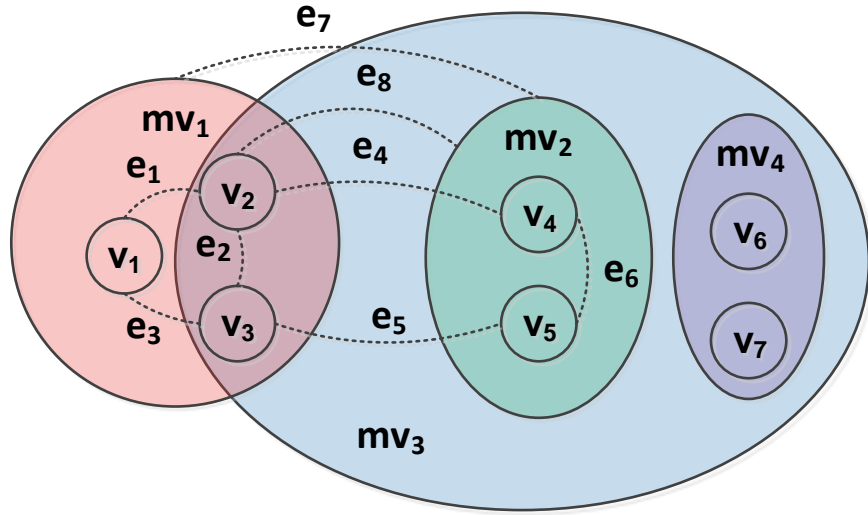


Рис. 1. Пример описания метаграфа.

Пример описания метаграфа показан на рис. 1. Данный метаграф содержит вершины, метавершины и ребра. На рис. 1 показаны четыре метавершины: mv_1 , mv_2 , mv_3 и mv_4 . Метавершина mv_1 включает вершины v_1 , v_2 , v_3 и связывающие их ребра e_1 , e_2 , e_3 . Метавершина mv_2 включает вершины v_4 , v_5 и связывающее их ребро e_6 . Ребра e_4 , e_5 являются примерами ребер, соединяющих вершины v_2 - v_4 и v_3 - v_5 , включенные в различные метавершины mv_1 и mv_2 . Ребро e_7 является примером ребра, соединяющего метавершины mv_1 и mv_2 . Ребро e_8 является примером ребра, соединяющего вершину v_2 и метавершину mv_2 . Метавершина mv_3 включает метавершины mv_2 и mv_4 , вершины v_2 , v_3 и ребро e_2 из метавершины mv_1 а также ребра e_4 , e_5 , e_8 . Метавершина mv_4 включает не соединенные ребрами вершины v_6 и v_7 .

Поскольку метаграфовая модель содержит несколько различных элементов (вершины, метавершины, ребра, метаребра), то возникает желание представить различные элементы модели в виде меньшего количества базовых элементов.

В работе [5] в качестве такого элемента предлагается использовать информационный элемент метаграфа (ИЭМ):

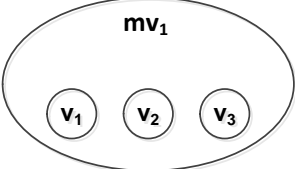
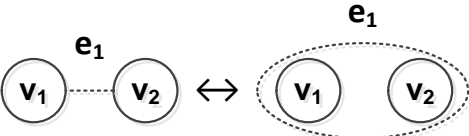
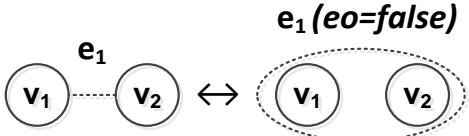
$$ИЭМ = \langle id, NM, VAL, RL, \{link_i\}, \{atr_j\} \rangle, RL \in \{RL_v, RL_{mv}, RL_E\},$$

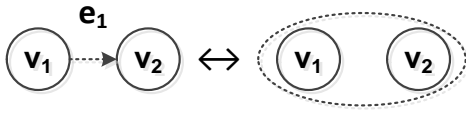
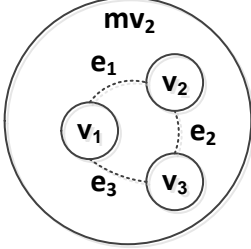
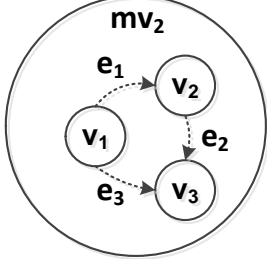

где id – уникальный идентификатор элемента; NM – наименование элемента; VAL – значение элемента; RL – роль элемента; $link_i$ – ссылка на другой ИЭМ; atr_j – атрибут; RL_v – роль элемента «вершина»; RL_{mv} – роль элемента «метавершина»; RL_e – роль элемента «ребро».

Основная проблема описания элементов метаграфа на основе ИЭМ состоит в том, что ИЭМ является недостаточно «атомарной» структурой. В частности, с использованием ИЭМ атрибут не может быть представлен как фрагмент метаграфа.

В работе [7] в качестве внутренней модели представления метаграфа вместо ИЭМ предлагается использовать предикатное описание. Классическим примером языка на основе предикатов является язык Пролог, который использует следующую форму предикатного описания: $predicate(atom_1, atom_2, \dots, atom_N)$. В работе [7] предлагается использовать расширенную форму предикатного описания: $predicate(atom, \dots, key = value, \dots, predicate(\dots), \dots)$. Данная форма, в дополнение к атомам, может также содержать пары ключ-значение и вложенные предикаты. Различные варианты отображения элементов метаграфовой модели в предикатное описание представлены в таблице 1.

Таблица 1. Соответствие фрагментов метаграфовой модели предикатному описанию.

Вариант	Фрагмент метаграфа	Предикатное описание
1		Metavertex(Name=mv ₁ , v ₁ , v ₂ , v ₃)
2		Edge(Name=e ₁ , v ₁ , v ₂)
3		Edge(Name=e ₁ , v ₁ , v ₂ , eo=false)

4		1. Edge(Name=e ₁ , v ₁ , v ₂ , eo=true) 2. Edge(Name=e ₁ , v _S =v ₁ , v _E =v ₂ , eo=true)
5		Metavertex(Name=mv ₂ , v ₁ , v ₂ , v ₃ , Edge (Name=e ₁ , v ₁ , v ₂), Edge(Name=e ₂ , v ₂ , v ₃), Edge(Name=e ₃ , v ₁ , v ₃))
6		Metavertex(Name=mv ₂ , v ₁ , v ₂ , v ₃ , Edge(Name=e ₁ , v _S =v ₁ , v _E =v ₂ , eo=true), Edge(Name=e ₂ , v _S =v ₂ , v _E =v ₃ , eo=true), Edge(Name=e ₃ , v _S =v ₁ , v _E =v ₃ , eo=true))
7		Attribute(количество, 5)

Предикатное описание позволяет представить метаграфовую модель в текстовом виде, и составные части предикатного описания достаточно «атомарны». Но предикатное описание не дает информации о том, какие действия можно выполнять над элементами метаграфовой модели.

Для преодоления рассмотренных недостатков предлагается использовать метаграфовое исчисление, которое позволяет выполнять формальные действия над атомарными структурами метаграфовой модели.

В следующем разделе рассмотрим мотивирующий пример, который будет являться основой для формального описания метаграфового исчисления.

3. Представление метаграфа как иерархически организованного множества предикатов

Рассмотрим пример описания метаграфа, показанный на рис. 1, в виде иерархического предикатного представления.

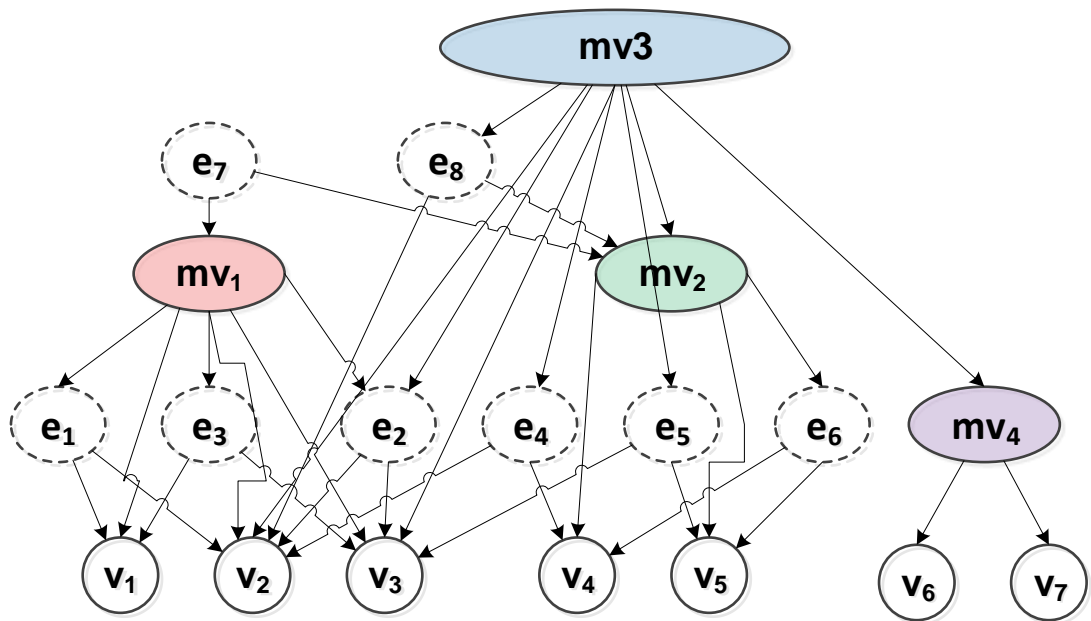


Рис. 2. Иерархическое предикатное представление метаграфа.

Иерархическая зависимость между предикатами представляет плоский граф. На рис. 2 предикаты показаны в виде вершин плоского графа. Предикаты, соответствующие ребрам, обозначены пунктиром. Стрелки направлены от предикатов верхнего уровня к предикатам нижнего уровня.

Граф, представленный на рис. 2, является стратифицированным. Нижний уровень содержит предикаты-вершины. Второй уровень включает предикаты, соответствующие связям между элементами первого уровня. Более высокие уровни содержат метавершины и связи между ними.

Необходимо отметить, что количество уровней стратифицированного графа можно рассматривать как косвенную характеристику сложности соответствующей метаграфовой модели.

Предикатное представление метавершин и ребер не имеет существенных различий. Так, второй уровень графа содержит шесть ребер и метавершину mv_4 . Ребро можно считать

частным случаем метавершины, которое может включать только две вложенных вершины и обладает признаком (атрибутом) направленности.

Этот вывод может показаться парадоксальным с точки зрения классической теории графов, где вершины и ребра считаются принципиально различными объектами. Но с точки зрения системного подхода в этом нет ничего удивительного. Можно рассматривать ребро как совокупность двух вложенных вершин, при этом факт наличия ребра между вершинами обозначает появление эмерджентности. Две вершины, соединенные ребром, представляют собой отдельный объект, более высоко организованный чем просто две отдельные вершины.

Тогда метавершина представляет собой эмерджентность более высокого порядка, которая включает как исходные вершины, так и соединяющие их ребра, и, возможно, метавершины нижнего уровня. При этом вершины, ребра и метавершины представлены отдельными предикатами.

Элементарные вершины представляют собой ноль-эмерджентность, то есть рассматриваются как атомарные элементы, внутренняя организация которых в модели не учитывается.

Рассмотренный подход представляет собой основу метаграфового исчисления.

Единственной структурой данных исчисления является вершина-предикат. Необходимо отметить, что у вершины-предиката очень прозрачная семантика – это множество вложенных вершин-предикатов нижнего уровня.

Все предлагаемые далее операторы предназначены для построения эмерджентных структур из вершин-предикатов.

4. Основные операторы метаграфового исчисления

В данном разделе рассмотрим основные операторы метаграфового исчисления. Автор преследовал цель сделать предлагаемое исчисление интуитивно понятным, поэтому

исчисление содержит простые конструкции, а обозначения операторов по возможности близки к традиционным обозначениям операторов в других подходах.

Все конструкции исчисления представляют собой строки, разделенные на левую и правую части оператором присваивания, в качестве которого используется знак равенства «=».

Слева от присваивания стоит имя конструируемого предиката, а справа предикаты и операторы исчисления.

В отличие от функциональных языков программирования на переменные, соответствующие вершинам-предикатам, не накладывается ограничение неизменяемости. Переменная может фигурировать и в правой, и в левой части одного выражения, то есть может быть перезаписана.

Для обозначения комментария используется общепринятая в C++-подобных языках конструкция «//».

4.1. Оператор конструирования

Оператор конструирования предназначен для создания новых вершин-предикатов на основе существующих. Применение этого оператора создает новую эмерджентность более высокого уровня.

Конструкции подобного вида присутствуют в различных парадигмах программирования. В объектно-ориентированном программировании это конструкторы классов; в функциональном программировании оператор «cons», предназначенный для рекурсивного создания коллекций.

Для синтаксического обозначения оператора конструирования используется символ сложения «+». Данный оператор является n-местным, так как метавершина может включать в себя произвольное количество вершин и ребра.

Для удобства записи все рассматриваемые операторы могут использовать традиционную и префиксную формы. В префиксной форме используется общепринятый синтаксис группировки данных с помощью круглых скобок и запятой.

```
// Пример конструирования метавершины mv1
```

```
// Традиционная форма
```

```
mv1 = v1+v2+v3+e1+e2+e3
```

```
// Префиксная форма
```

```
mv1 = +(v1,v2,v3,e1,e2,e3)
```

Поскольку метаграфовая модель изначально включала понятие ребра, то в исчисление включен частный случай оператора конструирования для создания ребер. В этом случае используется синтаксис «++». Единственной особенностью синтаксиса «++» является то, что данный вариант оператора является двухместным, то есть допускает не более двух операндов.

```
// Пример конструирования ребра e1
```

```
// Традиционная форма
```

```
e1 = v1++v2
```

```
// Префиксная форма
```

```
e1 = ++(v1,v2)
```

```
// Примеры некорректных конструкций
```

```
// нарушено ограничение на двухместность оператора
```

```
e1 = v1++v2++v3 //Ошибка
```

```
e1 = ++(v1,v2,v3) //Ошибка
```

Оператор конструирования не позволяет создавать конструкции, содержащие циклические ссылки:

```
mv3 = mv2+mv4
```

$mv4 = v11 + mv3$ //Ошибка – $mv3$ содержит $mv4$, поэтому $mv4$ не может включать $mv3$.

4.2. Операторы удаления и транзитивного удаления

Оператор удаления предназначен для удаления вершин-предикатов нижнего уровня из вершин-предикатов верхнего уровня. В этом случае используется синтаксис «-».

// Пример удаления ребра $e1$ из метавершины $mv1$

// Традиционная форма

$mv1 = mv1 - e1$

// Префиксная форма

$mv1 = -(mv1, e1)$

На оператор удаления в метаграфовой модели накладывается несколько ограничений. Эти ограничения связаны с обеспечением логической целостности элементов метаграфовой модели.

1. Поскольку ребро содержит ровно две вершины, то ни одну из этих двух вершин невозможно удалить. В этом случае нарушается логическая целостность ребра.
2. Невозможно удалить предикат A , из предиката B , если предикат B содержит вложенный предикат C , ссылающийся на предикат A . В этом случае нарушается логическая целостность предиката C . Пример нарушения целостности:

$mv1 = v1 + v2$

$mv2 = v1 + v2 + mv1$

$mv2 = mv2 - v1$ //Ошибка – ссылка на $v1$ присутствует в $mv1$

Для преодоления второго ограничения в исчисление введен оператор транзитивного удаления с синтаксисом «-*». В этом случае также транзитивно удаляются все элементы, теряющие логическую целостность в результате удаления. Пример транзитивного удаления:

$mv1 = v1 + v2$

```
mv2 = v1+v2+mv1
```

```
mv2 = mv2-*v1
```

```
//После удаления mv2 = v2
```

До удаления элемент `mv2` содержит элементы `v1`, `v2`, `mv1`. Элемент `v1` удаляется из `mv2` по запросу. Поскольку элемент `mv1`, вложенный в `mv2`, содержит ссылку на удаляемый элемент `v1`, то `mv1` должен быть транзитивно удален из `mv2`. Поэтому в результате выполнения оператора транзитивного удаления `mv2` содержит только элемент `v2`.

4.3. Оператор замены

Для изменения метаграфовой структуры предназначен оператор замены с синтаксисом «исходный элемент : элемент для поиска -> заменяющий элемент». Пример замены:

```
mv1 = v1+v2
```

```
mv1 = mv1 : v2 -> v3
```

```
// В результате замены mv1 содержит v1 и v3.
```

Поскольку оператор замены основывается на конструировании и удалении элементов, то нарушение рассмотренных выше ограничений для операторов конструирования и удаления приводит к возникновению ошибки замены.

4.4. Итоговый пример

С помощью предложенного исчисления сконструируем рассмотренный ранее пример описания метаграфа.

```
// Метавершина mv1 включает вершины v1, v2, v3
```

```
// и связывающие их ребра e1, e2, e3.
```

```
e1 = v1++v2
```

```
e2 = v2++v3
```

```
e3 = v1++v3
```

```
mv1 = v1+v2+v3+e1+e2+e3
```

```

// Метавершина  $mv_2$  включает вершины  $v_4$ ,  $v_5$  и связывающее их ребро  $e_6$ .
 $e_6 = v_4++v_5$ 
 $mv_2 = v_4+v_5+e_6$ 

// Метавершина  $mv_4$  включает не соединенные ребрами вершины  $v_6$  и  $v_7$ .
 $mv_4 = v_6+v_7$ 

// Ребра  $e_4$ ,  $e_5$  являются примерами ребер,
// соединяющих вершины  $v_2-v_4$  и  $v_3-v_5$ ,
// включенные в различные метавершины  $mv_1$  и  $mv_2$ .
 $e_4 = v_2++v_4$ 
 $e_5 = v_3++v_5$ 

// Ребро  $e_7$  является примером ребра,
// соединяющего метавершины  $mv_1$  и  $mv_2$ .
 $e_7 = mv_1++mv_2$ 

// Ребро  $e_8$  является примером ребра,
// соединяющего вершину  $v_2$  и метавершину  $mv_2$ .
 $e_8 = v_2++mv_2$ 

// Метавершина  $mv_3$  включает метавершины  $mv_2$  и  $mv_4$ ,
// вершины  $v_2$ ,  $v_3$  и ребро  $e_2$  из метавершины  $mv_1$ ,
// а также ребра  $e_4$ ,  $e_5$ ,  $e_8$ .
 $mv_3 = mv_2+mv_4+v_2+v_3+e_2+e_4+e_5+e_8$ 

// Удалим из метавершины  $mv_4$  ребро  $v_6$ .
 $mv_4 = mv_4-v_6$ 

// А ребро  $v_7$  заменим на  $v_1$ .
 $mv_4 = mv_4 : v_7 \rightarrow v_1$ 

```

Таким образом, предложенное метаграфовое исчисление позволяет конструировать и модифицировать элементы метаграфовой модели.

5. Выводы

В статье рассмотрено метаграфовое исчисление, реализующее возможность формального описания и преобразования элементов метаграфовой модели.

Единственной структурой данных исчисления является вершина-предикат. Вершина-предикат обладает прозрачной семантикой – это множество вложенных вершин-предикатов нижнего уровня.

Оператор конструирования предназначен для создания новых вершин-предикатов на основе существующих. Применение этого оператора создает новую эмерджентность более высокого уровня.

Оператор удаления предназначен для удаления вершин-предикатов нижнего уровня из вершин-предикатов верхнего уровня. Оператор транзитивного удаления также удаляет все элементы, теряющие логическую целостность в результате удаления.

Для изменения метаграфовой структуры используется оператор замены.

Предложенное метаграфовое исчисление позволяет конструировать и модифицировать элементы метаграфовой модели.

Литература

1. Евин И.А. Введение с теорию сложных сетей //Компьютерные исследования и моделирование. 2010, Том 2, №2, с. 121-141.
2. Кузнецов О.П., Жиликова Л.Ю. Сложные сети и когнитивные науки // Нейроинформатика-2015. XVII Всероссийская научно-техническая конференция. Сборник научных трудов. Ч. 1. М.: МИФИ. 2015. С. 18.
3. Анохин К.В. Когнитом: гиперсетевая модель мозга // Нейроинформатика-2015. XVII Всероссийская научно-техническая конференция. Сборник научных трудов. Ч. 1. М.: НИЯУ МИФИ. 2015. С. 14-15.

4. Черненький В.М., Терехов В.И., Гапанюк Ю.Е. Представление сложных сетей на основе метаграфов // Нейроинформатика-2016. XVIII Всероссийская научно-техническая конференция. Сборник научных трудов. Ч. 1. М.: НИЯУ МИФИ, 2016.
5. Самохвалов Э.Н., Ревунков Г.И., Гапанюк Ю.Е. Использование метаграфов для описания семантики и прагматики информационных систем. Вестник МГТУ им. Н.Э. Баумана. Сер. «Приборостроение». 2015. Выпуск №1. С. 83-99.
6. Черненький В.М., Терехов В.И., Гапанюк Ю.Е. Структура гибридной интеллектуальной информационной системы на основе метаграфов. Нейрокомпьютеры: разработка, применение. 2016. Выпуск №9. С. 3-14.
7. Гапанюк Ю.Е., Ревунков Г.И., Федоренко Ю.С. Предикатное описание метаграфовой модели данных. Информационно-измерительные и управляющие системы. 2016. Выпуск № 12. С. 122-131.
8. Ивин А. А., Никифоров А. Л. Словарь по логике - М: Гуманит. изд. центр ВЛАДОС, 1997.- 384 с.

References

1. Yevin I.A. *Vvedenie v teoriyu slozhnykh setey* [The introduction of the theory of complex networks]. *Kompiuterniye issledovaniya i modelirovaniye* [Computer research and modeling], 2010, no. 2(2), pp. 121-141.
2. Kuznetsov O.P., Zhilyakova L.Yu. *Slozhniye seti i kognitivniye nauki* [Complex networks and cognitive sciences]. *Trudi XVII vserossiyskoy konferencii "Neuroinformatics-2015"* [Proc. XVII all-russian conference "Neuroinformatics-2015"], Moscow, 2015, p. 18.
3. Anokhin K.V. *Kognitom: gipersetevaya model mozga* [The cognitome: a hypernetwork brain model]. *Trudi XVII vserossiyskoy konferencii "Neuroinformatics-2015"* [Proc. XVII all-russian conference "Neuroinformatics-2015"], Moscow, 2015, pp. 14-15.

4. Chernenkiy V.M., Terekhov V.I., Gapanyuk Yu.E. *Predstavleniye slozhnikh setey na osnove metagrafov* [Metagraph representation of complex networks]. *Trudi XVIII vserossiyskoy konferencii "Neuroinformatics-2016"* [Proc. XVIII all-russian conference "Neuroinformatics-2016"], Moscow, 2016, pp. 173-178.
5. Samokhvalov E.N., Revunkov G.I. Gapanyuk Yu.E. *Ispolzovaniye metagrafov dlya opisaniya semantiki i pragmatiki informatsionnykh sistem* [Metagraphs for information systems semantics and pragmatics definition]. *Vestnik MGTU im. N.E. Baumana, seriya "Priborostroeniye"* [Herald of the Bauman Moscow State Technical University, "Instrument Engineering"], 2015, no. 1, pp. 83-99.
6. Chernenkiy V.M., Terekhov V.I., Gapanyuk Yu.E. *Struktura gibridnoy intellektualnoy informacionnoy sistemy na osnove metagrafov* [Structure of the hybrid intelligent information system based on metagraphs]. *Neirokompyutery: razrabotka, primeneiye* [Neurocomputers: development, application], 2016, no. 9, pp. 3-14.
7. Gapanyuk Yu.E., Revunkov G.I., Fedorenko Yu.S. *Predikatnoye opisaniye metagrafovoy modeli dannykh* [Predicate representation of metagraph data model]. *Informatsionno-izmeritelniye i upravlyayushchie sistemi* [Information-measuring and Control Systems], 2016, no. 12, pp. 122-131.
8. Ivin A.A., Nikiforov A.L. *Slovar po logike* [Dictionary of logic] Moscow: Gumanit. izd. tsentr VLADOS, 1997. - 384 p.

Сведения об авторах

Гапанюк Юрий Евгеньевич

Год рождения: 1974

Год окончания вуза и его название: 1998, МГТУ им. Н.Э. Баумана

Ученая степень: к.т.н.

Место работы, должность: доцент кафедры ИУ-5 МГТУ им. Н.Э. Баумана

Полный адрес организации: Московский Государственный Технический Университет им. Н.Э.

Баумана, г. Москва, 2-я Бауманская ул., д. 5, стр. 1, почтовый индекс: 105005

Количество опубликованных работ: около 40

Область научных интересов: проектирование автоматизированных систем, проектирование гибридных интеллектуальных информационных систем, сложные графовые модели

Электронная почта: gapyu@bmstu.ru

Контактный телефон: 8 916 558 94 30

Information about authors

Gapanyuk Yuriy Evgenievich

Год рождения: 1974

Год окончания вуза и его название: 1998, Bauman Moscow State Technical University

Ученая степень: Ph.D. (Computer Sciences)

Место работы, должность: associate professor of Computer Science and Control Systems Department at Bauman Moscow State Technical University

Полный адрес организации: Bauman Moscow State Technical University, ul. Baumanskaya 2-ya, 5, Moscow, postcode: 105005

Количество опубликованных работ: about 40 publications

Область научных интересов: designing of automated systems, designing of hybrid intelligent information systems, complex graph models

Электронная почта: gapyu@bmstu.ru

Контактный телефон: 8 916 558 94 30