

Ю.Е. Гапанюк

**КОНСПЕКТ ЛЕКЦИЙ ПО СПЕЦКУРСУ
«ГИБРИДНЫЕ ИНТЕЛЛЕКТУАЛЬНЫЕ
ИНФОРМАЦИОННЫЕ СИСТЕМЫ
НА ОСНОВЕ МЕТАГРАФОВОГО ПОДХОДА»**

Учебно-методическое пособие

Издательство «Спутник +»

Москва 2018

Гапанюк Ю.Е.

Конспект лекций по спецкурсу «Гибридные интеллектуальные информационные системы на основе метаграфового подхода»: Учебно-методическое пособие. – М.: Издательство «Спутник +», 2018. – 53с., ил.

В учебно-методическом пособии рассматриваются теоретические вопросы разработки гибридных интеллектуальных информационных систем. В качестве модели знаний используется метаграфовая модель, которая подробно разобрана в пособии. Пособие выполнено в виде конспекта лекций, которые могут послужить основой или для небольшого спецкурса, или для модуля курса по искусственному интеллекту, интеллектуальным системам. Каждая лекция сопровождается контрольными вопросами для самопроверки обучающихся. Учебно-методическое пособие предназначено для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника».

Оглавление

ЛЕКЦИЯ 1. ПОНЯТИЕ ГИБРИДНОЙ ИНТЕЛЛЕКТУАЛЬНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ (ГИИС). ИСПОЛЬЗОВАНИЕ МНОГОАГЕНТНОЙ СИСТЕМЫ (МАС) ДЛЯ РЕАЛИЗАЦИИ ГИИС	4
ЛЕКЦИЯ 2. ОБОБЩЕННАЯ СТРУКТУРА ГИИС И ЕЕ ЧАСТНЫЕ СЛУЧАИ. СВЯЗЬ СТРУКТУРЫ ГИИС С ХОЛОНИЧЕСКОЙ МАС. ОСНОВНЫЕ ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ ГИИС	7
ЛЕКЦИЯ 3. ФОРМАЛИЗОВАННОЕ ОПИСАНИЕ МЕТАГРАФОВОЙ МОДЕЛИ	14
ЛЕКЦИЯ 4. СРАВНЕНИЕ МЕТАГРАФОВОЙ МОДЕЛИ С МОДЕЛЯМИ ГИПЕРГРАФА И ГИПЕРСЕТИ	19
ЛЕКЦИЯ 5. МОДЕЛЬ ХОЛОНИЧЕСКОЙ МАС, ПРЕДНАЗНАЧЕННОЙ ДЛЯ РЕАЛИЗАЦИИ ГИИС	25
ЛЕКЦИЯ 6. ОПИСАНИЕ МЕТАГРАФОВОЙ МОДЕЛИ С ИСПОЛЬЗОВАНИЕМ ИНФОРМАЦИОННЫХ ЭЛЕМЕНТОВ МЕТАГРАФА	31
ЛЕКЦИЯ 7. ПРЕДИКАТНОЕ ОПИСАНИЕ МЕТАГРАФОВОЙ МОДЕЛИ	38
ЛЕКЦИЯ 8. ПРИМЕРЫ ОПИСАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ МЕТАГРАФОВОЙ МОДЕЛИ.....	43
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	51

Лекция 1. Понятие гибридной интеллектуальной информационной системы (ГИИС). Использование многоагентной системы (МАС) для реализации ГИИС

В настоящее время для построения интеллектуальных систем используется большое количество подходов: продукционные правила, нейронные сети, нечеткая логика, эволюционные методы и др. При этом можно отметить явную тенденцию к совместному использованию разных методов для решения различных классов задач. Это привело к появлению такого направления, как **«гибридные интеллектуальные системы» (ГИС)**. основополагающими работами в области ГИС можно считать работы профессора А.В. Колесникова [1, 2].

При этом необходимо различать понятия **«гибридная интеллектуальная система»** и **«интегрированная интеллектуальная система»**. Второе понятие является более широким. Профессор Г.В. Рыбина отмечает [3], что интегрированная интеллектуальная система является гибридной только в случае полной интеграции ее компонентов.

Необходимо отметить, что в англоязычной литературе термин **«hybrid intelligence»** в основном используется для обозначения гибридизации различных методов мягких вычислений и экспертных систем [4, 5]: нейро-нечеткие системы, нечеткие экспертные системы, использование эволюционных методов для конструирования нейронных сетей и нечетких моделей, другие методы. Подобная **«глубокая» гибридизация** полностью соответствует критерию Г.В. Рыбиной о полной интеграции компонентов гибридной системы.

Ключевым вопросом является вопрос о том, каким образом реализовать принцип гибридности. Ответ на этот вопрос предлагается, в частности, в работе профессора Н.Г. Ярушкиной [6]. В ней сформулирован следующий принцип гибридности [6, с. 20-21]: «В литературе встречаются схемы гибридизации нейроинформатики и ИИ, построенные по следующему принципу: правое полушарие – нейрокомпьютер; левое полушарие – основанная на знаниях

система, а вопрос лишь в их взаимодействии или балансе право- и лево-полушарности. В реальном поведении человека невозможно разделить восприятие и логическую обработку, поэтому более успешной представляется схема глубинной интеграции».

Очевидно, что основой для такой интеграции должна являться модель знаний, которая обеспечит уровень сложности и многообразия, необходимый для решения задачи.

Довольно часто при проектировании ГИС применяют сервис-ориентированный подход, при котором каждый сервис может использовать собственную модель знаний. Но такой подход может создавать серьезные сложности при интеграции систем, состоящих даже из относительно небольшого числа сервисов.

В настоящее время в качестве моделей глубинной интеграции все чаще рассматриваются модели на основе сложных сетей. Сложные сети рассматриваются в работах И.А. Евина [7], О.П. Кузнецова и Л.Ю. Жиликовой [8], К.В. Анохина [9] и других исследователей.

На кафедре «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана в рамках данного направления предложена метаграфовая модель [10]. Данную модель предлагается применять как средство для описания сложных сетей [11], как средство для описания семантики и прагматики информационных систем [12], как средство для описания гибридных интеллектуальных информационных систем [13].

В настоящее время интеллектуальные системы, как правило, не разрабатываются отдельно, но встраиваются в виде модулей в традиционные информационные системы для решения задач, связанных с интеллектуальной обработкой данных и знаний. Такую комбинированную систему в соответствии с [13] назовем **гибридной интеллектуальной информационной системой (ГИИС)**. ГИИС обладает следующими особенностями:

1. сочетает различные методы, используемые для построения интеллектуальных систем, и в этом смысле является ГИС;

2. сочетает интеллектуальные методы с традиционными методами, используемыми для разработки данных в информационных системах, и в этом смысле является комбинацией ГИС и информационной системы, предназначенной для обработки данных;
3. использует единую модель знаний, которая обеспечивает «глубокую» или «глубинную» интеграцию по Г.В. Рыбиной и Н.Г. Ярушкиной. В качестве такой модели в данном курсе предлагается использовать метаграфовую модель, которая рассмотрена более подробно в следующих лекциях.

В настоящее время для реализации интеллектуальных систем (в том числе гибридных) часто используется подход на основе **многоагентной системы (МАС)**. В работе В.Б. Тарасова [14] приводятся несколько определений МАС и программных агентов, мы адаптируем данные определения применительно к ГИИС.

Под программным агентом будем понимать программный модуль, который выполняется в виде автономной задачи (не зависит от других агентов), способен обмениваться информацией со средой и другими агентами. Под МАС будем понимать систему однородных или разнородных агентов, функционирующих в среде.

Для реализации ГИИС наиболее интересным представляется подход на основе **холонической многоагентной системы (холонической МАС)**. Такой класс систем также подробно рассмотрен в [14]. В соответствии с определением [14, с. 234] **холон** – это «целое, рассматриваемое в то же время как часть целого». Этот подход является обобщением понятия «вложенности».

Холонический подход предполагает «глубинную» интеграцию как модели знаний, так и агентов, осуществляющих обработку знаний.

Контрольные вопросы

1. Что такое ГИС?
2. В чем различия между «гибридной интеллектуальной системой» и «интегрированной интеллектуальной системой»?

3. В чем недостатки сервис-ориентированного подхода для разработки ГИС?
4. Почему наличие единой модели знаний важно для глубинной интеграции компонентов ГИС?
5. Что такое ГИИС, чем она отличается от ГИС?
6. Каковы основные особенности ГИИС?
7. Что такое многоагентная система (МАС)?
8. В чем особенности холонической многоагентной системы?

Лекция 2. Обобщенная структура ГИИС и ее частные случаи. Связь структуры ГИИС с холонической МАС. Основные требования к реализации ГИИС

Как было сказано в предыдущей лекции, принцип гибридности был чрезвычайно удачно предложен профессором Н.Г. Ярушкиной [6].

Можно несколько развить метафору право- и лево-полушарности, предложенную в [6], и говорить о «подсознании» и «сознании» ГИИС. «Подсознание» строится на основе методов мягких вычислений (и условно соответствует правому полушарию), а «сознание» на основе традиционных методов обработки данных и знаний (и условно соответствует левому полушарию).

Обобщенная структура ГИИС, построенная на основе «сознания» и «подсознания» представлена на рис. 1.

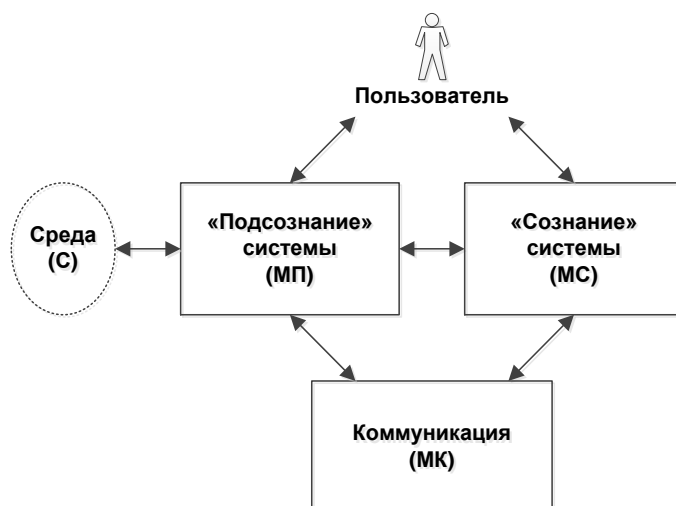


Рис. 1. Обобщенная структура ГИИС.

Основой системы являются **«подсознание» системы (модуль подсознания, МП)** и **«сознание» системы (модуль сознания, МС)**. «Подсознание» связано со средой, в которой функционирует ГИИС.

Основной задачей МП является обеспечение взаимодействия ГИИС со «средой», или «выживание» ГИИС в среде.

Поскольку среда может быть представлена в виде набора непрерывных сигналов, то в качестве методов обработки данных «подсознания» хорошо подходят методы, основанные на нейронных сетях, нечеткой логике, алгоритмах машинного обучения, в том числе и комбинированные (например, нейронечеткие) методы.

Модель данных «подсознания» максимально приближена к «понятийной системе» среды, представляет собой набор данных, который позволяет максимально эффективно взаимодействовать со средой.

Часть этих данных может не иметь «физического смысла» с точки зрения МС, однако позволяет МП взаимодействовать со средой с нужной производительностью. Такой подход, в частности, часто встречается в машинном обучении, когда для работы обучающихся алгоритмов используются синтетические признаки, которые не имеют «физического смысла», но позволяют повысить эффективность работы алгоритмов.

«Сознание» ГИИС строится на принципах обработки данных и знаний. Обработка данных в МС может вестись на основе традиционных языков программирования или технологии workflow. Однако, в последнее время все большую популярность приобретает подход на основе продукционных правил. Раньше данный подход использовался для принятия решения в экспертных системах, но в настоящее время на основе правил пишутся обычные программы. Такой подход называется программированием на основе правил (rule-based programming).

Отметим, что, в зависимости от особенностей предметной области, правила могут быть нечеткими или вероятностными, что вносит в МС элементы МП. Это одно из проявлений принципа холоничности.

К достоинствам подхода на основе правил можно отнести гибкость, так как в этом случае программа не кодируется жестко, а фактически «выводится» из правил на основе данных. К недостаткам можно отнести возможность заикливания правил, а также сложность обработки большого объема правил.

Отметим, что задача хранения требуемых данных решается отдельно на уровне МС и на уровне МП. Мы предполагаем, что на уровне обобщенной структуры соответствующие хранилища «встроены» в МС и МП, поэтому хранилища не представлены на рис. 1 в виде отдельных блоков.

С точки зрения коммуникации, в ГИИС возможны следующие варианты или их комбинации:

1. Коммуникация осуществляется через среду. МП читает данные из среды, преобразует и передает в МС. МС осуществляет логическую обработку и возвращает результаты обработки в МП. МП записывает результирующие данные в среду, откуда они могут быть прочитаны другими ГИИС.
2. Для коммуникации с другими ГИИС используется **модуль коммуникации (МК)**. В зависимости от решаемых задач с МК может взаимодействовать МС (что характерно для традиционных информационных систем) или МП (что более характерно для систем на основе мягких вычислений).
3. Взаимодействие с пользователем также может осуществляться через МС (что характерно для традиционных информационных систем) или через МП (что может быть использовано, например, в автоматизированных тренажерах).

На основе обобщенной структуры ГИИС могут быть рассмотрены частные случаи структуры, которые соответствуют конкретным ГИИС. Примеры таких

частных случаев представлены на рис. 2. Модули на рис. 2 представлены в виде обозначений, соответствующих рис. 1.

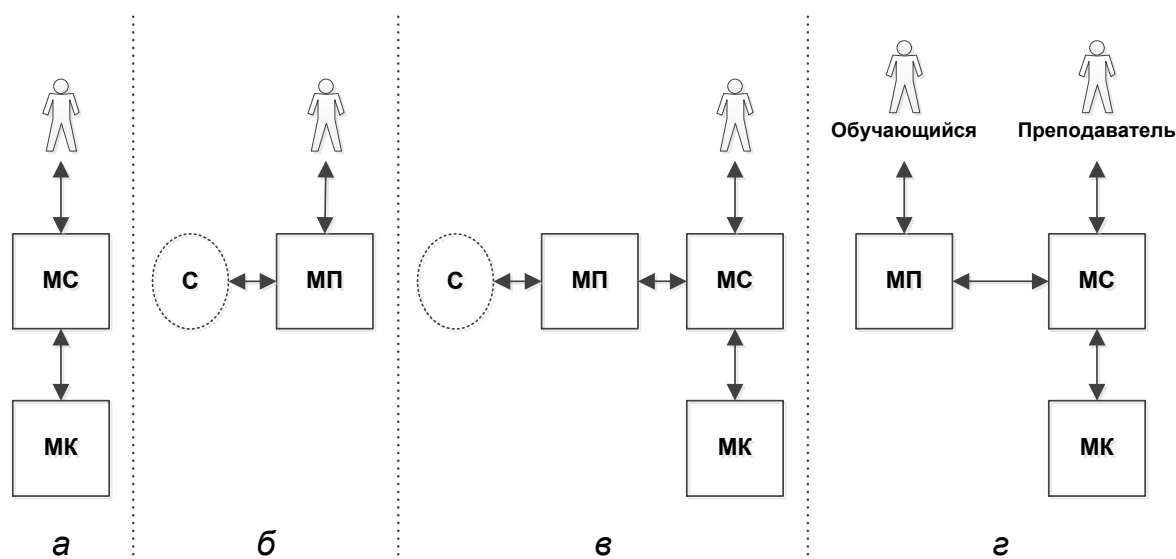


Рис. 2. Частные случаи структуры ГИИС.

Рисунку 2а соответствует классическая информационная система, в которой осуществляется только обработка данных и знаний, реализуется коммуникация с другими системами и взаимодействие с пользователем.

Рисунку 2б соответствует, в частности, простейшая система распознавания сигналов, поступающих из среды, с помощью МП. Сигналы могут иметь различную природу. Это может быть система распознавания музыкальной партитуры по звуковому сигналу, система распознавания элементов в видеопотоке и др. Данная система является простейшей, так как в ней отсутствует МС, который должен осуществлять коррекцию логических ошибок. Здесь эта задача возлагается на МП, что может приводить к сложным правилам при распознавании и обработке сигналов.

Рисунку 2в может соответствовать большое количество конкретных ГИИС. Приведем несколько примеров:

1. Усовершенствованная система распознавания сигналов. Сигналы выделяются из среды с помощью МП и преобразуются в элементы онтологии, которые обрабатывает МС. МС осуществляет дополнительный логический контроль. Например, для системы распознавания музыкальной

партитуры, МП выделяет ноты из входного сигнала (здесь ноты являются элементами онтологии), а МС может скорректировать неверно распознанную ноту на основе правил музыкальной гармонии. В этом случае модуль коммуникации может не использоваться.

2. Медицинская система функциональной диагностики. В этом случае роль среды выполняют сигналы от медицинских приборов. МП преобразует сигналы в элементы онтологии, МС на основе продукционных правил может осуществлять поддержку принятия решений. Пользователем является врач, модуль коммуникации может осуществлять коммуникацию с другими информационными системами.
3. АСУТП (автоматизированная система управления технологическими процессами), использующая методы мягких вычислений. В этом случае роль среды выполняют наблюдаемые параметры технологического процесса. МП преобразует сигналы в элементы онтологии, МС на основе продукционных правил может осуществлять логический контроль поступающей информации и поддержку принятия решений. Пользователем является оператор АСУТП, модуль коммуникации может осуществлять коммуникацию с другими информационными системами.

Рисунку 2г соответствует автоматизированная система виртуального тренажера. В этом случае действия обучающегося по управлению тренажером поступают на МП. МП преобразует сигналы в элементы онтологии, МС на основе продукционных правил может осуществлять логический контроль поступающей информации, поддержку принятия решений и выдачу информации преподавателю. Модуль коммуникации может быть использован в случае группы тренажеров.

С точки зрения многоагентного подхода, рассмотренные компоненты, такие как МП, МС, МК являются агентами. В то же время они являются частями системы, которая, в свою очередь, является агентом.

При этом МП является сложной структурой, которая включает агенты нижнего уровня, каждый из которых может в свою очередь включать МП, МС,

МК, предназначенные для решения конкретных задач данного агента. Несмотря на то, что агент нижнего уровня находится в составе МП, он может включать в свою структуру МС, предназначенный для решения задач МП более высокого уровня. Поэтому, с точки зрения данного подхода, нет ничего удивительного в том, что в МС могут использоваться нечеткие продукционные правила, а в МП входят «классические» модули обработки данных.

Отметим, что, хотя для решения задач МС, могут быть использованы методы обработки правил, а для решения задач МП нейронечеткие методы, все эти методы являются статическими. То есть предполагается, что логические правила, структура нейросети и т.д. задаются на этапе проектирования ГИИС и не изменяются в процессе работы.

Однако, подобный статический подход является недостаточным по следующим причинам:

- нет возможности использования эволюционных методов (генетические алгоритмы, генетическое программирование и др.);
- в настоящее время для разработки ГИС начинают активно использоваться самоорганизующиеся нейронные сети (в частности, такие топологии как SOINN [15], hyperNEAT [16]), их использование предполагает динамическое изменение топологии нейронной сети во время работы.
- нет возможности использования других подходов, связанных с изменением порядка действий, таких как динамические workflow, алгоритмы автоматизированного планирования.

Сформулируем основные требования к реализации ГИИС. В предыдущей лекции мы говорили о том, что ГИИС должна быть реализована на основе холонической МАС. Поэтому фактически мы формулируем основные требования к холонической МАС, предназначенной для реализации ГИИС (*отметим, что на эти требования неоднократно будут осуществлять ссылки в следующих лекциях*):

Требование 1. Агент должен реализовывать правила работы для МП или для МС.

Агент может быть аналогом программной процедуры, которая вычисляет функцию активации нейрона. Может быть реактивным агентом, который реализует поведение на основе заданных правил. Может быть проактивным агентом, который реализует интеллектуальные алгоритмы планирования действий и взаимодействия с другими агентами.

Требование 2. Агенты должны поддерживать принцип холонической организации. То есть агент может быть построен как структура из агентов нижнего уровня, которые агент считает «элементарными», но которые, в свою очередь, могут состоять из агентов более низкого уровня.

Требование 3. Для реализации свойства динамичности должна существовать возможность перестройки как структуры связей между агентами, так и внутренней структуры самого агента.

Отметим, что, поскольку речь идет о холонической организации агентов, то возможны случаи, когда изменение внутренней структуры самого агента предполагает изменение связей между входящими в его состав агентами нижнего уровня. Наиболее близким аналогом третьего требования в традиционных языках программирования является принцип самоотображаемости. Самоотображаемость (англ. homoiconicity) – это способность языка программирования анализировать программу на этом языке как структуру данных. Термин «самоотображаемость» часто используют вместе с термином «метапрограммирование». Самоотображаемость предполагает анализ программ как структур данных, а метапрограммирование предполагает генерацию на основе этих структур данных других программ.

В следующей лекции осуществляется поиск модели на основе сложных сетей, которая позволит реализовать сформулированные требования.

Контрольные вопросы

1. В чем состоит метафора «подсознания» и «сознания» ГИИС?

2. Нарисуйте обобщенную структуру ГИИС, прокомментируйте ее основные компоненты.
3. Для чего используется модуль подсознания ГИИС? Какие подходы и технологии могут быть использованы для его реализации?
4. Для чего используется модуль сознания ГИИС? Какие подходы и технологии могут быть использованы для его реализации?
5. Какие варианты коммуникации возможны в ГИИС?
6. Нарисуйте частный случай структуры ГИИС для классической информационной системы.
7. Нарисуйте частный случай структуры ГИИС для системы распознавания сигналов, поступающих из среды.
8. Нарисуйте частный случай структуры ГИИС для простого случая АСУТП.
9. Нарисуйте частный случай структуры ГИИС для автоматизированной системы виртуального тренажера.
10. Сформулируйте основные требования к реализации ГИИС.

Лекция 3. Формализованное описание метаграфовой модели

В данной лекции будет рассмотрена формализованная модель метаграфа как базовая модель для представления сложных сетей. В следующей лекции будут показаны преимущества метаграфовой модели перед гиперграфами и гиперсетями для описания «сетей с эмерджентностью».

Основополагающими работами по теории метаграфов являются работы А. Базу и Р. Блэннинга, которые в 2007 году были обобщены в виде монографии [17]. Модель, предложенная в [17], в дальнейшем была адаптирована для описания различных аспектов интеллектуальных информационных систем и представлена в статьях [10-13]. В данном разделе кратко рассмотрим формализованную модель метаграфа в соответствии с [10-13].

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

где MG – метаграф; V – множество вершин метаграфа; MV – множество метавершин метаграфа; E – множество ребер метаграфа; ME – множество метаребер 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:

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

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

Таким образом, фрагмент метаграфа в общем виде может содержать произвольные вершины (метавершины) и ребра (метаребра) без ограничений. Ограничения вводятся на фрагменты метаграфа, входящие в метавершину и метаребро.

Метавершина метаграфа:

$$mv_i = \langle \{atr_k\}, \{ev_j\} \rangle, mv_i \in MV, ev_j \in (V \cup E^{eo=false} \cup MV \cup ME^{eo=false}),$$

где mv_i – вершина метаграфа; atr_k – атрибут, ev_j – элемент, принадлежащий объединению множеств вершин (метавершин) и ребер (метаребер) метаграфа.

Таким образом, метавершина в дополнение к свойствам вершины включает вложенный фрагмент метаграфа. При этом ребра и метаребра этого фрагмента могут быть только ненаправленными, $eo=false$.

Метаребро метаграфа:

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

$$ev_j \in (V \cup E^{eo=true} \cup MV \cup ME^{eo=true}),$$

где me_i – метаребро метаграфа; v_s – исходная вершина (метавершина) ребра; v_E – конечная вершина (метавершина) ребра; eo – признак направленности метаребра ($eo=true$ – направленное метаребро, $eo=false$ – ненаправленное метаребро); atr_k – атрибут; ev_j – элемент, принадлежащий объединению множеств вершин (метавершин) и ребер (метаребер) метаграфа.

Таким образом, метаребро в дополнение к свойствам ребра включает вложенный фрагмент метаграфа. При этом ребра и метаребра этого фрагмента могут быть только направленными, $eo=true$.

Определения метавершины и метаребра являются рекурсивными, так как элементы ev_j могут быть, в свою очередь, метавершинами и метаребрами. Метавершина является формализмом описания данных, а метаребро – формализмом описания процессов.

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 «сеть с эмерджентностью», то есть фрагмент сети, состоящий из вершин и связей, может выступать как отдельное целое.

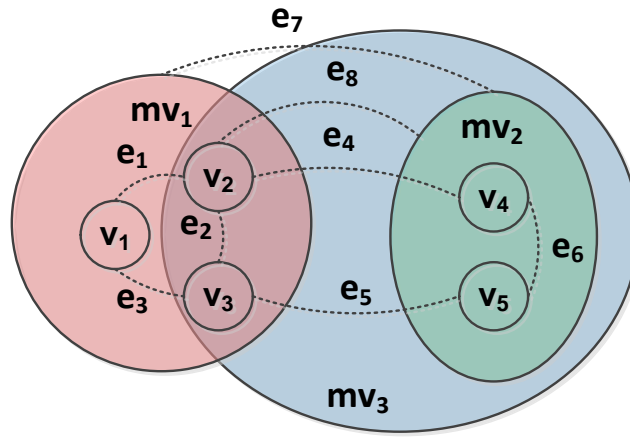


Рис. 3. Пример описания метаграфа.

Пример описания метаграфа показан на рис. 3. Данный метаграф содержит вершины, метавершины и ребра. На рис. 3 показаны три метавершины: mv_1 , mv_2 и mv_3 . 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 метавершину mv_2 , вершины v_2 , v_3 и ребро e_2 из метавершины mv_1 а также ребра e_4 , e_5 , e_8 , что показывает холоническую структуру метаграфа.

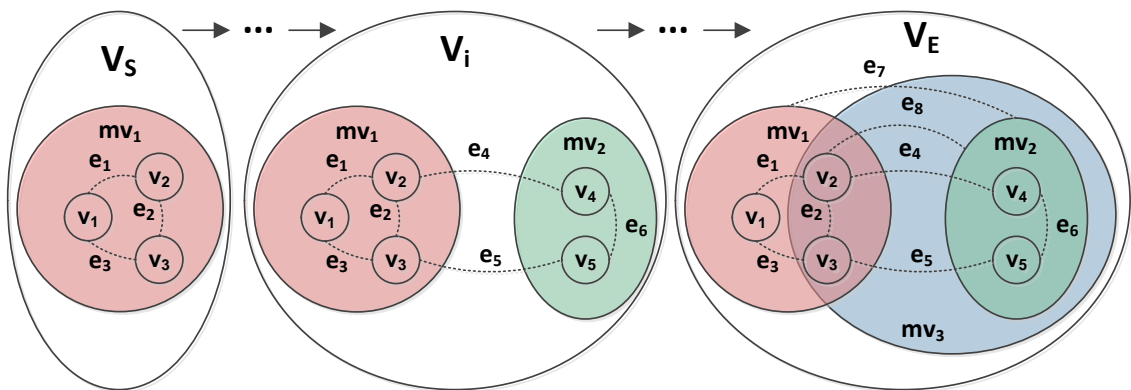


Рис. 4. Пример описания метаребра метаграфа.

Пример описания метаребра метаграфа представлен на рис. 4. Метаребро содержит метавершины v_s , \dots v_i , \dots v_E и связывающие их ребра. Исходная

метавершина содержит фрагмент метаграфа. В процессе преобразования исходной метавершины v_S в конечную метавершину v_E происходит дополнение содержимого метавершины, добавляются новые вершины, связи, вложенные метавершины.

Если метавершины предназначены прежде всего для описания данных и знаний, то метаребра предназначены в большей степени для описания процессов. Таким образом, метаграфовая модель позволяет в рамках единой модели описывать данные, знания и процессы.

Необходимо также отметить, что описание процессов требуется не во всех случаях. Поэтому метаребра применяются только в случае необходимости и их можно рассматривать как необязательный, вспомогательный элемент модели. Во многих примерах в следующих лекциях используются только вершины, ребра и метавершины. Однако мы также рассмотрим примеры, в которых использование метаребер окажется полезным.

Контрольные вопросы

1. Что такое метаграф, какие дополнительные составляющие по сравнению с обычным графом он содержит?
2. Что такое фрагмент метаграфа?
3. Что такое метавершина метаграфа, для каких задач может быть использован этот элемент модели?
4. Что такое метаребро метаграфа, для каких задач может быть использован этот элемент модели?
5. Для чего в модели используется признак направленности?
6. Каким образом метаграфовая модель реализует принцип эмерджентности?
7. Приведите пример метаграфа, содержащий метавершины.
8. Приведите пример метаграфа, содержащий метаребро.

Лекция 4. Сравнение метаграфовой модели с моделями гиперграфа и гиперсети

В этой лекции мы рассмотрим модели гиперграфа и гиперсети и сравним их с метаграфовой моделью.

Рассмотрим формализованную модель гиперграфа в соответствии с [18]:

$$HG = \langle V, HE \rangle, v_i \in V, he_j \in HE,$$

где HG – гиперграф; V – множество вершин гиперграфа; HE – множество непустых подмножеств V , называемых гиперребрами; v_i – вершина гиперграфа; he_j – гиперребро гиперграфа.

Гиперграф может быть направленным или ненаправленным. **Гиперребра** ненаправленного гиперграфа показывают включение вершин, в то время как гиперребра направленного гиперграфа задают порядок обхода вершин.

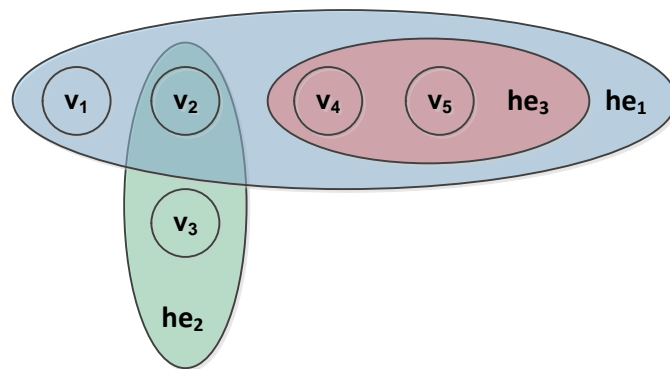


Рис. 5. Пример описания гиперграфа.

Пример описания гиперграфа показан на рис. 5. Данный ненаправленный гиперграф включает три гиперребра: he_1, he_2, he_3 . Гиперребро he_1 включает вершины v_1, v_2, v_4, v_5 . Гиперребро he_2 включает вершины v_2 и v_3 . Гиперребро he_3 включает вершины v_4 и v_5 . Гиперребра he_1 и he_2 имеют общую вершину v_2 . Все вершины гиперребра he_3 также являются вершинами гиперребра he_1 .

Сравнивая рис. 3 и 5, можно отметить некоторые сходства между метавершиной и гиперребром. Но эти сходства достаточно условны. Если гиперребро гиперграфа может включать только вершины, то метавершина метаграфа может включать как вершины (метавершины), так и ребра (метаребра).

В соответствии с определением гиперграфа, операция иерархической вложенности для гиперребер не определена явным образом, можно говорить только о пересечении множеств вершин, вложенных в гиперребра.

Поэтому, хотя гиперграф и содержит гиперребра, но не позволяет моделировать сложные иерархические зависимости и не является полноценной «сетью с эмерджентностью». В отличие от гиперграфа, метаграф позволяет естественным образом моделировать сложные иерархические зависимости и является «сетью с эмерджентностью».

Далее разберем более «богатую» **гиперсетевую модель**, которая строится на основе гиперграфовой модели.

Удивительным является факт, что гиперсетевая модель (с одинаковым названием) была открыта дважды. При этом само изложение материала авторами модели достаточно убедительно свидетельствует об отсутствии возможных заимствований. Сложно даже точно сказать какая модель была предложена ранее, потому что обе модели развивались авторами в течении длительного времени.

Гиперсетевая модель (по В.К. Попкову) предложена профессором В.К. Попковым. Первые публикации можно отнести к 1980-м годам. В качестве математического аппарата используются теория графов и отчасти теория категорий. Модель применяется в различных прикладных задачах, особенно много применений в области организации транспортных перевозок.

Гиперсетевая модель (по Дж. Джонсону) предложена профессором Джеффри Джонсоном. Несмотря на то, что основная монография профессора Дж. Джонсона [19] вышла в 2013 году (значительно позже работ профессора В.К. Попкова), необходимо отметить что профессор Дж. Джонсон в этой монографии ссылается на свои работы еще 1970-х годов, в том числе, опираясь на еще более ранние статьи своего учителя профессора Рональда Аткина. В качестве математического аппарата используются в основном методы топологии (симплициальные комплексы, топологические пространства), метод Q-анализа (предложенный профессором Рональдом Аткиным). Модель применяется в основном в области социологии.

Таким образом, сравнивая модели, можно отметить следующие факты:

1. несмотря на схожую постановку задачи, модели используют различный математический аппарат;
2. области практического применения моделей не пересекаются, при этом профессор Дж. Джонсон подчеркивает в своей монографии, что для него первичными были именно исследования в области социологии, а разработанная модель была только средством для этих исследований;
3. публикации профессора В.К. Попкова на английском языке практически отсутствуют, а результаты профессора Дж. Джонсона до 2013 года публиковались только в виде статей, которые вряд ли широко переводились в СССР и России.

На основе данных фактов можно сделать вывод о том, что гиперсетевая модель действительно была независимо предложена профессорами В.К. Попковым и Джеффри Джонсоном.

Рассмотрим определение **абстрактной гиперсети** (по В.К. Попкову) в соответствии с [20].

Пусть даны гиперграфы $PS \equiv WS_0, WS_1, WS_2, \dots, WS_K$. Гиперграф PS или WS_0 называется первичной сетью. Гиперграф WS_i называется вторичной сетью i -го порядка.

Пусть также задана последовательность отображений между сетями различных уровней $\{\Phi_i\}: WS_K \xrightarrow{\Phi_K} WS_{K-1} \xrightarrow{\Phi_{K-1}} \dots WS_1 \xrightarrow{\Phi_1} PS$.

Тогда иерархическую абстрактную гиперсеть порядка K можно определить как $AS^K = \langle PS, WS_1, \dots, WS_K; \Phi_1, \dots, \Phi_K \rangle$.

Эмерджентность в гиперсети (по В.К. Попкову) возникает при переходе между уровнями за счет использования отображений между «слоями» гиперребер.

Эмерджентность в гиперсети (по Дж. Джонсону) возникает при переходе между уровнями за счет возникновения **гиперсимплексов**.

Интересным является факт, что в своей модели когнитома [9], профессор К.В. Анохин использует модель гиперсети (по Дж. Джонсону).

В монографии [19] профессор Дж. Джонсон дает формальное определение гиперсимплекса через достаточно сложную цепочку определений, которые мы здесь не приводим. В статье [9] профессор К.В. Анохин приводит удачное неформальное определение гиперсимплекса: «основание гиперсимплекса содержит множество элементов одного уровня, а его вершина образуется описанием их отношений и приобретает интегральные свойства, делающие ее элементом сети более высокого уровня».

Примеры гиперсетей по В.К. Попкову и по Дж. Джонсону представлены на рис. 6 и 7.

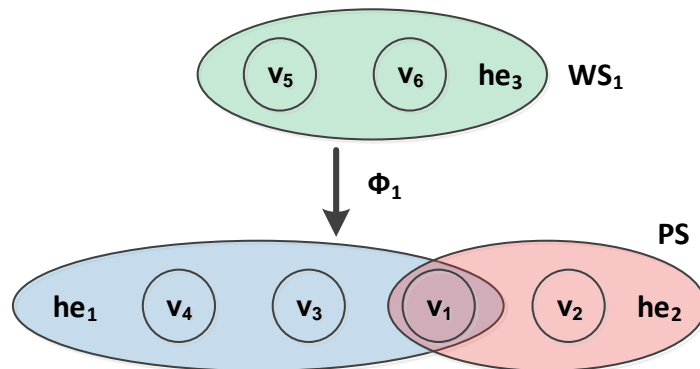


Рис. 6. Пример гиперсети (по В.К. Попкову).

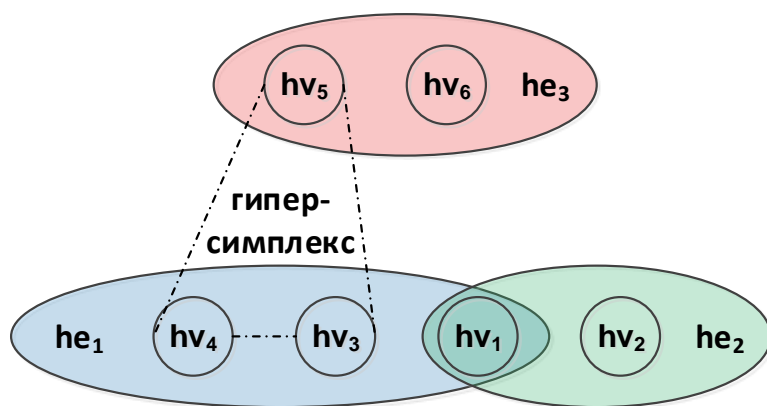


Рис. 7. Пример гиперсети (по Дж. Джонсону).

На рис. 6 первичная сеть PS образуется вершинами гиперребер he_1 и he_2 . Первый уровень вторичной сети образуется вершинами гиперребра he_3 . На рис. 7. штрихпунктирной линией выделен гиперсимплекс $hv_3-hv_4-hv_5$.

Таким образом, в обеих моделях гиперсеть является конструкцией, содержащей слои гиперграфов, при этом только соседние слои могут быть «сцеплены». Разница между моделями состоит в том, как «сцепляются» слои. В модели В.К. Попкова «сцепление» производится за счет отображений Φ . В модели Дж. Джонсона «сцепление» производится за счет использования гиперсимплексов.

Необходимо отметить, что гиперсимплекс, как совокупность элементов различных уровней, в теории метаграфов может быть представлен в виде метавершины (в соответствии с определением метавершины).

Рассмотрим отличия между моделью гиперсети и моделью метаграфа.

В соответствии с определением гиперсеть является «послойным» описанием графов. Предполагается, что слои-гиперграфы идут последовательно и имеют регулярную структуру. Метаграф позволяет с помощью метавершин группировать произвольные элементы, наличие регулярных уровней не обязательно, что делает подход метаграфов более гибким. Фактически, каждый гиперсимплекс может быть представлен отдельной метавершиной.

Гиперсеть состоит из разнородных элементов (гиперграфов и отображений по В.К. Попкову, гиперграфов и гиперсимплексов по Дж. Джонсону). Метаграф позволяет с помощью метавершин обеспечивать связь как между элементами одного уровня, так и между элементами различных уровней (при этом, не обязательно соседних). Это делает метаграфовый подход более унифицированным и удобным в описании, так как для описания используются не разнородные структуры (гиперграфы и отображения), а только метавершины (и связи, как элементы метавершин). Метаграфовый подход позволяет рассматривать сеть не только в виде «горизонтальных» слоев, но и в виде «вертикальных» колонок.

Эмерджентность в гиперсети обеспечивается за счет отображений или гиперсимплексов и фактически возникает только при переходе между соседними уровнями. Эмерджентность в метаграфах обеспечивается за счет использования метавершин и может применяться на одном уровне или между уровнями (не обязательно соседними), что делает реализацию эмерджентности в метаграфах более гибкой.

По результатам сравнения моделей можно сделать следующие выводы:

- Модель гиперсети фактически представляет собой попытку описания «сверху вниз» (по уровням), а модель метаграфа попытку описания «снизу вверх» (путем «выращивания» метавершин из более простых элементов).
- Модель метаграфа является более простой в описании, так как состоит из однородных элементов (метавершин и связей, как элементов метавершин).
- Модель метаграфа является более гибкой, так как не требует регулярности уровней. Произвольный подграф может быть превращен в метавершину.

Тем не менее, необходимо подчеркнуть, что метаграфы и гиперсети являются лишь различными формальными описаниями одних и тех же процессов, которые происходят в «сетях с эмерджентностью».

Также необходимо отметить, что в настоящее время теория гиперсетей является намного более зрелой по сравнению с теорией метаграфов и, благодаря теории гиперсетей, исследователям удалось понять многие аспекты «сетей с эмерджентностью».

В связи с вышеизложенным, именно метаграфовая модель будет использоваться в качестве модели данных ГИИС.

Контрольные вопросы

1. Сформулируйте определение гиперграфа.
2. В чем сходства и различия между гиперребром гиперграфа и метавершиной метаграфа?

3. Почему гиперграфовая модель не позволяет в полной мере реализовать принцип эмерджентности?
4. Дайте определение абстрактной гиперсети (по В.К. Попкову).
5. В чем сходство и различие между гиперсетевыми моделями В.К. Попкова и Дж. Джонсона?
6. Что такое гиперсимплекс, для чего он используется в гиперграфовой модели?
7. Нарисуйте фрагмент гиперсети в соответствии с моделями В.К. Попкова и Дж. Джонсона.
8. В чем сходства и различия между гиперсетевой и метаграфовой моделями?

Лекция 5. Модель холонической МАС, предназначенной для реализации ГИИС

В данной лекции на основе материалов статей [10, 13] мы рассмотрим, каким образом сформулированные требования к холонической МАС могут быть реализованы с использованием метаграфового подхода.

Определим множество агентов системы:

$$AG = \{ag_i\},$$

где AG – множество агентов; ag_i – агент.

Для реализации требования 1 нами предлагается использовать два вида агентов: агент-функцию и метаграфовый агент.

Определим **агент-функцию** следующим образом:

$$ag^F = \langle MG_{IN}, MG_{OUT}, AST \rangle,$$

где ag^F – агент-функция; MG_{IN} – метаграф, который выполняет роль входного параметра агента-функции; MG_{OUT} – метаграф, который выполняет роль выходного параметра агента-функции; AST – абстрактное синтаксическое дерево агента-функции, которое может быть представлено в виде метаграфа.

Определим **метаграфовый агент** следующим образом:

$$ag^M = \langle MG_D, R, AG^{ST} \rangle, R = \{r_j\},$$

где ag^M – метаграфовый агент; MG_D – метаграф данных и знаний, на основе которого выполняются правила агента; R – набор правил (множество правил r_j); AG^{ST} – стартовое условие выполнения агента (фрагмент метаграфа, который используется для стартовой проверки правил, или стартовое правило).

Структура правила метаграфового агента:

$$r_i : MG_j \rightarrow OP^{MG},$$

где r_i – правило; MG_j – фрагмент метаграфа, на основе которого выполняется правило; OP^{MG} – множество операций, выполняемых над метаграфом.

Антецедентом правила является фрагмент метаграфа, консеквентом правила является множество операций, выполняемых над метаграфом.

Множество возможных операций, выполняемых над метаграфом, будет более детально рассмотрено в следующих лекциях.

Отметим, что правила метаграфового агента можно разделить на замкнутые и разомкнутые.

Разомкнутые правила не меняют в правой части правила фрагмент метаграфа, относящийся к левой части правила. Можно разделить входной и выходной фрагменты метаграфа. Данные правила являются аналогом шаблона, который порождает выходной метаграф на основе входного.

Замкнутые правила меняют в правой части правила фрагмент метаграфа, относящийся к левой части правила. Изменение метаграфа в правой части правил заставляет срабатывать левые части других правил. Но при этом некорректно разработанные замкнутые правила могут привести к заикливанию метаграфового агента.

Таким образом, метаграфовый агент позволяет генерировать один метаграф на основе другого (с использованием разомкнутых правил) или модифицировать метаграф (с использованием замкнутых правил).

Пример представления метаграфового агента в виде фрагмента метаграфа приведен на рис. 8.

Метаграфовый агент представлен в виде метавершины метаграфа. В соответствии с определением, он связан с метаграфом MG_1 , на основе которого выполняются правила агента. Данная связь показана с помощью ребра e_4 .

Метаграфовый агент содержит множество вложенных метавершин, соответствующих правилам (правило 1 – правило N). Каждая метавершина правила содержит вершины антецедента и консеквента. В данном примере с антецедентом правила связана метавершина данных mv_2 , что показано ребром e_2 , а с консеквентом правила связана метавершина данных mv_3 , что показано ребром e_3 . Условия срабатывания антецедента и множество действий консеквента задаются в виде атрибутов соответствующих вершин.

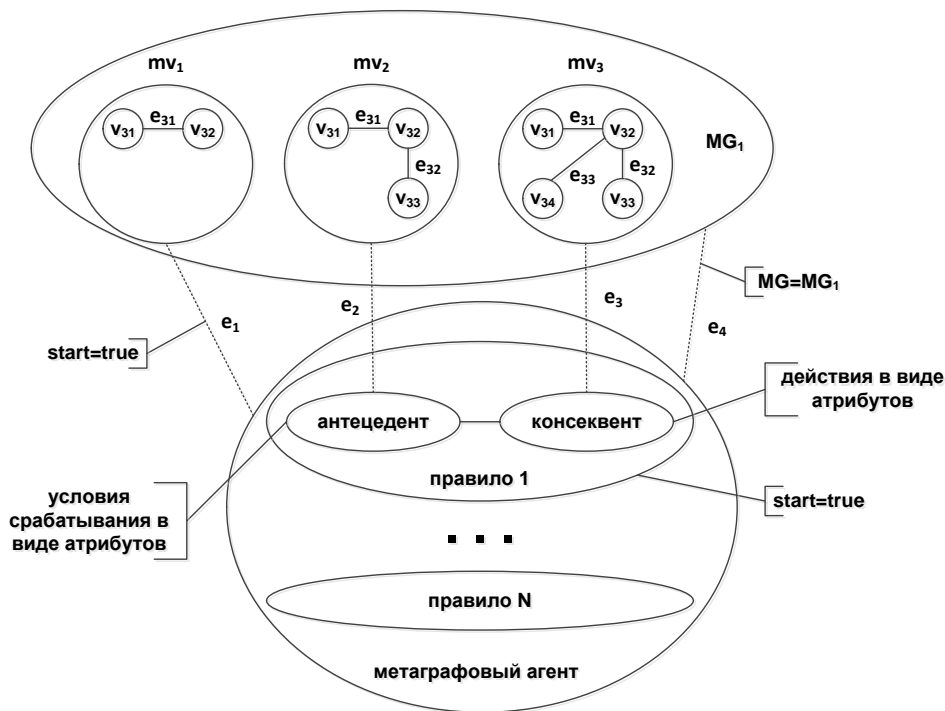


Рис. 8. Представление метаграфового агента в виде фрагмента метаграфа.

Стартовое условие выполнения агента задается с помощью атрибута «start=true». Если стартовое условие задается в виде стартового правила, то данным атрибутом помечается метавершина соответствующего правила, в данном примере это правило 1. Если стартовое условие задается в виде стартового фрагмента метаграфа, который используется для стартовой проверки правил, то атрибутом «start=true» помечается ребро, которое связывает стартовый фрагмент метаграфа с метавершиной агента, в данном примере это ребро e_1 .

Для реализации требования 2 предлагается использовать **контейнерный агент**, который представляет собой метаграф, вершины и метавершины которого являются агентами:

$$ag^C = MG, v_i \equiv ag_i, v_i \in V, mv_i \equiv ag_i, mv_i \in MV,$$

где ag^C – контейнерный агент; MG – метаграф; v_i – вершина метаграфа; ag_i – агент; V – множество вершин метаграфа; mv_i – метавершина метаграфа; MV – множество метавершин метаграфа.

Пример статической структуры ГИИС представлен в виде холонической структуры агентов на рис. 9.

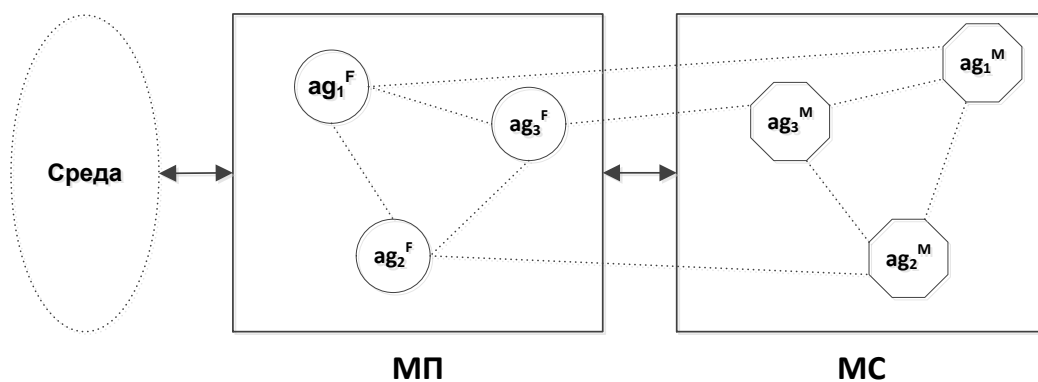


Рис. 9. Пример статической структуры ГИИС.

На рис. 9 представлена система, МП которой является нейронной сетью, а МС построен на основе обработки правил.

Агенты-функции показаны в виде окружностей, а метаграфовые агенты обработки данных в виде восьмиугольников. МП и МС выполняют роль контейнерных агентов. В данном примере используются одноуровневые контейнеры, однако, в соответствии с определением агента-контейнера, возможно использование произвольной вложенности контейнеров.

Отметим, что вся система голонических агентов представляет собой метаграф, при этом каждый агент также является метаграфом.

Для реализации требования 3 нами предлагается использовать **динамический метаграфовый агент**, правила обработки которого выполняются

не на метаграфе данных и знаний, а на метаграфе агентов для заданного контейнерного агента:

$$ag^{MD} = \langle (ag^C \cup ag^{MD}), R, AG^{ST} \rangle, R = \{r_j\},$$

где ag^{MD} – динамический метаграфовый агент; ag^C – контейнерный агент, на метаграфе которого выполняются правила агента; R – набор правил (множество правил r_j); AG^{ST} – стартовое условие выполнения агента (фрагмент метаграфа, который используется для стартовой проверки правил, или стартовое правило).

По определению, контейнерный агент включает все рассмотренные ранее виды агентов: агенты-функции и метаграфовые агенты. Поэтому динамический метаграфовый агент может изменять все виды агентов.

Определение данного агента использует тот факт, что в предлагаемой модели все агенты являются метаграфами, поэтому любые элементы структуры агентов доступны для обработки агентами верхнего уровня. Эта особенность является аналогом свойства «самоотображаемости» в традиционных языках программирования.

Отметим, что данное определение является рекурсивным. Динамические метаграфовые агенты первого уровня могут обрабатывать статические контейнерные агенты, метаграфовые агенты второго уровня могут обрабатывать метаграфовые агенты первого уровня и так далее. По мере необходимости систему можно надстраивать требуемыми уровнями динамики.

Правила динамического метаграфового агента базируются на операциях над метаграфами. В зависимости от условий, динамический метаграфовый агент может решать следующие задачи:

- первичное развертывание, создание системы агентов более низкого уровня;
- изменение системы нижнего уровня (изменение внутренней структуры агентов, изменение связей между агентами, удаление агентов).

Пример динамической структуры ГИИС представлен в виде холонической структуры агентов на рис. 10.

По сравнению с рис. 9 на рис. 10 добавились два динамических метаграфовых агента.

Агент, отвечающий за МП, может изменять структуру самоорганизующейся нейронной сети. Или может применять эволюционные методы для оптимизации конфигурации нейронной сети.

Агент, отвечающий за МС, может изменять связи между агентами для решения задачи автоматизированного планирования. Или может применять эволюционные методы для оптимизации конфигурации агентов.

Динамические метаграфовые агенты могут использовать метазнания, которые также могут быть представлены в виде метаграфа.

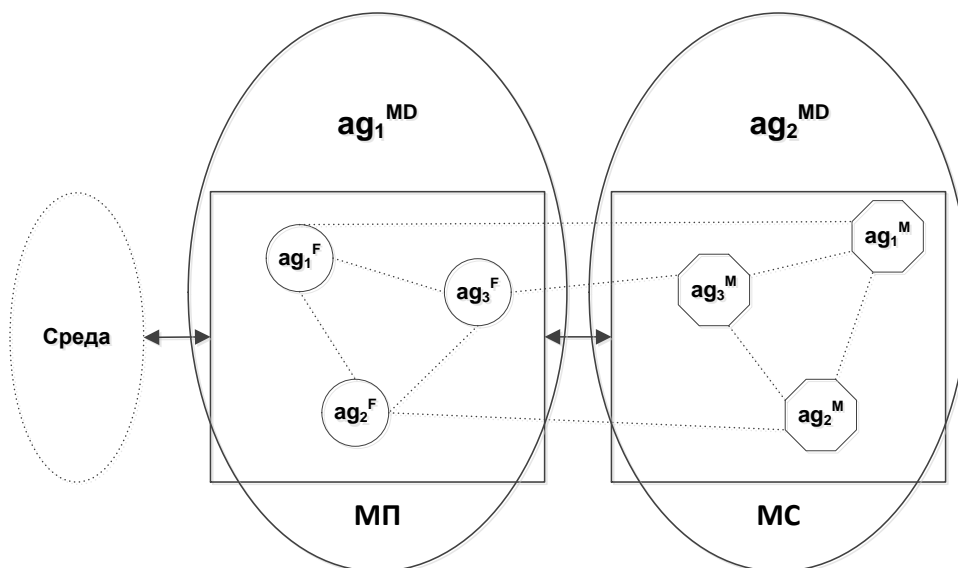


Рис. 10. Пример динамической структуры ГГИИС.

На рис. 10 показаны только динамические метаграфовые агенты первого уровня, однако, в соответствии с определением, количество таких уровней не ограничено. Над показанными динамическими метаграфовыми агентами могут быть надстроены динамические метаграфовые агенты более высоких уровней.

Контрольные вопросы

1. Дайте определение агента-функции.
2. Дайте определение метаграфового агента.
3. В чем разница между замкнутыми и разомкнутыми правилами метаграфового агента?

4. Приведите пример представления метаграфового агента в виде фрагмента метаграфа.
5. Дайте определение контейнерного агента.
6. Приведите пример статической структуры ГИИС на основе контейнерных агентов.
7. Дайте определение динамического метаграфового агента.
8. Приведите пример динамической структуры ГИИС на основе динамических метаграфовых агентов.

Лекция 6. Описание метаграфовой модели с использованием информационных элементов метаграфа

Определение метаграфа не дает полной информации о том, каким образом хранить метаграфы в информационной системе. Модель метаграфа является достаточно высокоуровневой моделью представления данных и знаний. С точки зрения СУБД такую модель можно рассматривать как «логическую» модель. Но для использования метаграфов в информационных системах необходимо также предложить варианты «физической» модели, с помощью которой метаграфовые данные можно хранить в СУБД.

В этой лекции мы рассмотрим первый вариант «физической» модели на основе информационных элементов метаграфа. Этот вариант позволяет сохранить метаграф в графовой или объектно-ориентированной СУБД.

В следующей лекции мы рассмотрим вариант «физической» модели на основе предикатного описания, который позволяет сохранить метаграф в документно-ориентированной СУБД.

Рассмотрим представление метавершины в виде графа, показанное на рис. 11.

На основании рис. 11 метавершину можно представить в виде комбинации обычной вершины и набора связей от данной вершины к другим вершинам (или метавершинам) и ребрам.

Поскольку метавершина содержит связи не только с другими вершинами, но и с ребрами, то ребро в этом случае удобно рассматривать в виде вершины

отдельного класса, которая имеет неименованные связи с вершинами (метавершинами).

Таким образом, вершины и ребра образуют двудольный граф с той поправкой, что если вершина является метавершиной, то она содержит дополнительные иерархические связи с другими вершинами (метавершинами) и ребрами. Если выделить метавершины в отдельный класс, то фактически граф становится трехдольным.

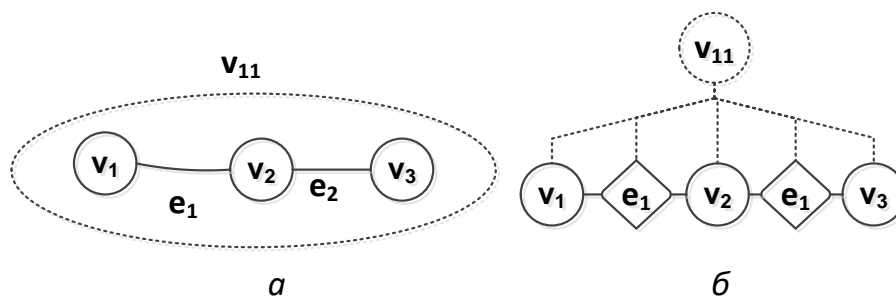


Рис. 11. Метавершина (показана пунктиром) в виде фрагмента метаграфа (а) и ее представление в виде графа (б).

Если метаграф является аннотированным, то аннотации могут быть заданы для вершин (метавершин) и ребер, которые показаны на рис. 11б в виде окружностей и ромбов, но не могут быть заданы для «примитивных» связей, которые показаны на рис. 11б в виде линий.

Фактически рис. 11 показывает вариант преобразования метаграфа к плоскому графу. На основании рис. 11б можно сделать вывод о том, что СУБД на основе графовой модели могут использоваться для хранения метаграфов, но при этом метавершины и ребра должны быть искусственно преобразованы к вершинам и соединены неименованными связями. В частности, одна из наиболее зрелых графовых СУБД Neo4J, которая обладает развитым языком графовых запросов, окажется в этом случае не очень эффективной по двум причинам:

1. резко увеличивается количество хранимых вершин;
2. язык запросов оказывается неэффективным, так как он предполагает написание запроса для графа с именованными связями, а в нашем случае именованные связи метаграфа будут представлены в виде

вершин, связи же в СУБД (соединяющие вершины и связи метаграфа, представленные в виде вершин в СУБД) окажутся неименованными.

Таким образом, в случае представления вершин (метавершин) и ребер с помощью двудольного (трехдольного) графа, вершины и ребра представляют собой относительно похожие структуры.

Предлагается моделировать эти похожие структуры в виде единой структуры данных. Назовем такую единую структуру данных «информационным элементом метаграфа» (ИЭМ). ИЭМ является элементарной строительной единицей для конструирования метаграфов, описывающих семантику информационной системы.

Рассмотрим формализованную модель предлагаемого ИЭМ. Представим ИЭМ в виде следующего кортежа:

$$\text{ИЭМ} = \langle id, NM, VAL, RL, \{lnk_i\}, \{atr_j\} \rangle, RL \in \{RL_V, RL_{MV}, RL_R\},$$

где id – уникальный идентификатор элемента; NM – наименование элемента; VAL – значение элемента; RL – роль элемента; lnk_i – ссылка на другой ИЭМ; atr_j – атрибут; RL_V – роль элемента «вершина»; RL_{MV} – роль элемента «метавершина»; RL_R – роль элемента «ребро».

Таким образом, ИЭМ может использоваться для хранения вершины, метавершины и ребра метаграфа. С использованием ссылок lnk_i реализуются нетипизированные связи между вершиной и ребром, а также связи между метавершиной и входящими в нее элементами.

С учетом предложенной модели ИЭМ можно определить метаграф как множество входящих в него ИЭМ:

$$MG = \{\text{ИЭМ}_i\}$$

Атрибут в предлагаемой модели является метавершиной метаграфа и может принадлежать одному из трех видов: ссылочный, типизированный, нетипизированный:

$$atr_j \stackrel{\text{def}}{=} MG, atr_j \in \{ATRT_{REF}, ATRT_T, ATRT_{NT}\},$$

где atr_j – атрибут; $ATRT_{REF}$ – вид атрибута «ссылочный»; $ATRT_T$ – вид атрибута «типизированный»; $ATRT_{NT}$ – вид атрибута «нетипизированный».

Детализированная структура атрибутов трех видов представлена на рис. 12.

Ссылочный атрибут используется для ссылок на произвольный фрагмент метаграфа.

Типизированный атрибут содержит имя, значение и тип данных. Тип данных приписан имени атрибута, атрибут может содержать только значения заданного типа.

Нетипизированный атрибут также содержит имя, значение и тип данных. Тип данных приписан значению атрибута, атрибуту можно присваивать значения различных типов. Тип данных атрибута меняется в зависимости от типа присвоенного значения.

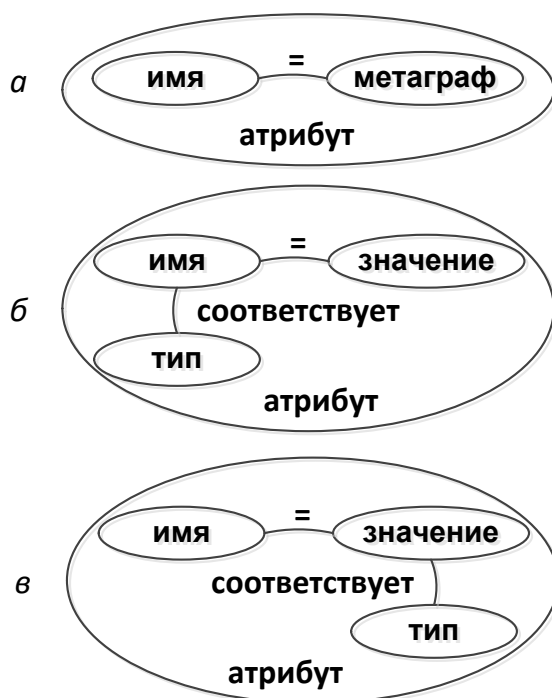


Рис. 12. Виды атрибутов: ссылочный (а), типизированный (б), нетипизированный (в).

Множество типов атрибутов может быть задано в зависимости от потребностей предметной области.

Атрибуты позволяют аннотировать элементы метаграфа текстовой, числовой информацией, информацией других типов.

Поскольку атрибут является метавершиной, то при необходимости он может быть аннотирован другими метавершинами.

Для динамической работы с метаграфами введем следующие основные операции над метаграфами OP^{MG} на основе информационных элементов метаграфов.

1) Создание нового ИЭМ:

$$ИЭМ = ИЭМ(\langle NM, VAL, RL, \{lnk_i\}, \{atr_j\} \rangle).$$

В результате выполнения операции создается новый ИЭМ с заданными параметрами, уникальный идентификатор id генерируется автоматически.

2) Добавление ссылки на ИЭМ:

$$ИЭМ_i = ИЭМ_i + lnk_j(ИЭМ_j).$$

В результате выполнения операции к информационному элементу $ИЭМ_i$ добавляется ссылка на информационный элемент $ИЭМ_j$.

3) Удаление ссылки на ИЭМ:

$$ИЭМ_i = ИЭМ_i - lnk_j(ИЭМ_j).$$

В результате выполнения операции у информационного элемента $ИЭМ_i$ удаляется ссылка на информационный элемент $ИЭМ_j$.

4) Добавление атрибута:

$$ИЭМ_i = ИЭМ_i + atr_j.$$

В результате выполнения операции к информационному элементу $ИЭМ_i$ добавляется атрибут atr_j .

5) Удаление атрибута:

$$ИЭМ_i = ИЭМ_i - atr_j.$$

В результате выполнения операции у информационного элемента $ИЭМ_i$ удаляется атрибут atr_j .

6) Добавление информационного элемента к метаграфу:

$$MG_2 = MG_1 + ИЭМ_i.$$

В результате выполнения операции к метаграфу добавляется ИЭМ.

7) Удаление информационного элемента из метаграфа:

$$MG_2 = MG_1 - ИЭМ_i.$$

В результате выполнения операции у метаграфа удаляется ИЭМ.

8) Создание переменной-ссылки на фрагмент метаграфа:

$$VAR_i = \text{ref}(MG_j).$$

В результате выполнения операции создается переменная, которая содержит ссылку на фрагмент метаграфа.

9) Создание переменной-копии фрагмента метаграфа:

$$VAR_i = \text{copy}(MG_j).$$

В результате выполнения операции создается переменная, которая содержит копию фрагмента метаграфа.

10) Поиск ИЭМ по заданным параметрам:

$$MG_i = \text{find}(\langle \text{id}, \text{NM}, \text{VAL}, \text{RL}, \{\text{lnk}_i\}, \{\text{atr}_i\} \rangle).$$

В качестве параметров поиска могут использоваться любые параметры ИЭМ. В результате выполнения операции формируется метаграф, который представляет собой множество найденных ИЭМ.

11) Оператор цикла:

$$\text{foreach}(MG_i) \rightarrow \{OP_j^{MG}\}$$

В цикле для каждого ИЭМ, входящего в состав метаграфа MG_i , выполняется заданное множество действий OP_j^{MG} . Для связи текущего обрабатываемого ИЭМ с выполняемыми в цикле действиями могут использоваться переменные.

12) Оператор создания функции:

$$\text{function}(\{FP_i\}) \rightarrow \{OP_j^{MG}\}$$

Функция с заданным множеством параметров FP_i позволяет сгруппировать множество операторов OP_j^{MG} . Параметры функции могут использоваться в

качестве параметров операторов OP_j^{MG} . В качестве примера можно рассмотреть функцию, которая принимает на вход элементы-вершины и формирует элемент-связь между этими вершинами.

Предложенные операторы могут быть достаточно просто реализованы в виде API (программного интерфейса) практически в любом современном языке программирования. При этом можно использовать уже готовые языковые механизмы работы с переменными и создания функций.

Таким образом, СУБД на основе графовой модели могут быть использованы для хранения метаграфов, но при этом необходимо учитывать рассмотренные выше особенности таких СУБД. В этом случае мы фактически «разворачиваем» метаграф в плоский граф, и графовая СУБД может его хранить, но при этом метаграфовая семантика становится сложно выразимой в языках запросов графовых СУБД.

С использованием информационных элементов метаграфов, фактически, метаграфовая модель описывается в объектно-ориентированном виде. Поэтому с использованием ИЭМ метаграфы можно сохранять в объектно-ориентированных СУБД, а также в реляционных СУБД.

Модель ИЭМ также позволяет предложить унифицированный способ определения основных операции над метаграфами, что используется для задания правил метаграфового агента.

Контрольные вопросы

1. Как метаграф можно представить в виде двудольного (трехдольного) графа?
2. Что такое информационный элемент метаграфа (ИЭМ)?
3. Дайте формальное определение ИЭМ.
4. Чем отличаются ссылочный, типизированный и нетипизированный атрибуты?
5. Каким образом ссылочный, типизированный и нетипизированный атрибуты могут быть представлены в виде фрагментов метаграфа?

6. Перечислите основные операции над метаграфами (OP^{MG}) на основе информационных элементов метаграфов.
7. Каковы основные недостатки СУБД на основе графовой модели для хранения метаграфов?
8. Можно ли использовать объектно-ориентированную СУБД для хранения метаграфовой модели?

Лекция 7. Предикатное описание метаграфовой модели

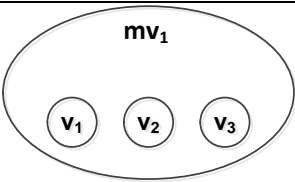
В этой лекции мы рассмотрим вариант «физической» модели метаграфа на основе предикатного описания, базируясь на статьях [10, 21].

Классическим примером предикатного языка является язык Пролог, который использует следующую форму предикатного описания:
 $predicate(atom_1, atom_2, \dots, atom_N)$.


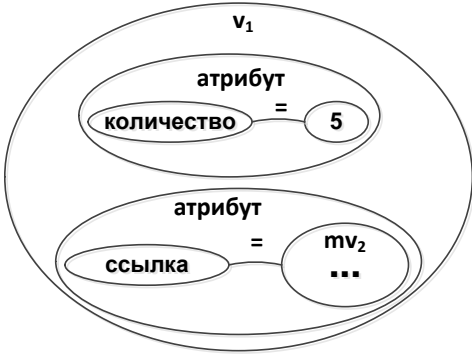
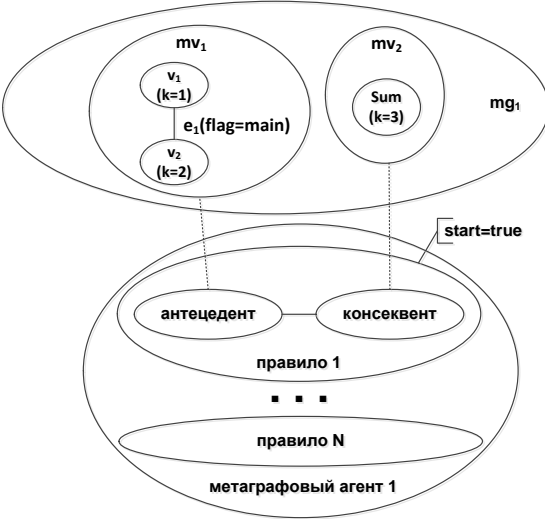
Нами предлагается использовать расширенную форму предикатного описания: $predicate(atom, \dots, key = value, \dots, predicate(\dots), \dots)$. Данная форма, в дополнение к атомам, может также содержать пары ключ-значение и вложенные предикаты.

Интересной особенностью метаграфовой модели является то, что она изоморфна системе предикатов высших порядков. Рассмотрим более подробно различные варианты отображения элементов метаграфовой модели в предикатное описание, которые представлены в таблице 1.

Таблица 1. Соответствие фрагментов метаграфовой модели предикатному описанию.

Вариант	Фрагмент метаграфа	Предикатное представление
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		Metaedge(Name= me_1 , $v_S=v_2$, $v_E=mv_3$, Metavertex(Name= mv_4 , ...), $eo=true$)
8		Metagraph(Name= mg_0 , Vertex(Name= v_2 , ...), Metavertex(Name= mv_3 , ...), Metavertex(Name= mv_5 , ...), Metaedge(Name= me_1 , $v_S=v_2$, $v_E=mv_3$, Metavertex(Name= mv_4 , ...), $eo=true$))

9		Attribute(количество, 5)
10		Vertex(Name=v ₁ , Attribute(количество, 5), Attribute(ссылка, mv ₂))
11		Agent(Name='метаграфовый агент 1', WorkMetagraph=mg ₁ , Rules(Rule(Name='правило 1', start=true, Condition(WorkMetagraph=mv ₁ , Vertex(Name=v ₁ , Attribute(k, \$k1)), Vertex(Name=v ₂ , Attribute(k, \$k2)), Edge(v ₁ , v ₂ , Attribute(flag, main))) Action(WorkMetagraph=mv ₂ , Add(Vertex(Name=Sum, Attribute(k, Eval(\$k1+\$k2)))))) , Rule(...) ...))

Метавершина изоморфна предикату. В таблице 1 (вариант 1) показан пример метавершины mv₁, который содержит три вложенных несвязанных вершины v₁, v₂ и v₃. Предикат соответствует метавершине, вершины изоморфны переменным, которые являются параметрами предиката. В качестве имени предиката используется соответствующий элемент метаграфовой модели (в случае метавершины «Metavertex»), имя метавершины задается именованным параметром Name. Данный случай является простейшим, поскольку вложенные вершины не связаны друг с другом, и метавершина в этом случае изоморфна гиперребру гиперграфа.

Ребро метаграфа можно рассматривать как частный случай метавершины, которая включает исходную и конечную вершину. Пример для ненаправленного ребра показан в виде варианта 2. В предикатном описании в этом случае ребро можно рассматривать как предикат, а исходную и конечную вершину как параметры предиката. Для ребра используется имя предиката «Edge».

Пример ненаправленного ребра, который полностью соответствует формальному определению, показан в виде варианта 3. В этом случае метавершина помечается соответствующей аннотацией направленности, а в предикат добавляется именованный параметр, соответствующий признаку направленности.

Пример направленного ребра показан в виде варианта 4. В предлагаемой предикатной модели все параметры могут быть именованными. В варианте 4 (вариант предикатного представления 2) именованные параметры соответствуют параметрам из формального определения ребра метаграфа.

Метавершина, содержащая вершины и ребра, может быть представлена с использованием предикатов высших порядков. Вариант 5 содержит пример метавершины с ненаправленными ребрами, а вариант 6 – пример метавершины с направленными ребрами. В соответствии с определением метаграфа, предикаты, отвечающие за вершины и ребра, вложены на одном уровне в предикат метавершины.

Метаребро метаграфа также может быть представлено с помощью предикатов высших порядков. Вариант 7 содержит пример метаребра me_1 , исходной вершиной которого является вершина v_2 , конечной метавершиной mv_3 , при этом метаребро содержит вложенную метавершину mv_4 (детальное содержимое вершины и метавершин не показано, чтобы не загромождать пример). Для метаребра используется имя предиката «Metaedge».

Фрагмент метаграфа может содержать произвольное количество вершин (метавершин) и ребер (метаребер) метаграфа. Вариант 8 содержит пример фрагмента метаграфа mg_0 , содержащего вершину v_2 , метавершины mv_3 и mv_5 ,

метаребро me_1 . Для фрагмента метаграфа используется имя предиката «Metagraph», для вершины имя предиката «Vertex».

Атрибут может быть представлен как частный случай метавершины, содержащий имя и значение. Вариант 9 показывает пример атрибута, содержащего целое число. Вариант 10 показывает пример вершины v_1 , содержащей атрибут из примера 9, а также атрибут, ссылающийся на метавершину mv_2 . Для атрибута используется имя предиката «Attribute».

Метаграфовый агент также может быть представлен в виде предикатного описания. Вариант 11 показывает пример метаграфового агента с именем «метаграфовый агент 1» (предикат «Agent»). Рабочим метаграфом является mg_1 (параметр «WorkMetagraph»). Описание правил содержит предикат «Rules». Правилу соответствует предикат «Rule». Стартовым правилом (параметр «start=true» предиката «Rule») является «правило 1» (остальные правила не показаны, чтобы не загромождать пример). Условию правила соответствует предикат «Condition». Параметр «WorkMetagraph» содержит ссылку на проверяемую метавершину mv_1 . Условие проверяет, что метавершина mv_1 содержит вершины v_1 и v_2 , содержащие атрибуты «k». Найденные значения атрибутов «k» помещаются в переменные $\$k1$ и $\$k2$. Вершины v_1 и v_2 должны быть соединены ребром, содержащим атрибут «flag=main». Если условие выполняется, и найденный фрагмент метаграфа найден, то выполняется действие правила, которому соответствует предикат «Action». Параметр «WorkMetagraph» содержит ссылку на результирующую метавершину mv_2 . В действии выполняется добавление новых элементов (предикат «Add»). Добавляется вершина Sum, содержащая атрибут «k= $\$k1+\$k2$ ». Для обозначения вычисляемого выражения используется предикат «Eval».

Отметим, что предложенное описание обладает свойством самоотображаемости. Поскольку и для описания данных, и для описания агентов используется предикатный подход, то предложенный способ позволяет агентам верхнего уровня модифицировать структуру агентов нижнего уровня на основе модификации предикатных описаний. Данная особенность является одним из

необходимых условий разработки гибридных интеллектуальных информационных систем.

Контрольные вопросы

1. В чем состоит идея предикатного описания?
2. Как представить метавершину в форме предикатного описания?
3. Как представить вершину с атрибутами в форме предикатного описания?
4. Как представить ненаправленную связь в форме предикатного описания?
5. Как представить направленную связь в форме предикатного описания?
6. Как представить метаребро в форме предикатного описания?
7. Как представить фрагмент метаграфа в форме предикатного описания?
8. Как представить метаграфовый агент в форме предикатного описания?

Лекция 8. Примеры описания информационных систем на основе метаграфовой модели

В этой лекции мы рассмотрим примеры использования метаграфового подхода в ГИИС. В первой части лекции рассмотрим описание структуры нейронной сети с помощью метаграфов на основе статьи [22]. Во второй части лекции рассмотрим использование метаграфов в обучающей программе по формированию пептидной цепи на основе статьи [23].

Описание структуры нейронной сети с помощью метаграфов

В соответствии с [10] по уровню активности метаграфовые структуры можно разделить на три уровня:

1. Представление данных в виде метаграфа; такой элемент не обладает самостоятельным поведением и предназначен для описания данных.
2. Агент-функцию, который, в частности, реализует поведение элементарного нейрона.
3. Метаграфовый агент, который реализует создание, изменение и обучение нейросети.

Для описания персептрона в виде агента-функции необходимо предварительно рассмотреть описание персептрона с использованием метаграфового подхода, что показано на рис. 13.

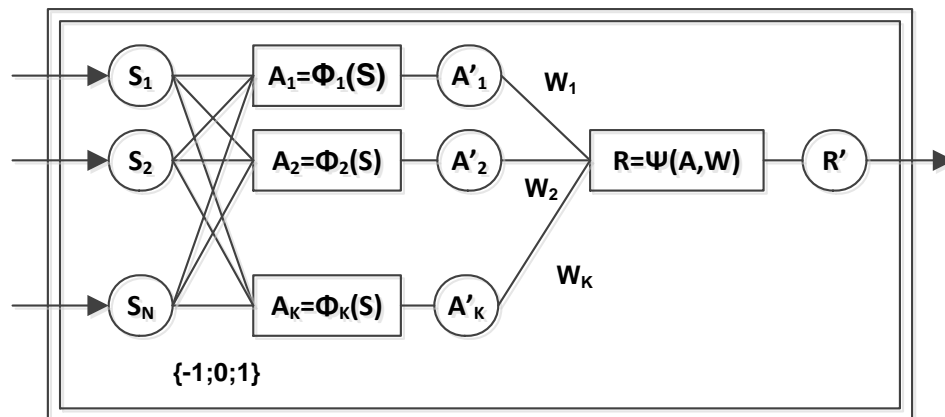


Рис. 13. Описание персептрона с использованием метаграфового подхода.

В соответствии с моделью Ф. Розенблатта [24], классический персептрон состоит из S, A и R элементов.

Слой сенсоров (S) представляет собой набор входных сигналов. Ассоциативный слой (A) включает набор промежуточных элементов, которые активизируются, если одновременно активизируется некоторый набор (образ) входных сигналов. Сумматор (R) активизируется, если одновременно активизируется некоторый набор A-элементов.

В соответствии с обозначениями, принятыми в [25], значение сигнала на A-элементе может быть представлено в виде предиката $\phi(S)$, а значение сигнала на сумматоре в виде предиката $\psi(A, W)$. Под предикатом в [25] понимается функция, принимающая только два значения «0» и «1».

В зависимости от конкретного вида персептрона, вид предикатов $\phi(S)$ и $\psi(A, W)$ может быть различным. Как правило, с помощью предиката $\phi(S)$ проверяется, что суммарный входной сигнал от сенсоров не превышает некоторый порог. Также с помощью предиката $\psi(A, W)$ проверяется, что взвешенная сумма от A-элементов не превышает некоторого порога, где W – вектор весов.

В нашем случае конкретный вид предикатов не важен, важно то, что $\phi(S)$ и $\psi(A,W)$ являются обычными функциональными зависимостями, которые на программном уровне могут быть представлены в виде абстрактного синтаксического дерева.

С точки зрения метаграфов, описание персептрона представляет собой метавершину (показана на рис. 13 в виде двойного прямоугольника). Входные сигналы S , промежуточные сигналы A' и выходной сигнал R' (пассивные данные) показаны на рис. 13 в виде окружностей. Агенты-функции, соответствующие A -элементам и R -элементам, показаны на рис. 13 в виде прямоугольников.

Представим рассмотренную структуру персептрона в виде комбинации агентов-функций, что показано на рис. 14.

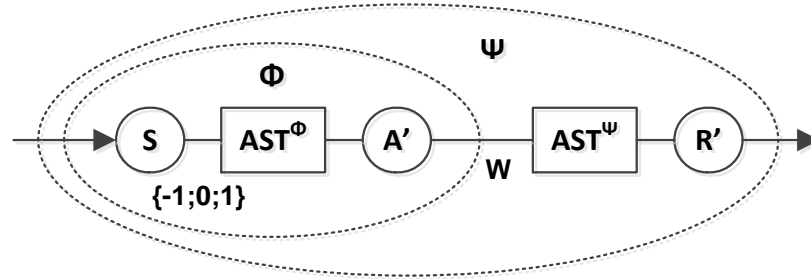


Рис. 14. Представление персептрона в виде агентов-функций.

Предикаты $\phi(S)$ и $\psi(A,W)$ представлены в виде метавершин (показаны на рис. 14 в виде пунктирных овалов). На основе рис. 14 можно описать персептрон в виде комбинации агентов-функций:

$$\phi^F = \langle S, A', AST^\phi \rangle, \psi^F = \langle \langle \phi^F \rangle, W \rangle, R', AST^\psi \rangle.$$

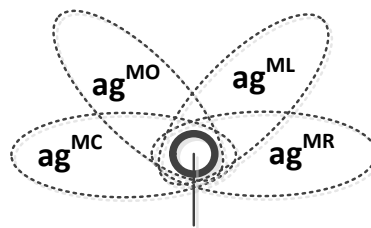
A -элемент может быть представлен в виде агента-функции ϕ^F . Входным параметром является вектор значений S , выходным параметром вектор значений A' .

Описание персептрона эквивалентно описанию агента-функции ψ^F . Входным параметром является метаграфовое представление кортежа, содержащего описание A -элементов в виде агентов-функций ϕ^F и вектора W . Выходным параметром является значение выходного сигнала R' .

Описание функций может содержать и другие параметры, например, пороги, но мы предполагаем, что эти параметры входят в описание абстрактного синтаксического дерева.

Для динамических действий с нейронной сетью используется метаграфовый агент. Для метаграфового агента структура персептрона представляется в виде метаграфа данных. В нашем случае используются следующие агенты:

1. ag^{MC} – агент создания нейросети;
2. ag^{MO} – агент изменения нейросети;
3. ag^{ML} – агент обучения нейросети;
4. ag^{MR} – агент запуска нейросети.



Представление нейронной сети в виде метаграфа

Рис. 15. Структура метаграфовых агентов для действий с нейронной сетью.

На рис. 15 агенты показаны в виде метавершин с помощью пунктирных овалов.

Агент создания нейросети (ag^{MC}) реализует правила создания начальной топологии нейросети. Данный агент содержит как правила создания отдельных нейронов, так и правила соединения нейронов в нейросеть. Данный агент, в частности, создает абстрактные синтаксические деревья агентов-функций ϕ^F и ψ^F .

Агент изменения нейросети (ag^{MO}) содержит правила изменения топологии сети в процессе работы. Это особенно важно для сетей с изменяемой топологией таких, как SOINN.

Агент обучения нейросети (ag^{ML}) реализует один из алгоритмов обучения. При этом в результате обучения измененные значения весов записываются в

метаграфовое представление нейросети. Возможна реализация нескольких алгоритмов обучения с использованием различных наборов правил для агента ag^{ML} .

Агент запуска нейросети (ag^{MR}) реализует запуск и работу обученной нейросети в штатном режиме.

Отметим, что агенты могут работать как независимо, так и совместно. При обучении сети SOINN агент ag^{ML} может вызывать правила агента ag^{MO} для изменения топологии в процессе обучения.

Каждый агент на основе заложенных в него правил фактически реализует специфическую программную «машину». Использование метаграфового подхода позволяет реализовать принцип «мультимашинности», когда несколько агентов с различными целями реализуют различные действия на одной и той же структуре данных.

Таким образом, с помощью метаграфового подхода реализуется описание нейронной сети, а также создание, изменение, обучение и штатная работа нейронной сети.

Использование метаграфов в обучающей программе по формированию пептидной цепи

Молекулярная биология по праву считается одним из наиболее сложных для изучения разделов биологической науки. С точки зрения информатики, сложность происходящих в биологических клетках информационных процессов может во много раз превышать сложность бизнес-процессов любой крупной информационной системы. Трудно поверить, что сложность функционирования невидимой человеческому глазу биологической клетки превышает сложность функционирования крупной ERP-системы, которая может содержать тысячи бизнес-процессов. Трудность изучения биологических процессов также обусловлена тем, что при изучении нельзя абстрагироваться от физических и химических особенностей, сопровождающих эти процессы. Поэтому разработка обучающих программ, помогающих лучше понять даже один сложный процесс, является актуальной задачей.

В качестве такого процесса в данной лекции рассматривается процесс формирования пептидной цепи, которая формируется в результате «трансляции» – процесса синтеза белка из аминокислот на матрице информационной РНК (иРНК), осуществляемый рибосомой [26]. Синтез белка является основой жизнедеятельности клетки, поэтому процесс формирования пептидной цепи широко изучается в различных учебных курсах по молекулярной биологии.

Для синтеза белка в клетках всех без исключения организмов существуют специализированные органеллы – рибосомы, которые состоят из двух субъединиц: большой и малой. С информационной точки зрения цепочку иРНК можно рассматривать как каркас процесса, а рибосомы – как специфические «токены», которые движутся по этому каркасу. В процессе движения происходит синтез белка.

Пептид представляет собой «заготовку» белка. В молекулярной биологии белком обычно называется длинная пептидная цепь (полипептид), которая содержит от 50 аминокислотных остатков и имеет определенную молекулярную массу.

Трансляция является процессом, состоящим из инициации, элонгации и терминации. Более подробно в соответствии с [26] шаги процесса могут быть представлены следующим образом.

На этапе инициации происходит сборка рибосомы из большой и малой субъединиц и узнавание рибосомой стартового кодона (начального участка) иРНК.

На этапе элонгации рибосома с помощью малой субъединицы последовательно считывает кодоны (фрагменты) иРНК. Считывание кодона сопровождается синтезом пептидной цепи с помощью большой субъединицы путем последовательного добавления аминокислотных остатков к растущей цепи, происходит элонгация (удлинение) пептида.

Этап терминации связан с узнаванием рибосомой стопового кодона иРНК. На этом этапе производится отсоединение синтезированного белка, в некоторых случаях может происходить диссоциация рибосомы.

Укрупненная схема модели на основе метаграфового подхода представлена на рис. 16.

Цепочка иРНК изображена на рис. 16 в виде метаребра $me_{PHK} = \langle \{C_i\}, \langle \{C_S, C_E\} \rangle \rangle$, которое характеризуется множеством метавершин-кодонов C_i и множеством связей между парами кодонов, где C_S – начальный кодон связи, C_E – конечный кодон связи; C_{START} – стартовый кодон иРНК; C_{STOP} – стоповый кодон иРНК; C_K – промежуточный кодон иРНК, участвующий в этапе элонгации.

С точки зрения метаграфовой модели данных метавершина-кодон является сложным графом, который содержит вложенные вершины и связи с нужным уровнем детализации, но это не показано на рис. 16, чтобы не загромождать рисунок.

Рибосома моделируется с помощью метаграфового агента $ag^M = \langle me_{PHK}, C_{START}, R \rangle$, $R = \{r_j\}$, $r_i : C_K \rightarrow P_K$, где ag^M – метаграфовый агент; me_{PHK} – метаребро, на основе которого выполняются правила агента; C_{START} – стартовый кодон, который используется для стартовой проверки правил; R – множество продукционных правил r_j , где левой частью правила является проверяемый кодон C_K , а правой частью правила является добавляемый фрагмент пептидной цепи P_K . Левая часть правила условно соответствует малой субъединице, а правая часть правила условно соответствует большой субъединице рибосомы.

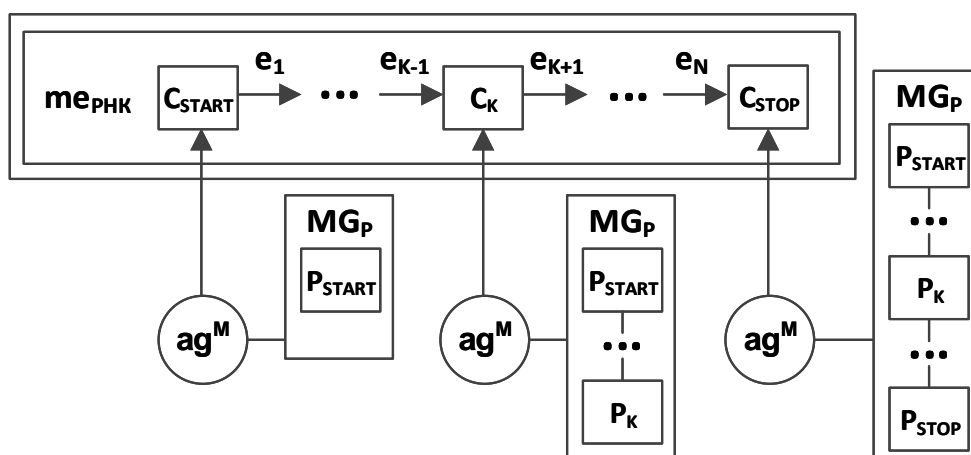


Рис. 16. Укрупненная схема модели формирования пептидной цепи на основе метаграфового подхода.

Агент ag^M является разомкнутым, он генерирует выходной метаграф MG_P на основе входного метаребра me_{PHK} , при этом входной и выходной метаграфы не пересекаются. По мере прохождения кодонов, в выходной метаграф MG_P по очереди добавляются фрагменты пептидной цепи (P_{START} , ..., P_K , ..., P_{STOP}), связанные между собой ненаправленными связями.

Необходимо отметить, что метаграфовый подход позволяет моделировать связанные процессы на нескольких уровнях иерархии. Высокоуровневый процесс, представленный на рис. 16, может быть параллельно представлен на более детальных уровнях химических органических соединений, где отдельные молекулы кодируются метавершинами. Химические реакции могут моделироваться с использованием специализированных метаграфовых агентов.

Таким образом, метаграфовый подход позволил сформировать модель, которая может являться основой для обучающей программы по формированию пептидной цепи.

Контрольные вопросы

1. Как можно представить персептрон в виде комбинации агентов-функций?
2. Как использование метавершин упрощает описание персептрона?
3. Как можно описать нейронную сеть на основе комбинации метаграфовых агентов?
4. Как используется агент создания нейросети?
5. Как используется агент изменения нейросети?
6. Как используется агент обучения нейросети?
7. Как используется агент запуска нейросети?
8. Как можно описать модель формирования пептидной цепи на основе метаграфового подхода?
9. Как использование метаребер упрощает описание модели формирования пептидной цепи?
10. Как можно моделировать работу рибосомы с использованием метаграфового агента?

Список используемых источников

1. Колесников А.В. Гибридные интеллектуальные системы. Теория и технология разработки. – СПб.: СПбГТУ, 2001. – 137 с.
2. Колесников А.В., Куриков И.А., Листопад С.В. Гибридные интеллектуальные системы с самоорганизацией: координация, согласованность, спор. – М.: ИПИ РАН, 2014. – 189 с.
3. Рыбина Г.В., Паронджанов С.С. Технология построения динамических интеллектуальных систем: Учебное пособие. М.: НИЯУ МИФИ, 2011.
4. Zadeh L.A., Abbasov A.M., Yager R.R., Shahbazova S.N., Reformat M.Z. Recent Developments and New Directions in Soft Computing. Springer-Verlag, 2014.
5. Melin P., Castillo O., Kacprzyk J. Nature-Inspired Design of Hybrid Intelligent Systems. Springer-Verlag, 2017.
6. Прикладные интеллектуальные системы, основанные на мягких вычислениях. / под ред. Н.Г. Ярушкиной. – Ульяновск: УлГТУ, 2004. – 139 с.
7. Евин И.А. Введение с теорию сложных сетей // Компьютерные исследования и моделирование. 2010, Том 2, №2, С. 121-141.
8. 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.
9. 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.
10. Черненький В.М., Гапанюк Ю.Е., Ревунков Г.И., Терехов В.И., Каганов Ю.Т. Метаграфовый подход для описания гибридных интеллектуальных информационных систем. Прикладная информатика. 2017. № 3 (69). Том 12. С. 57–79.
11. 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. С. 173-178.

12. 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.
13. 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.
14. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. – М.: Эдиториал УРСС, 2002. – 352 с.
15. Xiao X., Zhang H., Hasegawa O. Density Estimation Method Based on Self-Organizing Incremental Neural Network and Error Estimation // Proceedings of the Neural Information Processing: 20th International Conference, ICONIP 2013. — Daegu, Korea, 2013. — 43–50 p.
16. Pugh J.K., Stanley K.O. Evolving Multimodal Controllers with HyperNEAT. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013). New York, NY: ACM, 2013. – 8 p.
17. Basu A., Blanning R. Metagraphs and their applications. Springer, 2007. 174 p.
18. Voloshin Vitaly I. Introduction to Graph and Hypergraph Theory. Nova Science Publishers, Inc., 2009, 287 p.
19. Johnson J. Hypernetworks in the Science of Complex Systems. London, Imperial College Press, 2013. 349 p.
20. Попков В.К. Математические модели связности. Новосибирск: ИВМиМГ СО РАН, 2006. — 490 с.
21. 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.
22. Ревунков Г.И., Гапанюк Ю.Е., Федоренко Ю.С. Описание нейронной сети с использованием метаграфового подхода. Естественные и технические науки. 2016. Выпуск № 12. С. 278-281.
23. Гапанюк Ю.Е., Ревунков Г.И., Спиридонов С.Б., Белянова М.А., Бутылева Ю.М., Туркевич А.С., Кадиев З.Д. Использование метаграфового подхода в

обучающей программе по формированию пептидной цепи. Естественные и технические науки. 2017. Выпуск № 6. С. 144-147.

24. *Rosenblatt F.* Principles of Neurodynamics. Spartan Books, 1962. 453 p.

25. *Minsky M.L., Papert S.A.* Perceptrons. The MIT Press, 1988. 311 p.

26. *Спиринов А.С.* Молекулярная биология: рибосомы и биосинтез белка: учебник для студ. высш. проф. образования. М.: Издательский центр «Академия», 2011.

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

