УДК 004.822
ВАК 05.13.17
РИНЦ 28.23.29

О.А. Невзорова, В.Н. Невзоров

Модель онтологической системы с рефлексивным ядром[1]

В статье рассматривается структура многоуровневой онтологической системы, используемой для планирования прикладных лингвистических задач в системе «OntoIntegrator». Онтологическая система включает онтологию планирования задач, онтологию моделей и прикладные онтологии. Рассматривается концепция онтологии с рефлексивным ядром, обеспечивающим возможность модификации структуры онтологии в динамическом режиме.

Ключевые слова: *онтологии, рефлексивное программирование, онтологическое моделирование.*

MODEL OF ONTOLOGICAL SYSTEM WITH REFLEXIVE CORE

The structure of multilevel ontological system is described in the article. This model is used for planning linguistic problems in the «OntoIntegrator» system. Ontological system includes the task designing ontology, model ontology and applied ontology. The model of ontology with reflexive core provides the ability to modify the structure of ontology in a dynamic mode.

Keywords: *ontology, reflexive programming, ontology modeling.*

Введение

В последнее десятилетие в мире разработаны различные программные архитектуры для решения задач обработки текстов. Такие успешные проекты, как GATE [1], UIMA [2], Textpresso [3], ориентированы на лингвистическое аннотирование текстов и интегрируют различные программные средства для обработки текстов. Система «OntoIntegrator», разработанная авторами, также является интегрированной инструментальной средой для решения прикладных задач, связанных с автоматической обработкой текстов. При этом в системе последовательно реализуется онтолого-лингвистический подход [4], интегрирующий концептуальные и технологические решения, позволяющие проектировать решения сложных задач обработки текстов в семантическом пространстве знаний, представленных как система онтологических моделей. С одной стороны, система онтологических моделей структурирует семантическое пространство, с другой – управляет решением прикладных лингвистических задач. Разработанные инструментальные средства в системе «OntoIntegrator» ориентированы на обработку текстов на русском языке и содержат в своем составе ряд специализированных баз данных, построенных на основе обработки корпусов текстов на русском языке.

В статье рассматривается новая концептуальная модель иерархической системы онтологических моделей с рефлексивным ядром, позволяющая модифицировать структуру онтологий и интерпретацию решений прикладных задач в динамическом режиме. Нами рассматриваются такие задачи, для которых в процессе их выполнения требуется динамически перестраивать процесс решения, например, за счет изменения интерпретации решения, не меняя характера вычислительного процесса. Представляется, что задачи такого класса возникают в процессе обработки текстов с богатым набором объектов (сущностей) и различными способами их обработки. Выбор соответствующего объекта обработки (смена интерпретации) должен выполняться в рамках заданного вычислительного процесса и перестраиваться специальным способом, который реализуется на основе предлагаемого в статье рефлексивного механизма.

1. Организация системы онтологических моделей

Система «OntoIntegrator» содержит в своем составе следующие функциональные подсистемы:

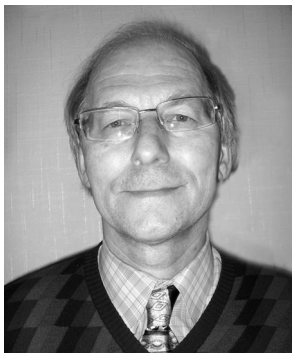
- подсистема «Интегратор»;
- подсистема проектирования онтологий «OntoEditor+»;
- подсистема «Анализатор текста»;
- подсистема ведения внешних лингвистических ресурсов;
- подсистема онтологических моделей.

Подсистема проектирования онтологий «OntoEditor+» [5] обеспечивает основные табличные функции работы с онтологией (добавление, изменение, удаление за-

[1] Исследование выполнено при частичной финансовой поддержке РФФИ, грант № 11-07-00507.

Ольга Авенировна Невзорова,
к.т.н., заместитель директора
Тел.: 8 (843) 292-49-14
Эл. почта: onevzoro@gmail.com
Научно-исследовательский институт «Прикладная семиотика» Академии наук Республики Татарстан
www.ips.antat.ru

Olga A. Nevzorova,
PhD, Deputy Director
Tel.: 8 (843) 292-49-14
E-mail: onevzoro@gmail.com
Research Institute of Applied Semiotics of Tatarstan Academy of Sciences
www.ips.antat.ru



Владимир Николаевич Невзоров,
к.т.н., доцент кафедры Информационных технологий проектирования электронно-вычислительной аппаратуры
Тел.: 8 (843) 231-00-81
Эл. почта: nevzorovvn@gmail.com
Казанский национальный исследовательский технический университет имени А.Н. Туполева
www.kai.ru

Vladimir N. Nevzorov,
PhD, Associate Professor of the IT Designing Department
Tel.: 8 (843) 231-00-81
E-mail: nevzorovvn@gmail.com
Kazan National Research Technical University named after A.N. Tupolev
www.kai.ru

писей; автоматическая коррекция записей; ведение нескольких онтологий, в том числе смешанных, т.е. с общими списками типов отношений, классов, текстовых эквивалентов и др.; импорт онтологий различных форматов; фильтрация онтологии; ведение автоматической статистики по объектам онтологии; поиск цепочек отношений и др.). Функции модуля визуализации поддерживают различные графические режимы системы, в том числе графический режим проектирования онтологий.

Подсистема «Анализатор текста» включает лингвистический инструментарий, предназначенный для решения задач морфологического анализа, онтологической разметки, разрешения многозначности, сегментации и прикладного лингвистического моделирования.

Подсистема ведения внешних лингвистических ресурсов поддерживает ведение основных лингвистических ресурсов, в составе которых выделяются размеченный грамматический словарь и ряд специализированных лингвистических баз данных.

Подсистема «Интегратор» обеспечивает интеграционную основу решения прикладной лингвистической задачи и управление построением решения прикладной задачи.

Процесс решения прикладной лингвистической задачи в системе «OntoIntegrator» реализуется под управлением системы онтологических моделей [6]. Система онтологических моделей включает различные типы онтологий: прикладные онтологии, онтологию моделей и онтологию планирования задач (рис. 1).

С точки зрения структурной организации система онтологических моделей представляет собой трехкомпонентную ассоциативную систему. Компонентами являются онтология планирования задач, онтология моделей и прикладная онтология. В системе допускается интерпретация прикладной онтологии как совокупности онтологий разных проблемных областей (и соответственно, разных семантических интерпретаций): внешних, подключаемых пользователем, и внутренних, встроенных в систему «OntoIntegrator» (с возможностью пополнения, редактирования и вычислительной поддержкой) с целью решения широкого круга прикладных задач. Примерами встроенных онтологий являются онтология общепринятых аббревиатур, онтология маркеров для аннотирования выходного текста по результатам решения прикладной задачи и т.д.

Формально, трехкомпонентная ассоциативная система есть структура вида $S = \langle \Psi, \Sigma, \Omega \rangle$, где Ψ – онтология планирования задач; Σ – онтология моделей; Ω – прикладная онтология.

Онтология планирования задач $\Psi = (\Psi_C, \Psi_R, Z\Psi)$ есть семантическая сеть, в которой Ψ_C – множество

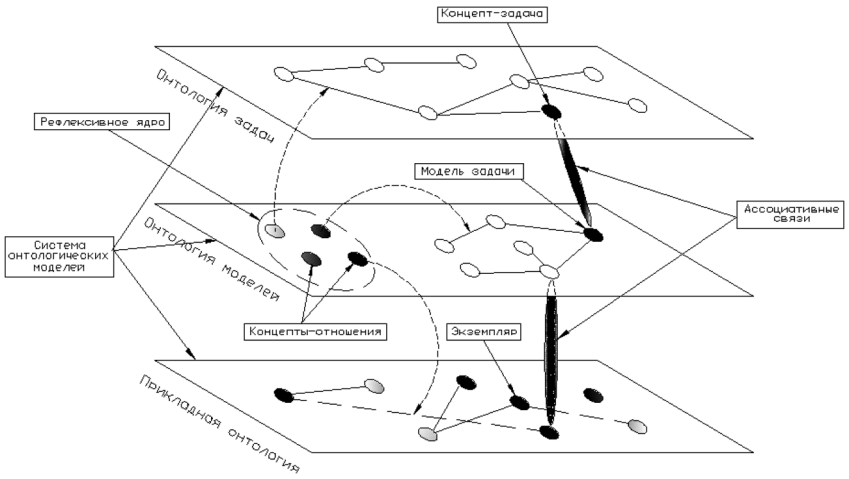


Рис. 1. **Структура системы онтологических моделей**

концептов; Ψ_R – множество отношений, в общем случае *n*-местных; $\Psi_R \subseteq \Psi_C^n$; Z_Ψ – множество функций интерпретации.

Онтология моделей $\Sigma = (\Sigma_C, \Sigma_R, H_{ref}, Z_\Sigma)$ есть семантическая сеть, в которой Σ_C – множество концептов; Σ_R – множество отношений, в общем случае *n*-местных $\Sigma_R \subseteq \Sigma_C^n$; Z_Σ – множество функций интерпретации, H_{ref} – рефлексивное ядро онтологии моделей (подробно будет рассмотрено в разделе 2).

Прикладная онтология $\Omega = (\Omega_C, \Omega_R, Z_\Omega)$ есть семантическая сеть, в которой Ω_C – множество концептов; Ω_R – множество отношений, в общем случае *n*-местных $\Omega_R \subseteq \Omega_C^n$; Z_Ω – множество функций интерпретации.

Между компонентами (семантическими сетями) онтологической системы могут быть установлены ассоциативные связи (ассоциативные отношения) R, $R \subseteq \Psi \times \Sigma$, $R \subseteq \Sigma \times \Omega$. Ассоциативные отношения устанавливаются между подсистемами соответствующих онтологий.

2. Модель рефлексивного ядра

Важной составляющей онтологической системы является компонента онтологии моделей «рефлексивное ядро», которая позволяет моделировать в системе способность к рефлексии. В информатике рефлексия означает динамический процесс модифицирования программной системой собственной структуры и поведения. Соответствующая парадигма программирования называется рефлексивным программированием (функциональное расширение парадигмы объектно ориентированного программирования) и является одним из видов метапрограммирования. Программы, написанные на языках программирования, поддерживающих рефлексию, обладают дополнительными возможностями, реализация которых на языках низкого уровня затруднительна. Перечислим некоторые из них:

– поиск и модификация конструкций исходного кода (блоков, классов, методов, протоколов и т. п.) как объекты первого класса (first class object) во время выполнения;
– изменение имен классов и функций во время выполнения;
– анализ и выполнение строк кода, поступающих извне;
– создание интерпретатора байт-кода нового языка.

Представляется, что рассмотренное понятие рефлексии может быть применено для моделирования процессов модификации структуры онтологической системы. Определим рефлексивное ядро как структуру $H_{ref} = (H_{ref}^C, H_{ref}^R, Z_{ref})$, $H_{ref} \in \Sigma$ (где Σ – онтология моделей, H_{ref}^C – множество ссылок на концепты, H_{ref}^R – множество ссылок на отношения, Z_{ref} – множество функций интерпретации). Рефлексивное ядро является системообразующей частью в построении онтологической системы, оно содержит ссылки на все типы концептов и отношений в онтологической системе, а также множество допустимых функций интерпретации, определяющих выбор используемых типов концептов и отношений, и эта информация представлена как внутренняя модель в онтологии моделей. Другими словами, в онтологической системе представлено знание о структуре системы, организованное как внутреннее знание системы. Рефлексивное ядро может настраиваться (переопределяться) пользователем системы «OntoIntegrator» на основе собственной интерпретации концептов онтологии моделей.

Рассмотрим некоторые типичные модели использования рефлексии в онтологической системе. На рис. 2 показан пример рефлексии, связанный с изменением (переопределением) связей в структуре некоторого концепта-модели. Новая модель сохраняет структурные свойства исходной модели, но меняет семантику связей. Все рассмотренные преобразования выполняются на уровне рефлексивного ядра. На рис. 3 показан пример рефлексии, связанной со сменой представлений о моделях в онтологии моделей. В этой схеме новая структура сохраняет структурные связи исходной модели, но меняет семантику составляющих элементов. Также все преобразования выполняются на уровне рефлексивного ядра.

3. Компоненты онтологической системы

Экземпляр онтологии планирования задач (*p*-концепт) описывает формальную структуру прикладной задачи в виде ориентированного графа, вершинами которого являются концепты-задачи определенных функциональных типов, а отношения между вершинами выбираются из множества отношений онтологии планирования задач. Таким образом, онтология планирования задач формально определяется как структура вида $\Psi = (\Psi_K, \Psi_R, Z_\Psi)$, где Ψ_K – множеств концептов-задач; Ψ_R – множество отношений; Z_Ψ – множество функций интерпретации. Функциональный тип концепта-задачи выбирается из открытого множества допус-

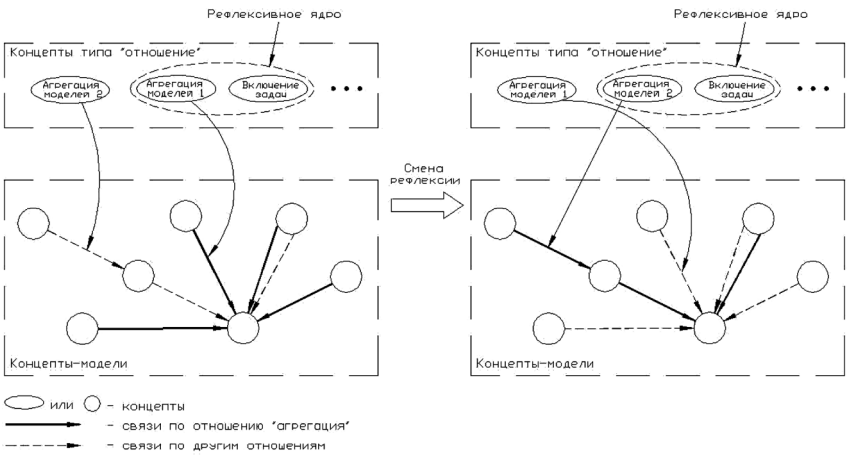


Рис. 2. **Рефлексия как управление структурой модели (задачи)**

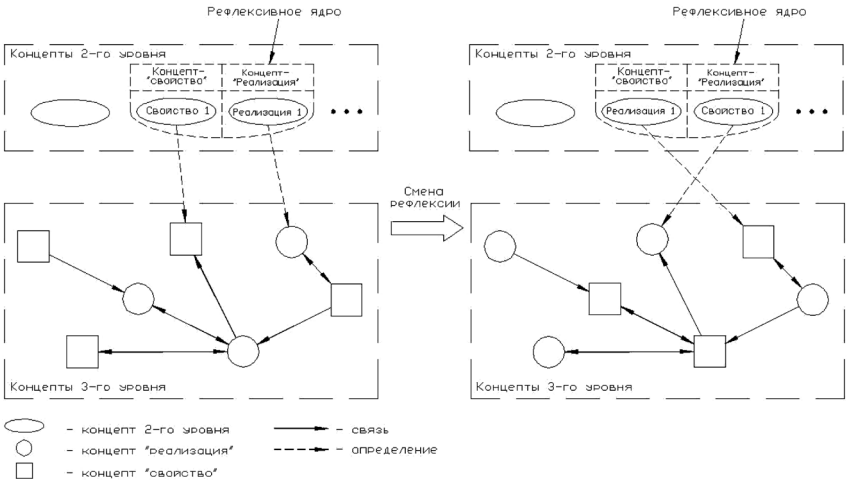


Рис. 3. **Механизм рефлексии как смена представлений о моделях**

тимых функциональных классов: операции (*TOperations*), источники (*TSources*), приемники (*TSinks*), ψ-реализации (Tψ-Realizations).

Класс *ψ-реализации* определяет структуру решения задачи в виде последовательности базовых операций и ψ-реализаций. Экземпляр класса ψ-реализации представляет собой исполняемый модуль для решения определенной прикладной задачи. Класс *операции* содержит расширяемый набор базовых операций, для которых фиксируется конкретная семантика. Классы *источники и приемники* определяют средства преобразования данных, различающиеся функциональностью. Формирование экземпляра класса *ψ-реализации* происходит на основе специального механизма назначения последовательности операций, реализуемого на основе отношения включения из множества Ψ_R отношений онтологии планирования задач.

Планирование логической структуры реализации осуществляется в окне программного модуля «Конструктор задач» с последующей автоматической генерацией кода новой реализации.

Экземпляр онтологии моделей (*s*-модель) формально описывает модель представления исследуемого объекта, который во входном тексте реализуется в форме *t*-модели (текстовый эквивалент *s*-модели). Таким образом, задача распознавания *t*-модели рассматривается как задача классификации. В системе «OntoIntegrator» предложена комплексная технология решения задачи классификации *t*-модели, базирующаяся на следующих механизмах.

Структурные свойства *s*-модели отражают, с одной стороны, грамматические (синтаксические) свойства объектов *t*-модели, а с другой – семантическую интерпретацию этих объектов в конкретных прикладных задачах. Таким образом, структура прикладной s-модели позволяет распознавать реализацию этой модели в тексте и задавать семантическую интерпретацию выделенных объектов.

Задача классификации *t*-модели (отнесение к классу *s*-модели) решается на основе следующих механизмов:

– обнаружение в тексте конкретных экземпляров прикладной онтологии (*e*-моделей) на основе технологии онтологической разметки с последующим построением *t*-модели на основе экземпляров прикладной онтологии;
– установление эквивалентности структуры *t*-модели структуре *s*-модели.

На системном уровне реализация онтологии моделей имеет трехуровневое строение (рис. 4).

На первом уровне при создании новой онтологии моделей выполняется генерация двух абстрактных концептов высшего типа: нулевого концепта (абстрактный концепт,

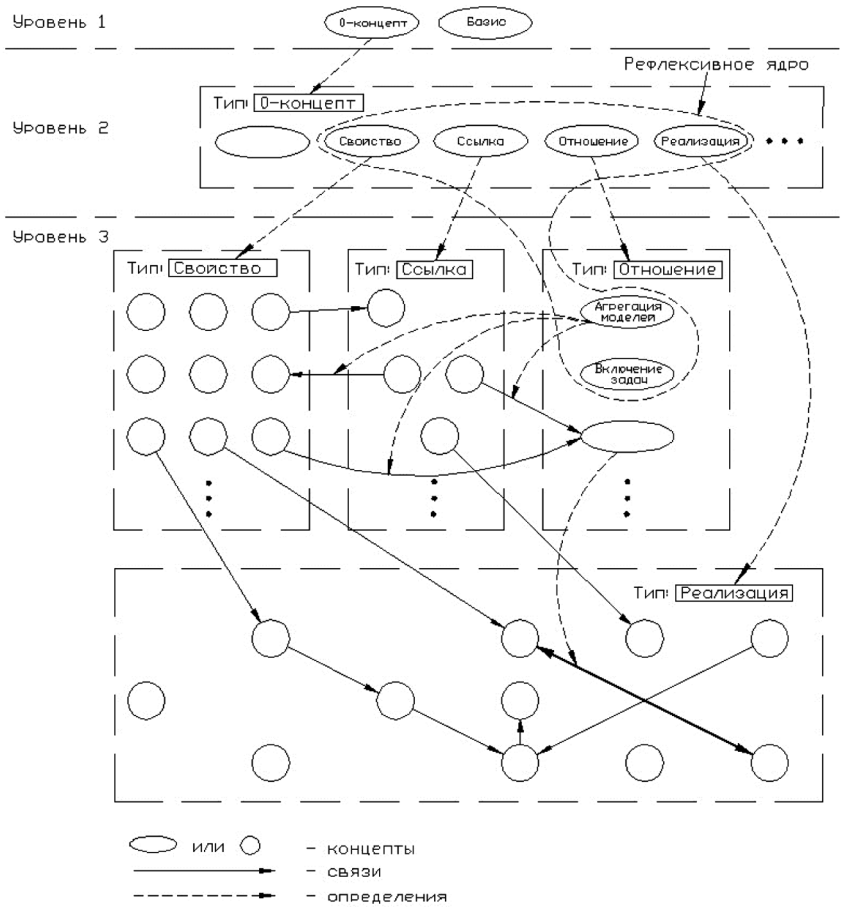


Рис. 4. **Структура онтологии моделей**

универсальный супертип) и абстрактного концепта («*базис*-концепт»), порождаемого от нулевого концепта с абстрактными свойствами и ссылками, обеспечивающими базовую функциональность свойств и ссылок концептов прикладной онтологии.

Все концепты второго и третьего уровней будут иметь специфический атрибут «тип», определяющий базовый класс концепта.

На втором уровне через переопределения нулевых концептов создаются концепты абстрактных базовых типов, интерпретируемые как типы «отношения», «свойства», «ссылки» и «реализации». Выбор пользователем системы четырех концептов 2-го уровня специфицирует рефлексивное ядро системы.

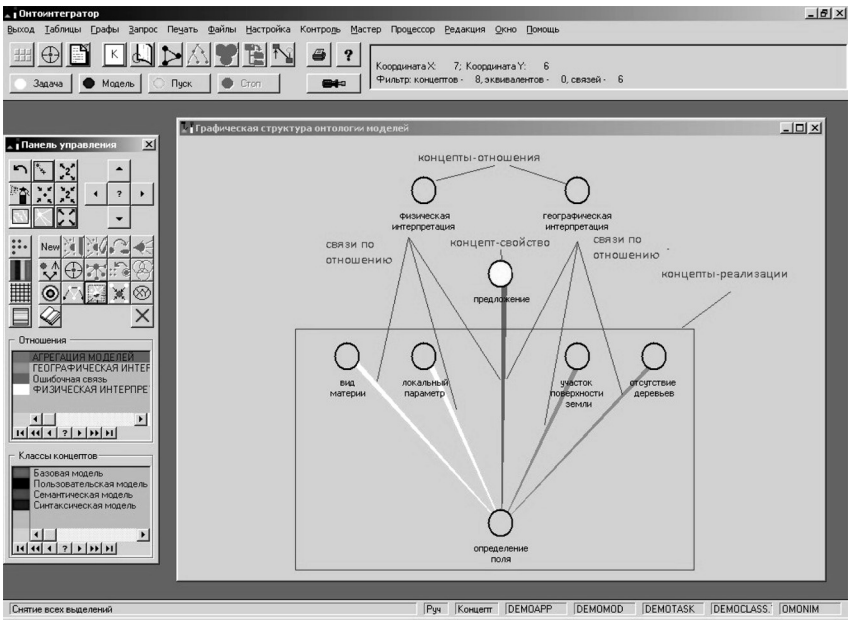


Рис. 5. **Семантические подструктуры в онтологии моделей**

Третий уровень состоит из 4-х групп концептов базовых типов, определяемых концептами рефлексивного ядра. Первая группа концептов типа «отношение» описывает набор отношений, используемых для формирования связей между концептами любой из онтологий (прикладной, моделей и планирования задач). Концепты-«отношения» («включение задач» и «агрегация моделей») используются процессором при построении сложных комплексов концептов в онтологии моделей и онтологии планирования задач и поэтому являются частью рефлексивного ядра.

Концепты-«отношения», не входящие в рефлексивное ядро, могут иметь связи по отношению «агрегация моделей» с концептами типа «свойства» и «ссылки», определяющими свойства этих отношений, и ссылки на их описание из других источников. Вторая группа концептов типа «свойство» специфицирует набор свойств, которые могут быть агрегированы в концепты-«отношения», концепты-«ссылки», концепты-«реализации» и могут обладать вычислительной поддержкой на уровне процессора системы. К третьей группе концептов относятся концепты с типом «ссылка», их значением является адресная ссылка на внешний источник данных. К четвертой группе относятся концепты с типом «реализация», которые представляют собой создаваемые пользователем исполняемые модули. Эти модели обладают свойствами и ссылками агрегированных в них концептов-«свойств», концептов-«ссылок» и концептов-«реализаций».

Смена концептов на уровне рефлексивного ядра приводит к смене концептуальной структуры моделей в онтологии моделей и задач в онтологии планирования задач. Для предотвращения коллизий концепты онтологии моделей не могут быть удалены, если они принадлежат рефлексивному ядру. Связи переопределенного концепта-«отношение» признаются нелегитимными (но не удаляются), что позволяет динамически актуализировать структуры моделей в процессе решения задач.

Рассмотрим пример применения рефлексивных механизмов в задаче анализа экспериментального текста статьи толкового словаря, вводящего определение некоторого многозначного понятия, например, «…под **полем** в физике понимается ***вид материи***, каждая точка которого характеризуется некоторым ***локальным параметром***. В географии же **поле** – это ***участок поверхности земли***, на котором наблюдается ***отсутствие деревьев***». Цель анализа заключается в применении соответствующей интерпретации (физической или географической) для понятия «поле» при обработке различных фрагментов (предложений) текста данной статьи. Решение строится на основе введения новых отношений «физическая интерпретация» и «географическая интерпретация» в онтологию моделей. Понятия текста «поле», «вид материи», «локальный параметр», «участок поверхности земли», «отсутствие деревьев» на основе отношений агрегации образуют соответствующие семантические подграфы (рис. 5).

Для обработки текста может быть выбрана та или иная интерпретация и в тексте будут выделены объекты, относящиеся к заданной интерпретации. Таким образом, рефлексивный механизм управления обработкой позволяет удерживать в фокусе внимания заданные интерпретации, и следующим шагом в данном направлении могло бы стать автоматическое изменение фокуса при смене интерпретации процесса обработки.

Заключение

В статье рассмотрена формальная модель системы онтологий с рефлексивным ядром, реализованная в онтолого-лингвистической системе «OntoIntegrator». Многоуровневая система онтоло-

гий включает различные типы онтологий: прикладные онтологии, онтологию моделей и онтологию планирования задач. Важной составляющей онтологической системы является компонента онтологии моделей «рефлексивное ядро», которая позволяет моделировать в системе способность к рефлексии.

Описанные в статье механизмы рефлексии системы реализованы в системе «OntoIntegrator», разработанные программные решения применены в задаче представления многозначных концептов.

Модель онтологии с рефлексивным ядром обеспечивает динамический процесс модифицирования собственной структуры и поведения программной системой, что существенно расширяет возможности системы для моделирования решений сложных интеллектуальных задач.

Литература

1. *Boncheva, K., Tablan, V., Maynard, D., Cunningham, H.* Evolving GATE to meet new challenges in language engineering // Natural Language Processing. – V. 10, № 3–4. – P. 349–374.

2. *Ferrucci, D., Lally, A*. UIMA: an architecture approach to unstructured information processing in corporate research environment // Natural Language Processing. – V. 10, № 3–4. – P. 327–348.

3. *Muller, H.-M., Kenny, E.E., Sternberg, P.W.* Textpresso: an ontology-based information retrieval and extraction system for biological literature // PLoS Biology. – V. 2, № 11. – P. 1984–1998.

4. *Невзорова О.А.* Онтолингвистические системы: технологии взаимодействия с прикладной онтологией / О.А. Невзорова // Ученые записки Казанского государственного университета. Серия физико-математические науки. – 2007. – Том 149. – Кн. 2. – С. 105–115.

5. *Невзорова О.А.* Инструментальная система визуального проектирования онтологий «OntoEditor+» в лингвистических приложениях / О.А. Невзорова // Вестник КГТУ им. А.Н.Туполева. – 2006. – № 3. – С. 56–60.

6. *Невзорова О.А., Невзоров В.Н.* Многоуровневая онтологическая система для планирования решений прикладных задач // Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS-2011): материалы Междунар. научн.-техн. конф., Минск, 10–12 февраля 2011. – Минск: БГУИР, 2011. – С. 323–330.