Универсальная система синтаксической разметки текстов ОвjectATE

Universal syntax annotation system ОвjectATE

Зобнин А. И. (Alexey.Zobnin@gmail.com)
Московский государственный университет им. М. В. Ломоносова, Москва, Россия, Институт русского языка им. В.В. Виноградова РАН

Сахарова А. В. (nenen@mail.ru)
Институт русского языка им. В.В. Виноградова РАН

Представлена система универсальной разметки текста ObjectATE, основанная на принципах объектно-ориентированного проектирования. Она используется в Отделе лингвистического источниковедения Институте русского языка им. В. В. Виноградова РАН для морфологической и синтаксической разметки древнерусских текстов (переводных памятников и летописей) в полуавтоматическом режиме. Система является гибкой и позволяет пользователю самому как задавать макет разметки в терминах шаблонов и надстроек (классов), так и описывать способы ее визуализации.

1. Предпосылки создания системы

На данный момент отсутствуют многофункциональные средства создания лингвистических текстовых корпусов, позволяющие заниматься лингвистической разметкой корпуса начиная с того уровня (морфологического, поверхностно-синтаксического, семантического и т. п.), который выбирает разметчик и по тем параметрам, которые он задает сам. Однако именно такое средство необходимо для создания лингвистически размеченного корпуса древних письменных памятников. Поскольку лексика и грамматика древних памятников не изучена в полном объеме, а сами тексты не свободны от разного рода ошибок и темных мест, их грамматическая разметка должна быть ручной. Однако было бы хорошо, если бы применяемая для такой разметки информационная система позволяла частично автоматизировать эту разметку.

Создаваемая система обработки текста ObjectATE (Object-oriented ancient text editor) призвана решить эти проблемы. Она разрабатывается и используется в Отделе лингвистического источниковедения Институте русского языка им. В. В. Виноградова РАН с 2006 г. [1–5]. Она пришла на смену предыдущей системе ATE, с помощью которой велась ручная и полуавтоматическая разметка морфологии в древнерусских текстах — переводных памятниках и летописях (переводная антология «Пчела», Киевская летопись по Ипатьевскому списку, Новгородская первая летопись и др.).

Новая система создавалась с целью вести прежде всего ручную синтаксическую разметку этих текстов. Однако рутинную часть работы в ней можно автоматизировать с учетом имеющейся морфологической разметки и формулируемых пользователем правил (морфологических и формально-синтаксических). При этом она призвана быть максимально гибкой и многофункциональной, позволяющей создателю корпуса строить в принципе любые единицы лингвистического анализа по своим собственным (а не только по тем или иным общепринятым) моделям. Было предложено решение на основе объектно-ориентированного подхода, широко применяемого в программировании. На разработку программы оказала большое влияние информационно-аналитическая система «Манускрипт»[1]. Уже в процессе создания ObjectATE авторы познакомились с такими системами обработки текста, как Emdros[2] и GATE[3]. Эти системы имеют свою специфику, и их трудно приспособить к решению поставленной задачи с максимальной общностью и универсальностью. Так, система «Манускрипт» довольно сложна и не дает пользователю

[1] http://manuscripts.ru.

[2] Emdros — the database engine for analyzed or annotated text. http://emdros.org.

[3] GATE — General Architecture for Text Engineering. http://gate.ac.uk.

возможности гибко настраивать макет разметки. Система GATE предусматривает множество различных обработчиков текста и позволяет его *аннотировать*, но, к сожалению, не предусматривает непосредственной работы с синтаксическими конструкциями. Больше всего соответствует поставленной задаче система Emdros, в которой всякий объект разметки представлен как множество элементарных составляющих (называемых монадами) и может содержать атрибуты и ссылки на другие объекты. Такой подход позволяет описывать синтаксис предложения как структуру составляющих. Запросы в Emdros строятся как описания таких структурных включений, но они не рассматриваются как типы данных.

В отличие от всех перечисленных наш подход обладает большим уровнем абстракции, а также механизмом ограничений, которые позволяют контролировать и даже частично автоматизировать разметку. Отметим, что рекомендации TEI[4], в первую очередь предназначенные для описания структуры текста, также не позволяют вести разметку на достаточном уровне абстракции. При этом объектно-ориентированный подход не предполагает особой формы хранения данных, которое может быть основано как на реляционной базе данных, так и на xml-подобном языке. Таким образом, сомнений в необходимости создания собственной разработки не возникало, однако знакомство с идеями, заложенными в этих системах, оказалось очень полезным и обязательно найдет применение в будущих версиях ObjectATE.

2. Функциональные возможности

Система ObjectATE разрабатывается как программное средство для создания, хранения и обработки текстов, проанализированных на любом лингвистическом уровне. Она позволяет заниматься в ручном режиме морфологической разметкой предварительно уже разделенного на словоформы текста, т. е. присваивать словоформам значения морфологических категорий (полей словоформ); при этом пользователь может сам создавать или редактировать списки этих категорий и их значений. В настоящее время в систему внедряются механизмы лемматизации, автоподстановки морфологических параметров, создания словников и указателей (хотя пока что для этих целей в Отделе лингвистического источниковедения применяется предыдущая версия редактора ATE).

Даже при отсутствии морфологической информации (т.е. если текст только разделен на словоформы) система ObjectATE обеспечивает возможность ручной синтаксической разметки текста, т.е. создания в базе данных новых объектов — единиц синтаксического анализа. Такие объекты могут быть организованы как угодно: состоять из одной, двух или большего количества словоформ, равноправных по отношению друг к другу или нет. При этом, разумеется, пользователь может описывать синтаксис и стандартным образом.

Так, чтобы разбирать текст по зависимостям, он должен создать список типов синтаксических связей и начать связывать друг с другом наборы словоформ. После того, как пользователь сформулирует, какой узел у каждой связи является вершинным, а какой — подчиненным, становится возможным построение ориентированного дерева зависимостей. Система также позволяет создавать вспомогательные для синтаксического анализа узлы, функционирующие как аналоги словоформ, например нулевые подлежащие личных глаголов или фантомные эллиптические нули с указанием на опущенную словоформу.

Чтобы маркировать некую синтаксическую группу, пользователь может просто одинаковым образом выделить словоформы или нули, входящие в нее, и создать соответствующий объект. Но система предоставляет возможность также заниматься полноценным синтаксическим анализом по группам, образующим иерархическую структуру. Это значит, что можно создавать синтаксические объекты из других уже существующих, которые только в частном случае представляют собой словоформы. Для этого, создав класс синтаксических объектов (в терминологии грамматики составляющих — фразовую категорию), пользователь затем должен оговорить, чем он может быть представлен (например, сформулировать, что сказуемое предложения может представлять собой одну словоформу, восстановленный ноль, аналитическую конструкцию и т.п.). Именно для того чтобы описать это явление, в системе предусмотрено применение механизма надстроек, позволяющих описывать множество разнородных объектов, удовлетворяющих определенным условиям. Например, надстройка «Глагол-связка» включает в себя как словоформу (глагол *быти* в личной форме), так и синтаксический объект под названием «Аналитическая личная форма» *(еси был, был бы)*. Создав такую надстройку «Глагол-связка», мы должны оговорить, что синтаксическая группа «Глагол-связка» должна образовываться только из объектов, входящих в эту надстройку.

Если морфологическая информация о словоформах для разбираемого текста уже имеется, система может использоваться для упрощения и частичной автоматизации синтаксической разметки. Предположим, можно создать надстройку «Сказуемое», в которую будут входить только все личные глаголы, и надстройку «Подлежащее», куда будут входить все субстантивы в именительном падеже. Вхождение в надстройку «Подлежащее» окажется в данном слу-

[4] TEI — Text Encoding Initiative. http://www.tei-c.org.

чае не достаточным, а только необходимым условием создания связи «Подлежащее–Сказуемое», поскольку, как известно, имя в именительном падеже может играть и другую синтаксическую роль.

Также можно оговорить не только условия вхождения словоформы или синтаксического объекта в надстройку, но и ограничения самого синтаксического объекта (например, согласование подлежащего и сказуемого по лицу: если лицо сказуемого — первое, то его подлежащее — либо ноль, либо местоимение первого лица). При необходимости можно оговаривать порядок словоформ относительно друг друга. В подобной системе также можно заниматься и ручной разметкой текста на более глубоких языковых уровнях, вводя специальные метки (коммуникативный статус, семантическая роль и т.п.) для синтаксических объектов или отрезков предложений. Наконец, для переводных текстов разрабатываются макеты разметки для описания соответствий между оригиналом и переводом.

3. Объектная модель данных

Как уже было сказано, система построена на основе объектно-ориентированного подхода, широко применяемого в программировании. Этот подход тесно связан с понятием онтологии в информатике. Весь размеченный документ представляется как набор объектов. Процесс разметки состоит в создании и модификации объектов.

В начале работы пользователь задает метаданные, то есть данные о структуре будущих объектов. Метаданные состоят из шаблонов и надстроек над ними. Шаблон следует понимать как абстрактный тип данных, определяющий вид объекта. Например, в стандартных текстах, с которыми работает система разметки, предполагаются такие шаблоны, как «Страница», «Строка», «Словоформа». Напротив, конкретные страница, строка или словоформа в тексте — это объекты соответствующих шаблонов. Всякий шаблон имеет уникальное имя.

Каждому шаблону приписан определенный набор полей и ограничений. С помощью полей одни объекты в документе могут быть связаны с другими. Так, строка текста относится к какой-то странице, слова расположены в определенных строках, а всякая словоформа обладает определенной частью речи. Поля шаблона — это набор типов признаков, которые могут быть у объекта этого шаблона. Соответственно, каждому полю шаблона приписано имя, а также указано, какие объекты могут выступать в качестве значения этого поля у объектов данного шаблона. Так, пользователь может определить шаблон «Главные члены предложения» с полями «Подлежащее» и «Сказуемое». Ограничения могут относиться и к типу данных значений полей (ясно, что подлежащее не может быть «Страницей», «Строкой» или «Частью речи»), и на сами значения полей и их подполей (например, если подлежащее это отдельная словоформа, имеющая падеж, то этот падеж должен быть именительным). Ограничения последнего вида можно накладывать на весь шаблон в целом. Такие ограничения записываются в виде логических условий на поля (и их подполя с любым уровнем вложенности). Истинность этих ограничений зависит от потенциального набора значений полей. Предполагается, что для всякого объекта данного шаблона эти ограничения должны превращаться в тождественно истинные выражения.

Шаблоны могут выстраиваться в иерархии наследования. Эта возможность оказывается очень удобной при описании метаданных. Шаблон-наследник приобретает все свойства (поля и ограничения) шаблона-предка и может их расширять. Шаблон-предок может быть объявлен абстрактным. Это значит, что он используется только как общий предок для других шаблонов-наследников, а создавать объекты такого шаблона нельзя. Например, если пользователь хочет наделить все объекты синтаксической разметки полем «Комментарий», он может определить это поле у общего абстрактного шаблона «Синтаксический объект» и вывести из него другие шаблоны.

В системе реализован механизм множественного наследования, позволяющий включать один и тот же шаблон в различные иерархии. Благодаря механизму надстроек в системе можно описывать условное наследование, когда набор базовых типов, вообще говоря, зависит от выполнения условий для конкретного объекта.

Надстройка отдаленно напоминает абстрактный шаблон. Она строится над уже существующими шаблонами или надстройками, которые называются кандидатами на вхождение в эту надстройку. Каждому кандидату надстройки может быть приписано условие на его вхождение в надстройку. Как и ограничение шаблона, это условие представляет собой логическое выражение, зависящее от конкретного объекта, его полей, подполей и т. д. Можно индуктивно определить понятие реализации объектом надстройки или шаблона. Во-первых, всякий объект O реализует свой собственный шаблон и все шаблоны-предки этого шаблона. Далее, пусть K — кандидат надстройки H и объект O реализует K. Тогда считается, что O реализует надстройку H, если для объекта O выполнено условие на вхождение кандидата K в H.

Надстройки появились в модели по двум причинам. Во-первых, этот механизм позволяет детально задать условия на поля шаблонов, а во-вторых, надстройки позволяют описывать простые запросы к данным. Рассмотрим эти возможности подробнее. Ранее полю шаблона строго сопоставлялся его тип — другой шаблон. Считалось, что только объек-

ты этого другого шаблона могут являться значениями полей. Это вызывало определенные трудности, прежде всего с «нулевыми» синтаксическими объектами. Нужно было сделать так, чтобы синтаксические нули наравне со словоформами могли быть полями других синтаксических объектов. Однако в случае, когда такие поля выражены словоформами, должно было выполняться дополнительное условие. В предлагаемой модели типом поля шаблона может быть или шаблон, или надстройка. Соответственно, объект может быть значением такого поля, если он реализует его тип. Такой подход позволяет более гибким образом описать модель разметки. С другой стороны, в программе имеется возможность проверить, реализует ли данный объект указанную надстройку, вывести список надстроек, реализуемых данным объектом, а также вывести все объекты, реализующие данную надстройку. Сами эти объекты могут иметь разные шаблоны; их объединяет лишь то, что при выполнении условий вхождения мы относим их к данной надстройке. Поэтому надстройки удобно рассматривать как описания простых запросов к данным, то есть таких запросов, которые возвращают отдельный список объектов.

Надстройка, как уже было сказано, задает достаточные условия для отнесения объекта к некоторой категории. В системе предусмотрен простой механизм, позволяющий показать, что для данного объекта данная надстройка задает и необходимые условия.

Всякий объект имеет обязательную текстовую компоненту «Содержание». Содержание объекта может задаваться пользователем, либо вычисляться по определенным правилам через содержания полей. Объекты имеют также два поля для сортировки и сравнения: дескрипторы начала и конца объекта. Считается, что все объекты данного фиксированного шаблона можно естественным образом упорядочить по их дескрипторам. Если дескрипторы начала и конца различаются, то объект считается «протяженным». Так, естественный порядок имеется на страницах, строках и словах текста. Удобно считать слово «атомарным» объектом, а дескрипторы начала и конца строки приравнивать к дескрипторам первого и последнего слова в строке. Аналогично, дескрипторы начала и конца страницы приравниваются соответственно к дескрипторам первой и последней строки в этой странице. Правила назначения дескрипторов новым объектам можно задавать при описании метаданных. Дескрипторы позволяют строить запросы и ограничения на порядок слов (например, найти все связи «Субстантив–атрибут», в которых субстантив находится раньше атрибута).

Поля шаблонов могут быть трех видов: обычные поля, коллекции и диапазоны (архитектурно предусмотрен четвёртый вид — коллекция диапазонов, но он пока не реализован, так как на данный момент не востребован). Поле-коллекция отличается от обычного поля тем, что предполагает сразу несколько различных значений. Диапазон — это «связная» коллекция, то есть множество объектов, идущих подряд в смысле упорядочения по дескрипторам. Для диапазона достаточно задать начальный и конечный объект. Типичный пример диапазонов — строки в странице или какие-либо естественные связные большие фрагменты текста (например, блоки, части, прямая речь и т. д.).

Поля шаблонов делятся на обязательные и опциональные. Обязательное поле заполняется при создании объекта (например, при синтаксической разметке). Для опциональных полей предлагается список возможных вариантов заполнения. Данный список формируется на основе ограничений шаблона и уже заполненных полей.

Если все кандидаты надстройки имеют общие поля, то при записи условия на поле типа этой надстройки такие поля можно использовать в выражениях. Кроме того, надстройки могут иметь свои поля. Собственные поля надстройки всегда являются опциональными. Объект приобретает такое поле только в том случае, если он реализует надстройку. С помощью такого механизма можно удобно описывать морфологическую разметку. Именно так была организована морфологическая разметка в базе данных «Новгородская первая летопись». В этой модели, например, словоформа имела лишь поле «Часть речи», а другие морфологические поля появлялись у нее только если она реализовывала какие-либо надстройки. Так, поле «Падеж» появлялось, если словоформа реализовывала надстройку «Имя», и т. д.

Условия и ограничения в метаданных задаются на специальном языке, который интерпретируется программой. Пользователь может создавать их как с помощью конструктора ограничений, так и записывать вручную. Язык содержит основные логические операторы AND, OR, NOT, операторы равенства (=), неравенства (<>), принадлежности (IN) и непринадлежности множеству (NotIN). В выражениях могут участвовать поля и их подполя с любым уровнем вложенности. Имена подполей задаются в квадратных скобках и разделяются точкой. Поле-коллекция всегда рассматривается как множество; кроме того, множество может быть задано явно с помощью фигурных скобок и перечислением входящих в него объектов. По умолчанию сравнение объектов производится по их содержанию. Вот пример ограничения на шаблон «Связь с согласованным атрибутом»:

```
([Атрибут].[Часть речи] IN {'прилагательное',
'причастие'})
OR ((([Атрибут].[Часть речи] = 'местоимение')
  AND ([Атрибут].[Лицо] NotIN {'1-е', '2-е', '3-е'})
  AND ([Атрибут].[Лексема] NotIN {'и'}))
OR ([Атрибут].[Часть речи] = 'числительное').
```

(Здесь так записанное условие на лицо атрибута просто означает, что это лицо отсутствует.)

Перечислим еще некоторые важные операторы этого языка:

1. Оператор проверки реализации IS. Он позволяет проверить, что данное поле объекта реализует указанную надстройку или шаблон. Например, «[Атрибут] IS Словоформа». Также в синтаксис языка ограничений добавлено ключевое слово Me, обозначающее сам проверяемый объект. В условиях на вхождение в надстройку удобно писать выражения вроде «Me IS Субстантив».

2. Условный оператор IF. С его помощью можно корректно обращаться к полям объектов, которые, вообще говоря, не являются общими. Вместе с оператором IS он частично заменяет механизм надстроек, обеспечивая большую гибкость. Пусть, например, поле «Подлежащее» может быть выражено как словоформой, так и нулем. Пусть шаблоны «Словоформа» и «Ноль» безусловно входят в некоторую надстройку. У шаблона «Ноль» нет поля «Падеж»; к падежу можно обратиться только у «Словоформы». Поэтому условие на подлежащее можно записать так:

```
IF ([Подлежащее] IS Словоформа, [Подлежащее].
                [Падеж] = 'именительный').
```

3. Операторы сравнения <= и >= позволяют сравнивать объекты по их дескрипторам сортировки и, в частности, строить запросы на порядок слов.

4. Интерфейс программы

Программная оболочка ObjectATE представляет собой графический пользовательский интерфейс для работы с размеченным текстом по описанной объектной модели. Она позволяет просматривать текст и отдельные объекты разметки с их свойствами, редактировать, создавать и удалять объекты, выполнять запросы. Программа имеет мощные и гибкие средства графической визуализации разметки и синтаксических связей, которые описываются внешним образом в xml-файле. Это тоже своего рода «метаданные», относящиеся к интерфейсу. С их помощью можно визуализировать в тексте результаты запроса и связанные объекты, строить графы иерархических зависимостей (синтаксические деревья, словоуказатели) и т. д. Фрагменты окон работающей программы приведены на рисунках 1, 2 и 3.

Текущая версия системы реализована на платформе Microsoft .NET Framework с использованием реляционных баз данных Microsoft Access и Microsoft SQL Server.

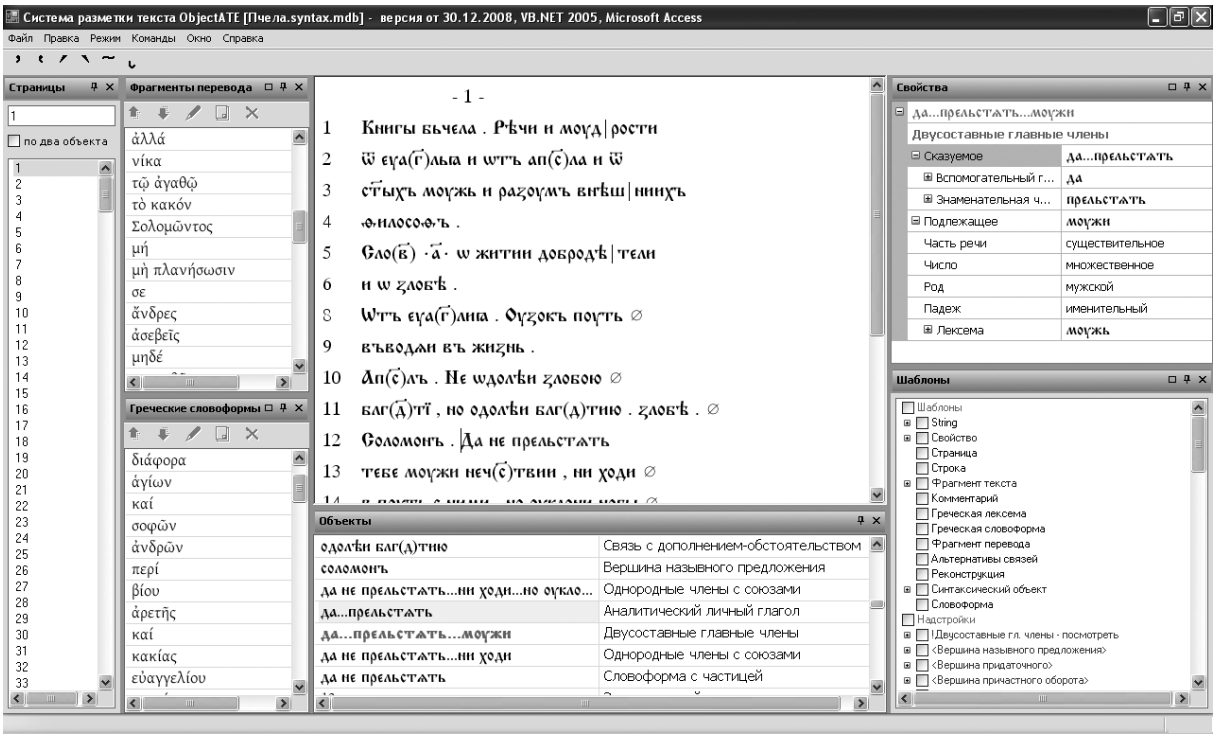

Рис. 1. Панели графического интерфейса пользователя программы ObjectATE

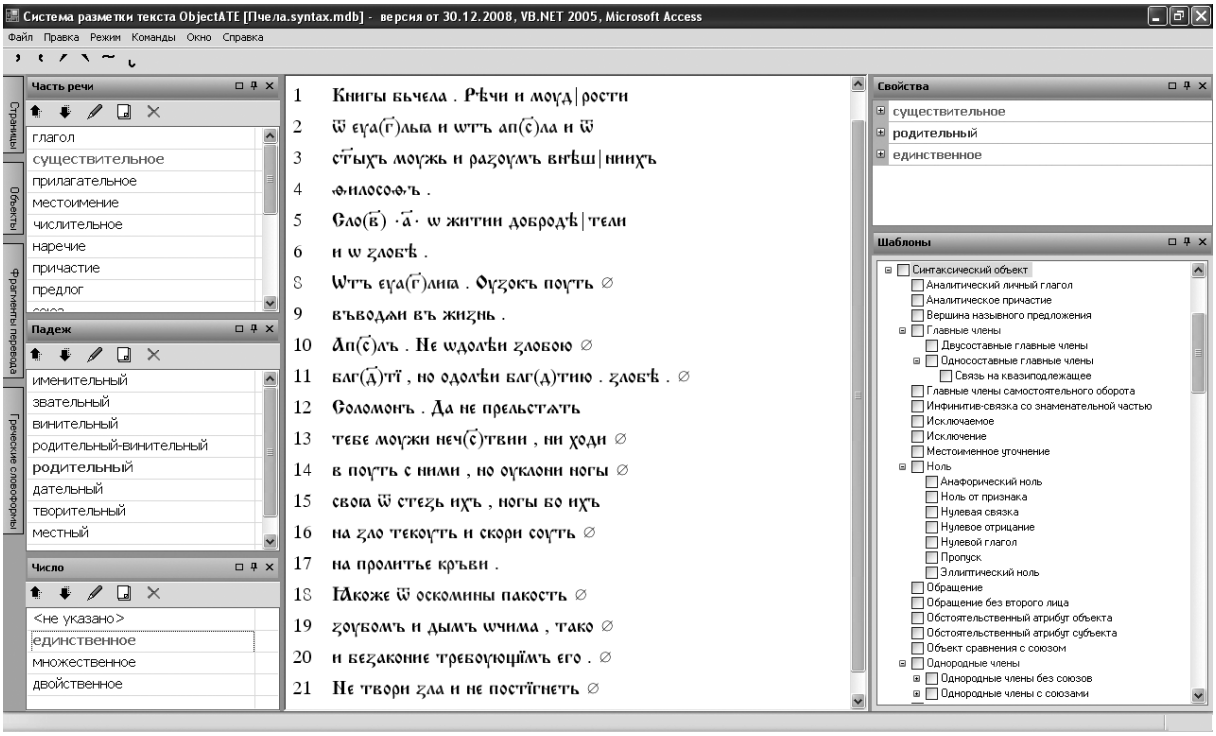

Рис. 2. Выполнение простейших запросов с помощью подсветки
(запрос на существительные в единственном числе и в именительном падеже)

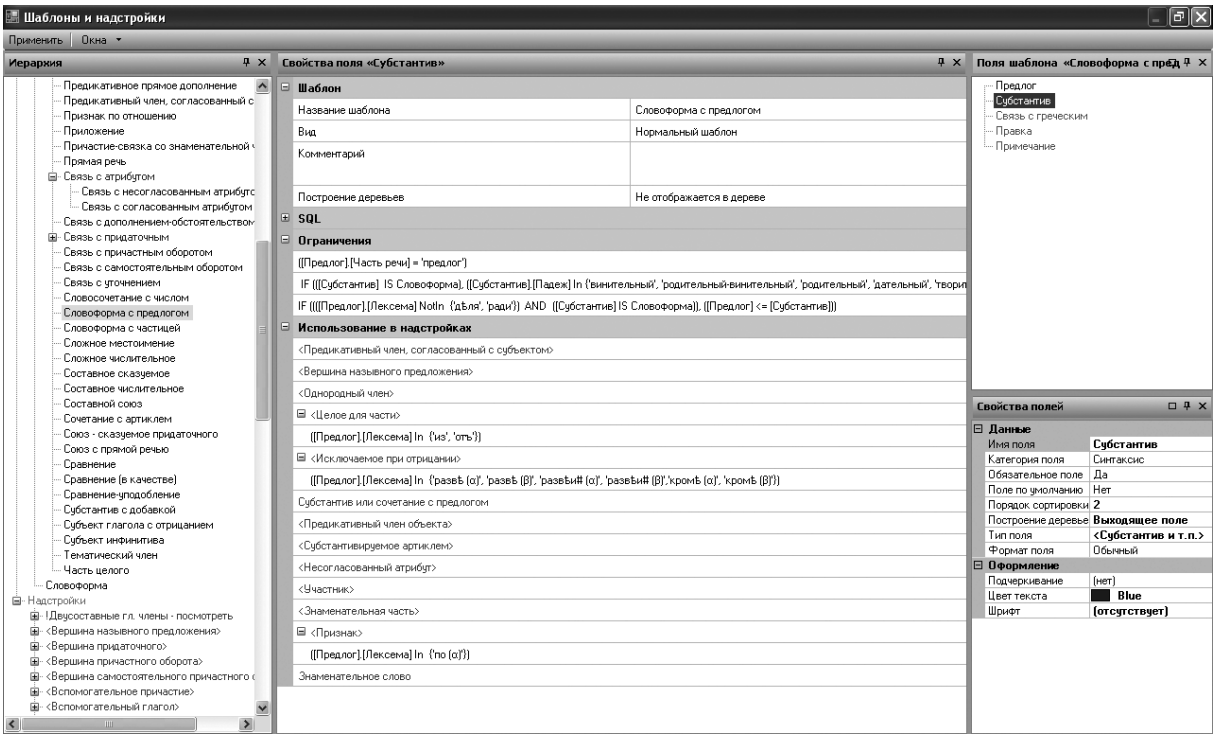

Рис. 3. Диспетчер шаблонов и надстроек.
Показаны свойства шаблона «Словоформа с предлогом»

Литература

1. *Зобнин А. И., Маркелова А. В.* Универсальная система разметки текста ATE-2 // Современные информационные технологии и письменное наследие: от древних рукописей к электронным текстам. Материалы международной научной конференции. Ижевск, 2006. С. 51–55.
2. *Зобнин А. И., Маркелова А. В.* Универсальная система разметки текста ObjectATE // Современные информационные технологии и письменное наследие: от древних текстов к электронным библиотекам. Материалы международной научной конференции. Казань, 2008. С. 114–117.
3. *Сахарова А. В.* Возможности применения универсальной системы синтаксической разметки текста ObjectATE // Современные информационные технологии и письменное наследие: от древних текстов к электронным библиотекам. Материалы международной научной конференции. Казань, 2008. С. 247–249.
4. *Зобнин А. И., Пичхадзе А. А.* Корпус древнерусских переводов XI–XII веков: результаты и перспективы // Научно-техническая информация. Информационные процессы и системы. М., 2005. Сер. 2, № 3, С. 44–47.
5. *Зобнин А. И., Сахарова А. В.* Универсальная система разметки текста ObjectATE // Национальный корпус русского языка: 2006–2008. Результаты и перспективы. М., 2008. С. 283–297.