Универсальная система разметки текста ObjectATE

А. И. Зобнин, А. В. Маркелова
Институт русского языка им. В. В. Виноградова РАН, МГУ им. М. В. Ломоносова, Москва, Россия

Система разметки текста ObjectATE (от Object-oriented ancient text editor) разрабатывается и используется в Отделе лингвистического источниковедения ИРЯ РАН с 2006 года [Зобнин и др. 2006]. Текущая версия системы реализована на платформе Microsoft .NET Framework 2.0 с использованием реляционной базы данных Microsoft Access.

В основе системы лежит *объектно-ориентированный подход*, широко применяемый в программировании. Весь размеченный документ представляется как набор *объектов*. Процесс разметки состоит в создании и модификации объектов.

В начале работы пользователь сам задает *метаданные*, то есть данные о структуре будущих объектов. Это является главной особенностью системы: правила разметки задаются самим пользователем, а не жестко записаны в программе. Метаданные состоят из *шаблонов* и *надстроек* над ними. *Шаблон* (или *класс*) можно 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.

Каждому шаблону приписан определенный набор *полей* и *ограничений*. С помощью полей одни объекты в документе могут быть связаны с другими. Так можно описать, что строка текста относится к какой-то странице, слова расположены в определенных строках, а всякая словоформа имеет часть речи. Поля шаблона – это набор типов признаков, которые могут быть у объекта этого шаблона. На них могут быть наложены естественные ограничения. Эти ограничения относятся и к типу данных значений полей, и к самим значениям полей и их подполей (например, если подлежащее в шаблоне «главные члены» – отдельная словоформа, имеющая падеж, то этот падеж должен быть именительным). Такие ограничения записываются в виде логических условий

на поля (и их подполя с любым уровнем вложенности). Считается, что для всякого объекта данного шаблона эти ограничения должны быть тождественно истинными.

Шаблоны могут выстраиваться в иерархии наследования. Эта возможность оказывается очень удобной при описании метаданных. Шаблон-наследник приобретает все свойства (поля и ограничения) шаблона-предка, добавляя к ним, возможно, свой набор полей и ограничений. Шаблон-предок может быть *абстрактным* (т.е. использоваться в качестве общего предка других шаблонов-наследников). Создавать объекты абстрактного 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 этого шаблона другие шаблоны.

Надстройка напоминает абстрактный шаблон. Она строится над уже существующими шаблонами или надстройками, которые называются *кандидатами на вхождение в эту надстройку*. Каждому кандидату 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* конкретным 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*.

Надстройка, как и шаблон, может быть типом поля. Соответственно, объект может быть значением такого поля, если он реализует такой тип. Это дает определенную гибкость в определении метаданных. Во-вторых, 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. Примером может служить надстройка «подлежащее», в которую входит «словоформа» (при наличии условия на падеж), а также шаблон «ноль» без условий.

Всякий объект имеет обязательную текстовую компоненту «содержание». Содержание объекта может задаваться пользователем, либо вычисляться по определенным правилам через содержания полей. Объекты имеют также специальные *дескрипторы* для сортировки и сравнения. Они, в частности, 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*. Поле *коллекция* отличается от обычного поля тем, что предполагает сразу несколько различных значений (например, однородные члены). 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. Поля шаблонов также 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 (опциональные) поля. Объект приобретает такое поле только в том случае, если он реализует надстройку. С помощью такого механизма можно удобно описывать морфологическую разметку (это применялось в базе данных «Новгородская первая летопись»).

Условия и ограничения в метаданных задаются на специальном языке, который интерпретируется программой. Пользователь может как создавать их с помощью конструктора ограничений, так и записывать вручную. Язык со-

держит основные логические операторы AND, OR, NOT, операторы равенства (=), неравенства (<>) и сравнения, принадлежности (IN) и непринадлежности множеству (NotIN). В выражениях могут участвовать поля и их подполя с любым уровнем вложенности. Поле-коллекция всегда рассматривается как множество; кроме того, множество может быть задано явно с помощью фигурных скобок и перечислением входящих в него объектов. 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.)

Среди основных операторов языка есть также оператор проверки реализации объектом надстройки или шаблона IS и условный оператор IF (для обращения к полям объектов, которые, вообще говоря, не являются общими). 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»; соответственно, поле «падеж» есть только у шаблона «словоформа». Поэтому условие на «подлежащее» можно записать так:

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

Интерфейс программы представлен *панелями объектов*. Эти панели бывают разных видов; их основная задача – отображать специальным образом определенные объекты. На данный момент в программе предусмотрены та-

кие виды панелей объектов, как панель навигации, панель основного текста, панель-список, панель свойств и т. д. Взаимосвязи между ними и их поведение описывается в отдельном xml-файле.

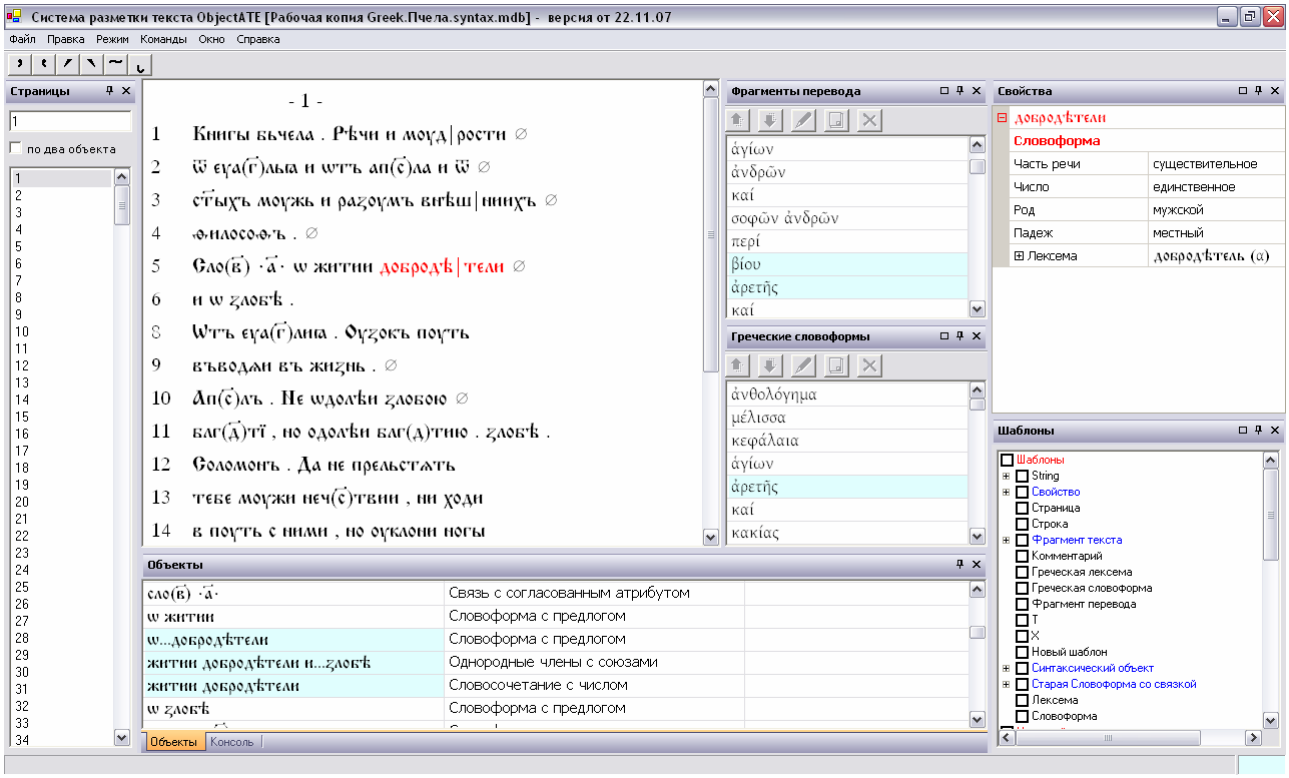

Рис. 1. Панели программы. Выделенные объекты и подсветка связанных объектов

Для работы пользователь может выделять в панелях группы объектов. Каждая группа обозначается своим цветом.

Система будет совершенствоваться дальше. Предполагается внедрить в нее механизмы полуавтоматической разметки, работы с «предложениями», способы визуализации разметки и т. д.

Список литературы

Зобнин и др. 2006 – *Зобнин, А. И.* Универсальная система разметки текста АТЕ-2 / А. И. Зобнин, А. В. Маркелова // Современные информационные технологии и письменное наследие: от древних рукописей к электронным текстам : материалы междунар. научн. конф., Ижевск, 13-17 июля 2006 г. – Ижевск, 2006. – С. 51–55.

ObjectATE, a universal system for text markup

Alexey I. Zobnin, Alexandra V. Markelova

Vinogradov Institute of the Russian Language of the Russian Academy of Sciences, Lomonosov Moscow State University, Moscow, Russia

The object model and new features of ObjectATE, a universal text annotation system, are described. This system allows a user to define his own annotation models by describing classes, add-ins, fields, and relations in the metadata layer.