ОБЪЕКТНАЯ МОДЕЛЬ МНОГОФУНКЦИОНАЛЬНЫХ СЛОВАРЕЙ, ОСНОВАННАЯ НА СИНТЕЗЕ ЛИНГВИСТИЧЕСКИХ ЕДИНИЦ

Ивличева О.О., Епифанов М.Е., Лахути Д.Г.

В настоящее время электронные словари стали распространенным программным продуктом. Однако часто их возможности не намного шире, чем у их полиграфических вариантов, – разница лишь в более удобном и быстром поиске нужного слова. Кроме того, такие словари, как правило, содержат достаточно специализированную информацию, поэтому могут быть полезны только ограниченному кругу пользователей. При этом в словари обычного типа нельзя добавлять информацию другого рода, отличного от того, для обработки которого словарь специализирован. С другой стороны, существуют практически важные вопросы, ответ на которые при обычном устройстве словаря получить по сути дела невозможно: для этого пришлось бы прочесть подряд весь словарь. Эти вопросы обычно связаны не с поиском отдельных слов, а с выделением групп слов по определенному признаку.

В работе предлагается новый подход к программным реализациям многофункциональных электронных словарей (под электронным словарем мы понимаем программу, способную хранить и обрабатывать словарные данные). Мы стремились выработать универсальные форматы хранения информации, за счет которых можно было бы уменьшить строгую специализацию словарей, ограничивающую их применимость. Предлагаемое представление информации многофункционально и универсально в том смысле, что допускает различные дополнения и преобразования внутри себя.

Многофункциональность такого словаря достигается за счет его архитектуры, которая принципиально отличается от структуры известных нам автоматических словарей. Его возможности и потенциальная сфера применения, как нам представляется, значительно шире, поскольку свойства предлагаемой модели позволяют на основании одного и того же формата представления словарной информации решать различные задачи - как лексикографические, так и связанные с классическими проблемами обработки текстов на естественном языке.

Наш подход основан на идее представления языковых единиц, информации о них и отношений между ними в виде объектов, объединенных в многоссылочную структуру, которая позволяет:

- представлять данные в словаре локально (то есть так, что каждая сущность описывается только один раз);
- отражать свойства каждого элемента в отдельности и специфику исследуемого объекта в целом через свойства его частей;
- оперативно анализировать модель и получать из нее информацию как о содержании отдельных словарных статей, так и о составе всего словаря.

Разработка электронных словарей - это традиционно сложная и объемная задача. Как и в случае других сложных систем, сокращение времени и затраченного программистами труда может быть достигнуто за счет перехода от уникальных проектов к использованию универсальных, но при этом отражающих специфику данной предметной области, моделей вычисления. Как нам представляется, вычислительную схема, предлагаемая в рассматриваемой объектной модели для разработки и ведения словарей, помогает проще организовывать и решать следующие прикладные задачи:

- разрабатывать новые словари в ситуации, когда состав и структура словарных статей изначально не до конца прояснены и могут изменяться в процессе работы;
- упростить задачу пополнения имеющегося словаря;
- модифицировать словарь, изменяя не только состав, но и структуру данных в нем;

- автоматизировать пополнение и модификацию словаря как массовую задачу, возникающую при изменении или добавлении (например, из словаря другого формата) большого количества данных;
- накапливая и расширяя информацию о лексике какого-либо языка, автоматизировать процесс выделения актуального подмножества входов словаря и получения новых, специализированных словарей в соответствующем формате для решения прикладной задачи на основе универсального, наиболее общего, словаря.

Следует оговориться, что в существующем виде такая модель программно поддерживается как библиотека объектов и использующих их функций, ориентированная на совместное использование лингвистом и квалифицированным программистом. Пока это лишь реализация комплекса мер для организации решения задач подобного рода, способ облегчить работу программиста по имплементации ядра словарных систем.

В основе предлагаемой объектной модели лежат следующие соображения и наблюдения:

- лингвистические единицы с точки зрения их структуры (в смысле представления данных) организованы в многоуровневую иерархию;
- с тем или иным уровнем детализации лингвистическая единица может быть получена как результат применения какой-либо достаточно простой операции к составляющим ее единицам более простого уровня.. Например, в русском языке словоформа может быть представлена как конкатенация составляющих ее морфов;
- с лингвистическими единицами обычно ассоциируются какие-либо свойства, при этом одни из этих свойств являются индивидуальными, относящимися только к этой сущности, а другие подчиняются «наследованию снизу-вверх», то есть проявляются у более сложных единиц, включающих в себя данную сущность. Мы можем считать, например, что словоформы прилагательных, которые получены как конкатенация основы и окончания, наследуют от окончаний их число, род и падеж. При этом такое наследование действует только до определенного уровня;
- со свойствами лингвистических единиц может быть связано определенное поведение, влияющее на поведение самих единиц. Например, порождение сложных единиц из простых регулируется правилами (такими, как управление и согласование), накладывающими ограничения на допустимый набор свойств единиц и их значений, поэтому не каждая единица, которая может быть порождена, будет правильной.

Исходя из этих соображений, в общих чертах определим основные элементы нашей модели и опишем их поведение.

Простым текстовым объектом (далее – текст-объект, текст-элемент) назовем собственно «простой текст», понимаемый как конечное слово (цепочку символов в некотором языке), в совокупности с некоторым конечным множеством его свойств.

Свойство (атрибут) текст-объекта – это тройка p=<d, n, v>, где d(p) – *исполнитель, обработчик* свойства, который пока будем просто понимать как его тип, n(p) – имя свойства,а v(p) – его значение. У текст-объекта не может быть двух свойств с одним и тем же именем.

Мы будем различать так называемые *аддитивные* свойства текст-элемента и его *внутренние*, *индивидуальные* свойства. Обозначим ***PNames*** – множество всех имен свойств, используемых в объектах модели, ***ANames*** – множество всех аддитивных, а ***INames*** – всех внутренних имен свойств. ***PNames***=***ANames***$\cup$***INames*** и ***ANames***$\cap$***INames***=$\varnothing$, т.е. аддитивные и внутренние имена образуют разбиение множества всех имен свойств.

Таким образом, текст-элемент представляется как тройка e=<t, AData, IData>, где t – строка – хранимый в объекте текст, а AData и IData – соответственно конечные множества аддитивных и внутренних свойств, ассоциированных с этим текстом. Пусть ANames(e) $\subseteq$ ***ANames*** – множество имен свойств из AData(e), INames(e) множество имен внутренних свойств из IData(e). Тогда должно выполняться ANames(e)$\subseteq$***ANames***, INames(e)$\subseteq$***Inames***, соответственно ANames(e)$\cap$INames(e)=$\varnothing$, ANames(e)$\leftrightarrow$AData(e), INames(e)$\leftrightarrow$IData(e), (здесь знаком ‘$\leftrightarrow$’ обозначаем эквивалентность множеств, т.е. возможность установить между ними биекцию).

Мы также будем рассматривать конечные множества текст-элементов с обычными теоретико-множественными операциями над ними. На практике в рассматриваемых ниже примерах синтеза используется только объединение.

Теперь определим операцию, используемую нами для получения новых текст-объектов в задачах синтеза. Ее аргументами и результатом будут множества текст-элементов. Как мы увидим, несмотря на некоторую громоздкость, принципиально она будет очень похожа на операцию «join» (соединение) в реляционной алгебре. Поэтому мы ее так и назовем.

Пусть даны два текст-объекта e_1 и e_2. Начнем с неформального (мы не в состоянии сделать это более строго в рамках данной статьи) описания *соединения* множеств их аддитивных свойств $AData(e_1)$ и $AData(e_2)$. В рассматриваемой модели это соединение полностью определяется используемыми в $AData(e_1) \cup AData(e_2)$ обработчиками свойств, множество которых обозначим $D_A(e_1,e_2)$.

Для каждого типа свойств d определяется оператор add_d, который для произвольной пары объектов e_1 и e_2 либо возвращает специальное значение False, либо пару $<AAdd_d(e_1,e_2), IAdd_d(e_1,e_2)>$ построенных им множеств аддитивных $AAdd_d(e_1,e_2)$ и внутренних $IAdd_d(e_1,e_2)$ свойств. В объектной библиотеке функциональность каждого типа свойств реализована соответствующим объектом – обработчиком. Систему можно пополнять новыми типами свойств, добавляя новые обработчики свойств. Важно лишь, чтобы интерфейсы обработчиков удовлетворяли определенным соглашениям и, в частности, поддерживали метод, реализующий оператор add_d .

Например, используемый в примерах ниже обработчик «согласование по значениям» S (от англ. the Same as) выдаст False, если из исходных множеств аддитивных свойств можно выбрать пару одноименных типа S, но имеющих разные значения. В противном случае он выдаст объединение множеств аддитивных свойств типа S в объектах-операндах:

$add_S(e_1,e_2)$=False, если $\exists <S, n_1, v_1> \in AData(e_1)$ и $\exists <S, n_2, v_2> \in AData(e_2)$,
такие, что $n_1=n_2$, но $v_1 \neq v_2$

в противном случае

$AAdd_S(e_1,e_2)=\{<S, n, v> \mid <S, n, v> \in AData(e_1) \cup AData(e_2)\}$

$IAdd_S(e_1,e_2)= \varnothing$

Назовем текст-объекты *соединимыми*, если для всех $d \in D_A(e_1,e_2)$ оператор add_d вернет пару множеств свойств, а не значение False. Для соединимых объектов естественным образом определены множества

$AAdd(e_1,e_2)= \cup (AAdd_d(e_1,e_2) \mid d \in D_A(e_1,e_2))$

$IAdd(e_1,e_2)= \cup (IAdd_d(e_1,e_2) \mid d \in D_A(e_1,e_2))$

То есть пара этих множеств и является результатом *соединения* множеств аддитивных свойств $AData(e_1)$ и $AData(e_2)$ для соединимых текст-объектов e_1 и e_2.

В различных языках синтез одних языковых единиц из других, осмысленных с точки зрения рассматриваемого языка, удобно представлять, как результат подходящих для каждого случая *операций соединения текстов*. Это могут быть, например, подстановки исходных строк в какой-либо образец или конкатенация с перестановкой или заменой каких-либо элементов. Для русского языка такое соединение чаще всего означает просто конкатенацию цепочек символов.

Соединением множеств E_1 и E_2 текст-объектов по некоторой операции соединения текстов θ называется

$join_\theta(E_1, E_2)=\{<t(e_1)\theta t(e_2), AAdd(e_1,e_2), IAdd(e_1,e_2)> \mid e_1 \in E_1$ и $e_2 \in E_2$ – соединимы$\}$,
где $t(e_1)$ и $t(e_2)$ – тексты объектов e_1 и e_2.

При определенных ограничениях на используемые в модели обработчики соединение множеств аддитивных свойств текст-объектов может быть ассоциативным. Можно рассматривать также только ассоциативные операции соединения текстов. Тогда ассоциативным окажется и соединение множеств текст-объектов, что позволяет обобщить его на случай произвольного количества аргументов $join_\theta(E_1, E_2, \ldots, E_k)$.

Так называемые *R-объекты* (*объекты представления*) и связи между ними представляют в словаре множества текст-объектов. Множество Ext(e) текст-объектов, представленное R-объектом e назовем его объемом или экстенсионалом. По существу, R-объекты являются входами словаря.

В настоящее время используется четыре класса R-объектов, однако, как и в случае с обработчиками свойств, модель может быть пополнена их новыми классами. При этом важно лишь поддерживать дисциплину интерфейсов.

Для описания структуры R-объектов используем следующий простой синтаксис. Сразу после сокращения (аббревиатуры) для имени класса в квадратных скобках перечислим имена его атрибутов (слотов): $clname[attname_1,\ldots, attname_k]$. Этот же синтаксис будем использовать и для записи экземпляров R-объектов, указывая значения атрибутов: $clname[attname_1=val_1, \ldots, attname_k=val_k]$. Кроме того используем обозначение attname(e), чтобы указать значение атрибута attname объекта e.

Для каждого класса R-объектов покажем их структуру и определим их экстенсионал.

Класс SimpleText отображает собственно текст-объекты: ST[text, adata, idata]. Пусть e – некоторый объект этого класса. Тогда его экстенсионал определяется как множество Ext(e)={e′} с единственным элементом e′ - копией e.

Объекты класса Property предназначены для хранения только свойств: P[adata, idata]. Экстенсионал объекта e класса Property также является множеством из одного элемента Ext(e)={ ST[text="", adata=adata(e), idata=idata(e)] }.

Видно, что введение этого класса служит лишь для оптимизации модели, вместо него вполне можно было бы обойтись объектами класса SimpleText с пустой строкой.

При помощи объектов класса Union строятся объединения экстенсионалов других объектов, возможно с некоторым изменением их текстов и множеств свойств. Их структура:

U[text, adata, idata, children, textoperation]. Атрибуты text, adata, idata имеют здесь тот же смысл, что и для текст-объектов. Атрибут children хранит конечное множество R-объектов, а textoperation - некоторую операцию соединения текстов.

Объем объекта e класса Union определяется следующим образом. Пусть

$E = \cup (Ext(r) \mid r \in children(e))$, $\theta = textoperation(e)$

Пусть также e′ - вспомогательный объект класса ST со значениями атрибутов, равными значениям одноименных атрибутов объекта e.

Тогда $Ext(e) = join_\theta(\{e'\}, E)$. В случае text(e)="", конкатенации как операции соединения текстов, Ext(e) совпадет с объединением экстенсионалов объектов из children(e).

Наконец, для представления соединения экстенсионалов объектов вводится класс Join. Его объекты имеют ту же структуру, что и объекты класса Union: J[text, adata, idata, children, textoperation].

Для определения экстенсионала объекта e класса Join как и в предыдущем случае введем аналогичный вспомогательный объект e′ класса ST. Пусть θ=textoperation(e), и $children(e)=\{ e_1, e_2, \ldots, e_k\}$. Тогда $Ext(e) = join_\theta(\{e'\}, Ext(e_1), Ext(e_2), \ldots, Ext(e_k))$.

Из определения классов Union и Join следует, что в них можно было бы обойтись без слотов text, adata и idata, которые, по существу представляют соединение с экстенсионалом некоторого возможного текст-объекта. Однако такая оптимизация несколько сокращает количество объектов представления, используемых в конкретной модели.

Установим изначально, что входом словаря может быть любая языковая единица, как элементарная (например, морф), так и составного характера (словоформа, словосочетание). Это позволит в дальнейшем, не меняя модели, расширять словарь в зависимости от задачи, для которой он используется.

В настоящее время проводится разработка средств, с помощью которых введенный вход составного характера можно было бы превратить в комбинацию уже существующих или вновь созданных минимальных входов, сохранив при этом его признаки как единой сущности. Иными словами, физически в словаре будут храниться не сами составные входы, а своеобразное представление, использующее R-объекты, для формул, по которым их можно получить.

Возможность определять некоторые сущности как комбинации более простых входов – это одна из ключевых особенностей нашего словаря. Благодаря этой возможности модель отражает естественный способ многоуровневой организации языковых единиц, позволяет соблюдать принцип локальности данных.

Структурная иерархия лингвистических единиц поддерживается в модели за счет ссылок R-объектов классов Union и Join на другие R-объекты через значения атрибута children. Листьями в дереве этой иерархии являются минимальные значимые единицы текста – морфы, представленные объектами класса SimpleText, и общие свойства языковых единиц, хранимые в объектах класса Property. Т.о. при соединении (объект e класса Join с операцией конкатенации текстов), например, основы (SimpleText) со списком (Union) окончаний (объекты SimpleText) основа «сцепляется» с каждым окончанием, в результате чего получается множество словоформ (объекты SimpleText) как объем Ext(e) объекта e.

В предлагаемой системе имеется возможность выделять актуальное подмножество Ext(e) R-объекта e – входа в словарь как результат выполнения некоторого запроса к e. Выражения запросов содержат ограничения на значения свойств текст-объектов из Ext(e). До сих пор нам вполне хватало запросов, аналогичных селекции из реляционной алгебры, условие выбора в которой – конъюнкция «требований» вида *имя_свойства=значение*. Результат выполнения запроса сохраняется в объекте Query вида Q[qlist, …], где значением атрибута qlist является множество выбранных объектов класса SimpleText, а остальные атрибуты мы в данной работе не рассматриваем.

Требования к функциональности электронного словаря показывают необходимость нового класса объектов, так называемых коллекций, методы которых осуществляют операции над словарем в целом. Основной тип таких

операций – модификация целых групп словарных входов и построение новых словарей, содержание которых может быть динамически специфицировано пользователем.

Не имея возможности подробнее описать коллекции R-объектов в рамках данной статьи, укажем лишь, что в разрабатываемой нами модели коллекции соответствуют уровням иерархии лингвистических единиц, как это показано на рис.1. Но модель открыта и для определения новых коллекций. Связь коллекции с объектом поддерживается не только вхождением R-объекта в коллекцию, но и поведением обработчика внутреннего классифицирующего объект свойства.

иногда заменять поддерево объектом класса Query, понимая его как результат запроса к корню этого поддерева с «пустым» списком ограничений. В этом случае результат запроса есть экстенсионал корня.

Рассмотрим теперь представление всей парадигмы лексемы 'вершина' (рис. 3).

В этом примере объекты класса Union объединяют в списки окончания, отдельно для единственного и для множественного числа. Это удобно для многократного использования, потому что есть существительные, которые присоединяют те же наборы окончаний, но только в формах одного из чисел ('синева', 'ножницы'). Далее объекты Join комбинируют эти списки с грамматическими признаками единственного и множественного числа, и еще один объект Union объединяет две получившихся неполных таблицы флексий в одну стандартную таблицу флексий, характерную для многих существительных I склонения.

Саму лексему будет представлять объект класса Join, являющийся корнем всего поддерева и комбинирующий основу 'вершин' с таблицей флексий. Будем считать, что именно этот объект обладает признаком женского рода, а признаки неодушевленности и падежно-числовые значения он наследует от основы и таблицы флексий соответственно.

Этот пример демонстрирует способность соединения отвергать некоторые единицы, получающиеся в результате синтеза, как неправильные. Мы объединили в одной таблице флексий два окончания винительного падежа, одно из которых присоединяют одушевленные существительные, а другое - неодушевленные. Объект, представляющий признак одушевленности, "знает", что каждая единица может обладать только одним значением такого признака, поэтому столкновение двух таких объектов в результате соединения обеспечит порождение только правильных словоформ.

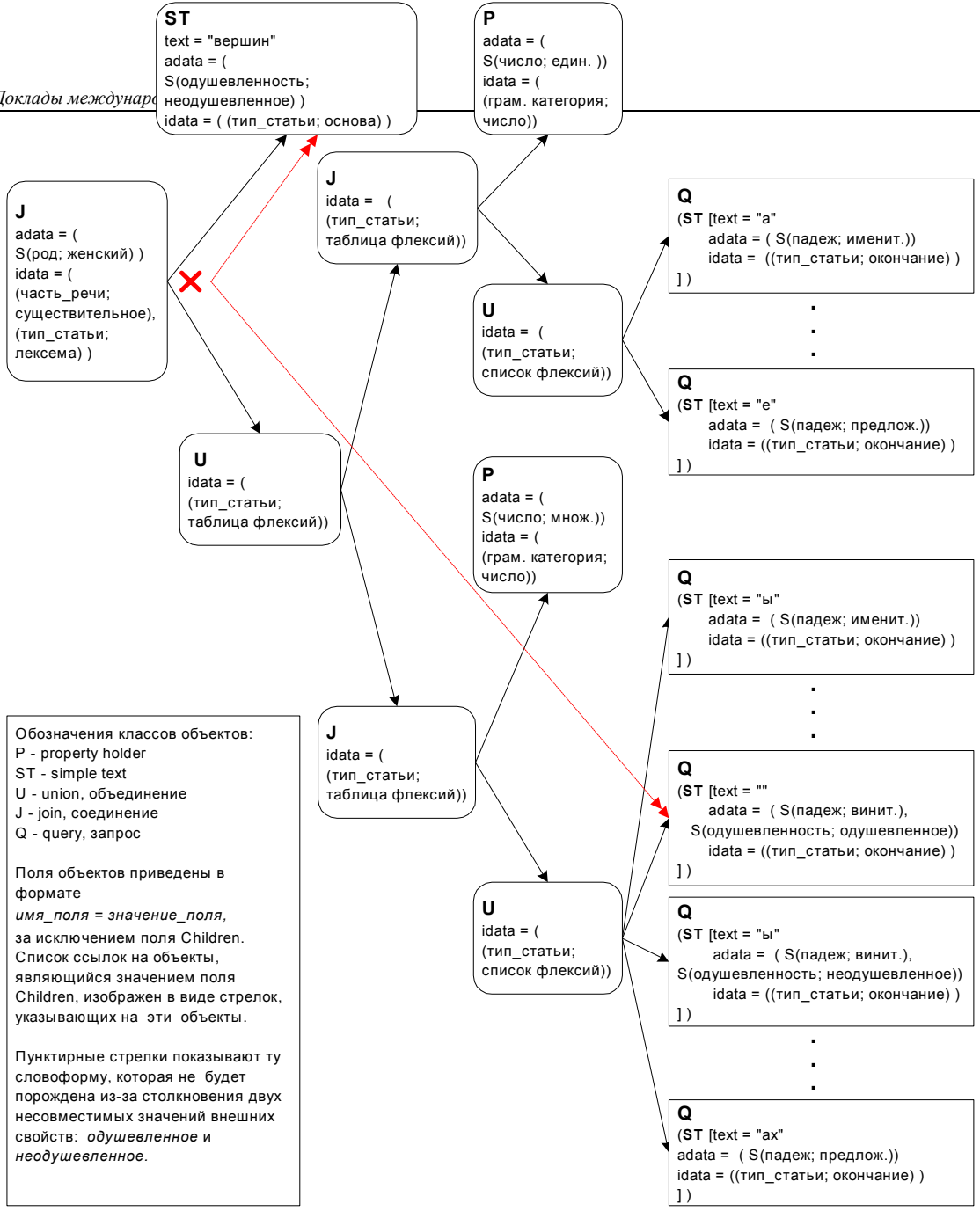

Рис. 3. Представление лексемы 'вершина'

Вообще, появление при вычислении составной единицы разных значений одного и того же свойства является аварийной ситуацией. Возможно два способа разрешить эту ситуацию. Первый – запретить порождение данной словоформы, как в рассматриваемом примере. Второй механизм – уничтожить по правилу какое-либо из значений свойства.

Второй пример относится к модификации уже созданной в словаре структуры данных. Пополнение словаря может осуществляться не только путем добавления новых входов, т. е. путем увеличения объема, но и за счет преобразования и усложнения уже построенной объектной модели. В полученный словарь можно последовательно добавлять дополнительную информацию. Начав с отражения словоизменения, мы сможем позже преобразовать уже накопленный материал так, чтобы он отражал и состав слов, и гнездовую организацию лексики.

В представлении лексемы 'вершина' мы использовали основу, которая не является минимальной единицей и может быть разложена на морфы - корень и суффикс. При этом комбинирующий их объект класса Join будет хранить признаки основы как единой сущности. Для добавления этой информации в словарь мы должны будем заменить в структуре данных один объект SimpleText на поддерево из трех объектов (см. рис. 4). При этом множество лингвистических единиц, которые представляет объект-лексема, останется тем же.

Модификация структуры данных может идти и в другом аспекте, например, если мы хотим добавить к объекту новую информацию о нем, не заменяя его новой версией и не меняя значений его полей для того, чтобы старый объект мог продолжать участвовать в других деревьях объектов. Для иллюстрации приведем добавление толкования к уже построенному суффиксу *-ин-*.

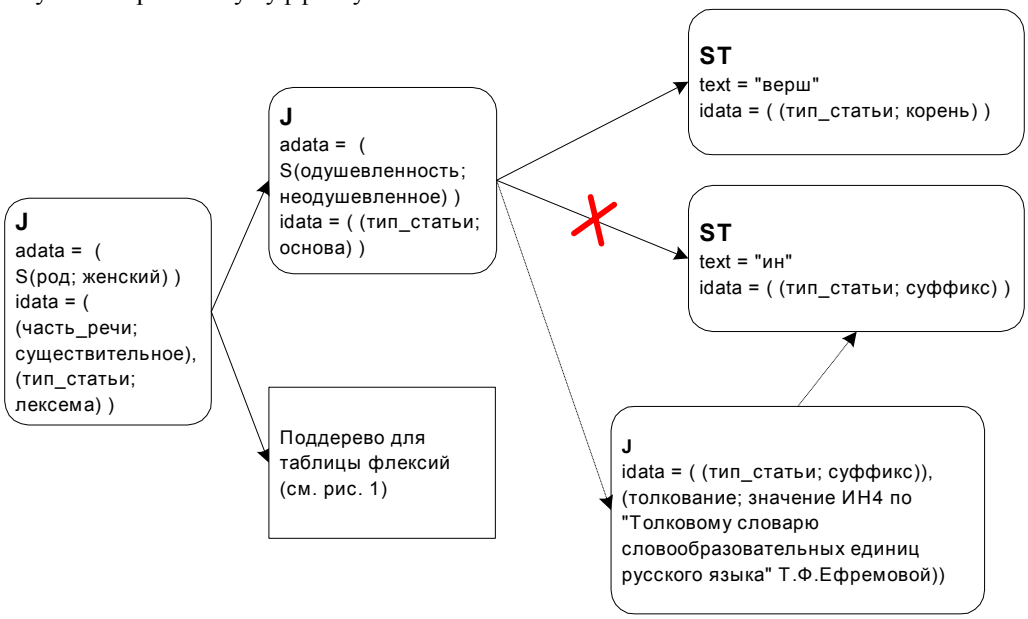

Рис. 4. Разделение основы на морфы и добавление
толкования для суффикса

В "Толковом словаре словообразовательных единиц русского языка" Т.Ф. Ефремовой (Русский язык, Москва, 1996) выделено 14 значений «ин» как суффикса существительных. В лексеме 'вершина' реализуется значение $ИН_4$. Создадим промежуточный объект класса Join, который будет хранить это значение и ссылаться на объект, представляющий суффикс *-ин-*. Тогда в тех лексемах, где *-ин-* имеет то же значение, можно будет заменять ссылку на него ссылкой на новое поддерево из двух объектов (см. рис. 4). Другие лексемы будут по-прежнему ссылаться непосредственно на прежний объект. Тот же механизм может быть применен для сохранения сведений о распределении синонимических единиц.

Еще одна проблема, которую желательно отразить в информационной модели – представление сложных слов. Они должны быть представлены отдельным входом, потому что это самостоятельные сущности с набором свойств. В то же время согласно нашей идее это должны быть составные единицы.

Рассмотрим модели словоизменения сложных слов и их соотношение со словоизменением составляющих слов. Всего можно выделить 4 модели:

- сложное слово пишется слитно, изменяется только последняя часть, а первой частью является основа, быть может, с соединительной гласной («нефтепровод», «железнодорожный»);
- сложное слово пишется через дефис, изменяется только последняя часть, а первой частью является либо основа, быть может, с соединительной гласной («бело-голубой»), либо целое слово в какой-либо форме («черным-черно», «альфа-частица», «перекати-поле»);
- сложное слово пишется через дефис, изменяются обе его части («дед-мороз», «язык-посредник»);
- слово пишется слитно, изменяются обе его части (русские числительные, например, "пятьдесят").

Существование этих моделей позволяет нам говорить о введении объектов – шаблонов образования сложных слов. Как можно заметить, многие словосочетания, которые называют одно понятие и являются, таким образом, одним дескриптором при описании документа (например, «железная дорога», «орден Победы»), своим совместным словоизменением напоминают эти же шаблоны.

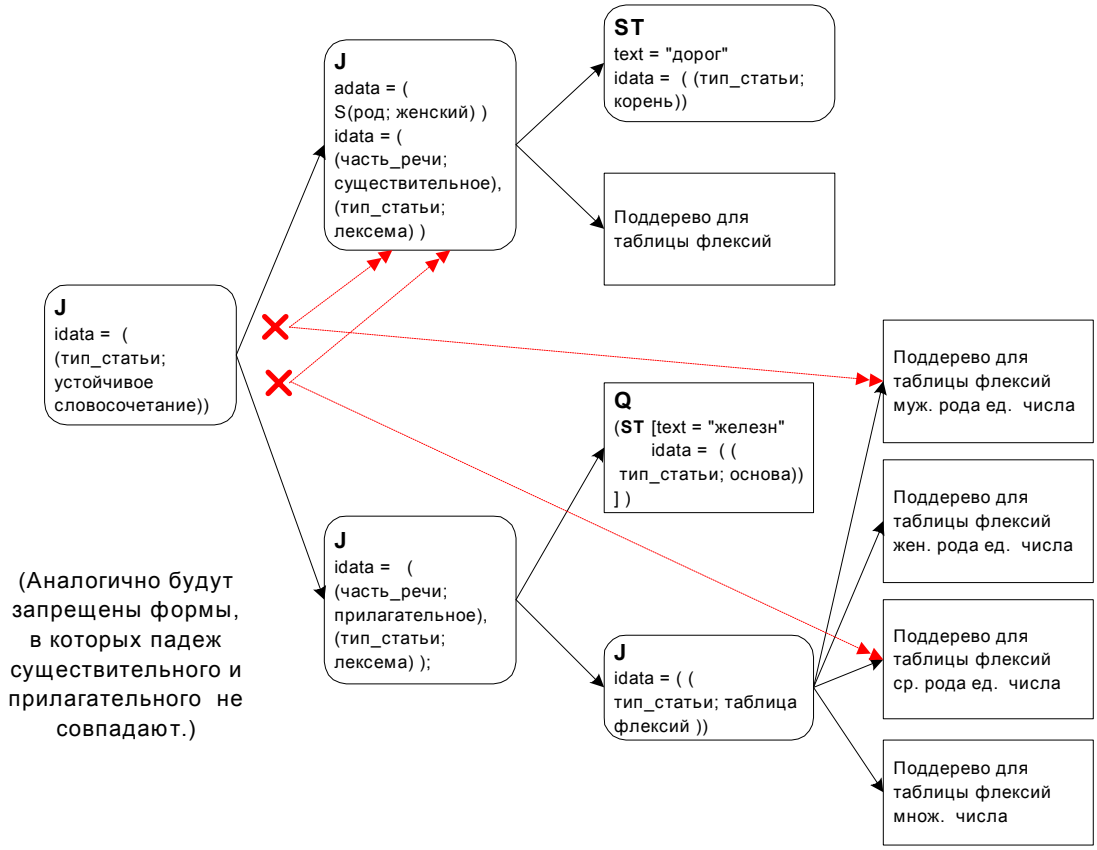

Рис. 5. Представление словосочетания

Рассмотрим представление словосочетания «железная дорога», которое изменяется аналогично третьему из рассмотренных типов изменения сложных слов (см. рис. 5). Наличие признака женского рода у существительного позволит "вырезать" из таблицы флексий прилагательного поддеревья с окончаниями, выражающими значения мужского и среднего рода. Аналогично при соединении получатся не все возможные комбинации падежных значений у существительного и прилагательного, а только те, где падежи совпадают.

В приведенных выше примерах мы рассмотрели, конечно, не все явления, с которыми можно столкнуться при описании лексического состава языка. Но, как нам представляется, выбранная нами модель обладает широкими выразительными возможностями. Однако мы не можем быть уверены в том, что содержание объектов и их свойств – это окончательны и устоялись. В настоящее время для проверки наших идей проводится работа по преобразованию информации из словаря русского языка "Скобки", разработанного в институте "Информэлектро", в формат нашей модели.

В этом словаре лексемы представлены в виде основ и связанных с ними таблиц флексий. В качестве следующего шага предполагается с помощью объектов-коллекций итеративно преобразовать такой формат описания лексем в морфемное представление. Одним из важных следствий описания морфемного состава слов является возможность естественным образом, за счет локальности описания данных, сделать словарь гнездовым.

Параллельно с преобразованием исходных словарей проводится работа по решению с использованием свойств объектной модели такой лексикографической задачи, как автоматизированное построение индексов словаря по запросу пользователя.

Список литературы

1. Буч Г. Объектно-ориентированный анализ и проектирование. М.: Издательство Бином, 2000.
2. Graham, Paul. ANSI Common Lisp. Prentice-Hall, New Jersey (USA), 1996
3. Диконов В.Г. Словарный запас. (Адрес статьи в Internet www.ets.ru/arc15-r.htm)

4. Описания и обзоры электронных словарей на сайтах www.ets.ru, www.lingvo.ru, www.multilex.ru, www.informatic.ru