ОБ ОДНОЙ СИСТЕМЕ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ МОРФЕМНО-ЛЕКСИЧЕСКОГО СТРОЯ АДЫГЕЙСКОГО ЯЗЫКА

В.Ш. Тлюстен

Адыгейский государственный университет, г. Майкоп

В работе рассматривается вопрос о построении формальных моделей морфемики адыгских языков в компьютерно-лингвистической системе DATR. На примере анализа словоизменительных парадигм глагола адыгейского языка демонстрируется методология подобных построений. Приводится DATR-теория, составляющая основу соответствующей формализации для некоторого представительного класса адыгейских глаголов.

Введение

Важнейшей задачей современной лингвистики является, как известно [1,3], задача интегрального описания языка, т.е. такого описания, в котором грамматика и словарь как части единой теории были бы согласованы друг с другом в отношении распределения языковой информации и форм ее записи.

Благодаря интенсивному развитию автоматизированных информационных технологий, на пути к решению указанной выше задачи, все большее распространение в лингвистической практике в последние годы получают методы математического и компьютерного моделирования, позволяющие с исчерпывающей точностью и полнотой выражать всё более сложные аспекты соотношения элементов внутренней структуры естественных языков.

К настоящему времени разработано и успешно используется большое число разнообразных компьютерно-лингвистических систем как прикладного, так и исследовательского назначения. Из прикладных систем, наибольшее применение получили программы проверки орфографии (спеллеры), специализированные диалоговые системы, “понимающие” естественный язык, системы автоматизированного перевода, обучающие лингвистические системы и т. п.

Автоматизированные процессоры исследовательского назначения представлены широким спектром морфологических, лексических, синтаксических и фонетических анализаторов, а так же компьютерных программ, ориентированных на комплексное представление, каталогизацию и обработку языковой информации.

Одной из наиболее развитых систем подобного рода, является разработанная Дэвидом Гиббоном исследовательская система DATR [1]. Целью проекта DATR, явно декларированной её автором, является создание мощной языково-инструментальной среды, которая как раз и позволяла бы строить формализованные лингвистические теории, удовлетворяющие требованию вышеуказанной задачи интегрального описания языка.

Представляя собой формальный метаязык, пригодный для точной фиксации многих грамматических явлений естественных языков, DATR, в то же время, реализует в себе функции языкового процессора. Это означает, что на основе указанных функций, исследователь, использующий DATR, может не только разрабатывать формально-лингвистические модели естественных языков (d-теории), но и легко проверять эти модели в автоматизированном режиме на их соответствие эмпирическим данным.

Такая возможность обусловлена тем, что вывод следствий из заложенных в d-теорию обобщений производится автоматически. Например, описав в среде DATR (в форме запроса к DATR - процессору) грамматические атрибуты некоторого лексического объекта, или класса таких объектов, при наличии соответствующей d-теории, можно получить список всех удовлетворяющих этому запросу лексем.

Основной целью настоящего исследования является, во-первых, изучение степени применимости системы DATR к решению задач формализации морфемики адыгских языков и, во-вторых, фактическое построение соответствующей формально-лингвистической datr-модели для достаточно представительного фрагмента адыгейской грамматики (в данной работе - для словоизменительной системы некоторых базовых форм адыгейского глагола).

Вышеуказанная цель, как и вытекающие из нее задачи, не тривиальны, т.к. грамматика адыгских языков является одной из самых сложных. Широко развитый аппарат аффиксации, большое количество диалектов, сложная фонетическая система - это далеко не все особенности языков этой группы. Но тем более полезна, как нам представляется, попытка построения формально-лингвистических моделей, уточняющих и/или обобщающих это многообразие во всей его полноте и сложности.

Морфология, которой посвящена данная работа, представляет собой ту часть грамматики, которая изучает различные формы существования и структурно-вариативного поведения лексических единиц самых различных грамматических категорий языка. Вместе с тем, в адыгейском языке, проблемы точного описания морфологических явлений, в наиболее концентрированном виде проявляются при структурно-логическом анализе морфемики глагольных словоизменений. Причины этого факта достаточно очевидны и обусловлены, в частности, следующими моментами [2].

Глагол - это исторически базовый элемент адыгейского языка, он сформировался раньше имени. В основе развития и формирования глагола лежит развитие местоимений. Именно притяжательные местоимения образуют личные префиксы адыгейского глагола.

Глагол в адыгейском языке является наиболее информативной частью речи. В предельных случаях, предложение может состоять из единственного глагола, который, в некоторых своих формах, включает в себя не только действие, но и поведение субъекта и объекта по отношению друг к другу.

Глагол отличается от других частей речи особой, свойственной только ему системой словообразования - спряжением. Спряжение глагола в адыгейском языке представляет собой его изменение по временам, наклонениям, лицам и числам. В адыгейском глаголе широко представлена система локальных приставок. Он располагает так же большим количеством модальных частиц - суффиксов.

В глаголе адыгейского языка морфологически выражаются не только категории лица, числа, времени и наклонения, но и отрицание, утверждение, вопрос, каузатив, совместность действия, локализация, направительные отношения и др.

И, наконец, адыгейский глагол очень богат своими формами. Он может одновременно изменяться по нескольким лицам - в максимальных формах по четырем и пяти лицам.

Таким образом, глагол, несомненно, представляет собой одно из самых интересных, сложных, информативных и первичных явлений, во многом определяющих логику построения и функционирования всех остальных грамматических категорий адыгейского языка.

Поэтому данную работу, насколько нам известно, по существу, первую в области компьютерной лингвистики адыгейского языка, мы посвятим построению формально-лингвистической теории, охватывающей некоторые ключевые моменты образования именно глагольных словоформ.

Говоря точнее, данная работа преследует ряд конкретных целей.

В лингвистическом и методологическом плане, основная ее цель состоит в построении исходного ядра формально-компьютерной модели (d-теории), решающей, в частности, задачу эффективного представления глагольно-лексической базы адыгейского языка, а также в демонстрации методологии подобных построений, для адыгских языков вообще.

В математическом и технологическом плане, целью данной работы является теоретическая и практическая проверка адекватности современных методов и инструментальных средств компьютерной лингвистики, в частности, системы DATR, наиболее сложным явлениям грамматик естественных языков, примером которых может служить контекстно-функциональное поведение и морфемная структура глаголов адыгейского языка.

И, наконец, еще одной, возможно, главной целью данной публикации, является желание её автора привлечь внимание ученых-адыговедов к средствам и методам современной компьютерной лингвистики, быть может, инициировав, тем самым, зарождение новой ветви адыго-лингвистических исследований, направленных, в конечном счете, как на обогащение методологии развития теории адыгских языков, так и на создание эффективных прикладных систем автоматизированной их обработки.

1. КОМПЬЮТЕРНО-ЛИНГВИСТИЧЕСКАЯ СИСТЕМА DATR

1.1. D-теория и лексикон

Начнем с краткого введения в систему DATR. В стандартной терминологии архитектуры экспертных систем и баз знаний, на DATR можно смотреть как на среду представления лингвистических знаний, подобную семантической сети.

Такая сеть, хранящая определенный фрагмент формализованных знаний о некотором естественном языке, называется *теорией* (в нашей терминологии, *d-теорией*) и представляет собой ориентированный граф, вершины (узлы) которого соответствуют языковым понятиям (например, грамматическим категориям), а ребра выражают связи (родовидовые и иные) между этими понятиями. Все узлы именованы, и каждый из них содержит определенное множество атрибутов, характеризующих соответствующее понятие. Каждый атрибут обладает значением, которое либо фиксировано (тогда оно явно хранится в данном узле), либо косвенно может быть вычислено на основе информации, наследуемой данным узлом из других узлов, связанных с данным и представляющих, например, более общие понятия (возможно, надклассы в иерархии родовидового соподчинения классов).

Синтаксически, информация о каждом узле задаётся в следующей форме:

ИМЯ_УЗЛА:
Атрибут1==выражение1
Атрибут2==выражение2
…
АтрибутN==выражениеN .

Ниже представлен пример тривиальной (для простоты, пока огрублённый в части форм настоящего времени) теории, состоящей всего из пяти узлов:

PAST_SUFX: % представление суффиксов прошедшего времени
<indef> == гъэ
<perf> == гъагъэ
<cont> == щтыгъэ
<imperf> == щтыгъагъэ.

FUTURE_SUFX: % представление суффиксов будущего времени
<1_fut> == щт
<2_fut> == н.

VERB: % представление морфологии времени адыгейского глаголе
<mor prsnt> == "<mor rt>"
<mor past> == "<mor rt>" PAST_SUFX:<>
<mor fut> == "<mor rt>" FUTURE_SUFX:<>.

GO: % представление временных форм адыг. глагола к1о
< > == VERB
<mor rt> == к1о.

WRITE:% представление временных форм адыг. глагола тхэ
< > == VERB
<mor rt> == тхэ.

Здесь, PAST_SFX, FUTURE_SFX, VERB, GO, WRITE - это имена узлов, имеющих очевидную грамматическую интерпретацию. Любая последовательность знаков, начинающаяся с символа ‘ %’ (и до конца соответствующей строки) является комментарием, а строки вида “**Атрибут$_i$==выражение$_i$**” называются *правилами* и служат в качестве основной формы представления лингвистической информации в узлах.

Знаковые последовательности, стоящие в правилах слева от знака "==" (такие как ***<indef>, <mor prsnt>*** и др.) - это обозначения *атрибутов* соответствующих грамматических категорий (узлов), а конструкции, стоящие справа от знака "==" (например, ***гъэ, щтыгъагъэ, "<mor rt>", "<mor rt>" FUTURE_SUFX:<>*** и др.)- это либо константные *значения* соответствующих атрибутов ***(гъэ, щтыгъагъэ),*** либо *выражения* общего вида ***("<mor rt>", "<mor rt>" FUTURE_SUFX:<>***), позволяющие вычислять такие значения, т. е. находить их, возможно, на основе дополнительной информации в том же самом или в других узлах.

Например, рассмотрим узел VERB. Он отвечает грамматической категории глагола и характеризуется тремя *морфологическими* атрибутами: ***<mor prsnt>, <mor past>*** и ***<mor fut>,*** выражающими, соответственно, морфологию настоящего, прошедшего и будущего времени некоторого подкласса адыгейских глаголов. Значение атрибута ***<mor prsnt>*** вычисляется на основе *выражения* “***<mor rt>",***

являющегося, в свою очередь, ссылкой (на что указывают кавычки) на соответствующий атрибут узла, представляющего ту или иную *лексему* (GO или WRITE**),** в которой этот, соответствующий атрибут (т.е. **<*mor rt*>**) имеет конкретное *значение* (***к1о*** или ***тхэ*)**. Таким образом, окончательное (терминальное) значение атрибута **<*mor prsnt*>** узла VERB может быть вычислено только при фиксации конкретной лексемы класса VERB и в данной теории будет равным либо значению ***к1о***, либо значению ***тхэ*.**

С другой стороны, значение атрибута **<*mor past*>** узла VERB вычисляется в форме последовательности (конкатенации) двух значений - вычисленного так как указано выше значения **“<*mor rt*>”** и одного из значений атрибутов, наследованных от узла PAST_SUFX.

Заметим, что если такой узел, например, как GO выражает единственную словоформу ***к1о*,** то другому, *абстрактному* узлу VERB, в данной теории соответствует целый класс словоформ:

к1о	***тхэ***
к1о гъэ	***тхэ гъэ***
к1о гъагъэ	***тхэ гъагъэ***
к1о щтыгъэ	***тхэ щтыгъэ***
к1о щтыгъагъэ	***тхэ щтыгъагъэ***
к1о щт	***тхэ щт***
к1о н	***тхэ н***

Таким образом, “абстрактные” узлы PAST_SFX, FUTURE_SFX, VERB, рассматриваемые с учётом их взаимосвязей, собственно, и выражают теорию выделенного нами фрагмента грамматики адыгейского языка (обобщённое представление соответствующих морфологических правил), в то время, как узлы GO и WRITE, напротив, - представляют конкретную лексемную базу (*лексикон*) на которой может быть реализована словообразовательная функция данной теории.

Собственно, d-теорию, дополненную соответствующим лексиконом в дальнейшем будем называть *расширенной d-теорией* или, более кратко, *lex-теориией*.

Добавляя к лексикону новые статьи (узлы, соответствующие другим глагольным основам), можно расширять охватываемую данной теорией лексику. В то же время, обогащая саму теорию, можно достигать большей вариативности словоформ, потенциально порождаемых на основе данной лексической базы.

При этом следует иметь в виду, что приведённая здесь в качестве примера теория отражает *морфологию*, но *не орфографию* соответствующих глагольных форм. Так, словоформа ***“к1о гъэ”*** орфографически должна была бы выглядеть как “***к1уа гъэ***”. Введение в теорию статей, определяющих правила орфографии в принципе не представляет трудностей и в данной работе специально не рассматривается.

1.2. Запросы к DATR-теории

Упомянутая выше интерпретация расширенной (т.е. включающей лексикон) d-теории как базы знаний, позволяет строить систему извлечения данных из этой базы в форме соответствующих к ней *запросов*. В DATR принята следующая структура запроса:

ИМЯ_УЗЛА: запросный_атрибут

Здесь, ИМЯ_УЗЛА определяет узел-адресат запроса, а *запросный_атрибут* представляет собой заключённую в угловые скобки (<,>) строку, содержащую упорядоченный набор (кортеж, путь) *атомарных атрибутов (атомов),* каждый из которых задаёт отдельный признак запрашиваемой лексической единицы.

Например, задаваемая в контексте построенной выше lex-теории строка:

GO: <mor past cont >

выражает запрос хранимой в узле GO лексемы ***к1о*** (*идти*), в морфологической форме (***mor***) прошедшего (***past***) продолженного (***cont***) времени. В этом запросе, кортеж <***mor past cont***>, состоящий из трёх атомарных атрибутов, очевидно и есть запросный атрибут. Сопоставив данный запрос с рассматриваемой нами lex-теорией, DATR-процессор, после соответствующих логических “умозаключений”, в качестве ответа выдаст строку: ***к1о щтыгъэ.***

Рассмотрим в общих чертах, на приведённом примере, порядок обработки запросов DATR-процессором.

Первоначальный запрос ***GO: <mor past cont>***, как уже указывалось, адресуется узлу GO. Среди атрибутов этого узла (левых частей правил вида **Атрибут$_i$ == выражение$_i$**) ищется тот, который представляет собой самый длинный путь, совпадающий с началом пути, указанной в запросе. Если такого атрибута не найдётся, то запрос считается неудачным и процесс завершается.

В нашем случае, узел GO включает два правила и, соответственно, имеет два атрибута: <> (атрибут, содержащий пустой путь) и ***<mor rt>***.

Путь, представленный в запросном атрибуте ***(<mor past cont>)*** не начинается со строки ***<mor rt>.*** Поэтому отвечающее этой строке в узле GO правило ***(<mor rt> == к1о***) отвергается. Но зато, как можно всегда считать для любой строки, тот же путь префиксируется пустой строкой. А значит и будет выбрано правило, атрибут в левой части которого представлен пустой строкой (т.е. правило <> == ***VERB***).

Далее, строка текущего запросного атрибута накладывается на строку, представляющую выбранный атрибут (в нашем примере, <>) так, что начала этих строк оказываются совмещёнными, а правая, не совместившаяся часть запросного атрибута (при наличии таковой), дописывается в качестве “хвоста” любого пути, представленного в выражении (…== **выражение$_i$**) в правой части выбранного правила.

На основе возможно, модифицированного таким образом выражения (…== **выражение$_i$**), формируется, в общем случае, новый запрос и т.д.

Процесс в целом считается результативным, если он завершается построением непустой последовательности атомарных атрибутов.

В нашем примере, правая часть выбранного на данном этапе правила никаких путей не содержит и представляет собой ссылку на другой узел (VERB), куда и переадресуется текущий запрос. В результате, получаем вторичный запрос: ***VERB:<mor past cont>.***

При отслеживании динамики обработки исходного запроса, особого внимания заслуживают подстроки вида “***< … >***” в правых частях выбираемых правил. Кавычки в таких подстроках означают, что соответствующий атрибут, при формировании на его основе вторичного запроса, следует рассматривать в контексте узла *исходного* запроса (в нашем примере, узла GO).

Так, продолжая наш пример, мы видим что после возникновения вторичного запроса ***VERB: <mor past cont>,*** дальнейшее продвижение по цепочке вычислений, приведёт к выбору правила ***<mor past> == “<mor rt>” PAST_SUFX:<>*** в локальном контексте узла VERB. Далее, в результате наложения запросного атрибута ***<mor past cont>*** на левую часть указанного правила, оказавшийся избыточным атом ***cont*** будет, согласно ранее сказанному, добавлен в качестве постфикса всех путей выражения в правой части этого правила. В результате, в правой части получим модифицированное выражение***: “<mor rt cont>” PAST_SUFX:<cont>*** , которое разбивает дальнейший поиск интересующей нас лексемы на последовательную обработку двух независимых запросов***: GO: <mor rt cont>*** (контекст GO возник в результате кавычек!) и ***PAST_SUFX:<cont>***. Эти запросы завершаются, соответственно, выдачей атомов ***кIо*** и ***щтыгъэ***, что окончательно даёт результирующую лексему ***кIо щтыгъэ.***

Более точное и полное описание как самой системы DATR, так и правил интерпретации DATR-процессором поисковых запросов, можно найти, например, в работах [1,3].

2. ОСНОВЫ DATR-ТЕОРИИ ОДНОЛИЧНЫХ ГЛАГОЛОВ АДЫГЕЙСКОГО ЯЗЫКА

2.1 Временные формы адыгейского глагола

Рассмотрим принципы аксиоматизации механизма порождения временных форм адыгейского глагола, на примере непереходного динамического глагола ***к1о***. Для этого воспроизведём наши предыдущие построения в виде следующей системы узлов начальной lex-теории.

Узел, содержащий суффиксы прошедшего времени:

```
PAST_SUFX:
<indef> == гъэ              % прошедшее совершенное время;
<perf> == гъагъэ            % давнопрошедшее время;
 <cont> == щтыгъэ           % прошедшее несовершенное;
<imperf> == щтыгъагъэ.  % давнопрошедшее несовершенное
```


Узел, содержащий суффиксы будущего времени:
FUTURE_SUFX:
<1_fut> == щт % будущее первое или фактическое
<2_fut> == н. % будущее второе или необходимое;

Отметим что, так как настоящее время образуется непосредственно от основы глагола, без всяких специальных суффиксов, то и соответствующего узла в lex - теории нет.

Узел, содержащий корень глагола (представляет лексикон):
GO:
< > == VERB % наследует все признаки глагола;
<mor rt> == к1о. % дополнительный признак - корень

В данной работе считается, что основной формой (основой) глагола в адыгейском языке служит форма второго лица, единственного числа, повелительного наклонения.

Следует отметить, что если любое время глагола, кроме настоящего, образуется от основы глагола путем присоединения к ней нужного суффикса, то узел, содержащий правила образования глагольных словоформ, будет включать следующие предложения:
VERB:
<mor prsnt> == "<mor rt>" % морфологически, форма
% настоящего времени определяется корнем
<mor past> == "<mor rt>" PAST_SUFX:<>
<mor fut> == "<mor rt>" FUTURE_SUFX:<>.

Далее можно обратиться к построенной lex - теории со следующими, например, запросами:
GO: <mor prsnt>
GO: <mor past indef >
GO: <mor fut 1_fut>.

Исходя из алгоритма обработки запросов, описанного нами ранее, в качестве ответов будут выданы, соответственно, слова: ***к1о*** - "идет" (данная "безличная" версия теории здесь приводит к некорректности!), ***к1огъэ*** - "пошел", ***к1ощт*** - "пойдет".

Образование настоящего времени происходит путем присоединения к основе глагола нулевого суффикса, а образование всех других времен происходит по правилу: *Основа_глагола + суффикс*.

Заметим, кроме того, что формы адыгейского глагола настоящего времени требуют указания префикса лица, и его отсутствие - явная ошибка (см. первую из трёх выше указанных форм), которая будет устранена при уточнении теории в следующем разделе.

Структуру узла VERB можно упростить путем введения дополнительного узла для обобщения суффикса времени:
TENSE_SUFX:
<prsnt> ==
<past> == PAST_SUFX: <>
<fut> == FUTURE_SUFX: <>.

Введенный узел, в зависимости от указанного в запросе времени, ссылается на тот или иной из индивидуальных суффиксальных узлов.

Теперь структура узла VERB допускает следующую перестройку:
VERB:
<mor > == "<mor rt>" TENSE_SUFX: < >.

При этом сохраняется прежняя функциональность. Действительно, выдав, например, запрос ***GO:<mor past imperf>,*** мы получим следующую цепочку логического вывода:

GO:<mor past imperf> => VERB:<mor past imperf> =>
=> VERB:"<mor rt past imperf>" TENSE_SUFX:<past imperf> =>

=> ***GO:<mor rt past imperf> TENSE_SUFX:<past imperf>*** =>
=> ***кIо TENSE_SUFX:<past imperf>*** => ***кIо PAST_SUFX: <imperf>***
=> ***кIо щтыгъагъэ***

Добавим теперь в теорию возможность выражения отрицаний. Отрицательные временные формы образуются с помощью суффикса ***-эп***, находящегося в конце глагола после всех словообразовательных, временных и числовых формантов.

Отметив так же и то, что особенностью отрицательной формы настоящего времени динамических глаголов является присоединение суффикса ***-эп*** к основе глагола посредством суффикса ***-р***, в узел VERB добавим соответствующие правила и введем новый, содержащий аффиксы отрицания узел SUFX :

```
VERB:   % учёт отрицательных форм
        …
      <mor not> ==  <mor> SUFX: <notf>
      <mor not prsnt> == <mor prsnt> p SUFX: <notf>.

        SUFX:  %  аффиксы отрицания
          <notf> == эп.
```

Теперь, при обработке, например, запросов:
GO: <mor not prsnt>
GO: <mor not past indef >
GO: <mor not fut 1_fut> ,
процессор DATR сгенерирует, соответственно, следующие результаты: ***к1орэп*** - “не идет”, ***к1огъээп*** - “не шел”, ***к1ощтэп*** - “не пойдет”.

Следует заметить, что полученное слово ***к1огъээп*** является не вполне корректным, т.к. правильно было бы ***к1уагъэп***. Здесь снова возникает упоминавшаяся выше проблема возможного нарушения правил орфографии в некоторых из порождаемых теорией словоформ.

2.2 Изменение глагола по лицам и числам

Личный префикс в адыгейском глаголе является показателем лица и при этом, префиксы, выражающие лица, образуются от личных местоимений. Изменение глагола по лицам является интересной особенностью, которая отличает рассматриваемый нами язык от большинства европейских и восточных языков. В одной и той же глагольной форме одновременно может быть выражено несколько лиц (до четырех-пяти), в то время как, например, в русском - лишь одно.

Как правило, личные аффиксы первого и второго лица всегда представлены в глаголе. Третье же лицо может быть представлено в одних случаях - префиксом (в настоящем времени), а в других - нулевой формой (в прошедшем и будущем временах). Показателем множественности третьего лица является суффикс ***-х.***

Необходимо так же учесть, что личные префиксы настоящего времени отличаются от личных префиксов прошедшего и будущего времени, что может быть отражено введением в lex-теорию узлов PERS_PRSNT и PERS_NPRSNT, имеющих следующую структуру:

```
PERS_PRSNT:  %  содержит личные аффиксы для
                        % настоящего  времени
   <s_1> ==  сэ          %  ед.ч., 1 лицо
   <p_1> == тэ               %  мн.ч., 1 лицо
   <s_2> == о                %  ед.ч., 2 лицо
   <p_2> == шъо                 %  мн.ч., 2 лицо
   <s_3> == ма           %  ед.ч., 3 лицо
   <p_3> ==<s_3> .           %  мн.ч., 3 лицо

PERS_NPRSNT:   % содержит личные аффиксы для прошедшего
                        % и будущего  времени
    <s_1> == сы       %  ед.ч., 1 лицо
```


<p_1> == ты % мн.ч., 1 лицо
<s_2> == у % ед.ч., 2 лицо
<p_2> == шъу % мн.ч., 2 лицо
<s_3> == % для ед.ч.,3 лицо - нулевой аффикс
<p_3> == <s_3>. % и для мн.ч., 3 лиц. - то же

Для обобщения категории лица, введём ещё один узел - PERS. Этот узел, выполняет функцию диспетчеризации, в зависимости от рассматриваемой временной формы глагола, ссылаясь на тот или иной из указанных выше двух узлов:

PERS:
<prsnt> == PERS_PRSNT:<>
<past> == PERS_NPRSNT:<>
<fut> == PERS_NPRSNT:<>.

В связи с тем, что необходимо так же учесть лицо, стоящее в запросе после временного суффикса, обозначим атом, выражающий любое сочетание “время-лицо”, переменной $t_prs и изменим структуру узла TENSE_SUFX следующим образом:

 # vars $t_prs: s_1 p_1 s_2 p_ two s_3 p_3.
TENSE_SUFX:
<prsnt $t_prs> ==
<past $t_prs> == "PAST_SUFX:<>"
<fut $t_prs> == "FUTURE_SUFX:<>".

Предшествующая определению узла строка называется *директивой* и описывает переменную ***$t_prs***, перечислением множества возможных её значений.

Использование переменных существенно сокращает записи. Так, обобщённое правило ***<past $t_prs> == …*** теперь способно заменить шесть правил:

<past s_1> == …
<past s_2> == …
…
<past p_3> == … .

Более того, выбор правила ***<past $t_prs> == …,*** например, на основе запросного атрибута ***<past s_2>,*** приводит к запоминанию значения атома ***s_2*** в переменной ***$t_prs,*** которая может использоваться для настройки правой части указанного правила именно на это атомарное значение.

Продолжая построение нашей теории, в узел SUFX добавим еще одно определение, соответствующее суффиксу множественного числа:

SUFX:
<notf> == эп
<plur> == х. % множественное число

а узел VERB перепишем следующим образом:

 # vars $tens: prsnt past fut.
VERB:
<mor> == PERS: < > "<mor rt>" TENSE_SUFX:<>
<mor $tens p_3 > == PERS:<$tens p_3> "<mor rt>"
TENSE_SUFX:<$tens p_3> SUFX:<plur>
<mor not> == <mor> SUFX:<notf>
<mor not prsnt> == <mor prsnt> p SUFX:<notf>.

Теперь при обработке соответствующего типа запроса к теории, дойдя до узла TENSE_SUFX, машина вывода вместо переменной подставит значение, отвечающее категории времени в запросе. Если это значение не равно ни одному из возможных значений переменной, то ответом на запрос будет нулевой атом.

Рассмотрим, например, следующие запросы:

GO: <mor prsnt s_1>
GO: <mor past p_3 perf>

GO: <mor fut p_2 1_fut>

При обработке второго из указанных запросов машина вывода вместо переменной ***$tens*** подставит ***past***. А в результате обработки всех трёх запросов, соответственно, получим: ***сэк1о*** - “я иду”, ***к1огъагъэх*** - “они ходили”, ***шъук1ощт*** - “вы пойдете”.

2.3 Наклонения

Известно, что глагол адыгейского языка изменяется не только по временам и лицам. Категория времени тесно переплетается с категорией наклонения. Из одной и той же основы могут быть образованы не только времена, но и наклонения (модальные формы): сослагательное, желательное, условное и т.д.

Глагольная форма наклонения выражает собой отношение содержания речи к действительности. В общем случае глагольная форма наклонения образуется от основы глагола путем присоединения соответствующего суффикса. Изменим раннее созданную lex-теорию таким образом, чтобы возможно было образовывать и формы глагола с наклонением. Для этого добавим следующие узлы:

```
BEND:  % наклонения  глагола
    <infin_1> == у            % деепричастие_1
    <cond_1> == мэ            % условное_1
    <expct_1> == щтын .   % предположительное_1
```

Так как в различных временах, после атома, фиксирующего временную категорию и перед атомом, указывающим наклонение, стоит разное количество дополнительных слов, которые следует отфильтровать с помощью переменных, оказывается необходимым ещё один узел:

```
# vars  $any  .  % так описанная  переменная принимает любые
                 % значения
MOOD:
     <prsnt $t_prs> == BEND:<>
     <past $t_prs $any> == BEND:<>
     <fut $t_prs $any> == BEND:<>.
```

Теперь в узел VERB можно добавить правило:

<mor md>== <mor> MOOD:<>

Тогда при обработке следующих запросов:

GO: <mor md prsnt s_2 infin_1>

GO: <mor md past p_3 perf cond_1>

получим словоформы: ***ок1оу*** - “ты, идя”, ***к1огъагъэхмэ*** - “если они ходили” (в орфографии, требующей коррекции).

Отрицательные формы наклонений глагола образуются с помощью префикса ***-мы*** , который ставится после показателя лица, и суффикса ***-эп***. Для отражения этого факта в lex-теории, в узел SUFX добавим еще одно значение, соответствующее префиксу ***-мы***. В итоге получим следующее:

```
SUFX: % суффиксы отрицания и множественности
     <notf> == эп
     <notpr> == мы
     <plur> == х.

VERB:
 % образование времен:
  <mor > == "<mor rt>" TENSE_SUFX:< >
  <mor $tens p_3 > == PERS:<$tens p_3> "<mor rt>"
                          TENSE_SUFX:<$tens p_3> SUFX:<plur>
 % отрицательные формы:
  <mor not> ==  <mor> SUFX:<notf>
  <mor not prsnt> == <mor prsnt>p SUFX:<notf>
 % аффиксы наклонений:
  <mor md> == <mor > MOOD:<>
```


<mor notf md> == <mor not> MOOD:<>
<mor notpr md> == PERS: < > SUFX: <notpr> "<mor rt>"
TENSE_SUFX:< > MOOD: <>
<mor notpr md $tens p_3> == PERS: <$tens p_3> SUFX:
<notpr> "<mor rt>" TENSE_SUFX:<$tens p_3>
SUFX: <plur> MOOD: <$tens p_3>.

Примерами получения отрицательных форм является обработка следующих запросов:
GO: <mor notpr md prsnt s_1 infin_1>
GO: <mor notf md past p_1 cont expct_1>

В результате этой обработки, в качестве ответа, lex-теорией будут выданы, соответственно, словоформы: ***сэмык1оу*** - “я не идя***”, тык1ощтыгъээпщтын*** - “мы, вероятно, не ходили”.

2.4 Глагольные частицы

Частицы являются еще одной разновидностью суффиксальных окончаний глагола в адыгейском языке. Они могут вносить в основное значение глагола различные видовые оттенки: законченности, повторяемости и т.д.

Большинство глагольных частиц ставится в конце глагола после всех словообразовательных, временных и числовых формантов.

Однако существуют частицы, которые стоят перед временным показателем, например, частица повторности (***-жьы***). Учёт таких частиц удобно осуществить следующим образом. Обобщим понятие морфологического корня до понятия *базы*, включив в это новое понятие помимо, собственно, корня, так же и все примыкающие к нему частицы. Теперь, всюду в нашей теории (не затрагивая при этом лексикона), заменим ссылки на морфологический корень, ссылками на базу. Саму же базу определим как совокупность дополнительных атрибутов узла VERB, которые выражают, в зависимости от запроса, любую из интересующих нас комбинаций морфологического корня с глагольными частицами.

Для реализации этой идеи и учёта соответствующих частицам морфологических вариаций глагола:

А) пополним теорию двумя новыми узлами:

PARTIC: % представление глагольных частиц
<confrm1> == ба % подтвердительная
<rept> == жьы. % повторности
PARTICLE: % узел диспетчеризации запросов глагольных частиц
<prsnt $t_prs> == PARTIC:<>
<past $t_prs $any> == PARTIC:<>
<fut $t_prs $any> == PARTIC:<>.

Б) Заменим в узле VERB все вхождения подстроки ***“<mor rt>”*** на подстроку ***<base>;***

В) Добавим в узел VERB правила:

% глагольные частицы:
<mor prticl> == <mor> PARTICLE:<>
<mor prtic> == PERS: <> <base> PARTICLE:<> TENSE_SUFX:<>
<mor prtic $tens p_3> == PERS: < $tens p_3> <base>
PARTICLE:< $tens p_3> TENSE_SUFX:
< $tens p_3> SUFX:<plur>
% отрицательные формы глаголов с частицами:
<mor notf prtic> == <mor prtic> SUFX:<notf>
<mor notf prtic $tens p_3> == <mor prtic $tens p_3>
SUFX:<notf>
% база глагола:
<base> == “<mor rt>”
<base rept> == “<mor rt>” PARTIC:<rept>

Можно заметить, что в новой редакции узла VERB учтены так же и отрицательные варианты глагольных словоформ с частицами. Эти варианты образуются как при помощи префикса ***-мы***, так и (для частиц, стоящих перед временным показателем) посредством суффикса ***-эп.***

Рассмотрим теперь запросы:

GO: <mor prticl prsnt s_1 confrm1>
GO: <mor prtic fut s_2 1_fut rept>
GO: <mor notf prtic fut s_2 1_fut rept>

В результате обработки указанных запросов, получим, соответственно, лексемы: ***сэк1оба*** - “ведь иду же”, ***ук1ожьыщт***- “ты вернешься (туда)”, ***ук1ожьыщтэп*** - “ты не вернешься (туда)”, имеющие правильную морфологическую структуру.

Разработанная нами lex-теория в окончательном виде, с дополнительно уточненными правилами и с частично подстроенной орфографией, приведена ниже.

% Система словоизменения одноличного, динамического, непереходного глагола адыгейского языка: настоящее, прошедшее и будущее времена с наклонениями и частицами

```
%    D A T R - T E O P И Я:

% список переменных и их возможных значений:
# vars $t_prs: s_1 p_1 s_2 p_2 s_3 p_3.
# vars $non_p3: s_1 p_1 s_2 p_2 s_3 .
# vars $1_2_fut: 1_fut 2_fut .
# vars $nprsnt: past fut.
# vars $p3: p_3 .
# vars $tens: prsnt past fut.
# vars $any . % значение переменной может быть любым

PAST_SUFX:
   <indef> ==  гъэ          % прошедшее совершенное время
   <perf> ==  гъагъэ        % давнопрошедшее время
   <cont> ==  щтыгъэ        % прошедшее несовершенное
   <imperf> ==  щтыгъагъэ.  % давнопрошедшее несовершенное

FUTURE_SUFX:
   <1_fut> ==  щт           % будущее первое или фактическое
   <2_fut> ==  н.           % будущее второе или необходимое

PERS_PRSNT:        % содержит личные аффиксы наст.вр.
   <s_1> ==  сэ            % ед.ч., 1 лицо
   <p_1> == тэ             % мн.ч., 1 лицо
   <s_2> == о              % ед.ч., 2 лицо
   <p_2> == шъо            % мн.ч., 2 лицо
   <s_3> == ма             % ед.ч., 3 лицо
   <p_3> ==<s_3> .         % мн.ч., 3 лицо

PERS_NPRSNT:   % содержит личные аффиксы прошедшего и буд. вр.
   <s_1> == сы             % ед.ч., 1 лицо
   <p_1> == ты             % мн.ч., 1 лицо
   <s_2> == у              % ед.ч., 2 лицо
   <p_2> == шъу            % мн.ч., 2 лицо
   <s_3> ==                % ед.ч., 3 лицо- нулевой аффикс
   <p_3> == <s_3>.         % мн.ч., 3 лиц. - то же

PERS: % узел диспетчеризации личн. префиксов по времени
   <prsnt> == PERS_PRSNT:<>
   <past> == PERS_NPRSNT:<>
```



```
    <fut> == PERS_NPRSNT:<>.

BEND:  % наклонения  глагола:
    <infin_1> ==  у            % деепричастие_1
    <cond_1> ==  мэ            % условное_1
    <expct_1> ==  щтын.        % предположительное_1

MOOD: % узел диспетчеризации наклонений по времени
    <prsnt $t_prs> == BEND:<>
    <past $t_prs $any> == BEND:<>
    <fut $t_prs $any> == BEND:<>.

TENSE_SUFX: % узел диспетчеризации временных суффиксов
    <prsnt $t_prs> ==
    <past $t_prs> == "PAST_SUFX:<>"
    <fut $t_prs> == "FUTURE_SUFX:<>".

SUFX: % суффиксальные окончания отрицаний и множ.числа
    <notf> ==  эп
    <notpr> ==  мы
    <plur p_3 $1_2_fut> == ы х      % учёт орфографии
    <plur p_3 $any confrm1> == х э  % учёт орфографии
    <plur> ==  х.
PARTIC:% глагольные частицы
    <confrm1> == ба                 % подтвердительная
    <rept> == жьы.                  % повторности
PARTICLE: % диспетчеризация глагольных частиц
    <prsnt $t_prs> == PARTIC:<>
    <past $t_prs $any> == PARTIC:<>
    <fut $t_prs $any> == PARTIC:<>.

VERB: % представление морфологии адыг. глагола
    % образование времен:
    <mor > == PERS: < > <base> TENSE_SUFX:< >
    <mor $tens p_3 > == PERS:<$tens p_3> <base>
                TENSE_SUFX:< $tens p_3 > SUFX:<plur p_3 >

    % отрицательные формы:
    <mor not> ==  <mor> SUFX:<notf>
    <mor not prsnt $t_prs> == PERS_NPRSNT:<$t_prs> <base> p
                                  SUFX:<notf>
    <mor not prsnt p_3> == <base>SUFX:<plur>э р SUFX:<notf>

    % аффиксы наклонений:
    <mor md> == <mor> MOOD:<>
    <mor md $tens p_3> == <mor $tens p_3>э
                          MOOD:<$tens p_3>    % орфография
    <mor md fut $non_p3 1_fut infin_1> ==          % орфография
            <mor fut $non_p3 1_fut infin_1>э BEND:<infin_1>
    <mor notf md> == <mor not> MOOD:<>
    <mor notpr md $tens> == PERS_NPRSNT:<> SUFX:<notpr> <base>
                         TENSE_SUFX:<$tens> MOOD:<$tens>
    <mor notpr md $tens p_3> == PERS_NPRSNT: <p_3> SUFX:<notpr p_3>
              <base>TENSE_SUFX:<$tens p_3> SUFX:<plur p_3>э
                                  MOOD:<$tens p_3>
    <mor notpr md fut $non_p3 1_fut infin_1> ==
            PERS_NPRSNT: <$non_p3> SUFX:<notpr> <base>
```



```
            TENSE_SUFX:<fut $non_p3 1_fut>э BEND:<infin_1>
    <mor notpr md $nprsnt $t_prs $any expct_1 >==
            <mor notf md $nprsnt $t_prs $any expct_1>

    % частицы:
    <mor prticl> == <mor> PARTICLE:<>
    <mor prtic> == PERS: < > <base> PARTICLE:<>  TENSE_SUFX:<>
    <mor prtic $tens p_3 > == PERS:<$tens p_3> <base>
      PARTICLE:<$tens p_3>TENSE_SUFX:< $tens p_3 >SUFX:<plur p_3>
    <mor prtic prsnt p_3 rept> ==<mor prsnt p_3 rept>
    <mor notf prtic> == <mor prtic> SUFX:<notf>
    <mor notf prtic $tens p_3> == <mor prtic $tens p_3>SUFX:<notf>
    <mor notf prtic prsnt $t_prs rept>== <mor not prsnt $t_prs rept>

    % база глагола:
    <base> == "<mor rt>"
    <base rept> == "<mor rt>"  PARTIC:<rept>

%   Л Е К С И К О Н:
GO:  % модельная лексическая основа (глагол "идти")
    <> == VERB          % наследует все признаки глагола;
    <mor rt> == к1о.    % дополнительный признак - корень
```

Заключение

Пример системы автоматически порождённых DATR- процессором следствий из указанной теории представлен в Приложении1.

На основе анализа этого примера можно заключить, что предложенная в данной работе d-теория, в целом, адекватно отражает морфологию соответствующего подкласса одноличных глаголов.

Следовательно, как сама эта теория, так и тот формально-логический и программный инструментарий, на котором она основана, могут быть использованы в качестве исходной базы построения полных систем формального моделирования морфемики адыгских языков - необходимого шага на пути к созданию высокотехнологичных программных средств компьютерной обработки текстов, представленных на этих языках.

Литература

1. *Evans, Roger and Gerald Gazdar*. 1996. DATR: A language for lexical knowledge representation. Computational Linguistics. Vol. 22, No 2, 167-216.

2. *Рогава Г.В., Керашева З.И.* Грамматика адыгейского языка. – Майкоп: Адыг. книж. изд-во, 1966. - 464 с.

3. http: // isabase.philol.msu.ru / DATR / zdatr_doc / html. - электронный учебно-методический ресурс, посвященный использованию DATR в МГУ им.Ломоносова.

ПРИЛОЖЕНИЕ 1.

ЛЕКСЕМНО-МОРФОЛОГИЧЕСКИЙ РЯД УЗЛА: GO

==========

< prsnt s_1 >	сэ-к1о
< prsnt s_2 >	о-к1о
< prsnt s_3 >	ма-к1о
< prsnt p_1 >	тэ-к1о
< prsnt p_2 >	шъо-к1о
< prsnt p_3 >	ма-к1о-х
< past s_1 indef >	сы-к1о-гъэ

< past s_2 indef >	у-к1о-гъэ
< past s_3 indef >	к1о-гъэ
< past p_1 indef >	ты-к1о-гъэ
< past p_2 indef >	шъу-к1о-гъэ
< past p_3 indef >	к1о-гъэ-х
< past s_1 perf >	сы-к1о-гъа-гъэ
< past s_2 perf >	у-к1о-гъа-гъэ
< past s_3 perf >	к1о-гъа-гъэ
< past p_1 perf >	ты-к1о-гъа-гъэ
< past p_2 perf >	шъу-к1о-гъа-гъэ
< past p_3 perf >	к1о-гъа-гъэ-х
< past s_1 cont >	сы-к1о-щты-гъэ
< past s_2 cont >	у-к1о-щты-гъэ
< past s_3 cont >	к1о-щты-гъэ
< past p_1 cont >	ты-к1о-щты-гъэ
< past p_2 cont >	шъу-к1о-щты-гъэ
< past p_3 cont >	к1о-щты-гъэ-х
< past s_1 imperf >	сы-к1о-щты-гъа-гъэ
< past s_2 imperf >	у-к1о-щты-гъа-гъэ
< past s_3 imperf >	к1о-щты-гъа-гъэ
< past p_1 imperf >	ты-к1о-щты-гъа-гъэ
< past p_2 imperf >	шъу-к1о-щты-гъа-гъэ
< past p_3 imperf >	к1о-щты-гъа-гъэ-х
< fut s_1 1_fut >	сы-к1о-щт
< fut s_2 1_fut >	у-к1о-щт
< fut s_3 1_fut >	к1о-щт
< fut p_1 1_fut >	ты-к1о-щт
< fut p_2 1_fut >	шъу-к1о-щт
< fut p_3 1_fut >	к1о-щт-ы-х
< fut s_1 2_fut >	сы-к1о-н
< fut s_2 2_fut >	у-к1о-н
< fut s_3 2_fut >	к1о-н
< fut p_1 2_fut >	ты-к1о-н
< fut p_2 2_fut >	шъу-к1о-н
< fut p_3 2_fut >	к1о-н-ы-х
< not prsnt s_1 >	сы-к1о-р-эп
< not prsnt s_2 >	у-к1о-р-эп
< not prsnt s_3 >	к1о-р-эп
< not prsnt p_1 >	ты-к1о-р-эп
< not prsnt p_2 >	шъу-к1о-р-эп
< not prsnt p_3 >	к1о-х-э-р-эп
< not past s_1 indef >	сы-к1о-гъэ-эп
< not past s_2 indef >	у-к1о-гъэ-эп
< not past s_3 indef >	к1о-гъэ-эп
< not past p_1 indef >	ты-к1о-гъэ-эп
< not past p_2 indef >	шъу-к1о-гъэ-эп
< not past p_3 indef >	к1о-гъэ-х-эп
< not past s_1 perf >	сы-к1о-гъа-гъэ-эп
< not past s_2 perf >	у-к1о-гъа-гъэ-эп
< not past s_3 perf >	к1о-гъа-гъэ-эп
< not past p_1 perf >	ты-к1о-гъа-гъэ-эп
< not past p_2 perf >	шъу-к1о-гъа-гъэ-эп
< not past p_3 perf >	к1о-гъа-гъэ-х-эп
< not past s_1 cont >	сы-к1о-щты-гъэ-эп
< not past s_2 cont >	у-к1о-щты-гъэ-эп
< not past s_3 cont >	к1о-щты-гъэ-эп

< not past p_1 cont >	ты-к1о-щты-гъэ-эп
< not past p_2 cont >	шъу-к1о-щты-гъэ-эп
< not past p_3 cont >	к1о-щты-гъэ-х-эп
< not past s_1 imperf >	сы-к1о-щты-гъа-гъэ-эп
< not past s_2 imperf >	у-к1о-щты-гъа-гъэ-эп
< not past s_3 imperf >	к1о-щты-гъа-гъэ-эп
< not past p_1 imperf >	ты-к1о-щты-гъа-гъэ-эп
< not past p_2 imperf >	шъу-к1о-щты-гъа-гъэ-эп
< not past p_3 imperf >	к1о-щты-гъа-гъэ-х-эп
< not fut s_1 1_fut >	сы-к1о-щт-эп
< not fut s_2 1_fut >	у-к1о-щт-эп
< not fut s_3 1_fut >	к1о-щт-эп
< not fut p_1 1_fut >	ты-к1о-щт-эп
< not fut p_2 1_fut >	шъу-к1о-щт-эп
< not fut p_3 1_fut >	к1о-щт-ы-х-эп
< not fut s_1 2_fut >	сы-к1о-н-эп
< not fut s_2 2_fut >	у-к1о-н-эп
< not fut s_3 2_fut >	к1о-н-эп
< not fut p_1 2_fut >	ты-к1о-н-эп
< not fut p_2 2_fut >	шъу-к1о-н-эп
< not fut p_3 2_fut >	к1о-н-ы-х-эп
< md prsnt s_1 infin_1 >	сэ-к1о-у
< md prsnt s_2 infin_1 >	о-к1о-у
< md prsnt s_3 infin_1 >	ма-к1о-у
< md prsnt p_1 infin_1 >	тэ-к1о-у
< md prsnt p_2 infin_1 >	шъо-к1о-у
< md prsnt p_3 infin_1 >	ма-к1о-х-э-у
< md past s_1 indef infin_1 >	сы-к1о-гъэ-у
< md past s_2 indef infin_1 >	у-к1о-гъэ-у
< md past s_3 indef infin_1 >	к1о-гъэ-у
< md past p_1 indef infin_1 >	ты-к1о-гъэ-у
< md past p_2 indef infin_1 >	шъу-к1о-гъэ-у
< md past p_3 indef infin_1 >	к1о-гъэ-х-э-у
< md past s_1 perf infin_1 >	сы-к1о-гъа-гъэ-у
< md past s_2 perf infin_1 >	у-к1о-гъа-гъэ-у
< md past s_3 perf infin_1 >	к1о-гъа-гъэ-у
< md past p_1 perf infin_1 >	ты-к1о-гъа-гъэ-у
< md past p_2 perf infin_1 >	шъу-к1о-гъа-гъэ-у
< md past p_3 perf infin_1 >	к1о-гъа-гъэ-х-э-у
< md past s_1 cont infin_1 >	сы-к1о-щты-гъэ-у
< md past s_2 cont infin_1 >	у-к1о-щты-гъэ-у
< md past s_3 cont infin_1 >	к1о-щты-гъэ-у
< md past p_1 cont infin_1 >	ты-к1о-щты-гъэ-у
< md past p_2 cont infin_1 >	шъу-к1о-щты-гъэ-у
< md past p_3 cont infin_1 >	к1о-щты-гъэ-х-э-у
< md past s_1 imperf infin_1 >	сы-к1о-щты-гъа-гъэ-у
< md past s_2 imperf infin_1 >	у-к1о-щты-гъа-гъэ-у
< md past s_3 imperf infin_1 >	к1о-щты-гъа-гъэ-у
< md past p_1 imperf infin_1 >	ты-к1о-щты-гъа-гъэ-у
< md past p_2 imperf infin_1 >	шъу-к1о-щты-гъа-гъэ-у
< md past p_3 imperf infin_1 >	к1о-щты-гъа-гъэ-х-э-у
< md fut s_1 1_fut infin_1 >	сы-к1о-щт-э-у
< md fut s_2 1_fut infin_1 >	у-к1о-щт-э-у
< md fut s_3 1_fut infin_1 >	к1о-щт-э-у
< md fut p_1 1_fut infin_1 >	ты-к1о-щт-э-у
< md fut p_2 1_fut infin_1 >	шъу-к1о-щт-э-у

< md fut p_3 1_fut infin_1 > к1о-щт-ы-х-э-у
< md prsnt s_1 cond_1 > сэ-к1о-мэ
< md prsnt s_2 cond_1 > о-к1о-мэ
< md prsnt s_3 cond_1 > ма-к1о-мэ
< md prsnt p_1 cond_1 > тэ-к1о-мэ
< md prsnt p_2 cond_1 > шъо-к1о-мэ
< md prsnt p_3 cond_1 > ма-к1о-х-э-мэ
< md past s_1 indef cond_1 > сы-к1о-гъэ-мэ
< md past s_2 indef cond_1 > у-к1о-гъэ-мэ
< md past s_3 indef cond_1 > к1о-гъэ-мэ
< md past p_1 indef cond_1 > ты-к1о-гъэ-мэ
< md past p_2 indef cond_1 > шъу-к1о-гъэ-мэ
< md past p_3 indef cond_1 > к1о-гъэ-х-э-мэ
< md past s_1 perf cond_1 > сы-к1о-гъа-гъэ-мэ
< md past s_2 perf cond_1 > у-к1о-гъа-гъэ-мэ
< md past s_3 perf cond_1 > к1о-гъа-гъэ-мэ
< md past p_1 perf cond_1 > ты-к1о-гъа-гъэ-мэ
< md past p_2 perf cond_1 > шъу-к1о-гъа-гъэ-мэ
< md past p_3 perf cond_1 > к1о-гъа-гъэ-х-э-мэ
< md past s_1 cont cond_1 > сы-к1о-щты-гъэ-мэ
< md past s_2 cont cond_1 > у-к1о-щты-гъэ-мэ
< md past s_3 cont cond_1 > к1о-щты-гъэ-мэ
< md past p_1 cont cond_1 > ты-к1о-щты-гъэ-мэ
< md past p_2 cont cond_1 > шъу-к1о-щты-гъэ-мэ
< md past p_3 cont cond_1 > к1о-щты-гъэ-х-э-мэ
< md past s_1 imperf cond_1 > сы-к1о-щты-гъа-гъэ-мэ
< md past s_2 imperf cond_1 > у-к1о-щты-гъа-гъэ-мэ
< md past s_3 imperf cond_1 > к1о-щты-гъа-гъэ-мэ
< md past p_1 imperf cond_1 > ты-к1о-щты-гъа-гъэ-мэ
< md past p_2 imperf cond_1 > шъу-к1о-щты-гъа-гъэ-мэ
< md past p_3 imperf cond_1 > к1о-щты-гъа-гъэ-х-э-мэ
< md fut s_1 1_fut cond_1 > сы-к1о-щт-мэ
< md fut s_2 1_fut cond_1 > у-к1о-щт-мэ
< md fut s_3 1_fut cond_1 > к1о-щт-мэ
< md fut p_1 1_fut cond_1 > ты-к1о-щт-мэ
< md fut p_2 1_fut cond_1 > шъу-к1о-щт-мэ
< md fut p_3 1_fut cond_1 > к1о-щт-ы-х-э-мэ
< notpr md prsnt s_1 infin_1 > сы-мы-к1о-у
< notpr md prsnt s_2 infin_1 > у-мы-к1о-у
< notpr md prsnt s_3 infin_1 > мы-к1о-у
< notpr md prsnt p_1 infin_1 > ты-мы-к1о-у
< notpr md prsnt p_2 infin_1 > шъу-мы-к1о-у
< notpr md prsnt p_3 infin_1 > мы-к1о-х-э-у
< notpr md past s_1 indef infin_1 > сы-мы-к1о-гъэ-у
< notpr md past s_2 indef infin_1 > у-мы-к1о-гъэ-у
< notpr md past s_3 indef infin_1 > мы-к1о-гъэ-у
< notpr md past p_1 indef infin_1 > ты-мы-к1о-гъэ-у
< notpr md past p_2 indef infin_1 > шъу-мы-к1о-гъэ-у
< notpr md past p_3 indef infin_1 > мы-к1о-гъэ-х-э-у
< notpr md past s_1 perf infin_1 > сы-мы-к1о-гъа-гъэ-у
< notpr md past s_2 perf infin_1 > у-мы-к1о-гъа-гъэ-у
< notpr md past s_3 perf infin_1 > мы-к1о-гъа-гъэ-у
< notpr md past p_1 perf infin_1 > ты-мы-к1о-гъа-гъэ-у
< notpr md past p_2 perf infin_1 > шъу-мы-к1о-гъа-гъэ-у
< notpr md past p_3 perf infin_1 > мы-к1о-гъа-гъэ-х-э-у
< notpr md past s_1 cont infin_1 > сы-мы-к1о-щты-гъэ-у

< notpr md past s_2 cont infin_1 > у-мы-к1о-щты-гъэ-у
< notpr md past s_3 cont infin_1 > мы-к1о-щты-гъэ-у
< notpr md past p_1 cont infin_1 > ты-мы-к1о-щты-гъэ-у
< notpr md past p_2 cont infin_1 > шъу-мы-к1о-щты-гъэ-у
< notpr md past p_3 cont infin_1 > мы-к1о-щты-гъэ-х-э-у
< notpr md past s_1 imperf infin_1 > сы-мы-к1о-щты-гъа-гъэ-у
< notpr md past s_2 imperf infin_1 > у-мы-к1о-щты-гъа-гъэ-у
< notpr md past s_3 imperf infin_1 > мы-к1о-щты-гъа-гъэ-у
< notpr md past p_1 imperf infin_1 > ты-мы-к1о-щты-гъа-гъэ-у
< notpr md past p_2 imperf infin_1 > шъу-мы-к1о-щты-гъа-гъэ-у
< notpr md past p_3 imperf infin_1 > мы-к1о-щты-гъа-гъэ-х-э-у
< notpr md fut s_1 1_fut infin_1 > сы-мы-к1о-щт-э-у
< notpr md fut s_2 1_fut infin_1 > у-мы-к1о-щт-э-у
< notpr md fut s_3 1_fut infin_1 > мы-к1о-щт-э-у
< notpr md fut p_1 1_fut infin_1 > ты-мы-к1о-щт-э-у
< notpr md fut p_2 1_fut infin_1 > шъу-мы-к1о-щт-э-у
< notpr md fut p_3 1_fut infin_1 > мы-к1о-щт-ы-х-э-у
< notpr md prsnt s_1 cond_1 > сы-мы-к1о-мэ
< notpr md prsnt s_2 cond_1 > у-мы-к1о-мэ
< notpr md prsnt s_3 cond_1 > мы-к1о-мэ
< notpr md prsnt p_1 cond_1 > ты-мы-к1о-мэ
< notpr md prsnt p_2 cond_1 > шъу-мы-к1о-мэ
< notpr md prsnt p_3 cond_1 > мы-к1о-х-э-мэ
< notpr md past s_1 indef cond_1 > сы-мы-к1о-гъэ-мэ
< notpr md past s_2 indef cond_1 > у-мы-к1о-гъэ-мэ
< notpr md past s_3 indef cond_1 > мы-к1о-гъэ-мэ
< notpr md past p_1 indef cond_1 > ты-мы-к1о-гъэ-мэ
< notpr md past p_2 indef cond_1 > шъу-мы-к1о-гъэ-мэ
< notpr md past p_3 indef cond_1 > мы-к1о-гъэ-х-э-мэ
< notpr md past s_1 perf cond_1 > сы-мы-к1о-гъа-гъэ-мэ
< notpr md past s_2 perf cond_1 > у-мы-к1о-гъа-гъэ-мэ
< notpr md past s_3 perf cond_1 > мы-к1о-гъа-гъэ-мэ
< notpr md past p_1 perf cond_1 > ты-мы-к1о-гъа-гъэ-мэ
< notpr md past p_2 perf cond_1 > шъу-мы-к1о-гъа-гъэ-мэ
< notpr md past p_3 perf cond_1 > мы-к1о-гъа-гъэ-х-э-мэ
< notpr md past s_1 cont cond_1 > сы-мы-к1о-щты-гъэ-мэ
< notpr md past s_2 cont cond_1 > у-мы-к1о-щты-гъэ-мэ
< notpr md past s_3 cont cond_1 > мы-к1о-щты-гъэ-мэ
< notpr md past p_1 cont cond_1 > ты-мы-к1о-щты-гъэ-мэ
< notpr md past p_2 cont cond_1 > шъу-мы-к1о-щты-гъэ-мэ
< notpr md past p_3 cont cond_1 > мы-к1о-щты-гъэ-х-э-мэ
< notpr md past s_1 imperf cond_1 > сы-мы-к1о-щты-гъа-гъэ-мэ
< notpr md past s_2 imperf cond_1 > у-мы-к1о-щты-гъа-гъэ-мэ
< notpr md past s_3 imperf cond_1 > мы-к1о-щты-гъа-гъэ-мэ
< notpr md past p_1 imperf cond_1 > ты-мы-к1о-щты-гъа-гъэ-мэ
< notpr md past p_2 imperf cond_1 > шъу-мы-к1о-щты-гъа-гъэ-мэ
< notpr md past p_3 imperf cond_1 > мы-к1о-щты-гъа-гъэ-х-э-мэ
< notpr md fut s_1 1_fut cond_1 > сы-мы-к1о-щт-мэ
< notpr md fut s_2 1_fut cond_1 > у-мы-к1о-щт-мэ
< notpr md fut s_3 1_fut cond_1 > мы-к1о-щт-мэ
< notpr md fut p_1 1_fut cond_1 > ты-мы-к1о-щт-мэ
< notpr md fut p_2 1_fut cond_1 > шъу-мы-к1о-щт-мэ
< notpr md fut p_3 1_fut cond_1 > мы-к1о-щт-ы-х-э-мэ
< notf md prsnt s_1 expct_1 > сы-к1о-р-эп-щтын
< notf md prsnt s_2 expct_1 > у-к1о-р-эп-щтын
< notf md prsnt s_3 expct_1 > к1о-р-эп-щтын

< notf md prsnt p_1 expct_1 > ты-к1о-р-эп-щтын
< notf md prsnt p_2 expct_1 > шъу-к1о-р-эп-щтын
< notf md prsnt p_3 expct_1 > к1о-х-э-р-эп-щтын
< notf md past s_1 indef expct_1 > сы-к1о-гъэ-эп-щтын
< notf md past s_2 indef expct_1 > у-к1о-гъэ-эп-щтын
< notf md past s_3 indef expct_1 > к1о-гъэ-эп-щтын
< notf md past p_1 indef expct_1 > ты-к1о-гъэ-эп-щтын
< notf md past p_2 indef expct_1 > шъу-к1о-гъэ-эп-щтын
< notf md past p_3 indef expct_1 > к1о-гъэ-х-эп-щтын
< notpr md past s_1 perf expct_1 > сы-к1о-гъа-гъэ-эп-щтын
< notpr md past s_2 perf expct_1 > у-к1о-гъа-гъэ-эп-щтын
< notpr md past s_3 perf expct_1 > к1о-гъа-гъэ-эп-щтын
< notpr md past p_1 perf expct_1 > ты-к1о-гъа-гъэ-эп-щтын
< notpr md past p_2 perf expct_1 > шъу-к1о-гъа-гъэ-эп-щтын
< notpr md past p_3 perf expct_1 > к1о-гъа-гъэ-х-эп-щтын
< notpr md past s_1 cont expct_1 > сы-к1о-щты-гъэ-эп-щтын
< notpr md past s_2 cont expct_1 > у-к1о-щты-гъэ-эп-щтын
< notpr md past s_3 cont expct_1 > к1о-щты-гъэ-эп-щтын
< notpr md past p_1 cont expct_1 > ты-к1о-щты-гъэ-эп-щтын
< notpr md past p_2 cont expct_1 > шъу-к1о-щты-гъэ-эп-щтын
< notpr md past p_3 cont expct_1 > к1о-щты-гъэ-х-эп-щтын
< notpr md past s_1 imperf expct_1 > сы-к1о-щты-гъа-гъэ-эп-щтын
< notpr md past s_2 imperf expct_1 > у-к1о-щты-гъа-гъэ-эп-щтын
< notpr md past s_3 imperf expct_1 > к1о-щты-гъа-гъэ-эп-щтын
< notpr md past p_1 imperf expct_1 > ты-к1о-щты-гъа-гъэ-эп-щтын
< notpr md past p_2 imperf expct_1 > шъу-к1о-щты-гъа-гъэ-эп-щтын
< notpr md past p_3 imperf expct_1 > к1о-щты-гъа-гъэ-х-эп-щтын
< notpr md fut s_1 1_fut expct_1 > сы-к1о-щт-эп-щтын
< notpr md fut s_2 1_fut expct_1 > у-к1о-щт-эп-щтын
< notpr md fut s_3 1_fut expct_1 > к1о-щт-эп-щтын
< notpr md fut p_1 1_fut expct_1 > ты-к1о-щт-эп-щтын
< notpr md fut p_2 1_fut expct_1 > шъу-к1о-щт-эп-щтын
< notpr md fut p_3 1_fut expct_1 > к1о-щт-ы-х-эп-щтын
< prticl prsnt s_1 confrm1 > сэ-к1о-ба
< prticl prsnt s_2 confrm1 > о-к1о-ба
< prticl prsnt s_3 confrm1 > ма-к1о-ба
< prticl prsnt p_1 confrm1 > тэ-к1о-ба
< prticl prsnt p_2 confrm1 > шъо-к1о-ба
< prticl prsnt p_3 confrm1 > ма-к1о-х-ба
< prticl past s_1 indef confrm1 > сы-к1о-гъэ-ба
< prticl past s_2 indef confrm1 > у-к1о-гъэ-ба
< prticl past s_3 indef confrm1 > к1о-гъэ-ба
< prticl past p_1 indef confrm1 > ты-к1о-гъэ-ба
< prticl past p_2 indef confrm1 > шъу-к1о-гъэ-ба
< prticl past p_3 indef confrm1 > к1о-гъэ-х-э-ба
< prticl past s_1 perf confrm1 > сы-к1о-гъа-гъэ-ба
< prticl past s_2 perf confrm1 > у-к1о-гъа-гъэ-ба
< prticl past s_3 perf confrm1 > к1о-гъа-гъэ-ба
< prticl past p_1 perf confrm1 > ты-к1о-гъа-гъэ-ба
< prticl past p_2 perf confrm1 > шъу-к1о-гъа-гъэ-ба
< prticl past p_3 perf confrm1 > к1о-гъа-гъэ-х-э-ба
< prticl past s_1 cont confrm1 > сы-к1о-щты-гъэ-ба
< prticl past s_2 cont confrm1 > у-к1о-щты-гъэ-ба
< prticl past s_3 cont confrm1 > к1о-щты-гъэ-ба
< prticl past p_1 cont confrm1 > ты-к1о-щты-гъэ-ба
< prticl past p_2 cont confrm1 > шъу-к1о-щты-гъэ-ба

< prticl past p_3 cont confrm1 > к1о-щты-гъэ-х-э-ба
< prticl fut s_1 1_fut confrm1 > сы-к1о-щт-ба
< prticl fut s_2 1_fut confrm1 > у-к1о-щт-ба
< prticl fut s_3 1_fut confrm1 > к1о-щт-ба
< prticl fut p_1 1_fut confrm1 > ты-к1о-щт-ба
< prticl fut p_2 1_fut confrm1 > шъу-к1о-щт-ба
< prticl fut p_3 1_fut confrm1 > к1о-щт-х-э-ба
< prticl fut s_1 2_fut confrm1 > сы-к1о-н-ба
< prticl fut s_2 2_fut confrm1 > у-к1о-н-ба
< prticl fut s_3 2_fut confrm1 > к1о-н-ба
< prticl fut p_1 2_fut confrm1 > ты-к1о-н-ба
< prticl fut p_2 2_fut confrm1 > шъу-к1о-н-ба
< prticl fut p_3 2_fut confrm1 > к1о-н-х-э-ба
< prtic prsnt s_1 rept > сэ-к1о-жьы
< prtic prsnt s_2 rept > о-к1о-жьы
< prtic prsnt s_3 rept > ма-к1о-жьы
< prtic prsnt p_1 rept > тэ-к1о-жьы
< prtic prsnt p_2 rept > шъо-к1о-жьы
< prtic prsnt p_3 rept > ма-к1о-жьы-х
< prtic past s_1 indef rept > сы-к1о-жьы-гъэ
< prtic past s_2 indef rept > у-к1о-жьы-гъэ
< prtic past s_3 indef rept > к1о-жьы-гъэ
< prtic past p_1 indef rept > ты-к1о-жьы-гъэ
< prtic past p_2 indef rept > шъу-к1о-жьы-гъэ
< prtic past p_3 indef rept > к1о-жьы-гъэ-х
< prtic past s_1 perf rept > сы-к1о-жьы-гъа-гъэ
< prtic past s_2 perf rept > у-к1о-жьы-гъа-гъэ
< prtic past s_3 perf rept > к1о-жьы-гъа-гъэ
< prtic past p_1 perf rept > ты-к1о-жьы-гъа-гъэ
< prtic past p_2 perf rept > шъу-к1о-жьы-гъа-гъэ
< prtic past p_3 perf rept > к1о-жьы-гъа-гъэ-х
< prtic past s_1 cont rept > сы-к1о-жьы-щты-гъэ
< prtic past s_2 cont rept > у-к1о-жьы-щты-гъэ
< prtic past s_3 cont rept > к1о-жьы-щты-гъэ
< prtic past p_1 cont rept > ты-к1о-жьы-щты-гъэ
< prtic past p_2 cont rept > шъу-к1о-жьы-щты-гъэ
< prtic past p_3 cont rept > к1о-жьы-щты-гъэ-х
< prtic past s_1 imperf rept > сы-к1о-жьы-щты-гъа-гъэ
< prtic past s_2 imperf rept > у-к1о-жьы-щты-гъа-гъэ
< prtic past s_3 imperf rept > к1о-жьы-щты-гъа-гъэ
< prtic past p_1 imperf rept > ты-к1о-жьы-щты-гъа-гъэ
< prtic past p_2 imperf rept > шъу-к1о-жьы-щты-гъа-гъэ
< prtic past p_3 imperf rept > к1о-жьы-щты-гъа-гъэ-х
< prtic fut s_1 1_fut rept > сы-к1о-жьы-щт
< prtic fut s_2 1_fut rept > у-к1о-жьы-щт
< prtic fut s_3 1_fut rept > к1о-жьы-щт
< prtic fut p_1 1_fut rept > ты-к1о-жьы-щт
< prtic fut p_2 1_fut rept > шъу-к1о-жьы-щт
< prtic fut p_3 1_fut rept > к1о-жьы-щт-ы-х
< prtic fut s_1 2_fut rept > сы-к1о-жьы-н
< prtic fut s_2 2_fut rept > у-к1о-жьы-н
< prtic fut s_3 2_fut rept > к1о-жьы-н
< prtic fut p_1 2_fut rept > ты-к1о-жьы-н
< prtic fut p_2 2_fut rept > шъу-к1о-жьы-н
< prtic fut p_3 2_fut rept > к1о-жьы-н-ы-х
< notf prtic prsnt s_1 rept > сы-к1о-жьы-р-эп

< notf prtic prsnt s_2 rept > у-к1о-жьы-р-эп
< notf prtic prsnt s_3 rept > к1о-жьы-р-эп
< notf prtic prsnt p_1 rept > ты-к1о-жьы-р-эп
< notf prtic prsnt p_2 rept > шъу-к1о-жьы-р-эп
< notf prtic prsnt p_3 rept > к1о-жьы-х-э-р-эп
< notf prtic past s_1 indef rept > сы-к1о-жьы-гъэ-эп
< notf prtic past s_2 indef rept > у-к1о-жьы-гъэ-эп
< notf prtic past s_3 indef rept > к1о-жьы-гъэ-эп
< notf prtic past p_1 indef rept > ты-к1о-жьы-гъэ-эп
< notf prtic past p_2 indef rept > шъу-к1о-жьы-гъэ-эп
< notf prtic past p_3 indef rept > к1о-жьы-гъэ-х-эп
< notf prtic past s_1 perf rept > сы-к1о-жьы-гъа-гъэ-эп
< notf prtic past s_2 perf rept > у-к1о-жьы-гъа-гъэ-эп
< notf prtic past s_3 perf rept > к1о-жьы-гъа-гъэ-эп
< notf prtic past p_1 perf rept > ты-к1о-жьы-гъа-гъэ-эп
< notf prtic past p_2 perf rept > шъу-к1о-жьы-гъа-гъэ-эп
< notf prtic past p_3 perf rept > к1о-жьы-гъа-гъэ-х-эп
< notf prtic past s_1 cont rept > сы-к1о-жьы-щты-гъэ-эп
< notf prtic past s_2 cont rept > у-к1о-жьы-щты-гъэ-эп
< notf prtic past s_3 cont rept > к1о-жьы-щты-гъэ-эп
< notf prtic past p_1 cont rept > ты-к1о-жьы-щты-гъэ-эп
< notf prtic past p_2 cont rept > шъу-к1о-жьы-щты-гъэ-эп
< notf prtic past p_3 cont rept > к1о-жьы-щты-гъэ-х-эп
< notf prtic past s_1 imperf rept > сы-к1о-жьы-щты-гъа-гъэ-эп
< notf prtic past s_2 imperf rept > у-к1о-жьы-щты-гъа-гъэ-эп
< notf prtic past s_3 imperf rept > к1о-жьы-щты-гъа-гъэ-эп
< notf prtic past p_1 imperf rept > ты-к1о-жьы-щты-гъа-гъэ-эп
< notf prtic past p_2 imperf rept > шъу-к1о-жьы-щты-гъа-гъэ-эп
< notf prtic past p_3 imperf rept > к1о-жьы-щты-гъа-гъэ-х-эп
< notf prtic fut s_1 1_fut rept > сы-к1о-жьы-щт-эп
< notf prtic fut s_2 1_fut rept > у-к1о-жьы-щт-эп
< notf prtic fut s_3 1_fut rept > к1о-жьы-щт-эп
< notf prtic fut p_1 1_fut rept > ты-к1о-жьы-щт-эп
< notf prtic fut p_2 1_fut rept > шъу-к1о-жьы-щт-эп
< notf prtic fut p_3 1_fut rept > к1о-жьы-щт-ы-х-эп
< notf prtic fut s_1 2_fut rept > сы-к1о-жьы-н-эп
< notf prtic fut s_2 2_fut rept > у-к1о-жьы-н-эп
< notf prtic fut s_3 2_fut rept > к1о-жьы-н-эп
< notf prtic fut p_1 2_fut rept > ты-к1о-жьы-н-эп
< notf prtic fut p_2 2_fut rept > шъу-к1о-жьы-н-эп
< notf prtic fut p_3 2_fut rept > к1о-жьы-н-ы-х-эп

Computer modeling system of morphemic-linguistic structure of adyghe language

V.Sh. Tlusten

In this article the question of Adyghe (Circassion) languages morphemic structure formal modeling by DATR computer-linguistic system is considered. By the Adyghe language verbs inflectional paradigms example the methodology of a such constructions is shown. The DATR-theory that can be used as a basis of corresponding formalization for some representative class of Adyghe verbs is resulted.

