ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М. В. ЛОМОНОСОВА»

ФИЛОЛОГИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ЛИНГВИСТИКИ
НАПРАВЛЕНИЕ ПОДГОТОВКИ
«ФУНДАМЕНТАЛЬНАЯ И ПРИКЛАДНАЯ ЛИНГВИСТИКА»

КАРПЕНКО КИРИЛЛ МИХАЙЛОВИЧ

МОРФОЛОГИЧЕСКИЙ ПАРСЕР ДЛЯ ЯЗЫКА ТИГРЕ

Выпускная квалификационная работа бакалавра

Научный руководитель – д. ф. н. П. В. Гращенков

г. Москва

2019 г.

Содержание

Введение

Данная работа посвящена разработке системы автоматического анализа морфологии для языка тигре́. Тигре является представителем эфиосемитской группы языков, третьим по числу носителей в этой группе после амхарского и тигриньи и пятым (после арабского, иврита и упомянутых двух) среди семитских языков (Voigt 2015). Число носителей в 2010 году оценивалось в 1,4 миллиона человек и возрастало; носители проживали преимущественно в Эритрее и юго-восточных областях Судана (Ethnologue.com).

Объем письменных текстов на языке тигре существенно вырос в последние десятилетия, в особенности с момента обретения Государством Эритрея независимости в 1991 году (обзор литературы содержится, например, в недавней публикации (Voigt 2015)). В настоящее время письменные тексты представлены учебной и художественной литературой, массовыми изданиями, наиболее заметным среди которых является газета Eritrea Haddas (тигре ኦረትርየ ሐዳስ, в переводе на русский – "Новая Эритрея"), размещаемая в том числе на сайте Министерства информации Государства Эритрея (Shabait); кроме того, нами был обнаружен блог на языке тигре (Shaertigre).

Таким образом, различного рода задачи автоматической обработки текстов на языке тигре становятся актуальными. Тем не менее, нам неизвестны какие-либо работы, в которых затрагивались бы вопросы прикладной лингвистики вообще и автоматического анализа морфологии в частности в применении к тигре. Учитывая, что системы такого рода существуют не только для крупнейших семитских языков, – арабского и иврита, – но и для амхарского и тигриньи (см. обзор в 1.1), а также принимая во внимание тот факт, что число носителей и объем письменных текстов на языке растут, а сам язык имеет в Эритрее региональный статус (Ethnologue.com), предвидится интерес исследователей в различных областях прикладной лингвистики к этому языку.

В настоящей работе предпринята первая – по крайней мере, из известных нам – попытка восполнить эту лакуну и создать морфологический анализатор для языка тигре. Как отмечает автор HornMorpho, наиболее успешной на данный момент системы анализа морфологии для наиболее близкородственных языку тигре языков –

амхарского и тигриньи (Gasser 2011), в отношении языков с такой сложной морфологией, как семитские языки, продвижение в решении многих задач вычислительной лингвистики зависит от наличия средств для анализа морфологии. Мы надеемся, что предлагаемая система окажется полезной для будущих исследований языка тигре – как в решении прикладных задач, так и для помощи в теоретических лингвистических исследованиях.

1. Обзор литературы

1.1. Автоматический анализ морфологии: подходы, опыт применения к эфиосемитским языкам

Подходы к автоматическому анализу морфологии можно разделить на две большие группы: подходы, основанные на корпусах и подходы, основанные на правилах (Kazakov and Manandhar 2001), или, по другой терминологии, эмпирические подходы и подходы, основанные на знаниях, соответственно (Soudi et al. 2007).

1.1.1. Подходы, основанные на корпусах

Подходы, основанные на корпусах, предполагают извлечение правил сегментации с использованием алгоритмов, обученных на обучающем корпусе (Kazakov and Manandhar 2001), то есть машинного обучения. При этом корпус может быть как размеченным (для обучения с учителем; см., например, van den Bosch 1997, Wicentowski 2004), так и неразмеченным (для обучения без учителя; см., например, Goldsmith 2000, Hammarström and Borin 2011).

Основным преимуществом таких подходов является, в идеале, отсутствие необходимости написания вручную правил сегментации (см., впрочем, обсуждение работы Tesfaye 2002 ниже). Отдельно также следует отметить, что, поскольку методы, основанные на обучении без учителя, не требуют для своего применения размеченного корпуса, они являются весьма привлекательной альтернативой как подходам, основанным на правилах, так и методам, основанным на обучении с учителем; это обуславливает интерес к их применению в отношении малоресурсных языков, к которым принадлежат эфиосемитские (Tachbelie 2010, Wondwossen and Gasser 2012).

Корпусные подходы, однако, имеют свои недостатки. Основным препятствием к их применению является необходимость наличия достаточно большого корпуса; это в особенности справедливо для обучения без учителя. Создание таких корпусов является очень затратным по времени и – в особенности для размеченных корпусов – трудоемким процессом. В результате исследователи, занимающиеся применением статистических методов при анализе морфологии малоресурсных языков, зачастую тратят основную часть времени на составление обучающего корпуса и его разметку

(Tachbelie 2010). Отсутствие достаточно большого по размеру корпуса негативно влияет на качество результата: в целом, качество сегментации, как правило, меньше, чем у систем, основанных на написанных вручную правилах.

Корпусные подходы в автоматическом анализе морфологии среди эфиосемитских языков наиболее часто применяются к амхарскому. С начала 2000-х годов в этой области было выполнено несколько работ, в числе которых находятся стеммер (Nega and Willett 2002), сегментатор (Sisay 2005) и стеммер (Alemu and Asker 2007); кроме того, имеется опыт ограниченного применения универсальных систем анализа морфологии, основанных на обучении без учителя – работа (Tesfaye 2002) с применением системы Linguistica (Goldsmith 2000) и диссертация (Tachbelie 2010), описывающая применение, среди прочего, системы Morfessor (Creutz and Lagus 2005).

Применение статистических методов к эфиосемитским языкам встречается с серьезными трудностями. Главная из них, общая для всех малоресурсных языков – малочисленность, а чаще отсутствие, подходящих корпусов и малый объем существующих корпусов.

В данном случае наилучшим образом дела обстоят с амхарским языком и, в меньшей степени, с тигриньей, в силу их роли в регионе Африканского Рога как государственных языков. Так, для амхарского имеется несколько корпусов; крупнейшие неразмеченные включают корпус Walta Information Center – WIC (описан в Argaw and Asker 2007), содержащий 1,7 миллиона слов (источник – новостные статьи), из которых около 207 тысяч транскрибировано, и корпус An Crúbadán (Scannell 2007) в 17 миллионов слов (источник – Википедия на амхарском, Всеобщая декларация прав человека и сайт Свидетелей Иеговы); кроме того, в обзорной статье (Gambäck and Asker 2010) есть упоминание о корпусе в 3,5 миллиона слов (источник – новостные статьи). Имеются также как минимум три корпуса с морфологической и частеречной разметкой: подкорпус WIC в 210 тысяч слов (Demeke and Getachew 2006), размеченный вручную; корпус Amharic Web as a Corpus – amWaC (Rychlý and Suchomel 2016), содержащий 30,5 миллиона слов с частеречной разметкой, для которой был использован TreeTagger, обученный на размеченном подкорпусе WIC, и корпус Contemporary Amharic Cropus – CACO (Gezmu et al. 2018), содержащий около 24 миллионов слов (сферы – новости, публицистика, Библия, художественная литература и др.), размеченных с помощью анализатора HornMorpho (Gasser 2011), подробнее описанного ниже.

Для тигриньи первые корпуса появились лишь недавно и более скромны по объему; на настоящий момент имеются два корпуса – Nagaoka Tigrinya Corpus (Tedla et al. 2016), содержащий 72 тысячи слов (сфера – новости, публицистика, право и др.), с частеречной разметкой, выполненной вручную, и корпус tiWaC (Rychlý and Suchomel 2016) в 2,5 миллиона слов, также с частеречной разметкой.

Нам не удалось найти каких-либо корпусов языка тигре – как размеченных, так и неразмеченных. Представляется, что сбор и разметка данных для потенциального обучающего корпуса, объем и качество которого позволили бы добиться достаточно высоких результатов, потребовали бы, в частности, привлечения труда носителей тигре и лингвистов, специализирующихся на его изучении, и заняли бы огромное время. Придя к заключению, что написание парсера вместе с созданием корпуса по объему работ вышло бы за рамки возможного для выпускной квалификационной работы бакалавра, мы решили создать систему, основанную на правилах, написанных вручную.

Следует упомянуть и другой фактор, осложняющий применение некоторых подходов в машинном обучении (а также и некоторых из подходов, основанных на правилах, о чем будет сказано ниже) для эфиосемитских языков – неконкатенативная морфология основы, отличающая семитские языки. Явления неконкатенативной, или нелинейной (Simpson 2009), морфологии подразумевают отклонение от конкатенативной модели морфологии, в которой морфемы непрерывно следуют друг за другом как линейные единицы (Haspelmath and Sims 2010). В случае с моделями, основанными на обучении без учителя, о трудностях, связанных с неконкатенативными явлениями, сообщается, например, в вышеупомянутой работе (Tesfaye 2002); необходимость создать стеммер, ставший основной частью этой работы, была продиктована невозможностью полного анализа амхарских словоформ с помощью программы Linguistica, которая, как сообщали авторы, при анализе способна была отделить только аффиксы, присоединяемые линейно, но не обрабатывала явления шаблонной морфологии внутри основы.

1.1.2. Подходы, основанные на правилах

Подходы, основанные на правилах, предполагают предварительное написание правил сегментации вручную на основе лингвистических знаний. Опора на знания обеспечивает основное преимущество таких подходов – высокое качество сегментации (Karttunen 1994). Для его обеспечения, однако, требуется большое количество правил, что делает создание систем, использующих эти подходы, весьма трудоемким процессом. Кроме того, созданную для одного языка систему бывает сложно применить к другим языкам, хотя бы и близкородственным (Gasser 2011).

Концептуально простейшей моделью морфологии в рамках подходов, основанных на правилах, является лексикон полных форм (англ. full form lexicon), в котором перечислены все возможные словоформы языка в паре с их морфологическими описаниями (Trost 2003). Очевидными недостатками лексиконов полных форм являются избыточность данных, а также невозможность их применения для анализа словоформ, не содержащихся в них. Проблему избыточности по большей части решает другая модель – лексикон лемм (англ. lemma lexicon); в таких лексиконах вместо полных словоформ хранятся леммы – канонические формы слов, и для решения задачи анализа применяется алгоритм, сопоставляющий словоформе лемму и морфосинтаксический анализ; алгоритм, таким образом, описывает морфотактику языка. Для обработки феноменов нерегулярного словоизменения, таких, как явления супплетивизма и различные морфонологические явления, применяются отдельные расширения лексикона и алгоритма; степень успешности такой обработки, однако, зависит от конкретного языка. Помимо этого, в целом, лексиконы лемм призваны решать задачу морфосинтаксического анализа, и поэтому успешная обработка также и словообразовательной морфологии с их использованием труднодостижима (Trost 2003).

Наиболее популярной моделью морфологии в подходах, основанных на правилах, является двухуровневая морфология (Koskenniemi 1983). В этой модели применяется два уровня представления – лексический и поверхностный; взаимодействие между ними обеспечивается набором правил, реализованных посредством конечных автоматов-преобразователей, или трансдьюсеров (англ. finite state transducers).

Классическая изначальная модель двухуровневой морфологии основана на предположении, что словоформы на поверхностном уровне образуются посредством

простой конкатенации морфем. Этим обусловлен тот факт, что языки с неконкатенативной морфологией представляют для автоматного анализа в рамках этой модели известную сложность (Tachbelie 2010). Анализу вопросов применения автоматного подхода к таким языкам и попытке его расширения для обработки отдельных явлений неконкатенативной морфологии, в частности, трансфиксации в арабском языке посвящена, например, работа (Городенцева 2010).

Наиболее успешным опытом создания систем, основанных на правилах, предназначенных для обработки морфологии эфиосемитских языков, является анализатор HornMorpho (Gasser 2011 – описана первая версия). Текущая версия HornMorpho позволяет обрабатывать морфологию амхарского и тигриньи, а также оромо (кушитского языка, распространенного в Эфиопии), и осуществляет для каждого из указанных языков сегментацию словоформ и морфологический анализ. Автоматный подход в HornMorpho модифицирован путем применения метода, описанного в (Amtrup 2003); в этом методе в качестве весов на переходы между состояниями автомата используются признаковые структуры (англ. feature structures), имеющие вид “атрибут—значение”. В статье Wondwossen and Gasser 2012 создатель HornMorpho отмечает, помимо трудоемкости процесса написания правил для каждого отдельного языка, что ему, несмотря на генетическую и ареальную близость амхарского и тигриньи, не удалось в сколько-нибудь значительной степени обобщить правила для подсистем, отвечающих за эти два языка, и в результате эти подсистемы оказались совершенно изолированы друг от друга. Тем не менее, сообщается о высоком качестве работы HornMorpho, в особенности на амхарских глаголах (аккуратность 99% на тестовом корпусе из 200 глаголов, случайным образом выбранных из списка в 397 тысяч уникальных словоформ различных частей речи); менее высокая аккуратность в 95,5% для амхарских существительных и прилагательных (на тестовом корпусе из 200 существительных и прилагательных, выбранных из того же списка) и в 94% для глаголов тигриньи (на тестовом корпусе из 200 глаголов, выбранных из списка 228 тысяч уникальных словоформ тигриньи) создатель системы объясняет отсутствием соответствующих корней в лексиконе (что в особенности справедливо для тигриньи – языка с меньшим количеством доступных лингвистических ресурсов, чем амхарский) и ошибками в имплементации отдельных фонологических правил.

Успешность опыта Гассера с HornMorpho в большой степени обусловила наше решение создать подобную систему, основанную на правилах, для языка тигре.

1.2. Лингвистические ресурсы для языка тигре

Создание системы анализа морфологии, основанной на правилах, требует достаточного количества литературы, адекватно описывающего морфологические процессы и закономерности, действующие в данном языке. В сравнении с более крупными (в отношении числа носителей) эфиосемитскими языками, то есть с амхарским и тигриньей, доступной лингвистической литературы для языка тигре значительно меньше. Среди работ, описывающих морфологию тигре, стоит упомянуть нижеследующие.

Наиболее полное описание морфологии, доступное на данный момент, содержится в грамматике (Raz 1983). Помимо закономерностей морфологии, морфонологии и фонологии, изложению которых автор посвятил большую часть работы, в ней также представлено описание синтаксиса и несколько глоссированных естественных текстов на языке.

Помимо этого, различная информация о морфологии тигре представлена в нескольких других трудах. Присутствует более ранняя грамматика тигре несколько меньшего объема (Leslau 1945), а также объемное описание именной морфологии (Palmer 1962). Обзорное описание различных явлений морфологии тигре представлено в статье (Voigt 2008), описание глагольной морфологии – в (Raz 1980). Отдельные явления морфологии и морфонологии рассматриваются в нескольких статьях: статья (Voigt 2009) посвящена анализу форм имперфекта, статья (Faust 2014) – анализу метатез в глагольных формах, в статье (Bulakh 2015) рассматривается морфокомбинаторика и семантика деривационного глагольного префикса *at-*. Описанию диалекта гинда (тигре: ጊንዓዕ) посвящена обширная работа (Elias 2014).

Помимо грамматик и статей, описывающих явления морфологии и морфонологии, безусловную важность для обработки текстов на языке представляют лексикографические ресурсы. Существует несколько известных нам словарей тигре; наиболее полным немонолингвальным словарем является словарь (Littmann and Höffner 1962), содержащий записи на языке тигре (в эфиопской графике), немецком и английском, а также этимологическую информацию с записями на геэзе, тигринье и амхарском; есть также список когнатов в тигре, геэзе, амхарском и тигринье (Leslau

1982) и тезаурус для диалекта бени-амер (тигре: በኒ ኣመር) с записями на тигре (бени-амер, в эфиопской графике) и английском (Nakano 1982).

Помимо упомянутых работ, в последние десятилетия стали появляться лингвистические работы, написанные на языке тигре. В статье (Voigt 2015) перечисляются несколько из них, в том числе грамматика и билингвальный словарь английский — тигре. К сожалению, получить доступ к этим ресурсам для нас не представилось возможным.

Таким образом, несмотря на то, что объем литературы о языке тигре ограничен, представляется, что доступных ресурсов вполне достаточно для того, чтобы предпринять попытку создания системы морфологического анализа, в особенности основанной на правилах.

2. Постановка задачи

В рамках настоящей работы нами была поставлена задача создать систему, которая для произвольного текста на языке тигре для всех словоформ в этом тексте создавала бы список возможных морфологических разборов, то есть морфологический парсер.

Следует сразу отметить некоторые ограничения, установленные нами для такой системы:

1. В данном исследовании рассматриваются только словоформы, понимаемыми в узком, орфографическом, смысле – как сегменты текста, разделенные пробелами и/или знаками пунктуации. Учёт зависимостей как от соседних орфографических слов, так и на большей дистанции не входит в задачу настоящего исследования.

2. В данном исследовании, кроме того, не создаётся лексикон корней. Мы будем считать достаточным создание системы, которая, в идеале, производила бы анализ словоформ вплоть до извлечения корня. Тем не менее, в задачу входит предусмотреть возможность включения в систему такого лексикона и его пополнения. Кроме того, помимо обеспечения обработки морфем, нужно будет создать лексикон служебных слов, в особенности таких, которые выражают бесспорно грамматические значения. Примером для сравнения для нас здесь может служить компонент описанного ранее (1.1.2) анализатора HornMorpho, ответственный за обработку словоформ тигриньи. Как отмечает его создатель, доступных лингвистических ресурсов для тигриньи значительно меньше, чем для амхарского, поэтому в основе этого компонента лежит то, что он назвал “отгадывателем” (англ. guesser), то есть система, выдающая возможные разборы исключительно на основании данных о морфологии языка. Следует еще раз подчеркнуть, что, хотя и, как было отмечено нами ранее, ресурсов по языку тигре мало в сравнении с другими эфиосемитскими языками, большими по численности носителей (то есть с амхарским и тигриньей), вообще говоря, описания лексики существуют, в частности, особенно полезным может являться словарь (Littmann and Höffner 1962). Тем не менее, переработка доступных лексикологических ресурсов в лексиконы для системы выходит за рамки данной работы.

Предполагается, что система будет обрабатывать, помимо словоизменительной, также и продуктивную словообразовательную морфологию. Среди наиболее сложных для анализа феноменов словоизменительной морфологии следует выделить множественные неконкатенативные явления (в первую очередь – так называемое "ломаное множественное" число, сохранившее продуктивность в языке тигре и потому требующее описания в качестве регулярного явления, а также, как и в случае с другими семитскими языками, вокалический шаблон глагольной основы). В числе продуктивных явлений словообразовательной морфологии наиболее часто встречающимися являются показатели залога и актантной деривации, обыкновенно представленные в глагольной форме префиксом (в частности, показателем пассива *t(ə)-*, показателями каузатива *a-* и *at-*). Актантная деривация, однако, иногда проявляется не только в добавлении префикса, но и в некоторых изменениях в спряжении (в частности, в изменении вокалического шаблона основы). Кроме того, продуктивно, например, образование различных так называемых отглагольных имен, в частности, имени деятеля, имени места; кроме того, инфинитив выступает в роли имени действия. Эти формы также образуются с помощью изменения вокалического паттерна основы.

В языке тигре используется две письменности – эфиопская и арабская графика: эфиопская – в Эритрее (как и для тигриньи), арабская – в районах проживания носителей тигре в Южном Судане. В этой работе рассматривается обработка только тех текстов на языке тигре, которые записаны с применением эфиопской письменности (как в силу ее наибольшей распространённости и большой актуальности ее обработки в силу наличия литературы и массовых изданий на языке тигре в Эритрее, так и ввиду уверенного знакомства автора только с эфиопской, но не с арабской графикой).

Относительно эфиопской графики следует отметить, что, хотя она в целом однозначно передает фонемный состав словоформ тигре, в отношении этого соответствия имеются две особенности.

1. Слог шестого порядка может передавать как сочетание согласного с [ə], так и просто согласный без гласного. Звук [ə], однако, является в языке тигре эпентетическим вокалическим элементом, и закономерности его вставки относятся исключительно к фонотактике. Это позволяет автору грамматики (Raz 1983), например, при обсуждении ее роли в фонологической системе тигре утверждать, что [ə] не следует включать в число фонем – этот звук не участвует в образовании

фонологически контрастных пар (Raz 1983:10—11). Наш анализ морфологических правил приводит нас к выводу, что [ə], в самом деле, никак не влияет на морфологические процессы; существуют контексты, в которых она этот звук заменяется на другой, имеющий представление в фонологической системе, однако само присутствие [ə] в этом контексте объясняется правилами фонотактики и для целей морфологического анализа может быть представлено как чередование нуля звука и соответствующей фонемы.

2. Другая, более важная для анализа морфологии особенность эфиопской графики заключается в том, что она не передает геминацию согласных. Геминация в языке тигре, как и в других эфиосемитских языках (в частности, в амхарском) является морфологически значимой. Существует множество пар словоформ, противопоставленных друг другу исключительно удвоением согласной. При использовании эфиопской графики составляющие этих пар, таким образом, становятся омографами, и возникает задача, с одной стороны, определить, удвоен ли тот или иной согласный в данном контексте, а с другой стороны – при наличии нескольких возможных вариантов в отношении геминации предоставлять их все. При этом следует иметь в виду, что определение геминации ограничивается в данной системе только теми случаями, которые можно рассмотреть на основании исключительно правил, относящихся к уровню морфологии; снятие неоднозначности в тех случаях, когда наличие омографов можно определить только при наличии лексикона, не входит в задачу, ставимую здесь нами – в отличие, например, от определения геминации в личных формах глаголов постольку, поскольку информацию о ней можно вывести из признаков, однозначно отраженных в орфографии, в частности, набора линейно присоединяемых аффиксов и вокалического шаблона основы.

Таким образом, задача заключается в создании системы, которая производила бы анализ морфологической структуры произвольной словоформы языка тигре, записанной в эфиопской графике, настолько полно, насколько это возможно без привлечения лексикологических знаний (предусмотрев, однако, возможность их включения в систему).

3. Описание алгоритма

3.1. Общее устройство

Предлагаемый анализатор представляет собой программное обеспечение, исходный код которого реализован на языке программирования Java, с прилагаемыми к нему служебными файлами форматов .txt и .json.

В качестве входных данных анализатор принимает файл, содержащий текст на языке тигре в эфиопской графике.

Результатом работы программы является выходной файл, в котором каждому орфографическому слову из входного файла сопоставлены его глоссированные разборы. Поскольку в настоящий момент с анализатором не используется лексикон корней, вывод сконфигурирован таким образом, чтобы в числе первых показывались разборы с наибольшим количеством выделенных морфем. Пользователь может задать максимальное количество разборов для каждого слова, которые будут выведены в выходной файл, либо не указывать значение этого параметра, в таком случае в выходном файле будут присутствовать все разборы.

Рисунок 1. Фрагмент выходного файла анализатора

```
Word: ወለልትቀረች
Analyses:

wa-la-l-t-qarrac
COORD-DEF-3.M.SG-PASS-qr(2)c:IMPF

wa-la-l-t-qarrac
COORD-DEF-3.M.SG-PASS-qr(2)c:JUSS

wa-la-l-t-qarrac
COORD-REL-3.M.SG-PASS-qr(2)c:IMPF

wa-la-l-t-qarrac
COORD-REL-3.M.SG-PASS-qr(2)c:JUSS
```

3.2. Препроцессинг

Поскольку для целей морфологического анализа каждую фонему необходимо рассматривать отдельно, а силлабические графемы эфиопской письменности являются в большинстве случаев обозначением не для одной, а для двух фонем, программой вначале производится предварительная обработка (препроцессинг) текста с целью перевода его в промежуточную форму, применимую для работы парсера.

Для целей препроцессинга был создан файл с описанием взаимно однозначных соответствий графем эфиопской письменности и знаков промежуточной формы, основанных на латинской графике с добавлением нескольких символов для передачи графем с базовыми вариантами ʰ и ʾ (образующие согласные – звонкий фарингальный фрикативный ʕ и гортанная смычка ʔ соответственно). Несмотря на то, что в самом парсере используются только представления фонем и, таким образом, не учитывается [ə], при препроцессинге графемы шестого порядка, передающие сочетание согласного с [ə] либо с нулем звука, ставятся в соответствие сочетанию "согласный + знак ‘ə’".

Следующий этап препроцессинга призван решить задачу определения позиций, на которых могут присутствовать геминированные согласные. В предлагаемом анализаторе на данном, раннем, этапе для каждого орфографического слова генерируются все варианты с возможными комбинациями значений признака "удвоенность" у согласных. Под возможными комбинациями здесь понимаются комбинации, ограниченные следующими условиями:

1. Начальная согласная фонема не может быть реализована с геминацией.

2. Фонемы, представляющие ларингальные согласные (базовые варианты ʰ */ha/*, ʱ */ħa/*, ˣ */xa/*, ʲ */ʔa/*, ʿ */ʕa/*) и аппроксиманты (ʷ /wa/, ʸ /ja/) не могут быть реализованы с удвоением ни в какой позиции.

3. Две одинаковые согласные, разделенные знаком ‘ə’ в промежуточной записи на данном этапе, представлены в исходной орфографической записи двумя графемами и, следовательно, не являются одной удвоенной согласной (примечательно, что, в самом деле, в этих позициях первая из графем, записанная в шестом порядке, передает и вокалический элемент [ə], реализуемый в них в соответствии с фонотактикой тигре).

После генерации вариантов с удвоенными согласными создается служебный объект, содержащий орфографическую запись слова, негеминированная промежуточная запись (без знака ‘ə’) и все возможные варианты с учетом удвоения согласных. Для каждого из вариантов анализ далее осуществляется отдельно.

3.3. Анализатор

3.3.1. Анализ “от краёв к центру”

Анализ начинается с первого компонента, в котором посредством применения каскада наборов регулярных выражений и соответствующих им замен производится разбор словоформы в направлении "от краёв к центру" и постепенное построение разбора, представленного объектом, в котором сегментированной записи фонемного состава слова сопоставлена также сегментированная запись, содержащая глоссы.

В данном компоненте выделяются клитики, местоименные объектные суффиксы, производится анализ именной морфологии целиком и в глагольной – неличных форм глагола.

Правила, применяемые на уровнях данного компонента, описаны в отдельном для каждого уровня каскада файле формата .json, содержащем массив объектов, каждый из которых включает правило захвата и правило замены на языке регулярных выражений, а также поле для описания правила; в качестве иллюстрации на рис. 2 приведено несколько таких объектов. Правила на каждом уровне применяются параллельно; если регулярное выражение в правиле захвата соответствует необработанной части строки, оно применяется к ней и создается новая ветка анализа. Уровни каскада применяются последовательно для каждой ветки. При этом система разработана таким образом, что на каждый последующий уровень каскада поступают как разборы из предыдущего уровня, так и разборы, к которым не были применены один или больше предыдущих уровней; в частности, до каждого уровня дойдут как изначальная запись, к которой не было применено ни одно из правил, так и записи, к которым были применены только несколько из предыдущих уровней.

Составление правил для анализа неличных глагольных форм и именной морфологии явилось одним из самых трудоемких этапов разработки системы. Ниже

описаны некоторые особенности морфологии тигре, представленные в правилах этого компонента программы.

Основные грамматические значения, представленные в именной морфологии тигре, – род (мужской и женский) и число (единственное, паукальное и множественное). Представленный анализатор позволяет получить информацию о роде существительного в тех случаях, в которых эта информация представлена в словоизменительных показателях, чаще всего – суффиксах. Здесь, однако, следует отметить некоторые особенности морфологии тигре, заключающиеся в том, что некоторые словообразовательные процессы могут вовлекать использование при склонении показателей рода, отличного от приписанного лексеме. Так, пейоратив выражается присоединением к существительному женского рода показателей числа мужского рода и, в меньшей степени, наоборот:

(Raz 1983:25—26)

(1а) garh-at
поле-F.SG
‘поле’

(1б) garh-etāy
поле-M.SG
‘плохое поле’

Рисунок 2. Фрагмент файла .json с информацией об обработке форм “ломаного множественного”

```
{
    "comment": "broken plural: >aCCAC; qor -> >aqwAr, kis -> >akyAs",
    "regex":
    "^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy
    hHX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
    "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL"
},
{
    "comment": "broken plural: >aCC[uA]?C (2nd consonant is not a laryngeal or a semivowel);
    dabr -> >adbr, daqal -> adqul, kalb -> >aklAb",
    "regex":
    "^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpP])
    ([uA]?)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
    "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL"
},
{
    "comment": "broken plural: L[aA]C[uA]?C, L = laryngeal; Hlm -> HAlAm, >adg -> >adug",
    "regex":
    "^(?<root>(?<C1>[hHX><])([aA])(?<C2>[lmrsxqbtnkzdGgTCSfcZpP])([uA]?)(?<C3>[lmrsxqbtnkzdGgT
    CSfcZpPwyhHX><]))$",
    "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
}
```

По наблюдаемым показателям числа в таких случаях невозможно достоверно определить такие деривационные значения, как пейоратив, без обращения к словарю. Однако поскольку данные морфемы, безусловно, являются продуктивными, было принято решение выделять их.

Именное склонение представлено в тигре двумя основными моделями. Первая – суффиксация, совмещенная в некоторых случаях с применением специальных для каждого суффикса морфонологических правил, например, геминации при образовании множественного числа от существительных структуры CVC в единственном числе:

(Raz 1983:18)

(2а) ʕad
деревня/племя.SG

(2б) ʕadd-otāt
деревня/племя-F.PL

Эти закономерности рассматриваются в большинстве случаев как совмещенные с присоединением конкретного суффикса и поэтому в основном описаны отдельными правилами замены.

Вторая, более сложная модель образования множественного числа, носит в семитологии название "ломаного множественного" (англ. broken plural) и заключается главным образом в различии вокалических шаблонов основы для разных значений числа, наряду с возможной в разных моделях аффиксацией и геминацией. В тигре для существительных насчитывается шесть основных типов образования "ломаного множественного", наряду с их различными подтипами, в которых присутствует, в частности, дополнительное распределение удлинения предпоследней гласной (фонемный переход */a/ > /a:/*) и геминации предпоследнего согласного основы, и которые, в свою очередь, характеризуются особыми морфонологическими правилами.

Хотя, как было отмечено выше, не все деривационные процессы поддаются анализу без привлечения лексикологических данных, некоторые морфемы и шаблоны можно рассмотреть как преимущественно словообразовательные. так, данный анализатор выделяет показатели диминутива, различающиеся по роду (*-a:y*, *-eta:y* для мужского рода; *-at*, *-it*, *-atit* для женского рода).

Помимо описанных выше закономерностей, правила в данном компоненте анализатора были созданы также для описания образования неличных форм глагола. Прежде всего это причастия. в языке тигре представлены действительные и страдательные причастия; модели, по которым они строятся, зависят от типа глагола и наличия показателей залога или актантной деривации. помимо причастий, глагольные основы в языке тигре участвуют в образовании инфинитивов и существительных – имен инструмента, места, деятеля, результата действия; все эти формы образуются с участием продуктивных моделей, в основе которых лежат вокалический шаблон, паттерн геминации, а также аффиксация.

Сегменты помимо описанных выше присоединяются к словоформе, в целом, конкатенативно и для анализа представляют существенно меньшую сложность, чем явления, описанные выше. Среди сегментов в левой части орфографического слова это прежде всего определенный артикль *la-*, отрицательная частица *ʔi-* и сочинительный союз *wa-*; среди сегментов в правой части – местоименные суффиксы существительных и глаголов, присоединение которых сопряжено с применением некоторых морфонологических правил.

Кроме того, в состав данного компонента системы включен лексикон. Лексикон, в соответствии с поставленной задачей, содержит преимущественно служебные слова. Лексикон реализован в качестве последнего уровня каскада; это позволяет применять его к формам, содержащим также морфемы, описанные в правилах предыдущих уровней.

Разборы, полученные в результате работы первого компонента системы, могут быть помечены как финальные или нефинальные. Разборы, помеченные как нефинальные, поступают во второй компонент системы, обрабатывающий личные формы глаголов.

3.3.2. Анализ личных форм глагола

Обработка личных форм глагола представляет собой самую затратную по ресурсам часть анализа в данной системе. подход, примененный для анализа личных форм глагола, возможно описать термином "анализ через генерацию" (англ. analysis by generation, см. (Wintner 2014:50), где этот термин употреблен для описания, в частности, морфологического анализатора для арабского BAMA). Компонент,

ответственный за анализ личных форм глаголов, получая на вход цепочку символов, перебором генерирует все возможные сочетания согласных и возможные при данном сочетании типы глагола и для каждого полученного корня-"кандидата" строит полную парадигму, отбирая для анализа те из полученных форм, которые совпадают с входной цепочкой. Следует, безусловно, отметить, что такой подход делает анализ затратным по времени, однако при достаточно адекватном описании парадигм спряжений способен обеспечить высокую точность анализа личных форм глагола и, кроме того, высокую полноту в отношении ячеек парадигмы, имеющих идентичные показатели, но различающихся набором выражаемых грамматических значений.

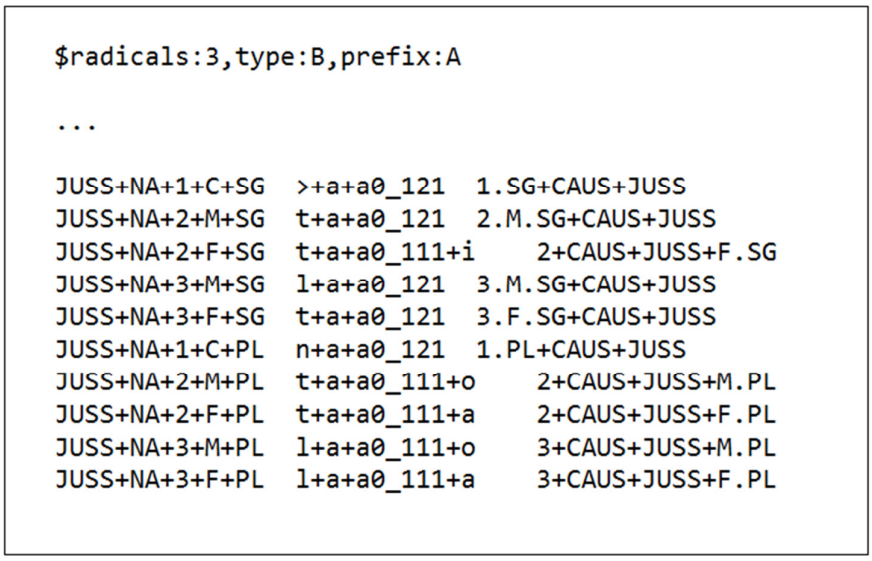
Для разработки глагольного анализатора было смоделировано глагольное спряжение тигре. принципы его обработки, такие, как наличие различных типов спряжений (A—D по (Raz 1983)), учет подтипов по значениям залога и актантной деривации (подтипы *t(ə)-*, *a-*, *at-*, *atta-*), заложены внутри программного кода. Описание самих спряжений, однако, для облегчения его создания было осуществлено в отдельном файле, имеющему несложный для чтения формат и легко поддающемуся редактированию (хотя предоставление такой возможности пользователю не входило в задачи разработки и явилось, таким образом, положительным побочным эффектом).

Фрагмент описания спряжения приведён для иллюстрации на рис. 3. Каждая парадигма состоит из: 1. заголовка, в котором задается число согласных в основе, тип глагола, подтип, определяемый присоединяемым деривационным префиксом; 2. описаний всех ячеек парадигмы. Порядок следования записей внутри парадигмы не имеет значения. Для данного файла предусмотрены описания всех возможных в языке тигре личных форм глаголов; при этом в исходном коде установлены некоторые внутренние ограничения на сочетаемость граммем (так, формы для первого лица не различаются по роду, в то время как для второго и третьего – различаются по роду для любого глагола; для императива возможно задать только формы второго лица).

Данный компонент системы является завершающим в разборе словоформы. В случае совпадения одной из сгенерированных форм с входной цепочкой символов корень при глоссировании представлен на лексическом уровне набором консонантов и вспомогательными символами, позволяющими определить его тип (для глаголов типа B после второго согласного ставится цепочка "(2)", обозначающая геминацию соответствующего согласного в большинстве ячеек парадигмы, для глаголов типа C и

D – цепочка "(A)" после согласного, образующего слог с */a:/* во всех ячейках парадигмы). Совпавший с входной цепочкой разбор помечается как финальный.

Рисунок 3. Фрагмент файла описания глагольного спряжения



```
$radicals:3,type:B,prefix:A

...

JUSS+NA+1+C+SG  >+a+a0_121  1.SG+CAUS+JUSS
JUSS+NA+2+M+SG  t+a+a0_121  2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG  t+a+a0_111+i    2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG  l+a+a0_121  3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG  t+a+a0_121  3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL  n+a+a0_121  1.PL+CAUS+JUSS
JUSS+NA+2+M+PL  t+a+a0_111+o    2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL  t+a+a0_111+a    2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL  l+a+a0_111+o    3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL  l+a+a0_111+a    3+CAUS+JUSS+F.PL
```

4. Оценка качества

Оценка качества анализа морфологии для относительно малоресурсного языка со столь сложной морфологией, как тигре, представляет собой трудоемкую задачу. Как было отмечено ранее (Введение), для языка доступно достаточно большое число письменных текстов. Текстов с морфологической разметкой, однако, доступно существенно меньше, и все они содержатся в грамматиках и статьях, упомянутых нами в 1.2. В грамматике (Raz 1983) представлены глоссированные тексты на тигре, однако они записаны не в эфиопской графике, а в фонемной транскрипции; несмотря на то, что, как было отмечено ранее, графемы эфиопской письменности, в особенности в том виде, в котором они используются для записи тигре[1], отражают фонемный состав слога однозначно и, таким образом, возможна обратная транслитерация этих текстов в эфиопскую графику, предпочтительно для анализа качества использовать текст, изначально записанный в ней, поскольку тестирование системы на нем лучше отразит работу системы с предназначенными для нее входными данными, то есть с естественным текстом в эфиопской графике.

Материал для тестового корпуса составили две статьи из номеров газеты Eritrea Haddas (ኦረትርየ ሐዳስ), содержащие 278 уникальных словоформ (общий объем статей – 491 слово). Для всех словоформ был произведен морфологический разбор вручную с использованием, преимущественным образом, словаря (Littmann and Höffner 1962) и грамматики (Raz 1983). Результаты ручного анализа затем сравнивались с выходными данными анализатора, полученными при его работе на тестовом корпусе.

Среди метрик, традиционно используемых для оценки качества систем автоматического анализа морфологии, следует упомянуть точность и полноту, а также высчитываемую на их основе сбалансированную F- (F1-) меру. Для целей оценки качества работы парсера возможно использовать модификации этих метрик, определяемые следующим образом: для точности – $P(w_i) = \frac{|t_i \cap r_i|}{|t_i|}$, для полноты – $R(w_i) = \frac{|t_i \cap r_i|}{|r_i|}$, где w_i – словоформа в выборке, t_i – множество разборов словоформы,

[1] Так, в тигре нет графем, дублирующих слоги с одной и той же согласной фонемой, что отличает его, например, от варианта эфиопской письменности, используемого для записи амхарского; последний в силу диахронических причин содержит, в частности, два набора графем для записи слогов с образующей согласной /s/ (ሠ и ሰ), четыре – для слогов с /h/ (ሀ ሐ ኸ ኀ), а также предоставляет по два способа для записи сочетания большинства согласных с звонким билабиальным аппроксимантом /w/.

порожденных анализатором, r_i – множество разборов, составленных экспертом. Данные метрики подсчитываются для каждой словоформы; для всей выборки метриками являются средние их значения (Paroubek 2007). Сбалансированная F-мера подсчитывается по формуле: $F_1 = 2 \times \frac{P \times R}{P + R}$.

Особенности работы анализатора требуют перед описанием результатов пояснить методику определения правильности полученных программой разборов. Как было упомянуто выше (3.3.1), программа помечает разборы как финальные, для которых постулируется полный разбор до консонантного корня, или нефинальные, содержащие только часть морфем в данной словоформе. Поскольку в настоящий момент в систему не включен лексикон корней, представляется правильным учитывать только финальные разборы, а среди них правильными считать только те, в которых верно выделен как консонантный корень, так и все остальные морфемы, включая вокалический паттерн и паттерн геминации.

С учетом этих особенностей оценки приведем метрики, подсчитанные для результатов работы парсера на тестовом корпусе.

Таблица 1. Значения метрик по результатам тестирования

Точность	Полнота	F-мера
15,56%	64,42%	25,07%

Следует отметить, что полученное значение точности является достаточно низким. Это означает, что многие из порождаемых программой разборов являются, по нашей классификации выше, неверными.

Представляется, что этот показатель можно улучшить прежде всего добавлением полноценного лексикона корней. В самом деле, успешных разборов больше для форм, представленных на настоящий момент в лексиконе; при этом это утверждение справедливо не только в отношении содержащихся в лексиконе цельных словоформ (в частности, парадигм нескольких частотных глаголов, включая связки ገበ */gabʔa/*, ሀለ */halla/*, አለ */ʔala/*), но и в отношении корней (например, связки ብዲብ */bdib/*, присоединяющей объектные местоименные суффиксы по регулярной модели). Поскольку на данном этапе лексикон корней не предполагалось считать

основополагающей частью системы и опираться на его содержимое для определения анализатором правомерности включения разбора с тем или иным корнем в выходные данные, ожидаемо порождается очень большое число не существующих в языке корней. Особенно часто эта проблема наблюдается, когда в одном из разборов анализатором верно выделен какой-либо конкатенативно присоединяемый аффикс, а в нескольких других разборах этой же словоформы соответствующая цепочка символов включена в состав неверно выделенного корня.

Проблема усугубляется порождением всех возможных геминированных вариантов для словоформы. В то время как для аффиксов несуществующие варианты с удвоением содержащихся в них согласных не проходят на следующий уровень каскада, некоторые цепочки попадают в финальные разборы. До некоторой степени точность в этих случаях может быть увеличена совершенствованием правил, содержащихся в регулярных выражениях на соответствующих уровнях; тем не менее, многие из объемных наборов паттернов геминации отражают реальные последовательности, возможные в потенциальном корне, что в особенности касается форм множественного числа (как образуемого суффиксальным способом, так и "ломаного множественного"). Представляется, что для существенного улучшения точности реализация полноценного лексикона корней является необходимой.

Мы оцениваем полученное значение полноты как свидетельствующее о в целом удовлетворительном отражении созданными нами правилами морфологии словоформ тигре. Отметим, что наилучшие результаты в отношении порождения наибольшего числа корректных разборов наблюдаются для глаголов, в особенности для их личных форм, что, с учетом сложности семитской глагольной морфологии, мы считаем значительным успехом настоящей работы. Тот факт, что личные формы глагола, за исключением присоединяемых к ним объектных местоименных суффиксов и клитик, обрабатываются в отдельном компоненте анализатора принципиально отличным от других форм образом, посредством порождения, по нашему мнению, может – в рамках обсуждения подхода, основанного на правилах – говорить о том, что эксплицитное задание всех парадигм словоизменения вносит меньше ошибок и несет меньший потенциал для существования не учтенных в правилах контекстов, чем используемый в остальной части системы анализ в направлении "снаружи внутрь". Для извлечения возможно большего числа корректных разборов, очевидно, требуется как создание лексикона корней, так и дальнейшее совершенствование правил. В частности, более

точного описания в правилах анализатора требует фонотактика языка тигре. Следует, несмотря на данные недостатки, отметить, что некорневые морфемы в неполных разборах неличных форм глаголов и существительных система в целом обрабатывает хорошо.

Заключение

В рамках настоящего исследования была создана система автоматического анализа морфологии для текстов на языке тигре в эфиопской графике. В системе совмещены два блока, основанные на направлениях анализа словоформ "снаружи внутрь" и "изнутри снаружу" (то есть в направлении к корню и от корня соответственно). Для обеспечения работы данных блоков созданы многочисленные правила, в процессе разработки которых были – с разной степенью успешности (см. подробнее раздел 4) – решены отдельные проблемы обработки преимущественно неконкатенативной морфологии тигре.

Наибольших успехов удалось добиться в анализе личных форм глаголов; направлением дальнейшего совершенствования системы может стать распространение подхода, примененного при разработке правил для соответствующего компонента, на анализ и других явлений морфологии тигре.

Кроме того, важным для улучшения невысокого на данный момент значения точности представляется, во-первых, создание полноценного лексикона корней для анализатора, во-вторых, уточнение контекстов, в которых допустима – либо, при подключении в ходе дальнейшей разработки статистических методов, вероятна – геминация.

Мы выражаем надежду, что первые результаты на пути автоматического анализа морфологии языка тигре, полученные нами в ходе настоящего исследования, окажутся полезными для исследователей языка и для прикладных задач, связанных с ним.

Список литературы

Городенцева 2010 – Городенцева, Т. А. Автоматный анализ неконкатенативной морфологии: дипломная работа. Московский гос. университет, Москва, 2010.

Alemayehu and Willett 2002 – Alemayehu, N., Willett, P. Stemming of Amharic words for information retrieval. *Literary and Linguistic Computing*, 2002, 17(1), 1–17.

Amtrup 2003 – Amtrup, J. Morphology in machine translation systems: Efficient integration of finite state transducers and feature structure descriptions. *Machine Translation*, 2003, 18, 213–235.

Argaw and Asker 2007 – Argaw, A. A., Asker, L. An Amharic stemmer: Reducing words to their citation forms. In: *Proceedings of the Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources – ACL 2007*. 2007. Pp. 104—110.

van den Bosch 1997 – van den Bosch, A. *Learning to Pronounce Written Words: A Study in Inductive Language Learning. PhD dissertation*. University of Maastricht, Maastricht, The Netherlands, 1997.

Bulakh 2015 – Bulakh, M. The Prefix ʾat- in Tigre. In: R. Voigt (ed.), *Tigre Studies in the 21st Century / Tigre-Studien im 21. Jahrhundert. Studien zum Horn von Afrika*. 2. Köln, 2015. Pp. 15–45.

Creutz and Lagus 2005 – Creutz, M., Lagus., K. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. Technical Report A81, *Publications in Computer and Information Science*. Helsinki University of Technology, 2005.

Demeke and Getachew 2006 –Demeke, G. A., Getachew, M. Manual annotation of Amharic news items with part-of-speech tags and its challenges. In: *Ethiopian Languages Research Center Working Papers*, 2006, 2, pp. 1—16.

Elias 2014 – Elias, D. *The Tigre language of Ginda', Eritrea: short grammar and texts. PhD thesis.* Harvard University, USA, 2005.

Ethnologue – Eberhard, D. M., Simons, G. F., Fennig, C. D. (Eds.). *Ethnologue: Languages of the World.* 22nd edition. Dallas, Texas: SIL International, 2019. Online version: http://www.ethnologue.com.

Faust 2014 – Faust, N. Templatic metathesis in Tigre imperatives. *Phonology*, 2014, 31, 209—227.

Gambäck and Asker 2010 – Gambäck, B., Asker, L. Experiences with developing language processing tools and corpora for Amharic. In: *2010 IST-Africa*, Durban, 2010, pp. 1-8.

Gasser 2011 – Gasser, M. *HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya.* Paper presented at the Conference on Human Language Technology for Development, Alexandria, Egypt. 2011.

Gezmu et al. 2018 – Gezmu, A. M., Seyoum, B. E., Gasser, M., Nürnberger, A. Contemporary Amharic corpus: automatically morpho-syntactically tagged Amharic corpus. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*. Santa Fe, New Mexico, USA, 2018. Pp. 65—70.

Goldsmith 2000 – Goldsmith, J. Linguistica: An automatic morphological analyzer. In *Proceedings of the Main Session of the Chicago Linguistic Society's 36th Meeting*, vol. 36-1, 2000.

Hammarström and Borin 2011 – Hammarström, H., Borin, L. Unsupervised Learning of Morphology. *Computational Linguistics*, 2011, 37(2), 309—350.

Haspelmath and Sims 2010 – Haspelmath, M., Sims, A. D. *Understanding Morphology.* 2nd ed. London, UK: Hodder Education, 2010.

Karttunen 1994 – Karttunen, L. Constructing Lexical Transducers. In *Coling 94. The 15th International Conference on Computational Liguistics. Proceedings*, 1994. Pp. 406—411.

Kazakov and Manandhar 2001 – Kazakov, D., Manandhar, S. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 2001, 43, 121–162.

Koskenniemi 1983 – Koskenniemi, K. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production.* Technical Report No. 11. Department of General Linguistics, University of Helsinki. 1983.

Leslau 1945 – Leslau, Wolf. 1945. Short grammar of Tigré. *American Oriental Society*, 65, 164—203.

Leslau 1982 – Leslau, W. North Ethiopic and Amharic cognates in Tigre. *Supplemento agli Annali*. N. 31. Vol. 42. Napoli: Instituto orientale di Napoli, 1982.

Littmann and Höffner 1962 – Littmann, E., Höfner, M. *Wörterbuch der Tigre-Sprache: Tigre-Deutsch-Englisch.* Wiesbaden: Franz Steiner Verlag, 1962.

Nakano 1982 – Nakano, A., Yoichi T. *A Vocabulary of Beni Amer Dialect of Tigre*. Tokyo: Institute for the Study of Languages and Cultures of Asia and Africa, 1982.

Palmer 1962 – Palmer, F. R. The Morphology of the Tigre Noun. *London Oriental Series*. No. 13. London: Oxford University Press, 1962.

Paroubek 2007 – Paroubek, P. Evaluating Part Of Speech Tagging and Parsing. In: *Evaluation of Text and Speech Systems*, L. Dybkjær, H. Hemsen and W. Minker (eds.). Kluwer Academic Publisher, 2007. Pp. 97-116.

Raz 1980 – Raz, S. The Morphology of the Tigre Verb (Mansa Dialect). *Journal of Semitic Studies* 25/1, 66-84; 25/2, 205-238. 1980.

Raz 1983 – Raz, S. *Tigre Grammar and Texts*. Malibu, California, USA: Undena Publications, 1983.

Rychlý and Suchomel 2016 – Rychlý, P., Suchomel, V. Annotated Amharic Corpora. In: *International Conference on Text, Speech, and Dialogue*, pp. 295-302.

Scannell 2007 – Scannell, K. (2007). The Crúbadán project: Corpus building for under-resourced languages. In C. Fairon, H. Naets, A. Kilgarriff, G.-M. de Schryver (Eds.), *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop*. Vol. 4. Pp. 5–15.

Shabait – *Eritrea – Ministry of Information*. http://www.shabait.com/

Shaertigre – *Shaertigre* [Blog]. http://www.shaertigre.blogspot.com/

Simpson 2009 – Simpson, A. K. *The Origin and Development of Nonconcatenative Morphology. PhD dissertation*. University of California, Berkeley, USA, 2009.

Sisay 2005 – Sisay, F. A. Part of speech tagging for Amharic using conditional random fields. In: *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. 2005. Pp. 47–54.

Soudi et al. 2007 – Soudi A., Neumann G., van den Bosch A. *Arabic Computational Morphology: Knowledge-Based and Empirical Methods*. Dordrecht, The Netherlands: Springer, 2007.

Tachbelie 2010 – Tachbelie, M. Y. *Morphology-Based Language Modeling for Amharic. Dissertationsschrift ... Doktors der Naturwissenschaften*. Universität Hamburg, Hamburg, Deutschland, 2010.

Tedla et al. 2016 – Tedla, Y. K., Yamamoto, K., Marasinghe, A. Nagaoka Tigrinya Corpus: Design and Development of Part-of-speech Tagged Corpus. In: *Language Processing Society 22nd Annual Meeting Papers Collection, Tohoku, Japan*. The Association for Natural Language Processing, 2016.

Tesfaye 2002 – Tesfaye, B. *Automatic morphological analyzer for Amharic: An experiment employing unsupervised learning and autosegmental analysis approaches. Master's thesis*. Addis Ababa University, Addis Ababa, Ethiopia, 2002.

Trost 2003 – Trost H. Morphology. In: R.Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*, Oxford University Press, 2003. Pp. 25—47.

Voigt 2008 – Voigt, R. Zum Tigre. *Aethiopica*, 2008, 11. 173—193.

Voigt 2009 – Voigt, R. Das Präsens im Tigre. *Aethiopica*, 2009, 12, 155—163.

Voigt 2015 – Voigt, R. The Development of Tigre Literature. In: R. Voigt (Ed.), *Tigre Studies in the 21st Century / Tigre-Studien im 21. Jahrhundert*, *Studien zum Horn von Afrika*, 2. Köln, 2015. Pp. 115–135.

Wicentowski 2004 – Wicentowski, R. Multilingual Noise-Robust Supervised Morphological Analysis using the WordFrame Model. In: *Proceedings of the Workshop of the ACL Special Interest Group on Computational Phonology (SIGPHON)*. Barcelona, Spain: Association for Computational Linguistics 2004. 70—77.

Wintner 2014 – Wintner, S. Morphological Processing of Semitic Languages. In: I. Zitouni (Ed.), *Natural Language Processing of Semitic Languages*. Berlin, Heidelberg: Springer Verlag, 2014. 43—66. doi:10.1007/978-3-642-45358-8

Wondwossen and Gasser 2012 – Wondwossen, M., and Gasser, M. Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming. In: *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages (SALTMIL8/AfLaT2012)*, 2012.

Приложения

Приложение 1. Исходный код анализатора на языке программирования Java

```
// ---------------------- Класс Launcher ----------------------
import java.io.IOException;
import com.beust.jcommander.JCommander;
import com.beust.jcommander.Parameter;
public class Launcher {
  @Parameter(names = { "-i", "-input" }, description = "input file path")
  String inputFilePath = "input.txt";
  @Parameter(names = { "-o", "-output" }, description = "output file path")
  String outputFilePath = "output.txt";
  @Parameter(names = "-numanalyses", description = "number of analyses to show (non-negative integer;
0 to show all analyses; or don't specify this paramenter)")
  int numAnalysesToShow = 0;
  public static void main(String[] args) {
    Launcher launcher = new Launcher();
    JCommander.newBuilder()
    .addObject(launcher)
    .build()
    .parse(args);
    launcher.run();
  }
  public void run () {
    try {
      if (numAnalysesToShow == 0) {
        new Builder().processFile(inputFilePath, outputFilePath);
      } else if (numAnalysesToShow > 0) {
        new Builder().processFile(inputFilePath, outputFilePath, numAnalysesToShow);
      } else {
        throw new IllegalArgumentException("-numanalyses must be a non-negative integer (0 to show all
analyses)");
      }
    } catch (IOException e) { e.printStackTrace(); }
  }
}
// ---------------------- Класс Builder ----------------------
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.LinkedHashSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import com.google.gson.stream.JsonReader;
public class Builder {
  VerbParadigm verbParadigm;
  ArrayList<ArrayList<PatternReplacePair>> patternCascade;
  RegexIterator regexIterator;
  Transliterator transliterator;
  private static String unprocessedPartRegex = ".*\\[(?<unprocessed>.*)\\].*";
  private static Pattern unprocessedExtractorPattern = Pattern.compile(unprocessedPartRegex);
  public Builder () {
    try {
      this.verbParadigm = new VerbParadigmBuilder().build("paradigm.txt");
      this.patternCascade = readPatterns();
      this.regexIterator = new RegexIterator();
      // ə in map file stands for disambiguation of cases like [kə][ka] from [kka] (geminated).
      // The actual [ə] sound may or may not occur in that position; this is determined by
phonotactics.
      // The ə symbol MUST be removed from any fields of GeezAnalysisPair objects immediately after
generating geminated variants.
      this.transliterator = new Transliterator("romanization-map.file");
    } catch (IOException | ParseException e) { e.printStackTrace(); }
  }
  private static ArrayList<ArrayList<PatternReplacePair>> readPatterns () throws IOException {
```

```
    ArrayList<ArrayList<PatternReplacePair>> patternList = new ArrayList<>();
    String[] patternFilePaths = { "pref-coord.json", "pref-relz.json", "pref-neg.json", "pref-
iobj.json", "suf-expl.json", "suf-pron.json", "pau_2.json", "pass-ptcp-deriv-pref.json", "nominal-
stem.json", "lexicon.json" };
    for (String path : patternFilePaths) {
      JsonReader reader = new JsonReader (new InputStreamReader(new FileInputStream(path), "UTF-8"));
      ArrayList<PatternReplacePair> curPatterns = new ArrayList<>();
      reader.beginArray();
      while (reader.hasNext()) {
        reader.beginObject();
        while (reader.hasNext()) {
          // skip comment
          reader.nextName();
          reader.nextString();
          // read regex
          reader.nextName();
          String regex = reader.nextString();
          // read replacement
          reader.nextName();
          String replacement = reader.nextString();
          // create new PRPair
          curPatterns.add(new PatternReplacePair(regex, replacement));
        }
        reader.endObject();
      }
      reader.endArray();
      reader.close();
      patternList.add(curPatterns);
    }
    return patternList;
  }
  public void processFile (String inputPath, String outputPath, int numAnalysesToShow) throws
IllegalArgumentException, IOException {
    if (numAnalysesToShow < 1) {
      throw new IllegalArgumentException("Illegal argument: number of analyses to show should be a
positive integer.");
    } else {
      buildFile (inputPath, outputPath, numAnalysesToShow);
    }
  }
  public void processFile (String inputPath, String outputPath) throws IOException {
    buildFile (inputPath, outputPath, 0);
  }
  private void buildFile (String inputPath, String outputPath, int numAnalysesToShow) throws
IllegalArgumentException, IOException {
    if (numAnalysesToShow < 0) { throw new IllegalArgumentException("numAnalysesToShow must be a non-
negative integer; 0 for showing all analyses"); }
    BufferedReader textReader = new BufferedReader(new InputStreamReader(new
FileInputStream(inputPath), "UTF-8"));
    PrintWriter writer = new PrintWriter(outputPath, "UTF-8");
    if (numAnalysesToShow == 0) {
      writer.print("Mode: Show all analyses\n\n");
    } else {
      writer.printf("Mode: Show first %d analyses%n%n", numAnalysesToShow);
    }
    ArrayList<String> lines = new ArrayList<>();
    String curLine = "";
    while ((curLine = textReader.readLine()) != null) {
      lines.add(curLine);
    }
    System.out.printf("Lines in total: %d%n", lines.size());
    int counter = 0;
    WordGlossPairComparator wgpComparator = new WordGlossPairComparator();
    for (String line : lines) {
      counter++;
      System.out.printf("Processing line: %d out of %d%n", counter, lines.size());
      ArrayList<GeezAnalysisPair> wordList = transliterator.buildFromLine(line);
      for (GeezAnalysisPair word : wordList) {
        writer.printf("* * * * * * *%n%nWord: %s%nAnalyses:%n%n", word.geezWord);
        for (String gemOrtho : word.geminatedOrthos) {
          word.analysisList.addAll(this.analyzeLine(gemOrtho));
        }
        Collections.sort(word.analysisList, Collections.reverseOrder(wgpComparator));
        int curNumAnalysesToPrint;
```

```
        // numAnalysesToShow >= 0: checked in the beginning of this method
        if (numAnalysesToShow == 0) {
          curNumAnalysesToPrint = word.analysisList.size();
        } else if (numAnalysesToShow < word.analysisList.size()) {
          curNumAnalysesToPrint = numAnalysesToShow;
        } else {
          curNumAnalysesToPrint = word.analysisList.size();
        }
        for (int i = 0; i < curNumAnalysesToPrint; i++) {
          WordGlossPair analysis = word.analysisList.get(i);
          writer.printf("%s%n%s%n%n", analysis.surfaceForm, analysis.lexicalForm);
        }
      }
    }
    System.out.print("\n\nProcessing completed!\n\n");
    textReader.close();
    writer.close();
  }
  private ArrayList<WordGlossPair> analyzeLine (String line) {
    ArrayList<WordGlossPair> analysisList = new ArrayList<>();
    analysisList.add(new WordGlossPair("[" + line + "]", "#", false));
    for (ArrayList<PatternReplacePair> curPatterns : patternCascade) {
      ArrayList<WordGlossPair> newAnalysisList = regexIterator.processLevel(analysisList,
curPatterns);
      analysisList.clear();
      for (WordGlossPair analysis : newAnalysisList) {
        analysisList.add(WordGlossPair.newInstance(analysis));
      }
    }
    ArrayList<WordGlossPair> analyzedVerbs = new ArrayList<>();
    for (WordGlossPair analysis : analysisList) {
      if (!analysis.isFinalAnalysis) {
        ArrayList<WordGlossPair> newAnalysisList = this.analyzeAsVerb(analysis);
        for (WordGlossPair verbAnalysis : newAnalysisList) {
          analyzedVerbs.add(WordGlossPair.newInstance(verbAnalysis));
        }
      }
    }
    analysisList.addAll(analyzedVerbs);
    LinkedHashSet<WordGlossPair> analysisSet = new LinkedHashSet<>(analysisList);
    analysisList = new ArrayList<>(analysisSet);
    return analysisList;
  }
  private ArrayList<WordGlossPair> analyzeAsVerb (WordGlossPair line) {
    ArrayList<WordGlossPair> analysesList = new ArrayList<>();
    // add the same unprocessed part as a variant to newLinesSet
    analysesList.add(WordGlossPair.newInstance(line));
    // extract the part to be processed
    String lineToProcess = extractUnprocessedPart(line);
    lineToProcess = lineToProcess.replaceAll("[\\[\\]]", "");
    // run all patterns from this level on the unprocessed part of the current analysis
    RootListGenerator builder = new RootListGenerator(lineToProcess);
    try {
      builder.buildRootList();
    } catch (IllegalArgumentException e) {}
    for (Root root : builder.roots) {
      LinkedHashSet<WordGlossPair> formSet = new LinkedHashSet<>();
      try {
        VerbStem baseStem = new VerbStem(root);
        ArrayList<VerbStem> derivedStems = new ArrayList<>();
        for (VerbPreformative prefix : VerbPreformative.values()) {
          switch (prefix) {
            case NO_PREFORMATIVE:
            case A:
            case T:
            case AT:
            case ATTA:
              if (baseStem.setDerivationalPrefix(prefix)) {
                VerbStem stemToAdd = VerbStem.newInstance(baseStem);
                derivedStems.add(stemToAdd);
              }
              break;
            default: break;
          }
```

```
        }
        for (VerbStem stem : derivedStems) {
          formSet.addAll(this.verbParadigm.buildAllForms(stem));
        }
        ArrayList<WordGlossPair> formList = new ArrayList<>(formSet);
        for (WordGlossPair form : formList) {
          if (form.getRawWord().equals(lineToProcess)) {
            analysesList.add(constructVerbWgPair(line, form));
          }
        }
      } catch (IllegalArgumentException e) {}
    }
    return analysesList;
  }
  private static String extractUnprocessedPart (WordGlossPair wgPair) {
    Matcher m = unprocessedExtractorPattern.matcher(wgPair.surfaceForm);
    if (m.find()) {
      return m.group("unprocessed");
    }
    return "";
  }
  private static WordGlossPair constructVerbWgPair (WordGlossPair oldWgPair, WordGlossPair
stemAnalysis) {
    WordGlossPair newWgPair = new WordGlossPair();
    boolean isFinalAnalysis = true;
    String newSurfaceForm = oldWgPair.surfaceForm;
    newSurfaceForm = newSurfaceForm.replaceAll("\\[.*\\]", stemAnalysis.surfaceForm);
    newWgPair.surfaceForm = newSurfaceForm;
    String newLexForm = oldWgPair.lexicalForm;
    newLexForm = newLexForm.replaceAll("#", stemAnalysis.lexicalForm);
    newWgPair.lexicalForm = newLexForm;
    newWgPair.isFinalAnalysis = isFinalAnalysis;
    return newWgPair;
  }
}
// ---------------------- Класс RegexIterator ----------------------
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class RegexIterator {
  // everything inside square brackets
  private static String unprocessedPartRegex = ".*\\[(?<unprocessed>.*)\\].*";
  private static Pattern unprocessedExtractorPattern = Pattern.compile(unprocessedPartRegex);
  private Pattern processorPattern;
  private Matcher processorMatcher;
  public RegexIterator() {
    this.processorPattern = Pattern.compile("");
    this.processorMatcher = this.processorPattern.matcher("");
  }
  public ArrayList<WordGlossPair> processLevel (ArrayList<WordGlossPair> lines,
      ArrayList<PatternReplacePair> patterns) {
    LinkedHashSet<WordGlossPair> newLinesSet = new LinkedHashSet<>();
    for (WordGlossPair line : lines) {
      // add the same unprocessed part as a variant to newLinesSet
      newLinesSet.add(WordGlossPair.newInstance(line));
      // extract the part to be processed
      if (!line.isFinalAnalysis) {
        String lineToProcess = extractUnprocessedPart(line);
        // run all patterns from this level on the unprocessed part of the current analysis
        for (PatternReplacePair pattern : patterns) {
          this.processorPattern = Pattern.compile(pattern.matchPattern);
          this.processorMatcher = this.processorPattern.matcher(lineToProcess);
          if (this.processorMatcher.find()) {
            String replacement = this.processorMatcher.replaceAll(pattern.replacePattern);
            // add the replacement to newLinesSet
            newLinesSet.add(constructWgPair(line, replacement));
          }
        }
      }
    }
    return new ArrayList<WordGlossPair>(newLinesSet);
  }
  static String extractUnprocessedPart (WordGlossPair wgPair) {
```

```java
    Matcher m = unprocessedExtractorPattern.matcher(wgPair.surfaceForm);
    if (m.find()) {
      return m.group("unprocessed");
    }
    return "";
  }
  static WordGlossPair constructWgPair (WordGlossPair oldWgPair, String analysis) throws
IllegalArgumentException {
    WordGlossPair newWgPair = new WordGlossPair();
    String[] morphemes = analysis.split("\\-");
    String analysisSurface = "";
    String analysisLex = "";
    boolean isFinalAnalysis = true;
    for (int i = 0; i < morphemes.length; i++) {
      String morpheme = morphemes[i];
      String[] morphemeParts = morpheme.split("\\:");
      if (morphemeParts.length != 2) { throw new IllegalArgumentException("analysis is not formatted
properly"); }
      if (morphemeParts[0].charAt(0) == '[' && morphemeParts[1].charAt(0) == '#') {
        // unanalyzed part -> the returned WGPair is not a final analysis
        isFinalAnalysis = false;
      }
      analysisSurface += morphemeParts[0];
      analysisLex += morphemeParts[1];
      if (i < morphemes.length - 1) {
        analysisSurface += "-";
        analysisLex += "-";
      }
    }
    String newSurfaceForm = oldWgPair.surfaceForm;
    newSurfaceForm = newSurfaceForm.replaceAll("\\[.*\\]", analysisSurface);
    newWgPair.surfaceForm = newSurfaceForm;
    String newLexForm = oldWgPair.lexicalForm;
    newLexForm = newLexForm.replaceAll("#", analysisLex);
    newWgPair.lexicalForm = newLexForm;
    newWgPair.isFinalAnalysis = isFinalAnalysis;
    return newWgPair;
  }
}
// ---------------------- Класс Transliterator ----------------------
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.commons.math3.util.CombinatoricsUtils;
public class Transliterator {
  // ə in the map stands for disambiguation of cases like [kə][ka] from [kka] (geminated).
  // The actual [ə] sound may or may not occur in that position; this is determined by phonotactics.
  // The ə symbol MUST be removed from any fields of GeezAnalysisPair objects immediately after
generating geminated variants.
  HashMap<Character, String> map;
  static String punctuationMarks = "[፠፡።፣፤፥፦፧፨\\.!,\\-\\:;\"'/\\\\\\|-–—‘’“”\\(\\)\\[\\]<>\\{\\}]";
  public Transliterator (String mapFilePath) throws IOException {
    this.map = new HashMap<>();
    BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(mapFilePath),
"UTF-8"));
    String currentLine;
    Pattern pPair = Pattern.compile("^(.)\\t(.+)$");
    Matcher mPair;
    while ((currentLine = reader.readLine()) != null) {
     mPair = pPair.matcher(currentLine);
     if (mPair.find() && mPair.groupCount() == 2) {
      this.map.put(mPair.group(1).charAt(0), mPair.group(2));
     }
    }
    reader.close();
  }
  public void romanizeFileReadWrite (String inputPath, String outputPath, boolean printGeez) throws
IOException {
```

```
    BufferedReader reader = new BufferedReader(new FileReader(inputPath));
    PrintWriter writer = new PrintWriter(outputPath);
    String inputLine;
    String outputLine;
    while ((inputLine = reader.readLine()) != null) {
      outputLine = this.romanizeLine(inputLine);
      if (printGeez) { writer.println(inputLine); }
      writer.println(outputLine);
      if (printGeez) { writer.println(); }
    }
    reader.close();
    writer.close();
  }
  public String romanizeLine (String geezLine) {
    String romanizedLine = "";
    char curChar;
    for (int i = 0; i < geezLine.length(); i++) {
      curChar = geezLine.charAt(i);
      if (this.map.containsKey(curChar)) {
        romanizedLine += this.map.get(curChar);
      } else {
        romanizedLine += curChar;
      }
    }
    return romanizedLine;
  }
  static ArrayList<String> generateGeminatedVariants (String ungemWordWithSchwas) {
    // ungemLine is a _non-geminated_ romanized version of a word written in Ge'ez script.
    // does NOT check whether ungemLine was generated from a genuine Ge'ez word
    // assumes conformity to pattern: (CV)+; V may include schwa
    ArrayList<String> orthoVariants = new ArrayList<>();
    ArrayList<Integer> geminablePositions = new ArrayList<>();
    orthoVariants.add(ungemWordWithSchwas.replaceAll("ə", ""));
    // the first letter is always a consonant; the first consonant can never be geminated
    // the second letter is a vowel or a schwa in ungemWordWithSchwas
    for (int i = 2; i < ungemWordWithSchwas.length(); i++) {
      char curChar = ungemWordWithSchwas.charAt(i);
      if (LetterType.isConsonant(curChar)
          && !(LetterType.isLaryngeal(curChar) || LetterType.isSemivowel(curChar))) {
        geminablePositions.add(i);
      }
    }
    for (int k = 1; k <= geminablePositions.size(); k++) {
      Iterator<int[]> combinationsIterator =
CombinatoricsUtils.combinationsIterator(geminablePositions.size(), k);
      while (combinationsIterator.hasNext()) {
        // e. g. {2, 4}
        int[] combination = combinationsIterator.next();
        ArrayList<Integer> curGeminatedPositions = new ArrayList<>();
        // e. g. combination == {2, 4}; geminablePositions == {2, 3, 5, 7, 8}; =>
curGeminatedPositions == {5, 8}
        for (int i = 0; i < combination.length; i++) {
          curGeminatedPositions.add(geminablePositions.get(combination[i]));
        }
        String curOrthoVariant = "";
        Iterator<Integer> gemPosIterator = curGeminatedPositions.iterator();
        int curPosToGeminate = gemPosIterator.next();
        for (int i = 0; i < ungemWordWithSchwas.length(); i++) {
          char curChar = ungemWordWithSchwas.charAt(i);
          if (!LetterType.isSchwa(curChar)) {
            curOrthoVariant += curChar;
          }
          if (i == curPosToGeminate) {
            if (!LetterType.isSchwa(curChar)) {
              curOrthoVariant += curChar;
            }
            if (gemPosIterator.hasNext()) {
              curPosToGeminate = gemPosIterator.next();
            }
          }
        }
        orthoVariants.add(curOrthoVariant);
      }
    }
```

```
    return orthoVariants;
  }
  // takes a line of text in Ge'ez as input
  public ArrayList<GeezAnalysisPair> buildFromLine (String inputLine) {
    ArrayList<GeezAnalysisPair> list = new ArrayList<>();
    String[] tokens = inputLine.split("[ \\-]");
    for (String token : tokens) {
      token = token.replaceAll(punctuationMarks, "");
      if (!token.isEmpty() && isInGeez(token)) {
        GeezAnalysisPair curObj = new GeezAnalysisPair();
        curObj.geezWord = token;
        String ungemOrthoWithSchwas = this.romanizeLine(token);
        curObj.geminatedOrthos = generateGeminatedVariants(ungemOrthoWithSchwas);
        // remove schwa and assign to ungeminatedOrtho. Schwas already not present in geminatedOrthos.
        curObj.ungeminatedOrtho = ungemOrthoWithSchwas.replaceAll("ə", "");
        list.add(curObj);
      }
    }
    return list;
  }
  static boolean isInGeez (String line) {
    for (int i = 0; i < line.length(); i++) {
      if (!Character.UnicodeBlock.of(line.charAt(i)).equals(Character.UnicodeBlock.ETHIOPIC)) {
        return false;
      }
    }
    return true;
  }
}
// ---------------------- Класс VerbParadigmBuilder ----------------------
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.ListIterator;
import java.util.NoSuchElementException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.Map.Entry;
public class VerbParadigmBuilder {
  // Same as VerbParadigm.paradigm, but stores cells in Set rather than in a List, for duplicate
management when parsing the paradigm file.
  // Sets of cells must be converted to Lists when creating a VerbParadigm object, for subsequent
efficient iteration.
  LinkedHashMap<
          Integer, LinkedHashMap< // number of radicals
                      VerbType, LinkedHashMap< // verb type (A, B, C, D)
                                  VerbPreformative, LinkedHashSet<VerbParadigmCell> // derivational
prefix (t-, a-, at- ...)
                                >
                    >
        > paradigm;
  private ArrayList<String> lines;
  private ListIterator<String> lineIterator;
  private int currentLineNum;
  public VerbParadigmBuilder () {
    this.paradigm = new LinkedHashMap<>();
    this.lines = new ArrayList<>();
    this.lineIterator = lines.listIterator();
    this.currentLineNum = 0;
  }
  public VerbParadigm build (String paradigmFilePath) throws IOException, ParseException {
    BufferedReader reader = new BufferedReader(new FileReader(paradigmFilePath));
    String currentLine;
    while ((currentLine = reader.readLine()) != null) {
      this.lines.add(currentLine);
    }
    reader.close();
    this.lineIterator = lines.listIterator();
    while (this.lineIterator.hasNext()) {
```

```
      currentLine = this.lineIterator.next();
      currentLineNum++;
      if (lineHasContent(currentLine)) { // skip comments and empty lines
        if (currentLine.charAt(0) == '$') {
          this.lineIterator.previous();
          currentLineNum--;
          try {
            this.readSingleParadigm();
          } catch (ParseException e) { e.printStackTrace(); } // ParseException handled here; parsing
will continue with the next paradigm in the file.
        } else {
          throw new ParseException(String.format("Line %d: Each paradigm must start with $ (dollar
sign).", this.currentLineNum), this.currentLineNum);
        }
      }
    }
    VerbParadigm paradigmObject = new VerbParadigm();
    for (Entry<Integer, LinkedHashMap<VerbType, LinkedHashMap<VerbPreformative,
LinkedHashSet<VerbParadigmCell>>>> typeMap : this.paradigm.entrySet()) {
      int curNumRadicals = typeMap.getKey();
      paradigmObject.paradigm.put(curNumRadicals, new LinkedHashMap<VerbType,
LinkedHashMap<VerbPreformative, ArrayList<VerbParadigmCell>>>());
      for (Entry<VerbType, LinkedHashMap<VerbPreformative, LinkedHashSet<VerbParadigmCell>>>
derivPrefixMap : typeMap.getValue().entrySet()) {
        VerbType curType = derivPrefixMap.getKey();
        paradigmObject.paradigm.get(curNumRadicals).put(curType, new LinkedHashMap<VerbPreformative,
ArrayList<VerbParadigmCell>>());
        for (Entry<VerbPreformative, LinkedHashSet<VerbParadigmCell>> cellSet :
derivPrefixMap.getValue().entrySet()) {
          VerbPreformative curDerivPrefix = cellSet.getKey();
          paradigmObject.paradigm.get(curNumRadicals).get(curType).put(curDerivPrefix, new
ArrayList<VerbParadigmCell>(cellSet.getValue()));
        }
      }
    }
    return paradigmObject;
  }
  private void readSingleParadigm () throws ParseException { // return true when done
    this.currentLineNum++;
    String[] paradigmHeaderTriple = this.readParadigmHeader(this.lineIterator.next());
    int curNumRadicals;
    VerbType curType;
    VerbPreformative curDerivPrefix;
    try {
      curNumRadicals = Integer.parseInt(paradigmHeaderTriple[0]);
      curType = VerbType.parseVerbType(paradigmHeaderTriple[1]);
      curDerivPrefix = VerbPreformative.parseVerbPreformative(paradigmHeaderTriple[2]);
    } catch (IllegalArgumentException e) {
      throw new ParseException(String.format(e.getMessage() + "Error in paradigm header at line %s.
Paradigm not read.", this.currentLineNum), this.currentLineNum);
    }
    // read everything before the next paradigm header (or the end of this file if there are no more
headers)
    ArrayList<String> curParadigmLines = new ArrayList<>();
    String curParadigmLine = "";
    while (this.lineIterator.hasNext()) {
      curParadigmLine = this.lineIterator.next();
      this.currentLineNum++;
      if (!curParadigmLine.isEmpty() &&
        curParadigmLine.charAt(0) == '$') {
        break;
      }
      curParadigmLines.add(curParadigmLine); // including empty lines and comments
    }
    if (this.lineIterator.hasNext()) {
      this.lineIterator.previous();
      this.currentLineNum--;
    }
    // build cell list
    LinkedHashSet<VerbParadigmCell> cellsAsSet = new LinkedHashSet<>();
    for (String line : curParadigmLines) {
      this.currentLineNum++;
      if (lineHasContent(line)) {
        try {
```

```
          cellsAsSet.add(this.buildCellFromLine(line, curNumRadicals, curType, curDerivPrefix));
        } catch (ParseException e) {
          e.printStackTrace();
        }
      }
    }
    ArrayList<VerbParadigmCell> cellsAsList = new ArrayList<>(cellsAsSet);
    // check whether paradigm already has cells for this num radicals + verb type + derivational
prefix.
    // If it does, add cells to the corresponding set.
    // If it doesn't, create a new set.
    LinkedHashSet<VerbParadigmCell> curParadigm;
    if (this.paradigm.containsKey(curNumRadicals) &&
      this.paradigm.get(curNumRadicals).containsKey(curType) &&
      this.paradigm.get(curNumRadicals).get(curType).containsKey(curDerivPrefix)) {
      curParadigm = this.paradigm.get(curNumRadicals).get(curType).get(curDerivPrefix); // get
existing paradigm
      curParadigm.addAll(cellsAsList); // add non-duplicate entries
      this.paradigm.get(curNumRadicals).get(curType).put(curDerivPrefix, curParadigm); // put the
updated paradigm back
    } else if (this.paradigm.containsKey(curNumRadicals) &&
          this.paradigm.get(curNumRadicals).containsKey(curType) &&
          !this.paradigm.get(curNumRadicals).get(curType).containsKey(curDerivPrefix)) {
      paradigm.get(curNumRadicals).get(curType).put(curDerivPrefix, cellsAsSet);
    } else if (this.paradigm.containsKey(curNumRadicals) &&
          !this.paradigm.get(curNumRadicals).containsKey(curType)) {
      LinkedHashMap<VerbPreformative, LinkedHashSet<VerbParadigmCell>> derivPrefixMap = new
LinkedHashMap<>();
      derivPrefixMap.put(curDerivPrefix, cellsAsSet);
      this.paradigm.get(curNumRadicals).put(curType, derivPrefixMap);
    } else { // if !paradigm.containsKey(curNumRadicals)
      LinkedHashMap<VerbPreformative, LinkedHashSet<VerbParadigmCell>> derivPrefixMap = new
LinkedHashMap<>();
      derivPrefixMap.put(curDerivPrefix, cellsAsSet);
      LinkedHashMap<VerbType, LinkedHashMap<VerbPreformative, LinkedHashSet<VerbParadigmCell>>>
typeMap = new LinkedHashMap<>();
      typeMap.put(curType, derivPrefixMap);
      this.paradigm.put(curNumRadicals, typeMap);
    }
  }
  private String[] readParadigmHeader(String line) throws ParseException {
    String[] headerTriple = new String[3];
    Pattern p =
Pattern.compile("^\\$radicals\\:(?<rad>[345]),type\\:(?<type>[ABCD]),prefix\\:(?<prefix>0|A|T|AT|ATTA|
AN|AS|ATTAN|ATTAS|ASTA)$");
    Matcher m = p.matcher(line);
    if (m.find() && m.groupCount() == 3) {
      headerTriple[0] = m.group("rad");
      headerTriple[1] = m.group("type");
      headerTriple[2] = m.group("prefix");
    } else {
      throw new ParseException(String.format("Error in paradigm header at line %s. Paradigm not
read.", this.currentLineNum), this.currentLineNum);
    }
    return headerTriple;
  }
  // NB: The cell being returned has the following fields empty: acceptedNumRadicals;
acceptedVerbType; acceptedDerivPrefix.
  // Manipulate the cell accordingly in the calling method before adding this cell to paradigm.
  private VerbParadigmCell buildCellFromLine (String line, int curNumRadicals, VerbType curType,
VerbPreformative curDerivPrefix) throws ParseException {
    VerbParadigmCell cell = new VerbParadigmCell();
    cell.acceptedNumRadicals = curNumRadicals;
    cell.acceptedVerbType = curType;
    cell.acceptedDerivPrefix = curDerivPrefix;
    try {
      Pattern p = Pattern.compile("^(?<grammemeset>.+)\\t(?<surfacepattern>.+)\\t(?<lexpattern>.+)$");
      Matcher m = p.matcher(line);
      String[] lineParts = new String[3];
      m.find();
      lineParts[0] = m.group("grammemeset");
      lineParts[1] = m.group("surfacepattern");
      lineParts[2] = m.group("lexpattern");
      // *** read grammemeset ***
```

```
      p = Pattern.compile("(?<mood>INDIC|JUSS|IMP)" +
                "\\+(?<tense>IMPF|PRF|NA)" +
                "\\+(?<person>[123])" +
                "\\+(?<gender>[MFC])" +
                "\\+(?<number>SG|PL)");
      m = p.matcher(lineParts[0]);
      m.find();
      cell.grammemeSet.mood = Mood.parseMood(m.group("mood"));
      cell.grammemeSet.tense = Tense.parseTense(m.group("tense"));
      cell.grammemeSet.person = Person.parsePerson(m.group("person"));
      cell.grammemeSet.gender = Gender.parseGender(m.group("gender"));
      cell.grammemeSet.number = Number.parseNumber(m.group("number"));
      // *** read surfacepattern and lexpattern ***
      ArrayList<String> surfacePatternParts = new
ArrayList<>(Arrays.asList(lineParts[1].split("\\+")));
      ArrayList<String> lexPatternParts = new ArrayList<>(Arrays.asList(lineParts[2].split("\\+")));
      // prefixes: in linear order
      ListIterator<String> surfaceIterator = surfacePatternParts.listIterator();
      ListIterator<String> lexIterator = lexPatternParts.listIterator();
      String curMorphemeSurface;
      String curMorphemeLex;
      // read prefixes
      while (surfaceIterator.hasNext() && lexIterator.hasNext()) {
        curMorphemeSurface = surfaceIterator.next();
        curMorphemeLex = lexIterator.next();
        if (curMorphemeSurface.matches("^.*_.*$")) { // contains underscore, i. e. is a root morpheme
          surfaceIterator.previous();
          lexIterator.previous();
          break;
        }
        // comply with the inverse ordering of prefixes in MorphemeDescriptionPair.prefixes
        if (!curMorphemeSurface.matches("0")) {
          cell.prefixes.add(0, new MorphemeDescriptionPair(curMorphemeSurface, curMorphemeLex));
        }
      }
      // read root
      curMorphemeSurface = surfaceIterator.next();
      curMorphemeLex = lexIterator.next();
      String[] rootSurfaceParts = curMorphemeSurface.split("_"); // i. e. "aa_121" -> {"aa", "121"}
      int[] gemPattern = new int[rootSurfaceParts[1].length()];
      for (int i = 0; i < rootSurfaceParts[1].length(); i++) {
        gemPattern[i] = Integer.parseInt(rootSurfaceParts[1].substring(i, i+1));
      }
      cell.vowelPattern = new MorphemeDescriptionPair(rootSurfaceParts[0], curMorphemeLex);
      cell.geminationPattern = gemPattern;
      // read suffixes
      while (surfaceIterator.hasNext() && lexIterator.hasNext()) {
        curMorphemeSurface = surfaceIterator.next();
        curMorphemeLex = lexIterator.next();
        if (!curMorphemeSurface.matches("0")) {
          cell.suffixes.add(new MorphemeDescriptionPair(curMorphemeSurface, curMorphemeLex));
        }
      }
      return cell;
    } catch (IndexOutOfBoundsException|NoSuchElementException|IllegalArgumentException e) {
      throw new ParseException(String.format(e.getMessage() + "%nError in form description at line %s.
Line not read.", this.currentLineNum), this.currentLineNum);
    }
  }
  private static boolean lineHasContent (String line) {
    if (line.isEmpty() ||
      line.charAt(0) == '#' ||
      line.matches("^[ \\t]+$")) { // skip comments and empty lines
      return false;
    }
    return true;
  }
}
// ---------------------- Класс VerbFormBuilder ----------------------
import java.util.ListIterator;
public class VerbFormBuilder {
  VerbParadigmCell cell;
  VerbStem stem;
  public VerbFormBuilder (VerbParadigmCell cell, VerbStem stem) throws IllegalArgumentException {
```

```
    if (stem.getRoot().length != cell.acceptedNumRadicals ||
        stem.getVerbType() != cell.acceptedVerbType ||
        stem.getDerivationalPrefix() != cell.acceptedDerivPrefix) {
      throw new IllegalArgumentException ("Stem does not correspond to paradigm cell.");
    }
    this.cell = cell;
    this.stem = stem;
  }
  public WordGlossPair build () {
    WordGlossPair word = new WordGlossPair();
    word.isFinalAnalysis = true;
    ListIterator<MorphemeDescriptionPair> it;
    MorphemeDescriptionPair curMorpheme;
    // appending prefixes
    it = cell.prefixes.listIterator(cell.prefixes.size());
    while (it.hasPrevious()) {
      curMorpheme = it.previous();
      word.surfaceForm += curMorpheme.surfaceForm;
      word.lexicalForm += curMorpheme.lexicalForm;
      if (it.hasPrevious()) {
        word.surfaceForm += "-";
        word.lexicalForm += "-";
      }
    }
    if (!cell.prefixes.isEmpty()) {
      word.surfaceForm += "-";
      word.lexicalForm += "-";
    }
    // appending root
    for (int radIndex = 0; radIndex < stem.getNumRadicals(); radIndex++) {
      // geminate if applicable
      char consToAdd = stem.getRootConsonant(radIndex).character;
      if (cell.geminationPattern[radIndex] > 0) {
        word.surfaceForm += consToAdd;
      }
      if (cell.geminationPattern[radIndex] == 2 &&
stem.getRootConsonant(radIndex).isSubjectToGemination()) {
        word.surfaceForm += consToAdd;
      }
      // append vowel if applicable
      if (radIndex != stem.getNumRadicals() - 1) {
        char charToAppend = cell.vowelPattern.surfaceForm.charAt(radIndex);
        if (charToAppend != '0') {
          word.surfaceForm += charToAppend;
        }
      }
    }
    word.lexicalForm += stem.getRootAsString() + ":";
    word.lexicalForm += cell.vowelPattern.lexicalForm;
    // appending suffixes
    if (!cell.suffixes.isEmpty()) {
      word.surfaceForm += "-";
      word.lexicalForm += "-";
    }
    it = cell.suffixes.listIterator();
    while (it.hasNext()) {
      curMorpheme = it.next();
      word.surfaceForm += curMorpheme.surfaceForm;
      word.lexicalForm += curMorpheme.lexicalForm;
      if (it.hasNext()) {
        word.surfaceForm += "-";
        word.lexicalForm += "-";
      }
    }
    return word;
  }
}
// ---------------------- Класс RootListGenerator ----------------------
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashSet;
import org.apache.commons.math3.util.CombinatoricsUtils;
public class RootListGenerator {
  String word;
```

```
  LinkedHashSet<Root> possibleRootCons; // mikattaqAla -> myktql; mykt2ql; myktqAl; myk2qAl
  LinkedHashSet<Root> roots; // myktql -> myk; myt; myq; myl; ...; mykt; mykq; ...; myktq, myktl,
mktql.
  public RootListGenerator (String word) {
    this.word = word;
    this.possibleRootCons = new LinkedHashSet<>();
    this.roots = new LinkedHashSet<>();
  }
  public void buildRootList () {
    generatePossibleRootConsNew_2(new Root(new ArrayList<ConsDescription>()), this.word, 0);
    for (Root consSet : this.possibleRootCons) {
      this.roots.addAll(generatePossibleRootsNew(consSet));
    }
  }
  private boolean generatePossibleRootConsNew_2 (Root rootCandidate, String sourceWord, int nextPos) {
    ArrayList<Root> forks = new ArrayList<>();
    if (sourceWord.isEmpty()) {
      return false;
    } else if (rootCandidate.consTemplate.isEmpty()) {
      rootCandidate.consTemplate.add(new ConsDescription(sourceWord.charAt(0), false, false));
      forks.add(Root.newInstance(rootCandidate));
    } else { // has 1 item at least
      Letter lastLetter = Letter.newInstance(rootCandidate.consTemplate.get(rootCandidate.size() -
1).consonant);
      if (nextPos == sourceWord.length()) {
        this.possibleRootCons.add(Root.newInstance(rootCandidate));
      } else {
        Letter letterToAdd = new Letter(sourceWord.charAt(nextPos));
        if (letterToAdd.isVowel() && !letterToAdd.isLongA()) {
          if (letterToAdd.character == 'o' || letterToAdd.character == 'u') {
            rootCandidate.consTemplate.add(new ConsDescription('w', false, false));
          } else if (letterToAdd.character == 'e' || letterToAdd.character == 'i') {
            rootCandidate.consTemplate.add(new ConsDescription('y', false, false));
          }
          forks.add(Root.newInstance(rootCandidate));
        } else if (letterToAdd.isLongA()) {
          // Two forks: 1. Verb of type C or D, long A is penultimate vowel belonging to the root. 2.
Long A is part of a prefix/suffix.
          // Fork 1: update last ConsDescription in rootCandidate: followed by long A
          Root fork1 = new Root(new ArrayList<>());
          for (ConsDescription cd : rootCandidate.consTemplate) {
            fork1.consTemplate.add(ConsDescription.newInstance(cd));
          }
          ConsDescription cd_A =
ConsDescription.newInstance(fork1.consTemplate.get(fork1.consTemplate.size() - 1));
          cd_A.followedByLongA = true;
          fork1.consTemplate.remove(fork1.consTemplate.size() - 1);
          fork1.consTemplate.add(cd_A);
          // Fork 2: long A is a part of a prefix/suffix; do not add to root description.
          Root fork2 = new Root(new ArrayList<>());
          for (ConsDescription cd : rootCandidate.consTemplate) {
            fork2.consTemplate.add(ConsDescription.newInstance(cd));
          }
          forks.add(fork1);
          forks.add(fork2);
        } else if (letterToAdd.isConsonant()) {
          if (letterToAdd.equals(lastLetter) &&
              !rootCandidate.consTemplate.get(rootCandidate.size() - 1).followedByLongA) {
            // update last ConsDescription in rootCandidate: geminated
            Root fork1 = new Root(new ArrayList<>());
            for (ConsDescription cd : rootCandidate.consTemplate) {
              fork1.consTemplate.add(ConsDescription.newInstance(cd));
            }
            fork1.consTemplate.remove(fork1.consTemplate.size() - 1);
            fork1.consTemplate.add(new ConsDescription(lastLetter, true, false));
            forks.add(fork1);
          } else { // lastChar == cons1, charToAdd == cons2, cons1 != cons2
            rootCandidate.consTemplate.add(new ConsDescription(letterToAdd, false, false));
            forks.add(Root.newInstance(rootCandidate));
          }
        }
      }
    }
    for (Root fork : forks) {
```

```
      generatePossibleRootConsNew_2(fork, sourceWord, nextPos + 1);
    }
    return true;
  }
  private LinkedHashSet<Root> generatePossibleRootsNew (Root consSet) {
    LinkedHashSet<Root> rootList = new LinkedHashSet<>();
    // Roots containing 3 to 5 consonants are considered.
    int n = ( consSet.size() > 5 ) ? 5 : consSet.size();
    for (int k = 3; k <= n; k++) {
      Iterator<int[]> combinationsIterator = CombinatoricsUtils.combinationsIterator(consSet.size(),
k);
      while (combinationsIterator.hasNext()) {
        int[] combination = combinationsIterator.next();
        Root curRoot = new Root();
        for (int consIndex : combination) {
          curRoot.consTemplate.add(consSet.consTemplate.get(consIndex));
        }
        rootList.add(curRoot);
      }
    }
    return rootList;
  }
}
// ---------------------- Класс VerbParadigm ----------------------
import java.util.ArrayList;
import java.util.LinkedHashMap;
public class VerbParadigm {
  /*
   * Structure of paradigm:
   * paradigm -- number_of_radicals : typeParadigm
   * typeParadigm -- verb_type : derivativesParadigm
   * derivativesParadigm -- derivational_prefix : paradigm_cells
   *
   */
  LinkedHashMap<
          Integer, LinkedHashMap< // number of radicals
                      VerbType, LinkedHashMap< // verb type (A, B, C, D)
                                  VerbPreformative, ArrayList<VerbParadigmCell> // derivational prefix
(t-, a-, at- ...)
                                  >
                      >
          > paradigm;
  public VerbParadigm () {
    this.paradigm = new LinkedHashMap<>();
  }
  public ArrayList<WordGlossPair> buildAllForms (VerbStem stem) {
    ArrayList<WordGlossPair> wordList = new ArrayList<>();
    ArrayList<VerbParadigmCell> cellList = new ArrayList<>();
    try {
      cellList = this.paradigm.get(stem.getNumRadicals())
          .get(stem.getVerbType())
          .get(stem.getDerivationalPrefix());
    } catch (NullPointerException e) { }
    for (VerbParadigmCell cell : cellList) {
      wordList.add(new VerbFormBuilder(cell, stem).build());
    }
    return wordList;
  }
}
// ---------------------- Класс VerbParadigmCell ----------------------
import java.util.ArrayList;
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
public class VerbParadigmCell {
  int acceptedNumRadicals;
  VerbType acceptedVerbType;
  VerbPreformative acceptedDerivPrefix;
  // In inverse order. E. g.: l+a+naggf; prefixes.get(0).surfaceForm == "a";
prefixes.get(1).surfaceForm == "l".
  ArrayList<MorphemeDescriptionPair> prefixes;
  MorphemeDescriptionPair vowelPattern;
  int[] geminationPattern;
  ArrayList<MorphemeDescriptionPair> suffixes;
  VerbGrammemeSet grammemeSet;
```

```
  public VerbParadigmCell () {
    this.acceptedNumRadicals = 0;
    this.acceptedVerbType = VerbType.UNKNOWN;
    this.acceptedDerivPrefix = VerbPreformative.UNKNOWN;
    this.prefixes = new ArrayList<>();
    this.vowelPattern = new MorphemeDescriptionPair("", "");
    this.geminationPattern = new int[0];
    this.suffixes = new ArrayList<>();
    this.grammemeSet = new VerbGrammemeSet();
  }
  @Override
  public int hashCode() {
    return new HashCodeBuilder(109, 113)
        .append(acceptedNumRadicals)
        .append(acceptedVerbType)
        .append(acceptedDerivPrefix)
        .append(prefixes)
        .append(vowelPattern)
        .append(geminationPattern)
        .append(suffixes)
        .append(grammemeSet)
        .toHashCode();
  }
  @Override
  public boolean equals(Object obj) {
       if (!(obj instanceof VerbParadigmCell))
            return false;
       if (obj == this)
            return true;
       VerbParadigmCell rhs = (VerbParadigmCell) obj;
       return new EqualsBuilder()
            .append(acceptedNumRadicals, rhs.acceptedNumRadicals)
            .append(acceptedVerbType, rhs.acceptedVerbType)
            .append(acceptedDerivPrefix, rhs.acceptedDerivPrefix)
            .append(prefixes, rhs.prefixes)
            .append(vowelPattern, rhs.vowelPattern)
            .append(geminationPattern, rhs.geminationPattern)
            .append(suffixes, rhs.suffixes)
            .append(grammemeSet, rhs.grammemeSet)
            .isEquals();
    }
}
// ---------------------- Класс VerbStem ----------------------
public class VerbStem {
  private Letter[] rootAsLetters;
  private VerbPreformative derivationalPrefix;
  private VerbType verbType;
  private int numRadicals;
  // Warning: Not updated if rootAsLetters or verbType are changed.
  private String rootAsString;
  public VerbStem () {
    this.rootAsLetters = new Letter[0];
    this.derivationalPrefix = VerbPreformative.NO_PREFORMATIVE;
    this.verbType = VerbType.UNKNOWN;
    this.derivationalPrefix = VerbPreformative.UNKNOWN;
    this.numRadicals = 0;
    this.rootAsString = "";
  }
  public VerbStem (Root root) throws IllegalArgumentException {
    if (root.size() < 3 || root.size() > 5) {
      throw new IllegalArgumentException("Root length must be between 3 and 5 consonants.");
    }
    int longACount = 0;
    for (ConsDescription cd : root.consTemplate) {
      if (cd.followedByLongA) { longACount++; }
      if (longACount > 1) { throw new IllegalArgumentException("Root cannot contain more than two
consonants followed by [a:] (long A)."); }
    }
    this.rootAsString = root.toString();
    this.rootAsLetters = new Letter[root.size()];
    for (int i = 0; i < root.size(); i++) {
      this.rootAsLetters[i] = root.consTemplate.get(i).consonant;
    }
    this.verbType = determineVerbType(root);
```

```
    this.derivationalPrefix = VerbPreformative.NO_PREFORMATIVE;
    this.numRadicals = this.rootAsLetters.length;
  }
  public static VerbStem newInstance (VerbStem stem) {
    VerbStem newStem = new VerbStem();
    Letter[] newRootAsLetters = new Letter[stem.rootAsLetters.length];
    for (int i = 0; i < stem.rootAsLetters.length; i++) {
      newRootAsLetters[i] = Letter.newInstance(stem.rootAsLetters[i]);
    }
    newStem.rootAsLetters = newRootAsLetters;
    newStem.rootAsString = stem.rootAsString;
    newStem.numRadicals = stem.numRadicals;
    newStem.derivationalPrefix = stem.derivationalPrefix;
    newStem.verbType = stem.verbType;
    return newStem;
  }
  public Letter[] getRoot() {
    return this.rootAsLetters;
  }
  public String getRootAsString () {
    return this.rootAsString;
  }
  public VerbPreformative getDerivationalPrefix() {
    return this.derivationalPrefix;
  }
  public VerbType getVerbType() {
    return this.verbType;
  }
  public int getNumRadicals () {
    return this.numRadicals;
  }
  public Letter getRootConsonant (int i) throws IllegalArgumentException {
    if (i < 0 || i > this.rootAsLetters.length - 1) {
      throw new IllegalArgumentException("Argument must be between 0 and this root's length.");
    } else {
      return this.rootAsLetters[i];
    }
  }
  public boolean setDerivationalPrefix (VerbPreformative prefix) {
    switch (prefix) {
      case T: this.derivationalPrefix = VerbPreformative.T; return true;
      case A: if (this.verbType == VerbType.D || this.rootAsLetters[0].isLaryngeal()) {
            return false;
          } else {
            this.derivationalPrefix = prefix;
            return true;
          }
      case ATTA: if (this.rootAsLetters[0].isLaryngeal()) {
            return false;
          } else {
            this.derivationalPrefix = prefix;
            return true;
          }
      case AT: if (this.verbType == VerbType.A) {
            // Check if any of the consonants is a laryngeal. If yes, assigning type A is possible;
otherwise it isn't.
            for (Letter consonant : this.rootAsLetters) {
              if (consonant.isLaryngeal()) { this.derivationalPrefix = prefix; return true; }
            }
            return false;
          } else {
            this.derivationalPrefix = prefix;
            return true;
          }
      default:
        this.derivationalPrefix = prefix;
        return true;
    }
  }
  public static VerbType determineVerbType (Root root) {
    if (root.size() == 3 && root.consTemplate.get(1).isGeminated) {
      return VerbType.B;
    } else if (root.size() == 4
        && root.consTemplate.get(1).followedByLongA
```

```
        && root.consTemplate.get(1).consonant.equals(root.consTemplate.get(2).consonant)) {
      return VerbType.D;
    } else if ((root.size() == 3 && root.consTemplate.get(0).followedByLongA)
        || (root.size() == 4 && root.consTemplate.get(1).followedByLongA)
        || (root.size() == 5 && root.consTemplate.get(2).followedByLongA)) {
      return VerbType.C;
    } else {
      return VerbType.A;
    }
  }
}
// ---------------------- Класс Word ----------------------
import java.util.ArrayList;
public class Word {
  ArrayList<MorphemeDescriptionPair> morphemeSet;
  public Word(ArrayList<MorphemeDescriptionPair> morphemeSet) {
    this.morphemeSet = morphemeSet;
  }
}
// ----------- Класс WordGlossPair -----------
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
public class WordGlossPair {
  String surfaceForm;
  String lexicalForm;
  boolean isFinalAnalysis;
  public WordGlossPair(String surfaceForm, String lexicalForm, boolean isFinalAnalysis) {
    this.surfaceForm = surfaceForm;
    this.lexicalForm = lexicalForm;
    this.isFinalAnalysis = isFinalAnalysis;
  }
  public WordGlossPair() {
    this.surfaceForm = "";
    this.lexicalForm = "";
    this.isFinalAnalysis = false;
  }
  public static WordGlossPair newInstance(WordGlossPair wdPair) {
    return new WordGlossPair(wdPair.surfaceForm, wdPair.lexicalForm, wdPair.isFinalAnalysis);
  }
  public String getRawWord () {
    return surfaceForm.replaceAll("\\-", "");
  }
  public int hashCode() {
    return new HashCodeBuilder(109, 113)
        .append(surfaceForm)
        .append(lexicalForm)
        .toHashCode();
  }
  @Override
  public boolean equals(Object obj) {
       if (!(obj instanceof WordGlossPair))
            return false;
        if (obj == this)
            return true;
        WordGlossPair rhs = (WordGlossPair) obj;
        return new EqualsBuilder().
            // if deriving: appendSuper(super.equals(obj)).
            append(surfaceForm, rhs.surfaceForm)
            .append(lexicalForm, rhs.lexicalForm)
            .append(isFinalAnalysis, rhs.isFinalAnalysis)
            .isEquals();
    }
}
// ---------------------- Класс WordGlossPairComparator ----------------------
import java.util.Comparator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class WordGlossPairComparator implements Comparator<WordGlossPair> {
  @Override
  public int compare (WordGlossPair wgp_1, WordGlossPair wgp_2) {
    if (wgp_1.isFinalAnalysis && !wgp_2.isFinalAnalysis) {
      return 1;
    } else if (!wgp_1.isFinalAnalysis && wgp_2.isFinalAnalysis) {
      return -1;
```

```
    } else {
      int numMorph_1;
      int numMorph_2;
      Pattern p = Pattern.compile("\\-");
      Matcher m = p.matcher(wgp_1.lexicalForm);
      numMorph_1 = (m.find()) ? m.groupCount() + 1 : 1;
      m = p.matcher(wgp_2.lexicalForm);
      numMorph_2 = (m.find()) ? m.groupCount() + 1 : 1;
      if (wgp_1.isFinalAnalysis && wgp_2.isFinalAnalysis) {
        // less morphemes => less than
        if (numMorph_1 < numMorph_2) { return 1; }
        else if (numMorph_1 == numMorph_2) { return 0; }
        else { return -1; }
      } else {
        // more morphemes => less than
        if (numMorph_1 < numMorph_2) { return -1; }
        else if (numMorph_1 == numMorph_2) { return 0; }
        else { return 1; }
      }
    }
  }
}
// ---------------------- Класс ConsDescription ----------------------
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
public class ConsDescription {
  // tArm -> {t, true}, {r, false}, {m, false}
  Letter consonant;
  boolean isGeminated;
  boolean followedByLongA;
  public ConsDescription(char consonant, boolean isGeminated, boolean followedByLongA) throws
IllegalArgumentException {
    Letter letter = new Letter(consonant);
    this.consonant = letter;
    this.isGeminated = isGeminated;
    this.followedByLongA = followedByLongA;
  }
  public ConsDescription(Letter consonantLetter, boolean isGeminated, boolean followedByLongA) throws
IllegalArgumentException {
    this.consonant = consonantLetter;
    this.isGeminated = isGeminated;
    this.followedByLongA = followedByLongA;
  }
  public static ConsDescription newInstance (ConsDescription cd) {
    return new ConsDescription(Letter.newInstance(cd.consonant), cd.isGeminated, cd.followedByLongA);
  }
  @Override
  public int hashCode() {
    return new HashCodeBuilder(29,
241).append(consonant).append(isGeminated).append(followedByLongA).toHashCode();
  }
  @Override
  public boolean equals(Object obj) {
       if (!(obj instanceof ConsDescription))
            return false;
        if (obj == this)
            return true;
        ConsDescription rhs = (ConsDescription) obj;
        return new EqualsBuilder().
            append(consonant, rhs.consonant).
            append(isGeminated, rhs.isGeminated).
            append(followedByLongA, rhs.followedByLongA).
            isEquals();
    }
}
// ---------------------- Класс GeezAnalysisPair ----------------------
import java.util.ArrayList;
public class GeezAnalysisPair {
  String geezWord;
  String ungeminatedOrtho;
  ArrayList<String> geminatedOrthos;
  ArrayList<WordGlossPair> analysisList;
  public GeezAnalysisPair() {
    this.geezWord = "";
```

```
    this.ungeminatedOrtho = "";
    this.geminatedOrthos = new ArrayList<>();
    this.analysisList = new ArrayList<>();
  }
}
// ---------------------- Класс Letter ----------------------
import org.apache.commons.lang3.ArrayUtils;
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
public class Letter {
  char character;
  static char[] consonants = {'h', 'l', 'H', 'm', 'r', 's', 'x', 'q', 'b', 't', 'n', '>', 'k', 'w',
'<', 'z', 'y', 'd', 'G', 'g', 'T', 'C', 'S', 'f', 'c', 'Z', 'p', 'P', 'X'};
  private static char[] vowels = {'i', 'e', 'a', 'A', 'u', 'o'};
  private static char[] laryngeals = {'<', '>', 'h', 'H', 'X'};
  private static char[] semivowels = {'w', 'y'};
  private static char longA = 'A';
  public Letter (char c) {
    this.character = c;
  }
  public static Letter newInstance (Letter letter) {
    return new Letter(letter.character);
  }
  boolean isConsonant () {
    return ArrayUtils.contains(consonants, this.character) ? true : false;
  }
  boolean isVowel () {
    return ArrayUtils.contains(vowels, this.character) ? true : false;
  }
  boolean isLaryngeal () {
    return ArrayUtils.contains(laryngeals, this.character) ? true : false;
  }
  boolean isSemivowel () {
    return ArrayUtils.contains(semivowels, this.character) ? true : false;
  }
  boolean isLongA () {
    return this.character == longA ? true : false;
  }
  boolean isSubjectToGemination () {
    // laryngeals and semivowels are never geminated
    return !(this.isLaryngeal() || this.isSemivowel());
  }
  @Override
  public int hashCode() {
    return new HashCodeBuilder(109, 113).append(character).toHashCode();
  }
  @Override
  public boolean equals(Object obj) {
       if (!(obj instanceof Letter))
            return false;
        if (obj == this)
            return true;
        Letter rhs = (Letter) obj;
        return new EqualsBuilder()
            .append(character, rhs.character).isEquals();
    }
}
// ---------------------- Класс Verb ----------------------
import java.util.ArrayList;
public class Verb {
  ArrayList<MorphemeDescriptionPair> prefixes;
  VerbStem stem;
  MorphemeDescriptionPair vowelTemplate;
  ArrayList<MorphemeDescriptionPair> suffixes;
  VerbGrammemeSet grammemeSet;
  public Verb(ArrayList<MorphemeDescriptionPair> prefixes, VerbStem stem, MorphemeDescriptionPair
vowelTemplate,
      ArrayList<MorphemeDescriptionPair> suffixes, VerbGrammemeSet grammemeSet) throws
IllegalArgumentException {
    this.prefixes = prefixes;
    this.stem = stem;
    // Check whether the vowel template argument corresponds to root in stem argument.
    // E. g., passing the "aa" template with a quadriradical root such as "trgm" would be invalid.
    if (stem.getRoot().length != vowelTemplate.surfaceForm.length() + 1) {
```

```
      throw new IllegalArgumentException("Vowel template does not correspond to this root.");
    }
    this.vowelTemplate = vowelTemplate;
    this.suffixes = suffixes;
    this.grammemeSet = grammemeSet;
  }
}
// ---------------------- Класс VerbGrammemeSet ----------------------
public class VerbGrammemeSet {
  Mood mood;
  Tense tense;
  Person person;
  Gender gender;
  Number number;
  public VerbGrammemeSet () {
    this.mood = Mood.UNKNOWN;
    this.tense = Tense.UNKNOWN;
    this.person = Person.UNKNOWN;
    this.gender = Gender.UNKNOWN;
    this.number = Number.UNKNOWN;
  }
  public VerbGrammemeSet(Mood mood, Tense tense, Person person, Gender gender, Number number) throws
IllegalArgumentException {
    if (person == Person.UNKNOWN ||
      gender == Gender.UNKNOWN ||
      number == Number.UNKNOWN ||
      (mood == Mood.INDICATIVE && tense == Tense.UNKNOWN) ||
      ((mood == Mood.JUSSIVE || mood == Mood.IMPERATIVE) && tense != Tense.UNKNOWN) ||
      (mood == Mood.IMPERATIVE && person != Person.P2) ||
      (person == Person.P1 && gender != Gender.COMMON)) {
      throw new IllegalArgumentException ("Invalid grammeme set passed as argument.");
    }
    this.mood = mood;
    this.tense = tense;
    this.person = person;
    this.gender = gender;
    this.number = number;
  }
}
// ---------------------- Класс MorphemeDescriptionPair ----------------------
public class MorphemeDescriptionPair {
  String surfaceForm; // e. g. "ka"
  String lexicalForm; // e. g. "POSS.2.F.SG"
  public MorphemeDescriptionPair(String surfaceForm, String lexicalForm) {
    this.surfaceForm = surfaceForm;
    this.lexicalForm = lexicalForm;
  }
}
// ----------- Класс PatternReplacePair -----------
public class PatternReplacePair {
  String matchPattern;
  String replacePattern;
  public PatternReplacePair (String matchPattern, String replacePattern) {
    this.matchPattern = matchPattern;
    this.replacePattern = replacePattern;
  }
}
// ---------------------- Класс Root ----------------------
import java.util.ArrayList;
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
public class Root {
  ArrayList<ConsDescription> consTemplate;
  public Root (ArrayList<ConsDescription> template) {
    this.consTemplate = template;
  }
  public Root () {
    this.consTemplate = new ArrayList<>();
  }
  public static Root newInstance (Root root) {
    ArrayList<ConsDescription> newConsTemplate = new ArrayList<>();
    for (ConsDescription cd : root.consTemplate) {
      newConsTemplate.add(ConsDescription.newInstance(cd));
    }
```

```
    return new Root(newConsTemplate);
  }
  public int size () {
    return this.consTemplate.size();
  }
  @Override
  public String toString () {
    String output = "";
    for (ConsDescription radical : this.consTemplate) {
      String toAppend = "";
      toAppend += radical.consonant.character;
      if (radical.isGeminated) { toAppend += "(2)"; }
      if (radical.followedByLongA) { toAppend += "(A)"; }
      output += toAppend;
    }
    return output;
  }
  @Override
  public int hashCode() {
    return new HashCodeBuilder(109, 113).append(consTemplate).toHashCode();
  }
  @Override
  public boolean equals(Object obj) {
       if (!(obj instanceof Root))
            return false;
        if (obj == this)
            return true;
        Root rhs = (Root) obj;
        return new EqualsBuilder()
            .append(consTemplate, rhs.consTemplate)
            .isEquals();
    }
}
// ---------------------- Перечисляемый тип Gender ----------------------
public enum Gender {
  FEMININE, MASCULINE, COMMON, UNKNOWN;
  public static Gender parseGender (String s) throws IllegalArgumentException {
    switch (s) {
      case "M": return Gender.MASCULINE;
      case "F": return Gender.FEMININE;
      case "C": return Gender.COMMON;
      default: throw new IllegalArgumentException("Couldn't parse gender.");
    }
  }
}
// ---------------------- Перечисляемый тип LetterType ----------------------
import org.apache.commons.lang3.ArrayUtils;
public enum LetterType {
  CONSONANT, VOWEL;
  private static char[] consonants = {'h', 'l', 'H', 'm', 'r', 's', 'x', 'q', 'b', 't', 'n', '>', 'k',
'w', '<', 'z', 'y', 'd', 'G', 'g', 'T', 'C', 'S', 'f', 'c', 'Z', 'p', 'P', 'X'};
  private static char[] vowels = {'i', 'e', 'a', 'A', 'u', 'o'};
  private static char[] laryngeals = {'<', '>', 'h', 'H', 'X'};
  private static char[] semivowels = {'w', 'y'};
  private static char longA = 'A';
  // schwa symbol is used during preprocessing (i. e. generating geminated variants in Transliterator)
to differentiate between two syllables which have identical consonants, and a geminated consonant (e.
g. between [kə][ka] and kka)
  // schwa is NOT treated as a (vowel) phoneme
  private static char schwa = 'ə';
  static boolean isConsonant (char c) {
    return ArrayUtils.contains(consonants, c) ? true : false;
  }
  static boolean isVowel (char c) {
    return ArrayUtils.contains(vowels, c) ? true : false;
  }
  static boolean isLongA (char c) {
    return c == longA ? true : false;
  }
  static boolean isLaryngeal (char c) {
    return ArrayUtils.contains(laryngeals, c) ? true : false;
  }
  static boolean isSemivowel (char c) {
    return ArrayUtils.contains(semivowels, c) ? true : false;
```

```
  }
  static boolean isSchwa (char c) {
    return c == schwa ? true : false;
  }
}
// ---------------------- Перечисляемый тип Mood ----------------------
public enum Mood {
  INDICATIVE, JUSSIVE, IMPERATIVE, UNKNOWN;
  public static Mood parseMood (String s) throws IllegalArgumentException {
    switch (s) {
      case "INDIC": return Mood.INDICATIVE;
      case "JUSS": return Mood.JUSSIVE;
      case "IMP": return Mood.IMPERATIVE;
      default: throw new IllegalArgumentException("Couldn't parse mood.");
    }
  }
}
// ---------------------- Перечисляемый тип Number ----------------------
public enum Number {
  SINGULAR, PLURAL, UNKNOWN;
  public static Number parseNumber (String s) throws IllegalArgumentException {
    switch (s) {
      case "SG": return Number.SINGULAR;
      case "PL": return Number.PLURAL;
      default: throw new IllegalArgumentException("Couldn't parse number.");
    }
  }
}
// ---------------------- Перечисляемый тип Person ----------------------
public enum Person {
  P1, P2, P3, UNKNOWN;
  public static Person parsePerson (String s) throws IllegalArgumentException {
    switch (s) {
      case "1": return Person.P1;
      case "2": return Person.P2;
      case "3": return Person.P3;
      default: throw new IllegalArgumentException("Couldn't parse person.");
    }
  }
}
// ---------------------- Перечисляемый тип Tense ----------------------
public enum Tense {
  PERFECT, IMPERFECT, UNKNOWN;
  public static Tense parseTense (String s) throws IllegalArgumentException {
    switch (s) {
      case "IMPF": return Tense.IMPERFECT;
      case "PRF": return Tense.PERFECT;
      case "NA": return Tense.UNKNOWN;
      default: throw new IllegalArgumentException("Couldn't parse tense.");
    }
  }
}
// ---------------------- Перечисляемый тип VerbPreformative ----------------
public enum VerbPreformative {
  T, A, AT, ATTA, NO_PREFORMATIVE, UNKNOWN;
  public static VerbPreformative parseVerbPreformative (String s) throws IllegalArgumentException {
    switch (s) {
      case "T": return VerbPreformative.T;
      case "A": return VerbPreformative.A;
      case "AT": return VerbPreformative.AT;
      case "ATTA": return VerbPreformative.ATTA;
      case "0": return VerbPreformative.NO_PREFORMATIVE;
      default: throw new IllegalArgumentException("Couldn't parse derivational prefix.");
    }
  }
}
// ---------------------- Перечисляемый тип VerbType ----------------------
public enum VerbType {
  A, B, C, D, UNKNOWN;
  public static VerbType parseVerbType (String s) throws IllegalArgumentException {
    switch (s) {
      case "A": return VerbType.A;
      case "B": return VerbType.B;
      case "C": return VerbType.C;
```

```
      case "D": return VerbType.D;
      default: throw new IllegalArgumentException("Couldn't parse verb type.");
    }
  }
}
```

Приложение 2. Файл описания глагольной парадигмы paradigm.txt

```
# MOOD+TENSE+PERSON+GENDER+NUMBER

### Biradicals ###

$radicals:3,type:A,prefix:0

INDIC+PRF+1+C+SG 0+aa_111+ko      0+PRF+1.SG
INDIC+PRF+2+M+SG 0+aa_111+ka      0+PRF+2.M.SG
INDIC+PRF+2+F+SG 0+aa_111+ki      0+PRF+2.F.SG
INDIC+PRF+3+M+SG 0+a0_111+a       0+PRF+3.M.SG
INDIC+PRF+3+F+SG 0+a0_111+at      0+PRF+3.F.SG
INDIC+PRF+1+C+PL 0+aa_111+na      0+PRF+1.PL
INDIC+PRF+2+M+PL 0+aa_111+kum     0+PRF+2.M.PL
INDIC+PRF+2+F+PL 0+aa_111+kn      0+PRF+2.F.PL
INDIC+PRF+3+M+PL 0+a0_111+aw      0+PRF+3.M.PL
INDIC+PRF+3+F+PL 0+a0_111+aya     0+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a0_1211.SG+IMPF
INDIC+IMPF+2+M+SG        t+a0_1212.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+a0_111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a0_1213.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+a0_1213.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+a0_121        1.PL+IMPF
INDIC+IMPF+2+M+PL        t+a0_111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a0_111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a0_111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a0_111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+0a_1111.SG+JUSS
JUSS+NA+2+M+SG   t+0a_1112.M.SG+JUSS
JUSS+NA+2+F+SG   t+0a_111+i       2+JUSS+F.SG
JUSS+NA+3+M+SG   l+0a_1113.M.SG+JUSS
JUSS+NA+3+F+SG   t+0a_1113.F.SG+JUSS
JUSS+NA+1+C+PL   n+0a_1111.PL+JUSS
JUSS+NA+2+M+PL   t+0a_111+o       2+JUSS+M.PL
JUSS+NA+2+F+PL   t+0a_111+a       2+JUSS+F.PL
JUSS+NA+3+M+PL   l+0a_111+o       3+JUSS+M.PL
JUSS+NA+3+F+PL   l+0a_111+a       3+JUSS+F.PL

IMP+NA+2+M+SG    0+0a_1110+IMP.2.M.SG
IMP+NA+2+F+SG    0+0a_111+i       0+IMP+2.F.SG
IMP+NA+2+M+PL    0+0a_111+o       0+IMP+2.M.PL
IMP+NA+2+F+PL    0+0a_111+a       0+IMP+2.F.PL

$radicals:3,type:A,prefix:T

INDIC+PRF+1+C+SG t+aa_121+ko      PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aa_121+ka      PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aa_121+ki      PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aa_121+a       PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aa_121+at      PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aa_121+na      PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aa_121+kum     PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aa_121+kn      PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aa_121+aw      PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aa_121+aya     PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+aa_121
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+aa_121
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+aa_121+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+aa_121
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+aa_121
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        n+t+aa_121
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+aa_121+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+aa_121+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+aa_121+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+aa_121+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+aa_121       1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+aa_121       2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+aa_121+i     2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+aa_121       3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+aa_121       3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   n+t+aa_121       1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+aa_121+o     2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+aa_121+a     2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+aa_121+o     3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+aa_121+a     3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+aa_121PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+aa_121+i       PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+aa_121+o       PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+aa_121+a       PASS+IMP+2.F.PL

$radicals:3,type:A,prefix:A

INDIC+PRF+1+C+SG >a+0a_111+ko     CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+0a_111+ka     CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+0a_111+ki     CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+0a_111+a      CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+0a_111+at     CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+0a_111+na     CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+0a_111+kum    CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+0a_111+kn     CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+0a_111+aw     CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+0a_111+aya    CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a+a0_121
        1.SG+CAUS+IMPF
INDIC+IMPF+2+M+SG        t+a+a0_121
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+a0_111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+a0_121
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+a0_121
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+a+a0_121
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+a0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+a0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+a0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+a0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >a+00_111        CAUS+JUSS.1.SG
JUSS+NA+2+M+SG   t+a+00_111       2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+00_111+i     2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+00_111       3.M.SG+CAUS+JUSS
```

```
JUSS+NA+3+F+SG   t+a+00_111       3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+a+00_111       1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+00_111+o     2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+00_111+a     2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+a+00_111+o     3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+00_111+a     3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+00_111        CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+00_111+i      CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+00_111+o      CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+00_111+a      CAUS+IMP+2.F.PL

$radicals:3,type:A,prefix:AT

INDIC+PRF+1+C+SG >at+aa_111+ko    CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >at+aa_111+ka    CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aa_111+ki    CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aa_111+a     CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+aa_111+at    CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aa_111+na    CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+aa_111+kum   CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+aa_111+kn    CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aa_111+aw    CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aa_111+aya   CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+a0_111
        CAUS+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+at+a0_111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+at+a0_111+i
        2+IMPF+CAUS+F.SG
INDIC+IMPF+3+M+SG        l+at+a0_111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+a0_111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+at+a0_111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+a0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+a0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+a0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+a0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >at+a0_111       CAUS+JUSS.1.SG
JUSS+NA+2+M+SG   t+at+a0_111      2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+a0_111+i    2+JUSS+CAUS+F.SG
JUSS+NA+3+M+SG   l+at+a0_111      3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+a0_111      3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+at+a0_111      1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+a0_111+o    2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+a0_111+a    2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+a0_111+o    3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+a0_111+a    3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >at+a0_111       CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >at+a0_111+i     CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+a0_111+o     CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+a0_111+a     CAUS+IMP+2.F.PL

$radicals:3,type:A,prefix:ATTA

# FCT - factitive
INDIC+PRF+1+C+SG >atta+0a_111+ko  FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+0a_111+ka  FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+0a_111+ki  FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+0a_111+a   FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+0a_111+at  FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+0a_111+na  FCT+PRF+1.C.PL
INDIC+PRF+2+M+PL >atta+0a_111+kum FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+0a_111+kn  FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+0a_111+aw  FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+0a_111+aya FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+00_111
        FCT+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+atta+00_111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+00_111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+atta+00_111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+00_111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+atta+00_111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+00_111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+00_111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+00_111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+00_111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+00_111     FCT+JUSS.1.SG
JUSS+NA+2+M+SG   t+atta+00_111    2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+00_111+i  2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+atta+00_111    3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+00_111    3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   n+atta+00_111    1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+00_111+o  2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+00_111+a  2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+00_111+o  3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+00_111+a  3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+00_111     FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+00_111+i   FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+00_111+o   FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+00_111+a   FCT+IMP+2.F.PL

$radicals:3,type:B,prefix:0

INDIC+PRF+1+C+SG aa_121+ko        PRF+1.SG
INDIC+PRF+2+M+SG aa_121+ka        PRF+2.M.SG
INDIC+PRF+2+F+SG aa_121+ki        PRF+2.F.SG
INDIC+PRF+3+M+SG aa_121+a PRF+3.M.SG
INDIC+PRF+3+F+SG aa_121+at        PRF+3.F.SG
INDIC+PRF+1+C+PL aa_121+na        PRF+1.PL
INDIC+PRF+2+M+PL aa_121+kum       PRF+2.M.PL
INDIC+PRF+2+F+PL aa_121+kn        PRF+2.F.PL
INDIC+PRF+3+M+PL aa_121+aw        PRF+3.M.PL
INDIC+PRF+3+F+PL aa_121+aya       PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a0_1211.SG+IMPF
INDIC+IMPF+2+M+SG        t+a0_1212.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+a0_111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a0_1213.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+a0_1213.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+a0_121       1.PL+IMPF
INDIC+IMPF+2+M+PL        t+a0_111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a0_111+a
        2+IMPF+F.PL
```

```
INDIC+IMPF+3+M+PL        l+a0_111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a0_111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+a0_1211.SG+JUSS
JUSS+NA+2+M+SG   t+a0_1212.M.SG+JUSS
JUSS+NA+2+F+SG   t+a0_111+i      2+JUSS+F.SG
JUSS+NA+3+M+SG   l+a0_1213.M.SG+JUSS
JUSS+NA+3+F+SG   t+a0_1213.F.SG+JUSS
JUSS+NA+1+C+PL   n+a0_1211.PL+JUSS
JUSS+NA+2+M+PL   t+a0_111+o      2+JUSS+M.PL
JUSS+NA+2+F+PL   t+a0_111+a      2+JUSS+F.PL
JUSS+NA+3+M+PL   l+a0_111+o      3+JUSS+M.PL
JUSS+NA+3+F+PL   l+a0_111+a      3+JUSS+F.PL

IMP+NA+2+M+SG    a0_121  IMP.2.M.SG
IMP+NA+2+F+SG    a0_111+iIMP+2.F.SG
IMP+NA+2+M+PL    a0_111+oIMP+2.M.PL
IMP+NA+2+F+PL    a0_111+aIMP+2.F.PL

$radicals:3,type:B,prefix:T

INDIC+PRF+1+C+SG t+aa_121+ko     PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aa_121+ka     PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aa_121+ki     PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aa_121+a      PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aa_121+at     PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aa_121+na     PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aa_121+kum    PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aa_121+kn     PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aa_121+aw     PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aa_121+aya    PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+aa_121
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+aa_121
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+aa_121+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+aa_121
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+aa_121
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        n+t+aa_121
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+aa_121+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+aa_121+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+aa_121+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+aa_121+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+aa_121      1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+aa_121      2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+aa_121+i    2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+aa_121      3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+aa_121      3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   n+t+aa_121      1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+aa_121+o    2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+aa_121+a    2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+aa_121+o    3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+aa_121+a    3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+aa_121PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+aa_121+i      PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+aa_121+o      PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+aa_121+a      PASS+IMP+2.F.PL

$radicals:3,type:B,prefix:A

INDIC+PRF+1+C+SG >a+aa_121+ko    CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+aa_121+ka    CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+aa_121+ki    CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+aa_121+a     CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+aa_121+at    CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+aa_121+na    CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+aa_121+kum   CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+aa_121+kn    CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+aa_121+aw    CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+aa_121+aya   CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a+a0_121
        1.SG+CAUS+IMPF
INDIC+IMPF+2+M+SG        t+a+a0_121
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+a0_111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+a0_121
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+a0_121
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+a+a0_121
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+a0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+a0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+a0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+a0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >+a+a0_121      1.SG+CAUS+JUSS
JUSS+NA+2+M+SG   t+a+a0_121      2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+a0_111+i    2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+a0_121      3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+a+a0_121      3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+a+a0_121      1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+a0_111+o    2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+a0_111+a    2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+a+a0_111+o    3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+a0_111+a    3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+a0_121       CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+a0_111+i     CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+a0_111+o     CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+a0_111+a     CAUS+IMP+2.F.PL

$radicals:3,type:B,prefix:AT

INDIC+PRF+1+C+SG >at+aa_121+ko   CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >at+aa_121+ka   CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aa_121+ki   CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aa_121+a    CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+aa_121+at   CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aa_121+na   CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+aa_121+kum  CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+aa_121+kn   CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aa_121+aw   CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aa_121+aya  CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+a0_121
        CAUS+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+at+a0_121
        2.M.SG+CAUS+IMPF
```

```
INDIC+IMPF+2+F+SG        t+at+a0_111+i
        2+IMPF+CAUS+F.SG
INDIC+IMPF+3+M+SG        l+at+a0_121
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+a0_121
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+at+a0_121
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+a0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+a0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+a0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+a0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >at+a0_121       CAUS+JUSS.1.SG
JUSS+NA+2+M+SG   t+at+a0_121      2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+a0_111+i    2+JUSS+CAUS+F.SG
JUSS+NA+3+M+SG   l+at+a0_121      3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+a0_121      3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+at+a0_121      1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+a0_111+o    2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+a0_111+a    2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+a0_111+o    3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+a0_111+a    3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >at+a0_121       CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >at+a0_111+i     CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+a0_111+o     CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+a0_111+a     CAUS+IMP+2.F.PL

$radicals:3,type:B,prefix:ATTA

# FCT - factitive
INDIC+PRF+1+C+SG >atta+aa_121+ko  FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+aa_121+ka  FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+aa_121+ki  FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+aa_121+a   FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+aa_121+atta
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+aa_121+na  FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+aa_121+kum FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+aa_121+kn  FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+aa_121+aw  FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+aa_121+aya FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+a0_121
        FCT+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+atta+a0_121
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+a0_111+i
        2+IMPF+FCT+F.SG
INDIC+IMPF+3+M+SG        l+atta+a0_121
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+a0_121
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        n+atta+a0_121
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+a0_111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+a0_111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+a0_111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+a0_111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+a0_121     FCT+JUSS.1.SG
JUSS+NA+2+M+SG   t+atta+a0_121    2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+a0_111+i  2+JUSS+FCT+F.SG
JUSS+NA+3+M+SG   l+atta+a0_121    3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+a0_121    3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   n+atta+a0_121    1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+a0_111+o  2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+a0_111+a  2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+a0_111+o  3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+a0_111+a  3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+a0_121     FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+a0_111+i   FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+a0_111+o   FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+a0_111+a   FCT+IMP+2.F.PL

$radicals:3,type:C,prefix:0

INDIC+PRF+1+C+SG Aa_111+ko        PRF+1.SG
INDIC+PRF+2+M+SG Aa_111+ka        PRF+2.M.SG
INDIC+PRF+2+F+SG Aa_111+ki        PRF+2.F.SG
INDIC+PRF+3+M+SG Aa_111+a PRF+3.M.SG
INDIC+PRF+3+F+SG Aa_111+at        PRF+3.F.SG
INDIC+PRF+1+C+PL Aa_111+na        PRF+1.PL
INDIC+PRF+2+M+PL Aa_111+kum       PRF+2.M.PL
INDIC+PRF+2+F+PL Aa_111+kn        PRF+2.F.PL
INDIC+PRF+3+M+PL Aa_111+aw        PRF+3.M.PL
INDIC+PRF+3+F+PL Aa_111+aya       PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+A0_1111.SG+IMPF
INDIC+IMPF+2+M+SG        t+A0_1112.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+A0_111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+A0_1113.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+A0_1113.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+A0_111       1.PL+IMPF
INDIC+IMPF+2+M+PL        t+A0_111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+A0_111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+A0_111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+A0_111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+A0_1111.SG+JUSS
JUSS+NA+2+M+SG   t+A0_1112.M.SG+JUSS
JUSS+NA+2+F+SG   t+A0_111+i       2+JUSS+F.SG
JUSS+NA+3+M+SG   l+A0_1113.M.SG+JUSS
JUSS+NA+3+F+SG   t+A0_1113.F.SG+JUSS
JUSS+NA+1+C+PL   n+A0_1111.PL+JUSS
JUSS+NA+2+M+PL   t+A0_111+o       2+JUSS+M.PL
JUSS+NA+2+F+PL   t+A0_111+a       2+JUSS+F.PL
JUSS+NA+3+M+PL   l+A0_111+o       3+JUSS+M.PL
JUSS+NA+3+F+PL   l+A0_111+a       3+JUSS+F.PL

IMP+NA+2+M+SG    A0_111   IMP.2.M.SG
IMP+NA+2+F+SG    A0_111+i IMP+2.F.SG
IMP+NA+2+M+PL    A0_111+o IMP+2.M.PL
IMP+NA+2+F+PL    A0_111+a IMP+2.F.PL

$radicals:3,type:C,prefix:T

INDIC+PRF+1+C+SG t+Aa_111+ko      PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+Aa_111+ka      PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+Aa_111+ki      PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+Aa_111+a       PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+Aa_111+at      PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+Aa_111+na      PASS+PRF+1.PL
```

```
INDIC+PRF+2+M+PL t+Aa_111+kum     PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+Aa_111+kn      PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+Aa_111+aw      PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+Aa_111+aya     PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+Aa_111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+Aa_111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+Aa_111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+Aa_111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+Aa_111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        n+t+Aa_111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+Aa_111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+Aa_111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+Aa_111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+Aa_111+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+Aa_111       1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+Aa_111       2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+Aa_111+i     2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+Aa_111       3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+Aa_111       3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   n+t+Aa_111       1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+Aa_111+o     2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+Aa_111+a     2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+Aa_111+o     3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+Aa_111+a     3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+Aa_111PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+Aa_111+i       PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+Aa_111+o       PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+Aa_111+a       PASS+IMP+2.F.PL

$radicals:3,type:C,prefix:A

INDIC+PRF+1+C+SG >a+Aa_111+ko     CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+Aa_111+ka     CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+Aa_111+ki     CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+Aa_111+a      CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+Aa_111+at     CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+Aa_111+na     CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+Aa_111+kum    CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+Aa_111+kn     CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+Aa_111+aw     CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+Aa_111+aya    CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a+A0_111
        1.SG+CAUS+IMPF
INDIC+IMPF+2+M+SG        t+a+A0_111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+A0_111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+A0_111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+A0_111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+a+A0_111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+A0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+A0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+A0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+A0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >+a+A0_111       1.SG+CAUS+JUSS
JUSS+NA+2+M+SG   t+a+A0_111       2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+A0_111+i     2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+A0_111       3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+a+A0_111       3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+a+A0_111       1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+A0_111+o     2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+A0_111+a     2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+a+A0_111+o     3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+A0_111+a     3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+A0_111        CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+A0_111+i      CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+A0_111+o      CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+A0_111+a      CAUS+IMP+2.F.PL

$radicals:3,type:C,prefix:AT

INDIC+PRF+1+C+SG >at+Aa_111+ko    CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >at+Aa_111+ka    CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+Aa_111+ki    CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+Aa_111+a     CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+Aa_111+at    CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+Aa_111+na    CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+Aa_111+kum   CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+Aa_111+kn    CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+Aa_111+aw    CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+Aa_111+aya   CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+A0_111
        CAUS+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+at+A0_111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+at+A0_111+i
        2+IMPF+CAUS+F.SG
INDIC+IMPF+3+M+SG        l+at+A0_111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+A0_111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+at+A0_111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+A0_111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+A0_111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+A0_111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+A0_111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >at+A0_111       CAUS+JUSS.1.SG
JUSS+NA+2+M+SG   t+at+A0_111      2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+A0_111+i    2+JUSS+CAUS+F.SG
JUSS+NA+3+M+SG   l+at+A0_111      3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+A0_111      3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+at+A0_111      1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+A0_111+o    2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+A0_111+a    2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+A0_111+o    3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+A0_111+a    3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >at+A0_111       CAUS+IMP.2.M.SG
```

```
IMP+NA+2+F+SG    >at+A0_111+i     CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+A0_111+o     CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+A0_111+a     CAUS+IMP+2.F.PL

$radicals:3,type:C,prefix:ATTA

# FCT - factitive
INDIC+PRF+1+C+SG >atta+Aa_111+ko  FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+Aa_111+ka  FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+Aa_111+ki  FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+Aa_111+a   FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+Aa_111+atta
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+Aa_111+na  FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+Aa_111+kum FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+Aa_111+kn  FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+Aa_111+aw  FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+Aa_111+aya FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+A0_111
        FCT+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+atta+A0_111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+A0_111+i
        2+IMPF+FCT+F.SG
INDIC+IMPF+3+M+SG        l+atta+A0_111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+A0_111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        n+atta+A0_111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+A0_111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+A0_111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+A0_111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+A0_111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+A0_111     FCT+JUSS.1.SG
JUSS+NA+2+M+SG   t+atta+A0_111    2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+A0_111+i  2+JUSS+FCT+F.SG
JUSS+NA+3+M+SG   l+atta+A0_111    3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+A0_111    3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   n+atta+A0_111    1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+A0_111+o  2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+A0_111+a  2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+A0_111+o  3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+A0_111+a  3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+A0_111     FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+A0_111+i   FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+A0_111+o   FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+A0_111+a   FCT+IMP+2.F.PL

$radicals:4,type:D,prefix:0

INDIC+PRF+1+C+SG aAa_1111+ko      PRF+1.SG
INDIC+PRF+2+M+SG aAa_1111+ka      PRF+2.M.SG
INDIC+PRF+2+F+SG aAa_1111+ki      PRF+2.F.SG
INDIC+PRF+3+M+SG aAa_1111+a       PRF+3.M.SG
INDIC+PRF+3+F+SG aAa_1111+at      PRF+3.F.SG
INDIC+PRF+1+C+PL aAa_1111+na      PRF+1.PL
INDIC+PRF+2+M+PL aAa_1111+kum     PRF+2.M.PL
INDIC+PRF+2+F+PL aAa_1111+kn      PRF+2.F.PL
INDIC+PRF+3+M+PL aAa_1111+aw      PRF+3.M.PL
INDIC+PRF+3+F+PL aAa_1111+aya     PRF+3.F.PL
```

```
INDIC+IMPF+1+C+SG        >+aA0_1111       1.SG+IMPF
INDIC+IMPF+2+M+SG        t+aA0_1111
        2.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+aA0_1111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+aA0_1111
        3.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+aA0_1111
        3.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+aA0_1111      1.PL+IMPF
INDIC+IMPF+2+M+PL        t+aA0_1111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+aA0_1111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+aA0_1111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+aA0_1111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+aA0_1111       1.SG+JUSS
JUSS+NA+2+M+SG   t+aA0_1111       2.M.SG+JUSS
JUSS+NA+2+F+SG   t+aA0_1111+i     2+JUSS+F.SG
JUSS+NA+3+M+SG   l+aA0_1111       3.M.SG+JUSS
JUSS+NA+3+F+SG   t+aA0_1111       3.F.SG+JUSS
JUSS+NA+1+C+PL   n+aA0_1111       1.PL+JUSS
JUSS+NA+2+M+PL   t+aA0_1111+o     2+JUSS+M.PL
JUSS+NA+2+F+PL   t+aA0_1111+a     2+JUSS+F.PL
JUSS+NA+3+M+PL   l+aA0_1111+o     3+JUSS+M.PL
JUSS+NA+3+F+PL   l+aA0_1111+a     3+JUSS+F.PL

IMP+NA+2+M+SG    aA0_1111IMP.2.M.SG
IMP+NA+2+F+SG    aA0_1111+i       IMP+2.F.SG
IMP+NA+2+M+PL    aA0_1111+o       IMP+2.M.PL
IMP+NA+2+F+PL    aA0_1111+a       IMP+2.F.PL

$radicals:4,type:D,prefix:T

INDIC+PRF+1+C+SG t+aAa_1111+ko    PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aAa_1111+ka    PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aAa_1111+ki    PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aAa_1111+a     PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aAa_1111+at    PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aAa_1111+na    PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aAa_1111+kum   PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aAa_1111+kn    PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aAa_1111+aw    PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aAa_1111+aya   PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+aAa_1111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+aAa_1111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+aAa_1111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+aAa_1111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+aAa_1111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        n+t+aAa_1111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+aAa_1111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+aAa_1111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+aAa_1111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+aAa_1111+a
        3+PASS+IMPF+F.PL
```

```
JUSS+NA+1+C+SG   >+t+aAa_1111     1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+aAa_1111     2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+aAa_1111+i   2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+aAa_1111     3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+aAa_1111     3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   n+t+aAa_1111     1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+aAa_1111+o   2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+aAa_1111+a   2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+aAa_1111+o   3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+aAa_1111+a   3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+aAa_1111       PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+aAa_1111+i     PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+aAa_1111+o     PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+aAa_1111+a     PASS+IMP+2.F.PL

# no a-D derivatives

$radicals:4,type:D,prefix:AT

INDIC+PRF+1+C+SG >at+aAa_1111+ko  CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >at+aAa_1111+ka  CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aAa_1111+ki  CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aAa_1111+a   CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+aAa_1111+at  CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aAa_1111+na  CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+aAa_1111+kum CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+aAa_1111+kn  CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aAa_1111+aw  CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aAa_1111+aya CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+aA0_1111
        CAUS+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+at+aA0_1111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+at+aA0_1111+i
        2+IMPF+CAUS+F.SG
INDIC+IMPF+3+M+SG        l+at+aA0_1111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+aA0_1111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        n+at+aA0_1111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+aA0_1111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+aA0_1111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+aA0_1111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+aA0_1111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >at+aA0_1111     CAUS+JUSS.1.SG
JUSS+NA+2+M+SG   t+at+aA0_1111    2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+aA0_1111+i  2+JUSS+CAUS+F.SG
JUSS+NA+3+M+SG   l+at+aA0_1111    3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+aA0_1111    3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   n+at+aA0_1111    1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+aA0_1111+o  2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+aA0_1111+a  2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+aA0_1111+o  3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+aA0_1111+a  3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >at+aA0_1111     CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >at+aA0_1111+i   CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+aA0_1111+o   CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+aA0_1111+a   CAUS+IMP+2.F.PL

$radicals:4,type:D,prefix:ATTA
```

```
# FCT - factitive
INDIC+PRF+1+C+SG >atta+aAa_1111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+aAa_1111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+aAa_1111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+aAa_1111+a FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+aAa_1111+atta
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+aAa_1111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+aAa_1111+kum
        FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+aAa_1111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+aAa_1111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+aAa_1111+aya
        FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+aA0_1111
        FCT+IMPF.1.SG
INDIC+IMPF+2+M+SG        t+atta+aA0_1111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+aA0_1111+i
        2+IMPF+FCT+F.SG
INDIC+IMPF+3+M+SG        l+atta+aA0_1111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+aA0_1111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        n+atta+aA0_1111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+aA0_1111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+aA0_1111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+aA0_1111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+aA0_1111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+aA0_1111   FCT+JUSS.1.SG
JUSS+NA+2+M+SG   t+atta+aA0_1111  2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+aA0_1111+i
        2+JUSS+FCT+F.SG
JUSS+NA+3+M+SG   l+atta+aA0_1111  3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+aA0_1111  3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   n+atta+aA0_1111  1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+aA0_1111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+aA0_1111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+aA0_1111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+aA0_1111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+aA0_1111   FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+aA0_1111+i FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+aA0_1111+o FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+aA0_1111+a FCT+IMP+2.F.PL

### Quadriradicals ###

$radicals:4,type:A,prefix:0

INDIC+PRF+1+C+SG 0+a0a_1111+ko    0+PRF+1.SG
```

```
INDIC+PRF+2+M+SG 0+a0a_1111+ka    0+PRF+2.M.SG
INDIC+PRF+2+F+SG 0+a0a_1111+ki    0+PRF+2.F.SG
INDIC+PRF+3+M+SG 0+a0a_1111+a     0+PRF+3.M.SG
INDIC+PRF+3+F+SG 0+a0a_1111+at    0+PRF+3.F.SG
INDIC+PRF+1+C+PL 0+a0a_1111+na    0+PRF+1.PL
INDIC+PRF+2+M+PL 0+a0a_1111+kum   0+PRF+2.M.PL
INDIC+PRF+2+F+PL 0+a0a_1111+kn    0+PRF+2.F.PL
INDIC+PRF+3+M+PL 0+a0a_1111+aw    0+PRF+3.M.PL
INDIC+PRF+3+F+PL 0+a0a_1111+aya   0+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+a00_1111       1.SG+IMPF
INDIC+IMPF+2+M+SG        t+a00_1111
        2.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+a00_1111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a00_1111
        3.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+a00_1111
        3.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+a00_1111      1.PL+IMPF
INDIC+IMPF+2+M+PL        t+a00_1111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a00_1111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a00_1111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a00_1111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+a00_1111       1.SG+JUSS
JUSS+NA+2+M+SG   t+a00_1111       2.M.SG+JUSS
JUSS+NA+2+F+SG   t+a00_1111+i     2+JUSS+F.SG
JUSS+NA+3+M+SG   l+a00_1111       3.M.SG+JUSS
JUSS+NA+3+F+SG   t+a00_1111       3.F.SG+JUSS
JUSS+NA+1+C+PL   n+a00_1111       1.PL+JUSS
JUSS+NA+2+M+PL   t+a00_1111+o     2+JUSS+M.PL
JUSS+NA+2+F+PL   t+a00_1111+a     2+JUSS+F.PL
JUSS+NA+3+M+PL   l+a00_1111+o     3+JUSS+M.PL
JUSS+NA+3+F+PL   l+a00_1111+a     3+JUSS+F.PL

IMP+NA+2+M+SG    0+a00_1111       0+IMP.2.M.SG
IMP+NA+2+F+SG    0+a00_1111+i     0+IMP+2.F.SG
IMP+NA+2+M+PL    0+a00_1111+o     0+IMP+2.M.PL
IMP+NA+2+F+PL    0+a00_1111+a     0+IMP+2.F.PL

$radicals:4,type:A,prefix:T

INDIC+PRF+1+C+SG t+a0a_1111+ko    PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+a0a_1111+ka    PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+a0a_1111+ki    PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+a0a_1111+a     PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+a0a_1111+at    PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+a0a_1111+na    PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+a0a_1111+kum   PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+a0a_1111+kn    PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+a0a_1111+aw    PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+a0a_1111+aya   PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+a0a_1111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+a0a_1111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+a0a_1111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+a0a_1111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+a0a_1111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        >n+t+a0a_1111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+a0a_1111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+a0a_1111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+a0a_1111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+a0a_1111+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+a0a_1111     1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+a0a_1111     2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+a0a_1111+i   2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+a0a_1111     3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+a0a_1111     3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   >n+t+a0a_1111    1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+a0a_1111+o   2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+a0a_1111+a   2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+a0a_1111+o   3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+a0a_1111+a   3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+a0a_1111       PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+a0a_1111+i     PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+a0a_1111+o     PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+a0a_1111+a     PASS+IMP+2.F.PL

$radicals:4,type:A,prefix:A

INDIC+PRF+1+C+SG >a+a0a_1111+ko   CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+a0a_1111+ka   CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+a0a_1111+ki   CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+a0a_1111+a    CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+a0a_1111+at   CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+a0a_1111+na   CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+a0a_1111+kum  CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+a0a_1111+kn   CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+a0a_1111+aw   CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+a0a_1111+aya  CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >a+a00_1111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+a+a00_1111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+a00_1111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+a00_1111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+a00_1111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+a+a00_1111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+a00_1111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+a00_1111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+a00_1111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+a00_1111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >a+a00_1111      CAUS+1.SG+JUSS
JUSS+NA+2+M+SG   t+a+a00_1111     2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+a00_1111+i   2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+a00_1111     3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+a+a00_1111     3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   >n+a+a00_1111    1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+a00_1111+o   2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+a00_1111+a   2+CAUS+JUSS+F.PL
```

```
JUSS+NA+3+M+PL   l+a+a00_1111+o   3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+a00_1111+a   3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+a00_1111      CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+a00_1111+i    CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+a00_1111+o    CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+a00_1111+a    CAUS+IMP+2.F.PL

$radicals:4,type:A,prefix:AT

INDIC+PRF+1+C+SG >at+a0a_1111+ko  FCT+PRF+1.SG
INDIC+PRF+2+M+SG >at+a0a_1111+ka  FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+a0a_1111+ki  FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+a0a_1111+a   FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+a0a_1111+at  FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+a0a_1111+na  FCT+PRF+1.PL
INDIC+PRF+2+M+PL >at+a0a_1111+kum FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+a0a_1111+kn  FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+a0a_1111+aw  FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+a0a_1111+aya FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+a00_1111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+at+a00_1111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+at+a00_1111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+at+a00_1111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+at+a00_1111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+at+a00_1111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+at+a00_1111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+a00_1111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+a00_1111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+a00_1111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >at+a00_1111     FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+at+a00_1111    2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+at+a00_1111+i  2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+at+a00_1111    3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+at+a00_1111    3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+at+a00_1111   1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+at+a00_1111+o  2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+a00_1111+a  2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+a00_1111+o  3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+a00_1111+a  3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >at+a00_1111     FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >at+a00_1111+i   FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >at+a00_1111+o   FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >at+a00_1111+a   FCT+IMP+2.F.PL

$radicals:4,type:A,prefix:ATTA

INDIC+PRF+1+C+SG >atta+a0a_1111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+a0a_1111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+a0a_1111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+a0a_1111+a FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+a0a_1111+at
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+a0a_1111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+a0a_1111+kum
        FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+a0a_1111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+a0a_1111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+a0a_1111+aya
        FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+a00_1111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+atta+a00_1111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+a00_1111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+atta+a00_1111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+a00_1111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+atta+a00_1111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+a00_1111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+a00_1111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+a00_1111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+a00_1111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+a00_1111   FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+atta+a00_1111  2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+a00_1111+i
        2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+atta+a00_1111  3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+a00_1111  3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+atta+a00_1111 1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+a00_1111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+a00_1111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+a00_1111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+a00_1111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+a00_1111   FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+a00_1111+i FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+a00_1111+o FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+a00_1111+a FCT+IMP+2.F.PL

$radicals:4,type:C,prefix:0

INDIC+PRF+1+C+SG 0+aAa_1111+ko   0+PRF+1.SG
INDIC+PRF+2+M+SG 0+aAa_1111+ka   0+PRF+2.M.SG
INDIC+PRF+2+F+SG 0+aAa_1111+ki   0+PRF+2.F.SG
INDIC+PRF+3+M+SG 0+aAa_1111+a    0+PRF+3.M.SG
INDIC+PRF+3+F+SG 0+aAa_1111+at   0+PRF+3.F.SG
INDIC+PRF+1+C+PL 0+aAa_1111+na   0+PRF+1.PL
INDIC+PRF+2+M+PL 0+aAa_1111+kum  0+PRF+2.M.PL
INDIC+PRF+2+F+PL 0+aAa_1111+kn   0+PRF+2.F.PL
INDIC+PRF+3+M+PL 0+aAa_1111+aw   0+PRF+3.M.PL
INDIC+PRF+3+F+PL 0+aAa_1111+aya  0+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+aA0_1111      1.SG+IMPF
INDIC+IMPF+2+M+SG        t+aA0_1111
        2.M.SG+IMPF
```

```
INDIC+IMPF+2+F+SG        t+aA0_1111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+aA0_1111
        3.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+aA0_1111
        3.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+aA0_1111      1.PL+IMPF
INDIC+IMPF+2+M+PL        t+aA0_1111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+aA0_1111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+aA0_1111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+aA0_1111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+aA0_1111       1.SG+JUSS
JUSS+NA+2+M+SG   t+aA0_1111       2.M.SG+JUSS
JUSS+NA+2+F+SG   t+aA0_1111+i     2+JUSS+F.SG
JUSS+NA+3+M+SG   l+aA0_1111       3.M.SG+JUSS
JUSS+NA+3+F+SG   t+aA0_1111       3.F.SG+JUSS
JUSS+NA+1+C+PL   n+aA0_1111       1.PL+JUSS
JUSS+NA+2+M+PL   t+aA0_1111+o     2+JUSS+M.PL
JUSS+NA+2+F+PL   t+aA0_1111+a     2+JUSS+F.PL
JUSS+NA+3+M+PL   l+aA0_1111+o     3+JUSS+M.PL
JUSS+NA+3+F+PL   l+aA0_1111+a     3+JUSS+F.PL

IMP+NA+2+M+SG    0+aA0_1111       0+IMP.2.M.SG
IMP+NA+2+F+SG    0+aA0_1111+i     0+IMP+2.F.SG
IMP+NA+2+M+PL    0+aA0_1111+o     0+IMP+2.M.PL
IMP+NA+2+F+PL    0+aA0_1111+a     0+IMP+2.F.PL

$radicals:4,type:C,prefix:T

INDIC+PRF+1+C+SG t+aAa_1111+ko    PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aAa_1111+ka    PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aAa_1111+ki    PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aAa_1111+a     PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aAa_1111+at    PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aAa_1111+na    PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aAa_1111+kum   PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aAa_1111+kn    PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aAa_1111+aw    PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aAa_1111+aya   PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+aAa_1111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+aAa_1111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+aAa_1111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+aAa_1111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+aAa_1111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        >n+t+aAa_1111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+aAa_1111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+aAa_1111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+aAa_1111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+aAa_1111+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+aAa_1111     1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+aAa_1111     2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+aAa_1111+i   2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+aAa_1111     3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+aAa_1111     3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   >n+t+aAa_1111    1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+aAa_1111+o   2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+aAa_1111+a   2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+aAa_1111+o   3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+aAa_1111+a   3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+aAa_1111       PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+aAa_1111+i     PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+aAa_1111+o     PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+aAa_1111+a     PASS+IMP+2.F.PL

$radicals:4,type:C,prefix:A

INDIC+PRF+1+C+SG >a+aAa_1111+ko   CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+aAa_1111+ka   CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+aAa_1111+ki   CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+aAa_1111+a    CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+aAa_1111+at   CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+aAa_1111+na   CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+aAa_1111+kum  CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+aAa_1111+kn   CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+aAa_1111+aw   CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+aAa_1111+aya  CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >a+aA0_1111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+a+aA0_1111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+aA0_1111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+aA0_1111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+aA0_1111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+a+aA0_1111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+aA0_1111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+aA0_1111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+aA0_1111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+aA0_1111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >a+aA0_1111      CAUS+1.SG+JUSS
JUSS+NA+2+M+SG   t+a+aA0_1111     2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+aA0_1111+i   2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+aA0_1111     3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+a+aA0_1111     3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   >n+a+aA0_1111    1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+aA0_1111+o   2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+aA0_1111+a   2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+a+aA0_1111+o   3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+aA0_1111+a   3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+aA0_1111      CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+aA0_1111+i    CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+aA0_1111+o    CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+aA0_1111+a    CAUS+IMP+2.F.PL

$radicals:4,type:C,prefix:AT

INDIC+PRF+1+C+SG >at+aAa_1111+ko  CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >at+aAa_1111+ka  CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aAa_1111+ki  CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aAa_1111+a   CAUS+PRF+3.M.SG
```

```
INDIC+PRF+3+F+SG >at+aAa_1111+at  CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aAa_1111+na  CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+aAa_1111+kum CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+aAa_1111+kn  CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aAa_1111+aw  CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aAa_1111+aya CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+aA0_1111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+at+aA0_1111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+at+aA0_1111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+at+aA0_1111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+aA0_1111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+at+aA0_1111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+aA0_1111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+aA0_1111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+aA0_1111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+aA0_1111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >at+aA0_1111    CAUS+1.SG+JUSS
JUSS+NA+2+M+SG   t+at+aA0_1111   2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+aA0_1111+i 2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+at+aA0_1111   3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+aA0_1111   3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   >n+at+aA0_1111  1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+aA0_1111+o 2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+aA0_1111+a 2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+aA0_1111+o 3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+aA0_1111+a 3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >at+aA0_1111    CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >at+aA0_1111+i  CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+aA0_1111+o  CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+aA0_1111+a  CAUS+IMP+2.F.PL

$radicals:4,type:C,prefix:ATTA

INDIC+PRF+1+C+SG >atta+aAa_1111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+aAa_1111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+aAa_1111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+aAa_1111+a FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+aAa_1111+at
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+aAa_1111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+aAa_1111+kum
        FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+aAa_1111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+aAa_1111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+aAa_1111+aya
        FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+aA0_1111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+atta+aA0_1111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+aA0_1111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+atta+aA0_1111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+aA0_1111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+atta+aA0_1111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+aA0_1111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+aA0_1111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+aA0_1111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+aA0_1111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+aA0_1111   FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+atta+aA0_1111  2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+aA0_1111+i
        2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+atta+aA0_1111  3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+aA0_1111  3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+atta+aA0_1111 1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+aA0_1111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+aA0_1111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+aA0_1111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+aA0_1111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+aA0_1111   FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+aA0_1111+i FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+aA0_1111+o FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+aA0_1111+a FCT+IMP+2.F.PL

### Quinqueradicals ###

$radicals:5,type:A,prefix:0

INDIC+PRF+1+C+SG 0+aa0a_11111+ko  0+PRF+1.SG
INDIC+PRF+2+M+SG 0+aa0a_11111+ka  0+PRF+2.M.SG
INDIC+PRF+2+F+SG 0+aa0a_11111+ki  0+PRF+2.F.SG
INDIC+PRF+3+M+SG 0+aa0a_11111+a   0+PRF+3.M.SG
INDIC+PRF+3+F+SG 0+aa0a_11111+at  0+PRF+3.F.SG
INDIC+PRF+1+C+PL 0+aa0a_11111+na  0+PRF+1.PL
INDIC+PRF+2+M+PL 0+aa0a_11111+kum 0+PRF+2.M.PL
INDIC+PRF+2+F+PL 0+aa0a_11111+kn  0+PRF+2.F.PL
INDIC+PRF+3+M+PL 0+aa0a_11111+aw  0+PRF+3.M.PL
INDIC+PRF+3+F+PL 0+aa0a_11111+aya 0+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+aa00_11111    1.SG+IMPF
INDIC+IMPF+2+M+SG        t+aa00_11111
        2.M.SG+IMPF
INDIC+IMPF+2+F+SG        t+aa00_11111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+aa00_11111
        3.M.SG+IMPF
INDIC+IMPF+3+F+SG        t+aa00_11111
        3.F.SG+IMPF
INDIC+IMPF+1+C+PL        >n+aa00_11111   1.PL+IMPF
INDIC+IMPF+2+M+PL        t+aa00_11111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+aa00_11111+a
        2+IMPF+F.PL
```

```
INDIC+IMPF+3+M+PL        l+aa00_11111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+aa00_11111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG   >+aa00_11111     1.SG+JUSS
JUSS+NA+2+M+SG   t+aa00_11111     2.M.SG+JUSS
JUSS+NA+2+F+SG   t+aa00_11111+i   2+JUSS+F.SG
JUSS+NA+3+M+SG   l+aa00_11111     3.M.SG+JUSS
JUSS+NA+3+F+SG   t+aa00_11111     3.F.SG+JUSS
JUSS+NA+1+C+PL   n+aa00_11111     1.PL+JUSS
JUSS+NA+2+M+PL   t+aa00_11111+o   2+JUSS+M.PL
JUSS+NA+2+F+PL   t+aa00_11111+a   2+JUSS+F.PL
JUSS+NA+3+M+PL   l+aa00_11111+o   3+JUSS+M.PL
JUSS+NA+3+F+PL   l+aa00_11111+a   3+JUSS+F.PL

IMP+NA+2+M+SG    0+aa00_11111     0+IMP.2.M.SG
IMP+NA+2+F+SG    0+aa00_11111+i   0+IMP+2.F.SG
IMP+NA+2+M+PL    0+aa00_11111+o   0+IMP+2.M.PL
IMP+NA+2+F+PL    0+aa00_11111+a   0+IMP+2.F.PL

$radicals:5,type:A,prefix:T

INDIC+PRF+1+C+SG t+aa0a_11111+ko  PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aa0a_11111+ka  PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aa0a_11111+ki  PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aa0a_11111+a   PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aa0a_11111+at  PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aa0a_11111+na  PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aa0a_11111+kum PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aa0a_11111+kn  PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aa0a_11111+aw  PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aa0a_11111+aya PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+t+aa0a_11111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG        t+t+aa0a_11111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG        t+t+aa0a_11111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+t+aa0a_11111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG        t+t+aa0a_11111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL        >n+t+aa0a_11111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL        t+t+aa0a_11111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+t+aa0a_11111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+t+aa0a_11111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+t+aa0a_11111+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG   >+t+aa0a_11111   1.SG+PASS+JUSS
JUSS+NA+2+M+SG   t+t+aa0a_11111   2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG   t+t+aa0a_11111+i 2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG   l+t+aa0a_11111   3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG   t+t+aa0a_11111   3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL   >n+t+aa0a_11111  1.PL+PASS+JUSS
JUSS+NA+2+M+PL   t+t+aa0a_11111+o 2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL   t+t+aa0a_11111+a 2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL   l+t+aa0a_11111+o 3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL   l+t+aa0a_11111+a 3+PASS+JUSS+F.PL

IMP+NA+2+M+SG    t+aa0a_11111     PASS+IMP.2.M.SG
IMP+NA+2+F+SG    t+aa0a_11111+i   PASS+IMP+2.F.SG
IMP+NA+2+M+PL    t+aa0a_11111+o   PASS+IMP+2.M.PL
IMP+NA+2+F+PL    t+aa0a_11111+a   PASS+IMP+2.F.PL

$radicals:5,type:A,prefix:A

INDIC+PRF+1+C+SG >a+aa0a_11111+ko CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+aa0a_11111+ka CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+aa0a_11111+ki CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+aa0a_11111+a  CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+aa0a_11111+at CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+aa0a_11111+na CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+aa0a_11111+kum
        CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+aa0a_11111+kn CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+aa0a_11111+aw CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+aa0a_11111+aya
        CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >a+aa00_11111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+a+aa00_11111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+a+aa00_11111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+a+aa00_11111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+a+aa00_11111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+a+aa00_11111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+a+aa00_11111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+a+aa00_11111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+a+aa00_11111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+a+aa00_11111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG   >a+aa00_11111    CAUS+1.SG+JUSS
JUSS+NA+2+M+SG   t+a+aa00_11111   2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+a+aa00_11111+i 2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+a+aa00_11111   3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+a+aa00_11111   3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   >n+a+aa00_11111  1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+a+aa00_11111+o 2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+a+aa00_11111+a 2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+a+aa00_11111+o 3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+a+aa00_11111+a 3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG    >a+aa00_11111    CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >a+aa00_11111+i  CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >a+aa00_11111+o  CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >a+aa00_11111+a  CAUS+IMP+2.F.PL

$radicals:5,type:A,prefix:AT

INDIC+PRF+1+C+SG >at+aa0a_11111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >at+aa0a_11111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aa0a_11111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aa0a_11111+a FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+aa0a_11111+at
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aa0a_11111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >at+aa0a_11111+kum
        FCT+PRF+2.M.PL
```

```
INDIC+PRF+2+F+PL >at+aa0a_11111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aa0a_11111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aa0a_11111+aya
        FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >at+aa00_11111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+at+aa00_11111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+at+aa00_11111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+at+aa00_11111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+at+aa00_11111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+at+aa00_11111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+at+aa00_11111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+aa00_11111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+aa00_11111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+aa00_11111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >at+aa00_11111   FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+at+aa00_11111  2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+at+aa00_11111+i
        2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+at+aa00_11111  3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+at+aa00_11111  3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+at+aa00_11111 1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+at+aa00_11111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+aa00_11111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+aa00_11111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+aa00_11111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >at+aa00_11111   FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >at+aa00_11111+i FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >at+aa00_11111+o FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >at+aa00_11111+a FCT+IMP+2.F.PL

$radicals:5,type:A,prefix:ATTA

INDIC+PRF+1+C+SG >atta+aa0a_11111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+aa0a_11111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+aa0a_11111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+aa0a_11111+a
        FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+aa0a_11111+at
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+aa0a_11111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+aa0a_11111+kum
        FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+aa0a_11111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+aa0a_11111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+aa0a_11111+aya
        FCT+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >atta+aa00_11111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+atta+aa00_11111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+aa00_11111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+atta+aa00_11111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+aa00_11111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+atta+aa00_11111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+aa00_11111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+aa00_11111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+aa00_11111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+aa00_11111+a
        3+FCT+IMPF+F.PL

JUSS+NA+1+C+SG   >atta+aa00_11111 FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+atta+aa00_11111
        2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+aa00_11111+i
        2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+atta+aa00_11111
        3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+aa00_11111
        3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+atta+aa00_11111
        1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+aa00_11111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+aa00_11111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+aa00_11111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+aa00_11111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+aa00_11111 FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+aa00_11111+i
        FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+aa00_11111+o
        FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+aa00_11111+a
        FCT+IMP+2.F.PL

$radicals:5,type:C,prefix:0

INDIC+PRF+1+C+SG 0+aaAa_11111+ko  0+PRF+1.SG
INDIC+PRF+2+M+SG 0+aaAa_11111+ka  0+PRF+2.M.SG
INDIC+PRF+2+F+SG 0+aaAa_11111+ki  0+PRF+2.F.SG
INDIC+PRF+3+M+SG 0+aaAa_11111+a   0+PRF+3.M.SG
INDIC+PRF+3+F+SG 0+aaAa_11111+at  0+PRF+3.F.SG
INDIC+PRF+1+C+PL 0+aaAa_11111+na  0+PRF+1.PL
INDIC+PRF+2+M+PL 0+aaAa_11111+kum 0+PRF+2.M.PL
INDIC+PRF+2+F+PL 0+aaAa_11111+kn  0+PRF+2.F.PL
INDIC+PRF+3+M+PL 0+aaAa_11111+aw  0+PRF+3.M.PL
INDIC+PRF+3+F+PL 0+aaAa_11111+aya 0+PRF+3.F.PL

INDIC+IMPF+1+C+SG        >+aaA0_11111    1.SG+IMPF
INDIC+IMPF+2+M+SG        t+aaA0_11111
        2.M.SG+IMPF
```

```
INDIC+IMPF+2+F+SG       t+aaA0_11111+i
        2+IMPF+F.SG
INDIC+IMPF+3+M+SG       l+aaA0_11111
        3.M.SG+IMPF
INDIC+IMPF+3+F+SG       t+aaA0_11111
        3.F.SG+IMPF
INDIC+IMPF+1+C+PL       >n+aaA0_11111   1.PL+IMPF
INDIC+IMPF+2+M+PL       t+aaA0_11111+o
        2+IMPF+M.PL
INDIC+IMPF+2+F+PL       t+aaA0_11111+a
        2+IMPF+F.PL
INDIC+IMPF+3+M+PL       l+aaA0_11111+o
        3+IMPF+M.PL
INDIC+IMPF+3+F+PL       l+aaA0_11111+a
        3+IMPF+F.PL

JUSS+NA+1+C+SG  >+aaA0_11111    1.SG+JUSS
JUSS+NA+2+M+SG  t+aaA0_11111    2.M.SG+JUSS
JUSS+NA+2+F+SG  t+aaA0_11111+i  2+JUSS+F.SG
JUSS+NA+3+M+SG  l+aaA0_11111    3.M.SG+JUSS
JUSS+NA+3+F+SG  t+aaA0_11111    3.F.SG+JUSS
JUSS+NA+1+C+PL  n+aaA0_11111    1.PL+JUSS
JUSS+NA+2+M+PL  t+aaA0_11111+o  2+JUSS+M.PL
JUSS+NA+2+F+PL  t+aaA0_11111+a  2+JUSS+F.PL
JUSS+NA+3+M+PL  l+aaA0_11111+o  3+JUSS+M.PL
JUSS+NA+3+F+PL  l+aaA0_11111+a  3+JUSS+F.PL

IMP+NA+2+M+SG   0+aaA0_11111    0+IMP.2.M.SG
IMP+NA+2+F+SG   0+aaA0_11111+i  0+IMP+2.F.SG
IMP+NA+2+M+PL   0+aaA0_11111+o  0+IMP+2.M.PL
IMP+NA+2+F+PL   0+aaA0_11111+a  0+IMP+2.F.PL

$radicals:5,type:C,prefix:T

INDIC+PRF+1+C+SG t+aaAa_11111+ko  PASS+PRF+1.SG
INDIC+PRF+2+M+SG t+aaAa_11111+ka  PASS+PRF+2.M.SG
INDIC+PRF+2+F+SG t+aaAa_11111+ki  PASS+PRF+2.F.SG
INDIC+PRF+3+M+SG t+aaAa_11111+a   PASS+PRF+3.M.SG
INDIC+PRF+3+F+SG t+aaAa_11111+at  PASS+PRF+3.F.SG
INDIC+PRF+1+C+PL t+aaAa_11111+na  PASS+PRF+1.PL
INDIC+PRF+2+M+PL t+aaAa_11111+kum PASS+PRF+2.M.PL
INDIC+PRF+2+F+PL t+aaAa_11111+kn  PASS+PRF+2.F.PL
INDIC+PRF+3+M+PL t+aaAa_11111+aw  PASS+PRF+3.M.PL
INDIC+PRF+3+F+PL t+aaAa_11111+aya PASS+PRF+3.F.PL

INDIC+IMPF+1+C+SG       >+t+aaAa_11111
        1.SG+PASS+IMPF
INDIC+IMPF+2+M+SG       t+t+aaAa_11111
        2.M.SG+PASS+IMPF
INDIC+IMPF+2+F+SG       t+t+aaAa_11111+i
        2+PASS+IMPF+F.SG
INDIC+IMPF+3+M+SG       l+t+aaAa_11111
        3.M.SG+PASS+IMPF
INDIC+IMPF+3+F+SG       t+t+aaAa_11111
        3.F.SG+PASS+IMPF
INDIC+IMPF+1+C+PL       >n+t+aaAa_11111
        1.PL+PASS+IMPF
INDIC+IMPF+2+M+PL       t+t+aaAa_11111+o
        2+PASS+IMPF+M.PL
INDIC+IMPF+2+F+PL       t+t+aaAa_11111+a
        2+PASS+IMPF+F.PL
INDIC+IMPF+3+M+PL       l+t+aaAa_11111+o
        3+PASS+IMPF+M.PL
INDIC+IMPF+3+F+PL       l+t+aaAa_11111+a
        3+PASS+IMPF+F.PL

JUSS+NA+1+C+SG  >+t+aaAa_11111  1.SG+PASS+JUSS
JUSS+NA+2+M+SG  t+t+aaAa_11111  2.M.SG+PASS+JUSS
JUSS+NA+2+F+SG  t+t+aaAa_11111+i 2+PASS+JUSS+F.SG
JUSS+NA+3+M+SG  l+t+aaAa_11111  3.M.SG+PASS+JUSS
JUSS+NA+3+F+SG  t+t+aaAa_11111  3.F.SG+PASS+JUSS
JUSS+NA+1+C+PL  >n+t+aaAa_11111 1.PL+PASS+JUSS
JUSS+NA+2+M+PL  t+t+aaAa_11111+o 2+PASS+JUSS+M.PL
JUSS+NA+2+F+PL  t+t+aaAa_11111+a 2+PASS+JUSS+F.PL
JUSS+NA+3+M+PL  l+t+aaAa_11111+o 3+PASS+JUSS+M.PL
JUSS+NA+3+F+PL  l+t+aaAa_11111+a 3+PASS+JUSS+F.PL

IMP+NA+2+M+SG   t+aaAa_11111    PASS+IMP.2.M.SG
IMP+NA+2+F+SG   t+aaAa_11111+i  PASS+IMP+2.F.SG
IMP+NA+2+M+PL   t+aaAa_11111+o  PASS+IMP+2.M.PL
IMP+NA+2+F+PL   t+aaAa_11111+a  PASS+IMP+2.F.PL

$radicals:5,type:C,prefix:A

INDIC+PRF+1+C+SG >a+aaAa_11111+ko CAUS+PRF+1.SG
INDIC+PRF+2+M+SG >a+aaAa_11111+ka CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >a+aaAa_11111+ki CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >a+aaAa_11111+a  CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >a+aaAa_11111+at CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >a+aaAa_11111+na CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >a+aaAa_11111+kum
        CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >a+aaAa_11111+kn CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >a+aaAa_11111+aw CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >a+aaAa_11111+aya
        CAUS+PRF+3.F.PL

INDIC+IMPF+1+C+SG       >a+aaA0_11111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG       t+a+aaA0_11111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG       t+a+aaA0_11111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG       l+a+aaA0_11111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG       t+a+aaA0_11111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL       >n+a+aaA0_11111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL       t+a+aaA0_11111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL       t+a+aaA0_11111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL       l+a+aaA0_11111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL       l+a+aaA0_11111+a
        3+CAUS+IMPF+F.PL

JUSS+NA+1+C+SG  >a+aaA0_11111   CAUS+1.SG+JUSS
JUSS+NA+2+M+SG  t+a+aaA0_11111  2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG  t+a+aaA0_11111+i 2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG  l+a+aaA0_11111  3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG  t+a+aaA0_11111  3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL  >n+a+aaA0_11111 1.PL+CAUS+JUSS
JUSS+NA+2+M+PL  t+a+aaA0_11111+o 2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL  t+a+aaA0_11111+a 2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL  l+a+aaA0_11111+o 3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL  l+a+aaA0_11111+a 3+CAUS+JUSS+F.PL

IMP+NA+2+M+SG   >a+aaA0_11111   CAUS+IMP.2.M.SG
IMP+NA+2+F+SG   >a+aaA0_11111+i CAUS+IMP+2.F.SG
IMP+NA+2+M+PL   >a+aaA0_11111+o CAUS+IMP+2.M.PL
IMP+NA+2+F+PL   >a+aaA0_11111+a CAUS+IMP+2.F.PL

$radicals:5,type:C,prefix:AT

INDIC+PRF+1+C+SG >at+aaAa_11111+ko
        CAUS+PRF+1.SG
```

```
INDIC+PRF+2+M+SG >at+aaAa_11111+ka
        CAUS+PRF+2.M.SG
INDIC+PRF+2+F+SG >at+aaAa_11111+ki
        CAUS+PRF+2.F.SG
INDIC+PRF+3+M+SG >at+aaAa_11111+a CAUS+PRF+3.M.SG
INDIC+PRF+3+F+SG >at+aaAa_11111+at
        CAUS+PRF+3.F.SG
INDIC+PRF+1+C+PL >at+aaAa_11111+na
        CAUS+PRF+1.PL
INDIC+PRF+2+M+PL >at+aaAa_11111+kum
        CAUS+PRF+2.M.PL
INDIC+PRF+2+F+PL >at+aaAa_11111+kn
        CAUS+PRF+2.F.PL
INDIC+PRF+3+M+PL >at+aaAa_11111+aw
        CAUS+PRF+3.M.PL
INDIC+PRF+3+F+PL >at+aaAa_11111+aya
        CAUS+PRF+3.F.PL
INDIC+IMPF+1+C+SG        >at+aaA0_11111
        CAUS+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+at+aaA0_11111
        2.M.SG+CAUS+IMPF
INDIC+IMPF+2+F+SG        t+at+aaA0_11111+i
        2+CAUS+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+at+aaA0_11111
        3.M.SG+CAUS+IMPF
INDIC+IMPF+3+F+SG        t+at+aaA0_11111
        3.F.SG+CAUS+IMPF
INDIC+IMPF+1+C+PL        >n+at+aaA0_11111
        1.PL+CAUS+IMPF
INDIC+IMPF+2+M+PL        t+at+aaA0_11111+o
        2+CAUS+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+at+aaA0_11111+a
        2+CAUS+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+at+aaA0_11111+o
        3+CAUS+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+at+aaA0_11111+a
        3+CAUS+IMPF+F.PL
JUSS+NA+1+C+SG   >at+aaA0_11111   CAUS+1.SG+JUSS
JUSS+NA+2+M+SG   t+at+aaA0_11111  2.M.SG+CAUS+JUSS
JUSS+NA+2+F+SG   t+at+aaA0_11111+i
        2+CAUS+JUSS+F.SG
JUSS+NA+3+M+SG   l+at+aaA0_11111  3.M.SG+CAUS+JUSS
JUSS+NA+3+F+SG   t+at+aaA0_11111  3.F.SG+CAUS+JUSS
JUSS+NA+1+C+PL   >n+at+aaA0_11111 1.PL+CAUS+JUSS
JUSS+NA+2+M+PL   t+at+aaA0_11111+o
        2+CAUS+JUSS+M.PL
JUSS+NA+2+F+PL   t+at+aaA0_11111+a
        2+CAUS+JUSS+F.PL
JUSS+NA+3+M+PL   l+at+aaA0_11111+o
        3+CAUS+JUSS+M.PL
JUSS+NA+3+F+PL   l+at+aaA0_11111+a
        3+CAUS+JUSS+F.PL
IMP+NA+2+M+SG    >at+aaA0_11111   CAUS+IMP.2.M.SG
IMP+NA+2+F+SG    >at+aaA0_11111+i CAUS+IMP+2.F.SG
IMP+NA+2+M+PL    >at+aaA0_11111+o CAUS+IMP+2.M.PL
IMP+NA+2+F+PL    >at+aaA0_11111+a CAUS+IMP+2.F.PL

$radicals:5,type:C,prefix:ATTA
INDIC+PRF+1+C+SG >atta+aaAa_11111+ko
        FCT+PRF+1.SG
INDIC+PRF+2+M+SG >atta+aaAa_11111+ka
        FCT+PRF+2.M.SG
INDIC+PRF+2+F+SG >atta+aaAa_11111+ki
        FCT+PRF+2.F.SG
INDIC+PRF+3+M+SG >atta+aaAa_11111+a
        FCT+PRF+3.M.SG
INDIC+PRF+3+F+SG >atta+aaAa_11111+at
        FCT+PRF+3.F.SG
INDIC+PRF+1+C+PL >atta+aaAa_11111+na
        FCT+PRF+1.PL
INDIC+PRF+2+M+PL >atta+aaAa_11111+kum
        FCT+PRF+2.M.PL
INDIC+PRF+2+F+PL >atta+aaAa_11111+kn
        FCT+PRF+2.F.PL
INDIC+PRF+3+M+PL >atta+aaAa_11111+aw
        FCT+PRF+3.M.PL
INDIC+PRF+3+F+PL >atta+aaAa_11111+aya
        FCT+PRF+3.F.PL
INDIC+IMPF+1+C+SG        >atta+aaA0_11111
        FCT+1.SG+IMPF
INDIC+IMPF+2+M+SG        t+atta+aaA0_11111
        2.M.SG+FCT+IMPF
INDIC+IMPF+2+F+SG        t+atta+aaA0_11111+i
        2+FCT+IMPF+F.SG
INDIC+IMPF+3+M+SG        l+atta+aaA0_11111
        3.M.SG+FCT+IMPF
INDIC+IMPF+3+F+SG        t+atta+aaA0_11111
        3.F.SG+FCT+IMPF
INDIC+IMPF+1+C+PL        >n+atta+aaA0_11111
        1.PL+FCT+IMPF
INDIC+IMPF+2+M+PL        t+atta+aaA0_11111+o
        2+FCT+IMPF+M.PL
INDIC+IMPF+2+F+PL        t+atta+aaA0_11111+a
        2+FCT+IMPF+F.PL
INDIC+IMPF+3+M+PL        l+atta+aaA0_11111+o
        3+FCT+IMPF+M.PL
INDIC+IMPF+3+F+PL        l+atta+aaA0_11111+a
        3+FCT+IMPF+F.PL
JUSS+NA+1+C+SG   >atta+aaA0_11111 FCT+1.SG+JUSS
JUSS+NA+2+M+SG   t+atta+aaA0_11111
        2.M.SG+FCT+JUSS
JUSS+NA+2+F+SG   t+atta+aaA0_11111+i
        2+FCT+JUSS+F.SG
JUSS+NA+3+M+SG   l+atta+aaA0_11111
        3.M.SG+FCT+JUSS
JUSS+NA+3+F+SG   t+atta+aaA0_11111
        3.F.SG+FCT+JUSS
JUSS+NA+1+C+PL   >n+atta+aaA0_11111
        1.PL+FCT+JUSS
JUSS+NA+2+M+PL   t+atta+aaA0_11111+o
        2+FCT+JUSS+M.PL
JUSS+NA+2+F+PL   t+atta+aaA0_11111+a
        2+FCT+JUSS+F.PL
JUSS+NA+3+M+PL   l+atta+aaA0_11111+o
        3+FCT+JUSS+M.PL
JUSS+NA+3+F+PL   l+atta+aaA0_11111+a
        3+FCT+JUSS+F.PL

IMP+NA+2+M+SG    >atta+aaA0_11111 FCT+IMP.2.M.SG
IMP+NA+2+F+SG    >atta+aaA0_11111+i
        FCT+IMP+2.F.SG
IMP+NA+2+M+PL    >atta+aaA0_11111+o
        FCT+IMP+2.M.PL
IMP+NA+2+F+PL    >atta+aaA0_11111+a
        FCT+IMP+2.F.PL
```

Приложение 3. Регулярные выражения первого компонента анализатора

```
================================ nominal-stem.json ================================
[
 {
  "comment": "broken plural: balas -> >aballs, naggal -> >anaggl",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\
k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2cons}${C3}\\).PL"
 },
 {
  "comment": "broken plural: bggu< -> >abAg<",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural, final laryngeal: kal< -> >akall<t, >akall<at",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\
k<C2cons>)(?<C3>[hHX><]))(?<suf>a?t)$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2cons}${C3}\\).PL-${suf}:PL"
 },
 {
  "comment": "broken plural: CaCaC{1,2}C",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>(?<C3cons>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(\\k<C3cons>)?)(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3cons}${C4}\\).PL"
 },
 {
  "comment": "broken plural: CaCAC[iu]?C: qaSir -> qaSayr",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])([iu]?)(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}${C4}\\).PL"
 },
 {
  "comment": "broken plural: CawaC{2}C",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])awa(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>
)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2cons}${C3}\\).PL"
 },
 {
  "comment": "broken plural: CawACC",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])awA(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGg
TCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: CaCAyC",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C3>[lmrsxqbtnkzdGg
TCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: CaCaC{2}it?; rora -> rawarri(t), manfat -> manAfit",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>(?<C3cons>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])\\k<C3cons>))(?<suf>it?)$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL-${suf}:PL"
 },
 {
```

```
  "comment": "broken plural: CawaC{2}it? <- (CC).PL-PL",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])awa(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>
))(?<suf>it?)$",
  "replace": "${root}:\\(${C1}${C2}\\).PL-${suf}:PL"
 },
 {
  "comment": "broken plural: CawACit? <- (CC).PL-PL",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])awA(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>it?)$",
  "replace": "${root}:\\(${C1}${C2}\\).PL-${suf}:PL"
 },
 {
  "comment": "broken plural: >aCCAC; qor -> >aqwAr, kis -> >akyAs",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: >aCC[uA]?C (2nd consonant is not a laryngeal or a semivowel); dabr -> >adbr,
daqal -> adqul, kalb -> >aklAb",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpP])([uA]?)(?<C3>[lmrsxq
btnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: L[aA]C[uA]?C, L = laryngeal; Hlm -> HAlAm, >adg -> >adug",
  "regex":
"^(?<root>(?<C1>[hHX><])([aA])(?<C2>[lmrsxqbtnkzdGgTCSfcZpP])([uA]?)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: >aCCACAt; mdr -> >amdArAt",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL-${suf}:PL"
 },
 {
  "comment": "broken plural: >aCCCat; luH -> >alwHAt, >alwaHAt",
  "regex":
"^(?<pref>>a)(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxq
btnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PL-${root}:\\(${C1}${C2}${C3}\\).PL-${suf}:PL"
 },
 {
  "comment": "broken plural: CCaC; kuk -> kwak, Hilat -> Hyal",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: CCAC, 2nd cons not a semivowel",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])A(?<C3>[lmrsxqbtnkzdGgTCSf
cZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
  "comment": "broken plural: CaC[aA]C, 2nd cons not a semivowel",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])([aA]?)(?<C3>[lmrsxqbtnkz
dGgTCSfcZpPwyhHX><]))$",
  "replace": "${root}:\\(${C1}${C2}${C3}\\).PL"
 },
 {
```

```
 "comment": "suffixed plural: -at (lomin -> lominat)",
 "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>at)$",
 "replace": "\\[${root}\\]:#-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: -Ay",
 "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Ay)$",
 "replace": "\\[${root}\\]:#-${suf}:SG.M"
},
{
 "comment": "suffixed singulative: -Ayt",
 "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Ayt)$",
 "replace": "\\[${root}\\]:#-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: a-deletion (qaTaf -> qaTfat)",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(at))$",
 "replace": "${root}:${g1}a${C}-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: a-deletion",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ay))$",
 "replace": "${root}:${g1}a${C}-${suf}:SG.M"
},
{
 "comment": "suffixed singulative: a-deletion",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ayt))$",
 "replace": "${root}:${g1}a${C}-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: A-deletion (GlAb -> Glbat)",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(at))$",
 "replace": "${root}:${g1}A${C}-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: A-deletion",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ay))$",
 "replace": "${root}:${g1}A${C}-${suf}:SG.M"
},
{
 "comment": "suffixed singulative: A-deletion",
 "regex": "^(?<root>(?<g1>.+)(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ayt))$",
 "replace": "${root}:${g1}A${C}-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: t-insertion, final vowel (wagre -> wagretat)",
 "regex": "^(?<root>.+[ieAuo])(?<suf>t(at))$",
 "replace": "\\[${root}\\]:#-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: t-insertion",
 "regex": "^(?<root>.+[ieAuo])(?<suf>t(Ay))$",
 "replace": "\\[${root}\\]:#-${suf}:SG.M"
},
{
 "comment": "suffixed singulative: t-insertion",
 "regex": "^(?<root>.+[ieAuo])(?<suf>t(Ayt))$",
 "replace": "\\[${root}\\]:#-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: final a deletion (Cewa -> Cewat)",
 "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>(at))$",
 "replace": "\\[${root}\\]:#a-${suf}:SG.F"
},
{
 "comment": "suffixed singulative: final a deletion",
 "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>(Ay))$",
 "replace": "\\[${root}\\]:#a-${suf}:SG.M"
},
```

```
 {
  "comment": "suffixed singulative: final a deletion",
  "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>(Ay))$",
  "replace": "\\[${root}\\]:#a-${suf}:SG.F"
 },
 {
  "comment": "suffixed singulative: -Ay 0 -> e (kstAn -> kstenAy, kstenAyt)",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ay))$",
  "replace": "${root}:${g1}${C}-${suf}:SG.M"
 },
 {
  "comment": "suffixed singulative: -Ay a -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ay))$",
  "replace": "${root}:${g1}a${C}-${suf}:SG.M"
 },
 {
  "comment": "suffixed singulative: -Ay A -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ay))$",
  "replace": "${root}:${g1}A${C}-${suf}:SG.M"
 },
 {
  "comment": "suffixed singulative: -Ayt 0 -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ayt))$",
  "replace": "${root}:${g1}${C}-${suf}:SG.F"
 },
 {
  "comment": "suffixed singulative: -Ayt a -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ayt))$",
  "replace": "${root}:${g1}a${C}-${suf}:SG.F"
 },
 {
  "comment": "suffixed singulative: -Ayt A -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>(Ayt))$",
  "replace": "${root}:${g1}A${C}-${suf}:SG.F"
 },
 {
  "comment": "-At plural: ...CAC+At // dAr -> dArAt, >akAn -> >akAnAt",
  "regex": "^(?<root>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>At)$",
  "replace": "\\[${root}\\]:#-${suf}:PL"
 },
 {
  "comment": "-At plural: t-insertion after final vowel// masanqo -> masanqotAt",
  "regex": "^(?<root>.+[ieaAuo])(?<suf>tAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PL"
 },
 {
  "comment": "-At plural: CVCet -> CaCyAt // xawet -> xawyAt",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])y)(?<suf>At)$",
  "replace": "${root}:${C1}\\(V\\)${C2}et-${suf}:PL"
 },
 {
  "comment": "-At plural: CVCot -> CaCyAt // rkot -> rakwAt",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])w)(?<suf>At)$",
  "replace": "${root}:${C1}\\(V\\)${C2}ot-${suf}:PL"
 },
 {
  "comment": "-At plural: ..at -> ..At // xAkat -> xAkAt",
  "regex": "^(?<root>.+)(?<suf>At)$",
  "replace": "\\[${root}\\]:#-${suf}:PL"
 },
 {
  "comment": "-otAt plural: ..at -> ..otAt // sadAyat -> sadAyotAt",
  "regex": "^(?<root>.+)(?<suf>otAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PL"
 },
 {
  "comment": "-otAt plural: CACC+otAt // >Anf -> >AnfotAt",
```

```
  "regex":
"^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<su
f>otAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PL"
 },
 {
  "comment": "-otAt plural: CVC -> CVC{2}+otAt // >ad -> >addotAt",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<V>[ieaAuo])(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2
cons>)(?<suf>otAt)$",
  "replace": "${root}:${C1}${V}${C2cons}-${suf}:PL"
 },
 {
  "comment": "-otAt plural: semivowel metathesis: ..Ce -> ..yC+otat // zamAte -> zamAytotAt",
  "regex": "^(?<root>(?<g1>.+)y(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>otAt)$",
  "replace": "${root}:${g1}${C}e-${suf}:PL"
 },
 {
  "comment": "-otAt plural: semivowel metathesis: ..Ci -> ..yC+otat",
  "regex": "^(?<root>(?<g1>.+)y(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>otAt)$",
  "replace": "${root}:${g1}${C}i-${suf}:PL"
 },
 {
  "comment": "-otAt plural: semivowel metathesis: ..Co -> ..wC+otat",
  "regex": "^(?<root>(?<g1>.+)w(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>otAt)$",
  "replace": "${root}:${g1}${C}o-${suf}:PL"
 },
 {
  "comment": "-otAt plural: semivowel metathesis: ..Cu -> ..wC+otat",
  "regex": "^(?<root>(?<g1>.+)w(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>otAt)$",
  "replace": "${root}:${g1}${C}u-${suf}:PL"
 },
 {
  "comment": "diminutive: -Ay (MASC): .+[iuo]CAy; ..eCAy handled below; ..[Cons,a,A]CAy do not occur (e
insertion)",
  "regex": "^(?<root>.+[iuo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Ay)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.M"
 },
 {
  "comment": "diminutive: -at (FEM): .{2,}[iuo]Cat; ..eCat handled below; ..[Cons,a,A]Cat do not occur (e
insertion); NB: *C[iuo]C-at (formed with -atit, see below)",
  "regex": "^(?<root>.{2,}[iuo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>at)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.F"
 },
 {
  "comment": "diminutive: -Ay (MASC): .+[iuo]tAy, t-insertion after vowel",
  "regex": "^(?<root>.+[iuo])(?<suf>tAy)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.M"
 },
 {
  "comment": "diminutive: -at (FEM): .+[iuo]tat, t-insertion after vowel",
  "regex": "^(?<root>.+[iuo])(?<suf>tat)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.F"
 },
 {
  "comment": "diminutive: -Ay: ..eCAy, 0 -> e; NB: *CACeCAy <- CACC (formed with -etAy)",
  "regex":
"^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<![lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Ay)$",
  "replace": "${root}:${g1}${C}-${suf}:DIM.M"
 },
 {
  "comment": "diminutive: -at: ..eCat, 0 -> e; NB: *CACeCat <- CACC (formed with -atit)",
  "regex":
"^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<![lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>at)$",
  "replace": "${root}:${g1}${C}-${suf}:DIM.F"
 },
 {
```

```
 "comment": "diminutive: -Ay: ..etAy, 0 -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAy)$",
 "replace": "${root}:${g1}-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..etat, 0 -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tat)$",
 "replace": "${root}:${g1}-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..eCAy, a -> e",
 "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
 "replace": "${root}:${g1}a${C}-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..eCat, a -> e",
 "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
 "replace": "${root}:${g1}a${C}-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..etAy, a -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAy)$",
 "replace": "${root}:${g1}a-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..etat, a -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tat)$",
 "replace": "${root}:${g1}a-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..eCAy, A -> e",
 "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
 "replace": "${root}:${g1}A${C}-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..eCat, A -> e; NB: *CeC-at <- CAC (formed with -atit)",
 "regex": "^(?<root>(?<g1>.{2,})e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
 "replace": "${root}:${g1}A${C}-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..etAy, A -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAy)$",
 "replace": "${root}:${g1}A-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..etat, A -> e, t-insertion",
 "regex": "^(?<root>(?<g1>.+)e)(?<suf>tat)$",
 "replace": "${root}:${g1}A-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..eCAy, e -> e",
 "regex": "^(?<root>.+e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Ay)$",
 "replace": "\\[${root}\\]:#-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..eCat, e -> e; NB: *CeC-at <- CeC",
 "regex": "^(?<root>.{2,}e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>at)$",
 "replace": "\\[${root}\\]:#-${suf}:DIM.F"
},
{
 "comment": "diminutive: -Ay: ..etAy, e -> e, t-insertion",
 "regex": "^(?<root>.+e)(?<suf>tAy)$",
 "replace": "\\[${root}\\]:#-${suf}:DIM.M"
},
{
 "comment": "diminutive: -at: ..etat, e -> e, t-insertion",
 "regex": "^(?<root>.+e)(?<suf>tat)$",
 "replace": "\\[${root}\\]:#-${suf}:DIM.F"
},
```

```
 {
  "comment": "CCC -> CC{2}C + Ay (along with -> e) // flq -> flleqAy",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e(
?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${root}:${C1}${C2cons}${C3}-${suf}:DIM.M"
 },
 {
  "comment": "CCC -> CC{2}C + at (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e(
?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${root}:${C1}${C2cons}${C3}-${suf}:DIM.F"
 },
 {
  "comment": "CaCaC -> CaC2aC + Ay (along with -> e) // zanab -> zannebAy",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${root}:${C1}a${C2cons}a${C3}-${suf}:DIM.M"
 },
 {
  "comment": "CaCaC -> CaC2aC + at (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${root}:${C1}a${C2cons}a${C3}-${suf}:DIM.F"
 },
 {
  "comment": "CaCC -> CaC2C + Ay (along with -> e) // wakd -> wakkedat",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${root}:${C1}a${C2cons}${C3}-${suf}:DIM.M"
 },
 {
  "comment": "CaCC -> CaC2C + at (along with -> e) // wakd -> wakkedat",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${root}:${C1}a${C2cons}${C3}-${suf}:DIM.F"
 },
 {
  "comment": "-at:F-it:DIM.F // Galb-at-it",
  "regex": "^(?<root>.+)(?<sufFem>at)(?<sufDim>it)$",
  "replace": "\\[${root}\\]:#-${sufFem}:F-${sufDim}:DIM.F"
 },
 {
  "comment": "-t:F-it:DIM.F // <adg-t-it",
  "regex": "^(?<root>.+)(?<sufFem>t)(?<sufDim>it)$",
  "replace": "\\[${root}\\]:#-${sufFem}:F-${sufDim}:DIM.F"
 },
 {
  "comment": "-etAy diminutive // ktkt-etay <- (ktkt-tay)",
  "regex": "^(?<root>.+)(?<suf>etAy)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.M"
 },
 {
  "comment": "-etAy diminutive, Ca?C{2}etAy <- CVC // dss-etAy <- ds",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<V>a?)(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2
cons>))(?<suf>etAy)$",
  "replace": "${root}:${C1}${V}${C2}-${suf}:DIM.M"
 },
 {
  "comment": "-atit diminutive, Ca?C{2}atit <- CVC // >mm-atit <- >m",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<V>a?)(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2
cons>))(?<suf>atit)$",
```

```
  "replace": "${root}:${C1}${V}${C2}-${suf}:DIM.F"
 },
 {
  "comment": "-atit diminutive, C[Auieo]Catit <- CVC // bet-atit <- bet",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][Auieo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>atit)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.F"
 },
 {
  "comment": "-etAy diminutive, CACCetAy <- CACC",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>etAy)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.M"
 },
 {
  "comment": "-atit diminutive, CACCatit <- CACC",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>atit)$",
  "replace": "\\[${root}\\]:#-${suf}:DIM.F"
 },
 {
  "comment": "paucative: -Am (MASC): .+[iuo]CAm; ..eCAm handled below; ..[Cons,a,A]CAm do not occur (e
insertion)",
  "regex": "^(?<root>.+[iuo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Am)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At (FEM): .{2,}[iuo]CAt; ..eCAt handled below; ..[Cons,a,A]CAt do not occur (e
insertion); NB: *C[iuo]C-At (formed with -etAt, see below)",
  "regex": "^(?<root>.{2,}[iuo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>At)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am (MASC): .+[iuo]tAm, t-insertion after vowel",
  "regex": "^(?<root>.+[iuo])(?<suf>tAm)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At (FEM): .+[iuo]tAt, t-insertion after vowel",
  "regex": "^(?<root>.+[iuo])(?<suf>tAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..eCAm, 0 -> e; NB: *CACeCAm <- CACC (formed with -etAm)",
  "regex":
"^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<![lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Am)$",
  "replace": "${root}:${g1}${C}-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..eCAt, 0 -> e; NB: *CACeCAt <- CACC (formed with -etAt)",
  "regex":
"^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<![lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>At)$",
  "replace": "${root}:${g1}${C}-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..etAm, 0 -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAm)$",
  "replace": "${root}:${g1}-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..etAt, 0 -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAt)$",
  "replace": "${root}:${g1}-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..eCAm, a -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
  "replace": "${root}:${g1}a${C}-${suf}:PAU.M"
 },
 {
```

```
  "comment": "paucative: -At: ..eCAt, a -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${root}:${g1}a${C}-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..etAm, a -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAm)$",
  "replace": "${root}:${g1}a-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..etAt, a -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAt)$",
  "replace": "${root}:${g1}a-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..eCAm, A -> e",
  "regex": "^(?<root>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
  "replace": "${root}:${g1}A${C}-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..eCAt, A -> e; NB: *CeC-At <- CAC (formed with -etAt)",
  "regex": "^(?<root>(?<g1>.{2,})e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${root}:${g1}A${C}-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..etAm, A -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAm)$",
  "replace": "${root}:${g1}A-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..etAt, A -> e, t-insertion",
  "regex": "^(?<root>(?<g1>.+)e)(?<suf>tAt)$",
  "replace": "${root}:${g1}A-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..eCAm, e -> e",
  "regex": "^(?<root>.+e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Am)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..eCAt, e -> e; NB: *CeC-At <- CeC",
  "regex": "^(?<root>.{2,}e[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>At)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "paucative: -Am: ..etAm, e -> e, t-insertion",
  "regex": "^(?<root>.+e)(?<suf>tAm)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "paucative: -At: ..etAt, e -> e, t-insertion",
  "regex": "^(?<root>.+e)(?<suf>tAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "CCC -> CC{2}C + Am (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e(
?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
  "replace": "${root}:${C1}${C2cons}${C3}-${suf}:PAU.M"
 },
 {
  "comment": "CCC -> CC{2}C + At (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e(
?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${root}:${C1}${C2cons}${C3}-${suf}:PAU.F"
 },
 {
```

```
  "comment": "CaCaC -> CaC2aC + Am (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
  "replace": "${root}:${C1}a${C2cons}a${C3}-${suf}:PAU.M"
 },
 {
  "comment": "CaCaC -> CaC2aC + At (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${root}:${C1}a${C2cons}a${C3}-${suf}:PAU.F"
 },
 {
  "comment": "CaCC -> CaC2C + Am (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
  "replace": "${root}:${C1}a${C2cons}${C3}-${suf}:PAU.M"
 },
 {
  "comment": "CaCC -> CaC2C + At (along with -> e)",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2cons>)e
(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
  "replace": "${root}:${C1}a${C2cons}${C3}-${suf}:PAU.F"
 },
 {
  "comment": "-At:F-it:PAU.F",
  "regex": "^(?<root>.+)(?<sufFem>At)(?<sufDim>it)$",
  "replace": "\\[${root}\\]:#-${sufFem}:F-${sufDim}:PAU.F"
 },
 {
  "comment": "-t:F-it:PAU.F",
  "regex": "^(?<root>.+)(?<sufFem>t)(?<sufDim>it)$",
  "replace": "\\[${root}\\]:#-${sufFem}:F-${sufDim}:PAU.F"
 },
 {
  "comment": "-etAm paucative",
  "regex": "^(?<root>.+)(?<suf>etAm)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "-etAm paucative, Ca?C{2}etAm <- CVC",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<V>a?)(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2
cons>))(?<suf>etAm)$",
  "replace": "${root}:${C1}${V}${C2}-${suf}:PAU.M"
 },
 {
  "comment": "-etAt paucative, Ca?C{2}etAt <- CVC",
  "regex":
"^(?<root>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<V>a?)(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2
cons>))(?<suf>etAt)$",
  "replace": "${root}:${C1}${V}${C2}-${suf}:PAU.F"
 },
 {
  "comment": "-etAt paucative, C[Auieo]CetAt <- CVC",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][Auieo][lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>etAt)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "-etAm paucative, CACCetAm <- CACC",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>etAm)$",
  "replace": "\\[${root}\\]:#-${suf}:PAU.M"
 },
 {
  "comment": "-etAt paucative, CACCetAt <- CACC",
  "regex": "^(?<root>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]A[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>etAt)$",
```

```
  "replace": "\\[${root}\\]:#-${suf}:PAU.F"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{-L}C{-SV} (2nd not a laryngeal, 3rd not a
semivowel) // qAtl",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{-L}C{-SV} // qAtl-at",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:M.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{-L}C{-SV} // qAtl-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{-L}C{-SV} // qAtl-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$
",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Am)$
",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>At)$
",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{-L}y // qly -> qAli",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{-L}y // qly -> qAly-at",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.SG"
 },
```

```
 {
  "comment": "3-radical type A active participle, f. sg., CC{-L}y; 2nd-3rd rad transposition // qly -> qAyl-
at",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{-L}y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{-L}y; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{-L}y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{-L}y; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{-L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{-L}w; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{-L}w; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, f. pl., CC{-L}w; 2nd-3rd rad transposition",
```

```
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A active participle, m. sg., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT.M.SG"
 },
 {
  "comment": "3-radical type A active participle, f. sg., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A active participle, m. pl., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.ACT-${suf}:F.PL"
 },
 {
  "comment": "3-radical type B active participle, m. sg. // >m(2)r -> ma->amr-Ay",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type B active participle, m. sg., no suffix // >m(2)r -> ma->ammr",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\
k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type B active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type B active participle, pl.",
```

```
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type C active participle, m. sg. // l(A)Sy -> ma-lASy-Ay",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type C active participle, m. sg., no suffix // l(A)Sy -> ma-lASy",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type C active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type C active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type A/B (from A) passive (t-) participle, m. sg. // ma-t-kabbt-Ay",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type A/B (from A) passive (t-) participle, m. sg., no suffix // ma-t-kabbt",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type A/B (from A) passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A/B (from A) passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type A/B (from B) passive (t-) participle, m. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\)-${suf}:M.SG"
 },
 {
```

```
  "comment": "3-radical type A/B (from B) passive (t-) participle, m. sg., no suffix",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type A/B (from B) passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A/B (from B) passive (t-) participle, pl.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><])\\k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type C passive (t-) participle, m. sg. // ma-t-gAmr-Ay",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type C passive (t-) participle, m. sg., no suffix // ma-t-gAmr",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type C passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type C passive (t-) participle, pl.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type >a-A active participle, m. sg. // ma-wld-Ay",
  "regex": "^(?<pref>ma)(?<stem>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{3})(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${stem}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type >a-A active participle, m. sg., no suffix // ma-wld",
  "regex": "^(?<pref>ma)(?<stem>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{3})$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${stem}\\).M.SG"
 },
 {
  "comment": "3-radical type >a-A active participle, f. sg.",
  "regex": "^(?<pref>ma)(?<stem>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{3})(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${stem}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type >a-A active participle, pl.",
  "regex": "^(?<pref>ma)(?<stem>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{3})(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${stem}\\)-${suf}:PL"
 },
 {
```

```
  "comment": "3-radical type >a-B active participle, m. sg.; identical to B active ptcp",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type >a-B active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\
k<C2cons>)(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type >a-B active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type >a-B active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type >a-C active participle, m. sg.; identical to C active ptcp",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type >a-C active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).M.SG"
 },
 {
  "comment": "3-radical type >a-C active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type >a-C active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C2}${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type >at-C active participle, m. sg.; identical to C passive (t-) ptcp",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:M.SG"
 },
 {
  "comment": "3-radical type >at-C active participle, m. sg., no suffix",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\).M.SG"
 },
```

```
 {
  "comment": "3-radical type >at-C active participle, f. sg.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:F.SG"
 },
 {
  "comment": "3-radical type >at-C active participle, pl.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>(?<C2cons>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}\\(A\\)${C2cons}${C3}\\)-${suf}:PL"
 },
 {
  "comment": "3-radical type A passive participle, m. sg. // qtul. NB: *CCuw (CCw -> CCuy), etc. for all other
forms",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., 3rd not y, no suffix // qtl",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPyhHX><]))$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., 3rd not y, with suffix // qtl-t",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A passive participle, m. pl., CCC{-L} // qtul-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A passive participle, f. pl., CCC{-L} // qtul-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A passive participle, m. sg., CCuy <- CCw // qtul. NB: *CCuw (CCw -> CCuy), etc.
for all other forms",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., CCi <- CCy, no suffix",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., CCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., CCi <- CCw, no suffix",
```

```
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type A passive participle, f. sg., CCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A passive participle, m. pl., CCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A passive participle, f. pl., CCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A passive participle, m. pl., CCC{+L} // bzH-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[hHX><]))(?<suf>Am
)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A passive participle, f. pl., CCC{+L} // bzH-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[hHX><]))(?<suf>At
)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type B passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)u(?<C3>[l
mrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., 3rd not y, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)(?<C3>[lm
rsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., 3rd not y, with suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)(?<C3>[lm
rsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type B passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)u(?<C3>[l
mrsxqbtnkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type B passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)u(?<C3>[l
mrsxqbtnkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS-${suf}:F.PL"
```

```
 },
 {
  "comment": "3-radical type B passive participle, m. sg., CCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)uy)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., CCi <- CCy, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)i)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)y\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., CCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)i)(?<suf>
t)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., CCi <- CCw, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)i)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)w\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "3-radical type B passive participle, f. sg., CCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)i)(?<suf>
t)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type B passive participle, m. pl., CCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)uy)(?<suf
>Am)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type B passive participle, f. pl., CCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)uy)(?<suf
>At)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type B passive participle, m. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)(?<C3>[hH
X><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type B passive participle, f. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>(?<C2cons>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2cons>)(?<C3>[hH
X><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2cons}\\(2\\)${C3}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type C passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS.M.SG"
 },
 {
```

```
  "comment": "3-radical type C passive participle, f. sg., 3rd not y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type C passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type C passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type C passive participle, m. sg., CuCuy <- CCw",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "3-radical type C passive participle, f. sg., CuCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type C passive participle, f. sg., CuCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "3-radical type C passive participle, m. pl., CuCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type C passive participle, f. pl., CuCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type C passive participle, m. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[hHX><]))(?<suf>A
m)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "3-radical type C passive participle, f. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[hHX><]))(?<suf>A
t)$",
  "replace": "${stem}:\\(${C1}\\(A\\)${C2}${C3}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type A active participle, m. sg. // xnkl -> ma-xankl-Ay",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:M.SG"
```

```
 },
 {
  "comment": "4-radical type A active participle, m. sg., no suffix // ma-xankl",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type A active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type A active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type C active participle, m. sg. // l(A)Sy -> ma-lASy-Ay",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type C active participle, m. sg., no suffix // l(A)Sy -> ma-lASy",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type C active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type C active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type A passive (t-) participle, m. sg. // ma-t-kabbt-Ay",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type A passive (t-) participle, m. sg., no suffix // ma-t-kabbt",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type A passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
```

```
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type A passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type C passive (t-) participle, m. sg. // ma-t-gAmr-Ay",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type C passive (t-) participle, m. sg., no suffix // ma-t-gAmr",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type C passive (t-) participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type C passive (t-) participle, pl.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type >a-A active participle, m. sg. // ma-wld-Ay",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type >a-A active participle, m. sg., no suffix // ma-wld",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type >a-A active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type >a-A active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type >a-C active participle, m. sg.; identical to C active ptcp",
```

```
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type >a-C active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}\\(A\\)${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type >a-C active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type >a-C active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT.CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type >at-C active participle, m. sg.; identical to C passive (t-) ptcp",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:M.SG"
 },
 {
  "comment": "4-radical type >at-C active participle, m. sg., no suffix",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).M.SG"
 },
 {
  "comment": "4-radical type >at-C active participle, f. sg.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:F.SG"
 },
 {
  "comment": "4-radical type >at-C active participle, pl.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP.ACT-${prefCaus}:CAUS-${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\)-${suf}:PL"
 },
 {
  "comment": "4-radical type D active participle, m. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A\\k<C2>(?<C
4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:M.SG"
 },
 {
  "comment": "4-radical type D active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A\\k<C2>(?<C
4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ.M.SG"
 },
 {
```

```
  "comment": "4-radical type D active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A\\k<C2>(?<C
4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:F.SG"
 },
 {
  "comment": "4-radical type D active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A\\k<C2>(?<C
4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:PL"
 },
 {
  "comment": "4-radical type t-D passive participle, m. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:M.SG"
 },
 {
  "comment": "4-radical type t-D passive participle, m. sg., no suffix",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C4}\\).FREQ.M.SG"
 },
 {
  "comment": "4-radical type t-D passive participle, f. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:F.SG"
 },
 {
  "comment": "4-radical type t-D passive participle, pl.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:PL"
 },
 {
  "comment": "4-radical type t-D passive participle, m. sg.",
  "regex":
"^(?<prefPtcp>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ay)$",
  "replace": "${prefPtcp}:PTCP-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:M.SG"
 },
 {
  "comment": "4-radical type >at-D active participle, m. sg., no suffix",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefPtcp}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ.M.SG"
 },
 {
  "comment": "4-radical type >at-D active participle, f. sg.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Ayt)$",
  "replace": "${prefPtcp}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:F.SG"
 },
 {
  "comment": "4-radical type >at-D active participle, pl.",
  "regex":
"^(?<prefPtcp>m)(?<prefCaus>at)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyh
HX><])A\\k<C2>(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${prefPtcp}:PTCP.ACT-${stem}:\\(${C1}${C2}${C4}\\).FREQ-${suf}:PL"
 },
```

```
 {
  "comment": "4-radical type A passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., 3rd not y, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., 3rd not y, with suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type A passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type A passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type A passive participle, m. sg., CCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., CCi <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}y\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., CCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., CCi <- CCw, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}w\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "4-radical type A passive participle, f. sg., CCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}w\\).PTCP.PASS-${suf}:F.SG"
```

```
 },
 {
  "comment": "4-radical type A passive participle, m. pl., CCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type A passive participle, f. pl., CCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type A passive participle, m. pl., CCC{+L} // bzH-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[hHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type A passive participle, f. pl., CCC{+L} // bzH-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[hHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type C passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type C passive participle, f. sg., 3rd not y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type C passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type C passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type C passive participle, m. sg., CuCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type C passive participle, f. sg., CuCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])i)(?<suf>t)$",
```

```
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type C passive participle, f. sg., CuCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type C passive participle, m. pl., CuCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type C passive participle, f. pl., CuCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type C passive participle, m. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])(?<C4>[hHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type C passive participle, f. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])(?<C4>[hHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}\\(A\\)${C3}${C4}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type D passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>u(?<C4>[lmrsxqbt
nkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type D passive participle, f. sg., 3rd not y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>(?<C4>[lmrsxqbtn
kzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type D passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>u(?<C4>[lmrsxqbt
nkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type D passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>u(?<C4>[lmrsxqbt
nkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type D passive participle, m. sg., CuCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>uy)$",
```

```
  "replace": "${stem}:\\(${C1}${C2}w\\).FREQ.PTCP.PASS.M.SG"
 },
 {
  "comment": "4-radical type D passive participle, f. sg., CuCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).FREQ.PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type D passive participle, f. sg., CuCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).FREQ.PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "4-radical type D passive participle, m. pl., CuCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).FREQ.PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type D passive participle, f. pl., CuCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}w\\).FREQ.PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "4-radical type D passive participle, m. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>(?<C4>[hHX><]))(
?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "4-radical type D passive participle, f. pl., CCC{+L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u\\k<C2>(?<C4>[hHX><]))(
?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C4}\\).FREQ.PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type A active participle, m. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\)-${suf}:M.SG"
 },
 {
  "comment": "5-radical type A active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).M.SG"
 },
 {
  "comment": "5-radical type A active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\)-${suf}:F.SG"
 },
 {
  "comment": "5-radical type A active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>at)$",
```

```
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\)-${suf}:PL"
 },
 {
  "comment": "5-radical type A passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${C2}${C3}${C4}${C5}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., 3rd not y, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., 3rd not y, with suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type A passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type A passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type A passive participle, m. sg., CCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., CCi <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}y\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., CCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., CCi <- CCw, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}w\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type A passive participle, f. sg., CCi-t <- CCw",
```

```
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type A passive participle, m. pl., CCuy-Am <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type A passive participle, f. pl., CCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type A passive participle, m. pl., CCC{+L} // bzH-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[hHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type A passive participle, f. pl., CCC{+L} // bzH-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[hHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type C active participle, m. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>Ay)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\)-${suf}:M.SG"
 },
 {
  "comment": "5-radical type C active participle, m. sg., no suffix",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\).M.SG"
 },
 {
  "comment": "5-radical type C active participle, f. sg.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>Ayt)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\)-${suf}:F.SG"
 },
 {
  "comment": "5-radical type C active participle, pl.",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<s
uf>at)$",
  "replace": "${pref}:PTCP.ACT-${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\)-${suf}:PL"
 },
 {
  "comment": "5-radical type C passive participle, m. sg.",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
```

```
  "replace": "${stem}:\\(${C2}${C3}\\(A\\)${C4}${C5}\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., 3rd not y, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPyhHX><]))$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., 3rd not y, with suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>t)$",
  "replace": "${stem}:\\(${stem}\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type C passive participle, m. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type C passive participle, f. pl., CCC{-L}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])u(?<C5>[lmrsxqbtnkzdGgTCSfcZpPy]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type C passive participle, m. sg., CCuy <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}w\\).PTCP.PASS.M.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., CCi <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}y\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., CCi-t <- CCy",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}y\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., CCi <- CCw, no suffix",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}w\\).PTCP.PASS.F.SG"
 },
 {
  "comment": "5-radical type C passive participle, f. sg., CCi-t <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)(?<suf>t)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}w\\).PTCP.PASS-${suf}:F.SG"
 },
 {
  "comment": "5-radical type C passive participle, m. pl., CCuy-Am <- CCw",
```

```
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}w\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type C passive participle, f. pl., CCuy-At <- CCw",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])uy)(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}w\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "5-radical type C passive participle, m. pl., CCC{+L} // bzH-Am",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[hHX><]))(?<suf>Am)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\).PTCP.PASS-${suf}:M.PL"
 },
 {
  "comment": "5-radical type C passive participle, f. pl., CCC{+L} // bzH-At",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])u(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C5>[hHX><]))(?<suf>At)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\).PTCP.PASS-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{-L}C{-SV} // qAtl-Ayt",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{-L}C{-SV} // qAtl-at",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{-L}C{-SV} // qAtl-Ay",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])(?<C3>[lmrsxqbtnkzdGgTCSfcZp
PhHX><]))(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ayt)
$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$
",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{+L}C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ay)$
",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:AGNZ.M.SG"
 },
 {
```

```
  "comment": "3-radical type A agentive noun, f. sg., CC{-L}y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{-L}y; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{-L}y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{-L}y; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{-L}y",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{-L}y; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{-L}w; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{-L}w; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{-L}w",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><])y)(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg., CC{-L}w; 2nd-3rd rad transposition",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])Ay(?<C2>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))(?<suf>Ay)$",
```

```
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{+L}w",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}w\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Ayt)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.PL"
 },
 {
  "comment": "3-radical type A agentive noun, pl., CC{+L}y",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])y)(?<suf>Ay)$",
  "replace": "${stem}:\\(${C1}${C2}y\\)-${suf}:AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg. // qatAl",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}y\\).AGNZ.M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, m. sg. // qatAl-i",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<suf>i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).AGNZ-${suf}:M.SG"
 },
 {
  "comment": "3-radical type A agentive noun, f. sg. // qatAl-it",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<suf>it)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).AGNZ-${suf}:F.SG"
 },
 {
  "comment": "3-radical type A agentive noun, m. pl. // qatAl-yAm",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<suf>yAm)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).AGNZ-${suf}:M.PL"
 },
 {
  "comment": "3-radical type A agentive noun, f. pl. // qatAl-yAt",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C3>[lmrsxqbtnkzdGgT
CSfcZpPwyhHX><]))(?<suf>yAt)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).AGNZ-${suf}:F.PL"
 },
 {
  "comment": "3-radical type A infinitive: CaC{-L}i?C",
```

```
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwy])i?(?<C3>[lmrsxqbtnkzdGgTCSfc
ZpPwyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).INF"
 },
 {
  "comment": "3-radical type A infinitive: CC{+L}i?C{-SV}",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])i?(?<C3>[lmrsxqbtnkzdGgTCSfcZpPhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).INF"
 },
 {
  "comment": "3-radical type A infinitive: CC{+L}i (i <- y)",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[hHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).INF"
 },
 {
  "comment": "3-radical type A infinitive: CCCat",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><]))(?<suf>at)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type A infinitive: CCCAn",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><]))(?<suf>An)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type A infinitive: CCCo",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><]))(?<suf>o)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type A infinitive: mCCCAl",
  "regex":
"^(?<pref>m)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqb
tnkzdGgTCSfcZpPwyhHX><]))(?<suf>Al)$",
  "replace": "${pref}:INF-${stem}:\\(${C1}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type A infinitive: CCCe",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><]))(?<suf>e)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type A infinitive: CaCi (i <- y)",
  "regex": "^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])i)$",
  "replace": "${stem}:\\(${C1}${C2}y\\).INF"
 },
 {
  "comment": "3-radical type B -ot infinitive",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpP])\\k<C2>(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPhHX><]))(?<suf>ot)$",
  "replace": "${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical type B -ot infinitive, C2 lar/SV",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[wyhHX><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpwyPhHX><]))(?<suf
>ot)$",
  "replace": "${stem}:\\(${C1}${C2}\\(2\\)${C3}\\)-${suf}:INF"
```

```
 },
 {
  "comment": "3,4,5-radical type C -ot infinitive",
  "regex":
"^(?<stem>(?<syl1>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a)?(?<syl2>(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a)?(?
<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C5>[lmrsxqbtnkzdGgTCSfcZpPwyhHX
><]))(?<suf>ot)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\(A\\)${C4}${C5}\\)-${suf}:INF"
 },
 {
  "comment": "4-radical type A -ot infinitive",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])a(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>ot)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\)-${suf}:INF"
 },
 {
  "comment": "4-radical type D -ot infinitive",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A\\k<C2>a(?<C3>[lmrsxqb
tnkzdGgTCSfcZpPwyhHX><]))(?<suf>ot)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).FREQ-${suf}:INF"
 },
 {
  "comment": "5-radical type A -ot infinitive",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmrsxqbtnkzdGgTC
SfcZpPwyhHX><])\\k<C2>a\\k<C3>)(?<suf>ot)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C2}${C3}\\)-${suf}:INF"
 },
 {
  "comment": "3-radical t-A infinitive // ma-t-qallA<",
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhH
X><])\\k<C2>A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}\\).INF"
 },
 {
  "comment": "3-radical t-B infinitive // ma-t-qallA<",
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhH
X><])\\k<C2>A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\).INF"
 },
 {
  "comment": "3,4,5-radical t-C infinitive",
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<syl1>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a)?(?<syl2>(?<C2>[lmrsxqbt
nkzdGgTCSfcZpPwyhHX><])a)?(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C5>
[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}${C5}\\).INF"
 },
 {
  "comment": "4-radical t-D infinitive",
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhH
X><])A\\k<C2>A(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}\\).FREQ.INF"
 },
 {
  "comment": "4-radical t-A infinitive",
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhH
X><])(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C4}\\).INF"
 },
 {
  "comment": "5-radical t-A infinitive",
```

```
  "regex":
"^(?<prefInf>ma)(?<prefPass>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX
><])a(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2>A\\k<C3>)$",
  "replace": "${prefInf}:INF-${prefPass}:PASS-${stem}:\\(${C1}${C2}${C3}${C2}${C3}\\).INF"
 },
 {
  "comment": "4-radical t-A infinitive: CCCACe",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C3>[lmrsxqbtnkzdGgTCS
fcZpPwyhHX><])A(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>e)$",
  "replace": "${stem}:\\(${C1}${C2}${C3}${C4}\\).INF-${suf}:INF"
 },
 {
  "comment": "3-radical A instrument nominalizer // ma-Ca?Ca?C-i",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(\\k<C2>)?a
?(?<C3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>i)$",
  "replace": "${pref}:INSNZ-${stem}:\\(${C1}${C2}${C3}\\).INSNZ-${suf}:INSNZ"
 },
 {
  "comment": "4-radical A instrument nominalizer // ma-Ca?CCa?C-i",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<C3>[lm
rsxqbtnkzdGgTCSfcZpPwyhHX><])(\\k<C3>)?a?(?<C4>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>i)$",
  "replace": "${pref}:INSNZ-${stem}:\\(${C1}${C2}${C3}${C4}\\).INSNZ-${suf}:INSNZ"
 },
 {
  "comment": "5-radical A instrument nominalizer // ma-CCaCCaC-i",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])\\k<C2>a\\k<C3>)(?<suf>i)$",
  "replace": "${pref}:INSNZ-${stem}:\\(${C1}${C2}${C3}${C2}${C3}\\).INSNZ-${suf}:INSNZ"
 },
 {
  "comment": "3-radical B instrument nominalizer // ma-saddaq-i",
  "regex":
"^(?<pref>ma)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2>a(?<C
3>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>i)$",
  "replace": "${pref}:INSNZ-${stem}:\\(${C1}${C2}\\(2\\)${C3}\\).INSNZ-${suf}:INSNZ"
 },
 {
  "comment": "result nominalizer // CC{2}AC",
  "regex":
"^(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])\\k<C2>A(?<C3>[lmrsxqbtn
kzdGgTCSfcZpPwyhHX><]))$",
  "replace": "${stem}:\\(${C1}${C2}${C3}\\).RESNZ"
 },
 {
  "comment": "abstractivizer // CCCC?-nnat?",
  "regex": "^(?<stem>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{3,4})(?<suf>nnat?)$",
  "replace": "${stem}:\\(${stem}\\)-${suf}:ABSZ"
 }
]
==================================== lexicon.json ========================================
```

<u>Примечание</u>: Текст ниже идет <u>в три колонки</u> на каждой странице до конца содержимого файла lexicon.json.

```
[
 {
  "comment": "personal pronoun:
1c sg",
  "regex": "^(?<stem>>ana)$",
  "replace":
"${stem}:PRON.1.SG"
 },
 {
  "comment": "personal pronoun:
2m sg",
  "regex": "^(?<stem>>nta)$",
  "replace":
"${stem}:PRON.2.M.SG"
 },
 {
  "comment": "personal pronoun:
2f sg",
  "regex": "^(?<stem>>nti)$",
  "replace":
"${stem}:PRON.2.F.SG"
 },
 {
  "comment": "personal pronoun:
3m sg",
  "regex": "^(?<stem>htu)$",
  "replace":
"${stem}:PRON.3.M.SG"
 },
 {
  "comment": "personal pronoun:
3f sg",
  "regex": "^(?<stem>hta)$",
  "replace":
"${stem}:PRON.3.F.SG"
```

```
 },
 {
  "comment": "personal pronoun:
1c pl",
  "regex": "^(?<stem>hna)$",
  "replace":
"${stem}:PRON.1.PL"
 },
 {
  "comment": "personal pronoun:
2m pl",
  "regex": "^(?<stem>>ntum)$",
  "replace":
"${stem}:PRON.2.M.PL"
 },
 {
  "comment": "personal pronoun:
2f pl",
  "regex": "^(?<stem>>ntn)$",
  "replace":
"${stem}:PRON.2.F.PL"
 },
 {
  "comment": "personal pronoun:
3m pl",
  "regex": "^(?<stem>htom)$",
  "replace":
"${stem}:PRON.3.M.PL"
 },
 {
  "comment": "personal pronoun:
3f pl",
  "regex": "^(?<stem>htan)$",
  "replace":
"${stem}:PRON.3.F.PL"
 },
 {
  "comment": "reflexive nos",
  "regex": "^(?<stem>nos)$",
  "replace": "${stem}:REFL"
 },
 {
  "comment": "reflexive nos",
  "regex": "^(?<stem>nos)$",
  "replace": "${stem}:REFL"
 },
 {
  "comment": "reciprocal nosnos
(pl)",
  "regex": "^(?<stem>nosnos)$",
  "replace": "${stem}:RECP"
 },
 {
  "comment": "reflexive nafs //
in New Testament",
  "regex": "^(?<stem>nafs)$",
  "replace": "${stem}:REFL"
 },
 {
  "comment": "reflexive ra>as",
  "regex": "^(?<stem>ra>as)$",
  "replace": "${stem}:REFL"
 },
 {
  "comment": "reciprocal Hd",
  "regex": "^(?<stem>Hd)$",
  "replace": "${stem}:RECP"
 },
 {
  "comment": "possessive nAy",
  "regex": "^(?<stem>nAy)$",
  "replace": "${stem}:POSS"
 },
 {
  "comment": "what",
  "regex": "^(?<stem>mi)$",
  "replace": "${stem}:what"
 },
 {
  "comment": "who",
  "regex": "^(?<stem>man)$",
  "replace": "${stem}:who"
 },
 {
  "comment": "which",
  "regex":
"^(?<stem>>ay)(?<suf>i)$",
  "replace": "${stem}:which-
${suf}:M.SG"
 },
 {
  "comment": "which",
  "regex":
"^(?<stem>>ay)(?<suf>a)$",
  "replace": "${stem}:which-
${suf}:F.SG"
 },
 {
  "comment": "which",
  "regex":
"^(?<stem>>ay)(?<suf>om)$",
  "replace": "${stem}:which-
${suf}:M.PL"
 },
 {
  "comment": "which",
  "regex":
"^(?<stem>>ay)(?<suf>an)$",
  "replace": "${stem}:which-
${suf}:F.PL"
 },
 {
  "comment": "someone",
  "regex": "^(?<stem>worot?)$",
  "replace":
"${stem}:someone.M.SG"
 },
 {
  "comment": "someone",
  "regex": "^(?<stem>Hatte)$",
  "replace":
"${stem}:someone.F.SG"
 },
 {
  "comment": "(a) certain",
  "regex": "^(?<stem>>gale)$",
  "replace": "${stem}:such and
such.ANIM"
 },
 {
  "comment": "(a) certain",
  "regex":
"^(?<stem>>gale)(?<suf>tAy)$",
  "replace": "${stem}:such and
such.ANIM-${suf}:M.SG"
 },
 {
  "comment": "(a) certain",
  "regex":
"^(?<stem>>gale)(?<suf>tat)$",
  "replace": "${stem}:such and
such.ANIM-${suf}:F.SG"
 },
 {
  "comment": "(a) certain",
  "regex": "^(?<stem>flAn)$",
  "replace": "${stem}:such and
such.INANIM"
 },
 {
  "comment": "some, something",
  "regex": "^(?<stem>gale)$",
  "replace": "${stem}:some"
 },
 {
  "comment": "something",
  "regex":
"^(?<stem1>gale)(?<stem2>gAr)$"
,
  "replace": "${stem1}:some-
${stem2}:thing"
 },
 {
  "comment": "that which
happened... / in the expression
of -what/whoever-",
  "regex":
"^(?<relz>la)(?<stem>ga?>)(?<su
f>a)$",
  "replace": "${relz}:REL-
${stem}:happen.PERF-
${suf}:3.M.SG"
 },
 {
  "comment": "...shall happen /
in the expression of -
what/whoever-",
  "regex":
"^(?<suf>l)(?<stem>gba>)$",
  "replace": "${suf}:3.M.SG-
${stem}:happen.JUSS"
 },
 {
  "comment": "whatever,
whoever, any",
  "regex": "^(?<stem>dla)$",
  "replace": "${stem}:any"
 },
 {
  "comment": "nothing",
  "regex": "^(?<stem>sema)$",
  "replace": "${stem}:nothing"
 },
 {
  "comment": "this, m.",
  "regex":
"^(?<stem>>ll)(?<suf>i)$",
  "replace": "${stem}:DEM.PROX-
${suf}:M.SG"
 },
 {
  "comment": "this, f.",
  "regex":
"^(?<stem>>ll)(?<suf>a)$",
  "replace": "${stem}:DEM.PROX-
${suf}:F.SG"
 },
```

```
 {
  "comment": "these, m.",
  "regex":
"^(?<stem>>ll)(?<suf>om)$",
  "replace": "${stem}:DEM.PROX-
${suf}:M.PL"
 },
 {
  "comment": "these, f.",
  "regex":
"^(?<stem>>ll)(?<suf>an)$",
  "replace": "${stem}:DEM.PROX-
${suf}:F.PL"
 },
 {
  "comment": "that, m.",
  "regex":
"^(?<stem>l[aoe]h)(?<suf>ay)$",
  "replace": "${stem}:DEM.DIST-
${suf}:M.SG"
 },
 {
  "comment": "that, f.",
  "regex":
"^(?<stem>l[aoe]h)(?<suf>a)$",
  "replace": "${stem}:DEM.DIST-
${suf}:F.SG"
 },
 {
  "comment": "those, m.",
  "regex":
"^(?<stem>l[aoe]h)(?<suf>om)$",
  "replace": "${stem}:DEM.DIST-
${suf}:M.PL"
 },
 {
  "comment": "those, f.",
  "regex":
"^(?<stem>l[aoe]h)(?<suf>an)$",
  "replace": "${stem}:DEM.DIST-
${suf}:F.PL"
 },
 {
  "comment": "copula: 1c sg",
  "regex": "^(?<stem>>ana)$",
  "replace": "${stem}:COP.1.SG"
 },
 {
  "comment": "copula: 2m sg",
  "regex": "^(?<stem>>nta)$",
  "replace":
"${stem}:COP.2.M.SG"
 },
 {
  "comment": "copula: 2f sg",
  "regex": "^(?<stem>>nti)$",
  "replace":
"${stem}:COP.2.F.SG"
 },
 {
  "comment": "copula: 3m sg",
  "regex": "^(?<stem>tu)$",
  "replace":
"${stem}:COP.3.M.SG"
 },
 {
  "comment": "copula: 3f sg",
  "regex": "^(?<stem>ta)$",
  "replace":
"${stem}:COP.3.F.SG"
 },
 {
  "comment": "copula: 1c pl",
  "regex": "^(?<stem>hna)$",
  "replace": "${stem}:COP.1.PL"
 },
 {
  "comment": "copula: 2m pl",
  "regex": "^(?<stem>>ntum)$",
  "replace":
"${stem}:COP.2.M.PL"
 },
 {
  "comment": "copula: 2f pl",
  "regex": "^(?<stem>>ntn)$",
  "replace":
"${stem}:COP.2.F.PL"
 },
 {
  "comment": "copula: 3m pl",
  "regex": "^(?<stem>tom)$",
  "replace":
"${stem}:COP.3.M.PL"
 },
 {
  "comment": "copula: 3f pl",
  "regex": "^(?<stem>tan)$",
  "replace":
"${stem}:COP.3.F.PL"
 },
 {
  "comment": "negative copula 3
m sg (fossilized)",
  "regex": "^(?<stem>>ikon)$",
  "replace":
"${stem}:COP.3.F.PL.NEG"
 },
 {
  "comment": "<ala-copula
(past)",
  "regex":
"^(?<root><al)(?<suf>ko)$",
  "replace": "${root}:COP.PST-
${suf}:1.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>ka)$",
  "replace": "${root}:COP.PST-
${suf}:2.M.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>ki)$",
  "replace": "${root}:COP.PST-
${suf}:2.F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>a)$",
  "replace": "${root}:COP.PST-
${suf}:3.M.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>at)$",
  "replace": "${root}:COP.PST-
${suf}:3.F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>na)$",
  "replace": "${root}:COP.PST-
${suf}:1.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>kum)$",
  "replace": "${root}:COP.PST-
${suf}:2.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>kn)$",
  "replace": "${root}:COP.PST-
${suf}:2.F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>aw)$",
  "replace": "${root}:COP.PST-
${suf}:2.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root><al)(?<suf>aya)$",
  "replace": "${root}:COP.PST-
${suf}:2.F.PL"
 },
 {
  "comment": "gab>a copula
(non-past)",
  "regex":
"^(?<pref>>)(?<root>gabb>?)$",
  "replace": "${pref}:1.SG-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabb>?)$",
  "replace": "${pref}:2.M.SG-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabbi>?)$",
  "replace": "${pref}:2.F.SG-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabb>)(?<su
f>i)$",
```

```
  "replace": "${pref}:2-
${root}:COP.NPST-${suf}:F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gabb>?)$",
  "replace": "${pref}:3.M.SG-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabb>?)$",
  "replace": "${pref}:3.F.SG-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>>n)(?<root>gabb>?)$",
  "replace": "${pref}:1.PL-
${root}:COP.NPST"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabbu>?)$",
  "replace": "${pref}:2-
${root}:COP.NPST.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabb>)(?<su
f>u)$",
  "replace": "${pref}:2-
${root}:COP.NPST-${suf}:M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabba>?)$",
  "replace": "${pref}:2-
${root}:COP.NPST.F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gabb>)(?<su
f>a)$",
  "replace": "${pref}:2-
${root}:COP.NPST-${suf}:F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gabbu>?)$",
  "replace": "${pref}:3-
${root}:COP.NPST.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gabb>)(?<su
f>u)$",
  "replace": "${pref}:3-
${root}:COP.NPST-${suf}:M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gabba>?)$",
  "replace": "${pref}:3-
${root}:COP.NPST.F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gabb>)(?<su
f>a)$",
  "replace": "${pref}:3-
${root}:COP.NPST-${suf}:F.PL"
 },
 {
  "comment": "gab>a copula
(non-past), jussive",
  "regex":
"^(?<pref>>)(?<root>gba>?)$",
  "replace": "${pref}:1.SG-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gba>?)",
  "replace": "${pref}:2.M.SG-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gbi?>?)",
  "replace": "${pref}:2.F.SG-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gba>?)",
  "replace": "${pref}:3.M.SG-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gba>?)",
  "replace": "${pref}:3.F.SG-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>n)(?<root>gba>?)$",
  "replace": "${pref}:1.PL-
${root}:COP.JUSS"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gbu>?)",
  "replace": "${pref}:2-
${root}:COP.JUSS.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gb>)(?<suf>
u)$",
  "replace": "${pref}:2-
${root}:COP.JUSS-${suf}:M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gb>)(?<suf>
a)$",
  "replace": "${pref}:2-
${root}:COP.JUSS-${suf}:F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>t)(?<root>gb>)(?<suf>
u)$",
  "replace": "${pref}:2-
${root}:COP.JUSS-${suf}:F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gbu>?)$",
  "replace": "${pref}:3-
${root}:COP.JUSS.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gb>)(?<suf>
u)$",
  "replace": "${pref}:3-
${root}:COP.JUSS-${suf}:M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<pref>l)(?<root>gb>)(?<suf>
a)$",
  "replace": "${pref}:3-
${root}:COP.JUSS-${suf}:F.PL"
 },
 {
  "comment": "gab>a copula
(non-past), imperative",
  "regex": "^(?<root>gba>)$",
  "replace":
"${root}:COP.IMP.M.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>gb>)(?<suf>i)$",
  "replace": "${root}:COP.IMP-
${suf}:F.SG"
 },
 {
  "comment": "",
  "regex": "^(?<root>gbi>)$",
  "replace":
"${root}:COP.IMP.F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>gb>)(?<suf>u)$",
```

```
  "replace": "${root}:COP.IMP-
${suf}:M.PL"
 },
 {
  "comment": "",
  "regex": "^(?<root>gbu>)$",
  "replace":
"${root}:COP.IMP.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root>gb>)(?<suf>a)$",
  "replace": "${root}:COP.IMP-
${suf}:F.PL"
 },
 {
  "comment": "ttu
(subordinating copula)",
  "regex":
"^(?<pref>t)(?<part>tu)$",
  "replace": "${pref}:SUBORD-
${part}:COP.3.SG"
 },
 {
  "comment": "halla existential
(prf forms, npst meaning)",
  "regex":
"^(?<root>halle)(?<suf>ko)$",
  "replace": "${root}:EXIST-
${suf}:1.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>ka)$",
  "replace": "${root}:EXIST-
${suf}:2.M.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>ki)$",
  "replace": "${root}:EXIST-
${suf}:2.F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>hall)(?<suf>a)$",
  "replace": "${root}:EXIST-
${suf}:3.M.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>t)$",
  "replace": "${root}:EXIST-
${suf}:3.F.SG"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>na)$",
  "replace": "${root}:EXIST-
${suf}:1.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>kum)$",
  "replace": "${root}:EXIST-
${suf}:2.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root>halle)(?<suf>kn)$",
  "replace": "${root}:EXIST-
${suf}:2.F.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root>hall)(?<suf>aw)$",
  "replace": "${root}:EXIST-
${suf}:2.M.PL"
 },
 {
  "comment": "",
  "regex":
"^(?<root>hall)(?<suf>aya)$",
  "replace": "${root}:EXIST-
${suf}:2.F.PL"
 },
 {
  "comment": "negation of
existential (there isn't),
suppletive",
  "regex": "^(?<stem>>alabu)$",
  "replace":
"${stem}:EXIST.3.M.SG.NEG"
 },
 {
  "comment": "when",
  "regex": "^(?<stem>>t)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "when",
  "regex": "^(?<stem>dib)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "if",
  "regex": "^(?<stem>mn)$",
  "replace": "${stem}:if"
 },
 {
  "comment": "have (b +
possessive suffixes)",
  "regex": "^(?<stem>b)$",
  "replace": "${stem}:have"
 },
 {
  "comment": "have (b +
possessive suffixes)",
  "regex": "^(?<stem>bdib)$",
  "replace": "${stem}:have"
 },
 {
  "comment": ">gl
subordinator",
  "regex": "^(?<stem>>gl)$",
  "replace": "${stem}:SUBORD"
 },
 {
  "comment": ">gl to",
  "regex": "^(?<stem>>gl)$",
  "replace": "${stem}:to"
 },
 {
  "comment": "numerals",
  "regex": "^(?<stem>worot?)$",
  "replace": "${stem}:one.M"
 },
 {
  "comment": "",
  "regex":
"^(?<stem>worworot)$",
  "replace":
"${stem}:one.M.DISTR"
 },
 {
  "comment": "",
  "regex": "^(?<stem>Hatte)$",
  "replace": "${stem}:one.F"
 },
 {
  "comment": "",
  "regex": "^(?<stem>salas)$",
  "replace": "${stem}:three"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>arba<)$",
  "replace": "${stem}:four"
 },
 {
  "comment": "",
  "regex": "^(?<stem>Hams)$",
  "replace": "${stem}:five"
 },
 {
  "comment": "",
  "regex": "^(?<stem>ss)$",
  "replace": "${stem}:six"
 },
 {
  "comment": "",
  "regex": "^(?<stem>sabu<)$",
  "replace": "${stem}:seven"
 },
 {
  "comment": "",
  "regex": "^(?<stem>samAn)$",
  "replace": "${stem}:eight"
 },
 {
  "comment": "",
  "regex": "^(?<stem>s<)$",
  "replace": "${stem}:nine"
 },
 {
  "comment": "",
  "regex": "^(?<stem><asr)$",
  "replace": "${stem}:ten"
 },
 {
  "comment": "",
  "regex": "^(?<stem><sra)$",
  "replace": "${stem}:twenty"
 },
 {
  "comment": "",
  "regex": "^(?<stem>salAsa)$",
  "replace": "${stem}:thirty"
```

```
 },
 {
  "comment": "",
  "regex": "^(?<stem>>arb<a)$",
  "replace": "${stem}:forty"
 },
 {
  "comment": "",
  "regex": "^(?<stem>Hmsa)$",
  "replace": "${stem}:fifty"
 },
 {
  "comment": "",
  "regex": "^(?<stem>ssa)$",
  "replace": "${stem}:sixty"
 },
 {
  "comment": "",
  "regex": "^(?<stem>sab<a)$",
  "replace": "${stem}:seventy"
 },
 {
  "comment": "",
  "regex":
"^(?<stem>samAnya)$",
  "replace": "${stem}:eighty"
 },
 {
  "comment": "",
  "regex": "^(?<stem>sa<a)$",
  "replace": "${stem}:ninety"
 },
 {
  "comment": "",
  "regex": "^(?<stem>tas<a)$",
  "replace": "${stem}:ninety"
 },
 {
  "comment": "",
  "regex": "^(?<stem>m>t)$",
  "replace": "${stem}:hundred"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>am>At)$",
  "replace":
"${stem}:hundred.PL"
 },
 {
  "comment": "",
  "regex": "^(?<stem>xH)$",
  "replace": "${stem}:thousand"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>axHat)$",
  "replace":
"${stem}:thousand.PL"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>alf)$",
  "replace": "${stem}:thousand"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>AlAf)$",
  "replace":
"${stem}:thousand.PL"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>lf)$",
  "replace": "${stem}:ten
thousand"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>lf)$",
  "replace": "${stem}:a great
many"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>AlAf)$",
  "replace": "${stem}:tens of
thousands"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>AlAf)$",
  "replace": "${stem}:a great
many"
 },
 {
  "comment": "Had (around,
about)",
  "regex": "^(?<stem>Had)$",
  "replace": "${stem}:APPROX"
 },
 {
  "comment": "<adad: number",
  "regex": "^(?<stem><adad)$",
  "replace": "${stem}:number"
 },
 {
  "comment": "<adad: with
aggregate numerals",
  "regex": "^(?<stem><adad)$",
  "replace": "${stem}:AGGR"
 },
 {
  "comment": "nafar appellative
(this -one-)",
  "regex": "^(?<stem>nafar)$",
  "replace":
"${stem}:APPEL.ANIM"
 },
 {
  "comment": "nafar appellative
(this -one-)",
  "regex": "^(?<stem>>anfAr)$",
  "replace":
"${stem}:APPEL.ANIM.PL"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>kAl>)$",
  "replace":
"${stem}:second.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>kAl>)(?<suf>Ay)$",
  "replace": "${stem}:second-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>kAl>)(?<suf>Ayt)$",
  "replace": "${stem}:second-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>sAls)$",
  "replace":
"${stem}:third.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAls)(?<suf>Ay)$",
  "replace": "${stem}:third-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAls)(?<suf>Ayt)$",
  "replace": "${stem}:third-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>>Arb)$",
  "replace":
"${stem}:fourth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>>Arb)(?<suf>Ay)$",
  "replace": "${stem}:fourth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>>Arb)(?<suf>Ayt)$",
  "replace": "${stem}:fourth-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>sAls)$",
  "replace":
"${stem}:fifth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAls)(?<suf>Ay)$",
  "replace": "${stem}:fifth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAls)(?<suf>Ayt)$",
  "replace": "${stem}:fifth-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
```

```
  "regex": "^(?<stem>sAds)$",
  "replace":
"${stem}:sixth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAds)(?<suf>Ay)$",
  "replace": "${stem}:sixth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAds)(?<suf>Ayt)$",
  "replace": "${stem}:sixth-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>sAb>)$",
  "replace":
"${stem}:seventh.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAb>)(?<suf>Ay)$",
  "replace": "${stem}:seventh-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAb>)(?<suf>Ayt)$",
  "replace": "${stem}:seventh-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>sAmn)$",
  "replace":
"${stem}:eighth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAmn)(?<suf>Ay)$",
  "replace": "${stem}:eighth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>sAmn)(?<suf>Ayt)$",
  "replace": "${stem}:eighth-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem>tAs<)$",
  "replace":
"${stem}:ninth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>tAs<)(?<suf>Ay)$",
  "replace": "${stem}:ninth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem>tAs<)(?<suf>Ayt)$",
  "replace": "${stem}:ninth-
${suf}:FEM"
 },
 {
  "comment": "ordinals",
  "regex": "^(?<stem><Asr)$",
  "replace":
"${stem}:tenth.MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem><Asr)(?<suf>Ay)$",
  "replace": "${stem}:tenth-
${suf}:MASC"
 },
 {
  "comment": "ordinals",
  "regex":
"^(?<stem><Asr)(?<suf>Ayt)$",
  "replace": "${stem}:tenth-
${suf}:FEM"
 },
 {
  "comment": "half",
  "regex": "^(?<stem>sar)$",
  "replace": "${stem}:half"
 },
 {
  "comment": "one third",
  "regex":
"^(?<stem>masallas)$",
  "replace": "${stem}:third
part"
 },
 {
  "comment": "one fourth",
  "regex": "^(?<stem>rb<a?)$",
  "replace": "${stem}:fourth
part"
 },
 {
  "comment": "after",
  "regex": "^(?<stem>Haqo)$",
  "replace": "${stem}:after"
 },
 {
  "comment": "under",
  "regex":
"^(?<stem>Han(te)?)$",
  "replace": "${stem}:after"
 },
 {
  "comment": "with",
  "regex": "^(?<stem>msl)$",
  "replace": "${stem}:COM"
 },
 {
  "comment": "above, over",
  "regex":
"^(?<stem>maxanqal)$",
  "replace": "${stem}:above"
 },
 {
  "comment": "near",
  "regex": "^(?<stem>maTor)$",
  "replace": "${stem}:near"
 },
 {
  "comment": "for the sake of",
  "regex": "^(?<stem>matAn)$",
  "replace": "${stem}:for the
sake of"
 },
 {
  "comment": "than (in
comparison)",
  "regex": "^(?<stem>mn)$",
  "replace": "${stem}:than"
 },
 {
  "comment": "head",
  "regex": "^(?<stem>ra>as)$",
  "replace": "${stem}:head"
 },
 {
  "comment": "at",
  "regex": "^(?<stem>>t)$",
  "replace": "${stem}:LOC"
 },
 {
  "comment": "because of",
  "regex": "^(?<stem>sabbat)$",
  "replace": "${stem}:because
of"
 },
 {
  "comment": "towards",
  "regex":
"^(?<stem>xan(kat|nak))$",
  "replace": "${stem}:towards"
 },
 {
  "comment": "side",
  "regex": "^(?<stem>smT)$",
  "replace": "${stem}:side"
 },
 {
  "comment": "back",
  "regex": "^(?<stem>raHar)$",
  "replace": "${stem}:back"
 },
 {
  "comment": "midst",
  "regex": "^(?<stem>mgb)$",
  "replace": "${stem}:midst"
 },
 {
  "comment": "front",
  "regex": "^(?<stem>qAbl)$",
  "replace": "${stem}:front"
 },
 {
  "comment": "front",
  "regex":
"^(?<stem>qbl)(?<suf>At)$",
  "replace": "${stem}:midst-
${suf}F.SG"
 },
 {
  "comment": "without, except",
```

```
  "regex": "^(?<stem>>mbal)$",
  "replace": "${stem}:without"
 },
 {
  "comment": "near",
  "regex": "^(?<stem>>raf)$",
  "replace": "${stem}:near"
 },
 {
  "comment": "after",
  "regex": "^(?<stem>>sar)$",
  "replace": "${stem}:after"
 },
 {
  "comment": "towards",
  "regex": "^(?<stem>>ask)$",
  "replace": "${stem}:towards"
 },
 {
  "comment": "in; about,
with...",
  "regex": "^(?<stem>>b)$",
  "replace": "${stem}:in"
 },
 {
  "comment": "have",
  "regex": "^(?<stem>b>tt?)$",
  "replace": "${stem}:have"
 },
 {
  "comment": "in, etc.",
  "regex": "^(?<stem>>?tt?)$",
  "replace": "${stem}:in"
 },
 {
  "comment": "as (in
comparison)",
  "regex": "^(?<stem>>akl)$",
  "replace": "${stem}:as"
 },
 {
  "comment": "place",
  "regex": "^(?<stem>>akAn)$",
  "replace": "${stem}:place"
 },
 {
  "comment": "place",
  "regex": "^(?<stem>>zAm)$",
  "replace": "${stem}:place"
 },
 {
  "comment": "mouth",
  "regex": "^(?<stem>>af)$",
  "replace": "${stem}:mouth"
 },
 {
  "comment": "as",
  "regex": "^(?<stem>km)$",
  "replace": "${stem}:as"
 },
 {
  "comment": "as",
  "regex": "^(?<stem>kmsal)$",
  "replace": "${stem}:as"
 },
 {
  "comment": "such as",
  "regex": "^(?<stem>kara)$",
  "replace": "${stem}:such as"
 },
 {
  "comment": "inside",
  "regex": "^(?<stem>kabd)$",
  "replace": "${stem}:inside"
 },
 {
  "comment": "according to",
  "regex": "^(?<stem><adad)$",
  "replace": "${stem}:according
to"
 },
 {
  "comment": "",
  "regex": "^(?<stem>darb)$",
  "replace": "${stem}:behind"
 },
 {
  "comment": "",
  "regex": "^(?<stem>dib)$",
  "replace": "${stem}:to"
 },
 {
  "comment": "",
  "regex": "^(?<stem>dwAr)$",
  "replace":
"${stem}:surroundings"
 },
 {
  "comment": "",
  "regex": "^(?<stem>GallAb)$",
  "replace": "${stem}:for sake"
 },
 {
  "comment": "",
  "regex": "^(?<stem>fza)$",
  "replace": "${stem}:for sake"
 },
 {
  "comment": "",
  "regex": "^(?<stem>grra)$",
  "replace": "${stem}:back"
 },
 {
  "comment": "",
  "regex": "^(?<stem>gor?)$",
  "replace": "${stem}:with"
 },
 {
  "comment": "",
  "regex": "^(?<stem>gdor)$",
  "replace": "${stem}:with"
 },
 {
  "comment": "",
  "regex": "^(?<stem>gabay)$",
  "replace": "${stem}:way"
 },
 {
  "comment": "",
  "regex":
"^(?<stem>[fm]nge)$",
  "replace": "${stem}:between"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>ndo)$",
  "replace": "${stem}:while"
 },
 {
  "comment": "",
  "regex": "^(?<stem>ha?ye)$",
  "replace": "${stem}:also"
 },
 {
  "comment": "",
  "regex": "^(?<stem>lAma)$",
  "replace": "${stem}:also"
 },
 {
  "comment": "",
  "regex": "^(?<stem>lAta)$",
  "replace": "${stem}:but"
 },
 {
  "comment": "",
  "regex": "^(?<stem>leTa)$",
  "replace": "${stem}:but"
 },
 {
  "comment": "",
  "regex": "^(?<stem>m?dol)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "",
  "regex": "^(?<stem>ma>aze)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>ttaya)$",
  "replace": "${stem}:where"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>aya)$",
  "replace": "${stem}:where"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>xw)$",
  "replace": "${stem}:where"
 },
 {
  "comment": "",
  "regex": "^(?<stem>ka>afo)$",
  "replace": "${stem}:how"
 },
 {
  "comment": "",
  "regex": "^(?<stem>kam)$",
  "replace": "${stem}:how much"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>afo)$",
  "replace": "${stem}:yes"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>Abe)$",
  "replace": "${stem}:yes"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>ifAlu)$",
  "replace": "${stem}:no"
```

```
 },
 {
  "comment": "",
  "regex": "^(?<stem>ma)$",
 {
  "comment": "",
  "regex": "^(?<stem>da>am)$",
  "replace": "${stem}:but"
 },
 {
  "comment": "",
  "regex":
"^(?<stem>da?>ikon)$",
  "replace": "${stem}:but"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>bmi)$",
  "replace": "${stem}:because"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>glmi)$",
  "replace": "${stem}:or"
 },
 {
  "comment": "",
  "replace": "${stem}:because"
 },
 {
  "comment": "",
  "regex": "^(?<stem>mn)$",
  "replace": "${stem}:since"
 },
 {
  "comment": "",
  "regex": "^(?<stem>sabbat)$",
  "replace": "${stem}:because"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>ndo)$",
  "replace": "${stem}:after"
 },
 {
  "comment": "",
  "regex": "^(?<stem>wok)$",
  "replace": "${stem}:or"
 },

  "regex": "^(?<stem>>nday)$",
  "replace": "${stem}:before"
 },
 {
  "comment": "",
  "regex": "^(?<stem>>wAn)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "",
  "regex": "^(?<stem>km)$",
  "replace": "${stem}:when"
 },
 {
  "comment": "",
  "regex": "^(?<stem>wakd)$",
  "replace": "${stem}:when"
 }
]
```

```
============================== pass-ptcp-deriv-pref.json ================================
[
    {
        "comment": ">a-derivatives pass ptcp: detach prefix: >; all potential passive participles in stem",
        "regex":
"^(?<pref>>)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C3>[lmrsx
qbtnkzdGgTCSfcZpPwyhHX><])u?(?<C4cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(\\k<C4cons>)?u?(?<C5AndI>[lmrsxqbtnkzdG
gTCSfcZpPwyhHX><i])(?<suf>t|Am|At)?)$",
        "replace": "${pref}:CAUS-\\[${stem}\\]:#"
    },
    {
        "comment": ">at-derivatives pass ptcp: detach prefix: >t",
        "regex":
"^(?<pref>>t)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C3>[lmrs
xqbtnkzdGgTCSfcZpPwyhHX><])u?(?<C4cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(\\k<C4cons>)?u?(?<C5AndI>[lmrsxqbtnkzd
GgTCSfcZpPwyhHX><i])(?<suf>t|Am|At)?)$",
        "replace": "${pref}:CAUS-\\[${stem}\\]:#"
    },
    {
        "comment": ">atta-derivatives pass ptcp: detach prefix: >tt",
        "regex":
"^(?<pref>>tt)(?<stem>(?<C1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])?(?<C3>[lmr
sxqbtnkzdGgTCSfcZpPwyhHX><])u?(?<C4cons>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(\\k<C4cons>)?u?(?<C5AndI>[lmrsxqbtnkz
dGgTCSfcZpPwyhHX><i])(?<suf>t|Am|At)?)$",
        "replace": "${pref}:FCT-\\[${stem}\\]:#"
    },
    {
        "comment": ">a-derivatives -ot inf: detach prefix: >a",
        "regex": "^(?<pref>>a)(?<stem>(?:[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][aA]?){2,5}(?<suf>ot))$",
        "replace": "${pref}:CAUS-\\[${stem}\\]:#"
    },
    {
        "comment": ">at-derivatives -ot inf: detach prefix: >at",
        "regex": "^(?<pref>>at)(?<stem>(?:[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][aA]?){2,5}(?<suf>ot))$",
        "replace": "${pref}:CAUS-\\[${stem}\\]:#"
    },
    {
        "comment": ">atta-derivatives -ot inf: detach prefix: >atta",
        "regex": "^(?<pref>>atta)(?<stem>(?:[lmrsxqbtnkzdGgTCSfcZpPwyhHX><][aA]?){2,5}(?<suf>ot))$",
        "replace": "${pref}:FCT-\\[${stem}\\]:#"
    }
]
```

```
============================== pau_2.json ==================================
[
    {
        "comment": "paucative (broken plural as base), ..[ieuo]?C + Am",
        "regex": "^(?<stem>.+[ieuo]?[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>Am)$",
        "replace": "\\[${stem}\\]:#-${suf}:PAU.M"
    },
    {
        "comment": "paucative (broken plural as base), ..[ieuo]?C + At",
        "regex": "^(?<stem>.+[ieuo]?[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf>At)$",
        "replace": "\\[${stem}\\]:#-${suf}:PAU.F"
    },
    {
        "comment": "paucative (broken plural as base), ..aC -> ..eCAm",
        "regex": "^(?<stem>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
        "replace": "\\[${g1}a${C}\\]:#-${suf}:PAU.M"
    },
    {
        "comment": "paucative (broken plural as base), ..aC -> ..eCAt",
        "regex": "^(?<stem>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
        "replace": "\\[${g1}a${C}\\]:#-${suf}:PAU.F"
    },
    {
        "comment": "paucative (broken plural as base), ..AC -> ..eCAm",
        "regex": "^(?<stem>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>Am)$",
        "replace": "\\[${g1}A${C}\\]:#-${suf}:PAU.M"
    },
    {
        "comment": "paucative (broken plural as base), ..AC -> ..eCAt",
        "regex": "^(?<stem>(?<g1>.+)e(?<C>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))(?<suf>At)$",
        "replace": "\\[${g1}A${C}\\]:#-${suf}:PAU.F"
    }
]

============================== pref_coord.json ==================================
[
    {
        "comment": "coord (and)",
        "regex": "^(?<coord>wa)(?<stem>.+)$",
        "replace": "${coord}:COORD-\\[${stem}\\]:#"
    },
    {
        "comment": "coord (and, so, therefore, then)",
        "regex": "^(?<coord>ka)(?<stem>.+)$",
        "replace": "${coord}:COORD-\\[${stem}\\]:#"
    },
    {
        "comment": "subord (that)",
        "regex": "^(?<coord>km)(?<stem>.+)$",
        "replace": "${coord}:SUBORD-\\[${stem}\\]:#"
    }
]
============================== pref_lobj.json ==================================
[
    {
        "comment": "to, for (indirect object)",
        "regex": "^(?<art>>l)(?<stem>.+)$",
        "replace": "${art}:IO-\\[${stem}\\]:#"
    }
]
============================== pref_neg.json ==================================
[
    {
        "comment": "negation",
        "regex": "^(?<art>>i)(?<stem>.+)$",
        "replace": "${art}:NEG-\\[${stem}\\]:#"
    }
]
```

```
============================== pref_relz.json ==================================
[
    {
        "comment": "definite article",
        "regex": "^(?<art>la)(?<stem>.+)$",
        "replace": "${art}:DEF-\\[${stem}\\]:#"
    },
    {
        "comment": "relativizer",
        "regex": "^(?<art>la)(?<stem>.+)$",
        "replace": "${art}:REL-\\[${stem}\\]:#"
    }
]
============================== suf_expl.json ==================================
[
    {
        "comment": "-ma (even, also)",
        "regex": "^(?<stem>.+)(?<suf>ma)$",
        "replace": "\\[${stem}\\]:#-${suf}:also"
    },
    {
        "comment": "-di (indeed)",
        "regex": "^(?<stem>.+)(?<suf>di)$",
        "replace": "\\[${stem}\\]:#-${suf}:indeed"
    }
]
============================== suf_pron.json ==================================
```

<u>Примечание</u>: Текст ниже идет <u>в две колонки</u> на каждой странице до конца содержимого файла suf_pron.json.

```
[
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>.*[^tdTsxzGCScZ])(?<suf>ye)$",
        "replace": "\\[${stem}\\]:#-
${suf}:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)cc)(?<suf>e)$",
        "replace": "\\[${g1}c\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)cc)(?<suf>e)$",
        "replace": "\\[${g1}t\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)GG)(?<suf>e)$",
        "replace": "\\[${g1}d\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)GG)(?<suf>e)$",
        "replace": "\\[${g1}g\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)CC)(?<suf>e)$",
        "replace": "\\[${g1}T\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)CC)(?<suf>e)$",
        "replace": "\\[${g1}S\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)CC)(?<suf>e)$",
        "replace": "\\[${g1}C\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)xx)(?<suf>e)$",
        "replace": "\\[${g1}s\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)xx)(?<suf>e)$",
        "replace": "\\[${g1}x\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)ZZ)(?<suf>e)$",
        "replace": "\\[${g1}z\\]:#-ye:POSS.1.SG"
    },
    {
```

```
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)ZZ)(?<suf>e)$",
        "replace": "\\[${g1}Z\\]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 1 SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)(?<!a[Cy])t)(?<suf>e)$",
        "replace": "\\[${g1}]:#-ye:POSS.1.SG"
    },
    {
        "comment": "POSS 2 M SG for nouns,
particles",
        "regex": "^(?<stem>.+)(?<suf>ka)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.2.M.SG"
    },
    {
        "comment": "POSS 2 F SG for nouns,
particles",
        "regex": "^(?<stem>.+)(?<suf>ki)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.2.M.SG"
    },
    {
        "comment": "POSS 3 M SG for nouns,
particles",
        "regex":
"^(?<stem>.*[^ieaAuo])(?<suf>u)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.3.M.SG"
    },
    {
        "comment": "POSS 3 M SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+[ieAuo])h)(?<suf>u)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.SG"
    },
    {
        "comment": "POSS 3 M SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]
h)(?<suf>u)$",
        "replace": "\\[${g1}a]:#-
${suf}:POSS.3.M.SG"
    },
    {
        "comment": "POSS 3 M SG for nouns,
particles (>aCCC-tu)",
        "regex":
"^(?<stem>(?<g1>>a[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{
3})t)(?<suf>u)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.SG"
    },
    {
        "comment": "POSS 3 M SG for nouns,
particles (Ca?C geminated)",
        "regex":
"^(?<stem>(?<g1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]a?(
?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))\\k<C2>)(?<s
uf>u)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.SG"
    },
    {
        "comment": "POSS 3 F SG for nouns,
particles",
        "regex":
"^(?<stem>.*[^ieaAuo])(?<suf>a)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.3.F.SG"
    },
    {
        "comment": "POSS 3 F SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+[ieAuo])h)(?<suf>a)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.SG"
    },
    {
        "comment": "POSS 3 F SG for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]
h)(?<suf>a)$",
        "replace": "\\[${g1}a]:#-
${suf}:POSS.3.F.SG"
    },
    {
        "comment": "POSS 3 F SG for nouns,
particles (>aCCC-ta)",
        "regex":
"^(?<stem>(?<g1>>a[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{
3})t)(?<suf>a)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.SG"
    },
    {
        "comment": "POSS 3 F SG for nouns,
particles (Ca?C geminated)",
        "regex":
"^(?<stem>(?<g1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]a?(
?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))\\k<C2>)(?<s
uf>a)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.SG"
    },
    {
        "comment": "POSS 3 M PL for nouns,
particles",
        "regex":
"^(?<stem>.*[^ieaAuo])(?<suf>om)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.3.M.PL"
    },
    {
        "comment": "POSS 3 M PL for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+[ieAuo])h)(?<suf>om)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.PL"
    },
    {
        "comment": "POSS 3 M PL for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]
h)(?<suf>om)$",
```

```
        "replace": "\\[${g1}a]:#-
${suf}:POSS.3.M.PL"
    },
    {
        "comment": "POSS 3 M PL for nouns,
particles (>aCCC-tom)",
        "regex":
"^(?<stem>(?<g1>>a[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{
3})t)(?<suf>om)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.PL"
    },
    {
        "comment": "POSS 3 M PL for nouns,
particles (Ca?C geminated)",
        "regex":
"^(?<stem>(?<g1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]a?(
?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))\\k<C2>)(?<s
uf>om)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.M.PL"
    },
    {
        "comment": "POSS 3 F PL for nouns,
particles",
        "regex":
"^(?<stem>.*[^ieaAuo])(?<suf>an)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.3.F.PL"
    },
    {
        "comment": "POSS 3 F PL for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+[ieAuo])h)(?<suf>an)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.PL"
    },
    {
        "comment": "POSS 3 F PL for nouns,
particles",
        "regex":
"^(?<stem>(?<g1>.+)[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]
h)(?<suf>an)$",
        "replace": "\\[${g1}a]:#-
${suf}:POSS.3.F.PL"
    },
    {
        "comment": "POSS 3 F PL for nouns,
particles (>aCCC-tan)",
        "regex":
"^(?<stem>(?<g1>>a[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]{
3})t)(?<suf>an)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.PL"
    },
    {
        "comment": "POSS 3 F PL for nouns,
particles (Ca?C geminated)",
        "regex":
"^(?<stem>(?<g1>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]a?(
?<C2>[lmrsxqbtnkzdGgTCSfcZpPwyhHX><]))\\k<C2>)(?<s
uf>an)$",
        "replace": "\\[${g1}]:#-
${suf}:POSS.3.F.PL"
    },
    {
        "comment": "POSS 1 PL for nouns,
particles",
        "regex": "^(?<stem>.+)(?<suf>na)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.1.PL"
    },
    {
        "comment": "POSS 2 M SG for nouns,
particles",
        "regex": "^(?<stem>.+)(?<suf>kum)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.2.M.SG"
    },
    {
        "comment": "POSS 2 F SG for nouns,
particles",
        "regex": "^(?<stem>.+)(?<suf>kn)$",
        "replace": "\\[${stem}]:#-
${suf}:POSS.2.F.SG"
    },
    {
        "comment": "OBJ 1 SG for verbs",
        "regex": "^(?<stem>.+)(?<suf>n?ni)$",
        "replace": "\\[${stem}]:#-${suf}:OBJ.1.SG"
    },
    {
        "comment": "OBJ 2 m sg for verbs",
        "regex": "^(?<stem>.+)(?<suf>k?ka)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.2.M.SG"
    },
    {
        "comment": "OBJ 2 f sg for verbs",
        "regex": "^(?<stem>.+)(?<suf>k?ki)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.2.F.SG"
    },
    {
        "comment": "OBJ 3 m sg for verbs",
        "regex":
"^(?<stem>.+)(?<suf>o|[wy][ou])$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.M.SG"
    },
    {
        "comment": "OBJ 3 m sg for verbs,
impf/juss/imp e -> 0/a",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<s
uf>yo)$",
        "replace": "\\[${stem}]e:#-
${suf}:OBJ.3.M.SG"
    },
    {
        "comment": "OBJ 3 m sg for verbs, impf u -
> 0",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf
>wo)$",
        "replace": "\\[${stem}]u:#-
${suf}:OBJ.3.M.SG"
    },
    {
        "comment": "OBJ 3 m sg for verbs, a -> 0
|| _ h",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<su
f>ho)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.M.SG"
    },
```

```
    {
        "comment": "OBJ 3 f sg for verbs",
        "regex": "^(?<stem>.+)(?<suf>[why]?a)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.F.SG"
    },
    {
        "comment": "OBJ 3 f sg for verbs,
impf/juss/imp e -> 0/a",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<s
uf>ya)$",
        "replace": "\\[${stem}]e:#-
${suf}:OBJ.3.F.SG"
    },
    {
        "comment": "OBJ 3 f sg for verbs, impf u -
> 0",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf
>wa)$",
        "replace": "\\[${stem}]u:#-
${suf}:OBJ.3.F.SG"
    },
    {
        "comment": "OBJ 3 f sg for verbs, ah -> 0h
",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<su
f>ha)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.F.SG"
    },
    {
        "comment": "OBJ 1 pl for verbs",
        "regex": "^(?<stem>.+)(?<suf>n?na)$",
        "replace": "\\[${stem}]:#-${suf}:OBJ.1.PL"
    },
    {
        "comment": "OBJ 2 m pl for verbs",
        "regex": "^(?<stem>.+)(?<suf>k?kum)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.2.M.PL"
    },
    {
        "comment": "OBJ 2 f pl for verbs",
        "regex": "^(?<stem>.+)(?<suf>k?kn)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.2.F.PL"
    },
    {
        "comment": "OBJ 3 m pl for verbs",
        "regex": "^(?<stem>.+)(?<suf>[wyh]?om)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.M.PL"
    },
    {
        "comment": "OBJ 3 m pl for verbs, impf u -
> 0",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf
>wom)$",
        "replace": "\\[${stem}]u:#-
${suf}:OBJ.3.M.PL"
    },
    {
        "comment": "OBJ 3 m pl for verbs, ah ->
0h",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<su
f>hom)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.M.PL"
    },
    {
        "comment": "OBJ 3 m pl for verbs,
impf/juss/imp e -> 0/a",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<s
uf>yom)$",
        "replace": "\\[${stem}]e:#-
${suf}:OBJ.3.M.PL"
    },
    {
        "comment": "OBJ 3 f pl for verbs",
        "regex": "^(?<stem>.+)(?<suf>[wyh]?an)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.F.PL"
    },
    {
        "comment": "OBJ 3 f pl for verbs,
impf/juss/imp e -> 0/a",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a?(?<s
uf>yan)$",
        "replace": "\\[${stem}]e:#-
${suf}:OBJ.3.F.PL"
    },
    {
        "comment": "OBJ 3 f pl for verbs, impf u -
> 0",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])(?<suf
>wan)$",
        "replace": "\\[${stem}]u:#-
${suf}:OBJ.3.F.PL"
    },
    {
        "comment": "OBJ 3 f pl for verbs, ah ->
0h",
        "regex":
"^(?<stem>.+[lmrsxqbtnkzdGgTCSfcZpPwyhHX><])a(?<su
f>han)$",
        "replace": "\\[${stem}]:#-
${suf}:OBJ.3.F.PL"
    }
]
```

Приложение 4. Описание транслитерации, применяемой при препроцессинге.

Примечание: все регулярные выражения, описанные в приложении 3, применяются в программе после удаления знака ‘ə’ (шва). См. описание препроцессинга в (3.2).

ሀ ha ሁ hu ሂ hi ሃ hA ሄ he ህ hə ሆ ho
ለ la ሉ lu ሊ li ላ lA ሌ le ል lə ሎ lo
ሐ Ha ሑ Hu ሒ Hi ሓ HA ሔ He ሕ Hə ሖ Ho
መ ma ሙ mu ሚ mi ማ mA ሜ me ም mə ሞ mo
ሠ sa ሡ su ሢ si ሣ sA ሤ se ሥ sə ሦ so ሧ swA
ረ ra ሩ ru ሪ ri ራ rA ሬ re ር rə ሮ ro
ሰ sa ሱ su ሲ si ሳ sA ሴ se ስ sə ሶ so
ሸ xa ሹ xu ሺ xi ሻ xA ሼ xe ሽ xə ሾ xo
ቀ qa ቁ qu ቂ qi ቃ qa ቄ qe ቅ qə ቆ qo ቈ kwa ቍ kwi ቋ kwA ቌ kwe ቍ kwə
በ ba ቡ bu ቢ bi ባ bA ቤ be ብ bə ቦ bo
ተ ta ቱ tu ቲ ti ታ tA ቴ te ት tə ቶ to
ቸ ca ቹ cu ቺ ci ቻ cA ቼ ce ች cə ቾ co
ኀ Xa ኁ Xu ኂ Xi ኃ XA ኄ Xe ኅ Xə ኆ Xo ኈ Xwa ኍ Xwu ኋ XwA ኌ Xwe ኍ Xwə
ነ na ኑ nu ኒ ni ና nA ኔ ne ን nə ኖ no
አ >a ኡ >u ኢ >i ኣ >A ኤ >e እ >ə ኦ >o
ከ ka ኩ ku ኪ ki ካ kA ኬ ke ክ kə ኮ ko ኰ kwa ኵ kwu ኳ kwA ኴ kwe ኲ kwə
ወ wa ዉ wu ዊ wi ዋ wA ዌ we ው wə ዎ wo
ዐ <a ዑ <u ዒ <i ዓ <A ዔ <e ዕ <ə ዖ <o
ዘ za ዙ zu ዚ zi ዛ zA ዜ ze ዝ zə ዞ zo
ዠ Za ዡ Zu ዢ Zi ዣ ZA ዤ Ze ዥ Zə ዦ Zo
የ ya ዩ yu ዪ yi ያ yA ዬ ye ይ yə ዮ yo
ደ da ዱ du ዲ di ዳ dA ዴ de ድ də ዶ do
ጀ Ga ጁ Gu ጂ Gi ጃ GA ጄ Ge ጅ Gə ጆ Go
ገ ga ጉ gu ጊ gi ጋ gA ጌ ge ግ gə ጎ go ጐ gwa ጕ gwu ጓ gwA ጔ gwe ጕ gwə
ጠ Ta ጡ Tu ጢ Ti ጣ TA ጤ Te ጥ Tə ጦ To
ጨ Ca ጩ Cu ጪ Ci ጫ CA ጬ Ce ጭ Cə ጮ Co
ጰ Pa ጱ Pu ጲ Pi ጳ PA ጴ Pe ጵ Pə ጶ Po
ጸ Sa ጹ Su ጺ Si ጻ SA ጼ Se ጽ Sə ጾ So
ፈ fa ፉ fu ፊ fi ፋ fA ፌ fe ፍ fə ፎ fo
ፐ pa ፑ pu ፒ pi ፓ pA ፔ pe ፕ pə ፖ po
ቨ va ቩ vu ቪ vi ቫ vA ቬ ve ቭ və ቮ vo ⷶ vwA
ኸ xa ኹ xu ኺ xi ኻ xA ኼ xe ኽ xə ኾ xo
፩ (1) ፪ (2) ፫ (3) ፬ (4) ፭ (5) ፮ (6) ፯ (7) ፰ (8) ፱ (9)
፲ (10) ፳ (20) ፴ (30) ፵ (40) ፶ (50) ፷ (60) ፸ (70) ፹ (80) ፺ (90) ፻ (100)
፼ (10000)