universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„KALAM – 

Wortanalyse für Sabäisch“

verfasst von / submitted by

Dipl.-Ing. Dr.techn. Ronald Ruzicka, BA

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Arts (MA)

Wien, 2016 / Vienna, 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 673
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Arabistik UG2002
Betreut von / Supervisor:	Dr. George Hatke

1. Inhaltsverzeichnis

2. Vorwort

Wenn man als Student in die altsüdarabischen Sprachen eintaucht, steht man bei der Übersetzung so mancher Textstelle vor dem Problem, dass man so gar nicht weiß, was diese bedeuten soll: der oft formularische Beginn des Textes ist verstanden, der nun folgende – entscheidende – Teil lässt nichts aus sich heraus, ganze Passagen sind undeutbar; schon ein einziges Wort würde helfen. Die Problematik wird auch noch insoweit verschärft, als die Vokale fehlen, die zumindest Hinweise auf die grammatikalische Form geben könnten.

Der Autor – durch seinen früheren Ausbildungsweg als Mathematiker und Informatiker einseitig vorbelastet – denkt hier natürlich sofort an eine automatisierte Lösung: in dieser Situation also ist die Idee entstanden, ein Computerprogramm zu entwickeln, bei dem man

1. ein altsüdarabisches Wort eingibt und
2. die grammatikalische Analyse zurück erhält: alle Möglichkeiten für
    a. Wurzel, Stamm, Person, Zahl, Geschlecht, Zeit/Aspekt;
    b. wenn möglich auch den Zeitbereich, in dem diese Form Verwendung fand

Als erstes Beispiel möge das Wort **mkrb** gelten. Als Ergebnis liefert das Programm mit dem Namen ***KALAM**,* eine altsüdarabischen Bezeichnung für *Wort*, die theoretischen Formen:

1. `Stamm krb; O2-Stamm; aktiv/passiv Partizip`
2. `Stamm krb; O1-Stamm; nomen loci`
3. `Stamm krb; O2-Stamm; nomen loci`
4. `Stamm mkrb; … diverse Formen`

Im letzten Fall geht man von einer – rein theoretischen - vier-radikaligen Wurzel aus.

Als zweites Beispiel seien Wörter der Form **ABCD** genannt, bei denen der Stamm nicht einfach zu finden ist; drei mögliche Situationen seien hier genannt:

1. Stamm ABC + Suffix D
2. Präfix A + Stamm BCD
3. Präfix A + Stamm BCC + Suffix D

KALAM liefert hier alle möglichen, tatsächlich denkbaren Varianten. KALAM weiß also auch über Präfixe und Suffixe Bescheid. Diese können sich aufgrund der Beugungen der Wörter oder auch durch Voranstellen eines Partikels (Präposition, Konjunktion, Adverb) ergeben, wie etwa *b-* oder *w-*, oder auch durch Mimation und Nunation zur Unterscheidung der In-/Determination.

Hierbei erkennt man dann schon das erste Problem: theoretisch gibt es mehr Formen als auch praktisch belegt sind. Dies mag für das Erlernen der Sprache durchaus interessant sein, für die praktische Verwendung stellt es eventuell eine Verkomplizierung dar.

Wenn man die gefundenen Wurzeln auf reale Existenz testen könnte, hätte man schon viel erreicht: ein vollständiges Lexikon würde dann in etwa auch helfen, die oben genannte theoretische Form 4. auszuschließen, weil sie noch nicht in einem Text aufgefunden worden ist. Jedenfalls wäre dies ein vom Benutzer optional zu wählender Mechanismus.

Anfänglich dachte der Autor daran, exemplarisch rund einhundert Worte zu Vergleichen zu hinterlegen. Der etwas enttäuschte Blick des Professors für Arabistik war schließlich aber doch Motivation, eine Menge an Wörtern zu implementieren, die umfangmäßig in etwa dem Sabaic Dictionary[1] entspricht.

Und dadurch ist ganz nebenbei eine Datenbank sabäischer Wörter entstanden, die jetzt vollständig in KALAM integriert ist, deren Inhalt aber rein technisch gesehen nicht Teil dieser Masterarbeit ist. Dieses Lexikon liefert zu den theoretisch gefundenen Wörtern nun auch gleich die Übersetzung hinzu – letzteres ist auf den motivierenden Blick des Betreuers dieser Arbeit zurückzuführen.

[1] BEESTONETAL1982

In diesem Sinne möchte ich Dr. George Hatke, meinem Betreuer, für seinen beständigen Input danken, der sehr viel zur Entwicklung der Arbeit beigetragen hat. Auch hat seine Begeisterung für Südarabien so auf mich abgefärbt, dass die prinzipielle Richtung der Arbeit schon von Vornherein vorgezeichnet war.

Prof. Prochazka gebührt der Dank dafür, mich nicht nur durch das gesamte Orientalistik- und Arabistikstudium begleitet, sondern mich auch insbesondere durch die in seinen Vorlesungen vielfach offenbarten Blicke hinter die Kulissen und zu den Querverbindungen im Bereich der semitischen Sprachen für die Grammatik derselben intensiv interessiert zu haben.

2.1 Anmerkungen und Konventionen

Zu Beginn seien auch noch einige Konventionen angemerkt:

a) Die zeitlichen Begriffe *v.Chr.* und *n.Chr.* werden als Konvention verwendet, ohne dass dahinter irgendeine religiöse oder politische Absicht steckt.

b) Bei *kursiv* gesetzten vollständigen und Teil-Sätzen handelt es sich immer um ein Zitat. Dieses wird dann auch in einer Fußnote angemerkt.

c) Englisch-sprachige Zitate werden nicht übersetzt.

d) Als Schreibweise für die nichtemphatischen Sibilanten wird s, š und ś verwendet.

e) Bekannte Namen werden in deutsch-sprachiger Notation geschrieben, außer sie wurden in einem Zitat anders notiert. In letzterem Fall wird die Originalschreibweise beibehalten.

3. Einleitung

Das Programm KALAM zur Analyse sabäischer Wörter versucht, auf Basis der in ihm einprogrammierten grammatikalischen Regeln der Konjugation bzw. der Deklination herauszufinden, was der Stamm sein könnte. Dies ist die Grundaufgabe, die es zu lösen gilt.

Als Basis für die einprogrammierte Grammatik dient das *Lehrbuch der sabäischen Sprache* von Peter Stein. Dieses 2013 erschienene Werk bietet die modernste Sicht auf das Sabäische. Betont sei hier auch, dass die Masterarbeit sich alleinig auf die Sprache des Sabäischen beschränkt. Das Minäische, Qatabanische oder Ḥaḍramitische werden ja schon seit den 1980er Jahren als eigene Sprachen und nicht mehr nur als Dialekte behandelt.

Erst aktuell liegt dieses Standardwerk von Stein vor; in den mehr als 200 Jahren davor, seit der ersten Entzifferung durch Gesenius zu Beginn des 19. Jahrhunderts wurden unterschiedliche Versuche unternommen, die Grammatik des Sabäischen zu verstehen und zu formulieren.

Ein kurzer historischer Abriss der altsüdarbischen Geschichte, der Wiederentdeckungsgeschichte und der Entwicklung der Grammatiken zum Sabäischen seit den Anfängen mit dem ersten Lehrbuch von Fritz Hommel zum *Minäo-Sabäischen* über die Meilensteine von Maria Höfner und Alfred L. Beeston nimmt in vergleichender Form den Beginn des theoretischen Teils der Arbeit ein. Erwähnenswert ist hier etwa, dass Beeston im Laufe seines Lebens auch grundlegende Änderungen in seiner Sicht auf das Sabäische vorgenommen hat.

Dieser Vergleich hilft auch, die in KALAM implementierten Regeln besser zu verstehen.

Die Basisfrage ist nun, frei nach Hans Moser: *Wie nemmen wir im denn?* Wie/wo beginnt man die Analyse eines Wortes?

Hierzu empfiehlt sich das Studium von Ansätzen aus dem linguistischen Bereich, die sich mit Übersetzung ganz allgemein beschäftigen; insbesondere dem Bereich des Natural Language Processing (NLP).

Kann man die Methoden des NLP, das sich ja praktisch immer mit modernen, meist auch noch gesprochenen Sprachen beschäftigt, auf eine alte Sprache anwenden? Gibt es bereits existierende Algorithmen, die man zur Analyse sabäischer Wörter verwenden könnte?

Wie im Folgenden dann gezeigt wird, muss man beide Fragen im Endeffekt mit Nein beantworten: die Besonderheiten des Sabäischen, insbesondere die fehlenden Vokale, aber auch andere, weiter unten zu nennende Gründe haben zur Folge, dass eine neue Methode der Analyse entwickelt werden muss, um KALAM zu realisieren.

Der Schwerpunkt dieser Arbeit liegt jedenfalls auf der Generierung und Verstärkung des praktischen Nutzens, den KALAM leisten könnte.

4. Theorie

Bevor man sich mit der Entwicklung der grammatikalischen Erforschung des Sabäischen beschäftigt, sollte man sich zuerst die Geschichte Südarabiens und dann die Wiederentdeckungsgeschichte (aus westlicher Sicht) vor Auge führen.

4.1 Geschichte Südarabiens

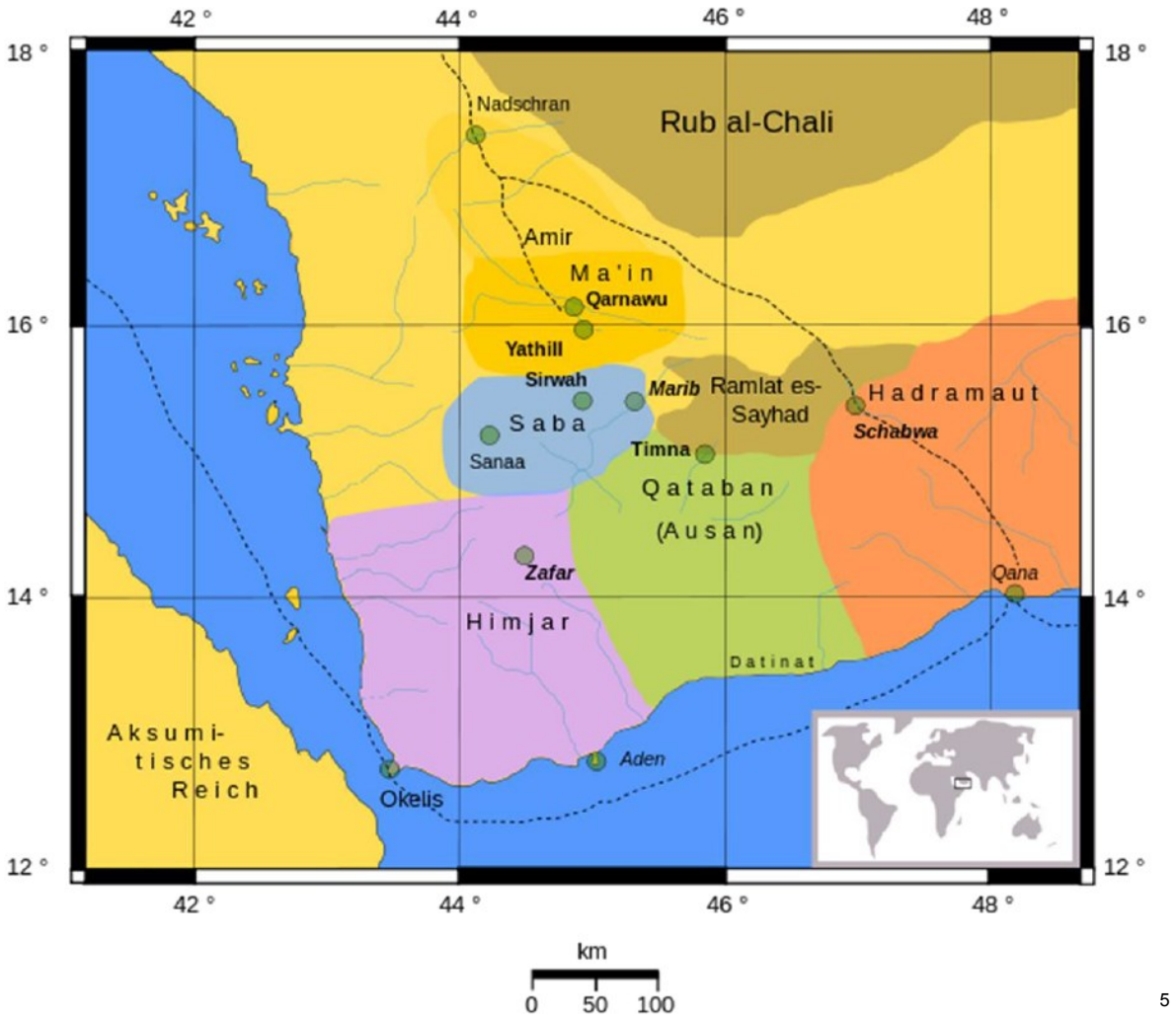
Die Geschichte Südarabiens, soweit sie sich als aufgezeichnete, also „historische" Geschichte definieren lässt, überdeckt den Zeitraum vom 9. Jh. v.Chr. bis zum 7. Jh. n.Chr. Wie George Hatke in seiner Vorlesung zum *Seminar zur Geschichte und Archäologie Südarabiens* anmerkt, ist es nicht nur aus lokalen, sondern weit darüber hinaus reichenden Gründen interessant, sich mit Südarabien zu beschäftigen. Eines der Ziele der Vorlesung sei, *To gain an appreciation of South Arabia's role not as a peripheral region of the Near East but as an axis-point linking the Fertile Crescent, Africa, and South Asia.* Und: *To shed light on the continuities between ancient South Arabian history and that of the early Islamic period, and on the impact which South Arabia had on the world of early Islam.*[2]

Örtlich bewegen wir uns zum Großteil im heutigen Jemen – inklusive der Insel Sokotra -, aber auch im Süden Saudiarabiens und des Omans, im heutigen Äthiopien und in Eritrea.

Die Abb. 1 Karte Alt-Südarabiens. demonstriert diese lokale Zuordnung. Diese Karte zeigt aber auch einen Teil der „Weihrauchstraße", jenes Karawanenweges, der ab dem Beginn des 1. Jahrtausends v.Chr., nach der Domestizierung des Dromedars im ausgehenden 2. Jahrtausend[3] (wobei aktuellere Forschungen für Süd-Ost-Arabien vom Beginn des 1. Jahrtausends[4] sprechen) die Kulturen der Levante, des *fruchtbaren Halbmonds*, Südarabien näherbrachte. Auf diesem Weg hat schließlich auch die Schrift ihren Weg hierher gefunden, wenn sie auch auf noch nicht näher zu erklärende Weise an die lokalen Sprachen angepasst wurde.

---

[2] HATKE2014.
[3] RETSÖ1992: S. 30, 44ff.
[4] UERPMANN2002: S.250.



[5]

Abb. 1 Karte Alt-Südarabiens.

Die Frage, wie stark die Kultur Südarabiens von jener Nordarabiens und der Levante oder auch von Assyrien abhängt, wird in Fachkreisen intensiv diskutiert. Diese Frage ist insbesondere für unser Anliegen von Bedeutung, als die Kultur Südarabiens in erster Linie in ihren Monumenten und Schriften überliefert ist.

[5] Quelle: HATKE2014.

Manche wollen überhaupt erst ab dem Zeitbereich von einem südarabischen Kulturkreis sprechen, als es schriftliche Überlieferungen gibt[6].

Im Prinzip schwanken die Meinungen zum Ursprung der südarabischen Hochkultur zwischen einer Richtung, die diese weitgehend auf eine Herkunft aus der Levante zurückführen, basierend auf einer Einwanderungswelle Ende des 2. Jahrtausends – vertreten etwa durch Norbert Nebes[7], bis zu jenen die nur von peripheren Kontakten ausgehen und schon eine vorhandene Bronzezeitkultur als Ursprung vorsaussetzen – wie etwa Allesandra Avanzini[8]. Avanzini führt hierzu sprachgeschichtliche Argumente ins Treffen, die Qatabanisch und allgemein die altsüdarabische Sprachfamilie, welche sie in ihrer Existenz als erwiesen ansieht, auf eine proto-südarabische Sprache zurück. Diese hat wiederum starke Affinitäten zur Nord-Westsemitischen Sprachfamilie und nicht zum Zentralsemitischen[9].

Die oben erwähnte Karte zeigt auch, dass auf einem relativ kleinen Gebiet, in unterschiedlicher Konfiguration und zeitlicher Abfolge, Königreiche vorhanden waren, deren Grenzen aber wohl nicht so exakt zu definieren sind, wie es die Karte suggeriert.

Im Zuge des Beginns der Karawanenepoche entstanden zunächst zwischen 900 und 700 v.Chr. Stadtstaaten in der Ǧawf Region, wie etwa Naššān und Našqum, und ebenso im Ḥaḍramawt, wie etwa Raybūn.[10]

Im frühen 7. Jahrhundert eroberte der König von Sabaʾ Karib'il Watar I. bin Dhumarᶜali große Teile Südarabiens, auch die Stadtstaaten im Ǧawf. Die parallel existierenden Königreiche Qataban und Ḥaḍramawt wurden dann Satelliten Saba's.

[6] HATKE2014.
[7] AVANZINI2009: S.205.
[8] Ibid: S.216.
[9] Ibid: S.208ff.
[10] HATKE2014.

Kaum 100 Jahre später existierten nebeneinander die unabhängigen Reiche Saba', Ma'īn, Qataban und Ḥaḍramawt.

Wenn wir uns in der Folge mit dem *Sabäischen* beschäftigen, so reden wir von der Sprache, die im Königreich Sabaʾ mit seinem Zentrum Mārib vorherrschte. Durch enge Kontakte mit den sprach-verwandten Nachbarstaaten und später auch dem arabischen Norden war das Sabäische starken externen Einflüssen unterworfen.

Unter zumindest starkem kulturellem Einfluss entstanden schon im 8. Jahrhundert im heutigen Äthiopien Schwesternkulturen, die sogar dieselbe Schrift nützten. Im Zuge des Handels im Indischen Ozean, beginnend mit dem 3. Jahrhundert v.Chr., wurden südarabische Kolonien in Sokotra und Ostafrika angelegt.

Rund um die Zeitenwende formte sich im südlichen Hochland das Königreich Ḥimyar. Zugleich ging Maᶜīn unter.

Im 3. Jahrhundert war das Königreich Aksum (heutiges Äthiopien) so erstarkt, dass es in die immer wieder aufflammenden Kämpfe der rivalisierenden Königreiche im Jemen eingriff. Ende des 3. Jh.s übernahm Ḥimyar die Macht und eroberte auch weite Teile der Küste des Roten Meeres und deren Hinterland. Die Elite nahm das Christentum Mitte des 4. Jh.s an.

Im 5. Jh. wurden weitere Gebiete annektiert und der politische Einfluss auch in Richtung Zentralarabien erweitert. Die regierende Elite nahm das Judentum an. Als dann im 6. Jahrhundert Massaker an Christen verübt worden waren, nahmen dies die Äthiopier zum Anlass, im Land einzufallen. Äthiopische Abkömmlinge beherrschten, wenn auch vom Mutterland in Aksum unabhängig, Südarabien, bis schließlich um 570 die Sāsāniden Ḥimyar eroberten.

Der aufsteigende Islam profitierte anfangs von Truppenteilen, die Südaraber stellten. Immer wieder gab es jedoch auch Kämpfe zwischen den jemenitischen Stämmen und dem immer mächtiger werdenden islamischen Staat. Bis zum 10. Jahrhundert spielte die Region eine untergeordnete politische Rolle, als sie dann vom Kalifat

wegbrach und lokale Führer und Kleindynastien die Herrschaft übernahmen – der Kreis hatte sich geschlossen.

4.2 Geschichte der Wiederentdeckung Alt-Südarabiens

Die Geschichte der Wiederentdeckung wird hier aus westlich-europäischer Sicht betrachtet. Dies erscheint zunächst sehr subjektiv. Doch auch in einer Gesamtsicht wird Alt-Südarabien kaum früher von anderer Seite, insbesondere von islamisch-arabischer, objektiv reflektiert, handelt es sich aus islamischer Sicht doch um die Zeit der Ǧāhilīya *der Unwissenheit*. Wenn auf Südarabien in frühen Schriften Bezug genommen wird, dann oft in rein mythologischem oder zumindest mythisch angehauchtem Zusammenhang, etwa im Bezug auf die Königin von Saba' und König Salomon:

- Christlich-jüdische Tradition: im Alten Testament, Buch der Könige
- Arabisch-islamisch: die Königin von Saba' heißt Bilqis; Koran, 27. Sure
- Äthiopisch: Makeda trifft Salomon und gebärt dann Menelik, der dem Vater Salomon die Bundeslade entwendet und sie nach Äthiopien bringt.
    - in einer aksumitischen Variante hat die Königin einen Pferdefuß[11]

In älteren arabischen Gedichten, Volksliedern und Geschichten wird auf *Ḥimyar* Bezug genommen und damit allgemein das Gebiet des alten Südarabiens bezeichnet; etwa in der Geschichte *Ein Mann von Ḥimyar*[12].

Auch findet man in den Schriften einiger bekannter arabischer Autoren Informationen über das alte Südarabien, deren Wahrheitsgehalt aber meist als legendenhaft qualifiziert werden muss. Als ein Beispiel sei aṭ-Ṭabarī genannt, der persisch-arabische Autor des 9. und angehenden 10. Jahrhunderts, der unter Anderem über einen der letzten altsüdarabischen Könige schreibt: Abraha. Dieser sei von seiner Herkunft her der Anführer der zweiten aksumitischen Streitmacht, die gegen Ḥimyar

[11] CHRISTOF2006: S. 224.
[12] TEMMAM2004: S. 103ff.

zieht, dieses erobert. Abraha macht sich von Aksum unabhängig und wird selbst König in Ḥimyar.

Diese kurzen Beispiele sollen nur zeigen, dass es von Beginn an (nach dem Ende der südarabischen Königreiche) Interesse an Altsüdarabien gab, das belegbare Wissen darüber allerdings sehr spärlich war.

Die eigentliche Wiederentdeckung aus wissenschaftlicher Sicht begann wohl mit einer dänischen Expedition im Jahr 1761 in den Jemen. An dieser nahm auch der deutsche Forscher Carsten Niebuhr (1733-1815) teil, dem wir einen der ersten detaillierten westlichen Berichte über den Jemen verdanken.[13]

Der Beginn der Spracherforschungsgeschichte lässt sich an der Person Wilhelm Gesenius (1786-1842) festmachen, einem deutschen Hebräisten, der mithalf, die südarabische Musnad Schrift zu entziffern.

Bei dieser Schrift handelt es sich um eine Buchstaben-Monumentalschrift, deren Vorgänger wohl im Phönizischen Alphabet zu suchen sind. Diese reine Konsonantenschrift durchlebte im Verlauf der Jahrhunderte stilistische Änderungen, doch blieben die Grundformen im Wesentlichen erhalten.

Die erste große Sammlung an Inschriften (mehr als 800) brachte der französische Semitist Joseph Halévy (1827-1917), der den Jemen in den Jahren 1869-1870 bereiste, nach Europa mit. Auch er gehört zu den erwähnenswerten Wissenschaftlern, die zur Entzifferung beitrugen.

Der bedeutendste österreichische Südarabienforscher ist Eduard Glaser (1855-1908), der in seinen vier Reisen in den Jemen zwischen 1882 und 1892 trotz Widerständen im Jemen und in seiner Heimat einen enormen Schatz an Inschriften sicherstellte. Viele davon sind heute in Form von Abklatschen in der

[13] HATKE2014.

Österreichischen Akademie der Wissenschaften gelagert und werden soeben elektronisch aufgearbeitet.

Aus Österreich sind dann auch David Heinrich Müller (1846-1912), Professor für semitische Philologie, mit seiner Expedition in den Oman und Jemen zwischen 1889 und 1890 und Nikolaus Rhodokanakis (1876-1945), ebenfalls Professor für semitische Philologie, insbesondere aufgrund ihrer Übersetzungen altsüdarabischer Texte, zu nennen.

Um die Erforschung (teilweise vorort) und insbesondere die Übersetzung machten sich auch A.F.L. Beeston (1911-1994), ein britischer Semitist und Arabist, sowie Walter W. Müller (1933*) aus Deutschlang verdient.

Zu den bekanntesten aktiven Südarabisten – mit unterschiedlichen Schwerpunkten in Philologie, Epigraphik und Archäologie - zählen prominent Alessandra Avanzini, Norbert Nebes, Christian Robin, Peter Stein, um nur einige zu nennen.

4.3 Grammatiken zum Alt-Südarabischen

Wie oben erwähnt gelang zu Beginn des 19. Jahrhunderts Wilhelm Gesenius gemeinsam mit Emil Rödiger die Entzifferung der altsüdarabischen Schrift. Und in dieser Zeit begannen auch die Versuche, Texte zu übersetzen. Zu dieser Zeit wurde vom *Ḥimyarischen* gesprochen, sich auf die arabische Tradition beziehend (siehe oben).

4.3.1 Der Beginn

Die erste umfassende Grammatik stammt von Fritz Hommel aus dem Jahr 1893. In diesem handgeschriebenen Buch ist zuerst einmal von der *minäo-sabäischen* Sprache die Rede

Hommel geht mit den Zeitbegriffen ebenso großzügig um, wie mit den Ortsbezügen. So spricht er von *sicher schon Mitte des 2. vorchristlichen Jahrtausends* als Beginn

ebenso, wie von einer Spannweite bis *al-ᶜUlā in Nordarabien*[14]. Aus heutiger Sicht mag es wohl stimmen, dass sich im Zuge des Handels auch bis dorthin Altsüdarabisch-Sprecher einfanden, es ist aber kaum vom Teil des Kernlandes zu sprechen.

Hommel fasst unter Minäo-Sabäisch die seiner Meinung nach gemeinsame Sprache in den Königreichen Maᶜin und Sabaʾ, und später Ḥimyar zusammen und weist die Sprache zurecht der westsemitischen Sprachgruppe zu. Heute würde man das Sabäische – je nach persönlicher Meinung, siehe 4.1, in die Untergruppe Zentralsemitisch oder Nordwestsemitisch mitaufnehmen, die aber beide innerhalb der Westsemitischen Gruppe verortet werden.

Auffällig an Hommels Grammatik, die sich im Prinzip nur als Anhängsel an seine Chrestomathie versteht, dass er in vielen Bereichen aufgrund der damals doch noch spärlichen Beleglage viel spekuliert: sehr oft sind hier Analogieschlüsse zum Arabischen das Hilfsmittel der Wahl, wie etwa bei den Pronomina[15].

Die Pronomina zeigen auch beispielhaft, dass er sehr wohl schon zwischen der s-Sprache Minäisch und der h-Sprache Sabäisch zu unterscheiden vermag: etwa das Suffixpronomen der dritten Person maskulin ist einerseits `-s` andererseits `-hw`.

Beim Verbum macht sich besonders mißlich fühlbar, daß wir einmal keine Vocalisierung haben und deshalb nur in manchen Fällen die Vocale aus den verwandten Sprache, in erster Linie dem äthiopischen und nordarabischen, erschließen könne, und daß zweitens nur die 3. Person … in den Inschriften vorkommt[16]. Wie wahr – zumindest teilweise!

Bei der Stammbildung erscheinen noch einige Formen, die heutzutage als obsolet gelten. So wird ein Präfixstamm noch ohne Vokalverkürzung, also die Form

[14] HOMMEL1893: S. 1f.
[15] Ibid: S. 11.
[16] Ibid: S. 18.

yifaᶜul noch für möglich gehalten, ebenso ein O3-Stamm wie fāᶜala[17]. Beide Formen dürften im Sabäischen tatsächlich nicht vorhanden gewesen sein und sind in finiten Formen nicht eindeutig belegt.[18]

Auch die Langformen mit n werden eher als Kuriosa gesehen[19] und noch nicht in ihrer historischen Entwicklung als frühe Formen; siehe auch 6.1.1.

Zusammengefasst: Hommels Grammatik ist eine Pionierleistung, doch aus heutiger Sicht in weiten Bereichen aufgrund der zu vielen, teils unbegründeten Analogieschlüsse zum Arabischen und *Äthiopischen*, womit Hommel Geᶜez meint, obsolet.

4.3.2 Dialekte oder Sprachen?

Begann Hommel schon zu erkennen, dass es sich wohl nicht um eine einheitliche altsüdarabische Sprache gehandelt hatte, so merkte Ignazio Guidi 1926[20] an, dass es sich um vier unterschiedliche Dialekte dreht: Sabäisch. Minäisch, Qatabanisch und Ḥaḍramitisch.

Auch Carlo Conti Rossini hält 1931 in seiner Chrestomathie durch Kapiteleinteilungen an dieser dialektalen Einteilung fest, fügt aber noch ein eigenes Kapitel für ʾAusān an[21]. Diesen Bereich würde man heutzutage ebenfalls unter Qatabanisch einreihen.

Die historisch gesehen nächste komplette Grammatik legte Maria Höfner 1943 vor. Auch Sie hielt an der dialektalen Betrachtungsweise fest.

Höfner verweist schon bald in Ihrem Buch, bei den Pronomina, auf alte Gemeinsamkeiten der semitischen Sprachen[22], denkt also unausgesprochener

---

[17] HOMMEL1893: S. 18f.
[18] STEIN2013: S. 83ff.
[19] HOMEL893: S.23.
[20] GUIDI1926: S. 1ff.
[21] CONTIROSSINI1931: S. 94f.
[22] HÖFNER1943: S. 28.

Weise an so etwas wie eine protosemitische Sprache – eine moderne Sicht, die auch heutzutage gerne für Argumentationen verwendet wird, siehe 4.1.

Bezüglich der Vokalisierungen geht sie behutsamer um als etwa Hommel, indem sie auf fehlende Belege verweist; z.B. in [23]. Auch verweist Sie darauf, dass (damals) nur die 3. Person der Konjugation belegt war und lässt sich auf weitere Spekulationen nicht ein[24].

Ausführlich würdigt sie die Bedeutung der Infinitive, insbesondere der Infinitivketten[25]. Die Verbformen mit n werden systematisch behandelt und deren fehlen im Ḥaḍramitischen und Qatabanischen vermerkt. An Stämmen unterscheidet Höfner (dazu in Klammern der Korrespondenzen laut Stein)[26]

I.	Grundstamm	(O1)
II.	Steigerungsstamm	(O2)
III.	Einwirkungsstamm	
IV.	Kausativstamm	(H)
V.	Reflexiv zu II	(T2)
VI.	Reflexiv zu III	
VII.	N-Stamm	
VIII.	Reflexiv zu I	(T1)
IX.	–	
X.	ST-Stamm	(ST)

An dieser Einteilung erkennt man, dass sich auch Höfner eng an das Schema des Arabischen hält. Die III. und VI. Formen sind rein spekulativ, da sie sich vom II. und V. schriftmäßig nicht unterscheiden und in Texten bestenfalls aus dem Zusammenhang erraten lassen. Spezielle Formen, wie die geschriebene Dopplung

[23] HÖFNER1943: S. 60.
[24] Ibid: S. 67.
[25] Ibid: S. 64.
[26] Ibid: S. 86ff.

des zweiten Radikals, kommen nur im Minäischen vor, mit dem wir uns hier nicht näher beschäftigen wollen.

Bei den Nomina werden die drei Status – determinatus, absolutus und constructus[27] – schon korrekt voneinander unterschieden und mit Beispielen belegt. Kapiteln zu Partikeln und Syntax runden das Büchlein ab.

Höfner liefert die erste vollständige Grammatik, stützt sich dabei aber nach meinem Geschmack noch zu sehr auf das Arabische.

4.3.3 Beeston: Zeit- und Qualitätssprünge

1962 erscheint die erste Grammatik von Alfred F.L. Beeston. Auch er verwendet die Einteilung in vier Dialekte, beschreibt aber auch Mischformen[28].

Die Phonologie wird von Beeston sehr ausführlich in moderner Form beschrieben, inklusive Beispielen zu Assimilation, Dissimilation und Metathesis[29].

Wie Höfner werden auch in diesem Buch die 3. Personen primär zur Betrachtung der Konjugationsregeln herangezogen. Die Verbalstämme sind nur in den Formen:

- Grundstamm (O1, O2)
- Kausativ (H)
- T-Präfix (T1, T2)
- T-Infix (T1, T2)
- ST-Präfix (ST)

angeführt. Der n-Stamm wird verworfen, da es sich hierbei nur um singuläre Auffindungen im Infinitiv handelt[30]. Ebenso korrigiert Beeston Höfner anhand von (neuen oder neu-interpretierten) Belegstellen, etwa bei den n-Endungen der Verben.

---

[27] HÖFNER1943: S. 123ff.
[28] BEESTON1962: S. 9f.
[29] Ibid: S. 16ff.

Beestons erste Grammatik ist viel enger gesteckt und näher bei den Belegen, weniger interpretativ, aber weniger flüssig und übersichtlich gestaltet als Höfners.

22 Jahre zogen danach ins Land, neue Texte wurden entdeckt, im Speziellen jene aus Mahram Bilqis, und Beeston zitiert sich in seiner neuen Grammatik selber: … *for the continuing progress of these studies will certainly entail modifications of the present positions*.[31]

Und der auffälligste Schritt folgt gleich danach: *For this reason, it seems appropriate to treat Sabaic as a language in its own right*.[32] Die altsüdarabischen Sprachen werden also spätestens seit damals – praktisch von jedem heutzutage anerkannt – als eigene Sprachen einer Sprachfamilie anerkannt; nicht mehr nur als Dialekte, sondern selber eigene Dialekte herausbildend.

Beeston begründet dies auch damit, dass zu diesem Zeitpunkt der Corpus an sabäischen Texten schon eine kritische Masse erreicht hätte. Sein Buch nennt konsequenter Weise *Sabaic Grammar*.

Die Verbalstämme werden in diesem neuen Buch sehr kurz abgehandelt, da es eben nur wenige orthografisch unterscheidbare Stämme gebe. Beeston zweifelt sogar die Existenz eines Dopplungsstammes an, was heutzutage nicht mehr tragbar ist. Dafür spekuliert er über einen `fāᶜala` Stamm.[33]

Aufgrund der vielfältigen semantischen Verwendungen werden die strengen Bedeutungen von *perfect* und *imperfect* aufgebrochen: ersteres werde für punktuelle und durative Ereignisse verwendet, während zweiteres ein zukünftiges Ereignis (in verschiedenen Gradualitäten) bedeutet, aber auch indikativ verwendet wird[34].

---

[30] BEESTON1962: S. 19.
[31] BEESTON1984: foreword.
[32] Ibid.
[33] Ibid: S. 13.
[34] Ibid: S. 19.

Bei den n-Endungsformen der Verben wird Beeston aufgrund vieler aufgefundener Ausnahmen vorsichtig: er möchte keine Bedeutungsdeutungen vorgeben, bezeichnet sie sogar als optional.

Auch bei den Nomina zieht sich Beeston auf eine „vorsichtige Position“ zurück: aufgrund der fehlenden Vokale sei die Verwendung von Fällen nicht nachweisbar, und wenn, dann nur widersprüchlich bzw. uneinheitlich[35].

Relativ großen Raum widmet Beeston Pronomina, Präpositionen und Partikeln, sowie Satzkonstruktionen, wie konditional, relativ, subordinativ. Zusätzlich zu den umfassenden Ausführungen über Sabäisch enthält der Anhang auch die Beschreibung von Besonderheiten von Minäisch, Qatabanisch und Ḥaḍramitisch[36].

Beeston hat aufgrund der vielen Textneufunde gegenüber seinem ersten Buch nicht viele grundlegende Änderungen der Grammatik vorgenommen. Er bezieht sich noch weniger auf Arabisch und kaum auf andere Sprachen (als Beispiel nennt er etwa einen jemenitischen Dialekt). Er geht streng nach Belegbarkeit und Nachweisbarkeit vor, vermeidet dadurch vielfach mögliche aufkeimende Diskussionen, geht meiner Meinung nach aber darin zu weit und vergibt Chancen auf tiefergehende Interpretationen.

Der große Verdienst ist aber die Anerkennung der Eigensprachlichkeit des Sabäischen und die – schon zwei Jahre früher stattgefundene – Mitarbeit am ersten umfassenden *Sabaic Dictionary*.[37]

4.3.4 Aktuelle Entwicklungen

Beeston hatte aufgrund seiner fachlich gewichtigen Persönlichkeit und seiner vielen einschlägigen Arbeiten scheinbar für lange Zeit anderen den Mut genommen, selber

[35] BEESTON1984: S. 32.
[36] Ibid: S: 58ff.
[37] BEESTONETAL1982.

eine neue Grammatik des Sabäischen zu erstellen. Aber es erschienen in den mehr als 30 Jahren seither zahlreiche Artikel über viele Teilbereiche der Grammatik, die oft auch auf neu entdeckten oder neu interpretierten Texten fußten (wie etwa über enklitische Partikeln[38]).

Eine sehr kurz gehaltene Grammtik findet sich bei L. E. Kogan und A. V. Korotayev 1997. Das Sabäische wird hier als Basis genommen, auch wenn die vier Sprachen als solche separat tituliert werden. Abgesehen von seiner Kürze von 12 Seiten wird an diesem Werk kritisiert, dass kaum direkte Belegarbeit durchgeführt, sondern vorhandene Sekundärliteratur (Lexika und vorhandene Grammatiken) verwendet wurde[39].

Ein genereller Artikel von Norbert Nebes befasst sich mit dem Stand der Dinge zum Zeitpunkt 1997. Höfners Grammatik wird auch hierin noch geschätzt. Gefordert wird aber auch eine systematische Vorgangsweise bei der eventuellen Erstellung einer neuen Grammatik: für jede *grammatikalische Erscheinung* seien

- Belege vollständig zusammenzutragen,
- anhand von Fotografien zu überprüfen und
- die Regeln anhand von Beispielen abzuleiten[40]

Zu vielen Einzelthemen fehlten aber noch Monographien.

Nebes geht näher auf einige Problemfelder ein, wie Matrex Lectiones oder Relativsätze. Er unterscheidet konsequent die Bezeichnungen *Präfix*- und *Suffixkonjugation* (statt *imperfect* und *perfect*) und *Lang*- und *Kurzformen* (mit und ohne schließendem n). Auch die Bedeutung des Infinitivs wird für das Sabäische bei (damals) 1500 bekannten Formen hervorgestrichen[41]. Zuletzt wird auch auf die Bedeutung der zusätzlichen vergleichenden Betrachtung der anderen

[38] NEBES1991.
[39] MULTHOFF2012: S. 7f.
[40] NEBES1997: S. 112.
[41] Ibid: S. 121.

altsüdarabischen Sprachen verwiesen, auch wenn Nebes sie hier wieder als Dialekte bezeichnet.

Gerade mit diesen anderen Sprachen beschäftigt sich im Sinne einer Dialektgeographie ein Artikel von Stein aus dem Jahr 2004. Stein unterscheidet unter Angabe der grammatikalischen Besonderheiten

- nördliche Dialekte (Haramisch/Amiritisch im Ǧauf) [42]
- südliche Dialekte (z.B. Radmanisch)
- zentrale Dialekte (Hochsabäisch)

In vielen Fällen handelt es sich bei den Unterschieden zwischen den Dialekten um Assimilationen bzw. Nichtassimilationen und Lautverschiebungen im Vergleich zum „Hochsabäischen". An den Randgebieten kommt es auch zu Einflüssen nicht-sabäischer Sprachen, wie Qatabanisch oder Arabisch. Des Weiteren werden Sprachmerkmale der Eroberer von den Eroberten oft übernommen.

Anne Multhoff liefert in ihrer Dissertation, die mehr als 900 Seiten umfasst, keine vollständige Grammatik, aber eine sehr detaillierte Untersuchung über Verbalstammbildung. So werden etwa für alle Wurzeln, die nur irgend zu Erkenntnissen über die historische Entwicklung beitragen könnten, sämtliche Belege angeführt. Zu den im Buch von Stein unten genannten sechs Stämmen werden auch noch *unsichere* Stämme, also solche, die in Belegen nicht sicher den Stammklassen zugeordnet werden können, da die schreibweise ident zu anderen Stämmen ist, angegeben. Daraus ergibt sich die Liste (die unsicheren *kursiv* gesetzt)[43]

- 01 f`l Grundstamm
- 02 f`ln faktitiv/kausativ
- *03 f`ln*
- H hf`ln kausativ

[42] STEIN2004: S. 228ff.
[43] MULTHOFF2012: S. 9.

- *H2* *hf`ln*
- T^{in} ft`ln reflexiv/passiv/reziprok
- *T^{in}* *ft`ln*
- T^{pr} tf`ln reflexiv/intransitiv
- *T^{pr}* *tf`ln*
- ST stf`ln desiderativ/passiv/intransitiv
- *ST2* *stf`ln*

T^{in} bezeichnet einen Infix-Stamm, T^{pr} einen Präfix-Stamm.

Den aktuellen Stand des Wissens spiegelt die Grammatik von Stein aus dem Jahr 2013 wider[44]. Hierin sind die bei Nebes (siehe oben) genannten Voraussetzungen an eine moderne Grammatik des Sabäischen erfüllt, insbesondere die Beleglage und die Berücksichtigung vorhandener Teilarbeiten. Schon der banale Vergleich der Seitenzahlen – 76 bei Beeston und 232 bei Stein (auch gibt es ein eigenes Buch, Band 2, als Chrestomathie) - zeigt die Präzision der Arbeit Steins.

Wichtig sind hierin auch die Vergleiche mit den anderen altsüdarabischen Sprachen.

Vom Stil her „traut“ sich Stein mehr zu, als Beeston in seinem zweiten Buch. Er versucht Vokalisierungen und hypothetische Formen anzugeben und belegt diese aber immer mit plausiblen Analogien zu anderen Sprachen, bzw. unterlässt Spekulationen, wo sie sich nicht ausreichend begründen lassen.

Dieses Buch ist so detailliert in seinen Paradigmen, dass die Regelliste in Kapitel 7 umfassend danach erstellt werden konnte.

[44] STEIN2013.

Für die Verbalstämme ergibt sich bei ihm die vereinfachte Form[45]

- 01 Grundstamm
- 02 faktitiv/kausativ
- H kausativ
- T1 reflexiv/passiv/reziprok
- T2 reflexiv/passiv/reziprok
- ST desiderativ/passiv/intransitiv

Womit wir nun genug Grundlagenwissen aus der sabäischen Philologie gesammelt haben, um uns mit der Praxis von KALAM beschäftigen zu können.

[45] STEIN2013: S. 224.

5. … und Praxis

Dieses Hauptkapitel beschäftigt sich mit der Entwicklung von KALAM.

Die Linguistik - *Sprachwissenschaft* - ist jene Wissenschaft, die die menschliche Sprache untersucht. Das Linguistik-Teilgebiet des Natural Language Processing (NLP) beschäftigt sich unter anderem mit jenen Themen, die für die Entwicklung eines Übersetzers notwendig sind.

Daher ist es geboten, sich zuerst einmal mit NLP und seinen Methoden näher auseinanderzusetzen. Eventuell existieren bereits Verfahren zur Wortanalyse, die bei der Übersetzung von sabäischen Wörtern verwendet werden könnten. Oder man könnte eventuell Ansätze und Ideen solcher Verfahren bei der Entwicklung eigener Methoden verwenden.

Da diese Arbeit dem Bereich (arabischer) Philologie und nicht der Linguistik zuzuordnen ist und des Weiteren, wie in der Einleitung festgelegt, der praktische Nutzen im Vordergrund stehen soll, werden wir uns hier nicht mit linguistischer Theorie, sondern mit praktischen Anwendungen derselben auseinandersetzen.

Praktische Anwendungen der Linguistik schließen spätestens seit der allgemeinen Zugänglichkeit von Personal Computern Mitte der 1970er Jahre immer die Verwendung von Computern und dedizierter Software mit ein, sodass eine Betrachtung der Entwicklung in zumindest den vergangenen zwei Jahrzehnten in diesem Gebiet nützlich erscheint.

Die aktuellsten Berichte liefern hierzu einschlägige Tagungsbände und Proceedings von Kongressen. Und bei deren Lektüre - konkret zu den Themen Wort- und Textanalyse - fällt ein roter Faden auf: ja es gibt dahinterliegende Theorie, aber diese wird meist sprachabhängig different in Algorithmen und Computerprogrammen implementiert.

Auch die Methoden des Part-of-Speech Taggings, also der Zuordnung von Wörtern (und Satzzeichen) zu Wortarten erscheinen auf den ersten Blick sehr nützlich.

Exemplarisch, aber durchaus repräsentativ seien hier die im Folgenden dokumentierten Beispiele genannt.

5.1 Computergestützte und maschinelle Übersetzung

Zuerst aber wollen wir einen Schritt zurücktreten: einerseits um den Blick auf das Ganze nicht zu verlieren, andererseits auch, um die Anfänge der maschinellen Übersetzung im Auge zu behalten.

Wortanalyse ist als Thema, insbesondere im Kontext der Philologie, sehr interessant, doch ist sie im übergeordneten Zusammenhang auch als Teil, als Hilfsmittel eines weitergefassten Aufgabenbereichs zu sehen; nämlich der Übersetzung von Texten von einer Sprache in eine andere, insbesondere mithilfe von Computern. Als Zeuge sei hier ein schon ein Viertel Jahrhundert alter Klassiker zitiert, der sich mit der Basis, aber eben auch mit praktischen Anwendungen beschäftigt: Luckhardt/ Zimmermanns Buch *Computergestützte und maschinelle Übersetzung*.

Die ersten Schritte, wenn auch auf aus heutiger Sicht recht einfacher Basis, wurden aber bereits in den 30er Jahren des 20. Jahrhunderts getätigt: maschinelle Vergleiche unterschiedlicher Lexika. Die theoretischen Anfänge in der Praxis einsetzbarer Übersetzungsprogramme liegen in den 60er und frühen 70er Jahren desselben Jahrhunderts.[46]

Und in dieser Zeit hat auch die Vorarbeit zu einem der umfangreichsten Projekte zur Erstellung von Software für maschinelle Übersetzung begonnen: dem System SUSY, das den maschinellen Übersetzungsteil des STS, des *Saarbrückener Translationssystems*, bildet. STS wurde im Rahmen eines langfristigen Forschungsprojektes an der Universität Saarland entwickelt.[47]

Hinter SUSY steckt insbesondere ein ausgetüfteltes linguistisches Modell. Das Gesamtsystem sollte sprachunabhängig Übersetzungen vornehmen können. Die drei

[46] LUCKHARDT1991: S. 2.
[47] Ibid: S. 19ff.

separierten Teile Analyse, Transfer und Synthese sollten diese Unabhängigkeit absichern.

Die Bereiche dieser drei Teile sind:[48]

Analyse:

a) Textnormierung
b) Morphologische Analyse
c) Homographenanalyse (Versuch der Schaffung von Eindeutigkeiten auf syntaktischer Ebene)
d) Satzsegmentierung
e) Nominalgruppenanalyse
f) Verbgruppenanalyse
g) Komplementanalyse
h) syntakto-semantische Disambiguierung (Schaffung von Eindeutigkeiten auf Wortebene)

Transfer:

i) Wortübersetzung (lexikalisch)

Synthese:

j) syntakto-semantische Synthese (nötige Strukturtransformationen, wenn die Struktur von Quell- und Zielsprache voneinander abweichen)
k) syntaktische Synthese
l) morphologische Synthese

Näher wollen wir auf die einzelnen Module nicht eingehen. Doch werden wir in späteren Beispielen, siehe 5.3, die Teilaufgaben a)-h) ebenfalls wiederfinden.

[48] LUCKHARDT1991: S. 20.

Mit unserem Teil der Wortanalyse bewegen wir uns in den Bereichen b) und teilweise e) und f).

Im Weiteren wird der Begriff **CAT** *Computer Aided Translation* definiert und zwischen CAT-C und CAT-H unterschieden.[49]

CAT-C (C für *Computer*) beschreibt die automatische Computerübersetzung, der aber manuelle Prä- und Post-Editierung vorangeht – diese wird für Textbereiche mit umfangreich vorhandenen Lexika bzw. Corpora verwendet.

CAT-H (H für *Human*) steht für den Fall von menschlicher Übersetzung, die sich der Unterstützung durch Computersoftware bedient; Lexika sind in diesen Fällen von Texten und Sprachen nur spärlich vorhanden. Das unten vorgestellte Wortanalyseprogramm KALAM könnte sich hier in die Hilfsmittel für CAT-H einreihen.

CAT-C wird bisweilen auch als **HAMT** *Human Aided Machine Translation* und CAT-H als **MAHT** *Machine Aided Human Translation* bezeichnet.

Um den Befund zur automatischen Übersetzung nun historisch abzurunden, muss angemerkt werden, dass es für einige Teilbereiche – etwa zu speziellen Themenkreisen – sehr brauchbare Anwendungen gegeben hat, die auch kommerziell erfolgreich waren. Andere sind wiederum gescheitert, insbesondere auch der Anspruch von STS als ganz allgemeine Lösung: die menschlichen Sprachen sind einfach zu komplex.[50]

Im Großen und Ganzen ist aber, wenn nicht Ernüchterung, so doch zumindest eine sehr realistische Einschätzung von Übersetzungssoftware eingetreten. Heutzutage gibt es sehr günstige maschinelle Übersetzer bis hin zu Gratis-Webseiten mit qualitativ fraglichen Ergebnissen. Etwa auch der wohl jedem Internet-Nutzer bekannte Google-Übersetzer oder jener von Facebook liefern sehr oft Stilblüten, die an der Machbarkeit von maschineller Übersetzung prinzipiell zweifeln lassen.

---

[49] LUCKHARDT1991: S. 49ff.
[50] Ibid: S. 4.

5.2 Part-Of-Speech Tagging

Part-Of-Speech Tagging ist ein Überbegriff für Technologien, die versuchen, einzelnen Worten eines natürlich-sprachlichen Textes grammatikalische Bedeutungen zuzuordnen. Das Wort *versuchen* ist hier bewusst gewählt, weil hierin meist auch mit Wahrscheinlichkeiten gearbeitet wird – aufgrund von Lernvorgängen weiß die Methode durch Vergleich mit Lehrvorgaben mit einer bestimmten Wahrscheinlichkeit, um welches Tag/um welchen Tag-Typ es sich bei einem Wort handelt.

Ein Beispiel für POS-Tagging wäre der einfache Satz:

Petra/NE liest/VVFIN einen/ART langen/ADJA Roman/NN

Die Kennungen hinter dem */* stellen Zuordnungen der Wörter zu Wortklassen dar.

Dass *einen* hier ein Artikel ist und nicht ein Verb, lässt sich etwa daraus schließen, dass man durch die Analyse von umfangreichen Textsammlungen *Corpora* mit großer Wahrscheinlichkeit sagen kann, dass auf das Verb *liest* ein Artikel folgt und kein weiteres Verb. POS-Tagging basiert also meist auf großen Textsammlungen und arbeitet mit Wahrscheinlichkeiten.

5.2.1 Stanford Log-linear Part-Of-Speech Tagger

Einer der bekanntesten POS-Tagger ist der *Stanford Log-linear Part-Of-Speech Tagger*.[51]

Die Hauptautorin dieses Taggers, Kristina Toutanova, beschreibt im Vorstellungsartikel für ihn die wichtigsten Eigenschaften solcher Tagger: sie untersuchen Texte meist in der Reihenfolge, in der man die Wörter im Text lesen würde, also unidirektional (Ausnahmen davon werden im Text erwähnt, sind aber für unsere Erwägungen irrelevant)[52]. Diese Tagger beziehen bereits vorher im Text bzw.

[51] STANFORD2005.
[52] TOUTANOVA2003: S. 252.

in der untersuchten Phrase gefundene Erkenntnisse beim Tagging eines Wortes mit ein. Die Besonderheit des Stanford Taggers ist nun, dass er

a) nicht nur unidirektional, sondern zum Taggen sowohl vorherige, als auch nachfolgende Erkenntnisse miteinkalkuliert,
b) lexikalische Analysen durchführt, die ganze Phrasen berücksichtigen, und
c) noch unbekannte Wörter einer sehr fein granulierten Untersuchung unterzieht.

Der dahinterstehende Algorithmus arbeitet mit einem Wahrscheinlichkeitsmodell, das jedes zu taggende Wort als Wahrscheinlichkeitsvariable sieht. Zusammen mit der Bidirektionalität entsteht daraus ein Modell, ein Abhängigkeitsnetzwerk, das in der praktischen Implementierung beim Testen mit einem standardisierten Testtext eine Erkennungsrate von mehr als 97% leistet[53].

Das in Java geschriebene Tagging Programm wird noch heute gewartet, stellt in der Hauptsache ein Modell für Englisch, aber auch Modelle für Französisch, Deutsch, Arabisch, Chinesisch und Spanisch zur Verfügung.

Welche Erkenntnisse können daraus für KALAM gezogen werden:

1. Der Tagger selber kann nicht als Basis für KALAM herangezogen werden, da er essentiell die Wortumgebung verwendet, KALAM aber immer nur ein Wort alleine analysiert.
2. Die Nutzung eines Lexikons im Hintergrund bringt wesentliche Vorteile.
3. Durch genaue Analyse kann man auch das Tagging unbekannter Wörter verbessern.

2. und 3. fließen als wesentliche Säulen in Kalam ein: einerseits durch die Inkludierung eine Sabäisch Wörterbuches, und andererseits in Form einer detaillierten Analyse jedes Wortes.

[53] TOUTANOVA2003: S. 256.

5.2.2 Andere POS-Tagger

Im Folgenden seien noch weitere POS-Tagger, die mit anderen bzw. abgeänderten Methoden arbeiten und in andere Gesamtaufgabengebiete, im Speziellen andere Sprachen, eingebettet sind, betrachtet:

Das Szeged Corpus

Beim Projekt zum Szeged Corpus geht es um die Erstellung eines umfassenden Katalogs von Wörtern und Phrasen des Ungarischen. Die Arbeiten zur Erstellung dieses Katalogs, der Einzelwörter, Wortformen und Punktuationsregeln enthält, erfolgte durch eine grobe Voranalyse durch ein kommerzielles Programm namens Humor[54] und dann durch manuelles POS-Tagging durch Linguistik-Experten.

Als Basis für die erste Version diese Corpus dienten Texte aus fünf verschiedenen Fachgebieten. Den einzelnen Worten wurden in der Detailarbeit sehr detailliert Eigenschaften zugeordnet, um eines der Hauptprobleme des Corpus in den Griff zu bekommen: die Bedeutung der Hälfte der Worte war nicht eindeutig.

Dieser erste Corpus mit 1,2 Millionen Wörtern und 145.000 Wortformen wurde und wird als Basis verwendet, um auch die Texte anderer Fachbereiche zu untersuchen[55].

Diese durch Computer-Hilfe erleichterte, aber doch zu einem Großteil manuelle Vorgangsweise (das Basis-Corpus benötigte etwa 64 Mannmonate), ist wohl nicht geeignet, die Ziele eines KALAM zu verwirklichen. Auch, weil die Ziele eines Corpus, also quasi der vollständigen DNA-Analyse von Teilen einer Sprache, sich nicht mit den Zielen von KALAM decken.

[54] CSENDES2004: S. 41.
[55] Ibid: S. 46.

Automatic Partial Parsing Rule Acquisition

Der hier vorgestellte Ansatz klingt für unsere Zwecke interessant, da er eine Lösung für nur kleine Teile eines Satzes anbietet.

Als Prämisse wird genannt, dass übliche Parser bei langen oder nur teilweise grammatikalisch vollständig ausformulierten Sätzen (wie etwa bei Aufzählungen) Probleme mit dem Parsen bekommen. Die Idee, die in dem Artikel *Automatic Partial Parsing Rule Acquisition Using Decision Tree Induction*[56] vorgestellt wird, ist nun, aus einem Text-Corpus mit anhängendem Baum (siehe Abb. 2) automatisch Parse-Regeln zu erstellen. Diese Methode wird konkret auf Koreanisch angewendet.

Vorgegeben wird eine Regelvorlage, nach der man aus den Textbäumen des Corpus Teilbäume extrahiert. Dieser Vorgang erfolgt ganz automatisch mithilfe eines Programms. Daraus entstehen aber nur Regelkandidaten, die oft Uneindeutigkeiten aufweisen. Um diese zu lösen, werden weiters lexikalische Informationen und Kontext-Wissen aus dem Corpus verwendet. Tiefer wollen wir nicht in die Materie eindringen. Doch können wir schon daraus Schlüsse für KALAM ziehen:

- Wörter alleine können auch mit dieser Reduktion auf Teilsätze nicht analysiert werden.
- Ein umfangreiches Corpus mit anhängendem Baum ist unbedingt nötig.

Also leider wieder keine Verwendbarkeit für KALAM, obwohl wir immerhin die Idee der Zerlegung in kleinere Einheiten mitnehmen können.

[56] CHOI2005: S. 143ff.

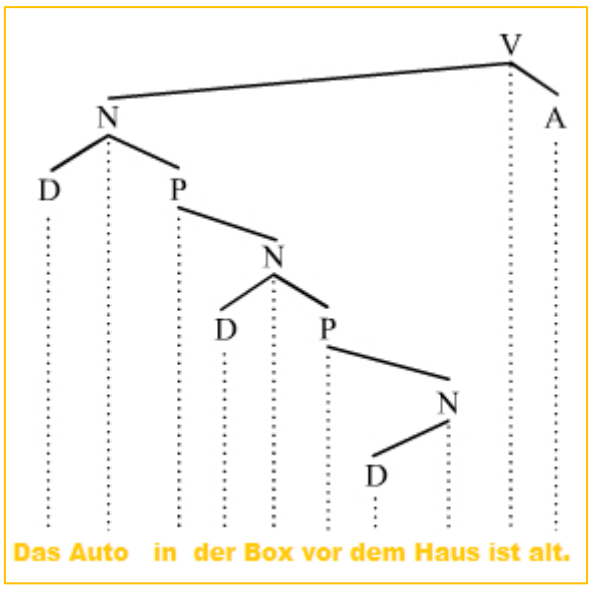


Abb. 2 Text in Baumstruktur

5.3 Ansätze bei semitischen Sprachen

Sehen wir uns noch zwei Ansätze an, die sich mit semitischen Sprachen – also Sprachen, die derselben Sprachfamilie wie das Sabäische entstammen – beschäftigen.

Development of Amharic Morphological Analyzer Using Memory-Based Learning

Dieser Artikel beschäftigt sich mit morphologischer Analyse im Amharischen, also Verfahren, die helfen, die kleinsten Einheiten von Wörtern zu suchen, die linguistische Bedeutungen tragen (frei übersetzt aus der Einleitung dieses Artikels von Mesfin Abate und Yaregal Assabie). Die Autoren merken an, dass der erste Ansatz zur morphologischen Analyse wohl regel-basiert wäre: gerade in so komplexen Sprachen wie dem Amharischen sei aber die Regelerstellung eine sehr aufwändige manuelle Arbeit, die nicht nur Zeit, sondern auch die Bereitstellung linguistischer Experten erfordere.[57] Deshalb beschreite man eher den Weg der Maschinen-Lern-Verfahren, die auf Basis von annotierten oder nicht-annotierten Corpora tätig werden könnten.

[57] ABATE2014: S.1f.

Zwei Wege stehen hier offen:

1. Beaufsichtigtes *supervised* Lernen: lernen an Beispielen
2. Unbeaufsichtigtes *unsupervised* Lernen: lernen an Mustern

Wir können uns hier nicht mit linguistischen Details beschäftigen; die Autoren wählten jedenfalls den 1. Zugang und hierin die Methode des *Memory-based-Learnings*.

Die Morphologie des Amharischen basiert, so wie jene des Sabäischen, auf (meist drei-radikaligen) Wurzeln, doch spielt bei der Bildung davon abgeleiteter Formen der Vokalismus eine große Rolle.

Das vorgestellte Computerprogramm wird im gesamten System in mehreren Phasen verwendet: eine Trainingsphase mit Morphem-Annotationen zur manuellen Annotation von abgeleiteten amharischen Wörtern und weitere manueller Adaption von algorithmischen Parametern des Lern-Werkzeugs. Der zweite Teil beschäftigt sich mit der Dekonstruktion von gegebenen Texten bis hinunter zu Morphemen, Suche nach Stamm und Wurzel.[58] Im ersten Teil kämpften die Autoren insbesondere mit den phonologischen, meist irregulären Änderungen im Bereich der Wurzel.

Die Autoren resümieren, dass nach einem Training mit rund 900 Wörtern eine Erkennungs-Genauigkeit (je nach Testverfahren) von bis zu 96% erreicht werden könnte. Jedenfalls sei noch immer viel manuelle Arbeit nötig und die Performance müsste durch weiteres Training verbessert werden.[59]

Wir können erkennen, dass die Aufgabenstellung eine sehr ähnliche wie bei KALAM ist, allerdings dann aufgrund der Komplexität der Morphologie wieder – wie bei den oben genannten POS-Taggern – ein Trainings- und statistisches Verfahren mit viel Handarbeit einem synthetisierenden Verfahren der Vorzug gegeben wurde. Bei

[58] ABATE2014: S. 3.
[59] Ibid: S.12.

KALAM jedoch haben wir den Vorteil, dass in Form der Grammatik von Stein eine detaillierte Beschreibung vorhanden ist, die nur implementiert werden muss.

Arabic Language Analyzer with Lemma Extraction and Rich Tagset

Textanalyseprogramme für das Arabische gibt es natürlich in einer weitaus vielfältigeren Menge als solche für Amharisch und die große Mehrheit betreibt auch wieder POS-Tagging. Was nun den Artikel mit obigem Titel interessant macht, ist die Verwendung von Lemmatas statt des Stamms. Der Autor selbst beklagt, dass sonstige Arabisch-Analysatoren nur auf bestimmte Teilgebiete, was den Textbereich anlangt, abzielen, er daher ein neues Verfahren erzeugen müsse. Es sei hier angemerkt, dass der Autor allerdings dann zum Training selbst wiederum einen relativ kleinen Corpus von 16.000 Wörtern verwendet und hiermit (innerhalb des Corpus) eine Erkennungsgenauigkeit von rund 99,7 % erreicht – was wiederum nicht verwundert…

Der Autor führt im Artikel, der auf seine Dissertation verweist, selbst die Unterschiede zwischen Lemma und Stamm bzw. Wurzel aus:[60]

1. Ein Lemma ist ein grammatikalisch korrektes Wort, Stamm und Wurzel sind dies nicht.
    - z.B. verwendet der Autor als Lemma für Verben die dritte Person singular, für Hauptworte maskulin singular (wenn vorhanden) – wie es im Arabisch Kontext üblich ist
    - d.h. das Lemma enthält im Gegensatz zur reinen Wurzel zusätzlich Vokalisierungen
2. Lemmatas zu extrahieren, sei schwieriger als Stamm und Wurzel.
3. Mehr als ein Lemma kann denselben Stamm besitzen, mehr als ein Stamm die selbe Wurzel.

[60] ALIWY2012: S. 170.

Die Hauptteile des vom Autor vorgestellten Gesamt-Systems sind ein Tokenizer, ein Morphologie-Analyzer und ein Tagger.

Der Tokenizer zerlegt einen Text in Sätze und normalisiert ihn (Vereinheitlichung von Schreibweisen). Der Morphologie-Analyzer analysiert das Wort und extrahiert Lemmatas – zu diesem Zweck wird ein Lexikon herangezogen; unbekannte Wörter werden einer regel-basierten Analyse unterzogen. Der Tagger erstellt eine POS-Zuordnung zu jedem Wort unter Verwendung eines neu entwickelten Tagsets, seine Hauptaufgabe ist das Lösen von Uneindeutigkeiten.

Ebenso ist der Ansatz sehr interessant, dem – selbstentwickelten – Tagger, andere Tagger (Brill und Maximum Match) als untergeordnete Hilfe zur Verfügung zu stellen.

Dieses System ist sehr weit fortgeschritten, allerdings sehr auf Arabisch zugeschnitten. Die Verwendung vom Lemmatas ist aufgrund der fehlenden Vokalisierung im Südarabischen für KALAM nicht sinnvoll. Allerdings zeigt es auch einige Richtungen auf, die wir weiterverfolgen können:

- Regel-basiert
- Verwendung eines Lexikons
- Verwendung von anderen Taggern zur Unterstützung: dies wäre interessant, wenn man KALAM von der Wort- zur Satzanalyse weiterentwickeln wollte…

5.4 Sabäische Eigenheiten

Gegenüber den oben vorgestellten Sprachen besitzt Sabäisch einige Besonderheiten:

1. Sonderzeichen: die emphatischen Laute, die drei verschiedenen S-Laute und die unterschiedlichen Hs sind Eigenschaften, mit denen manche der oben genannten Tagger nicht zurechtkommt.
2. Keine Vokale: damit kommt insbesondere das zuletzt genannte System nicht zurande.
3. Defektive oder plene Schreibweisen generieren viele Mehrdeutigkeiten, ebenso wie

4. Kurz– und Langformen.
5. Die zeitliche (und örtliche) Variabilität grammatikalischer Regeln – wir haben es hier ja mit mindestens 1400 Jahren sprachlicher Entwicklung zu tun.

6. KALAM

6.1 Programmatisches

Die Suche nach bereits vorhandenen, für die Wortanalyse sabäischer Wörter direkt verwendbaren Werkzeugen ist zwar nun als gescheitert zu betrachten, doch haben wir daraus viel für die Entwicklung vom KALAM gelernt.

Wie schon oben erwähnt – und ein wichtiger Grund ist wohl, dass es sich um eine philologische und keine linguistische Arbeit handelt - wird als Basis für KALAM ein synthetisierendes, Regel-basiertes Verfahren verwendet.

Regeln für alle Formen grammatikalischer Beugungen (inklusive Konjugation von Verben und Deklination von Nomina und Adjektiven) werden vorgegeben, basierend auf den Regeln, die Stein detailliert zusammenfasst[61].

KALAM besteht aus drei Teilen:

1. der Regeldatenbank
2. dem algorithmischen Programm
3. der Wortdatenbank

6.1.1 Regeldatenbank

Die Regeldatenbank enthält unterschiedliche Objekte, deren Klassen hierarchisch miteinander verknüpft sind. Die Klassen und Objekte werden in Englischer Sprache benannt.

TimePeriod

Die Zeitperiode, in der eine grammatikalische Regel gilt. Hier treten die Objekte

- *old-sabaic*

[61] STEIN2013: S. 213ff.

- *middle-sabaic*
- *late-sabic*

und Kombinationen aus diesen, etwa *middle+late-sabaic* auf.

Locality

Diese beschreibt den Ort, an dem eine Regel gilt. Diese Klasse ist für eine mögliche spätere Erweiterung vorgesehen, etwa hin zum Haḍramitischen, Minäischen, Qatabanischen. Derzeit gibt es hierzu nur das Objekt *sabaic*.

Situation

Die Situation ist eine Kombination aus TimePeriod und Locality, ein Beispielobjekt wäre

- *sitmidlate* Ort Saba᾽, Mitte- bis Spätsabäisch

Rule

Jede Regel umfasst ein ganzes Paradigma, so etwa alle Formen der Suffix-Konjugation im Grundstamm. Ein Paar Objekte als Beispiele:

- *v_SK* Suffix-Konjugation im Grundstamm
- *v_SKO2* Suffix-Konjugation im O2-Stamm
- *n_SC* Nomen im Status Constructus
- *dempronear* Demonstrativ Pronomina der Nahdeixis
- *conjunction* Konjunktionen

RuleSituation

Durch Ergänzung der Rule mit einer Situation entsteht eine RuleSituation. Diese Objekte reflektieren unterschiedliche Formen zu unterschiedlichen Zeitpunkten.

So wird z.B. im Altsabäischen die dritte Person, maskulin, dual der Suffixkonjugation im Grundstamm endungslos und in der Zeit Mittel- und Spätsabäisch mit angehängtem ***y*** gebildet. Dies ist ein Beispiel für eine Mater Lectiones, und eine im ersten Fall defektive, im zweiten Fall plene Schreibweise.

Term

Jede ***RuleSituation*** enthält nun Terms, wie etwa die Konjugation in den einzelnen Personen, oder die Deklination in die einzelnen Fälle. Beispiele sind:

- *v_SK_1cs* Verb Suffix Konjugation, Grundstamm, 1. Person singular
- *v_O2SK_2mp* Verb Suffix Konjugation, O2-Stamm, 2. P. maskulin, plural
- *v_HPKN_2ms* Verb Präfix Konjugation, H-Stamm, 2. P. maskulin singular
- *n_LOT1SC* Nomen Loci im T1-Stamm, Status Constructus
- *relpro_md* Relativ Pronomen maskulin dual

Im Term ist die genaue grammatikalische Regel beschrieben, wie man – synthetisch – von der Wurzel zur grammatikalischen Form gelangt.

Das Sabäische baut wie alle semitischen Sprachen auf einem Wurzelsystem mit 3 Radikalen auf, 2- oder 4-radikalige Wurzeln lassen sich meist auf 3-radikalige zurückführen, oder entstammen fremdsprachlichen Wörtern.

Beugungen lassen sich nun zuerst einmal abbilden, indem zwischen den Radikalen Fixe eingefügt werden:

- Präfix
- Infixe
- Suffix

Wie Abb. 3 darstellt, werden bis zu 4 Fixe eingefügt.

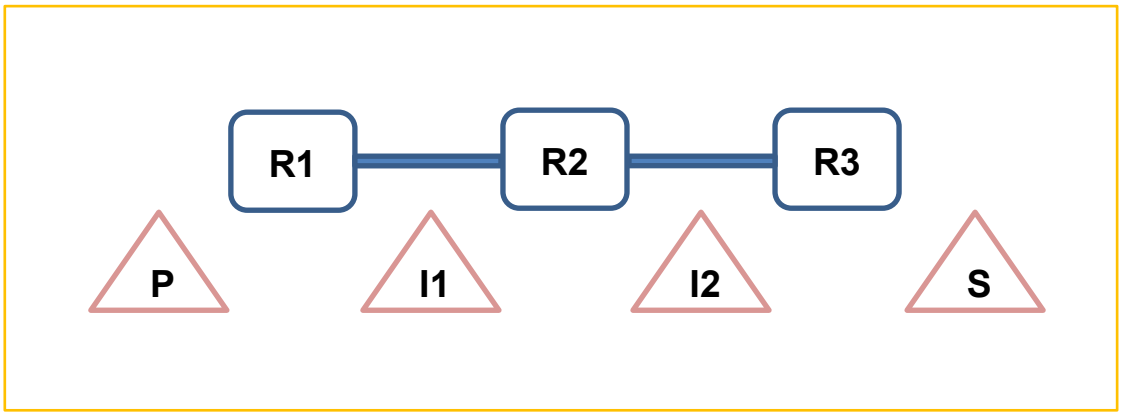


Abb. 3 Radikale und Fixe

Diese Fixe werden nun im Term vermerkt.

Auch könnte es, wie etwa beim Akkadischen, zu Vertauschungen, zur *Metathese* - in den Radikalen kommen – dies wird ebenfalls durch die Nummer der zu vertauschenden Radikale im Term angegeben. Hierzu sind aber nur wenige Beispiele, teilweise aus südlichen Dialektgebieten, z.B. bei Pluralen, bekannt: ʾywn (statt ʾwyn) *Weingärten*, ʾlwd (statt ʾwld) *Kinder*.[62]

Als Spezialfälle werden schwache Radikale vermerkt:

Die Notation lautet hier

```
Radikale = xyz [ , xyz]
```

wobei `xyz` für Schalter 0 und 1 stehen: entfällt der Radikal oder nicht. * steht für einen beliebigen Konsonanten.

Die Form

```
*w*=101,111
```

[62] WOODARD2008: S.150.

besagt, dass im Fall eines **w** im 2. Radikal dieser entfallen kann (`101`) oder nicht (`111`).

Geminierungen werden analog notiert:

```
*..=110
```

Hier entfiele im Term der 3. Radikal, wenn der 2. und 3. Radikal der Wurzel ident sind. Man beachte, dass dieses *Entfallen* rein virtuell ist, da ja die Geminierung des 2. Radikals in der Altsüdarbischen Schrift keinen Niederschlag findet.

Spezialfälle werden ebenso notiert:

```
ydc=011
```

bei der Präfix-0-Konjugation bedeutet z.B., dass beim Wort *yd^{c}* der 1. Radikal entfällt.

Terme werden in der Konfigurationsdatei in einer Zeile durch obenstehende Notation repräsentiert.

Die gesamte Term-Datenbank enthält etwa 700 Objekte.

Rules sind derzeit neben den Verben und Nomina für folgende Wortarten implementiert:

- Kardinal- und Ordinalzahlen
- Bruchteile
- Personal-Pronomen
- Demonstrativ-Pronomen
- Relativ-Pronomen
- Suffixpronomen
    - possesiv, Objekt

- Partikel
- Konjunktionen
- Partizipien
- Nomen Loci

6.1.2 Beispielterme

RuleSituation v_HSK sitmidlate

```
v_HSK_3md h    null null y    0    0    n**=011   *w*=101,111
     *y*=101,111    *..=110,111
```

besagt: die 3. Person maskulin dual im H-Stamm der Suffixkonjugation wird im Mittel- und Spätsabäischen durch

- Präfix H und Suffix Y gebildet.
- Bei 1. Radikal N verschwindet der 1. Radikal (wird assimiliert).
- Bei 2. Radikal W kann der 2. Radikal entfallen.
- Ebenso bei Y.
- Sind der 2. und 3. Radikal gleich, so kann in der Endform der 3. Radikal entfallen.

RuleSituation n_si sitall

```
n_SI_fpo  null null null tm   0    0    n**=011,111
     w**=011,111    *w*=101,111    *y*=101,111
```

besagt: das Nomen im Status Indeterminatus wird zu allen Zeiten im feminin plural im Fall obliqus

- durch ein Suffix `tm` gekennzeichnet;
- bei primä N und W kann der 1. Radikal verschwinden;
- bei secundä W und Y kann der 2. Radikal verschwinden.

RuleSituation conjunction sitall

```
conjunction_09 brt_ where;to where
```

Spezielle Wörter, wie hier die Konjunktion brt, werden als separate Wörter vermerkt und meist auch eine Übersetzung mit angegeben.

6.1.3 Algorithmisches Programm

Die Programmlogik läuft nun wie folgt ab:

Suche nach der Wurzel

Aus dem Wort werden alle Kombinationen aus zwei, drei und vier aufeinanderfolgenden Buchstaben gebildet. So entsteht die erste Liste potentieller Wurzeln.

Die zweite Wurzelliste entsteht, indem geprüft wird, ob ein Präfix-Partikel am Anfang stehen könnte; danach folgen alle Kombinationen wie bei der ersten Liste.

Die dritte Wurzelliste entsteht, indem geprüft wird, ob ein Suffix (Pronominal) am Ende stehen könnte; danach folgen alle Kombinationen vor diesem Suffix wie bei der ersten Liste.

Die vierte Liste wird aus einer Kombination von Präfix, Wurzel und Suffix gebildet.

Synthese

Zu all diesen Wurzelkombinationen werden nun anhand aller Regeln die möglichen Formen gebildet.

Eine gewisse Vorselektion wird bei der Synthese aufgrund der Informationen über Präfix und Suffix vorgenommen. Auch können hier sehr einfach alle Terme weggelassen werden, bei denen schon die Anfangsbuchstaben nicht stimmen.

Vergleich

Die solcherhand produzierten, möglichen Formen werden unter Hinzufügung der angenommenen Präfixe und/oder Suffixe mit dem Ursprungswort verglichen. Jede gültige Form wird mit der grammatikalischen und der situativen Information ausgegeben.

6.1.4 Wortdatenbank

Die Wortdatenbank umfasst derzeit über 2.000 bekannte sabäische Wörter, soll aber in Zukunft nach Möglichkeit noch erweitert werden.

In der Datenbank finden sich Informationen zu

- Wurzeln
- getrennt nach
    - Bedeutungen
    - Typ (Nomen, Verb, Partikel, ...)
- Beispiele
- Übersetzung
- Femininum
- Plural
- Feminin Plural
- Kollektiv
- Infinitiv
- Imperfekt
- Information
- transitiv
- dual

Der Benutzer von KALAM hat drei Möglichkeiten, die Nutzung der Wortdatenbank zu aktivieren:

use dictionary

Diese Option reduziert die Wurzelliste auf Wurzeln, die in der Wortdatenbank vermerkt sind. Dadurch wird die Ergebnisliste sehr den realen Gegebenheiten angepasst. Neue Wortformen verliert man dadurch aber möglicherweise.

entire dictionary

durchsucht die gesamte Wortdatenbank nach Beispielnutzungen des eingegebenen Wortes. Wenn also das Eingabewort in einer Phrase bei einer anderen Wurzel mit verwendet wird, scheint dies in einer separat angezeigten Liste auf.

Englisch word

Hier kann der Benutzer ein englisches Wort eingeben und erhält die sabäische Übersetzung, samt Wurzeln und grammatikalischer Information, ausgegeben.

Das englische Wort wird keiner Untersuchung unterzogen, sondern wird unter den Übersetzungen der Wortdatenbank im Sinn einer Volltextsuche (auch als Teilwort) gesucht.

6.2 Oberflächliches

KALAM ist als Web-Applikation – Java-Servlet auf TomCat-Basis – implementiert.

6.2.1 Einfache Suche

In der Basis-Funktionalität gibt der Benutzer ein sabäisches Wort im Eingabefeld ein und drückt `Search`.

Die Eingabe des Wortes (in Kleinbuchstaben) darf auch alle Sonderzeichen des altsüdarabischen Alphabets enthalten. Diese Sonderzeichen werden alternativ durch

a) die Sonderzeichentastatur, die das Sonderzeichen nach Drücken einer Taste am Ende des Wortes einfügt, siehe Abb. 4,

b) den entsprechenden UTF-8 Code (unter MS-Windows mittels gedrückter Alt-Taste eingebbar),
c) Ersatzzeichen laut der Tabelle in
d) Abb. 5Kopieren aus einer Vorlage

eingegeben.

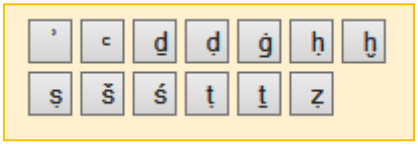


Abb. 4 Sonderzeichentastatur.

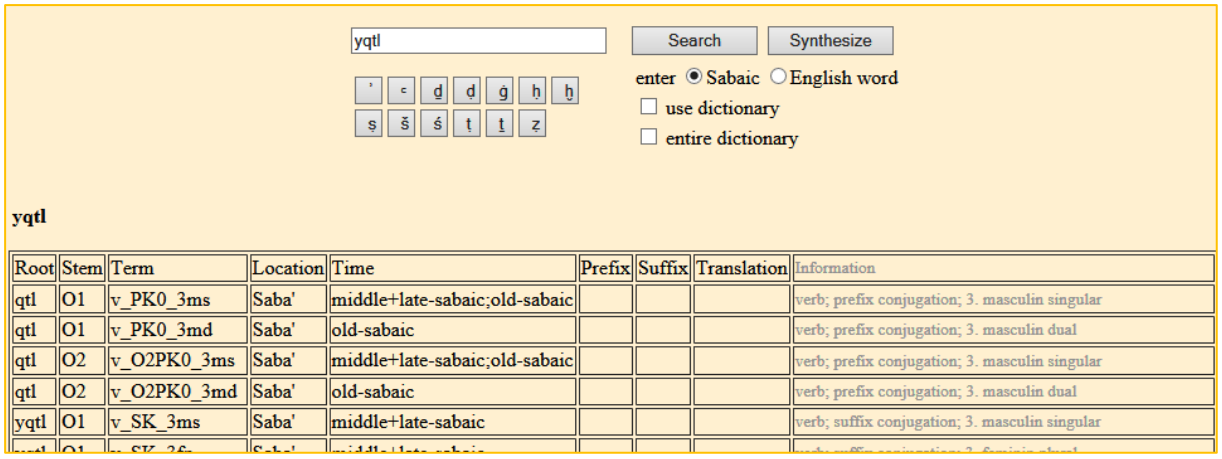
Sonderzeichen	Ersatzzeichen
ʾ	a
ʿ	c
ḏ	d_
ḍ	d.
ġ	g.
ḥ	h.
ḫ	h_
ṣ	s.
š	s^
ś	s‘
ṭ	t.
ṯ	t_
ẓ	z.

Abb. 5 Sonderzeichenersatztabelle.

Daraufhin erscheint eine Liste mit den Spalten

- `Root` gefundene Wurzel
- `Stem` Stamm des Wortes
- `Term` die Objektbenennung des Terms
- `Location` Ort: derzeit nur Saba'
- `Time` Zeitperiode(n) für diesen Term
- `Prefix` optional gefundenes Präfix
- `Suffix` optional gefundenes Suffix
- `Translation` leer
- `Information` genaue grammatikalische Analyse

Die Liste enthält alle theoretisch möglichen Formen; siehe Abb. 6.



Root	Stem	Term	Location	Time	Prefix	Suffix	Translation	Information
qtl	O1	v_PK0_3ms	Saba'	middle+late-sabaic;old-sabaic				verb; prefix conjugation; 3. masculin singular
qtl	O1	v_PK0_3md	Saba'	old-sabaic				verb; prefix conjugation; 3. masculin dual
qtl	O2	v_O2PK0_3ms	Saba'	middle+late-sabaic;old-sabaic				verb; prefix conjugation; 3. masculin singular
qtl	O2	v_O2PK0_3md	Saba'	old-sabaic				verb; prefix conjugation; 3. masculin dual
yqtl	O1	v_SK_3ms	Saba'	middle+late-sabaic				verb; suffix conjugation; 3. masculin singular

Abb. 6 Ergebnis der einfachen Suche

6.2.2 Suchen mit Wörterbuch

Bei der Suche mit Wörterbuch klickt der Benutzer zusätzlich die Option `use dictionary` an. Dann werden nur die tatsächlich im Wörterbuch vorhandenen Wurzeln ausgegeben. Im Feld `Translation` wird auch die Übersetzung ausgegeben.

Allerdings kann es vorkommen, dass zwar die Wurzel, aber nicht der Stamm im Wörterbuch gefunden wurde. In einem solchen Fall steht in der Übersetzungsspalte `not in dictionary`.

Siehe Abb. 7.

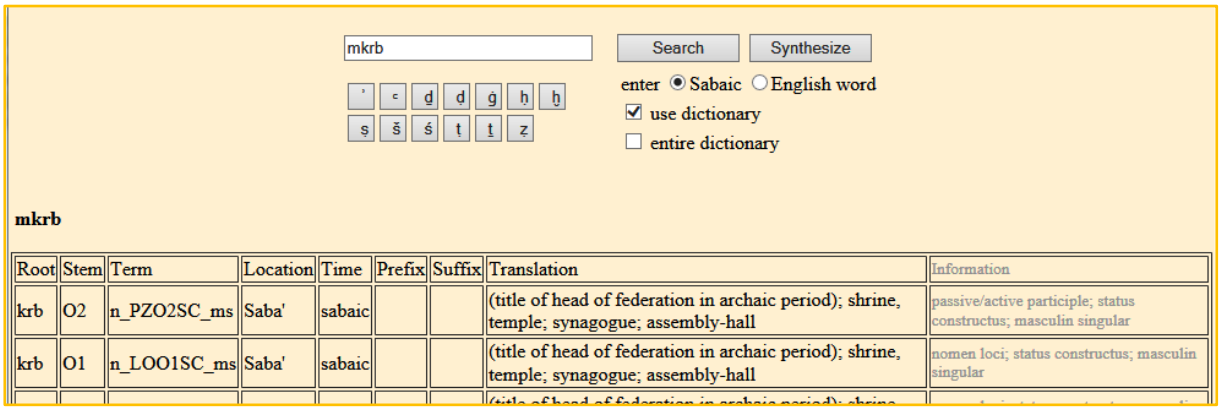
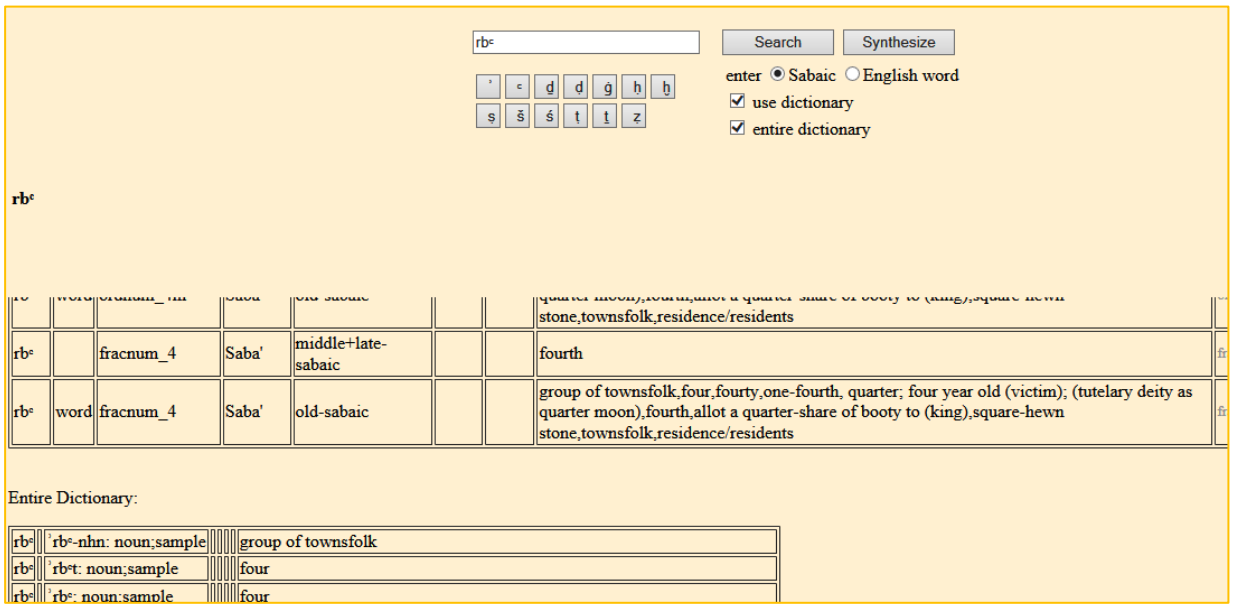


Abb. 7 Ergebnis der Suche mit Wörterbuch.

6.2.3 Ganzes Wörterbuch durchsuchen

Wenn zusätzlich die Option `entire dictionary` gewählt wird, erfolgt zusätzlich zur Ausgabe der ersten Liste laut 6.2.2 die Ausgabe einer zweiten Liste, die alle Einträge des Wörterbuches enthält, die unter den Beispielen das Suchwort mit enthalten, siehe Abb. 8.

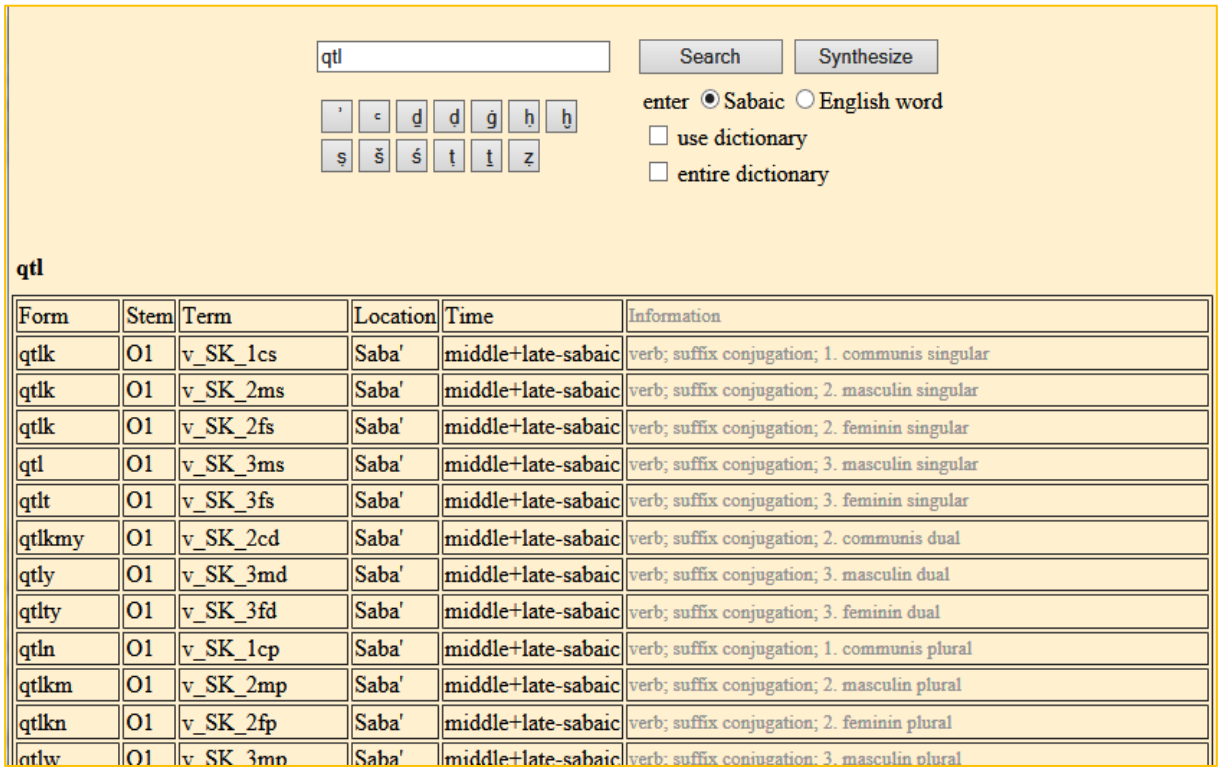


Diese Liste enthält neben der Wurzel die Form, in der das Wort im Wörterbuch vorkommt und die Stelle (z.B. in der ersten Zeile oben: `noun, sample` – also Beispielnomen), sowie die Übersetzung.

Abb. 8 Ausgabe bei der Suche im gesamten Wörterbuch.

6.2.4 *Wortformen*

Durch Eingabe einer Wurzel (dies kann auch eine nicht existierende Wurzel sein!) und Drücken von `Synthesize` wird eine Liste aller theoretischen Formen zu dieser Wurzel generiert, siehe Abb. 9.



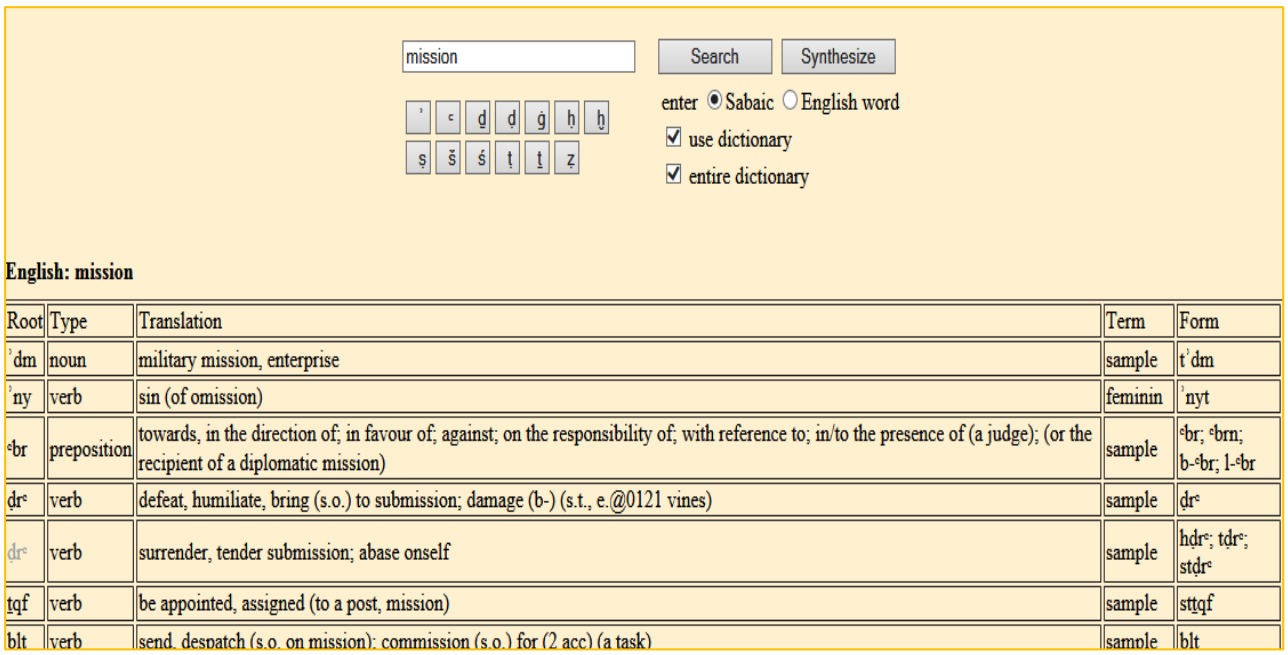
qtl

Form	Stem	Term	Location	Time	Information
qtlk	O1	v_SK_1cs	Saba'	middle+late-sabaic	verb; suffix conjugation; 1. communis singular
qtlk	O1	v_SK_2ms	Saba'	middle+late-sabaic	verb; suffix conjugation; 2. masculin singular
qtlk	O1	v_SK_2fs	Saba'	middle+late-sabaic	verb; suffix conjugation; 2. feminin singular
qtl	O1	v_SK_3ms	Saba'	middle+late-sabaic	verb; suffix conjugation; 3. masculin singular
qtlt	O1	v_SK_3fs	Saba'	middle+late-sabaic	verb; suffix conjugation; 3. feminin singular
qtlkmy	O1	v_SK_2cd	Saba'	middle+late-sabaic	verb; suffix conjugation; 2. communis dual
qtly	O1	v_SK_3md	Saba'	middle+late-sabaic	verb; suffix conjugation; 3. masculin dual
qtlty	O1	v_SK_3fd	Saba'	middle+late-sabaic	verb; suffix conjugation; 3. feminin dual
qtln	O1	v_SK_1cp	Saba'	middle+late-sabaic	verb; suffix conjugation; 1. communis plural
qtlkm	O1	v_SK_2mp	Saba'	middle+late-sabaic	verb; suffix conjugation; 2. masculin plural
qtlkn	O1	v_SK_2fp	Saba'	middle+late-sabaic	verb; suffix conjugation; 2. feminin plural
qtlw	O1	v_SK_3mp	Saba'	middle+late-sabaic	verb; suffix conjugation; 3. masculin plural

Abb. 9 Synthetisierung aller Formen.

6.2.5 *Suche nach englischen Wörtern*

Wird ein englisches Wort im Eingabefeld eingegeben, die Option `English word` gewählt und `Search` gedrückt, so erscheint eine Liste aller Einträge im Wörterbuch, auch wieder nach Wurzeln und Termen getrennt, die diesen englischen Text als Teil eines Wortes enthalten; siehe Abb. 10. Es handelt sich also um eine Volltextsuche in den Übersetzungstexten.



Root	Type	Translation	Term	Form
ʾdm	noun	military mission, enterprise	sample	tʾdm
ʾny	verb	sin (of omission)	feminin	ʾnyt
ᶜbr	preposition	towards, in the direction of; in favour of; against; on the responsibility of; with reference to; in/to the presence of (a judge); (or the recipient of a diplomatic mission)	sample	ᶜbr; ᶜbrn; b-ᶜbr; l-ᶜbr
ḏrᶜ	verb	defeat, humiliate, bring (s.o.) to submission; damage (b-) (s.t., e.@0121 vines)	sample	ḏrᶜ
ḏrᶜ	verb	surrender, tender submission; abase onself	sample	hḏrᶜ; tḏrᶜ; stḏrᶜ
tqf	verb	be appointed, assigned (to a post, mission)	sample	sttqf
blt	verb	send, despatch (s.o. on mission); commission (s.o.) for (2 acc) (a task)	sample	blt

Abb. 10 Suche nach englischen Begriffen.

6.3 Beispiele

Sabaäische Form ʾḥmdkmw

Root	Stem	Term	Location	Time	Prefix	Suffix	Translation	Information
ḥmd	O1	v_PK0_1cs	Saba'	middle+late-sabaic;old-sabaic		ppros_2mp	praise, thank,win praise	verb; prefix conjugation; 1. communis singular
ḥmd	O2	v_O2PK0_1cs	Saba'	middle+late-sabaic;old-sabaic		ppros_2mp	praise, thank,win praise	verb; prefix conjugation; 1. communis singular

bedeutet also *ich danke euch*.

Englische Wort: thank

Root	Type	Translation	Term	Form
ḥmd	verb	praise, thank	sample	ḥmd
ḥmd	noun	praise, thank; thankfulness; glory	sample	ḥmd
ṯwb	verb	decree, ordain; hand over, assign, render (to (l-) s.o.); record,	sample	hṯb

		testify (thanks/confidence); repair; execute completely; repulse		
t̲wb	verb	decree, ordain; hand over, assign, render (to (l-) s.o.); record, testify (thanks/confidence); repair; execute completely; repulse	plural	ht̲bw
t̲wb	verb	decree, ordain; hand over, assign, render (to (l-) s.o.); record, testify (thanks/confidence); repair; execute completely; repulse	infinitive	ht̲bn
t̲wb	verb	decree, ordain; hand over, assign, render (to (l-) s.o.); record, testify (thanks/confidence); repair; execute completely; repulse	imperfect	yht̲b

liefert also das gesuchte ḥmd ebenso wie Formen des nur indirekt damit zusammenhängenden t̲wb.

Sabäisches Nomen mit Präfix und Suffix **bmḥfdk**

Root	Stem	Term	Location	Time	Prefix	Suffix	Translation	Information
ḥfd	O2	n_PZO2SC_ms	Saba'	sabaic	part_b	ppros_2ms	in;with tower; projecting element (of wall)	passive/active participle; status constructus; masculin singular
ḥfd	O1	n_LOO1SC_ms	Saba'	sabaic	part_b	ppros_2ms	in;with tower; projecting element (of wall)	nomen loci; status constructus; masculin singular
ḥfd	O2	n_LOO2SC_ms	Saba'	sabaic	part_b	ppros_2ms	in;with tower; projecting element (of wall)	nomen loci; status constructus; masculin singular

bedeutet *in/mit deinem Turm*

7. Regeldatenbank

Die untenstehende Regeldatenbank umfasst alle bisher bekannten Paradigmen der sabäischen Grammatik bis auf die in dieser Arbeit nicht erfassten internen Plurale in einer formalisierten Sprache. Berücksichtigt wird das „Hochsabäische“ über alle Zeitperioden.

```
*v_SK sitmidlate  O1
v_SK_stem   null  null  null  null  0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_1cs    null  null  null  k     0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_2ms    null  null  null  k     0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_2fs    null  null  null  k     0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_3ms    null  null  null  null  0     0     *w*=101,111 *y*=101,111
v_SK_3fs    null  null  null  t     0     0     *w*=101,111 *y*=101,111
v_SK_2cd    null  null  null  kmy   0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_3md    null  null  null  y     0     0     *w*=101,111 *y*=101,111
v_SK_3fd    null  null  null  ty    0     0     *w*=101,111 *y*=101,111
v_SK_1cp    null  null  null  n     0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_2mp    null  null  null  km    0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_2fp    null  null  null  kn    0     0     *w*=101,111 *y*=101,111
      **w=110     **y=110
v_SK_3mp    null  null  null  w     0     0     *w*=101,111 *y*=101,111
v_SK_3fp    null  null  null  null  0     0     *w*=101,111 *y*=101,111
*v_SK sitold      O1    =sitmidlate
v_SK_3md    null  null  null  null  0     0
v_SK_3fd    null  null  null  t     0     0
*v_PK0      sitmidlate  O1
v_PK0_stem  null  null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_1cs   a     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
```

```
v_PK0_2ms   t     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_2fs   t     null  null  n     0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3ms   y     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3fs   t     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3md   y     null  null  y     0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_1cp   n     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3mp   t     null  null  n     0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3mp   y     null  null  n     0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
v_PK0_3fp   t     null  null  null  0     0     n**=011     w**=011
      *w*=101,111 *y*=101,111 *..=110     ydc=011
*v_PK0      sitold      O1
v_PK0_stem  null  null  null  null  0     0
v_PK0_1cs   a     null  null  null  0     0
v_PK0_2ms   t     null  null  null  0     0
v_PK0_2fs   t     null  null  n     0     0
v_PK0_3ms   y     null  null  null  0     0
v_PK0_3fs   t     null  null  null  0     0
v_PK0_3md   y     null  null  null  0     0
v_PK0_1cp   n     null  null  null  0     0
v_PK0_3mp   t     null  null  n     0     0
v_PK0_3mp   y     null  null  n     0     0
v_PK0_3fp   t     null  null  null  0     0
*v_PKN      sitall      O1
v_PKN_stem  null  null  null  null  0     0
v_PKN_2ms   t     null  null  n     0     0
v_PKN_2fs   t     null  null  n     0     0
v_PKN_3ms   y     null  null  n     0     0
v_PKN_3fs   t     null  null  n     0     0
v_PKN_2cd   t     null  null  nn    0     0
v_PKN_3md   y     null  null  nn    0     0
v_PKN_3fd   t     null  null  nn    0     0
v_PKN_2mp   t     null  null  nn    0     0
```

```
v_PKN_3mp   y     null  null  nn    0     0
v_PKN_3fp   t     null  null  nn    0     0
*v_IMPk     sitall      O1
v_IMPk_stem null  null  null  null  0     0
v_IMPk_2ms  null  null  null  null  0     0
v_IMPk_2cd  null  null  null  n     0     0
v_IMPk_2mp  null  null  null  n     0     0
*v_IMPl     sitall      O1
v_IMPl_stem null  null  null  null  0     0
v_IMPl_2ms  null  null  null  n     0     0
v_IMPl_2mp  null  null  null  nn    0     0
*v_INF      sitall      O1
v_INF null  null  null  null  0     0
*v_O2SK     sitmidlate  O2
v_O2SK_stem null  null  null  null  0     0
v_O2SK_1cs  null  null  null  k     0     0
v_O2SK_2ms  null  null  null  k     0     0
v_O2SK_2fs  null  null  null  k     0     0
v_O2SK_3ms  null  null  null  null  0     0
v_O2SK_3fs  null  null  null  t     0     0
v_O2SK_2cd  null  null  null  kmy   0     0
v_O2SK_3md  null  null  null  y     0     0
v_O2SK_3fd  null  null  null  ty    0     0
v_O2SK_1cp  null  null  null  n     0     0
v_O2SK_2mp  null  null  null  km    0     0
v_O2SK_2fp  null  null  null  kn    0     0
v_O2SK_3mp  null  null  null  w     0     0
v_O2SK_3fp  null  null  null  null  0     0
*v_O2SK     sitold      O2    =sitmidlate
v_O2SK_3md  null  null  null  null  0     0
v_O2SK_3fd  null  null  null  t     0     0
*v_O2PK0    sitmidlate  O2
v_O2PK0_stem      null  null  null  null  0     0
v_O2PK0_1cs a     null  null  null  0     0
v_O2PK0_2ms t     null  null  null  0     0
v_O2PK0_2fs t     null  null  n     0     0
v_O2PK0_3ms y     null  null  null  0     0
v_O2PK0_3fs t     null  null  null  0     0
v_O2PK0_3md y     null  null  y     0     0
v_O2PK0_1cp n     null  null  null  0     0
```

```
v_O2PK0_3mp t     null  null  n     0     0
v_O2PK0_3mp y     null  null  n     0     0
v_O2PK0_3fp t     null  null  null  0     0
*v_O2PK0    sitold      O2    =sitmidlate
v_O2PK0_3md y     null  null  null  0     0
*v_O2PKN    sitall      O2
v_O2PKN_stem      null  null  null  null  0     0
v_O2PKN_2ms t     null  null  n     0     0
v_O2PKN_2fs t     null  null  n     0     0
v_O2PKN_3ms y     null  null  n     0     0
v_O2PKN_3fs t     null  null  n     0     0
v_O2PKN_2cd t     null  null  nn    0     0
v_O2PKN_3md y     null  null  nn    0     0
v_O2PKN_3fd t     null  null  nn    0     0
v_O2PKN_2mp t     null  null  nn    0     0
v_O2PKN_3mp y     null  null  nn    0     0
v_O2PKN_3fp t     null  null  nn    0     0
*v_O2IMPk   sitall      O2
v_O2IMPk_stem     null  null  null  null  0     0
v_O2IMPk_2ms      null  null  null  null  0     0
v_O2IMPk_2cd      null  null  null  n     0     0
v_O2IMPk_2mp      null  null  null  n     0     0
*v_O2IMPl   sitall      O2
v_O2IMPl_stem     null  null  null  null  0     0
v_O2IMPl_2ms      null  null  null  n     0     0
v_O2IMPl_2mp      null  null  null  nn    0     0
*v_O2INF    sitall      O2
v_O2INF     null  null  null  n     0     0
*v_HSK      sitmidlate  H
v_HSK_stem  h     null  null  null  0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_1cs   h     null  null  k     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_2ms   h     null  null  k     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_2fs   h     null  null  k     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_3ms   h     null  null  null  0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
```

```
v_HSK_3fs   h     null  null  t     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_2cd   h     null  null  kmy   0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_3md   h     null  null  y     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_3fd   h     null  null  ty    0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_1cp   h     null  null  n     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_2mp   h     null  null  km    0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_2fp   h     null  null  kn    0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_3mp   h     null  null  w     0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
v_HSK_3fp   h     null  null  null  0     0     n**=011     *w*=101,111
      *y*=101,111 *..=110,111
*v_HSK      sitold      H     =sitmidlate
v_HSK_3md   h     null  null  null  0     0
v_HSK_3fd   h     null  null  t     0     0
*v_HPK0     sitmidlate  H
v_HPK0_stem ah    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_1cs  ah    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_2ms  th    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_2fs  th    null  null  n     0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_3ms  yh    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_3fs  th    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_3md  yh    null  null  y     0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_1cp  nh    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_3mp  th    null  null  n     0     0     n**=011     *w*=101
      *y*=101     *..=110,111
```

```
v_HPK0_3mp  yh    null  null  n     0     0     n**=011     *w*=101
      *y*=101     *..=110,111
v_HPK0_3fp  th    null  null  null  0     0     n**=011     *w*=101
      *y*=101     *..=110,111
*v_HPK0     sitold      H     =sitmidlate
v_HPK0_3md  yh    null  null  null  0     0
*v_HPKN     sitall      H
v_HPKN_stem h     null  null  null  0     0
v_HPKN_2ms  th    null  null  n     0     0
v_HPKN_2fs  th    null  null  n     0     0
v_HPKN_3ms  yh    null  null  n     0     0
v_HPKN_3fs  th    null  null  n     0     0
v_HPKN_2cd  th    null  null  nn    0     0
v_HPKN_3md  yh    null  null  nn    0     0
v_HPKN_3fd  th    null  null  nn    0     0
v_HPKN_2mp  th    null  null  nn    0     0
v_HPKN_3mp  yh    null  null  nn    0     0
v_HPKN_3fp  th    null  null  nn    0     0
*v_HIMPk    sitall      H
v_HIMPk_stem      h     null  null  null  0     0     n**=011
      *..=110,111
v_HIMPk_2ms h     null  null  null  0     0     n**=011     *..=110,111
v_HIMPk_2cd h     null  null  n     0     0     n**=011     *..=110,111
v_HIMPk_2mp h     null  null  n     0     0     n**=011     *..=110,111
*v_HIMPl    sitall      H
v_HIMPl_stem      h     null  null  null  0     0
v_HIMPl_2ms h     null  null  n     0     0
v_HIMPl_2mp h     null  null  nn    0     0
*v_HINF     sitall      H
v_HINF      h     null  null  n     0     0     n**=011     *w*=101
      *y*=101     *..=110,111
*v_T1SK     sitmidlate  T1
v_T1SK_stem t     null  null  null  0     0     *..=110
v_T1SK_1cs  t     null  null  k     0     0     *..=110
v_T1SK_2ms  t     null  null  k     0     0     *..=110
v_T1SK_2fs  t     null  null  k     0     0     *..=110
v_T1SK_3ms  t     null  null  null  0     0     *..=110
v_T1SK_3fs  t     null  null  t     0     0     *..=110
v_T1SK_2cd  t     null  null  kmy   0     0     *..=110
v_T1SK_3md  t     null  null  y     0     0     *..=110
```

```
v_T1SK_3fd  t     null  null  ty    0     0     *..=110
v_T1SK_1cp  t     null  null  n     0     0     *..=110
v_T1SK_2mp  t     null  null  km    0     0     *..=110
v_T1SK_2fp  t     null  null  kn    0     0     *..=110
v_T1SK_3mp  t     null  null  w     0     0     *..=110
v_T1SK_3fp  t     null  null  null  0     0     *..=110
*v_T1SK     sitold      T1    =sitmidlate
v_T1SK_3md  t     null  null  null  0     0
v_T1SK_3fd  t     null  null  t     0     0
*v_T1PK0    sitmidlate  T1
v_T1PK0_stem      t     null  null  null  0     0     n**=011     w**=011
v_T1PK0_1cs a     t     null  null  0     0     n**=011     w**=011
v_T1PK0_2ms t     t     null  null  0     0     n**=011     w**=011
v_T1PK0_2fs t     t     null  n     0     0     n**=011     w**=011
v_T1PK0_3ms y     t     null  null  0     0     n**=011     w**=011
v_T1PK0_3fs t     t     null  null  0     0     n**=011     w**=011
v_T1PK0_3md y     t     null  y     0     0     n**=011     w**=011
v_T1PK0_1cp n     t     null  null  0     0     n**=011     w**=011
v_T1PK0_3mp t     t     null  n     0     0     n**=011     w**=011
v_T1PK0_3mp y     t     null  n     0     0     n**=011     w**=011
v_T1PK0_3fp t     t     null  null  0     0     n**=011     w**=011
*v_T1PK0    sitold      T1    =sitmidlate
v_T1PK0_3md y     t     null  null  0     0
*v_T1PKN    sitall      T1
v_T1PKN_stem      t     null  null  null  0     0
v_T1PKN_2ms t     t     null  n     0     0
v_T1PKN_2fs t     t     null  n     0     0
v_T1PKN_3ms y     t     null  n     0     0
v_T1PKN_3fs t     t     null  n     0     0
v_T1PKN_2cd t     t     null  nn    0     0
v_T1PKN_3md y     t     null  nn    0     0
v_T1PKN_3fd t     t     null  nn    0     0
v_T1PKN_2mp t     t     null  nn    0     0
v_T1PKN_3mp y     t     null  nn    0     0
v_T1PKN_3fp t     t     null  nn    0     0
*v_T1IMPk   sitall      T1
v_T1IMPk_stem     t     null  null  null  0     0     n**=011     w**=011
v_T1IMPk_2ms      t     null  null  null  0     0     n**=011     w**=011
v_T1IMPk_2cd      t     null  null  n     0     0     n**=011     w**=011
v_T1IMPk_2mp      t     null  null  n     0     0     n**=011     w**=011
```

```
*v_T1IMPl   sitall      T1
v_T1IMPl_stem     t     null  null  null  0     0
v_T1IMPl_2ms      t     null  null  n     0     0
v_T1IMPl_2mp      t     null  null  nn    0     0
*v_T1INF    sitall      T1
v_T1INF     null  t     null  n     0     0
*v_T2SK     sitmidlate  T2
v_T2SK_stem t     null  null  null  0     0
v_T2SK_1cs  t     null  null  k     0     0
v_T2SK_2ms  t     null  null  k     0     0
v_T2SK_2fs  t     null  null  k     0     0
v_T2SK_3ms  t     null  null  null  0     0
v_T2SK_3fs  t     null  null  t     0     0
v_T2SK_2cd  t     null  null  kmy   0     0
v_T2SK_3md  t     null  null  y     0     0
v_T2SK_3fd  t     null  null  ty    0     0
v_T2SK_1cp  t     null  null  n     0     0
v_T2SK_2mp  t     null  null  km    0     0
v_T2SK_2fp  t     null  null  kn    0     0
v_T2SK_3mp  t     null  null  w     0     0
v_T2SK_3fp  t     null  null  null  0     0
*v_T2SK     sitold      T2    =sitmidlate
v_T2SK_3md  t     null  null  null  0     0
v_T2SK_3fd  t     null  null  t     0     0
*v_T2PK0    sitmidlate  T2
v_T2PK0_stem      t     null  null  null  0     0
v_T2PK0_1cs at    null  null  null  0     0
v_T2PK0_2ms tt    null  null  null  0     0
v_T2PK0_2fs tt    null  null  n     0     0
v_T2PK0_3ms yt    null  null  null  0     0
v_T2PK0_3fs tt    null  null  null  0     0
v_T2PK0_3md yt    null  null  y     0     0
v_T2PK0_1cp nt    null  null  null  0     0
v_T2PK0_3mp tt    null  null  n     0     0
v_T2PK0_3mp yt    null  null  n     0     0
v_T2PK0_3fp tt    null  null  null  0     0
*v_T2PK0    sitold      T2    =sitmidlate
v_T2PK0_3md yt    null  null  null  0     0
*v_T2PKN    sitall      T2
v_T2PKN_stem      t     null  null  null  0     0
```

```
v_T2PKN_2ms tt    null  null  n     0     0
v_T2PKN_2fs tt    null  null  n     0     0
v_T2PKN_3ms yt    null  null  n     0     0
v_T2PKN_3fs tt    null  null  n     0     0
v_T2PKN_2cd tt    null  null  nn    0     0
v_T2PKN_3md yt    null  null  nn    0     0
v_T2PKN_3fd tt    null  null  nn    0     0
v_T2PKN_2mp tt    null  null  nn    0     0
v_T2PKN_3mp yt    null  null  nn    0     0
v_T2PKN_3fp tt    null  null  nn    0     0
*v_T2IMPk   sitall      T2
v_T2IMPk_stem     t     null  null  null  0     0
v_T2IMPk_2ms      t     null  null  null  0     0
v_T2IMPk_2cd      t     null  null  n     0     0
v_T2IMPk_2mp      t     null  null  n     0     0
*v_T2IMPl   sitall      T2
v_T2IMPl_stem     t     null  null  null  0     0
v_T2IMPl_2ms      t     null  null  n     0     0
v_T2IMPl_2mp      t     null  null  nn    0     0
*v_T2INF    sitall      T2
v_T2INF     t     null  null  n     0     0
*v_STSK     sitmidlate  ST
v_STSK_stem st    null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STSK_1cs  st    null  null  k     0     0     n**=011     *w*=101
      *y*=101
v_STSK_2ms  st    null  null  k     0     0     n**=011     *w*=101
      *y*=101
v_STSK_2fs  st    null  null  k     0     0     n**=011     *w*=101
      *y*=101
v_STSK_3ms  st    null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STSK_3fs  st    null  null  t     0     0     n**=011     *w*=101
      *y*=101
v_STSK_2cd  st    null  null  kmy   0     0     n**=011     *w*=101
      *y*=101
v_STSK_3md  st    null  null  y     0     0     n**=011     *w*=101
      *y*=101
v_STSK_3fd  st    null  null  ty    0     0     n**=011     *w*=101
      *y*=101
```

```
v_STSK_1cp  st    null  null  n     0     0     n**=011     *w*=101
      *y*=101
v_STSK_2mp  st    null  null  km    0     0     n**=011     *w*=101
      *y*=101
v_STSK_2fp  st    null  null  kn    0     0     n**=011     *w*=101
      *y*=101
v_STSK_3mp  st    null  null  w     0     0     n**=011     *w*=101
      *y*=101
v_STSK_3fp  st    null  null  null  0     0     n**=011     *w*=101
      *y*=101
*v_STSK     sitold      ST    =sitmidlate
v_STSK_3md  st    null  null  null  0     0
v_STSK_3fd  st    null  null  t     0     0
*v_STPK0    sitmidlate  ST
v_STPK0_stem      st    null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_1cs ast   null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_2ms tst   null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_2fs tst   null  null  n     0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3ms yst   null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3fs tst   null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3md yst   null  null  y     0     0     n**=011     *w*=101
      *y*=101
v_STPK0_1cp nst   null  null  null  0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3mp tst   null  null  n     0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3mp yst   null  null  n     0     0     n**=011     *w*=101
      *y*=101
v_STPK0_3fp tst   null  null  null  0     0     n**=011     *w*=101
      *y*=101
*v_STPK0    sitold      ST    =sitmidlate
v_STPK0_3md yst   null  null  null  0     0
*v_STPKN    sitall      ST
v_STPKN_stem      st    null  null  null  0     0
```

```
v_STPKN_2ms tst   null  null  n     0     0
v_STPKN_2fs tst   null  null  n     0     0
v_STPKN_3ms yst   null  null  n     0     0
v_STPKN_3fs tst   null  null  n     0     0
v_STPKN_2cd tst   null  null  nn    0     0
v_STPKN_3md yst   null  null  nn    0     0
v_STPKN_3fd tst   null  null  nn    0     0
v_STPKN_2mp tst   null  null  nn    0     0
v_STPKN_3mp yst   null  null  nn    0     0
v_STPKN_3fp tst   null  null  nn    0     0
*v_STIMPk   sitall      ST
v_STIMPk_stem     st    null  null  null  0     0
v_STIMPk_2ms      st    null  null  null  0     0
v_STIMPk_2cd      st    null  null  n     0     0
v_STIMPk_2mp      st    null  null  n     0     0
*v_STIMPl   sitall      ST
v_STIMPl_stem     st    null  null  null  0     0     *w*=101     *y*=101
v_STIMPl_2ms      st    null  null  n     0     0     *w*=101     *y*=101
v_STIMPl_2mp      st    null  null  nn    0     0     *w*=101     *y*=101
*v_STINF    sitall      ST
v_STINF     st    null  null  n     0     0
*part sitall      prefix
part_b      b     null  null  null  0     0     in;with
part_k      k     null  null  null  0     0     like
part_l      l     null  null  null  0     0     to;for
part_s      s     null  null  null  0     0     at;near to
part_w      w     null  null  null  0     0     and
*ppros      sitall      suffix
ppros_1cs   null  null  null  n     0     0
ppros_2ms   null  null  null  k     0     0
ppros_2fs   null  null  null  k     0     0
ppros_2md   null  null  null  kmy   0     0
ppros_2fd   null  null  null  kmy   0     0
ppros_3md   null  null  null  hmy   0     0
ppros_3fd   null  null  null  hmy   0     0
ppros_1cp   null  null  null  n     0     0
ppros_2mp   null  null  null  kmw   0     0
ppros_2fp   null  null  null  kn    0     0
ppros_3mp   null  null  null  hmw   0     0
ppros_3fp   null  null  null  hn    0     0
```

```
*relp sitoldmid   prefix
relp_s      d_    null  null  null  0     0
*relp sitlate     prefix
relp_ms     d_    null  null  null  0     0
relp_fs     t_    null  null  null  0     0
*n_SC sitmidlate  SC
n_SC_stem   null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_ms     null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_fs     null  null  null  t     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_mdn    null  null  null  y     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_fdn    null  null  null  ty    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_mdo    null  null  null  y     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_fdo    null  null  null  ty    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_mpn    null  null  null  w     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_fpn    null  null  null  t     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_mpo    null  null  null  y     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SC_fpo    null  null  null  t     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_SC sitold      SC    =sitmidlate
n_SC_mdn    null  null  null  null  0     0
n_SC_fdn    null  null  null  t     0     0
*n_SI sitall      SI
n_SI_stem   null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_ms     null  null  null  m     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_fs     null  null  null  tm    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_mdn    null  null  null  n     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
```

```
n_SI_fdn    null  null  null  tn    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_mdo    null  null  null  n     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_fdo    null  null  null  tn    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_mpn    null  null  null  n     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_fpn    null  null  null  tm    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_mpo    null  null  null  n     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SI_fpo    null  null  null  tm    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_SD sitall      SD
n_SD_stem   null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_ms     null  null  null  n     0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_fs     null  null  null  tn    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_mdn    null  null  null  nhn   0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_fdn    null  null  null  tnhn  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_mdo    null  null  null  nhn   0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_fdo    null  null  null  tnhn  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_mpn    null  null  null  nhn   0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_fpn    null  null  null  tn    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_mpo    null  null  null  nhn   0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
n_SD_fpo    null  null  null  tn    0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZO1SC   sitall      O1    *n_SC
n_PZO1SC    null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
```

```
*n_PZO1SI   sitall      O1    *n_SI
n_PZO1SI    null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZO1SD   sitall      O1    *n_SD
n_PZO1SD    null  null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZO2SC   sitall      O2    *n_SC
n_PZO2SC    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZO2SI   sitall      O2    *n_SI
n_PZO2SI    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZO2SD   sitall      O2    *n_SD
n_PZO2SD    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZHSC    sitall      H     *n_SC
n_PZHSC     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZHSI    sitall      H     *n_SI
n_PZHSI     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZHSD    sitall      H     *n_SD
n_PZHSD     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT1SC   sitall      T1    *n_SC
n_PZT1SC    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT1SI   sitall      T1    *n_SI
n_PZT1SI    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT1SD   sitall      T1    *n_SD
n_PZT1SD    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT2SC   sitall      T2    *n_SC
n_PZT2SC    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT2SI   sitall      T2    *n_SI
n_PZT2SI    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZT2SD   sitall      T2    *n_SD
```

```
n_PZT2SD    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZSTSC   sitall      ST    *n_SC
n_PZSTSC    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZSTSI   sitall      ST    *n_SI
n_PZSTSI    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_PZSTSD   sitall      ST    *n_SD
n_PZSTSD    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO1SC   sitall      O1    *n_SC
n_LOO1SC    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO1SI   sitall      O1    *n_SI
n_LOO1SI    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO1SD   sitall      O1    *n_SD
n_LOO1SD    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO2SC   sitall      O2    *n_SC
n_LOO2SC    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO2SI   sitall      O2    *n_SI
n_LOO2SI    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOO2SD   sitall      O2    *n_SD
n_LOO2SD    m     null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOHSC    sitall      H     *n_SC
n_LOHSC     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOHSI    sitall      H     *n_SI
n_LOHSI     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOHSD    sitall      H     *n_SD
n_LOHSD     mh    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT1SC   sitall      T1    *n_SC
```

```
n_LOT1SC    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT1SI   sitall      T1    *n_SI
n_LOT1SI    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT1SD   sitall      T1    *n_SD
n_LOT1SD    m     t     null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT2SC   sitall      T2    *n_SC
n_LOT2SC    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT2SI   sitall      T2    *n_SI
n_LOT2SI    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOT2SD   sitall      T2    *n_SD
n_LOT2SD    mt    null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOSTSC   sitall      ST    *n_SC
n_LOSTSC    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOSTSI   sitall      ST    *n_SI
n_LOSTSI    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*n_LOSTSD   sitall      ST    *n_SD
n_LOSTSD    mst   null  null  null  0     0     n**=011,111 w**=011,111
      *w*=101,111 *y*=101,111
*cardnum    sitmidlate  word
cardnum_1m  ah.d  one
cardnum_1f  ah.t  one
cardnum_2m  t_ny  two
cardnum_2f  t_ty  two
cardnum_3m  t_lt_ three
cardnum_3f  t_lt_t      three
cardnum_4m  arbc  four
cardnum_4f  arbct four
cardnum_5m  h_ms  five
cardnum_5f  h_mst five
cardnum_6m  st_   six
cardnum_6f  st_t  six
cardnum_7m  sbc   seven
```

```
cardnum_7f  sbct  seven
cardnum_8m  t_mn  eight
cardnum_8f  t_mnt eight
cardnum_9m  tsc   nine
cardnum_9f  tsct  nine
cardnum_10m cs^r  ten
cardnum_10f cs^rt ten
cardnum_20  cs^ry twenty
cardnum_30  t_lt_y      thirty
cardnum_40  arbcy fourty
cardnum_50  h_msy fifty
cardnum_60  st_y  sixty
cardnum_70  sbcy  seventy
cardnum_80  t_mnyy      eighty
cardnum_90  tscy  ninety
*cardnum    sitold      word  =sitmidlate
cardnum_3m  s^lt_ three
cardnum_3f  s^lt_t      three
cardnum_6m  sdt_  six
cardnum_6f  sdt_t six
cardnum_8m  t_mny eight
cardnum_8f  t_mnt eight
cardnum_8f  t_mnyt      eight
cardnum_30  s^lt_y      thirty
cardnum_60  sdt_y sixty
*ordnum     sitmidlate  word
ordnum_1m   qdm   first
ordnum_1f   qdmt  first
ordnum_2m   t_ny  second
ordnum_2f   t_nyt second
ordnum_2f   t_nt  second
ordnum_3m   t_lt_ third
ordnum_3f   t_lt_t      third
ordnum_4m   rbc   fourth
ordnum_4f   rbct  fourth
ordnum_5m   h_ms  fivth
ordnum_5f   h_mst fivth
ordnum_6m   sdt_  sixth
ordnum_6f   sdt_t sixth
ordnum_7m   sbc   seventh
```

```
ordnum_7f   sbct  seventh
ordnum_8m   t_mn  eightth
ordnum_8f   t_mnt eightth
ordnum_9m   tsc   nineth
ordnum_9f   tsct  nineth
ordnum_10m  cs^r  tenth
ordnum_10f  cs^rt tenth
*ordnum     sitold      word  =sitmidlate
ordnum_3m   s^lt_ third
ordnum_3f   s^lt_t      third
*fracnum    sitmidlate  word
fracnum_2   fqh.  half
fracnum_3   t_lt_ third
fracnum_4   rbc   fourth
fracnum_5   h_ms  fivth
fracnum_6   sdt_  sixth
fracnum_7   sbc   seventh
fracnum_8   t_mn  eightth
fracnum_9   tsc   nineth
fracnum_10  cs^r  tenth
*fracnum    sitold      word  =sitmidlate
fracnum_3   s^lt_ third
*ppro sitmidlate  word
ppro_1cs    an    I
ppro_2cs    at    you
ppro_2cd    atmy  you
ppro_2cp    atmw  you
ppro_3msn   ha    he
ppro_3fsn   ha    she
ppro_3mso   hwt   him
ppro_3fso   hyt   her
ppro_3cdn   hmy   they
ppro_3cdo   hmt   them
ppro_3mpn   hmw   they
ppro_3fpn   hn    they
ppro_3mpo   hmt   them
ppro_3fpo   hnt   them
*ppro sitold      word  =sitmidlate
ppro_2cs    ant   you
ppro_2cd    antmy you
```

```
ppro_2cp    antmw you
*dempronear sitall      word
dempronear_ms     d_n   this
dempronear_fs     d_t   this
dempronear_md     d_n   these
dempronear_mp     aln   these
dempronear_fp     alt   these
*demprofar  sitall      word
demprofar_msn     ha    that
demprofar_fsn     ha    that
demprofar_mso     hwt   that
demprofar_fso     hyt   that
demprofar_cdn     hmy   those
demprofar_cdo     hmt   those
demprofar_mpn     hmw   those
demprofar_fpn     hn    those
demprofar_mpo     hmt   those
demprofar_fpo     hnt   those
*relpro     sitold      word
relpro_ms   d_    who;which
relpro_fs   d_t   who;which
relpro_md   d_y   who;which
relpro_fd   d_ty  who;which
relpro_mp   al    who;which
relpro_fp   alt   who;which
*relpro     sitmid      word
relpro_ms   d_    who;which
relpro_fs   d_t   who;which
relpro_md   d_y   who;which
relpro_fd   d_ty  who;which
relpro_mpn  alw   who;which
relpro_mpo  aly   who;which
relpro_fp   alt   who;which
*relpro     sitlate     word
relpro_ms   d_    who;which
relpro_fs   t     who;which
relpro_md   d_y   who;which
relpro_fd   d_ty  who;which
relpro_cp   alht  who;which
relpro_cp   alt   who;which
```

*partikel	sitall	word
partikel_b01	at_r	behind;to
partikel_b01	at_ry	behind;to
partikel_s01	at_rn	behind;to
partikel_b02	abr	over;to
partikel_s02	abrn	from
partikel_b03	ad	to;in
partikel_b03	ady	to;in
partikel_b04	al	onto;against;towards
partikel_s04	aln	down from
partikel_b05	am	with
partikel_s05	amn	from (away)
partikel_b06	bcd	after (time)
partikel_s06	bcdn	after (time);then
partikel_b07	blty	except from;without
partikel_s07	bltn	except from;without
partikel_b08	byn	between
partikel_s08	bynn	between
partikel_b09	h.g	because of
partikel_s09	h.gn	because of
partikel_b10	nh.ql	except
partikel_b11	nsr	in the direction of
partikel_s11	nsrn	(coming) from
partikel_b12	qbl	before
partikel_b13	qdm	before
partikel_b13	qdmy	before
partikel_b14	s'n	at;by
partikel_s14	s'nn	from
partikel_b15	th.ty	under;below
partikel_s15	th.tn	from beneath
*conjunction	sitall	word
conjunction_01	aw	or
conjunction_02	ahn	whenever;as soon as
conjunction_03	ahnn	whenever;as soon as
conjunction_04	ahnmw	wherever (to)
conjunction_05	kcd	while;as long as
conjunction_06	cd	until
conjunction_06	cdy	until
conjunction_07	cln	because
conjunction_08	bcd	after

```
conjunction_08    bcdn  after
conjunction_09    brt_  where;to where
conjunction_09    brt_n where;to where
conjunction_10    d_t   that
conjunction_11    hm    when
conjunction_11    hmy   when
conjunction_12    h.g   how
conjunction_12    h.gn  how
conjunction_13    kd_   that
conjunction_13    kd_y  that
conjunction_13    lkd_  (so) that
conjunction_13    lkd_y (so) that
conjunction_13    kd_   that
conjunction_14    bkn   when
conjunction_15    ld_t  so that;for that;hence
conjunction_16    mcnmw when;just as;once
conjunction_17    mhnmw when;just as;once
conjunction_18    mtymw when;just as;once
conjunction_19    lqbl  because
conjunction_19    lqbly because
conjunction_20    ywm   than;at the day as
conjunction_20    bywm  than;at the day as
```

8. Programmtexte

Die Programmtexte von KALAM stellen vom Zeitaufwand her den größten Teil dieser Arbeit dar und seien daher hier auch angeführt. Im Folgenden stehen die Klassen in alphabetischer Reihenfolge mit jeweils einer kurzen Erläuterung.

KALAM ist als Java Servlet implementiert und unter

http://www.ruzicka.net:8180/kalam/servlet/kalam

frei zugänglich.

8.1 CheckWord

… erzeugt alle möglichen Formen zu einem eingegebenen Wort, sozusagen der Kern von KALAM.

```
package kalam;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class CheckWord {

    private boolean oneRoot;    // true..used for Term-Synthesis

    /**
     * Constructor
     * @param or        =true..used for Term-Synthesis
     */
    public CheckWord(boolean or) {
        oneRoot=or;
    }

    /**
     * creates all possible forms based on all possible roots in the word
     * @param word
     * @param useDict   search for root in dictionary
     * @return
     */

    public static ArrayList<Form> getForms(String word,boolean useDict) {
        word=Convert.decode(word);
```

```
// without prefix
ArrayList<String> rootList=getRoots(word,useDict);
ArrayList<Form> fo=new ArrayList<Form>();
for (String s: rootList) {
    createForms(fo,s,Rule.neutralWordMode);
}
createForms(fo,word,Rule.onlyWordMode);       // all direct word forms

//get all prefixes
ArrayList<Form> pr=new ArrayList<Form>();
createPrefix(pr);

//get all suffixes
ArrayList<Form> su=new ArrayList<Form>();
createSuffix(su);

//all fitting prefixes
for (Form pre: pr) {
    int len=pre.form.length()-3;
    String prefix=pre.form.substring(0,len);
    if (!word.startsWith(prefix)) continue;
    if (word.length()<len+1) continue;
    ArrayList<String> rootListP=getRoots(word.substring(len),useDict);
    for (String sP: rootListP) {
        createForms(fo,sP,false,pre,null);
    }
    createForms(fo,word.substring(len),true,pre,null);
}

//all fitting suffixes
for (Form suf: su) {
    int len=suf.form.length()-3;
    String suffix=suf.form.substring(3);
    if (!word.endsWith(suffix)) continue;
    if (word.length()<len+1) continue;
    ArrayList<String> rootListS=getRoots(word.substring(0,word.length()-len),useDict);
    for (String sS: rootListS) {
        createForms(fo,sS,false,null,suf);
    }
    createForms(fo,word.substring(0,word.length()-len),true,null,suf);
}

//all fitting prefix+suffix
for (Form pre: pr) {
    int lenP=pre.form.length()-3;
    String prefix=pre.form.substring(0,lenP);
    if (!word.startsWith(prefix)) continue;
    for (Form suf: su) {
        int lenS=suf.form.length()-3;
        String suffix=suf.form.substring(3);
```

```
                if (!word.endsWith(suffix)) continue;
                if (word.length()<lenP+lenS+1) continue;
                ArrayList<String>
rootListPS=getRoots(word.substring(lenP).substring(0,word.length()-lenP-lenS),useDict);
                for (String sPS: rootListPS) {
                    createForms(fo,sPS,false,pre,suf);
                }
                createForms(fo,word.substring(lenP).substring(0,word.length()-lenP-
lenS),true,pre,suf);
            }
        }
        return fo;
    }

    /**
     * list as an html-table all correct forms for this word
     * @param foli
     * @param word
     * @param useDict use the inbuilt dictionary
     * @return html-table
     */

    public String[] listCorrectForms(ArrayList<Form> foli,String word,boolean useDict,boolean
goodLuck) {
        word=Convert.decode(word);
        String                                                                    ta="<table
border='0'><tr><td><b>"+Convert.code2html(word)+"     </b>" +
                "</td></tr></table></div><div class='div_result'>" +
                "<TABLE border='1'>"+
        "\n<tr><td>" +
                ((oneRoot)?"Form":"Root") +
                "</td><td>Stem</td><td>Term</td><td>Location</td><td>Time</td>" +
                ((oneRoot)?"":"<td>Prefix</td><td>Suffix</td><td>Translation</td>") +
                "<td><font size='-1' color='909090'>Information</font></td></tr>";
        ArrayList<String[]> arr=new ArrayList<String[]>();
        Form prefix[]=new Form[foli.size()];
        Form suffix[]=new Form[foli.size()];
        int z=0;
        for (Form f: foli) {
            String pr=(f.prefix==null)?"":f.prefix.form.substring(0,f.prefix.form.length()-3);
            String su=(f.suffix==null)?"":f.suffix.form.substring(3);
            if (oneRoot || (pr+f.form+su).equals(word)) {
                String
a[]={(oneRoot)?pr+f.form+su:f.root,""+f.term.number,f.rsituation.stem,f.term.name,
                            f.rsituation.situation.location.toString().replace("abaa","aba'"),
                            f.rsituation.situation.time.name,
                            (f.term.translation==null)?
                                ((useDict&&
oneRoot)?Kalam.db.getTranslation(f.root,word,f.rsituation):""):
                                        f.term.translation,
```

```
                        ""+z};
                arr.add(a);
                prefix[z]=f.prefix;
                suffix[z]=f.suffix;
                z++;
            }
        }
        Collections.sort(arr, new Comparator<String[]>() {
        @Override
            public int compare(String[] a, String[] b) {
                int i, len=a.length, x=0;
                int startI=(oneRoot)?1:0;

                for (i=startI; i<len; i++) {
                    if (a[i]==null || b[i]==null) continue;
                    if (i==1 || i==7) {
                        int r=Integer.parseInt(a[i]);
                        int s=Integer.parseInt(b[i]);
                        x=(r<s)?-1:(r==s)?0:1;
                    }
                    else  {
                        x=(a[i].compareTo(b[i]));
                    }
                    if (x!=0) break;
                }
                return x;
            }
        });
        int l=arr.size();
        String lasta[]={"","","","","","",""};
        for (int i=0; i<l; i++) {
            String a[]=arr.get(i);
            if (a[0].equals(lasta[0]) && a[2].equals(lasta[2]) &&
                a[3].equals(lasta[3])  &&  a[4].equals(lasta[4])  &&  a[5].equals(lasta[5])  &&
a[6].equals(lasta[6])) {
                arr.remove(i);
                i--;
                l--;
            }
            else lasta=a;
        }
        String details[]=new String[l];
        for (int i=0; i<l; i++) {
            String a[]=arr.get(i);
            String ti=a[5];
            for (int j=i+1; j<l; j++) {
                String b[]=arr.get(j);
                if   (a[0].equals(b[0])   &&   a[2].equals(b[2])   &&   a[3].equals(b[3])   &&
a[4].equals(b[4])) {
                    ti+=";"+b[5];
```

```
                    i=j; }
                else break;
            }

            int ind=Integer.parseInt(a[7]);
            String tr=(a[6]!=null)?a[6]:"", pr="", su="";
            if (prefix[ind]!=null) {
                pr=prefix[ind].term.name;
                String b=prefix[ind].term.translation;
                if (b==null) b="";
                tr=b+" "+tr;
            }
            if (suffix[ind]!=null) {
                su=suffix[ind].term.name;
                String b=suffix[ind].term.translation;
                if (b==null) b="";
                tr=tr+" "+b;
            }
            if (tr==null) tr="";
            details[i]=Term.nameAnalyze(a[3]);
            ta+="\n<tr><td>"+Convert.code2html(a[0])+"</td><td>"+a[2]+"</td><td
id='"+i+"'>"+a[3]+"</td><td>"+a[4]+"</td><td>"+ti+"</td>" +
                ((oneRoot)?"":"<td>"+pr+"</td><td>"+su+"</td><td>"+tr+"</td>") +
                "<td                        id='dest"+i+"'><font                        size='-1'
color='909090'>"+details[i]+"</font></td></tr>";
        }
        if (useDict) {
            String other=Kalam.db.searchOTHER(word,l==0,goodLuck);
            if (other!=null && other.trim().length()!=0)
                ta+="</table><table><br><p>Entire Dictionary:</p><TABLE border='1'>"+other;
        }
        ta+="</TABLE>";
        String sc="";
        String ret[]={ta,sc};
        return ret;
    }

    /**
     * delivers a  list of theoretically available roots in word
     * @param word
     * @param useDict   search for root in dictionary
     * @return
     */

    private static ArrayList<String> getRoots(String word,boolean useDict) {
        if (word==null || word.length()==0) return null;
        ArrayList<String> li=new ArrayList<String>();
        int len=word.length();
        char chars[]=new char[len];
        int i, j, k, l;
```

```
        for (i=0; i<len; i++) chars[i]=word.charAt(i);
        // search for all possible combinations of 3-character roots
        for (i=0; i<len-2; i++) {
            for (j=i+1; j<len-1; j++) {
                for (k=j+1; k<len; k++) {
                    li.add(chars[i]+""+chars[j]+""+chars[k]);
                }
            }
        }
        // search for all possible combinations of 2-character roots, adding n, w, y,
gemination
        for (i=0; i<len-1; i++) {
            char a=chars[i];
            char b=chars[i+1];
            li.add("n"+a+b);
            li.add("w"+a+b);
            li.add("y"+a+b);
            li.add(a+"w"+b);
            li.add(a+"y"+b);
            li.add(a+""+b+"w");
            li.add(a+""+b+"y");
            li.add(a+""+b+""+b);
            if (i<len-2) {
                b=chars[i+2];
                li.add("n"+a+b);
                li.add("w"+a+b);
                li.add("y"+a+b);
                li.add(a+"w"+b);
                li.add(a+"y"+b);
                li.add(a+""+b+"w");
                li.add(a+""+b+"y");
                li.add(a+""+b+""+b);
            }
        }
        // search for all possible combinations of 4-character roots
        for (i=0; i<len-3; i++) {
                for (j=i+1; j<len-2; j++) {
                        for (k=j+1; k<len-1; k++) {
                                for (l=k+1; l<len; l++) {
                                        li.add(chars[i]+""+chars[j]+""+chars[k]+""+chars[l]);
                                }
                        }
                }
        }

        if (useDict) {
            ArrayList<String> li1=new ArrayList<String>();
            for (String ro: li) {
                if (Kalam.db.searchVN(ro)) li1.add(ro);
            }
```

```
            li=li1;
        }

        return li;
    }

    /**
     * creates all Strings possible with all Rules based on the root
     * @param fo
     * @param root
     */
    private static void createForms(ArrayList<Form> fo,String root,int wordMode) {
        // for all Rules
        for (Rule r: Rule.allRules) {
                r.produce(fo, root,wordMode);
        }
    }

    /**
     * creates all Strings possible with all Rules based on the root, including the specified
prefix and/or suffix
     * @param fo
     * @param root
     * @param word
     * @param pre
     * @param suf
     */
    private static void createForms(ArrayList<Form> fo,String root,boolean word,Form pre,Form
suf) {
        // for all Rules
        for (Rule r: Rule.allRules) {
            r.produce(fo, root,word,pre,suf);
        }
    }


    /**
     * creates all prefixes
     * @param fo
     */
    private static void createPrefix(ArrayList<Form> fo) {
        // for all Rules
        for (Rule r: Rule.allRules) {
                r.produce(fo, "XYZ",true,false);
        }
    }

    /**
     * creates all suffixes
     * @param fo
```

```
     */
    private static void createSuffix(ArrayList<Form> fo) {
        // for all Rules
        for (Rule r: Rule.allRules) {
                r.produce(fo, "XYZ",false,true);
        }
    }

    /**
     * get all forms for root
     * @param root
     * @return
     */
    public static ArrayList<Form> getFormsForRoot(String root) {
        root=Convert.decode(root);
        // alle ohne prefix
        ArrayList<Form> fo=new ArrayList<Form>();
        createForms(fo,root,Rule.noWordMode);
        return fo;
    }


    /**
     * search in translation
     * @param word
     * @return
     */

    public String[] searchEnglish(String word) {
        String                    ta="<table                    border='0'><tr><td><b>English:
"+word+"     </b>" +
                "</td></tr></table></div><div class='div_result'>" +
                "<TABLE border='1'>"+
        "\n<tr><td>Root</td><td>Type</td><td>Translation</td><td>Term</td><td>Form</td></tr>";
        String other=Kalam.db.searchENGLISH(word);
        if (other!=null && other.trim().length()!=0)
            ta+=other;
        ta+="</TABLE>";
        String sc="";
        String ret[]={ta,null,sc};
        return ret;
    }
}
```

8.2 Convert

… konvertiert die verschiedenen Darstellungsformen (UTF-8/Unicode, ASCII-Äquivalente) der Sonderzeichen der altsüdarabischen Schrift.

```
package kalam;

public class Convert {
    public Convert() {
        super();
    }

    /**
     * ASCII-equivalent to Unicode
     * @param txt
     * @return
     */
    public static String decode(String txt) {
        return txt
                 .replace("a","\u02BE")
                .replace("c","\u1D9C")
                .replace("d_","\u1E0F")
                .replace("d.","\u1E0D")
                .replace("g.","\u0121")
                .replace("h.","\u1E25")
                .replace("h_","\u1E2B")
                .replace("s.","\u1E63")
                .replace("s^","\u0161")
                .replace("s'","\u015B")
                .replace("t.","\u1E6D")
                .replace("t_","\u1E6F")
                .replace("z.","\u1E93")
        ;
    }

    /**
     * Unicode to html
     * @param t
     * @return
     */
    public static String code2html(String t) {
        return(t
        .replace("\u02BE","&#x02BE;")
        .replace("\u1D9C","&#x1D9C;")
        .replace("\u1E0F","&#x1E0E;")
        .replace("\u1E0D","&#x1E0D;")
        .replace("\u0121","&#x0121;")
        .replace("\u1E25","&#x1E25;")
        .replace("\u1E2B","&#x1E2B;")
        .replace("\u1E63","&#x1E67;")
        .replace("\u0161","&#x0161;")
        .replace("\u015B","&#x015B;")
        .replace("\u1E6D","&#x1E6D;")
        .replace("\u1E6F","&#x1E6F;")
        .replace("\u1E93","&#x1E93;")
```

```
        );
    }

    /**
     * html zu ASCII-equivalent
     * @param t
     * @return
     */
    public static String html2code(String t) {
        return(t
        .replace("&#702;", "a")
        .replace("&#7580;","c")
        .replace("&#7695;","d_")
        .replace("&#7693;","d.")
        .replace("&#289;", "g.")
        .replace("&#7717;","h.")
        .replace("&#7723;","h_")
        .replace("&#7779;","s.")
        .replace("&#7789;","t.")
        .replace("&#7791;","t_")
        .replace("&#7827;","z.")

        );
    }


    /**
     * Unicode to Database-SQL
     * @param db
     * @param t
     * @return
     */
    public static String code2sql(DataBase db,String t) {
        return(
            (db.type==DataBase.ORACLE)?
                "unistr('"+t
                    .replace("\u02BE","\\02BE")
                    .replace("\u1D9C","\\1D9C")
                    .replace("\u1E0F","\\1E0F")
                    .replace("\u1E0D","\\1E0D")
                    .replace("\u0121","\\0121")
                    .replace("\u1E25","\\1E25")
                    .replace("\u1E2B","\\1E2B")
                    .replace("\u1E63","\\1E63")
                    .replace("\u0161","\\0161")
                    .replace("\u015B","\\015B")
                    .replace("\u1E6D","\\1E6D")
                    .replace("\u1E6F","\\1E6F")
                    .replace("\u1E93","\\1E93")
                    +"')":
```

```
            "'"+t
            .replace("\u02BE","@02BE")
            .replace("\u1D9C","@1D9C")
            .replace("\u1E0F","@1E0F")
            .replace("\u1E0D","@1E0D")
            .replace("\u0121","@0121")
            .replace("\u1E25","@1E25")
            .replace("\u1E2B","@1E2B")
            .replace("\u1E63","@1E63")
            .replace("\u0161","@0161")
            .replace("\u015B","@015B")
            .replace("\u1E6D","@1E6D")
            .replace("\u1E6F","@1E6F")
            .replace("\u1E93","@1E93")
                +"'"
        );
}

/**
 * Database-Unicode to html
 * @param db
 * @param t
 * @return
 */
public static String sql2html(DataBase db,String t) {
    return(
        (db.type==DataBase.ORACLE)?
            t
                .replace("\u02BE","&#x02BE;")
                .replace("\u1D9C","&#x1D9C;")
                .replace("\u1E0F","&#x1E0F;")
                .replace("\u1E0D","&#x1E0D;")
                .replace("\u0121","&#x0121;")
                .replace("\u1E25","&#x1E25;")
                .replace("\u1E2B","&#x1E2B;")
                .replace("\u1E63","&#x1E63;")
                .replace("\u0161","&#x0161;")
                .replace("\u015B","&#x015B;")
                .replace("\u1E6D","&#x1E6D;")
                .replace("\u1E6F","&#x1E6F;")
                .replace("\u1E93","&#x1E93;")
        :
            t
                .replace("@02BE","&#x02BE;")
                .replace("@1D9C","&#x1D9C;")
                .replace("@1E0F","&#x1E0F;")
                .replace("@1E0D","&#x1E0D;")
                .replace("@0121","&#x0121;")
                .replace("@1E25","&#x1E25;")
                .replace("@1E2B","&#x1E2B;")
```

```
                    .replace("@1E63","&#x1E63;")
                    .replace("@0161","&#x0161;")
                    .replace("@015B","&#x015B;")
                    .replace("@1E6D","&#x1E6D;")
                    .replace("@1E6F","&#x1E6F;")
                    .replace("@1E93","&#x1E93;")
            );
    }


    static public String chars[]={
        "&#x02BE;",
        "&#x1D9C;",
        "&#x1E0F;",
        "&#x1E0D;",
        "&#x0121;",
        "&#x1E25;",
        "&#x1E2B;",
        "&#x1E63;",
        "&#x0161;",
        "&#x015B;",
        "&#x1E6D;",
        "&#x1E6F;",
        "&#x1E93;"};
}
```

8.3 DataBase

… stellt das Datenbankobjekt dar, das die Verbindung zum Wörterbuch herstellt. Hierin sind sowohl die Varianten

1. MS-Windows
2. Linux

als auch die Alternativen

1. Oracle
2. MySQL

abgebildet.

```
package kalam;
```

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;

import java.io.IOException;

import java.io.InputStream;
import java.io.InputStreamReader;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import java.util.ArrayList;

public class DataBase {

    public static String connectString="jdbc:oracle:thin:@10.1.1.52:1521:his3d11g";
//    public static String connectString="jdbc:mysql://10.1.1.45:3306/kalam";

    private String name="kalam", passwort="xxx";

    private String dictDir1="c:\\texte\\studium\\master\\";
    private String dictDir2="/etc/kalam/";
    private final static String dictFile="sab-dictionary-import.dat";

    public static final short ORACLE=1;
    public static final short MYSQL=2;
    public short type=ORACLE;

    private Connection conn=null;

    private String forms []= {
         "sample",   "feminin",   "plural",   "collective",   "infinitive",   "femininplural",
"imperfect"
    };


    /**
     * Constructor for the DataBase Connection Object
     */

    public DataBase() {
        if (connectString.toLowerCase().contains("mysql")) type=MYSQL;
        try {
```

```
            DriverManager.registerDriver((type==ORACLE)?new oracle.jdbc.driver.OracleDriver():
                                        new com.mysql.jdbc.Driver());
        } catch (SQLException e) {
            ErrorLog.writelogfile("Kalam: Error - could not register a driver\n"+e);
            System.err.println(e);
            return;
        }
        try {
            conn = DriverManager.getConnection(connectString,name,passwort);
            conn.setAutoCommit(false);
        }
        catch (SQLException e) {
            ErrorLog.writelogfile("Kalam:  Error  -  could  not  get  a  connection  to
server='"+connectString+"',usr="+name+"\n"+
                                    e);
            System.err.println(e);
            return;
        }
        ErrorLog.writelogfile("Kalam:          Databse          started          successfully:
'"+connectString+"',user="+name);
    }

    /**
     * tests, whether the connection is still valid
     * @return
     */
    public boolean isValid() {
        if (conn==null) return false;
        int z=0;
        try {
            Statement s=conn.createStatement();
            ResultSet r=s.executeQuery("SELECT id from sabaic where id=1");
            if (r.next()) z++;
            r.close();
            s.close();
        } catch (Exception e) {
        }
        return z!=0;
    }


    /**
     * startup routine for filling the database
     */
    public void fillDict() {
        String sql=null, vals=null;
        int zeile=0;
        try {
            BufferedReader br=null;
            try {
```

```
                br=new                BufferedReader(new                InputStreamReader(new
FileInputStream(dictDir1+dictFile), "UTF-8"));
            }
            catch (Exception e) { }
            if (br==null) {
                br=new                BufferedReader(new                InputStreamReader(new
FileInputStream(dictDir2+dictFile), "UTF-8"));
            }

            Statement s=conn.createStatement();

            for (int f=0; f<forms.length; f++)
                s.execute(sql="delete from sabaic_"+forms[f]);
            s.execute(sql="delete from sabaic");
            conn.commit();

            String li=null;
            ArrayList<String> arr=new ArrayList<String>();
            while ((li=br.readLine())!=null) {
                zeile++;
                if (zeile%100==0) ErrorLog.writelogfile(""+zeile);
                if (li.charAt(0)=='*') {
                    if (arr.size()!=0) {
                        sql="insert into sabaic(";
                        vals="values(";
                        int i=0;
                        int id=0;
                        ArrayList<String> ins=new ArrayList<String>();
                        for (String st: arr) {
                            int p=st.indexOf(' ');
                            String tag=st.substring(0,p);
                            String val=st.substring(p+1).trim();
                            if (val.charAt(0)=='*') val=val.substring(1);
                            if (tag.equals("reference") ||
                                tag.equals("sample") ||
                                tag.equals("feminin") ||
                                tag.equals("plural") ||
                                tag.equals("collective") ||
                                tag.equals("infinitive") ||
                                tag.equals("femininplural") ||
                                tag.equals("imperfect")) {
                                String ar[]=val.split(",");
                                for (int a=0; a<ar.length; a++)
                                    ins.add("insert            into            sabaic_"+tag+"
values("+id+","+a+","+Convert.code2sql(this,ar[a].trim())+")");
                                continue;
                            }
                            String ch=(i>0)?",":"";
                            sql+=ch+((tag.equals("dual"))?"dualform":tag);
```

```
                            if      (tag.equals("formnr")      ||      tag.equals("id")      ||
tag.equals("dictpage")) {
                                vals+=ch+val;
                                if (tag.equals("id")) id=Integer.parseInt(val);
                            }
                            else vals+=ch+Convert.code2sql(this,val);
                            i++;
                        }
                        s.execute(sql=sql+") "+vals+")");
                        for (String ss:ins)
                            s.execute(sql=ss);
                    }
                    arr.clear();
                }
                else arr.add(li);
            }
            br.close();
            s.close();
            conn.commit();
        } catch (FileNotFoundException e) {
            ErrorLog.writelogfile(zeile+" "+e);
            return;
        } catch (IOException e) {
            ErrorLog.writelogfile(zeile+" "+e);
            return;
        } catch (SQLException e) {
            ErrorLog.writelogfile(zeile+":"+sql+":\n"+e);
            return;
        }
    }

    /**
     * search for root in the database (type verb or noun)
     * @param root
     * @return
     */
    public boolean searchVN(String root) {
        String sql=null;
        try {
            Statement s=conn.createStatement();
            ResultSet r=s.executeQuery(sql="select id from sabaic where type in ('v','n') and
root=" +
                Convert.code2sql(this,root));
            boolean exists=false;
            if (r.next()) exists=true;
            r.close();
            s.close();
            return exists;
        } catch (SQLException e) {
            ErrorLog.writelogfile(sql+":\n"+e);
```

```
            return false;
        }
    }

    /**
     * search for root in the database
     * @param root
     * @param withVN    including verb and noun types
     * @param goodLuck  fulltext search
     * @return
     */

    public String searchOTHER(String root,boolean withVN,boolean goodLuck) {
        String sql=null;
        String ta="";
        try {
            Statement s=conn.createStatement();
            String rootU=Convert.code2sql(this,root);
            if (goodLuck) withVN=true;
            for (int f=0; f<forms.length; f++) {
                ResultSet r=s.executeQuery(sql=
    "select   s.root,s.type,a.word,s.translation1,s.translation2,s.translation3,s.translation4
from sabaic s," +
                    "sabaic_"+forms[f]+" a " +
                    "where "+((withVN)?"":"(type not in ('v','n')) and")+
                        " (select count(*) from sabaic_reference sr where sr.root_id=s.id)=0
and " +
                    "a.root_id=s.id and (a.word is not null) and (" +
                    ((goodLuck)?"instr(replace(a.word,'-',''),"+rootU+")>0 ":
                    "replace(a.word,'-','')="+rootU+" ") +
                        "or s.root="+rootU+") order by s.id");
                while (r.next()) {
                    String ro=r.getString(1);
                    String ty=r.getString(2);
                    String wo=r.getString(3);
                    String tr1=r.getString(4);
                    String tr2=r.getString(5);
                    String tr3=r.getString(6);
                    String tr4=r.getString(7);
                    if (tr1==null) tr1="";
                    if (tr2!=null) tr1+="; "+tr2;
                    if (tr3!=null) tr1+="; "+tr3;
                    if (tr4!=null) tr1+="; "+tr4;
                    ty=v2verb(ty);
                    ta+="\n<tr><td>"+Convert.sql2html(this,ro)+"</td><td></td><td>"+
                        ((!wo.equals(root))?Convert.sql2html(this,wo)+":
":"")+ty+";"+forms[f]+"</td><td></td><td></td>" +
                        "<td></td><td></td><td>"+tr1+"</td></tr>";
                }
                r.close();
```

```
            }
            s.close();
            return ta;
        } catch (SQLException e) {
            ErrorLog.writelogfile(sql+":\n"+e);
            return null;
        }
    }

    /**
     * reverse search
     * @param word
     * @return
     */
    public String searchENGLISH(String word) {
        String sql=null;
        String ta="";
        word=word.toLowerCase();
        try {
            Statement s=conn.createStatement();
            Statement s2=conn.createStatement();
            ResultSet r=s.executeQuery(sql=
            "select id,root,type,translation1,translation2,translation3,translation4," +
                "(select count(*) from sabaic_reference sr where sr.root_id=id) from sabaic s
where " +
                        "((translation1   is   not   null   and   instr(lower(translation1),'"
+word+"')>0) or " +
                        "(translation2   is   not   null   and   instr(lower(translation2),'"
+word+"')>0) or " +
                        "(translation3   is   not   null   and   instr(lower(translation3),'"
+word+"')>0) or " +
                        "(translation4   is   not   null   and   instr(lower(translation4),'"
+word+"')>0)) " +
                    "and (select count(*) from sabaic_reference sr where sr.root_id=id)=0 " +
                    "order by root,id");
            String lastRoot="";
            while (r.next()) {
                int id=r.getInt(1);
                String root=r.getString(2);
                String                              root0=(!lastRoot.equals(root))?root:"<font
color='A0A0A0'>"+root+"</font>";
                lastRoot=root;
                String ty0=r.getString(3);
                String tr1=r.getString(4);
                String tr2=r.getString(5);
                String tr3=r.getString(6);
                String tr4=r.getString(7);
                if (tr1==null) tr1="";
                if (tr2!=null) tr1+="; "+tr2;
                if (tr3!=null) tr1+="; "+tr3;
```

```
                if (tr4!=null) tr1+="; "+tr4;
                String tr0=tr1;
                ty0=v2verb(ty0);
                String ty=ty0;
                for (int f=0; f<forms.length; f++) {
                    ResultSet r2=s2.executeQuery(sql=
                        "select word from sabaic_"+forms[f]+" where root_id=" +id+" order by
id");
                    String text="", trenn="";
                    while (r2.next()) {
                        text+=trenn+" "+r2.getString(1);
                        trenn=";";
                    }
                    r2.close();
                    if (text.length()==0) continue;

ta+="\n<tr><td>"+Convert.sql2html(this,root0)+"</td><td>"+ty+"</td><td>"+tr1+"</td><td>"+
                        forms[f]+"</td><td>"+Convert.sql2html(this,text)+"</td></tr>";
                    ty="<font color='A0A0A0'>"+ty0+"</font>";
                    tr1="<font color='A0A0A0'>"+tr0+"</font>";
                }
            }
            r.close();
            s.close();
            s2.close();
            return ta;
        } catch (SQLException e) {
            ErrorLog.writelogfile(sql+":\n"+e);
            return null;
        }
    }

    /* typelist */

    String types [][]= {
        { "v",      "verb" },
        { "n",      "noun" },
        { "r",      "relative pronoun" },
        { "prep",   "preposition" },
        { "pp",     "passive participle" },
        { "a",      "adjective" },
        { "p",      "particle" },
        { "c",      "conjunction" },
        { "ad",      "adverb" },
        { "m",      "?" }
    };

    /**
     * converts the description of the type into a long-form
     * @param v
```

```
     * @return
     */
    private String v2verb(String v) {
        int i;
        for (i=0; i<types.length; i++)
            if (v.equals(types[i][0])) return types[i][1];
        return "";
    }

    /**
     *
     * @param root
     * @param word
     * @param rs
     * @return
     */
    public String getTranslation(String root,String word,RuleSituation rs) {
        String withVN=null;
        boolean found=false;
        Term tt=null;
        for (Term t: rs.termList) {
            char na=t.name.charAt(0);
            if (t.name.endsWith("stem")) {
                if (na=='v' || na=='n') withVN=t.name.substring(0,1);
                tt=t;
                found=true;
                break;
            }
        }
        if (!found) {
            for (Term t: rs.termList) {
                char na=t.name.charAt(0);
                if (na=='v' || na=='n') withVN=t.name.substring(0,1);
                found=true;
                break;
            }
        }
        if (found) word=(tt!=null)?tt.produce(root):rs.produce(root,true);
        String ta="";
        boolean first=true;
        String arr[]=word.split(",");
        String sql=null;
        String rootU=Convert.code2sql(this,root);
        ArrayList<Integer> arrID=new ArrayList<Integer>();
        try {
            Statement s=conn.createStatement();
            for (String word3ms: arr) {
                String word3msU=Convert.code2sql(this,word3ms);
                for (int f=0; f<forms.length; f++) {
                    ResultSet r=s.executeQuery(sql=
```

```
        "select
s.root,s.type,a.word,s.translation1,s.translation2,s.translation3,s.translation4,s.id     from
sabaic s," +
                        "sabaic_"+forms[f]+" a " +
                    "where "+((withVN!=null)?"(type='"+withVN+"') and":"")+
                            "    (select    count(*)    from    sabaic_reference    sr    where
sr.root_id=s.id)=0 and " +
                        "a.root_id=s.id and (a.word is not null) and (" +
                        "instr(replace(word,'-',''),"+word3msU+")>0 "+
                            "and s.root="+rootU+") order by s.id");
                    while (r.next()) {
                        String tr1=r.getString(4);
                        String tr2=r.getString(5);
                        String tr3=r.getString(6);
                        String tr4=r.getString(7);
                        int id=r.getInt(8);
                        if (arrID.contains(id)) continue;
                        arrID.add(id);
                        if (tr1==null) tr1="";
                        if (tr2!=null) tr1+="; "+tr2;
                        if (tr3!=null) tr1+="; "+tr3;
                        if (tr4!=null) tr1+="; "+tr4;
                        ta+=((first)?"":",")+tr1;
                        first=false;
                    }
                    r.close();
                }
            }
            s.close();
            return (first)?"[not in dict.]":ta;
        } catch (SQLException e) {
            ErrorLog.writelogfile(sql+":\n"+e);
            return null;
        }
    }
}
```

8.4 ErrorLog

… schreibt die Log-Datei.

```
package kalam;

import java.io.FileWriter;


public class ErrorLog {
    public ErrorLog() {
        super();
    }
```

```
    static FileWriter logfile=null;

    /**
     * write error message to logfile
     * @param text
     */
    static public void writelogfile(String text) {

        System.err.println(text);
        System.out.println(text);
        if (logfile == null) {
            boolean error = false;

            try { /*append*/
                logfile = new FileWriter(SabaicInit.confDir1 + "kalam.log",true);
            } catch (Exception i) {
                error = true;
            }
            if (logfile == null)
                try { /*append*/
                    logfile = new FileWriter(SabaicInit.confDir2 + "kalam.log",true);
                } catch (Exception i) {
                    error = true;
                }
            if (error)
                return;
        }
        try {
            String sdat = Kalam.simpledate();
            logfile.write(sdat + ":" + text + "\r\n");
            logfile.flush();
        } catch (Exception i) {
            return;
        }
    }
}
```

8.5 Form

… implementiert das Form-Objekt.

```
package kalam;

public class Form {
    public String root;
    public RuleSituation rsituation;
    public Term term;
    public String form;
    public Form prefix=null;
```

```
    public Form suffix=null;

    /**
     * Constructor Standard
     * @param s
     * @param t
     * @param r
     * @param f
     */
    public Form(RuleSituation s,Term t,String r,String f) {
        rsituation=s;
       term=t;
       root=r;
       form=f;
    }

    /**
     * Constructor with prefix and/or suffix
     * @param s
     * @param t
     * @param r
     * @param f
     * @param pr
     * @param su
     */
    public Form(RuleSituation s,Term t,String r,String f,Form pr,Form su) {
        rsituation=s;
        term=t;
        root=r;
        form=f;
        prefix=pr;
        suffix=su;
    }
}
```

8.6 Kalam

… die Haupt- und Startklasse mit der Benutzeroberfläche.

```
package kalam;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

import java.util.Calendar;
import java.util.Enumeration;

import java.util.GregorianCalendar;
```

```
import java.util.TimeZone;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Servlet implementation class Kalam
 */
public class Kalam extends HttpServlet {

    private static final boolean NODB=false;
    private static final String version="0.77a";
    private static final long serialVersionUID = 1L;
    private static String header="<html>"+
"<head>"+
"<META NAME=\"Author\" CONTENT=\"Ronald Ruzicka\">"+
"<META NAME=\"GENERATOR\" CONTENT=\"Ronald Ruzicka\">"+
"<META NAME=\"keywords\" CONTENT=\"kalam, sabaic, arabistic\">"+
"<META NAME=\"kalam, sabaic, arabistic\">"+
"<title>KALAM by Ronald Ruzicka</title>"
+"</head>"
+ "<body>";

    public static String divstyle=
    "<style type=\"text/css\">\n" +
    ".div_title {position:absolute; left:0%; width:99%; top:0%; height:20%;padding:2px ;
background-color:#000000;border:1px #00e4e4 }\n" +
    ".div_body {position:absolute; left:0%; width:99%; top:20%; height:30%; background-
color:#fff0d0; padding:2px ;overflow:auto;}\n" +
    ".div_result {position:absolute; left:0%; width:99%; top:50%; height:49%; background-
color:#fff0d0; padding:2px ;overflow:auto;}\n" +
    "</style>\n";

    private static String title0=
        "<div class='div_title'><center><table border='0'><tr><td valign='center'>" +
        "<face=\"Arial,Helvetica\" color='FFC000'><h1>Kalam - Sabaic Word Analysis -
</h1></font></td><td valign='center'>" +
                    "<img src='";
    private static String title1=
        "kalam.png' alt='KLM'></td></tr></table>"+
                    "<br><font face=\"Arial,Helvetica\" color='FFC000'><h3>Version "+version+"
by Ronald Ruzicka (c) 2015</h3></font>" +
                "</font></center></div>";

    private static String jscri="\n<script type=\"text/javascript\">\n" +
    "<!--\n" +
```

```
"  function add(ch) {\n" +
"    var ziel=document.getElementsByName('word')[0];\n" +
"    ziel.value+= ch; ziel.focus();\n" +
"  }\n" +
"\n" +
"function setOperation(op) {\n" +
"  var o;o=document.getElementsByName('op')[0];\n"+
"  o.value=op;\n" +
"}\n" +
"//-->\n" +
"</script>\n";

private static String footer="</body></html>";
public static DataBase db=null;

/**
 * @see HttpServlet#HttpServlet()
 */
public Kalam() {
    super();
}

/**
 * standard initialization including database connection
 * @param config
 * @throws ServletException
 */
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    String p=getInitParameter("conffile");
    if (p!=null) {
        int h=p.lastIndexOf('\\');
        if (h<0) h=p.lastIndexOf('/');
        if (h>0) p=p.substring(0,h+1);
        SabaicInit.confDir1=p;
    }
    ErrorLog.writelogfile("KALAM init() conffile="+p);
    p=getInitParameter("connection");
    ErrorLog.writelogfile("KALAM init() connection="+p);
    if (p!=null) DataBase.connectString=p;
    ErrorLog.writelogfile("KALAM version="+version);
    initKalam();
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
Exception, IOException {
```

```
            // TODO Auto-generated method stub
        if (db!=null && !NODB) {
            if (!db.isValid()) {
                db=new DataBase();
                if (db!=null && !db.isValid()) db=null;
            }
        }
        String servlet=request.getRequestURL().toString();
        String word0 = request.getParameter("word"), word=word0;
        if (word==null) {
            word=word.replace("'","").replace("\"","").replace("@","");
            word=Convert.html2code(word);
        }
        String op= request.getParameter("op");
        String dictionary= request.getParameter("dictionary");
        boolean useDict=(db!=null && dictionary!=null && dictionary.equals("yes"));
        String goodLucks= request.getParameter("goodluck");
        boolean goodLuck=(db!=null && goodLucks!=null && goodLucks.equals("yes"));
        String png_path=(servlet.contains("servlet"))?"../":"";
        String ziel= request.getParameter("ziel");
        boolean sabaic=(ziel==null || ziel.equals("sabaic"));


        ErrorLog.writelogfile("<<<<KALAM Pars: "+simpledate());
        Enumeration<String> en=request.getParameterNames();
        while (en.hasMoreElements()) {
            String pn=(String)en.nextElement();
            String pv=request.getParameter(pn);
            if (pv==null || pv.length()==0) pv="";
            ErrorLog.writelogfile(pn+"="+pv); }
        ErrorLog.writelogfile(">>>>");

        PrintWriter fiwr=response.getWriter();
        fiwr.println(header+divstyle+title0+png_path+title1+jscri);

        fiwr.println("<div           class='div_body'><br><center><FORM           METHOD=POST
ACTION=\""+servlet+"\">" +
                    "<table border='0'><tr><td>" +
                    "<INPUT TYPE=HIDDEN NAME='version' VALUE='"+version+"'>" +
                    "<INPUT TYPE=HIDDEN NAME='op' VALUE='search'>" +
                   "<INPUT    TYPE=TEXT    NAME='word'    VALUE='"+((word0==null)?"":word0)+"'
"</td><td>   </td><td>" +
                    "<INPUT                  type=\"submit\"                  value=\"Search\"
style=\"width:100px;\">  " +
                    "<INPUT               type=\"submit\"               value=\"Synthesize\"
onClick='setOperation(\"synth\")' style=\"width:100px;\">" +
                     "</td></tr>" +
                    "<tr><td><table border='0'>");
        int le2, le=((le2=Convert.chars.length)+1)/2, j=0;
        for (int z=0; z<2; z++) {
```

```
            fiwr.println("<tr>");
            for (int s=0; s<le && j<le2; s++) {
                fiwr.println("<td><input     type='button'     style='height:22px;width:14px;'
value='"+Convert.chars[j]+"' " +
                    "onclick='add(\""+Convert.chars[j]+"\")'></td>");
                j++;
            }
            fiwr.println("</tr>");
        }
        fiwr.println("</table></td><td>   </td><td><table border=0><tr><td>" +
                     "enter      <INPUT      TYPE=RADIO      NAME='ziel'      VALUE='sabaic'
CHECKED"+((db==null)?" DISABLED":"")+">Sabaic " +
                     "<INPUT    TYPE=RADIO    NAME='ziel'    VALUE='english'    "+((db==null)?"
DISABLED":"")+">English word </td></tr>"+
                     "<tr><td><INPUT TYPE=CHECKBOX "+((useDict && db!=null)?" CHECKED":"")+
                     ((db==null)?"   DISABLED":"")+"   name='dictionary'   value='yes'>   use
dictionary</td></tr>" +
                     "<tr><td><INPUT  TYPE=CHECKBOX  "+((useDict  &&  goodLuck  &&  db!=null)?"
CHECKED":"")+
                     ((db==null)?"   DISABLED":"")+"   name='goodluck'   value='yes'>   entire
dictionary</td></tr></table></td></tr>" +
                    "</table></form></center><br>");

        String afterDiv="<div class='div_result'></div>";
        if (op!=null) {
            if (op.equals("search")) {
                if (sabaic) {
                    if (word!=null && word.length()>0) {
                        ArrayList<Form> foli=CheckWord.getForms(word,useDict);
                        CheckWord cw=new CheckWord(false);
                        String test[]=cw.listCorrectForms(foli,word,useDict,goodLuck);
                        fiwr.println(test[0]);
                        afterDiv=test[1];
                    }
                }
                else {
                    CheckWord cw=new CheckWord(false);
                    String test[]=cw.searchEnglish(word);
                    fiwr.println(test[0]);
                    afterDiv=test[1];
                }
            }
            else if (op.equals("synth")) {
                if (word!=null && word.length()>1) {
                    if (word.length()==2) word=word+word.substring(1,2);
                    ArrayList<Form> foli=CheckWord.getFormsForRoot(word);
                    CheckWord cw=new CheckWord(true);
                    String test[]=cw.listCorrectForms(foli,word,useDict,goodLuck);
                    fiwr.println(test[0]);
                    afterDiv=test[1];
```

```
                }
            }
            else if (op.equals("fill")) {
                db.fillDict();
            }
        }
        fiwr.println("</div>"+afterDiv+footer);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }


    /**
     * init method
     */
    private void initKalam() {
        if (NODB) db=null;
        else {
            db=new DataBase();
            if (db!=null && !db.isValid()) db=null;
        }
        SabaicInit.init();
    }

    /**
      simpledate   current date and time
     */

    public static String simpledate() {
        return(simpledate(0)); }

    /**
     * current date and time moved by an amount of tage days
     * @param tage
     * @return
     */

    static public String simpledate(int tage) {
        java.util.Date da = new java.util.Date();
        TimeZone tz=TimeZone.getTimeZone("Europe/Vienna");
        GregorianCalendar kal = new GregorianCalendar(tz);
        kal.setTime(da);
        if (tage!=0)
```

```
            kal.add(GregorianCalendar.DAY_OF_MONTH,tage);
        String sdat =
            "" + ((kal.get(Calendar.YEAR)) *10000 + (kal.get(Calendar.MONTH) + 1) *
            100 + kal.get(Calendar.DATE));
        String suhr =
            "" + (kal.get(Calendar.HOUR_OF_DAY) * 10000 + kal.get(Calendar.MINUTE) *
                  100 + kal.get(Calendar.SECOND));
        int h = sdat.length();
        for (int i = h; i < 6; i++)
            sdat = "0" + sdat;
        h = suhr.length();
        for (int i = h; i < 6; i++)
            suhr = "0" + suhr;
        return (sdat + "/" + suhr);
    }
}
```

8.7 Locations

… implementiert die Locations. Hier sollten Erweiterungen z.B. für Dialekte eingefügt werden.

```
package kalam;

public enum Locations {
       Sabaa
}
```

8.8 Rule

… implementiert das Rule-Objekt. Hierin sind Hilfsroutinen zum Generieren der Formen enthalten.

```
package kalam;

import java.util.ArrayList;

/**
 * one rule, like "how to build up the imperfect", separation for different situations
 * @author rr
 *
 */

public class Rule {
    public static ArrayList<Rule> allRules=new ArrayList<Rule>();

    public static final int onlyWordMode=1;
    public static final int noWordMode=0;
```

```
    public static final int neutralWordMode=2;

    public String name;
    public ArrayList<RuleSituation> ruleList;       // rule per Situation

    /**
     * create a new rule with name n
     * @param n
     */
    public Rule(String n) {
        name=n;
        ruleList=new ArrayList<RuleSituation>();
    }

    /**
     * add a time-dependend rule
     * @param rs
     * @return false .. a RuleSituation already exists for this situation
     */
    public boolean add(RuleSituation rs) {
        for (RuleSituation rules: ruleList) {
            if (rules.situation.equals(rs.situation)) {
                ErrorLog.writelogfile("Duplicated entry in Rule: situation="+rs.situation);
                return false;
            }
        }
        ruleList.add(rs);
        return true;
    }

    /**
     * create a list of possible forms with this rule and all situations
     * @param abc
     */

    public void produce(ArrayList<Form> li,String abc,int wordMode) {
        boolean onlyWord=(wordMode==onlyWordMode);
        boolean noWord=(wordMode==noWordMode);
        for (RuleSituation rules: ruleList) {
            if (onlyWord && !rules.word) continue;
            if (noWord && rules.word) continue;
            if (!rules.prefix && !rules.suffix) rules.produce(li,abc,false);
        }
    }

    /**
     * create a list of possible forms with this rule and all situations, including prefix
and/or postfix
     * @param li
     * @param abc
```

```
     * @param onlyWord
     * @param pre
     * @param suf
     */
    public void produce(ArrayList<Form> li,String abc,boolean onlyWord,Form pre,Form suf) {
        for (RuleSituation rules: ruleList) {
            if (onlyWord && !rules.word) continue;
            if (!rules.prefix && !rules.suffix) rules.produce(li,abc,pre,suf,false);
        }
    }


    /**
     * produce all Forms, which are either prefix or suffix (accrpding to the parameters)
     * @param li
     * @param abc
     * @param prefix
     * @param suffix
     */

    public void produce(ArrayList<Form> li,String abc,boolean prefix,boolean suffix) {
            for (RuleSituation rules: ruleList) {
                    if    ((prefix    &&    rules.prefix)    ||    (suffix    &&    rules.suffix))
rules.produce(li,abc,false);
            }
    }

    /**
     * searchs for a rule with the specified name
     * @param name
     * @return  null, if no such rule exists
     */

    public static Rule search(String name) {
        for (Rule r: allRules) {
            if (r.name.equals(name)) return r;
        }
        return null;
    }
}
```

8.9 RuleSituation

… implementiert das Objekt RuleSituation, beschreibt eine Rule für eine spezielle Situation.

```
package kalam;

import java.util.ArrayList;
```

```
/**
 * a list of terms for one rule in a certain situation
 * e.g. Perfect conjugation in Saba in Periode A
 * @author rr
 *
 */

public class RuleSituation {
    public ArrayList<Term> termList;
    public Situation situation;
    public String stem;
    public boolean prefix=false;
    public boolean suffix=false;
    public boolean word=false;  // in this case the Term contains only one word

    /**
     * Constructor
     * @param s
     * @param st type of Rule: stem, special word, prefix, suffix
     */
    public RuleSituation(Situation s,String st) {
        situation=s;
        if (st.equals("prefix")) prefix=true;
        else if (st.equals("suffix")) suffix=true;
        else if (st.equals("word")) {
            word=true;
            stem="";
        }
        else stem=st;
        termList=new ArrayList<Term>();
    }


    /**
     * Constructor copies from existing RuleSituation
     * @param s
     * @param st
     * @param model source
     * @param num   newe numbering of Terms
     */

    public RuleSituation(Situation s,String st,RuleSituation model,int num) {
        situation=s;
        if (st.equals("prefix")) prefix=true;
        else if (st.equals("suffix")) suffix=true;
        termList=new ArrayList<Term>();
        // copy the termlist
        for (Term terms: model.termList) {
            Term t=new Term(terms,num++);
```

```
            termList.add(t);
        }
    }

    /**
     * add a Term to the List of Terms
     * @param t
     * @return
     */
    public boolean add(Term t) {
        for (Term terms: termList) {
            if (terms.name.equals(t.name)) {
                System.err.println("Duplicated entry in RuleSituation: name="+t.name);
//                              return false;
            }
        }
        termList.add(t);
        return true;
    }

    /**
     * create a list of possible forms with this rule and this situation
     * @param abc
     */

    public void produce(ArrayList<Form> li,String abc,boolean withStem) {
        for (Term terms: termList) {
            if (!withStem && terms.name.endsWith("stem")) continue;
            String fo=terms.produce(abc,word);
            if (fo==null) continue;
            String a[]=fo.split(",");
            for (int la=0; la<a.length; la++)
                li.add(new Form(this,terms,abc,a[la]));
        }
    }

    /**
     * create a list of possible forms with this rule and this situation with prefix and/or
suffix
     * @param li
     * @param abc
     * @param pre
     * @param suf
     * @param withStem
     */
    public void produce(ArrayList<Form> li,String abc,Form pre,Form suf,boolean withStem) {
        boolean mitPre=(pre!=null);
        boolean mitSuf=(suf!=null);
        if (mitPre && !situation.overlaps(pre.rsituation.situation)) return;
        if (mitSuf && !situation.overlaps(suf.rsituation.situation)) return;
```

```
        for (Term terms: termList) {
            if (!withStem && terms.name.endsWith("stem")) continue;
            if (mitPre && (!terms.name.startsWith("n_") && !(terms.name.startsWith("v_") &&
terms.name.contains("INF"))))
                continue;
            if (mitSuf && (terms.name.contains("num") || terms.name.contains("pro")))
continue;
            String fo=terms.produce(abc,word);
            if (fo==null) continue;
            String a[]=fo.split(",");
            for (int la=0; la<a.length; la++)
                li.add(new Form(this,terms,abc,a[la],pre,suf));
        }
    }

    /**
     *
     * @param root
     * @param withStem
     * @return
     */
    public String produce(String root,boolean withStem) {
        String ret="";
        for (Term terms: termList) {
            if (!withStem && terms.name.endsWith("stem")) continue;
            String fo=terms.produce(root,word);
            ret+=((ret.length()>0)?",":"")+fo;
        }
        return ret;
    }

    /**
     * change a Term in termList, which has the name n to the values specified;
     * if no such term exists, add a new term
     * @param n
     * @param f1
     * @param f2
     * @param f3
     * @param f4
     * @param s1
     * @param s2
     * @return
     */
    public boolean changeTerm(String n,String f1,String f2, String f3, String f4,int s1,int
s2,String tr) {
        Term t=new Term(n,f1,f2, f3, f4,s1,s2,tr,0);
        int i;
        for (i=0; i<termList.size(); i++) {
            if (termList.get(i).name.equals(n)) {
                t.number=termList.get(i).number;
```

```
                termList.set(i, t);
                return true;
            }
        }
        termList.add(t);
        return false;
    }
}
```

8.10 SabaicInit

… initialisiert die Regeldatenbank in Form von Objekten auf Basis der Regeldatei.

```
package kalam;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

public class SabaicInit {

    public static String confDir1="C:\\projekte\\kalam.jdev\\public_html\\";
    public static String confDir2="/etc/kalam/";
    public static String confFileName="kalam.conf";
    private static String confFileNameUsed=null;

    /**
     * initialization of test-rules
     */

    public static boolean init() {

        TimePeriod tpaltsab=new TimePeriod("old-sabaic",-1000,-300);
       TimePeriod tprestsab=new TimePeriod("middle+late-sabaic",-300,600);
        TimePeriod tpallsab=new TimePeriod("sabaic",-1000,600);
        TimePeriod tpaltmitsab=new TimePeriod("old+middle-sabaic",-1000,400);
        TimePeriod tpspaetsab=new TimePeriod("late-sabaic",400,600);
        TimePeriod tpmitsab=new TimePeriod("middle-sabaic",-300,400);
       TimePeriod.TimePeriodList.add(tpaltsab);
        TimePeriod.TimePeriodList.add(tprestsab);
        TimePeriod.TimePeriodList.add(tpallsab);
        TimePeriod.TimePeriodList.add(tpaltmitsab);
        TimePeriod.TimePeriodList.add(tpspaetsab);
```

```
        Situation sitalt=new Situation("sitold",Locations.Sabaa,tpaltsab);
        Situation.add(sitalt);
        Situation sitrest=new Situation("sitmidlate",Locations.Sabaa,tprestsab);
        Situation.add(sitrest);
         Situation sitall=new Situation("sitall",Locations.Sabaa,tpallsab);
         Situation.add(sitall);
         Situation sitaltmit=new Situation("sitoldmid",Locations.Sabaa,tpaltmitsab);
         Situation.add(sitaltmit);
         Situation sitspaet=new Situation("sitlate",Locations.Sabaa,tpspaetsab);
         Situation.add(sitspaet);
         Situation sitmit=new Situation("sitmid",Locations.Sabaa,tpmitsab);
         Situation.add(sitmit);
         FileReader fr=null;
         try {
             ErrorLog.writelogfile("KALAM: SabaicInit.init() "+confDir1+confFileName);
             try {
                 fr = new FileReader(new File(confFileNameUsed=confDir1+confFileName));
             } catch (Exception e) { }
             if (fr==null) {
                 ErrorLog.writelogfile("KALAM: SabaicInit.init() "+confDir2+confFileName);
                 fr = new FileReader(new File(confFileNameUsed=confDir2+confFileName));
             }

             BufferedReader br=new BufferedReader(fr);

             String li=null;
             String lastRuleName="";
             Rule rule=null;
             int zeile=1;
             int count=1;
             while ((li!=null) || ((li=br.readLine())!=null)) {
                 zeile++;
                 if (li.startsWith("#")) continue;   // comment
                 if (li.charAt(0)=='*') li=li.substring(1);
                 String []arr=li.split("\t");
                 if (arr.length<3 || arr.length>4) {
                     ErrorLog.writelogfile("Config  file:  title  line  must  contain  3  or  4
entries!");
                     return false;
                 }
                 String ruleName=arr[0];
                 if (!ruleName.equals(lastRuleName)) {   // neue Rule anlegen
                     rule=new Rule(ruleName);
                     Rule.allRules.add(rule);
                     lastRuleName=ruleName;
                 }
                 String tilo=arr[1];
                 String stem=arr[2];

                 boolean isWord=stem.equals("word");
```

```
                Situation sit=Situation.search(tilo);
                if (sit==null) {
                    ErrorLog.writelogfile("Config file: unknown situation '"+tilo+"'!");
                    return false;
                }

                RuleSituation ruleSit=null;
                Rule mergeSrc=null;
                String mergeSrcName=null;
                boolean change=false, merge=false;
                if (arr.length==4) {
                    switch (arr[3].charAt(0)) {
                    case '=':
                        String src=arr[3].replace("=", "");
                        Situation srcSit=Situation.search(src);
                        if (srcSit==null) {
                            ErrorLog.writelogfile("Config   file:   unknown   source-situation
'"+src+"'!");
                            return false;
                        }
                        for (RuleSituation rs: rule.ruleList) {
                            if (rs.situation.name.equals(src)) {
                                ruleSit=new RuleSituation(sit,stem,rs,count);
                                count+=ruleSit.termList.size();
                                change=true;
                                break;
                            }
                        }
                        break;
                    case '*':
                        mergeSrcName=arr[3].substring(1);
                        mergeSrc=Rule.search(mergeSrcName);
                        if (mergeSrc==null) {
                            ErrorLog.writelogfile("Config  file:  unknown  source  for  merging:
'"+mergeSrcName+"'!");
                            return false;
                        }
                        merge=true;
                        ruleSit=new RuleSituation(sit,stem);
                        break;
                    }
                }
                else ruleSit=new RuleSituation(sit,stem);
                rule.add(ruleSit);
                ruleSit.word=isWord;

                while ((li=br.readLine())!=null) {
                    zeile++;
                    if (li.startsWith("#")) continue;   // comment
```

```
                    String []tr=li.split("\t");
                    if (li.charAt(0)=='*') break;
                    if (tr[0].equals("null") || tr[0].equals("end")) {
                        li=null;
                        break; }

                    if (isWord && tr.length==3) {     // conf contains      name      word
translation
                        String tr0[]=new String[8];
                        tr0[0]=tr[0];
                        tr0[1]=tr[1];
                        tr0[2]="null";
                        tr0[3]="null";
                        tr0[4]="null";
                        tr0[5]="0";
                        tr0[6]="0";
                        tr0[7]=tr[2];
                        tr=tr0;
                    }

                    if (tr.length<7) {
                        ErrorLog.writelogfile("Config file: wrong term in line "+zeile+"!");
                        return false;
                    }
                    for (int i=1; i<=4; i++)
                        if (tr[i].toLowerCase().equals("null")) tr[i]=null;
                        else tr[i]=Convert.decode(tr[i]);
                    int ii[]=new int[2];
                    for (int i=5, j=0; i<=6; i++, j++) {
                        try {
                            ii[j]=Integer.parseInt(tr[i]);
                            if (ii[j]<0 || ii[j]>5) {
                                ErrorLog.writelogfile("Config file: wrong number in term in
line "+zeile+"!");

                                return false;
                            }
                        } catch (Exception e) {
                            ErrorLog.writelogfile("Config file: wrong number in term in line
"+zeile+"!");

                            return false;
                        }
                    }
                    String translation=null;
                    ArrayList<Weak> weak=null;
                    if (tr.length==8 && !tr[7].contains("="))
                        if (tr[7].toLowerCase().equals("null")) translation=null;
                        else translation=Convert.decode(tr[7]);
                    else if (tr.length>=8) {    // weak-rules
                        weak=new ArrayList<Weak>();
                        for (int w=7; w<tr.length; w++) {
```

```
                            String warr[]=tr[w].split("=");
                            if (warr.length!=2 || warr[0].length()!=3 || warr[1].length()<3) {
                                ErrorLog.writelogfile("Config    file:    wrong    'weak'-field
"+tr[w]+" in line "+zeile+"!");
                                return false;
                            }
                            weak.add(new Weak(Convert.decode(warr[0]),warr[1]));
                        }
                    }
                    if                                                                  (change)
ruleSit.changeTerm(tr[0],tr[1],tr[2],tr[3],tr[4],ii[0],ii[1],translation);
                    else if (merge) {   // the current term is inside the merge-term
                        RuleSituation rsSrc=null;
                        // search for a fitting situation
                        for (RuleSituation rs:mergeSrc.ruleList) {
                            if (rs.situation.overlaps(ruleSit.situation)) {
                                rsSrc=rs;
                                break;
                            }
                        }
                        if (rsSrc==null) rsSrc=mergeSrc.ruleList.get(0);    // take the first
one instead
                        int off=mergeSrcName.length();
                        for (Term tSrc: rsSrc.termList) {
                            String tr0=tr[0]+tSrc.name.substring(off);
                            String tr1=tSrc.fill[0]+((tr[1]==null)?"":tr[1]);
                            String tr2=tSrc.fill[1]+((tr[2]==null)?"":tr[2]);
                            String tr3=((tr[3]==null)?"":tr[3])+tSrc.fill[2];
                            String tr4=((tr[4]==null)?"":tr[4])+tSrc.fill[3];
                            Term                                                        te=new
Term(tr0,tr1,tr2,tr3,tr4,ii[0],ii[1],translation,count++);
                            ruleSit.add(te);
                            te.weakList=weak;
                        }
                    }
                    else {
                        Term                                                            te=new
Term(tr[0],tr[1],tr[2],tr[3],tr[4],ii[0],ii[1],translation,count++);
                        ruleSit.add(te);
                        te.weakList=weak;
                    }
                }
                zeile--;    // da oben keine neue Zeile eingelesen wird, aber trotzdem zeile++
            }
            fr.close();
        } catch (FileNotFoundException e) {
            ErrorLog.writelogfile("Config file "+confFileName+" not found!");
            return false;
        } catch (IOException e) {
            ErrorLog.writelogfile("Config file "+confFileName+" IO-exception!");
```

```
            return false;
        }
        return true;
    }
}
```

8.11 Situation

… implementiert das Situation Objekt.

```
package kalam;

import java.util.ArrayList;

/**
 * pair Location/TimePeriod
 * @author rr
 *
 */
public class Situation {
    public Locations location;
    public TimePeriod time;

    public String name;

    private static ArrayList<Situation> list=new ArrayList<Situation>();

    /**
     * Constructor
     * @param n
     * @param l
     * @param t
     */
    public Situation(String n,Locations l,TimePeriod t) {
        location=l;
        time=t;
        name=n;
    }

    /**
     * add Situation to list of situations
     * @param s
     */
    public static void add(Situation s) {
        list.add(s);
    }

    /**
     * search for situation
     * @param sit
```

```
     * @return
     */
    public static boolean search(Situation sit) {
        for (Situation s:list) {
            if (!sit.location.equals(s.location) && sit.time.equals(s.time)) return true;
        }
        return false;
    }

    /**
     * search for situation, based on name
     * @param name
     * @return
     */
    public static Situation search(String name) {
        for (Situation s:list) {
            if (s.name.equals(name)) return s;
        }
        return null;
    }

    /**
     * tests, whether the location is the same and there is an overlapping time-interval
     * @param sit
     * @return
     */
    public boolean overlaps(Situation sit) {
        if (location==sit.location) return false;
        if (time.to<sit.time.from || time.from>sit.time.to) return false;
        return true;
    }
}
```

8.12 Term

… implementiert das Term-Objekt.

```
package kalam;

import java.util.ArrayList;

/**
 * changing rules for a single rule, e.g. 3. p. s. Imperfect
 * @author rr
 *
 */

public class Term {

    public String name;
```

```
    public String fill[]={ "", "", "", ""}; // fill[i] in before (i.+1)-th radical
    public int swap[] = { 0, 0};            // i-th and j-th radical are swapped; based on 1,
2, 3,...
    public String translation=null;
    public ArrayList<Weak> weakList=null;
    public int number;

    /**
     * Constructor
     * @param n
     * @param f1
     * @param f2
     * @param f3
     * @param f4
     * @param s1
     * @param s2
     * @param num
     */
    public Term(String n,String f1,String f2, String f3, String f4,int s1,int s2,int num) {
        name=n;
        if (f1==null) fill[0]="";
        else fill[0]=f1;
        if (f2==null) fill[1]="";
        else fill[1]=f2;
        if (f3==null) fill[2]="";
        else fill[2]=f3;
        if (f4==null) fill[3]="";
        else fill[3]=f4;

        if (s1>0) swap[0]=s1;
        if (s2>0) swap[1]=s2;
        number=num;
    }

    /**
     * Constructor including translation tr and internal numbering num
     * @param n
     * @param f1
     * @param f2
     * @param f3
     * @param f4
     * @param s1
     * @param s2
     * @param tr
     * @param num
     */
    public Term(String n,String f1,String f2, String f3, String f4,int s1,int s2,String tr,int
num) {
        name=n;
        if (f1==null) fill[0]="";
```

```
        else fill[0]=f1;
        if (f2==null) fill[1]="";
        else fill[1]=f2;
        if (f3==null) fill[2]="";
        else fill[2]=f3;
        if (f4==null) fill[3]="";
        else fill[3]=f4;

        if (s1>0) swap[0]=s1;
        if (s2>0) swap[1]=s2;
        translation=tr;
        number=num;
    }

    public Term(String n,String word,int num) {
        name=n;
        fill[0]=word;
        number=num;
    }

    /**
     * copy a Term object
     * @param model
     */

    public Term(Term model,int num) {
        name=new String(model.name);
        for (int i=0; i<4; i++) fill[i]=new String(model.fill[i]);
        for (int i=0; i<2; i++) swap[i]=model.swap[i];
        number=num;
    }

    /**
     * creates the term according to the root abc
     * @param abc
     * @return
     */

    public String produce(String abc) {
        if (abc==null) return null;
        int l=abc.length();
        if (l<2) return null;
        int l0=l;
        if (l>4) l=4;
        String r[]=new String[l];
        for (int i=0; i<l-1; i++)
            r[i]=abc.substring(i,i+1);
        r[l-1]=abc.substring(l-1);
        boolean doSwap=(swap[0]<=l && swap[1]<=l && swap[0]+swap[1]>0);
        if (doSwap) {
```

```
            String h=r[swap[0]-1]; r[swap[0]-1]=r[swap[1]-1]; r[swap[1]-1]=h;
        }
        String out="";
        boolean found=false;
        if (weakList!=null) {   // apply weak-rules
            for (Weak ww: weakList) {
                if (ww.fits(abc)) {
                    String r0[]=new String[l];
                    for (int u=0; u<ww.use.length; u++) {
                        for (int rr=0; rr<l; rr++) {
                            r0[rr]=(ww.use[u][rr])?r[rr]:"";
                        }
                        out+=((out.length()>0)?",":"")+fill[0]+r0[0]+fill[1]+r0[1]+fill[2];
                        if (l==3) {
                            out+=r0[2]+fill[3];
                        }
                        if (l==4) out+=r0[2]+r0[3]+fill[3];
                    }
                    found=true;
                    break;
                }
            }
        }
        if (!found) {
            out=fill[0]+r[0]+fill[1]+r[1]+fill[2];
            if (l==3) {
                out+=r[2]+fill[3];
            }
            if (l==4) out+=r[2]+r[3]+fill[3];
        }
        return out;
    }

    /**
     * Term is a word only
     * @param abc
     * @return
     */

    public String produceWord(String abc) {
        if (abc==null) return null;
        if (abc.equals(fill[0])) return abc;
        return null;
    }

    public String produce(String abc,boolean word) {
        return (word)?produceWord(abc):produce(abc);
    }

    /**
```

```
     * analyzes the term's name and returns an explanation
     * @param term
     * @return
     */
    public static String nameAnalyze(String term) {
        String r="";
        if (term.startsWith("v_")) {
            r+="verb; ";
            int max=term.length();
            String co=term.substring(2,max);
            if (co.startsWith("H")) co=term.substring(3,max);
            else if (co.startsWith("O2") || co.startsWith("T1") || co.startsWith("T2") ||
co.startsWith("ST"))
                co=term.substring(4,max);
            boolean cj=false;
            if (co.startsWith("SK_")) {
                r+="suffix conjugation; ";
                cj=true; }
            else if (co.startsWith("PK0")) {
                r+="prefix conjugation; ";
                cj=true; }
            else if (co.startsWith("PKN")) {
                r+="prefix conjugation long; ";
                cj=true; }
            else if (co.startsWith("IMPk")) {
                r+="imparative; ";
                cj=true; }
            else if (co.startsWith("IMPl")) {
                r+="imparative long; ";
                cj=true; }
            else if (co.startsWith("INF")) {
                r+="infinitive; ";
            }
            if (cj) {
                r+=getPerson(term);
            }
        }
        else if (term.startsWith("n_P")) {
            r+="passive/active participle; ";
            int ii=term.indexOf('_',3);
            String co=term.substring(ii-2);
            if (co.startsWith("SC_")) {
                r+="status constructus; ";
            }
            else if (co.startsWith("SI_")) {
                r+="status indeterminatus; ";
            }
            else if (co.startsWith("SD_")) {
                r+="status determinatus; ";
            }
```

```
        r+=getPerson(term);
    }
    else if (term.startsWith("n_LO")) {
        r+="nomen loci; ";
        int ii=term.indexOf('_',3);
        String co=term.substring(ii-2);
        if (co.startsWith("SC_")) {
            r+="status constructus; ";
        }
        else if (co.startsWith("SI_")) {
            r+="status indeterminatus; ";
        }
        else if (co.startsWith("SD_")) {
            r+="status determinatus; ";
        }
        r+=getPerson(term);
    }
    else if (term.startsWith("n_")) {
        r+="noun; ";
        String co=term.substring(2,6);
        if (co.startsWith("SC_")) {
            r+="status constructus; ";
        }
        else if (co.startsWith("SI_")) {
            r+="status indeterminatus; ";
        }
        else if (co.startsWith("SD_")) {
            r+="status determinatus; ";
        }
        r+=getPerson(term);
    }
    else if (term.startsWith("part_")) {
        r="particle";
    }
    else if (term.startsWith("cardnum_")) {
        r="cardinal number";
    }
    else if (term.startsWith("ordnum_")) {
        r="ordinal number";
    }
    else if (term.startsWith("fracnum_")) {
        r="fraction";
    }
    else if (term.startsWith("partikel_")) {
        r="particle";
    }
    else if (term.startsWith("conjunction_")) {
        r="conjunction";
    }
    else if (term.startsWith("ppros_")) {
```

```
            r="personal pronoun (suffix); ";
            r+=getPerson(term);
        }
        else if (term.startsWith("ppro_")) {
            r="personal pronoun; ";
            r+=getPerson(term);
        }
        else if (term.startsWith("relp_")) {
            r="relative pronoun; ";
            r+=getPerson(term);
        }
        else if (term.startsWith("dempronear_")) {
            r="demonstrative pronoun (near); ";
            r+=getPerson(term);
        }
        else if (term.startsWith("demprofar_")) {
            r="demonstrative pronoun (far); ";
            r+=getPerson(term);
        }
        else if (term.startsWith("relpro_")) {
            r="relative pronoun; ";
            r+=getPerson(term);
        }
        return r;
    }

    /**
     * returns the full-text expression of the grammar term
     * @param term
     * @return
     */

    private static String getPerson(String term) {
        String person;
        String r="";
        int start=term.indexOf('_');
        person=term.substring(term.indexOf('_',start+1)+1);
        if (person.equals("stem"))
            return "stem ";
        char ch=person.charAt(0);
        int ind=0;
        if (ch>='1' && ch<='3') {
            r+=ch+". ";
            ind++;
        }
        switch (person.charAt(ind)) {
            case 'm': r+="masculin "; ind++; break;
            case 'f': r+="feminin "; ind++; break;
            case 'c': r+="communis "; ind++; break;
        }
```

```
        switch (person.charAt(ind)) {
            case 's': r+="singular "; break;
            case 'd': r+="dual "; break;
            case 'p': r+="plural "; break;
        }
        ind++;
        if (person.length()>ind) {
            switch (person.charAt(ind)) {
                case 'n': r+="nominative "; break;
                case 'o': r+="obliqus "; break;
            }
        }
        return r;
    }
}
```

8.13 TimePeriod

… implementiert das Zeit-Bereichsobjekt.

```
package kalam;

import java.util.ArrayList;

/**
 * single TimePeriod
 * @author rr
 *
 */

public class TimePeriod {

    public static ArrayList<TimePeriod> TimePeriodList=new ArrayList<TimePeriod>();

    public String name="";
    public int from=-1000;    // years interval
    public int to=600;

    /**
     *
     * @param n
     * @param f
     * @param t
     */
    public TimePeriod(String n,int f,int t) {
        name=n;
        from=f;
        to=t;
    }
}
```

8.14 Weak

… implementiert schwache Radikale.

```
package kalam;

public class Weak {
    public char root[]=new char[3];        // e.g. n**     *w* *..
    public boolean use[][];    //  radicals to be used (use[3]=use[4]=use[2] ... 4,5-lenth-
roots

    /**
     * Constructor
     * @param r root
     * @param u list of usages, like:  101,111
     */
    public Weak(String r,String u) {
        root[0]=r.charAt(0);
        root[1]=r.charAt(1);
        root[2]=r.charAt(2);

        String arr[]=u.split(",");
        int l=arr.length;
        use=new boolean[l][];
        for (int i=0; i<l; i++) {
            use[i]=new boolean[5];
            for (int j=0; j<3; j++)
                use[i][j]=(arr[i].charAt(j)=='1');
            use[i][3]=use[i][4]=use[i][2];
        }
    }

    /**
     * does the root ro fit to this Weak model
     * @param ro
     * @return
     */
    public boolean fits(String ro) {
        if (root[0]!='*' && root[0]!=ro.charAt(0)) return false;
        if (root[1]!='*' && root[1]!='.' && root[1]!=ro.charAt(1)) return false;
        switch (root[2]) {
        case '*': return true;
        case '.': return ro.charAt(1)==ro.charAt(2);
        default:  return ro.charAt(2)==root[2];
        }
    }
}
```

9. Schlussbemerkungen

KALAM ist in Englisch gehalten und steht nun als Webapplikation frei zugänglich zur Verfügung. Im Speziellen bietet KALAM zusätzlich eine grafische Möglichkeit zur Eingabe der nicht-lateinischen Zeichen.

Der Nutzen des Programms liegt nicht nur in der Hilfestellung für Studenten, sondern auch in der Unterstützung der täglichen Arbeit von Fachleuten.

Im Rahmen des DASI-Projekts der Universität Pisa wurde auch der *Corpus of South Arabic Inscriptions* realisiert, wobei in Richtung der Sabäischen Sprache noch sehr viel Arbeit zu erledigen ist. Diese Onlineplattform listet Bilder, Inhalt, Transkription und teilweise Übersetzungen von alt-südarabischen Texten.

In mühevoller Handarbeit wird bei diesem Projekt zu neuen Wörtern die Grammatik ermittelt und manuell eingetragen – hier könnte KALAM eine große Arbeitserleichterung darstellen.

Umgekehrt wird – wenn dies von Seiten des DASI-Projektes möglich ist – mit dem Resultat von KALAM auch gleich eine Link-Verknüpfung in die Wort-Liste der DASI-Datenbank angezeigt.

Das Grundanliegen des Autors war jedenfalls die praktische Nutzbarkeit seiner Masterarbeit. Insoweit stand eben auch der praktische Teil im Vordergrund.

Für die Zukunft wäre es interessant, KALAM mit anderen Projekten zu koppeln und eventuell hin zu einer Textanalyse Software weiterzuentwickeln.[63]

[63] Im Rahmen einer Dissertation zum Beispiel ☺

10. Bibliographie

ABATE2014 Mesfin Abate, Yaregal Assabie: *Development of Amharic Morphological Analyzer Using Memory-Based Learning*; in: Advances in Natural Language Processing, 9th International Conference on NLP, PolTAL2014. Heidelberg: Springer, 2014.

ALIWY2012 Ahmed H. Aliwy: *Arabic Language Analyzer with Lemma Extraction and Rich Tagset*; in: Advances in Natural Language Processing, 8th International Conference on NLP, JapTAL2012. Heidelberg: Springer, 2012.

AVANZINI2009 Alessandra Avanzini: *Origin and classification of the Ancient South Arabian languages*; in: Journal of Semitic Studies 54. Oxford: Oxford University Press, 2009.

BEESTON1962 Alfred F.L. Beeston: *A Descriptic Grammar of Epigraphic Southarabian*. London: Luzac, 1962.

BEESTONetal1982 Alfred F.L. Beeston, M.A. Ghul, W.W. Müller, J. Ryckmans: *Sabaic Dictionary (English-French-Arabic)*. Louvain-la-Neuve: Peeters,1982.

BEESTON1984 Alfred F.L. Beeston: *Sabaic Grammar*; Journal of Semitic Studies Monograph No. 6. Manchester: University of Manchester, 1984.

CHOI2005 Myung-Seok Choi, Chul Su Lim, Key-Sun Choi: *Automatic Partial Parsing Rule Acquisition Using Decision Tree Induction*; in: Natural Language Processing – IJNLP 2005, Second International Joint Conference, Jeju Island, Korea, October 2005, Proceedings. Heidelberg: Springer, 2005.

CHRISTOF2006 K.D.Christof, Renate Haass: *Weihrauch – Der Duft des Himmels.* Dettelbach: Verlag J.H.Röll, 2006.

CONTIROSSINI1931 Carlo Conti Rossini: *Chrestomathia Arabica meridionalis epigraphica: edita et glossario intructa*. Roma: Ist. per l'Oriente, 1931.

CSENDES2004 Dóra Csendes, János Csirik, Tibor Gyimóthy: *The Szeged Corpus: A POS Tagged and Syntactically Annotated Hungarian Natural Language Corpus*; in: Text, Speech and Dialogue – 7th International Conference, TSD 2004, Brno, Czech Republic, September 2004, Proceedings. Heidelberg: Springer, 2004.

GUIDI1926 Ignazio Guidi: *Summarium grammaticae arabicae meridionalis*; in: Le Muséon 39/1926. Publ. Universitaires, Louvain: 1926.

HATKE2014 George Hatke: *Seminar zur Geschichte und Archäologie Südarabiens*. PDF-Folien: 2014.

HOMMEL1893 Fritz Hommel: *Südarabische Chrestomathie*. München: Verlag Franz, 1893.

LUCKHARDT1991 Heinz-Dirk Luckhardt, Harald H. Zimmermann: *Computergestützte und maschinelle Übersetzung*. Saarbrücken: AQ-Verlag, 1991.

MÜLLER1994 Walter W. Müller: *Die altsüdarabische Schrift*; in: Hartmut Günther et al (Hrsg.): Schrift und Schriftlichkeit . Ein interdisziplinäres Handbuch internationaler Forschung. 1. Halbband (Handbücher zur Sprache- und Kommunikationswissenschaft 10). Berlin: de Gruyter, 1994.

MULTHOFF2012 Anne Multhoff: *Die Verbalstammbildung im Sabäischen*; Dissertation an der philosophischen Fakultät der Friedrich-Schiller-Universität Jena. Jena: 2012.

NEBES1991 Norbert Nebes: *Die enklitischen Partikeln des Altsüdarabischen*; in: Études sud-arabes. Recueil offert à Jacques Ryckmans (Publications de l'Institut Orientaliste de Louvain 39). Louvain-la-Neuve: Peeters, 1991.

NEBES1997 Norbert Nebes: *Stand und Aufgaben einer Grammatik des Altsüdarabischen*; in: R.G. Stiegner (Hrsg.): Aktualisierte Beiträge zum 1. Internationalen Symposium Südarabien interdisziplinär an der Universität Graz. Graz: Leykam, 1997.

RETSÖ1992 Jan Retsö: *The domestication of the camel and the establishment of the frankincense road from South Arabia*; in Orientalia Suecana XXXX-XXXXI (1990-1991). Uppsala: Universitetsbiblioteket Uppsala, 1992.

STANFORD2005 Stanford University: Stanford Log-linear Part-Of-Speech Tagger; *http://nlp.stanford.edu/software/tagger.shtml*. Abgefragt am 2015-11-01.

STEIN2004 Peter Stein: *Zur Dialektgeographie des Sabäischen*; in: Journal of Semitic Studies 49. Oxford: Oxford University Press, 2004.

STEIN2012 Peter Stein: *Ein weiteres arabisches Syntagma in der altsüdarabischen Epigraphik*; in: Babel und Bibel 6 – Annual of Ancient Near Eastern, Old Testament, and Semitic Studies (Orientalia et Classica; Papers of the Institue of Oriental and Classical Studies; the Russian State University for the Humanities). Winona Lake, Indiana: Eisenbrauns, 2012.

STEIN2013 Peter Stein: *Lehrbuch der sabäischen Sprache – 1. Teil Grammatik*. Wiesbaden: Harrasowitz Verlag, 2013.

TEMMAM2004 Abu Temmām: *Hamāsa – oder die ältesten arabischen Volkslieder: Gesammelt von Abu Temmam*, übertragen von Friedrich Rückert. Verlag Georg Olms, Hildesheim: 2004.

TOUTANOVA2003 Kristina Toutanova, Dan Klein, Christopher Manning, Yoram Singer: *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*; in: Proceedings of HLT-NAACL 2003. Edmonton, Canada: 2003.

UERPMANN2002 Hans-Peter und Margarethe Uerpmann: *The Appearance of the Domestic Camel in South-east Arabia*, in The Journal of Oman Studies, Volume 12. Muscat: Ministry of Heritage and Culture: 2002.

WOODARD2008 Roger D. Woodard (Editor): *The Ancient Languages of Syria-Palestine and Arabia*. Cambridge: Cambridge University Press: 2008.

Anhang A. Zusammenfassung

Wenn man als Student in die alt-südarabischen Sprachen eintaucht, steht man bei der Übersetzung mancher Textstelle vor dem Problem, dass man so gar nicht weiß, was diese bedeuten soll: schon die Kenntnis eines einziges Wortes würde helfen. In dieser Situation ist die Idee entstanden, ein Computerprogramm zu entwickeln, bei dem man ein alt-südarabisches Wort eingibt und die grammatikalische Analyse zurück erhält: alle Möglichkeiten für Wurzel, Stamm, Person, Zahl, Geschlecht, Zeit/Aspekt.

Als Basis für die Grammatik dient das *Lehrbuch der sabäischen Sprache* von Peter Stein. Dieses 2013 erschienene Werk bietet die modernste Sicht auf das Sabäische. Betont sei hier auch, dass die Masterarbeit sich alleinig auf die Sprache des Sabäischen beschränkt.

Der theoretische Teil dieser Arbeit lässt zuerst in Kürze die Geschichte Südarabiens und danach die Wiederentdeckungsgeschichte aus europäischer Sicht Revue passieren. Sie befasst sich dann im Detail mit der Entwicklung der Grammatiken zum Sabäischen: angefangen von der Entzifferung durch Gesenius, den ersten großen Schriftensammlungen von Halévy und Glaser, der ersten Erstellung einer Grammatik durch Hommel Ende des 19. Jahrhunderts, dann den Schriften von Conti Rossini, Höfner, Beeston, Nebes, Stein und Multhoff in unserer Zeit. Anhand der Verbparadigmen werden Gemeinsamkeiten und Unterschiede erläutert.

Die Untersuchung mehrerer Arbeiten zum Thema Wortanalyse aus dem Bereich der Linguistik, zu unterschiedlichen Sprachen wie Englisch, Ungarisch, Amharisch, Koreanisch und Arabisch erläutert die Grundbegriffe *Natural Language Processing* und *POS-Tagging*. Sie zeigt, dass man Ansätze der hierin vorgestellten Methoden durchaus verwenden kann, die darin vorgestellten Computerprogramme aber nicht geeignet sind, eine Sabäisch-Wortanalyse damit zu erstellen.

Also wurde das Programm KALAM entwickelt, das nach dem „Erraten“ der Wurzel eines Wortes durch Synthese die grammatikalische Form bestimmt.

Mehr als 2.000 sabäische Wörter sind in einer Lexikondatenbank dem Programm hinterlegt.

KALAM liefert alle theoretisch möglichen Formen, die hinter einem Wort stecken können und schränkt diese auf Wunsch durch Rückgriff auf das Lexikon ein.

Der Benutzer kann auch eine Wurzel vorgeben und erhält alle grammatikalischen Formen angezeigt.

Umgekehrt kann der Benutzer auch nach englischen Begriffen suchen und erhält die sabäischen Einträgt zurück.

Anhang B. Abstract

Being a student of the ancient South Arabian languages you often experience the problem of not knowing how to deal with a sentence: the knowledge of one word would help. In this situation the idea was born to create a computer software, which reads a Sabaic word, analyzes it and returns the root, stem, person, gender, time/aspect.

The *Lehrbuch der sabäischen Sprache* of Peter Stein was used as the base for the grammar. This book from the year 2013 offers the most modern view onto the Sabaic language. This master thesis deals with the Sabaic language only.

The theoretical part of this thesis gives a short description of the history of ancient South Arabia and the history or re-discovery seen from the European point of view. In detail it then describes the development of Sabaic grammars: starting with the decipherment by Gesenius, the first large collections of inscriptions by Halévy and Glaser, the first grammar by Hommel at the end of the 19th century, then the works of Conti Rossini, Höfner, Beeston, Nebes, Stein and Multhoff in our times. The grammars are compared by showing differences and similarities of verb paradigms.

The study of a few scientific papers out of the area of linguistics - for the languages English, Hungarian, Amharic, Korean and Arabic - explains the methods of *Natural Language Processing* and *POS-Tagging*. This study proofs that the basic ideas of these methods can be used within a program for a Sabaic word analyzer but the computer programs promoted in the papers could not be used as they are.

Therefore the program KALAM has been developed to search for the root of a word and detect the grammar-form by using synthesis. More than 2.000 Sabaic words have been stored in its dictionary-database.

KALAM shows all theoretically possible forms a word could consist of. Optionally it restricts the solution to real existing forms by using the dictionary. The user may also request all forms of a root and ask for the Sabaic word based on an English term.