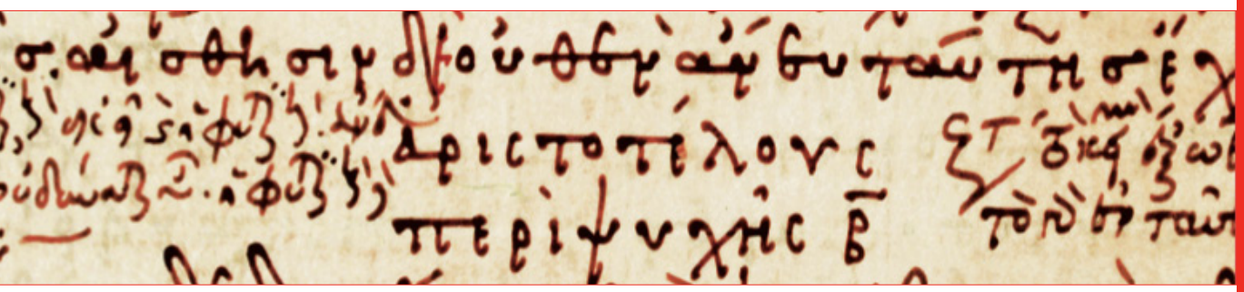
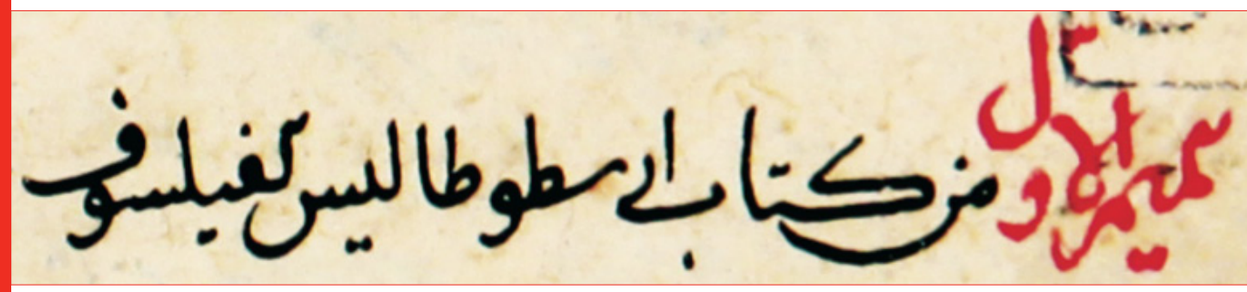
ISSN 2239-012X



With the support of the European Research Council

Studia graeco-arabica

Studia graeco-arabica



5

2015

Studia graeco-arabica

The Journal of the Project

Greek into Arabic

Philosophical Concepts and Linguistic Bridges

European Research Council Advanced Grant 249431

5

2015

Published by
ERC Greek into Arabic
Philosophical Concepts and Linguistic Bridges
European Research Council Advanced Grant 249431

studiagraecoarabica@greekintoarabic.eu

Web site: http://www.greekintoarabic.eu
Service Provider: Università di Pisa, Area Serra - Servizi di Rete Ateneo

ISSN 2239-012X (Online)

Registration at the law court of Pisa, 18/12, November 23, 2012.
Editor in chief Cristina D'Ancona.

Cover
Mašhad, Kitābḫāna-i Āsitān-i Quds-i Raḍawī 300, f. 1v
Paris, Bibliothèque Nationale de France, grec 1853, f. 186v

Studia graeco-arabica

5

2015

Alpha into Alif

Schnittstellen zwischen Schriftkunde und Informatik am Beispiel von Unicode im Glossarium Graeco-Arabicum

Torsten Roeder

Abstract
The project *Glossarium Graeco-Arabicum* connects the study of writing systems with the field of information science, utilizing the Unicode standard. This paper points out particular historical developments both in philology and informatics that help to develop a modern approach to working in digital poly-alphabetical environments, based on interdisciplinary expertise. A description of how Unicode was implemented to represent ancient Greek and classical Arabic is followed by a historical outline of computer writing systems. Essential aspects are letter collationing, writing directions and user interfaces. The conclusion emphasizes the importance of a bilateral understanding of historical and technical disciplines.

1. Einleitung

1.1 Thematik

Wissenschaftliche Anwendungen im Internet stellen in der Regel höchst spezielle Anforderungen an die Technik. Eine Ursache dafür liegt in der Differenziertheit, mit denen die Forschungsinhalte beschrieben und behandelt werden müssen. Der Grad an Vielschichtigkeit und Feingliedrigkeit dieser Inhalte besitzt im Tagesgeschäft des digitalen Mediums nur geringe Relevanz, so dass die Komplexität die üblichen Werkzeuge meist überfordert. Obwohl digitale Abbildung und Verarbeitung von wissenschaftlichen Daten ein hohes Potenzial für die Forschung bergen, werden auch weit verbreitete Anwendungen den bestehenden Ansprüchen in der Regel kaum gerecht, sofern sie auf alltägliche Szenarien gemünzt sind. Daher müssen häufig Speziallösungen entwickelt werden, die an die Fragestellungen und Forschungsgegenstände im Einzelfall angepasst sind.

Beispiele liefern Forschungsvorhaben, die Sprachen mit unterschiedlichen Schriftsystemen zum Gegenstand haben. Das Nebeneinander verschiedener Alphabete hat, allen globalen Annäherungen zum Trotz, in der von westlichen Schreibsystemen geprägten digitalen Kultur bisher nur wenig Beachtung gefunden. Angesichts der aktuellen Schriftenvielfalt des Mittelmeerraumes (vgl. Abb. 1), der die historisch-geographische Grundlage europäischer Kultur bildet, mag dies verwunderlich erscheinen.

Abb. 1. *Schreibsysteme im Mittelmeerraum*. Quelle: Ausschnitt von Wikimedia Commons, Writing Systems of the World <https://commons.wikimedia.org/wiki/File:WritingSystemsoftheWorld.png> (14.02.2015).

Begünstigt durch die historische Entwicklung der Informationstechnik, basiert der bislang dominierende "westliche" Teil des Internets auf dem rechtsläufigen lateinischen Alphabet, während die Unterstützung oder gar Einbindung anderer Schriftsysteme fast völlig außerhalb des Blickfeldes bleibt. In der Folge wirft die Koexistenz mehrerer alphabetischer Systeme innerhalb einer Webanwendung eine Reihe von Schwierigkeiten auf, denen die Anwender, Entwickler und Designer des Netzes jedoch im Alltag äußerst selten begegnen, denn die Notwendigkeit polyalphabetischer Schriftunterstützung kann in der Regel umschifft werden. In Forschungsprojekten sind Workarounds oder Kompromisse hingegen problematisch, da es auf die exakte Wiedergabe der überlieferten Schreibungen ankommt. Somit bleibt man den konkurrierenden Kodierungssystemen, gegenläufigen Schreibrichtungen, verschiedenen Zeichenformen desselben Graphems und unzureichenden Nutzerschnittstellen ausgeliefert, für deren Behandlung bisher wenig Expertise und Vorarbeit vorhanden ist.

1.2 Zielsetzung

Während die soeben aufgezählten Aufgabenfelder vom Standpunkt einzelner Schriftsysteme aus betrachtet längst als trivial erscheinen (wobei sich manchmal auch dies noch als Trugschluss offenbart, denn bereits an europäischen Umlaut- und Akzentzeichen scheitern viele Produkte aus dem englischsprachigen Raum), ruft das Nebeneinander mehrerer Schriftsysteme selbst die scheinbar trivialsten Fragestellungen erneut auf den Plan. So entfaltet sich zwischen den technischen Lösungsmöglichkeiten und den spezifischen Strukturen der Schriftsysteme ein Spannungsfeld, welches das Ergebnis verschiedenster historischer Bedingungen und Entscheidungen auf beiden Seiten ist.

Die anschließenden Ausführungen gliedern sich in folgende drei Schritte: Erstens eine Bestandsaufnahme typischer Problematiken an einem Fallbeispiel, zweitens das Beleuchten der historischen Bedingungen, und drittens eine Auflistung möglicher Lösungsansätze. Übergeordnetes Ziel ist das Knüpfen eines Leitfadens sowie die Stabilisierung einer *best practice* für die Implementierung von Webanwendungen mit mehreren parallelen Schriftsystemen, deren Funktionstüchtigkeit – aus den Perspektiven der Datenhaltung, der Datenabbildung und des Datenzugriffs – in keinem der Schriftsysteme eingeschränkt ist. Der Weg über den historischen Pfad erscheint notwendig, da das Verständnis der Lösungswege nicht durch deren bloßes Imitieren, sondern erst über das Begreifen der zugrundeliegenden Strukturen erreicht wird. Nur auf der Erkenntnis, dass sowohl die Schriftsysteme als auch die Informationstechnik historisch gewachsenen Strukturen gehorchen, die sich gegenseitig beeinflussen, kann die Zusammenarbeit zwischen Informatik und Philologie effizient gedeihen und ihr Potenzial für die Forschung erst im vollen Maße ausgeschöpft werden. So werden möglicherweise einige der nachfolgenden Erläuterungen jeweils für eine Seite trivial erscheinen. Sie bilden jedoch für ein gegenseitiges Verständnis eine unabdingbare Voraussetzung.

2. Fallbeispiel

2.1 Auswahlkriterien

Als Fallbeispiele kommen insbesondere Projekte infrage, in denen möglichst viele, stark unterschiedliche Schriftsysteme verwendet werden. Insbesondere philologische und linguistische Vorhaben, die sich um die Erforschung von Übersetzungen drehen, sind aufgrund der direkten Gegenüberstellung zweier grundverschiedener Schriftsysteme für obige Fragestellung von großem

Interesse. Ein Studienfeld, welches in dieser Hinsicht als besonders ergiebig erscheint, bilden u. a. die sogenannten Graeco-Arabica, die sich mit arabischen Übersetzungen griechischer Texte auseinandersetzen. Als bedeutende Projekte sind etwa die Perseus Digital Library[1] mit ihren Beständen an historischen Wörterbüchern, das etwas jüngere *Digital Corpus for Graeco-Arabic Studies*[2] oder auch die *G2A WebApp*[3] zu nennen.

Eine übliche Aufgabe in diesem Feld ist die Erstellung von mehrsprachigen Glossaren. Dies bedeutet in der Regel einen großen Aufwand an systematischer Arbeit, bei der die Informationstechnik als Hilfsmittel eine wichtige Rolle spielt. Die Grundlage eines Glossars bildet die Sammlung von Wortpaaren, in der alle relevanten Lemmata samt Kontext und verschiedenen Übersetzungen zusammengetragen werden. Die informationstechnischen Unterstützungsmöglichkeiten bei dem Sammlungsprozess liegen in der systematischen Erfassung mithilfe einer Datenbank (Formulare, Tabellen und kontrollierte Vokabulare) und in der Wiedergabe des Materials nach unterschiedlichen und spontan bestimmbaren Kriterien (Browsing und Suche). Die üblichen relationalen Datenbanksysteme wie MySQL oder auch eine XML-Datenbank könnten diese Aufgabe grundsätzlich leicht erfüllen. Zur Herausforderung wird dies erst durch die Verschiedenartigkeit der zu verarbeitenden Schriftzeichen, die je nach Schreibrichtung, Diakritika oder Vokalzeichen unterschiedlich behandelt werden müssen.

Abb. 2. Aristoteles (rechts) als Lehrer Alexander des Großen in einer illuminierten arabischen Handschrift. MS British Library, *Or*. 2784, fol. 96r.

2.2. Das GlossGA

Die webbasierte Datenbank des Projekts *Glossarium Graeco-Arabicum* (kurz: GlossGA), dessen Ziel die Erschließung eines griechisch-arabischen Übersetzungswortschatzes ist, soll aufgrund der genannten Kriterien als Fallbeispiel dienen.[4] Der Fokus des Projekts liegt auf Übersetzungen antiker wissenschaftlicher Literatur, die im Raum von Bagdad zwischen dem 9. und 11. Jahrhundert in einer Epoche intensiver Antikenrezeption entstanden sind (vgl. die Illustration in Abb. 2) und die arabischsprachige Hochkultur nachhaltig geprägt haben.[5]

---

[1] G.R. Crane (Hg.): *Perseus Digital Library* <http://www.perseus.tufts.edu/hopper/> (14.02.2015).

[2] *A Digital Corpus for Graeco-Arabic Studies* <http://www.graeco-arabic-studies.org/> (14.02.2015).

[3] *G2A Web App. Web Application for Literary Computing* <http://g2a.ilc.cnr.it> (14.02.2015).

[4] Eine umfassende Darstellung des Projekts findet sich bei G. Endress - R Arnzen - Y. Arzhanov, "Griechische Wissenschaft in arabischer Sprache. Ein griechisch-arabisches Fachwörterbuch der internationalen Wissensgesellschaft im klassischen Islam", *Studia graeco-arabica* 3 (2013), p. 141-56.

[5] Als kleine Einführung in die arabische Aristoteles-Rezeption empfiehlt sich z. B. der übersichtliche Artikel von

Inhaltlich befasst sich das GlossGA mit *hocharabischen Übersetzungen altgriechischer Texte aus der mittelalterlichen Zeit*. Anhand dieser synthetischen Beschreibung lässt sich die Vielschichtigkeit des Projekts und dessen Relevanz für die oben aufgeworfene Problemstellung aufzeigen. Der Forschungsgegenstand ist zweisprachig: Die Quellen liegen einerseits in *altgriechischer*, andererseits in *arabischer* Schrift vor. Das Altgriechische operiert mit bis zu dreifach *geschichteten Diakritika*, während im Arabischen fakultative *Vokalzeichen* (dem Grundzeichen beigefügte Lesezeichen) auftreten können. Für die Forschungskommunikation bedient man sich des Englischen sowie des Lateinischen, wodurch das *lateinische* Alphabet, welches für Transliterationen um *Diakritika* ergänzt ist, als ein drittes hinzukommt. Insgesamt treffen damit vier Sprachen aufeinander, die drei verschiedene Alphabete in zwei Schreibrichtungen mit bis zu drei Schichten von Diakritika bzw. einer Schicht von Vokalzeichen verwenden.

2.3 Unicode als Grundlage

Eine grundlegende Unterstützung für jede dieser Aufgaben bietet der inzwischen weit verbreitete Unicode-Standard. Die notwendige Voraussetzung ist zunächst, dass alle im Projekt benötigten Zeichen – auch die nicht aktuell, aber möglicherweise zukünftig auftretenden! – in diesem Standard enthalten sind.[6] Ist dies der Fall, muss dieser Standard auch konsequent auf allen Arbeitsebenen unterstützt werden. Das heißt: Alle eingesetzten Komponenten, von der Datenhaltung über die Eingabeschnittstelle bis hin zu den Suchfunktionen und zur Datenausgabe, sollten den Standard lückenlos, im Sinne einer durchgängigen supply line, unterstützen. Dies ist möglichst bereits vor der Implementierung gründlich zu prüfen, und zwar auf allen benötigten Ebenen, da ansonsten Lücken bei der Unterstützung entstehen könnten, die mit individuellen Workarounds (das bedeutet hier: aufwändig, fehleranfällig und kaum nachhaltig) behandelt werden müssten.

Unicode beinhaltet alle im GlossGA benötigten Zeichen, was trotz regelmäßiger Erweiterungen des Standards ein Glücksfall ist, denn viele Projekte benötigen Sondersymbole oder Schriftzeichen, die schlicht noch nicht erfasst wurden. Die gewünschten Softwarekomponenten PHP (>=5.3) und MySQL (>=5.0) unterstützen den Standard durchgehend in Datenverarbeitung, Datenhaltung und Datenabfrage. Auch unterschiedliche Kollationen, die beim Zeichenvergleich eine zentrale Rolle spielen, werden unterstützt.[7] Für die Eingabe und die Ausgabe in HTML gibt es bewährte Methoden seitens der Webserver und der Browser; UTF-8 (*Universal Character Set Translation Format*) hat sich hier als Kodierungsstandard für Unicode etabliert.[8] Insofern ist Unicode (>=6.0) ein günstiger Ausgangspunkt.

Jedoch steckt der Teufel im Detail: Während der etwa fünfjährigen, stufenweise verlaufenen Implementierung und Weiterentwicklung des GlossGA tauchte eine Reihe von Tücken auf, an denen sich sowohl entscheidende Stärken als auch einige Schwächen des Unicode-Standards offenbarten. Betroffene Bereiche waren:

M. Lausberg, "Arabische Philosophie im Mittelalter", *Tabula Rasa* 41 (Juli 2010) <http://www.tabvlarasa.de/41/Lausberg1.php> (14.02.2015).

[6] Ansonsten kann das private use area innerhalb von Unicode verwendet werden, wie es z. B. bei Junicode geschieht; vgl. *Junicode* <http://junicode.sourceforge.net> (14.02.2015). Dies ist jedoch, sofern nicht bereits eine Lösung vorhanden ist, eine aufwändige Angelegenheit.

[7] Der Begriff existiert auch in der Philologie, wird dort jedoch anders verwendet: Dort bezeichnet Kollation den Vergleich mehrerer Zeugen eines Textes, während die Informationstechnik auf die Differenzierung zwischen Graphemen (Schriftzeichen) und Allographen (deren Erscheinungsformen) und zielt.

[8] Vgl. *UTF-8, a Transformation Format of ISO 10646* <http://tools.ietf.org/html/rfc3629> (14.02.2015).

(1) Datenbank-Operationen: Wie beeinflusst man die Distinktivität von diakritischen Zeichen oder Vokalzeichen bzw. wie differenziert man die Kollation?
(2) Ausgabe: Wie operiert Unicode bei konkurrierenden Schreibrichtungen und wie kann man dies beeinflussen?
(3) Nutzerschnittstelle: Wie lassen sich die zahlreichen unterschiedlichen Zeichen und Alphabete komfortabel und gleichzeitig fehlerfrei eingeben, so dass die Konsistenz, aber auch die Nutzbarkeit der Datenbank gewahrt bleibt?

2.4 Erschließung des Zeichenvorrates

Der nächste Schritt nach der Überprüfung der Komponenten auf Unicode-Unterstützung sollte eine Inventarisierung des gesamten Zeichenvorrates sein, der im Projekt verwendet werden soll. In Unicode finden wir die im GlossGA benötigten Schriftzeichen in verschiedenen Tabellen, die in der Regel jeweils ein bestimmtes Alphabet oder eine bestimmte Zeichenart abdecken; Sonderzeichen oder ergänzende Zeichen sind oft in zusätzlichen Tabellen abgelegt. Bei der Inventarisierung sollten außerdem die gewünschte Kodierungsmethode sowie die Kollation, die damit in engem Zusammenhang steht, für jedes eingesetzte Alphabet dokumentiert werden. Beides beeinflusst sowohl die Eingabe– als auch Such– und Sortiervorgänge, und zum Teil sogar die Darstellung.

2.4.1 Lateinisch

Das lateinische Alphabet wird im GlossGA für die Projektsprache Englisch und für die Transliteration des Arabischen benötigt. Für die Umschrift wird der Standard der Deutschen Morgenländischen Gesellschaft (DMG)[9] mit minimalen Abweichungen verwendet.[10] Aufgrund ihres Verbreitungsgrades und einer guten allgemeinen Unterstützung stellen die lateinischen Diakritika die geringste Hürde dar. Insgesamt werden alle 26 Buchstaben des lateinischen Alphabets, dazu 14 lateinische Zeichen mit Diakritika und außerdem zwei arabische Zeichen benötigt, was zusammen mit den lateinischen Majuskelvarianten eine Gesamtmenge von 82 Zeichen ergibt.

Eine Problemstellung ergibt sich durch die unterschiedliche Distinktivität der Zeichen. Während im Englischen lediglich zwischen verschiedenen Basiszeichen unterschieden wird (z. B. A ≠ B, aber A = a sowie A = ä), ist die Kollation in der lateinischen Umschrift des Arabischen nicht auf der Ebene des Grundzeichens orientiert, sondern bereits auf der Ebene der Diakritika (a ≠ ā). Zu erwähnen ist außerdem, dass Diakritika in Unicode immer *vor* dem Basiszeichen genannt werden müssen.

2.4.2 Griechisch

Eine Besonderheit des Altgriechischen ist seine Tonalität. Im Gegensatz zu den heutigen westlichen Sprachen, deren Metrik durch betonte und unbetonte Silben gekennzeichnet ist, prägten Tonhöhe und Länge einer Silbe das Altgriechische bis ungefähr ins 3. Jahrhundert. Dies wurde in der Schrift durch diakritische Zeichen festgehalten. Dabei bezeichnete der Akut <´> einen Hochton (*oxia*), das Zirkumflex <˜> einen von oben fallenden Ton auf langen Silben (*perispomenon*), und der Gravis <`>, so vermutet man, einen tiefen, fallenden Ton (*varia*). Ein weiteres Charakteristikum sind Vokale, die ursprünglich Diphthonge waren, und die mit einem *iota subscriptum* gekennzeichnet werden. Dies kann bei <α>, <η> und <ω> auftreten. Auch ein [h] im Anlaut (*psili*) wurde mit einem

[9] Vgl. *Die Transliteration der arabischen Schrift. Denkschrift dem 19. internationalen Orientalistenkongreß in Rom* <http://www.aai.uni-hamburg.de/voror/Material/dmg.pdf> (14.02.2015).
[10] Vgl. *Glossarium Graeco-Arabicum. Transliteration* <http://telota.bbaw.de/glossga/transliteration.php> (14.02.2015).

Diakritikum festgehalten, dem *spiritus asper* <ʽ>, und kann bei allen sieben Vokalen auftreten. Etwas später wurde zusätzlich der *spiritus lenis* <ʼ> für den Anlaut ohne [h] (*dasia*) eingeführt. Alle diese diakritischen Merkmale (Intonation, Diphthong, Anlaut) können auch gemeinsam auftreten, so dass unter Berücksichtigung aller denkbaren Kombinationen und Majuskeln (diese gelten im GlossGA nur für die Grundzeichen und die Grundzeichen mit *iota subscriptum*) eine Menge von 124 Vokalzeichen zusammenkommt (siehe Abb. 3).

Grundzeichen	Diakritikum 1 (Intonation)	Diakritikum 2 (Anlaut)	Diakritikum 3 (Diphthong)	Gesamt
α (*alpha*)	Akut Gravis Zirkumflex	*spiritus asper* *spiritus lenis*	*iota subscriptum*	24
ε (*epsilon*)	Akut Gravis	*spiritus asper* *spiritus lenis*	–	9
η (*êta*)	Akut Gravis Zirkumflex	*spiritus asper* *spiritus lenis*	*iota subscriptum*	24
ι (*iota*)	Akut Gravis Zirkumflex	*spiritus asper* *spiritus lenis*	–	12
ο (*omikron*)	Akut Gravis	*spiritus asper* *spiritus lenis*	–	9
υ (*ypsilon*)	Akut Gravis Zirkumflex	*spiritus asper* *spiritus lenis*	–	12
ω (*ômega*)	Akut Gravis Zirkumflex	*spiritus asper* *spiritus lenis*	*iota subscriptum*	24

Abb. 3: Varianten altgriechischer Vokalzeichen

Da auf einem Grundzeichen bis zu drei Diakritika stehen können, muss für den Computer eine Eingabemethode ausgewählt werden, welche einfach und schnell handhabbar ist. Die Konsonanten des Altgriechischen sind weniger komplex und kommen mit 19 Grundformen und 18 Majuskeln aus. Somit liegen im GlossGA insgesamt 161 Zeichen für das Altgriechische vor.

Hinsichtlich der Kollation besitzen die griechischen Diakritika, im Gegensatz zu den lateinischen Umschriftzeichen, keine distinktive Qualität und können mit dem Grundzeichen kollationiert werden (z. B. Ω = ω = ὦ). Wie im Lateinischen gilt außerdem, dass alle Diakritika vor dem Grundzeichen zu nennen sind.

2.4.3 Arabisch

Im Vergleich zum Altgriechischen erscheint der Zeichenvorrat des Arabischen zunächst etwas übersichtlicher, jedoch birgt auch dieses Alphabet eine gewisse Komplexität.[11] Das

[11] Eine umfassende Darstellung findet sich bei G. Endress, "Die arabische Schrift", in W. Fischer (Hg.), *Grundriss der arabischen Philologie*, Reichert, Wiesbaden 1982, p. 165-97.

arabische Alphabet kennt zunächst 28 Grundzeichen. Jedes Grundzeichen kennt bis zu vier Erscheinungsformen, die von seiner Position innerhalb eines Wortes abhängen.[12] Unterschieden wird zwischen: (1) Wortanfang, (2) Wortende, (3) Wortmitte und (4) alleinstehend (isoliert). In Unicode wird dieses Darstellungsproblem automatisch durch das jeweiligen Betriebssystem gelöst, indem anhand der Umgebung eines Zeichens die richtige Erscheinungsform ermittelt wird. Der Wortsinn wird – bei richtiger Trennung der Zeichen – dadurch nicht verändert. Insgesamt liegen somit 112 Zeichenformen vor, die jedoch nur auf der Ebene des Grundzeichens distinktiv sind. Hinzu kommen die zwei Zusatzzeichen *hamza* <ʾ> und *madda* <ˆ> für den Glottisschlag [ʔ].

Beim Arabischen handelt es sich um ein konsonantisch ausgerichtetes Alphabet. Nur die drei langvokaligen Phoneme [ā ī ū] werden mithilfe der konsonantischen Werte <ʾ y w> dargestellt (z. B. <iy> = [ī]). Dazu kommen beigesetzte Hilfszeichen für die drei Vokalphoneme a, i, u, den Nullvokal (Vokallosigkeit), die Verdoppelung und für den Glottisverschlusslaut im An- und Inlaut. Die orthographischen Hilfszeichen werden nur unter besonderen Umständen verwendet, etwa wenn die Eindeutigkeit der Lesung erschwert oder von besonderem Belang ist. Dies ist z. B. besonders in poetischen oder religiösen Texten, in denen die Differenzierung der Lesarten besonders zum Tragen kommt, der Fall. Die Vokalzeichen sind in ihrer graphischen Erscheinung den Diakritika ähnlich, sind aber aufgrund ihrer phonetischen Bedeutung nicht als Modifikatoren bestehender Zeichen zu verstehen, sondern als eigenwertige Zeichen, die lediglich eine graphische Bindung an Trägerzeichen eingehen. Auch in Unicode wird diese Unterscheidung vollzogen, da die Vokalzeichen im Gegensatz zu den Diakritika dem Trägerzeichen nicht vorausgehen, sondern diesem, ganz der lautlichen Abfolge gemäß, nachgeordnet werden (Beispiel: بـ <b> mit nachgestelltem Damma ـُ <u> wird zu بُـ <bu>). Insofern sind arabische Zeichen mit Vokalzeichen nicht als ein Zeichen aus zwei Komponenten, sondern als zwei separate Zeichen zu behandeln. Da es sich um fakultative Zeichen handelt, dürfen sie auf Seiten der Kollation nicht distinkt behandelt werden, damit die Suche alle Schreibvarianten, sowohl mit als auch ohne Zusatzzeichen, auffinden kann.

Wie auch im Griechischen können durch die zahlreichen Kombinationsmöglichkeiten bei der Eingabe Verwirrungen entstehen, wenn dem Nutzer keine Schnittstelle angeboten wird, welche eine fehlerfreie Eingabe garantiert. Hinzu kommt die Problematik der linksläufigen Schreibrichtung.

2.4.4. Ergebnis

Die Bestandsaufnahme der benötigten Zeichen im GlossGA-Projekt resultiert in einem beachtlichen Katalog von Unicode-Tabellen (siehe Abb. 4). Insgesamt werden etwa 400 Zeichen benötigt (80 lateinische, 161 altgriechische, 120 arabische, dazu allgemeine Zahl- und Interpunktionszeichen). Auf Codierungsebene muss die geforderte Reihenfolge exakt eingehalten werden (Diakritika – Trägerzeichen – Vokalzeichen). Hinsichtlich der Kollation muss auf die unterschiedliche Behandlung der lateinischen Transliterationssymbole, der griechischen Vokale und der arabischen Vokalsymbole geachtet werden. Zudem wird eine für den Nutzer praktikable Eingabemethode benötigt.

[12] Teils zur zwei, da nicht alle Zeichen nach beiden Seiten verbunden werden.

Zeichengruppe	*Unicode-Block*	*Unicode-Bereich*	*Distinktivität*
Lateinisches Alphabet	C0 Controls and Basic Latin[13]	0000-007F	Diakritika: ja
Transliterationssymbole	Latin Extended-A[14]	0100-017F	
Transliterationssymbole	Latin Extended-Additional[15]	1E00-1EFF	
Griechisches Alphabet	Greek and Coptic[16]	0370-03FF	Diakritika: nein
Griechisches Diakritika	Greek Extended[17]	1F00-1FFF	
Arabisches Alphabet	Arabic[18]	0600-06FF	Vokalzeichen: nein
Arabisches Diakritika	Arabic Supplement[19]	0750-077F	
besondere arabische Zeichen	Spacing Modifier Letters[20]	02B0-02FF	

Abb. 4: Benötigte Unicode-Tabellen im GlossGA

3. Eine kurze Geschichte der Zeichencodierung

Wie kam es überhaupt zur Einführung von Unicode, und was hat man zuvor verwendet? Welche wichtigen Alternativen der Zeichencodierung gibt es? Dieser kurze historische Abriss berichtet über die Vorbedingungen von Unicode und über die Weiterentwicklungen bis hin zu Unicode 7.0 (2014).

3.1. Ein erster Meilenstein

Vor einigen Jahrzehnten war der Zeichenvorrat, auf den man als Computeranwender zurückgreifen konnte, noch stark begrenzt. Auf den allerersten Systemen standen üblicherweise nur die lateinischen Buchstaben und die arabischen Zahlen zur Verfügung. Ferner wurden die Zeichen, die man verwenden konnte, allein durch das Betriebssystem vorgegeben. Sofern man nicht über entsprechende Konvertierungsroutinen verfügte, waren Systeme untereinander nicht kompatibel. Erst im Jahr 1969 wurde mit dem *American Standard Code for Information Interchange* (ASCII) ein Vorrat von 128 (27) Zeichen definiert (Abb. 5), der für eine lange Zeit der wichtigste Standard sein

[13] *The Unicode Consortium: C0 Controls and Basic Latin* <http://unicode.org/charts/PDF/U0000.pdf>(14.02.2015).
[14] *The Unicode Consortium: Latin Extended-A* <http://unicode.org/charts/PDF/U0100.pdf>(14.02.2015).
[15] *The Unicode Consortium: Latin Extended-Additional* <http://unicode.org/charts/PDF/U1E00.pdf>(14.02.2015).
[16] *The Unicode Consortium: Greek and Coptic* <http://unicode.org/charts/PDF/U0370.pdf>(14.02.2015).
[17] *The Unicode Consortium: Greek Extended* <http://unicode.org/charts/PDF/U1F00.pdf> (14.02.2015). Die Kombination mehrerer griechischer Zeichen kann wiederum mit verschiedenen Methoden erreicht werden, entweder auf einem bereits kombinierten Basiszeichen und einem Diakritika, oder auf einem Basiszeichen und zwei Diakritika ("full decomposition"). Im *GlossGA* wurde zur Entstehungszeit der Datenbank die erste Methode angewendet, während Unicode jedoch die zweite empfiehlt; vgl. *The Unicode Consortium: Frequently Asked Questions, Greek Language and Script* [<http://unicode.org/faq/greek.html> (19.02.2015)].
[18] *The Unicode Consortium: Arabic* <http://unicode.org/charts/PDF/U0600.pdf> (14.02.2015).
[19] *The Unicode Consortium: Arabic Supplement* <http://unicode.org/charts/PDF/U0750.pdf>(14.02.2015).
[20] *The Unicode Consortium: Spacing Modifier Letters* <http://unicode.org/charts/PDF/U02B0.pdf>(14.02.2015).

sollte.[21] Die weitreichende Akzeptanz des ASCII legte eine wichtige Basis für den Datenaustausch zwischen verschiedenen Systemen.

Der Zeichenvorrat des ASCII deckte das lateinische Alphabet in Minuskeln und Majuskeln, die arabischen Zahlzeichen und eine Reihe geläufiger Interpunktionszeichen ab, was für die ersten Anwendungen zunächst vollkommen genügte. Spezialisierte oder international ausgerichtete Vorhaben, die weitere Zeichen benötigten, griffen häufig auf die Möglichkeit zurück, den Zeichenvorrat durch die Modifikation der Schriftsätze zu verändern. So konnten – unter Verzicht auf einige der vordefinierten Schriftzeichen, vorzugsweise der Minuskeln – andere Symbole oder Schriftzeichen verwendet werden, so dass der Zeichenvorrat den eigenen Anforderungen genügte, allerdings immer unter der Begrenzung auf 128 Zeichen. Diese Grenze war schnell ausgereizt, da ein Viertel des ASCII aus Steuerzeichen besteht, so dass praktisch nur 96 Zeichen für die eigene Verwendung modifiziert werden können. Möchte man nur die Grundzeichen des lateinischen, griechischen und arabischen Alphabets und Ziffern darstellen, benötigt man bereits 90 Zeichen; gerade einmal 6 Zeichen blieben für Interpunktion bzw. Diakritika übrig. Ein weiterer Nachteil der Schriftarten-Modifizierung lag darin, dass mit den reinen Textdaten auch gleichzeitig der passende Schriftsatz übergeben werden musste, damit der Text überhaupt auf anderen Systemen lesbar blieb, oder dass die Daten nur mit ganz spezifischen Programmen verwendet werden konnten.

Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex		Dec	Hex	
0	00	NUL	16	10	DLE	32	20		48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	EOT	20	14	DC4	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	BS	24	18	CAN	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	HT	25	19	EM	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

Abb. 5. ASCII-Tabelle mit Dezimal- und Hexadezimalwerten sowie Zeichen und Steuerzeichen.

Jedoch war die Zeit von ASCII damit nicht vorüber. Eine effektive Nutzungsmethode für andere Alphabete, die auch heute noch gerne angewendet wird, besteht in Pseudocodierungen, bei der nicht vorhandene Zeichen durch graphisch oder phonetisch ähnliche Zeichen ersetzt werden, z. B. ein griechisches <τ> (Tau) durch ein <t>, oder das russische <Ш> durch ein <W>. Für Diakritika werden Hilfszeichen hinzugenommen, wie es z. B. in *Betacode*[22] praktiziert wird. Einen ähnlichen Ansatz verfolgt auch das phonetisch ausgerichtete SAMPA.[23] Mit dieser Methode fällt zum einen die Begrenzung und zum anderen das Darstellungsproblem weg.

---

[21] *American Standard Code for Information Interchange* <http://tools.ietf.org/html/rfc20> (14.02.2015). ASCII ist die US-Variante des *International Alphabet No. 5* (IA5), welches in ISO 646 für verschiedene Sprachen definiert ist.

[22] Vgl. M., Neuhold, *Kodierungen des Griechischen*, darin: *Betacode* <http://members.aon.at/neuhold/antike/grkencs.html#betacode> (14.02.2015). Auch die *Perseus-Digital Library* <http://www.perseus.tufts.edu/hopper/> (14.02.2015) verwendet eine derartige Umschrift.

[23] SAMPA: Speech Assessment Methods Phonetic Alphabet. Siehe dazu J.C. Wells, "*SAMPA* Computer-readable Phonetic Alphabet", in D. Gibbon - R. Moore - R. Winski (eds.), *Handbook of Standards and Resources for Spoken Language Systems*, Mouton - De Gruyter, Berlin - New York, 1997.

3.2. Ein Schritt in die Vielfalt

Die Unterstützung für eine Reihe von Sprachen, die mit dem unveränderten lateinischen Basisalphabet nicht darstellbar sind, darunter zunächst vor allem europäische Sprachen, brachte ISO 8859. Die Zeichenmenge wurde auf 256 (28) erweitert, unter Beibehaltung der bereits durch den ASCII definierten Zeichen (0-127), welche durch einen Sonderzeichenbereich erweitert wurden (128-255), von denen die letzten 96 für weitere Zeichen nutzbar waren. Die Verdoppelung der möglichen Zeichenmenge und die zusätzliche Auffächerung in 15 verschiedene Teilnormen ermöglichte es also, das lateinische Alphabet und eine Auswahl darüber hinausgehender Sonderzeichen zu verwenden. So konnte ISO 8859 u. a. Unterstützung für das Arabische (ISO 8859-6)[24] und das (Neu-)Griechische (ISO 8859-7)[25] bieten.

Das Problem der begrenzten Zeichenmenge blieb dennoch bestehen. Beispielsweise waren viele Zeichenkombinationen des Altgriechischen durch diesen Standard längst nicht abgedeckt. Auch wenn der neue Standard einen erweiterten, flexibleren Vorrat erlaubte, blieb die Auswahl innerhalb eines Dokuments immer noch eingeschränkt, denn es gab keine Möglichkeit, unterschiedliche 8859-Kodierungen innerhalb eines Dokuments parallel zu verwenden, ganz abgesehen von der Möglichkeit, Sonderformen mit Diakritika und Vokalzeichen zu bilden. Somit behalf man sich weiterhin mit den schon zur Zeit des ASCII genutzten Methoden.

3.3. Kodierung und Typographie in Konkurrenz

Ein Problem, welches durch die wachsenden Bedürfnisse an Zeichendarstellungen nun immer häufiger zum Tragen kam, war eine zunehmende Beliebigkeit der Zuordnung von Kodierungen zu dargestellten Zeichen. Grundsätzlich steckt hinter einer Zeichenkodierung der Gedanke, dass ein Zahlencode einem exakt definierten Zeichen entspricht, z. B. "97 = lateinischer Buchstabe a". Die visuelle Darstellung findet hingegen erst durch die Anwendung einer Schrifttype statt, womit der Code "97" als <a>, <a> oder auch als <ɑ> dargestellt werden kann. Mit der Weiterentwicklung der Computertypographie und Textprozessoren konnte man bald auch innerhalb eines Dokuments die Schrifttype wechseln, z. B. von einer Serifenschrift zu einer Groteskschrift. Es wurde gängig, spezielle Schrifttypen für die Darstellung von z. B. mathematischen Symbolen oder technischen Zeichen zu verwenden, und auch andere Alphabete wurden durch Schrifttypen simuliert. In der Folgezeit erfuhren insbesondere die "Truetype"-Schriftarten eine wahre Blütezeit und reichten bis hin zur Darstellung von Wettersymbolen, Dominosteinen und Eisenbahnwaggons.

Mit dieser Praxis verlagerte sich jedoch die Bindung der Zeichenbedeutung weg von der zugrundeliegenden Definition hin zu der jeweils im Einzelfall darstellenden Schrifttype, die mehr oder weniger beliebig und kaum standardisiert war. Ohne die Kenntnis der verwendeten Schrifttype konnte der Code "97" alles mögliche bedeuten, nämlich nicht nur verschiedene Formen von "a", sondern auch alles, was in dem gerade verwendeten Schriftsatz auf dem Code 97 lag (z. B. ist das in der Schrift "Wingdings" das astrologische Symbol <a>). Dass dies auf lange Sicht eine sehr ungünstige Vermischung von Datenebene und Darstellungsebene bedeutete, erschien dem zweckorientierten Anwender zunächst unerheblich. Nachteile erkannte man jedoch schnell bei Weitergabe oder

---

[24] Eine unabhängige, kurze Darstellung findet sich bei R. Czyborra, *The ISO 8859 Alphabet Soup*, darin: *ISO-8859-6 (Arabic)*, <http://czyborra.com/charsets/iso8859.html#ISO-8859-6> (14.02.2015).

[25] Vgl. R. Czyborra, *The ISO 8859 Alphabet Soup*, darin: *ISO-8859-7 (Greek)*, <http://czyborra.com/charsets/iso8859.html#ISO-8859-7> (14.02.2015).

Migration von Dokumenten, die mit seltenen bzw. nicht standardisierten Schriftarten versehen waren, da die Schriftarten üblicherweise nie mit dem Dokument zusammen gespeichert wurden (Ausnahme: PDF), und damit die Möglichkeit einer sinnhaften Darstellung bis auf weiteres verloren war. Es konnte vorkommen, dass der Text aufgrund einer fehlenden oder ausgetauschten Schriftart überhaupt nicht mehr lesbar war oder sich inhaltlich veränderte. Häufig geschah dies z. B. bei der Anwendung von "Wingdings"-Symbolen, die auf Systemen ohne diese Schriftart als mehr oder weniger beliebige Buchstabenketten erscheinen.[26] Aber auch arabische oder griechische Zeichensätze existierten in zahlreichen Variationen und verursachten entsprechendes Durcheinander. Die Konvertierung dieser Dokumente in aktuelle Formate sollte die Fachwelt noch einige Jahre beschäftigen.[27]

Das Zeichenuniversum

Einen weitreichenden Lösungsansatz hinsichtlich der Begrenzung sowie der Darstellung bot erstmals der heute weit verbreitete Unicode-Standard (ISO 10646),[28] der seit 1991 existiert und seitdem regelmäßig erweitert wird.[29] Unicode ordnet jedem sinntragenden Zeichen (sowohl alphabetischen als auch symbolartigen) einen digitalen Code zu. Die theoretische Grenze lag zunächst bei 65.536 (216) Zeichen und wurde schon sehr bald auf 1.114.112 (216+220) mögliche Zeichen angehoben.

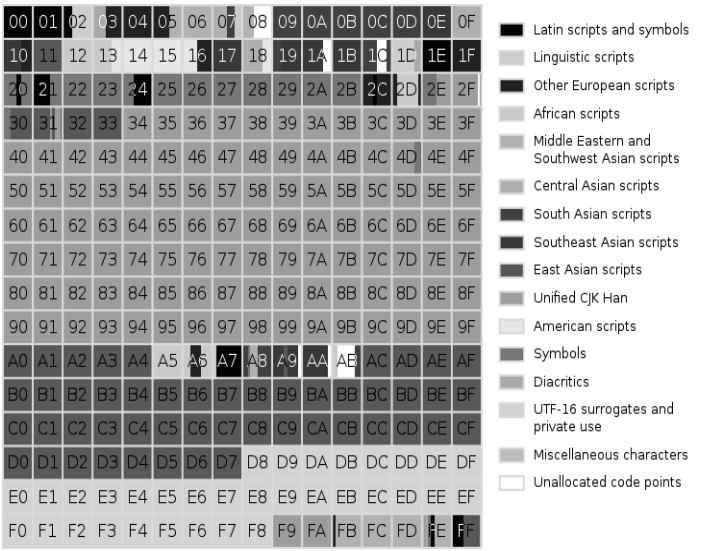


Abb. 6. Übersicht der 256 Unicode-Blöcke. Quelle: Wikimedia, *Roadmap to Unicode* <https://commons.wikimedia.org/wiki/File:Roadmap_to_Unicode_BMP_de.svg> (14.02.2015).

Unicode 7.0 umfasst eine Menge von 113.021 Zeichen. Einzelne Alphabete und Symbolschriften sind in sogenannten Unicode-Tabellen zusammengefasst und stehen in jeweils abgegrenzten Wertebereichen (Unicode-Blöcke, siehe Abb. 6). Darunter sind inzwischen auch einige aus TrueType-Schriftarten hervorgegangene Zeichen enthalten. Durch den Unicode-Standard wurde die Koppelung

[26] Da es sich bei Wingdings um einen weit verbreiteten Schriftsatz handelt, hat man viele der darin enthaltenen Zeichen in die Unicode-Tabelle Dingbats aufgenommen, vgl. <http://www.unicode.org/charts/PDF/U2700.pdf> (14.02.2015).

[27] Knut S. Vikør hat sich dieser Problematik angenommen und Conversion Macros für arabische Schriften entwickelt, darunter IslwTimes, TimesBeyroutRoman u. v. a.; vgl. *Macros for Converting Old Documents to Unicode* <http://www.smi.uib.no/ksv/macros.html> (14.02.2015).

[28] Im Januar 2015 wurde UTF-8 von 82,4% aller Websites verwendet (ISO-8859-1: 9,2%), während alle anderen Standards im Web seit Jahren an Bedeutung verlieren; in Januar 2010 waren es noch 50,6% (ISO-8859-1: 28,6%). Vgl. <http://w3techs.com/technologies/history_overview/character_encoding/ms/y> (14.02.2015).

[29] Unicode wird in über 80% aller Am 16. Juni 2014 wurde Unicode 7.0 veröffentlicht; vgl. <http://www.unicode.org/versions/Unicode7.0.0/> (14.02.2015).

von Zeichencodierung an die Zeichenbedeutung wieder gefestigt und die Bindung an individuelle Schriftarten deutlich gelöst. Prinzipiell ermöglicht Unicode eine von der jeweiligen Schriftart unabhängige Repräsentation von fast beliebigen Zeichen. Einschränkungen bestehen in den Möglichkeiten einzelner Schrifttypen, die gewünschten Unicode-Zeichen darzustellen; jedoch bleibt die Bedeutung aufgrund der darstellungsunabhängigen Definition prinzipiell nachvollziehbar.

4. Einzelbeispiele

Drei Facetten erscheinen bei der Implementierung des GlossGA von besonderer Relevanz für die hier angesprochenen Fragestellungen. (1) Das Kollationieren von Zeichen: Hier greifen Unicode-Kodierungsmethoden, Unterstützung durch das Datenbanksystem und wissenschaftlicher Zweck ineinander und müssen aufeinander abgestimmt werden. (2) Die Behandlung der Laufrichtung, welche ein problematischer Aspekt der Wiedergabe links- und rechtsläufiger Zeichen innerhalb einer Zeile ist. (3) Die Nutzerschnittstellen, welche einerseits eine einfache Eingabe ermöglichen und gleichzeitig für eine konsistente und fehlerfreie Kodierung sorgen sollen.

4.1. Kollationieren

Jede Datenbank muss Such- und Sortierfunktionen anbieten, welche auch die diversen Unterscheidungsgrade der Zeichen berücksichtigt. Hierzu wurde die unterschiedliche Distinktivität der verschiedenen Zeichengruppen festgestellt. Beispielsweise sollen im GlossGA bei der Suche nach <ṉ> auch <ṉ́>, <ṉ̃> und <ṉ̋> gefunden werden, oder bei der Suche nach <سبب> auch <سبّ>, aber im Falle der Umschrift bei der Suche nach <s> eben nicht <ṣ>. In der Informationstechnik dienen Kollationen dazu, die verschiedenen Erscheinungsformen eines Schriftzeichens als identisch zu behandeln. In Unicode existieren dazu Kollationstabellen, mit deren Hilfe die Zeichen in Gruppen zusammengefasst werden, die bei Such- oder Sortiervorgängen als gleichwertig zu behandeln sind. Kollationstabellen existieren sowohl für Latein und Griechisch als auch für Arabisch.[30] Diese sind gegenüber allen anderen Methoden deutlich von Vorteil, da sie an bestehende Standards anknüpfen. Das Datenbanksystem MySQL bietet native Unicode-Unterstützung durch UTF-8 und die entsprechenden Kollationstabellen an und bildet insofern eine ideale Basis.

ω 03C9	ω 1D6DA	ω 1D714	ω 1D74E	ω 1D788	ω 1D7C2	Ω 03A9	Ω 2126	Ω 1D6C0	Ω 1D6FA	Ω 1D734
	Ω 1D76E	Ω 1D7A8	ὠ 1F60	Ὼ 1F68	ᾠ 1F64	Ὼ 1F6C	ᾤ 1FA4	Ὢ 1FAC	ᾨ 1F62	Ὢ 1F6A
	ᾤ 1FA2	Ὢ 1FAA	ᾦ 1F66	Ὦ 1F6E	ᾧ 1FA6	Ὦ 1FAE	ᾡ 1FA0	Ὡ 1FA8	ὡ 1F61	Ὡ 1F69
	ᾠ 1F65	Ὼ 1F6D	ᾤ 1FA5	Ὢ 1FAD	ᾨ 1F63	Ὼ 1F6B	ᾣ 1FA3	Ὣ 1FAB	ᾧ 1F67	Ὦ 1F6F
	ᾧ 1FA7	Ὦ 1FAF	ᾡ 1FA1	Ὡ 1FA9	ὡ 03CE	ὡ 1F7D	Ὡ 038F	Ὡ 1FFB	ᾤ 1FF4	ὠ 1F7C
	Ὡ 1FFA	ᾡ 1FF2	ᾦ 1FF6	ᾧ 1FF7	ῳ 1FF3	Ω 1FFC				

Abb. 7. Unicode Collation Chart für Omega (Quelle: Unicode Consortium)

Die Kollationstabellen operieren mit einer hierarchischen Gewichtung der verschiedenen Varianten eines Zeichens. Im Beispiel (Abb. 7) ist für das griechische Zeichen "Omega" die Gewichtung

[30] The Unicode Consortium: *Collation Charts* <http://www.unicode.org/charts/collation/> (14.02.2015).

anhand von vier verschiedenen Farbschattierungen visualisiert. Wird auf Grundzeichenebene (dunkel) kollationiert, so sind alle Varianten gleichbedeutend, die dem Grundzeichen folgen und heller sind (also alle). Wird hingegen auf der Ebene von Diakritika kollationiert (mittel), so sind auch hier nur die jeweils folgenden und helleren Varianten gleichbedeutend (z. B. ω = Ω, aber ω ≠ ϖ). Die Minuskel- und Majuskelvarianten befinden sich wiederum eine Ebene darunter (helle Schattierung). Danach folgen noch die Schrifttypen-Varianten (weiß).

4.1.1. Anwendung

Im GlossGA genügt für das Altgriechische bereits die Kollation auf der Ebene des Grundzeichens, da alle untergeordneten Varianten keinen Bedeutungsunterschied mehr ausmachen. In MySQL genügt somit die Einstellung "unicode_ci" (*case insensitive*).

Beim Arabischen kommt die Tatsache zum Tragen, dass die Vokalzeichen als eigene, nicht kollationierte Zeichen definiert sind. Ein zusammengesetztes Schriftzeichen, bestehend aus Träger- und Vokalzeichen, kann in Unicode auf zwei Wegen erreicht werden: entweder durch die direkte Zuweisung des bereits kombinierten Zeichens oder durch die Zusammensetzung zweier Zeichen (*decomposition*). Da die bereits kombinierten Zeichen in Unicode alle distinkt kollationiert sind, muss, da hier nur das Trägerzeichen als relevant gelten soll, letztere Methode bevorzugt werden: Beispielsweise ist für *alif* mit *hamza* <أ> U+0627 U+0654 zu präferieren, anstelle von U+0623.[31] Ansonsten genügt in MySQL auch hier die Einstellung "unicode_ci" (*case insensitive*).

Bei den lateinischen Transliterationszeichen muss hingegen die Kollation auf der Ebene der Diakritika erfolgen, was in MySQL durch die Einstellung "unicode_bin" (*binary*) erreicht werden kann; dies ist ein sehr restriktives Vorgehen, erfüllt jedoch den Zweck, solange die Zeichen der Umschrift alle untereinander distinktiv behandelt werden sollen. Sollen aber einzelne Zeichen kollationiert werden, muss ggf. eine eigene Kollation erstellt werden.

4.2. Laufrichtung

Abgesehen von wenigen Ausnahmen bevorzugt jedes standardisierte Alphabet eine bestimmte Laufrichtung. In den vorwiegend durch das Lateinische geprägten westlichen Sprachen verläuft diese von links nach rechts (rechtsläufig), in der arabischen Schrift – im Gebrauch für die Sprachen Arabisch, Persisch, Urdu u. a. – und weiteren Schriftsystemen semitischer Sprachen (Hebräisch, Syrisch-Aramäisch) von rechts nach links (linksläufig), in asiatischen Sprachen auch mal von oben nach unten. Computersysteme sind seit ihrer Entstehung auf rechtsläufige Darstellung ausgerichtet worden. Erst nach und nach wurden einige Textverarbeitungsprogramme in die Lage versetzt, auch linksläufige Texte darstellen zu können. Anfangs galt eine Richtungsvorgabe für einen ganzen Text oder Absatz. Ein Wechsel der Laufrichtung innerhalb einer Zeile stellte hingegen noch lange eine Herausforderung dar. Sollte man die Zeichen entsprechend der Erscheinungsreihenfolge oder der Lesereihenfolge codieren? Ersteres macht das Auslesen andersläufiger Wörter unmöglich, da diese rückwärts kodiert würden; letzteres bereitet Schwierigkeiten bei der Wiedergabe, da nicht automatisch klar ist, wann sich die Laufrichtung ändert.

Lösen ließ sich dieses Problem erst mit der Kopplung des einzelnen Zeichens an seine Laufrichtung, wie sie in Unicode umgesetzt wurde. So konnte innerhalb eines Textes die Laufrichtung beliebig von

---

[31] In Datenbanksystemen, die ggf. die Kollation des Arabischen nicht unterstützen, würde man für Such- und Sortierfunktionen alle Datensätze einmal zusätzlich, aber in einer von Diakritika und Vokalzeichen bereinigten Version in der Datenbank ablegen (ähnlich würde man auch bei ASCII-basierten Alternativen wie z. B. Betacode verfahren können).

links nach rechts und wieder zurückspringen (siehe Beispiel in Abb. 8). Nur Interpunktionszeichen und einige andere Hilfszeichen besitzen keine bestimmte Laufrichtung und können in mehreren Laufrichtungen verwendet werden. Diese nennt man *weak characters*, da sie sich an die Laufrichtung der alphabetischen Zeichen, die man *strong characters* nennt, da sie die Laufrichtung bestimmen, anpassen.

φ	ο	ρ	ά		ا	م	ع	ا	ن		*m*	*a*	*ṣ*	*d*	*a*	*r*		m	ʿ	n		I	V
L	L	L	L	WS	R	R	R	R	R	WS	L	L	L	L	L	L	WS	L	L	L	WS	L	L
→	→	→	→		←	←	←	←	←		→	→	→	→	→	→		→	→	→		→	→

Abb. 8. Kodierungsbeispiel mit BIDI-Werten (L = LTR, R = RTL, WS = Whitespace).
Wiedergabe: φορά امعان *maṣdar mʿn* IV.

Dieses Verfahren ist so einfach wie genial, jedoch birgt es für den Anwender einige Tücken, die es zu kennen gilt. Die größten Schwierigkeiten birgt der Umgang mit *weak characters*, da deren Verhalten nicht intuitiv vorhersehbar ist, sondern nur, wenn man mit dem von Unicode vorgeschlagenen Algorithmus für bidirektionalen Text vertraut ist.[32] So muss man in vielen Fällen erst recherchieren, welches Zeichenverhalten eigentlich erwartbar ist. Ferner wird der Algorithmus nicht immer optimal implementiert, und durch eine Änderung in Unicode 6.3 kam es zu einer Unterstützungslücke in einigen Anwendungen.

4.2.1. Trunkierungszeichen

Ein kritisches Phänomen im Umgang mit nicht-alphabetischen Zeichen tritt beispielsweise in Suchformularen auf. Hier wird dem Benutzer zur Trunkierung gerne der Asterisk <*> angeboten. Angenommen, der Benutzer möchte nach Wörtern beginnend mit <مل> suchen, so würde die Suche z. B. ملح (Salz), ملك (König) oder auch ملك (Engel) ergeben. Wird der Wortanfang mit dem Trunkierungszeichen eingegeben, sieht es wie folgt aus:

Search for: مل*

Man würde erwarten, dass der Stern am Ende der Zeile in Leserichtung steht – in diesem Falle also am *linken* Rand. Der Asterisk[33] bekam aber die Laufrichtung des rechtsläufigen Suchfeldes zugeordnet, da es sich hier um ein "neutrales" Zeichen handelt, das die Schreibrichtung des Haupttextes erhält, sofern es sich – wie bei Trunkierungszeichen unvermeidbar – am Rand befindet. (Würden weitere arabische Zeichen folgen, spränge der Asterisk in die Mitte). Behelfen kann man sich, indem man entweder das Suchfeld rechtsläufig formatiert – oder für die Trunkierung arabischer Eingaben den linksläufigen Stern (U+066D)[34] anbietet.

---

[32] M. Davis, *Unicode Standard Annex #9. Unicode Bidirectional Algorithm* <http://www.unicode.org/reports/tr9/>(14.02.2015).

[33] FileFormat.Info: *Unicode Character 'ASTERISK' (U+002A)* <http://www.fileformat.info/info/unicode/char/002a>(14.02.2015). Der Asterisk Operator, welcher zur Differenzierung zwischen dem mathematischen Symbol und dem Textzeichen eingeführt wurde und identisch aussieht, verhält sich genauso; vgl. FileFormat.Info: *Unicode Character 'ASTERISK OPERATOR' (U+2217)* <http://www.fileformat.info/info/unicode/char/2217> (14.02.2015).

[34] FileFormat.Info: *Unicode Character, 'ARABIC FIVE POINTED STAR' (U+066D)* <http://www.fileformat.info/info/unicode/char/066d/>(14.02.2015).

Search for: *مل

Ebenso verhält es sich mit dem Fragezeichen <?>, welches manchmal als Jokerzeichen (für genau ein beliebiges Zeichen) angeboten wird. An seiner Stelle kann ebenfalls das linksläufige <؟> eingesetzt werden.

Bei der Implementierung der Suchfunktion ist zu berücksichtigen, dass die "westlichen" Zeichen < * > und < ? > nicht mit den arabischen Zeichen < * > und < ؟ > kollationiert sind, und man daher beide Zeichen einzeln behandeln muss.[35] Dies bedeutet einen leicht höheren Implementierungsaufwand; jedoch sollten ohnehin bereits aus Sicherheitsgründen sämtliche Platzhalterzeichen des Datenbanksystems gesondert behandelt und überprüft werden. Was bisher fehlt, ist eine *best practice* für den Einsatz von Trunkierungs- und Jokerzeichen, welche die Perspektive der Nutzer und die Voraussetzungen der Informationstechnik gleichermaßen berücksichtigt.

4.2.2. Breadcrumbs

Ein weiteres, sehr anschauliches Beispiel sind sogenannte Breadcrumb-Leisten, welche spätestens seit *Yahoo!* als typisches Navigationselement auf vielen Webseiten auftauchen. Sie halten den Weg von der Startseite bis zur jeweiligen Unterseite fest und ermöglichen es dem Nutzer, zu einer beliebigen Seite auf diesem Weg zurückzuspringen – im Sinne einer Spur aus Brotkrumen wie in dem bekannten Märchen der Gebrüder Grimm. Die einzelnen Elemente werden üblicherweise durch Pfeile oder Größer-als-Zeichen voneinander abgetrennt, wodurch der Eindruck eines Pfades entstehen soll.

Zwei Hürden sind bei der digitalen Umsetzung zu nehmen, wenn mehrere Schreibrichtungen zusammenkommen. Man nehme als Beispiel den folgenden Pfad an: Die Startseite des GlossGA ("Home") führt auf ein Glossar mit arabischen Wörtern ("Arabic Glossary"). Von dort aus geht es über die Auswahl des Buchstabens *ṣād* <ص> zum arabischen Wort für *Gesundheit* <صحة>. In der Breadcrumb-Darstellung sollen diese Elemente jetzt durch einen Pfeil <?> getrennt werden. Was wir erwarten, sähe wie folgt aus:

Home → Arabic Glossary → ص → صحة

Das tatsächliche Ergebnis sieht jedoch zunächst so aus:

Home → Arabic Glossary → صحة → ص

Scheinbar steht das *ṣād* an der falschen Stelle, obwohl der Text in der richtigen Reihenfolge eingegeben wurde. Dabei ereignete sich folgendes: Das *ṣād* drehte die Schreibrichung nach links. Der darauffolgende Pfeil übernahm in seiner Rolle als *weak character* die Schreibrichtung, anstatt sie nach rechts zurückzudrehen. Daher steht in der Ausgabe <ص> rechts von <صحة>.

Das zweite Phänomen tritt zutage, wenn wir statt eines Pfeils ein Größer-als-Zeichen verwenden:

Home > Arabic Glossary > صحة < ص

---

[35] Um eine MySQL-Abfrage in PHP vorzubereiten, kann man z. B. *str_replace()* einsetzen. Dabei kann man auch gleich die von MySQL verwendeten Trunkierungszeichen herausfiltern: str_replace (array ('%', '_', '*', '؟','?','⋆'), array ("", '%', '%', "", '_', '_'), $string);

Das Größer-als-Zeichen behält in diesem Fall nicht nur die arabische Schreibrichtung bei, sondern dreht sich selbst in die Richtung der vorherrschenden Schreibrichtung. Das Größer-als-Zeichen könnte man aufgrund dieses Phänomens als einen "very weak character" bezeichnen, da es offenbar nicht nur dazu bereit ist, die Schreibrichtung der Umgebung anzunehmen, sondern dabei sogar seine Erscheinungsform entsprechend anzupassen.

Für die Laufrichtungs-Problematik gibt es Lösungsansätze auf verschiedensten Ebenen:

(1) Man legt in HTML die Schreibrichtung im umspannenden Element durch z. B. (ltr = left-to-right = rechtsläufig) oder (rtl = right-to-left = linksläufig) selbst fest. Wird der Text dann allerdings in eine Nicht-HTML-Umgebung übertragen, bleibt die Schreibrichtung nicht erhalten.

(2) Man verwendet einen *strong character* als Breadcrumb-Zeichen. Ohnehin ist das Größer-als-Zeichen[36] streng genommen als Breadcrumb-Zeiger ungeeignet, da es sich laut Unicode-Definition um einen mathematischen Vergleichsoperator handelt. Die alternativ in Betracht kommenden Zeichen, z. B. aus dem Block *Geometric Shapes*[37], sind jedoch allesamt *weak characters*, weshalb diese Methode mit dem folgenden Ansatz verbunden werden muss.

(3) Man fügt vor und nach dem Breadcrumb-Zeichen ein "starkes Leerzeichen" ein, das eine feste Schreibrichtung besitzt. Solche Leerzeichen nennen sich *pseudo strong characters* und tragen die Unicodes U+200E (rechtsläufig) oder U+200F (linksläufig).[38] Diese Lösung hat gegenüber der erstgenannten den entscheidenden Vorteil, dass die Schreibrichtung unabhängig von HTML in jeder Unicode-konformen Umgebung erhalten bleibt und dass es keine Einschränkung hinsichtlich der verwendbaren Zeichen gibt.

Abschließend sei darauf hingewiesen, dass dieses Problem nicht nur in Breadcrumb-Leisten, sondern in allen möglichen anderen Kontexten auftauchen kann, in denen bidirektionale Texte lediglich durch *weak characters* getrennt sind. Dies kann z. B. in kommagetrennten Aufzählungen innerhalb von Lexikoneinträgen oder in bibliographischen Angaben der Fall sein. Die denkbaren Lösungsansätze bleiben jedoch die gleichen.

Ein weiterhin ungelöstes Problem liegt in er mangelnden Transparenz bei der Arbeit mit *pseudo strong characters*. Sie werden nicht auf der üblichen Tastatur angeboten und müssen mit Hilfsmitteln eingebunden werden. Ferner sind sie ebenso wie gewöhnliche Leerzeichen unsichtbar und werden auch nicht behelfsmäßig visualisiert, so dass bei der Handhabung stets Unklarheiten entstehen können. Wie genau das Verhalten von Schreibrichtungen in individuellen Umgebungen funktioniert, ist ebenfalls kaum transparent. Hier besteht Bedarf an stärker praxisorientierten Visualisierungsmethoden, die diese Vorgänge nachvollziehbarer machen.

---

[36] FileFormat.Info: *Unicode Character 'GREATER-THAN SIGN' (U+003E)* <http://www.fileformat.info/info/unicode/char/003E/> (14.02.2015).

[37] *Unicode Standard, Version 6.0, 25A0-25FF* <http://www.unicode.org/charts/PDF/U25A0.pdf> (14.02.2015).

[38] FileFormat.Info: *Unicode Character 'LEFT-TO-RIGHT MARK' (U+200E)* <http://www.fileformat.info/info/unicode/char/200E/> (14.02.2015); *Unicode Character 'RIGHT-TO-LEFT MARK' (U+200F)* <http://www.fileformat.info/info/unicode/char/200F/> (14.02.2015).

4.3. Nutzerschnittstellen

4.3.1. Sein vs. Schein

An der Schnittstelle zwischen Mensch und Maschine ereignen sich in der Regel die meisten Irrtümer. So verhält es sich auch bei Unicode-Zeichen. Ein grundlegender Irrtum besteht beispielsweise darin, dass die menschenlesbare Darstellung eines Zeichens mit seiner maschinenlesbaren Darstellung übereinstimme. Dies ist grundsätzlich nicht der Fall. Daher können in Zeichenkodierungen große Schwierigkeiten durch die optische Verwechslung von Zeichen auftreten. Beispielsweise stellt sich bei folgenden Zeichen leicht Verwirrung ein:

ʾ ʿ

ʾ ʿ

Man mag nun mutmaßen, dass es sich um zwei Zeichen handelt: einen nach links geöffneten und einen nach rechts geöffneten Halbkreis. Diese entsprechen den arabischen Schriftzeichen *hamza* und *ʿain* und unterscheiden sich scheinbar nur in ihrer vertikalen Position. Ein Bedeutungsunterschied ist dem Betrachter nicht ersichtlich, denn dem Augenschein nach handelt es sich um eine typographisch bedingte Verschiebung ein und desselben Zeichens. Konsultiert man die Unicode-Tabellen, findet man aber die vier Zeichen mit folgenden Beschreibungen:

ʾ	U+02BE	MODIFIER LETTER RIGHT HALF RING transliteration of Arabic *hamza* (glottal stop)
ʾ	U+02D2	MODIFIER LETTER CENTRED RIGHT HALF RING more rounded articulation [nicht = *hamza*]
ʿ	U+02BF	MODIFIER LETTER LEFT HALF RING transliteration of Arabic *ʿain* (voiced pharyngeal fricative)
ʿ	U+02D3	MODIFIER LETTER CENTRED LEFT HALF RING less rounded articulation [nicht = *ʿain*]

In Unicode haben diese Zeichen – auch wenn sie sich optisch nur marginal oder gar nicht unterscheiden mögen – in jeder Variante eine festgeschriebene Bedeutung. Die Zeichencodes sind also zweifach gebunden: an die Bedeutung *und* an die Darstellung. Im vorliegenden Fall, für die Darstellung von *hamza* und *ʿain*, sind daher nicht U+02D2 und U+02D3, die offensichtlich für phonetische Notation vorgesehen sind, zu verwenden, sondern U+02BE und U+02BF.[39]

Der Verwechslung durch den Nutzer beugt dies allerdings nicht vor, da dieser in der Regel keine Unicode-Tabellen konsultieren wird, sondern die benötigten Zeichen dem Augenschein nach auswählt. Es kann und wird somit vorkommen, dass gleich oder ähnlich aussehende Zeichen, die verschiedene Bedeutungen tragen, miteinander vermengt werden. Ist dies erst eingetreten,

[39] Häufig, wegen Lücken im verwendeten Zeichensatz, ist auch die Verwechslung von ǧ <ğ> (U+01E7) mit <ğ> (U+011F), dem aspirierten g (*yumuşak g*) des Türkischen.

kann dies zu Verwirrungen führen, deren Ursache möglicherweise erst spät erkannt wird, und deren Auswirkungen auf Nutzerschnittstellen, Datenhaltung und Durchsuchbarkeit sowie auf die Forschungsergebnisse gravierend sind. Auch die oben besprochenen unterschiedlichen Kodierungsmethoden für zusammengesetzte Zeichen spielen eine wichtige Rolle; gegebenenfalls wirkt sich eine Mischung aus verschiedenen Methoden (Komposition und Dekomposition) negativ auf die Qualität der Ergebnisse aus, ohne dass dies bemerkt wird.

4.3.2. Lösungen

Dem Problem der Verwechselbarkeit von Zeichen ist am besten gleich zu Beginn eines Projekts zu begegnen. Aufgrund der unerschöpflichen Unicode-Zeichenmenge empfiehlt es sich, den benutzbaren Zeichenvorrat von vornherein einzuschränken. Der Zeichenvorrat ist in den Unicode-Tabellen genau zu recherchieren; leicht zu verwechselnde Zeichen sollten nach Möglichkeit ausgeschlossen werden.

Im nächsten Schritt sollte klar definiert werden, auf welchem Wege der Nutzer Zeichen einzugeben hat, und auf welchem Wege der definierte Zeichenvorrat zur Verfügung gestellt wird. Dabei stehen verschiedene Möglichkeiten zur Debatte. Maßgeblich ist die Sicherheit, dass der Nutzer nur Zeichen eingibt, die sich in der Zeichenmenge des Projekts befinden.

(1) Naheliegend erscheint zunächst das Umschalten des Tastaturlayouts über das Betriebssystem, wodurch im Prinzip jedes Alphabet verwendet werden kann. Allerdings ist die Einstellung nutzerabhängig, womit wiederum durch eine falsche Einstellung fehlerhafte Zeichen eingegeben werden könnten; ferner ist der Zeichenvorrat von dem gewählten Tastaturlayout abhängig. Zudem ist es möglich, dass kein standardisiertes Layout den gewünschten Zeichenvorrat komplett abdeckt, was eine Anpassung des Nutzersystems zur Folge haben müsste.

(2) Alternativ dazu bietet sich der Einsatz einer Bildschirm-Tastatur an. Der Vorteil dabei ist, dass der Zeichenvorrat gezielt auf die benötigten Zeichen begrenzt werden kann. Dadurch braucht der Nutzer keine besonderen Tastatureinstellungen vornehmen und wird auch nicht zur Verwendung von Sonderzeichen-Programmen genötigt. Nachteilig ist, dass durch die Maussteuerung kein allzu hohes Eingabetempo erreicht werden kann.

(3) Als letzte Alternative kann dem Nutzer ein Pseudoalphabet angeboten werden, welches dann entsprechend nach UTF-8 zu konvertieren wäre. Der Nutzer muss das Pseudoalphabet erlernen, kann dann aber effektiv und ohne Anpassungen arbeiten. Für das Griechische existiert z. B. das oben bereits erwähnte Betacode.

Ansatz (3) steht dem ursprünglichen Ansatz, durchgehend UTF-8 zu verwenden, entgegen, und wird daher nicht weiter verfolgt. Für Ansatz (1) existieren mehrere Lösungen, allerdings keine plattformunabhängige. Für Ansatz (2) existieren mehrere Lösungen in JavaScript,[40] allerdings funktionieren diese nicht, wenn JavaScript auf dem Nutzergerät nicht verfügbar ist. Denkbar ist auch eine kombinierte Lösung z. B. aus den Ansätzen (1) und (2). Zudem fällt bei textintensiven Projekten die Eingabegeschwindigkeit ins Gewicht.

Darüber hinaus sollte auch immer durch das Datenbanksystem selbst sichergestellt werden, dass keine Zeichen außerhalb der Projektdefinition eingegeben werden können.

Für das GlossGA wurde eine Bildschirmtastatur entwickelt, welche je nach Kontext ein anderes Layout anbietet. Um die Anzahl der benötigten Klicks zu reduzieren, wurden Basiszeichen und

[40] Die im GlossGA verwendete Lösung ist (da sie flexibel anpassbar ist) ist eine virtuelle Tastatur auf dem Bildschirm, welche von Brian Huisman entwickelt wurde. Siehe *JavaScript Graphical / Virtual Keyboard Interface* <http://www.greywyvern.com/code/javascript/keyboard> (14.02.2015).

Diakritika vorgruppiert. Die Diakritika-Gruppen 1 und 2 werden als Vorauswahl-Tasten definiert (insgesamt 11). Als Basiszeichen dienen zusätzlich zu den Vokalen auch die Vokale mit iota subscriptum (insgesamt 10). Auf diese Weise konnte erreicht werden, dass der Nutzer – gleichgültig, wie viele Diakritika er benötigt – maximal zwei Tasten zu betätigen hat. So ist es möglich, 114 Zeichen mit insgesamt 21 Tasten abzubilden. Da die Anzahl der Vorauswahl-Tasten und die Anzahl der Basiszeichen dicht beieinander liegen, ist anzunehmen, dass sich diese Lösung sehr nahe am Optimum bewegt.

5. Zusammenfassung

Es zeigt sich, dass das Zusammenspiel von Geisteswissenschaft und Informatik souverän funktionieren kann. In dem Unicode-Standard laufen die historisch gewachsenen Strukturen der Informatik und der Schriftgeschichte zusammen: Auf der einen Seite stehen die komplexen Strukturen von mehreren parallel entstandenen Schriftsystemen, während auf der anderen Seite ein immer umfassenderer und zunehmend vielschichtiger Referenzapparat steht. Unicode scheint tatsächlich das Potential zu besitzen, die Bedürfnisse der an historischen Strukturen interessierten Wissenschaft fast vollständig zu befriedigen. Grundsätzlich sind mit den aktuellen Mitteln, welche durch Unicode bereitgestellt werden, alle derzeitigen Anforderungen an das GlossGA erfüllbar.

Ein peripheres, aber dennoch nicht zu unterschätzendes Problem scheint in der noch recht geringen Verbreitung von Unicode-Kompetenz zu liegen. Da bereits die Notwendigkeit zur Unterscheidung zwischen Zeichencode und Zeichendarstellung meistens nicht erkannt wird, handeln viele Nutzer dem visuellen Eindruck nach. Entsprechende Nutzerschnittstellen können den meisten grundlegenden Fehlern vorbeugen, jedoch sollte Kompetenz im Umgang mit Unicode nicht nur auf Seiten der technischen Entwickler vorhanden sein. Hier ist noch Schulungs- und Vermittlungsarbeit zu leisten; etwa könnten Struktur und Logik des Unicode-Standards durch eine entsprechende Plattform, welche den spezifischen Ansprüchen der Geisteswissenschaften stärker entgegenkommt, in der Nutzerschaft verbreitet werden. Ebenso muss aber auch auf der technischen Seite das Bewusstsein für die Feinheiten und Erfordernisse der jeweiligen Schriftsysteme geweckt werden.

Innerhalb des Unicode-Standards besteht möglicherweise noch Spielraum für weitere Optimierung. So bedarf es einer Unterstützung für bestimmte Anwendungsbedürfnisse, z. B. Breadcrumb-Zeichen, die die Hauptrichtung beibehalten. Noch entscheidender wären allgemeine Platzhalterzeichen, die sich der Laufrichtung anpassen können. Da jedoch nicht abzusehen ist, inwieweit sich das Unicode-Konsortium dieser Problematiken annehmen bzw. diese überhaupt als ein Aufgabenfeld betrachten wird, ist der vordringliche Bedarf derzeit vor allem in der Etablierung einer best practise zu sehen.

Die Annäherungsprozesse von Schriftkunde und Informatik, die hier am Beispiel des GlossGA nachvollzogen wurden, sind ein Musterbeispiel für das Zusammenwachsen zweier hochspezialisierter Disziplinen. Es zeigt sich aber auch, dass nur auf der Grundlage vertiefter Kenntnisse in allen beteiligten Disziplinen stabile Arbeitsinstrumente entwickelt werden können. Wenngleich der Erwerb von Grundwissen einen hohen Einarbeitungsaufwand auf beiden Seiten erfordert, zahlt sich dieser aus: Auf lange Sicht wird ein gegenseitiges Verständnis der Disziplinen füreinander zu einem deutlich effektiveren Einsatz der Informationstechnologie in der geisteswissenschaftlichen Forschung führen.