# UNEINDEUTIGKEITEN IN MINIMALISTISCHEN GRAMMATIKEN FÜR ZAHLWORTE – PROBLEME UND LÖSUNGSANSÄTZE

*Mira Schielke[1], Henriette Heinrich[1]*

*[1]MediaInterface GmbH*
*Schielke@mediainterface.de*

**Kurzfassung:** Zahlworte können als Komposita betrachtet werden, die sich nach festen Regeln aus nur wenigen Morphemen zusammensetzen lassen. Daher erscheint die Speicherung ganzer Zahlwörter in einem Lexikon überflüssig. Stattdessen können mithilfe einer minimalistischen Grammatik vollständige Zahlwörter aus den Einzelmorphemen gebildet und ihr Wert abgeleitet werden. Hierbei tauchen jedoch Uneindeutigkeiten auf, die in diesem Beitrag gemeinsam mit Lösungsansätzen vorgestellt werden. Um grammatisch falsche Wörter auszuschließen, werden für das mentale Lexikon differenziertere syntaktische Merkmale vorgeschlagen. Zudem erweisen sich für die Kombination von Zahlen größer 100 zusätzliche Lexikoneinträge als effektiv, die Summenoperationen auslösen können, jedoch kein zugehöriges Morphem besitzen.

## 1 Minimalistische Grammatik für Zahlworte

Zahlworte nehmen eine besonders interessante Rolle in Sprachtechnologien ein, da aus nur wenigen Morphemen vollständige und korrekte Wörter entstehen. Die Regeln zur Bildung sind intuitiv leicht zu erfassen. Formal umgesetzt kann so beispielsweise in der Spracherkennung das Hintergrundlexikon verkleinert werden.

Beim Graben et al. [1] stellen zu diesem Zwecke einen bidirektionalen Utterance-Meaning-Transducer vor, der mithilfe einer minimalistischen Grammatik (MG) den Zahlenwert einer gesprochenen Zahl ableitet. Statt vollständige Zahlen zu speichern, umfasst das zugehörige Lexikon Einträge aller benötigten Morpheme. Eingangs wird die Äußerung segmentiert und jedem Teilwort ein Lexikoneintrag zugeordnet. Aus den Merkmalen der Einträge wird die korrekte Kombination der Morpheme und der sich daraus ergebende Zahlenwert abgeleitet.

Da [1] die Funktionsweise der MG an einem vereinfachten Beispiel demonstrieren, werden Komplikationen, die sich für beliebig große ganze Zahlen ergeben, außer Acht gelassen. In diesem Beitrag werden diese Probleme vorgestellt und Lösungsvorschläge erarbeitet. Die MG wurde in einer Python-Anwendung implementiert.

Die von Stabler [2] vorgestellte minimalistische Grammatik besteht aus einem mentalen Lexikon sowie den Funktionen *merge* und *move*, mit denen Zeichen zusammengefügt werden können. Jeder Eintrag des mentalen Lexikons besteht aus drei Komponenten, die zusammen ein **Zeichen** bilden:

**a) Exponent**

Der Exponent eines Zeichens entspricht dem Morphem (z.B. *vier* oder *zig*). Durch die Kombination mehrerer Zeichen entstehen komplexere Exponenten aus Morphemsequenzen, beispielsweise *einundvierzig*.

**b) Syntaktische Merkmale**

In den syntaktischen Merkmalen ist bestimmt, mit welchen anderen Zeichen und auf welche Weise eine Kombination möglich ist. Zu den hierfür nötigen Merkmalen gehören die in Tabelle 1 aufgelisteten Kategorien. Die erlaubte Reihenfolge der Elemente ist durch reguläre Ausdrücke definiert [2, 3].

| **Syntaktisches Merkmal** | **Beschreibung** |
|---|---|
| Grundkategorie | syntaktische Grundkategorie (z.B. Wortklasse);<br>z.B. *num* (Numerale) |
| Selektor | selegiert Zeichen einer bestimmten Grundkategorie und befähigt *merge*-Operation;<br>z.B. =*num* |
| Lizensierer und Lizenznehmer | Lizensierer +*f* löst gemeinsam mit Lizenznehmer –*f* *move*-Operation aus |
| Typ | {::, :}<br>*::* markiert ein lexikalisches Zeichen, *:* ein abgeleitetes |

**Tabelle 1 -** Beschreibung der syntaktischen Merkmale

**c) Semantik**

Der dritte Teil des Zeichens, die Semantik, speichert den Zahlenwert beziehungsweise die zugehörige Operation (Summe oder Produkt) im Lambda-Kalkül. Für genauere Informationen zu dieser Schreibweise, siehe [1].

Für die vollständigen Zeichen wird die Kettenschreibweise aus [3] verwendet. Der erste Eintrag in Tabelle 2 mit Exponent *vier* gehört zur Grundkategorie *num* und kann daher vom zweiten Eintrag mit Selektor =*num* selegiert werden. Das ermöglicht eine *merge*-Operation für diese beiden Zeichen. Durch Lizensierer +*k* und Lizenznehmer –*k* wird anschließend eine *move*-Operation ausgelöst. Werden beide Zeichen kombiniert, erhält man das dritte gelistete Zeichen mit Exponent *vierzig*. Am Typ *:* ist erkenntlich, dass es sich im Unterschied zu den anderen beiden um ein abgeleitetes Zeichen der Grundkategorie *num* handelt.

(vier, :: num –k, 4)  (zig, :: =num +k num, λx. ×(10[1])(x))  (vierzig, : num, ×(10[1])(4))

**Tabelle 2 -** Beispieleinträge eines mentalen Lexikons

Schließlich bilden die *merge*- und *move*-Funktion neben dem Lexikon den zweiten Bestandteil der MG [3]. Für die folgenden Definitionen seien *s*, *t* die Exponenten, · ∈ {::, :}, *f* das erste syntaktische Merkmal einer Merkmalskette und *γ, δ* darauf folgende Merkmalsketten sowie *φ*, *ψ* semantische Ausdrücke.

Merge:

$$\frac{(s, ::= f\gamma, \varphi) \quad (t, \cdot f, \psi)}{(st, : \gamma, \varphi\psi)} \textbf{ merge1} \qquad (1)$$

$$\frac{(s, := f\gamma, \varphi) \quad (t, \cdot f, \psi)}{(ts, : \gamma, \varphi\psi)} \textbf{ merge2} \qquad (2)$$

$$\frac{(s, \cdot= f\gamma, \varphi) \quad (t, \cdot f\delta, \psi)}{(s, : \gamma, \varphi), \quad (\boldsymbol{t}, : \delta, \psi)} \textbf{ merge3} \qquad (3)$$

Move:

$$\frac{(s, : +f\gamma, \varphi) \quad (t, : -f, \psi)}{(ts, : \gamma, \varphi\psi)} \textbf{ move1} \qquad (4)$$

$$\frac{(s, : +f\gamma, \varphi) \quad (t, : -f\delta, \psi)}{(s, : \gamma, \varphi), \quad (t, : \delta, \psi)} \textbf{ move2} \qquad (5)$$

Durch geeignete Wahl der syntaktischen Merkmale eines Zeichens entsteht in Kombination mit *merge* und *move* eine vollständige MG, die alle Zahlworte und -werte bilden kann. Welche Herausforderungen sich dennoch bei der Entwicklung einer MG ergeben, die alle möglichen Zahlen bilden, aber gleichzeitig ungrammatische Kombinationen ausschließen kann, ist in den folgenden Abschnitten beschrieben.

## 2 Problem "zwanundzweizig" – Funktionen der Morpheme

Das Morphem *vier* tritt in den Wörtern *vierzig* und *vierundfünfzig* jeweils in der gleichen Form auf. Für die Zahl 2 dagegen ergeben sich im gleichen Kontext die Wörter *zwanzig* und *zweiundfünfzig*. Die jeweiligen morphologischen Ausprägungen dürfen ihre Funktion nicht tauschen, da dies ungrammatische Formen wie *zwanundzweizig* zur Folge hätte. Die syntaktischen Merkmale der Einträge müssen diese Einschränkung widerspiegeln: Einmal ist eine Verbindung mit *zig*, und einmal eine Platzierung vor *und* erlaubt.

### Vorschlag: Einträge mit gleichen Exponenten

Auch für *vier* bedeutet das zwei Einträge mit unterschiedlichen syntaktischen Merkmalen, jedoch mit gleichem Exponenten. Insgesamt weisen die Zahlwörter, welche die Zahlen 1 bis 9 repräsentieren, 4 Funktionen auf und benötigen daher auch je 4 Formen:

1. Suffix: alleinstehend (*vier*) oder als Einer in Zahlen größer 100 (hundert*vier*)
2. Zehnervielfaches: *zwan*zig, *vier*zig
3. *Zehn*-Einer: *vier*zehn, *sech*zehn
4. Präfix: als Einer vor *und* (*ein*undfünfzig) oder als Vielfaches von $10^n$, n > 1 (*sechs*hundert, *ein*tausend)

Für das Morphem *vier* ergeben sich so die vier unterschiedlichen Zeichen im unten gelisteten Beispiellexikon (vgl. Tabelle 3). Zusätzlich ist bei den Zeichen mit Exponenten *zig* und *und* sichtbar, dass sie Numeralien mit einem bestimmten Lizenznehmer benötigen, um ein korrektes Zahlwort bilden zu können.

| | | |
|---|---|---|
| (vier, :: num, 4) | (vier, :: num –k, 4) | (zig, :: =num +k num, λx. ×(10[1])(x)) |
| (vier, :: num –l, 4) | (vier, :: num –m, 4) | (und, :: =num =num +l c, λy. λy. +(y)(x)) |

**Tabelle 3** - Beispiellexikon mit Mehrfacheinträgen

**Vorschlag 2: Auswahl des korrekten Eintrags in Priority Queue**

Welcher Eintrag bei identischen Exponenten der richtige ist, hängt maßgeblich von der Reihenfolge ab, in der die Teilwörter geäußert wurden. Beispielsweise werden die syntaktischen Merkmale *num –k* gewählt, wenn der darauffolgende Exponent *zig* ist. Die Verarbeitung in einer *Priority Queue* berücksichtigt die Reihenfolge durch eine Indexierung ihrer Elemente [4]. Falls mehrere Einträge infrage kommen, wird erst bei der Verarbeitung eines Elements anhand vorher festgelegter Regeln bestimmt, welcher Eintrag der richtige ist. Dies kann als eine Erweiterung der MG verstanden werden. Auf andere Maße, wie etwa Wahrscheinlichkeiten, kann zur Lösung dieser Uneindeutigkeiten verzichtet werden, da die Zuordnung der Einträge im Falle von Zahlen festen Richtlinien folgt.

## 3 Problem "vierundvier" – Unterscheidung der Größenordnung

Ein Teil der ungrammatischen Ergebnisse kann somit verhindert werden. Allerdings erweist sich die gleichwertige Behandlung aller Zahlworte unter Grundkategorie *num* als Fehlerquelle. Fälschlicherweise ist so zum Beispiel die Kombination *vierundvier* möglich. Die Konjunktion *und* selegiert als Erstes ein Numerale. Das schließt alle Suffixe, wie *zwei* und *vier*, mit ein, obwohl nur eine Teilmenge aller Numeralien an dieser Position erlaubt ist.

**Vorschlag: Untergliederung der Grundkategorie**

Je nach Größenordnung ihres Werts und daraus resultierender unterschiedlicher Funktion erhalten die Zahlwörter eine eigene Klassifizierung[1] (vgl. Tabelle 4), was zu leicht veränderten syntaktischen Merkmalen im mentalen Lexikon führt (vgl. Beispiele in Tabelle 5). Die neuen Grundkategorien spielen auch für die Summenbildung bei Zahlen größer 100 eine Rolle.

| | | |
|---|---|---|
| 1-9: | num_e | (e: Einer) |
| 10-99: | num_z | (z: Zehner) |
| 20, 30, …, 90 | num_zig | |
| 100-999: | num_h | (h: Hundert) |
| 1000-999999: | num_t | (t: Tausend) |

**Tabelle 4** - Neue Grundkategorien

| | |
|---|---|
| (vier, :: num_e –k, 4) | (zig, :: =num_e +k num_zig, λx. ×(10[1])(x)) |
| (und, :: =num_y =num_e +l num_z, λy.λx. +(y)(x)) | |

**Tabelle 5 -** Beispiellexikon mit neuen Grundkategorien

[1] Durch die Wahl eines zusätzlichen Buchstabens zur Kategorienbenennung ist eine Unterkategorisierung möglich. Die Zahlwörter fallen noch immer unter die Kategorie Numerale, erhalten aber eine genauere Bestimmung. So beanspruchen sie keine Buchstaben anderer Grundkategorien, die vielleicht bei einer Zusammenfügung mit anderen MGs eine Rolle spielen.

## 4 Problem "hundertsieben" – Addition ohne auslösendes Morphem

Während im Beispiel oben das Morphem *und* eine Summenoperation für Zehner und Einer auslöst, enthält das Zahlwort *vierhundertsieben* kein extra Morphem, aus dem die Anweisung folgt, die Werte von *vierhundert* und *sieben* zu addieren.

### Vorschlag: Einträge mit leerem Exponenten

Trotzdem kann die Operation wie ein normaler Lexikoneintrag behandelt werden, dessen Exponent leer ist. Der griechische Buchstabe ε drückt den leeren String aus. Der zugehörige Eintrag für das hier genannte Beispiel wäre (ε, :: =num_e =num_h num_h, λy.λx. +(y)(x)). Da nicht alle beliebigen Zahlen addiert werden dürfen, ist die Unterscheidung der Grundkategorien auch hier essentiell. Als weitere Restriktion ist zu beachten, dass die ε-Einträge nur dann zur Anwendung kommen dürfen, wenn keine anderen *merge*- und *move*-Operationen mehr möglich sind. Für das Zahlwort *hundertsiebenundvierzig* darf zum Beispiel auf keinen Fall zuerst die Summe 107 berechnet werden, da sonst die durch *und* ausgelöste Addition verhindert wird.

## 5 Weitere Herausforderungen – Ausblick

Durch die Listung von vier Einträgen für jeden Einer und die notwendigen *merge*- und *move*-Operationen ist es ökonomischer, die Morpheme *zwanzig*, *dreißig*, …, *neunzig* sowie *dreizehn*, *vierzehn*, …, *neunzehn* als vollständige Einträge ins mentale Lexikon aufzunehmen. Die Größe des Lexikons verkleinert sich dadurch um drei Einträge, während sich die Verarbeitungsgeschwindigkeit erhöht.

In der hier vorgestellten Herangehensweise bleibt die Erkennung von Zahlwortgrenzen unberücksichtigt. Der Input *eins+vier+zig* etwa würde als ungrammatisch erkannt, obwohl es sich um die zwei aufeinanderfolgenden korrekten Wörter *eins* und *vierzig* handelt. Folglich wird auch immer die längste mögliche Kombination der Morpheme gewählt. Beim Input *hundert+sieben* kann es sich um die Zahl 107 oder die zwei getrennten Zahlen 100 und 7 handeln. Welche die wahrscheinlichste Variante ist, ist eine wichtige Entscheidung in der Spracherkennung. Die MG muss an dieser Stelle mit geeigneten Wahrscheinlichkeitsmaßen verknüpft werden.

## Literatur


[1] P. BEIM GRABEN, W. MEYER, R. RÖMER UND M. WOLFF: Bidirektionale Utterance-Meaning-Transducer für Zahlworte durch kompositionale minimalistische Grammatiken. In P. BIRKHOLZ UND S. STONE (Hrsg.): Tagungsband der 30. Konferenz Elektronische Sprachsignalverarbeitung (ESSV), vol. 91, S. 76-82, 2019.

[2] E. STABLER: *Derivational Minimalism*. In C. RETORÉ (Hrsg.): *Logical Aspects of Computational Linguistics*, S. 68-95, New York, Springer, 1997.

[3] E. STABLER UND E. KEENAN: Structural Similarity within and among Languages. In Theoretical Computer Science, vol 293, S. 345-363, 2003.

[4] E. STABLER: Top-Down Recognizers for MCFGs and MGs. In Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics, S. 39-48.