Studienarbeit

Fixed-Left-Right-Wrapper: Merkmale und Lernbarkeit

Matthias Degen

Darmstadt, den 01.10.2004

Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

Betreuer:
Dr. Gunter Grieser

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfaßt und keine anderen Hilfsmittel als die in der Arbeit angegebenen benutzt habe.

Darmstadt, den 1. Oktober 2004

Matthias Degen

Inhaltsverzeichnis

1 Einführung und Motivation

Das Internet bietet eine Fülle an Informationen. Es fällt jedoch aufgrund der Größe und Dynamik des Internets schwer, einen Überblick zu behalten. Deshalb wäre es schön, wenn der Computer erkennen könnte, wo wichtige Informationen stecken und sie automatisch extrahiert. Viele Systeme benutzen dazu sogenannte *Wrapper* Prozeduren, die das Extrahieren von bestimmten Informationen ermöglichen. Ein Wrapper nutzt die Tatsache aus, dass viele Dokumente für das Internet semi-strukturiert sind. Sie sind in HTML, XML oder einer ähnlichen Sprache geschrieben und müssen von einem Browser interpretiert werden. Aus diesem Grund besitzen sie syntaktische Ausdrücke, die die Interpretation der Dokumente steuern. Diese syntaktischen Ausdrücke sind vor dem Benutzer verborgen. Obwohl sich die Inhalte bei den Anbietern, z.B. bei Suchmaschinen, oft ändern, ist das Layout normalerweise konstant. Die Informationen sind also immer wieder gleich verpackt. Weiss man, wie eine bestimmte Art von Information eingepackt ist (kennt man den Wrapper), dann kann man dieses Wissen nutzen, um ähnliche Informationen aus anderen Dokumenten zu extrahieren. Für unterschiedliche Internetquellen benötigt man auch unterschiedliche Wrapper. Jeder Wrapper definiert einen anderen Extraktionsmechanismus. Diese Wrapper per Hand zu schreiben ist aufwändig und fehlerbehaftet. Bei einigen Internetseiten ändert sich außerdem manchmal das Layout. Dadurch ändert sich natürlich auch die Struktur und die Wrapper werden unbrauchbar. Der Inhalt ändert sich aber nicht. Besser wäre es deshalb, wenn ein System in der Lage wäre, für bestimmte Informationen den passenden Wrapper selbst abzuleiten. Eine Technik, Wrapper automatisch zu lernen, ist die *Wrapper Induktion*. Dabei wird aus einer Menge von Beispielseiten, in denen überall die zu extrahierenden Textteile markiert sind, versucht, den Wrapper zu bestimmen.

1.1 Wrapperklassen

In dieser Studienarbeit werden wir zwei verschiedene Wrapperklassen vergleichen.

Die erste Wrapperklasse ist die sogenannte *Island-Wrapper-Klasse*. Über diese Klasse gibt es schon einige Arbeiten (vgl. [GJLT00, GL01]). Im Fokus dieser Untersuchungen stand vor allem die Lernbarkeit dieser Klasse. Schaut man sich jedoch die Entstehungsgeschichte der Wrapper (beginnend bei [Kus97, Kus00]) an, dann waren nicht die Lernbarkeit des Wrappers, sondern oft der prozedurale Aspekt der Extraktion im Vordergrund der Betrachtung. Innerhalb des formalen Rahmens, der in [GJLT00, GL01] geschaf-

fen wurde, können aber auch andere Wrapperklassen beschrieben werden. Wir werden eine zweite Wrapperklasse innerhalb dieses Rahmens darstellen und untersuchen.

Die neue Klasse, die wir einführen, nennen wir *Fixed-Left-Right-Wrapper-Klasse (FLR-Wrapper-Klasse)*. Sie ist von der *Left-Right-Wrapper-Klasse* (vgl. [Kus97, Kus00]) abgeleitet.

Die Island-Wrapper und die FLR-Wrapper-Klasse haben gemeinsam, dass die durch die Wrapper definierten Mechanismen Texte extrahieren, die durch Begrenzer vor dem Textanfang und nach dem Textende gekennzeichnet sind. Bei der Wrapper Induktion dieser beiden Klassen werden nun Sprachen für die Begrenzer gesucht, die die zu extrahierende Information links und rechts einschließen. Beide Wrapper sind vollständig durch diese Begrenzersprachen bestimmt. Die Wrapper unterscheiden sich jedoch in der Ausführung der Extraktion. Diese Unterschiede werden wir versuchen auszuloten. Dabei werden wir den FLR-Wrapper formal definieren und hinsichtlich der Lernbarkeit untersuchen. Weiter werden wir diese Ergebnisse mit den bereits existierenden Erkenntnissen der Island-Wrapper vergleichen. Insbesondere werden wir die Lernbarkeit der Wrapperklassen für Begrenzersprachen begrenzter Kardinalität untersuchen.

1.2 Aufbau und Struktur der Studienarbeit

Diese Studienarbeit bezieht sich auf Ergebnisse der Untersuchungen [GJLT00, GL01]. Alle Informationen über den Island-Wrapper sind diesen Arbeiten entnommen. Da wir die FLR-Wrapper-Klasse innerhalb des selben formalen Rahmens untersuchen wollen, benutzen wir in unserer Arbeit dieselben Begriffe und Methoden.

Der Rest der Studienarbeit ist wie folgt gegliedert. Wir beginnen in Kapitel 2 mit der Definition der in dieser Arbeit benutzen Begriffe. In Kapitel 3 werden wir den Island-Wrapper erklären und formal definieren. Kapitel 4 beschäftigt sich analog mit dem FLR-Wrapper. Dabei werden wir, ausgehend von einer in Pseudo-Code angegebenen Extraktionsprozedur, die durch den FLR-Wrapper bestimmten Mechanismen der Extraktion in Termini formaler Sprachen definieren, um den FLR-Wrapper hinsichtlich der Lernbarkeit untersuchen zu können. In Kapitel 5 werden wir dann die Unterschiede der beiden Ansätze ausloten und mit einem Beispiel verdeutlichen. Wann und wie der FLR-Wrapper lernbar ist, wird in Kapitel 6 geklärt und in Kapitel 7 werden wir die Lernbarkeit der Wrapperklassen hinsichtlich Begrenzerspra-

chen begrenzter Kardinalität klären. Zum Abschluß werden die Ergebnisse in Kapitel 8 noch einmal zusammengefasst und einen Ausblick auf mögliche zukünftige Untersuchungen gegeben.

2 Benutzte Begriffe

In diesem Abschnitt werden die meisten Begriffe definiert, die wir später nutzen werden. Weitere Notationen werden in den jeweiligen Kapiteln erklärt.

Sei Σ ein endliches Alphabet. Mit Σ^* bezeichnen wir die Menge aller Wörter über Σ. $\Sigma^+ = \Sigma^* \backslash \{\varepsilon\}$, wobei ε das leere Wort ist. Jede Teilmenge $G \subseteq \Sigma^*$ ist eine Sprache. Die Länge eines Wortes $|.|$ ist als die Anzahl seiner Zeichen definiert. D.h. sei $x \in \Sigma^*$ mit $x = a_1 a_2 ... a_n$ und $a_i \in \Sigma$ für alle $1 \leq i \leq n$, dann gilt $|x| = n$. Angenommen, x und y seien Wörter. Dann steht xy für die Konkatenation von x und y. Genauer gesagt, sei $x = a_1 a_2 ... a_i$ und $y = b_1 b_2 ... b_j$, dann ist xy das Wort der Länge $i + j$: $xy = a_1 a_2 ... a_i b_1 b_2 ... b_j$. Für alle $G, G' \subseteq \Sigma^*$ sei $G \circ G' = \{wv \mid w \in G, v \in G'\}$, $\tilde{\Sigma}^*_G = \Sigma^* \backslash (\Sigma^* \circ G \circ \Sigma^*)$ und $\tilde{\Sigma}^+_G = \tilde{\Sigma}^*_G \backslash \{\varepsilon\}$.

$G' \sqsubseteq_p G$ bedeutet, dass G' ein Präfix von G ist und $G' \sqsubset_p G$, dass G' ein echter Präfix von G ist. $G' \sqsubseteq_s G$ schreiben wir, wenn G' ein Suffix von G ist und $G' \sqsubset_s G$, wenn G' ein echter Suffix von G ist.

Sei $\Sigma = \{a_1, a_2, ..., a_n\}$ ein Alphabet und sei $a_1 <_\Sigma a_2 <_\Sigma ... <_\Sigma a_n$ eine Ordnung auf Σ. Mit $<_{lex}$ bezeichnen wir die *lexikographische Ordnung* auf Σ^*, d.h. für $w, v \in \Sigma^*$ schreiben wir $w <_{lex} v$, wenn eine der beiden folgenden Bedingungen erfüllt ist:

(i) $|w| < |v|$

(ii) $|w| = |v|$ und es existieren $x, y, y' \in \Sigma^*$, $a, b \in \Sigma$, so dass $w = xay \wedge v = xby' \wedge a <_\Sigma b$ gilt.

Seien $G = \{g_1, g_2, ..., g_m\}$, $F = \{f_1, f_2, ..., f_n\}$ zwei Mengen mit $g_i, f_j \in \Sigma^*$ für alle $1 \leq i \leq n$ und $m, n \in \mathbb{N}$. Seien $g_1, g_2, ...g_m$ und $f_1, f_2, ..., f_n$ lexikographisch geordnet, d.h. es gelte $g_1 <_{lex} g_2 <_{lex} ... <_{lex} g_m$ und $f_1 <_{lex} f_2 <_{lex} ... <_{lex} f_n$. Wir schreiben $G <_{lex} F$, wenn eine der beiden folgenden Bedingungen erfüllt ist:

(i) $|G| < |F|$.

(ii) $|G| = |F|$ und es existiert ein k mit $1 \leq k \leq m$, so dass $g_l = f_l$ und $g_k <_{lex} f_k$ für alle $1 \leq l \leq k$ gilt.

Sei $\mathcal{G}$ eine Menge von Sprachen aus Σ^*. $G \in \mathcal{G}$ ist die *lexikographisch erste* Sprache in $\mathcal{G}$, falls für alle $G' \in \mathcal{G}$ mit $G' \neq G$ gilt: $G <_{lex} G'$.

Für alle $k \in \mathbb{N}$ ist $\mathcal{L}_k$ die Familie aller nichleeren Sprachen $G \subseteq \Sigma^+$ mit einer begrenzten Kardinalität nicht größer als k (d.h. $card(G) \leq k$).

Sei d ein Dokument. Für den Textabschnitt zwischen der Position x und der Position y in d einschließlich x und y schreiben wir $d[x, y]$.

Modell des induktiven Lernens im Limes

Im Folgenden erklären wir die Grundkonzepte von Golds Modell des *induktiven Lernens im Limes* (vgl. [Gol67]).

Eine Familie von nichtleeren Sprachen $\mathcal{L}$ heißt *indizierbare Familie rekursiver Sprachen* (kurz: *indizierbare Familie*) genau dann, wenn eine Aufzählung $(G_i)_{i\in\mathbb{N}}$ von $\mathcal{L}$ und ein totales und berechenbares Prädikat p existieren, so dass folgendes gilt:

- Für alle $i \in \mathbb{N}$ und alle $w \in \Sigma^+$ gilt: $p(i, w) \leftrightarrow w \in G_i$.

Viele praktisch relevante Sprachfamilien sind indizierbare Familien. Auch die Menge aller durch Island-Wrapper oder FLR-Wrapper beschreibbaren Sprachen ist eine indizierbare Familie, falls die Menge der potentiellen Begrenzersprachen eine indizierbare Familie ist.

Eine Möglichkeit, die Informationen über eine Zielsprache zu spezifizieren, ist via dem Konzept eines *Textes*. Ein *Text* ist eine unendliche Folge von Wörtern der Sprache, so dass jedes Wort der Zielsprache mindestens einmal in der Folge vorkommt. Sei G eine Sprache und $t = s_0, s_1, ...$ eine unendliche Folge der Wörter aus Σ^*, so dass $\{s_k \mid k \in \mathbb{N}\} = G$ gilt. Dann ist t ein Text der Sprache G. Für einen Text t bezeichnen wir mit t_x das Anfangsstück der Länge x, d.h. $t_x = s_0, ..., s_k$.

Ein Lernverfahren bekommt als Eingabe wachsende Abschnitte eines Textes t der Zielsprache G und generiert seine Hypothese. Es lernt eine Zielsprache G aus einem Text t, wenn sich die Folge der Ausgaben auf eine Hypothese stabilisiert, die genau G beschreibt. Fortan nennen wir diese Lernverfahren IIM (Abkürzung von *inductive inference machine*). Man sagt, dass eine IIM G aus Text lernt, wenn sie G aus jedem Text der Sprache lernt. Weiterhin sagt man, dass eine Sprachfamilie $\mathcal{L}$ aus Text lernbar ist, wenn es eine IIM gibt, die jede Sprache $G \in \mathcal{L}$ aus Text lernt. Mit $LimTxt$ bezeichnen wir die Menge aller indizierbaren Familien $\mathcal{L}$, für die es eine IIM gibt, die $\mathcal{L}$ aus Text lernt.

3 Island-Wrapper

Als nächstes definieren wir die Island-Wrapper-Klasse (vgl. [GJLT00, GL01]). Die Island-Wrapper bestimmen Mechanismen, die Textteile aus einem Dokument extrahieren. Dabei wird davon ausgegangen, dass ein wichtiger Textteil v durch linke und rechte Begrenzer (l und r) identifiziert werden kann. Diese Begrenzer sind nicht fest, sondern gehören zu linken und rechten Begrenzersprachen. Den Textabschnitt lvr nennen wir eine *Insel*. Oft möchte man nicht nur einen einzelnen Textteil extrahieren, sondern mehrere in Zusammenhang stehende Teile. Deshalb werden bei der Definition des Island-Wrappers mehrere Inseln in Beziehung gesetzt, die so nahe wie möglich beieinander stehen müssen. Drei Annahmen gehen dem Entwurf voraus. Erstens, die Tupel, die aus einem Dokument extrahiert werden, haben eine feste Stelligkeit. Zweitens, soll ein Tupel $(v_1, v_2, ...v_n)$ extrahiert werden, dann wird angenommen, dass das Wort v_1 im Dokument d vor dem Wort v_2 kommt. Drittens, die Tupel können durch die sie umgebenden Zeichen, ihre Begrenzer, identifiziert werden. Die einzelnen Tupelelemente müssen dabei nahe beieinander stehen.

Definition 1 ([GL01]) *Sei $n \geq 1$, seien $L_1, R_1, ..., L_n, R_n$ Begrenzersprachen und sei $W = (L_1, R_1, ..., L_n, R_n)$ der korrespondierende Island-Wrapper. Dann definiert der Island-Wrapper W die folgende Abbildung S_W von Dokumenten zu n-stelligen Tupeln: Sei irgendein Dokument d gegeben, dann ist $S_W(d)$ die Menge aller n-stelligen Tupel $\langle v_1, ..., v_n \rangle \in (\Sigma^+)^n$, wenn es $x_0 \in \Sigma^*, ..., x_n \in \Sigma^*$, $l_1 \in L_1, ..., l_n \in L_n$ und $r_1 \in R_1, ..., r_n \in R_n$ gibt, so dass folgende Bedingungen erfüllt sind:*

(i) $d = x_0 l_1 v_1 r_1 ... l_n v_n r_n x_n$.

(ii) *für alle $i \in \{1, ..., n\}$: v_i enthält kein Teilwort, das zu R_i gehört, d.b., $v_i \in \tilde{\Sigma}^+_{R_i}$.*

(iii) *für alle $i \in \{1, ..., n-1\}$: x_i enthält kein Teilwort, das zu L_{i+1} gehört, d.b., $x_i \in \tilde{\Sigma}^*_{L_{i+1}}$.*

△

Die Bedingungen (ii) und (iii) sollen dafür sorgen, dass die extrahierten Textteile klein sind und dass auch die Distanz zwischen den einzelnen Textteilen klein bleibt. Damit soll sichergestellt werden, dass die Textteile auch wirklich zu einem Tupel gehören. Wie wir aber im nächsten Kapitel sehen werden, wird das bei der anderen Wrapperklasse strenger definiert.

4 Fixed-Left-Right-Wrapper

In diesem Kapitel führen wir eine neue Wrapperklasse ein. Wir nennen sie *Fixed-Left-Right-Wrapper-Klasse (FLR-Wrapper)*. Der FLR-Wrapper ist an den *Left-Right-Wrapper* (vgl. [Kus97, Kus00]) angelehnt. Der Entwurf dieses Wrappers basiert auf denselben Annahmen, die wir bei der Definition des Island-Wrappers hatten. Der FLR-Wrapper wird vollständig durch rechte und linke Begrenzersprachen bestimmt. Ebenso nehmen wir an, dass die Stelligkeit der zu extrahierenden Tupel fest ist und für ein Tupel $(v_1, v_2, ..., v_n)$ das Wort v_1 im Dokument immer vor dem Wort v_2 kommt. Sei $n \geq 1$, für die Begrenzersprachen L_1, R_1, ..., L_n und R_n ist $W = (L_1, R_1, ..., L_n, R_n)$ der korrespondierende FLR-Wrapper. Die Extraktion der wichtigen Textteile aus einem Dokument unterscheidet sich jedoch bei den beiden Klassen. Die folgende in Pseudo-Code geschriebene Prozedur $exec_{FLR}$ verdeutlicht das. Sie spezifiziert, wie durch die FLR-Wrapper relevante Textteile extrahiert werden:

```
procedure exec_FLR(Wrapper W, Dokument d)
p, i, k := 0
repeat
     for j := 1 to n do
       p^l_j a := Position des ersten Auftretens eines Wortes
              aus L_j in d ab einschließlich Pos. p
       p^l_j e := Endposition des kürzesten dieser Worte
       p^r_j a := Position des ersten Auftretens eines Wortes
              aus R_j in d ab einschließlich Pos. p^l_j e + 2.
              Wird kein Wort aus R_j gefunden, dann
              return ⟨v_1^1, v_2^1, ..., v_n^1⟩, ..., ⟨v_1^{j-1}, v_2^{j-1}, ..., v_n^{j-1}⟩
       p^r_j e := Endposition des kürzesten dieser Worte
        v^i_j := d[p^l_j e + 1, p^r_j a - 1]
          p := p^r_j e + 1
     end
     k := i
     i := i + 1
until ''es kommt kein weiteres Element von L_1 in d vor''
return ⟨v_1^1, v_2^1, ..., v_n^1⟩, ⟨v_1^2, v_2^2, ..., v_n^2⟩, ..., ⟨v_1^k, v_2^k, ..., v_n^k⟩
```

Die Abbildung S_W spezifiziert die durch den Island-Wrapper bestimmte Extraktion der Texteile. Sie wird im Kapitel 3 in Termini formaler Sprachen definiert. Die Prozedur $exec_{FLR}$ ist ein Programm, das eine entsprechende Abbildung für den FLR-Wrapper berechnet. Damit wir die Lernbarkeit des FLR-Wrapper untersuchen und die beiden Wrapperklassen besser vergleichen können, werden wir nun ein Modell definieren, das diese Abbildung (im folgenden FLR_W genannt) in Termini formaler Sprachen beschreibt.

Aus der Prozedur $exec_{FLR}$ lassen sich folgende Bedingungen für die Begrenzersprachen des FLR-Wrapper ableiten:

1. Es darf kein kürzeres Wort derselben Begrenzerprache existieren, das dieselbe Anfangsposition im Dokument hat.
2. Es darf kein Element der linken Begrenzersprache L_j geben, das im Dokument d im Intervall $[p, p_j^l a]$ anfängt.
3. Es darf kein Element der rechten Begrenzersprache R_j geben, das im Dokument d im Intervall $[p_j^l e + 2, p_j^r a]$ anfängt.

Sei $l_j v_j r_j$ eine Insel in einem Dokument d, sei $x_j = d[p, p_a^l - 1]$ und sei x_{j+1} der Rest des Dokumentes d ab Pos. $p_j^r e + 1$. Dann definieren wir unsere Beobachtungen formal so:

1. Für linke Begrenzer: $\neg\exists l' \in L_j$ mit $l' \sqsubset_p l_j$

   Für rechte Begrenzer: $\neg\exists r' \in R_j$ mit $r' \sqsubset_p r_j$
2. $\neg\exists x' \in \Sigma^+$, $l' \in L_j$ mit $x' \sqsubset_p x_j \wedge x'l' \sqsubseteq_p x_j l_j v_j r x_{j+1}$
3. $\neg\exists v' \in \Sigma^+$, $r' \in R_j$ mit $v' \sqsubset_p v_j \wedge v'r' \sqsubseteq_p v_j r_j x_{j+1}$

Für die Tupel $\langle v_1^1, v_2^1, ..., v_n^1\rangle, \langle v_1^2, v_2^2, ..., v_n^2\rangle, ..., \left\langle v_1^k, v_2^k, ..., v_n^k\right\rangle$ gilt: Sei $1 \leq i < k$, v_n^i steht in einem Dokument immer vor v_1^{i+1}. Deshalb werden wir die Abbildung FLR_W mit Hilfe von zwei anderen Abbildungen rekursiv definieren. $zerlege_W$ ist eine Abbildung von einem Dokument d auf ein $(n+1)$-stelliges Tupel. Die ersten n Elemente des Tupel sind die zu extrahierenden Textteile $(v_1^1, v_2^1, ..., v_n^1)$. Das letzte Element ist der Rest des Dokumentes nach dem rechten Begrenzer von v_n^1. $rest_W$ ist eine Abbildung von einem Dokument d auf den Rest des Dokumentes nach dem rechten Begrenzer von v_n^1.

Die drei auf der letzten Seite getroffenen Beobachtungen verwenden wir in unserer Definition von $zerlege_W$:

Definition 2 *Sei $n \geq 1$, seien $L_1, R_1, ..., L_n, R_n$ Begrenzersprachen und sei $W = (L_1, R_1, ..., L_n, R_n)$ der korrespondierende Wrapper. Dann definiert der FLR-Wrapper W folgende Abbildung $zerlege_W$. Sei irgendein Dokument d gegeben, dann ist $zerlege_W(d)$ das Tupel $\langle v_1, ..., v_n, x_{n+1}\rangle$, mit $v_1, ..., v_n \in \Sigma^+$ und $x_{n+1} \in \Sigma^*$, wenn es $x_0 \in \Sigma^*, ..., x_n \in \Sigma^*$, $l_1 \in L_1, ..., l_n \in L_n, r_1 \in R_1, ..., r_n \in R_n$ gibt, so dass folgende Bedingungen erfüllt sind:*

1. $d = x_1 l_1 v_1 r_1 ... x_n l_n v_n r_n x_{n+1}$
2. *für* $i = 1$: $\neg\exists x' \in \Sigma^*, l' \in L_1$ *mit*
   $x' \sqsubset_p x_1 \wedge x'l' \sqsubseteq_p x_1 l_1 v_1 r_1 ... x_n l_n v_n r_n x_{n+1}$
3. *für alle* $i \in \{2, ..., n\}$: $\neg\exists x' \in \Sigma^+, l' \in L_i$ *mit* $x' \sqsubset_p x_i \wedge$
   $x'l' \sqsubseteq_p x_i l_i v_i r_i ... x_n l_n v_n r_n x_{n+1}$
4. *für alle* $i \in \{1, ..., n\}$: $\neg\exists l' \in L_i$ *mit* $l' \sqsubset_p l_i$
5. *für alle* $i \in \{1, ..., n-1\}$: $\neg\exists v' \in \Sigma^+, r' \in R_i$ *mit* $v' \sqsubset_p v_i \wedge$
   $v'r' \sqsubseteq_p v_i r_i x_{i+1} l_{i+1} ... x_n l_n v_n r_n x_{n+1}$
6. *für* $i = n$: $\neg\exists v' \in \Sigma^+, r' \in R_n$ *mit* $v' \sqsubset_p v_n \wedge$
   $v'r' \sqsubseteq_p v_n r_n x_{n+1}$
7. *für alle* $i \in \{1, ..., n\}$: $\neg\exists r' \in R_i$ *mit* $r' \sqsubset_p r_i$

$\triangle$

Es folgt die formale Definition von $rest_W$:

Definition 3 *Sei $n \geq 1$, seien $L_1, R_1, ..., L_n, R_n$ Begrenzersprachen und sei $W = (L_1, R_1, ..., L_n, R_n)$ der korrespondierende Wrapper. Dann definiert der FLR-Wrapper W folgende Abbildung $rest_W$. Sei irgendein Dokument d gegeben, dann gilt für $rest_W(d)$:*

$rest_W(d) = x_{n+1}$, *falls* $v_1, ..., v_n \in \Sigma^+$ *existieren, so dass* $(v_1, ..., v_n, x_{n+1}) \in zerlege_W(d)$ *gilt.*

$\triangle$

Mit Hilfe von $zerlege_W$ und $rest_W$ definieren wir jetzt die Abbildung FLR_W von einem Dokument auf eine Menge von n-stelligen Tupeln. In dieser Menge sind die ersten n Elemente des Ergebnisstupels von $zerlege_W(d)$ und die Ergebnisse von FLR_W angewendet auf $rest_W(d)$ enthalten:

Definition 4 *Sei $n \geq 1$, seien $L_1, R_1, ..., L_n, R_n$ Begrenzersprachen und sei $W = (L_1, R_1, ..., L_n, R_n)$ der korrespondierende Wrapper. Dann definiert der FLR-Wrapper W folgende Abbildung FLR_W. Sei irgendein Dokument d gegeben, dann ist $FLR_W(d)$ die Menge aller n-stelligen Tupeln $\langle v_1, ..., v_n \rangle \in (\Sigma^+)^n$, so dass folgende Bedingung erfüllt ist:*

$$FLR_W(d) = \{(v_1, ..., v_n) \mid \exists x_{n+1} \in \Sigma^* \text{ , so dass } (v_1, ..., v_n, x_{n+1}) \in zerlege_W(d)\} \cup FLR_W(rest_W(d))$$

$\triangle$

5 Unterschiede in den Ansätzen

Wir möchten nun die Unterschiede der beiden Klassen aufzeigen und mit einem Beispiel verdeutlichen. Die durch die Wrapper definierten Mechanismen extrahieren Texte, die von linken und rechten Begrenzern eingehüllt sind. Beide Wrapperklassen sind vollständig durch die linken und rechten Begrenzersprachen bestimmt. Es gibt aber einige Unterschiede zwischen den Klassen.

Zum einen definieren die Wrapperklassen sehr verschiedene Extraktionsmechanismen. Ist ein Tupel mithilfe des FLR-Wrappers gefunden, dann wird nicht das ganze Dokument nach weiteren Tupel durchsucht, sondern lediglich der Rest des Dokumentes nach dem Ende der letzten Insel des Tupels. Dadurch können sich die extrahierten Tupel in ihrer Position im Dokument nicht überschneiden. Der Island Wrapper definiert einen ganz anderen Extraktionsmechanismus. Nach dem Fund eines Tupels wird das ganze Dokument nach weiteren Tupeln durchsucht. So kann es vorkommen, dass sich die mithilfe des Island Wrappers extrahierten Tupel in ihrer Position im Dokument überschneiden.

Der zweite Unterschied liegt in den abweichenden Anforderungen an die Begrenzer. Zwar ist bei beiden Wrapperklassen der Zweck dieser Anforderungen derselbe: Es soll sichergestellt werden, dass die Textteile auch wirklich zu einem Tupel gehören. Die Bedingungen des FLR-Wrappers sind jedoch strenger.

Folgende Bedingungen werden an die Begrenzer des FLR Wrappers gestellt:

(i) Von einem Begrenzer darf kein Präfix existieren, der auch Element derselben Begrenzersprache ist.

(ii) Vor dem linken Begrenzer der ersten Insel eines Dokumentes darf kein Element der Sprache dieses Begrenzers beginnen.

(iii) In dem Textteil zwischen dem rechten Begrenzer einer Insel und dem linken Begrenzer der nachfolgenden Insel beginnt kein Wort, das Element der Sprache des linken Begrenzers ist. Dies gilt auch dann, wenn die nachfolgende Insel zu einem einem anderen Tupel gehört.

(iv) In dem Textteil zwischen dem linken und rechten Begrenzer einer Insel beginnt kein Wort, das zu der Sprache des rechten Begrenzer gehört.

Folgende Bedingungen werden an die Begrenzer des Island-Wrappers gestellt:

(i) Der Textteil vor dem linken Begrenzer der ersten Insel eines Dokumentes enthält kein Wort, das zu der Sprache dieses Begrenzers gehört.

(ii) Der Textteil zwischen dem rechten Begrenzer einer Insel und dem linken Begrenzer der nachfolgenden Insel enthält kein Wort, das zu der Sprache des linken Begrenzers gehört.

(iii) Der Textteil zwischen dem linken und rechten Begrenzer einer Insel enthält kein Wort, das zu der Sprache des rechten Begrenzers gehört.

Wie man erkennt sind die Anforderungen an die Begrenzer des Island-Wrappers schwächer. Deshalb kann es vorkommen, dass mithilfe des Island-Wrappers mehr Tupel extrahiert werden als mithilfe des FLR-Wrappers. Genauer gesagt, für alle Wrapper W gilt: $FLR_W \subseteq S_W$.

Betrachten wir dazu auf der nächsten Seite ein Beispiel.

Seien das Dokument $d = ababacdabbdac$ und die Begrenzersprachen $L_1 = \{ab\}$, $R_1 = \{c, d, cd\}$, $L_2 = \{ab\}$ und $R_2 = \{da, a\}$ gegeben.

d enthält unter der Island-Wrapper-Semantik folgende Tupel und Inseln. Zur besseren Übersicht haben wir die Inseln in Klammer und die zu extrahierenden Texteile in fetter Schrift gesetzt:

1. (ab**aba**c)d(ab**b**da)c
   $l_1 = ab$, $v_1 = \mathbf{aba}$, $r_1 = c$, $l_2 = ab$, $v_2 = \mathbf{b}$, $r_2 = da$
2. (ab**aba**c)d(ab**bd**a)c
   $l_1 = ab$, $v_1 = \mathbf{aba}$, $r_1 = c$, $l_2 = ab$, $v_2 = \mathbf{bd}$, $r_2 = a$
3. (ab**aba**cd)(ab**b**da)c
   $l_1 = ab$, $v_1 = \mathbf{aba}$, $r_1 = cd$, $l_2 = ab$, $v_2 = \mathbf{b}$, $r_2 = da$
4. (ab**aba**cd)(ab**bd**a)c
   $l_1 = ab$, $v_1 = \mathbf{aba}$, $r_1 = cd$, $l_2 = ab$, $v_2 = \mathbf{bd}$, $r_2 = a$
5. (abab**a**c)d(ab**b**da)c
   $l_1 = ab$, $v_1 = \mathbf{a}$, $r_1 = c$, $l_2 = ab$, $v_2 = \mathbf{b}$, $r_2 = da$
6. (abab**a**c)d(ab**bd**a)c
   $l_1 = ab$, $v_1 = \mathbf{a}$, $r_1 = c$, $l_2 = ab$, $v_2 = \mathbf{bd}$, $r_2 = a$
7. (abab**a**cd)(ab**b**da)c
   $l_1 = ab$, $v_1 = \mathbf{a}$, $r_1 = cd$, $l_2 = ab$, $v_2 = \mathbf{b}$, $r_2 = da$
8. (abab**a**cd)(ab**bd**a)c
   $l_1 = ab$, $v_1 = \mathbf{a}$, $r_1 = cd$, $l_2 = ab$, $v_2 = \mathbf{bd}$, $r_2 = a$

d enthält unter der FLR-Wrapper-Semantik nur einen Tupel:

1. (ab**aba**c)d(ab**b**da)c
   $l_1 = ab$, $v_1 = \mathbf{aba}$, $r_1 = c$, $l_2 = ab$, $v_2 = \mathbf{b}$, $r_2 = da$

Die Unterschiede in den Definitionen können sich vielleicht auch auf die Lernbarkeitsergebnisse auswirken. Das untersuchen wir in den nächsten beiden Abschnitten.

6 Lernbarkeit des FLR-Wrappers

Im Folgenden zeigen wir, wie der FLR-Wrapper gelernt werden kann. Der FLR-Wrapper ist vollständig durch die Begrenzersprachen bestimmt. Sind die Begrenzersprachen lernbar, dann ist auch der FLR-Wrapper lernbar. Wir haben es hier jedoch nicht mit einem traditionellem Lernbarkeitsproblem zu tun. Ein potentieller Benutzer gibt unserem System nämlich nur implizit Informationen über die Begrenzersprachen, in dem er in einem Beispiel die zu extrahierenden Texteile markiert. Deshalb werden wir das aus [GL01] stammende Konzept des Lernens von Sprachen aus *markiertem Text* übernehmen.

6.1 Lernen von Sprachen aus markiertem Text

Ein Benutzer markiert in einem Dokument d die zu extrahierenden Textteile. Diese Textteile sind die n-Tupel $\langle v_1, ..., v_n \rangle$. Er bestimmt dabei, wo ein Textteil anfängt und wo er aufhört. Angenommen, jedes v_i endet, bevor v_{i+1} beginnt. Dann teilt der Benutzer dadurch das Dokument in $2n + 1$ aufeinanderfolgende Textteile $u_0, v_1, u_1, ..., v_n, u_n$. Die Wortfolge $u_0 v_1 u_1 ... v_n u_n$ entspricht dabei dem Dokument d. Ein $2n+1$-Tupel $\langle u_0, v_1, u_1, ..., v_n, u_n \rangle$ nennen wir *n-markiertes Dokument*.

Sei $W = (L_1, R_1, ..., L_n, R_n)$ ein FLR-Wrapper und sei $m = \langle u_0, v_1, u_1, ..., v_n, u_n \rangle$ ein n-markiertes Dokument. Dann ist m ein *Beispiel* für W, wenn eine der beiden Bedingungen erfüllt ist:

1. Es existieren $x \in \Sigma^+$ und $x' \in \Sigma^*$, so dass gilt:
   $u_n = xx'$ und $(v_1, ..., v_n, x') \in zerlege_W(d)$
2. Es existieren $x, j \in \Sigma^+$ und $x' \in \Sigma^*$, so dass gilt:
   $u_n = xx'$ und $(v_1, ..., v_n, x') \in zerlege_W(d_2)$ mit $d_2 = rest_w^j(d)$

Dabei ist $d = u_0 v_1 u_1 ... v_n u_n$ und mit $rest_w^j(d)$ meinen wir die j-fache iterative Anwendung von $rest_w$ auf d. Ein *n-markierter Text* t für einen Wrapper W ist eine unendliche Folge von n-markierten Dokumenten, die als Beispiele für W dienen. Dabei muss jedes Beispiel von W mindestens einmal in t vorkommen. Ein Lernverfahren bekommt als Eingabe wachsende Abschnitte eines n-markierten Textes t für einen Zielwrapper W und generiert seine Hypothese. Es lernt den Wrapper W, wenn sich die Folge der Ausgaben auf eine Hypothese W' stabilisiert, so dass $FLR_{W'}(d) = FLR_W(d)$ für alle Dokumente $d \in \Sigma^*$ gilt.

Ein Beispiel für ein Lernverfahren ist somit keine ununterbrochene Wortfolge, sondern in mehrere aufeinanderfolgende Textteile aufgeteilt. Sei # ein

Zeichen, das nicht in Σ vorkommt. Wir benutzen #, um diese Aufteilung in einer zu lernenden Sprache darzustellen. Dabei wird vor dem Anfang und nach dem Ende der zu extrahierenden Information ein # eingefügt.

Um den FLR-Wrapper aus n-markiertem Text zu lernen gehen wir wie folgt vor: Wir teilen unsere Lernbarkeitsaufgabe in individuelle Teilaufgaben auf. Nun können wir für jede Teilaufgabe ein Lernverfahren bilden, das diese Aufgabe unabhängig und parallel von den anderen löst. Zum Schluß führen wir die Lösungen der Teilaufgaben zusammen, um unsere Begrenzersprachen zu bestimmen und damit auch unseren FLR Wrapper.

Folgende drei Arten von Teilaufgaben müssen wir lösen:

6.2 Teilaufgabe 1

Die Folge $(u_{0_j})_{j\in\mathbb{N}}$ liefert Informationen über die linke Begrenzersprache L_1, da jedes u_{0_j} einen Suffix enthält, der zu L_1 gehört. Es ist jedoch nicht klar, wo dieser Suffix startet. Der n-markierte Text enthält aber noch weitere Informationen. Nach Definition gilt, dass kein echter Präfix des Suffixes existieren darf, der auch Element von L_1 ist. Weiterhin gilt, dass kein anderes Wort aus L_1 vor dem Suffix in u_{0_j} beginnen darf. Dieses Wort darf auch dann nicht vorkommen, wenn es in $v_{1_j}u_{1_j}...v_{n_j}u_{n_j}$ endet. Um eine Sprache für diese Teilaufgabe formal zu definieren, bilden wir eine Hilfskonstruktion. In dieser Hilfskonstruktion halten wir die Bedingungen der Sprache fest.

Definition 5 *Sei $\mathcal{L}$ eine indizierbare Familie rekursiver Sprachen und $G \in \mathcal{L}$. Dann definiert G die folgende Menge von Tupeln T^1_G. T^1_G ist die Menge aller (a, b, c), so dass folgende Bedingungen erfüllt sind:*

1. $a \in \Sigma^*$
2. $c \in \Sigma^+$
3. $b \in G$
4. $\neg\exists b' \in G$ *mit* $b' \sqsubset_p b$
5. $\neg\exists a' \in \Sigma^*, b' \in G$ *mit* $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$

$\triangle$

Die Folge $(u_{0_j}\#v_{1_j}u_{1_j}...v_{n_j}u_{n_j})_{j\in\mathbb{N}}$ stellt nun einen Text der Sprache vom Typ 1 da:

- $\mathcal{P}_1(G) = \{ab\#c \mid (a, b, c) \in T^1_G\}$

6.3 Teilaufgabe 2

Die Folge $(u_{n_j})_{j\in\mathbb{N}}$ liefert Informationen über die rechte Begrenzersprache R_n, da jedes u_{n_j} einen Präfix enthält, der zu R_n gehört. Hier wissen wir, dass nach Definition folgendes gilt: es darf kein anderes Wort aus R_n in v_{n_j} beginnen und es darf kein kleineres Wort geben, das an derselben Position, wie der Präfix startet und auch Element von R_n ist. Auch hier definieren wir erst eine Hilfskonstruktion, in der die Bedingungen der Sprache dieser Teilaufgabe bestimmt sind:

Definition 6 *Sei $\mathcal{L}$ eine indizierbare Familie rekursiver Sprachen und $G \in \mathcal{L}$. Dann definiert G die folgende Menge von Tupeln T^2_G. T^2_G ist die Menge aller (a, b, c), so dass folgende Bedingungen erfüllt sind:*

1. $c \in \Sigma^*$
2. $a \in \Sigma^+$
3. $b \in G$
4. $\neg\exists b' \in G$ *mit* $b' \sqsubset_p b$
5. $\neg\exists a' \in \Sigma^*, b' \in G$ *mit* $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$

$\triangle$

Die Folge $(v_{n_j}\#u_{n_j})_{j\in\mathbb{N}}$ stellt dann einen Text der Sprache vom Typ 2 da:

- $\mathcal{P}_2(G) = \{a\#bc \mid (a, b, c) \in T^2_G\}$

6.4 Teilaufgabe 3

Drittens, die Folge $(u_{1_j})_{j\in\mathbb{N}}$ z.B. liefert sowohl Informationen über die rechte Begrenzersprache R_1, als auch Informationen über die linke Begrenzersprache L_2, da u_{1_j} einen Präfix besitzt, der zu R_1 gehört und einen Suffix, der zu L_2 gehört. Wie bei den beiden anderen Lernbarkeitsaufgaben liefert das Dokument aber noch mehr Informationen. Nach Definition gilt, dass kein echter Präfix des Suffixes von L_2 existieren darf, der auch Element von L_2 ist. Weiterhin gilt, dass kein anderes Wort aus L_2 vor dem Suffix in u_{1_j} beginnen darf. Es darf kein anderes Wort aus R_1 in v_{1_j} beginnen und es darf kein kleineres Wort geben, das an derselben Position, wie der Präfix startet und auch Element von R_1 ist.

Folgende Hilfskonstruktion definiert die Bedingungen für die Sprache dieser Teilaufgabe:

Definition 7 *Sei $\mathcal{L}$ eine indizierbare Familie rekursiver Sprachen und $G \in \mathcal{L}$. Dann definiert G die folgende Menge von Tupeln T^3_G. T^3_G ist die Menge aller (a, b, c), so dass folgende Bedingungen erfüllt sind:*

1. $a, c \in \Sigma^+$
2. $b \in G$
3. $\neg\exists b' \in G$ *mit* $b' \sqsubset_p b$
4. $\neg\exists a' \in \Sigma^+, b' \in G$ *mit* $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$

$\triangle$

$(v_{1_j}\#u_{1_j}\#v_{2_j}...v_{n_j}u_{n_j})_{j\in\mathbb{N}}$ stellt einen Text der Sprache vom Typ 3 da:

- $\mathcal{P}_3(G, G') = \{a\#ba'b'\#c' \mid (a, b, a'b'c') \in T^3_G, (a', b', c') \in T^3_{G'}\}$

6.5 Lernbarkeitsaufgaben

Wertet man alle Informationen eines n-markierten Textes aus, dann ergeben sich n+1 Aufgaben des Lernens einer Sprache aus einem Text. Folgende Lernbarkeitsaufgaben sind nun von Interesse:

Definition 8 *Sei $\mathcal{L}$ eine indizierbare Familie rekursiver Sprachen. Für alle $i \in \{0, ..., 3\}$ ist $LP_i(\mathcal{L})$ die Aufgabe, die Klasse $\mathcal{P}_i(\mathcal{L})$ zu lernen. Sie kann gelöst werden, wenn $\mathcal{P}_i(\mathcal{L}) \in LimTxt$ ist. Dabei gilt:*

- $\mathcal{P}_0(\mathcal{L}) = \mathcal{L}$,
- $\mathcal{P}_1(\mathcal{L}) = \{\{ab\#c \mid (a, b, c) \in T^1_G\} \mid G \in \mathcal{L}\}$,
- $\mathcal{P}_2(\mathcal{L}) = \{\{a\#bc \mid (a, b, c) \in T^2_G\} \mid G \in \mathcal{L}\}$,
- $\mathcal{P}_3(\mathcal{L}) = \{\{a\#ba'b'\#c' \mid (a, b, a'b'c') \in T^3_G, (a', b', c') \in T^3_{G'}\} \mid G, G' \in \mathcal{L}\}$

$LP_0(\mathcal{L})$ dient dabei als Vergleichsaufgabe zu den Teilaufgaben $LP_1(\mathcal{L})$ bis $LP_3(\mathcal{L})$.

$\triangle$

7 Lernbarkeit bei Begrenzersprachen begrenzter Kardinalität

In diesem Abschnitt wollen wir die Lernbarkeit des Island-Wrapper und des FLR-Wrapper mit Begrenzersprachen begrenzter Kardinalität untersuchen.

Wir wissen, dass Island-Wrapper für Begrenzersprachen begrenzter Kardinalität lernbar sind (vgl. [GJLT00]).

Wir werden nun versuchen, analoges für den FLR-Wrapper zu beweisen. Wie wir aus Kapitel 6 wissen, können wir die Lernbarkeitsaufgabe des FLR-Wrappers in Teilaufgaben zerlegen. Angenommen, die Teilaufgaben für Begrenzersprachen begrenzter Kardinalität seien lösbar, dann ist auch die gesamte Lernbarkeitsaufgabe lösbar und damit der FLR-Wrapper mit Begrenzersprachen begrenzter Kardinalität lernbar. Nun sind nur noch die Teilaufgaben zu lösen. Dies werden wir auf den nachfolgenden Seiten tun.

7.1 Lernbarkeitsaufgabe 1

Theorem 1 $\mathcal{P}_1(\mathcal{L}_k) \in LimTxt$.

Beweis. Wir wissen, dass eine Sprache lernbar ist, wenn wir ein Lernverfahren angeben können, dessen Ausgabe gegen die gesuchte Sprache konvergiert. Wir wollen nun so ein Lernverfahren für $\mathcal{P}_1(\mathcal{L}_k)$ definieren und zeigen, dass es die Sprache lernt. Das Lernverfahren, das wir vorstellen, ist aus einem Lernverfahren für die Sprachklasse $G \circ \Sigma^+_{G'} \circ G'$ mit $G, G' \in \mathcal{L}_k$ abgeleitet (vgl. [GL, Thm.2]). Wir haben es an $\mathcal{P}_1(\mathcal{L}_k)$ angepasst. Im Folgenden sei $P_G = \{ab\#c \mid (a, b, c) \in T^1_G\}$ für eine Sprache $G \in \mathcal{L}_k$.

Sei $A \in \mathcal{L}_k$ und sei $t = (s_n \# s'_n)_{n \in \mathbb{N}}$ ein Text für P_A. Dann lernt folgende IIM die Sprache P_A, wenn sie mit t gespeist wird.

IIM M_1: Setze $H_{-1} = \{\emptyset\}$. Bei Eingabe t_x gehe wie folgt vor:

Initialisiere $H_x = \emptyset$ und sei E die Menge aller nicht leeren Suffixe von s_x. Sei H_{x-1} gegeben. Für alle $G \in H_{x-1}$ führe folgendes durch: Teste, ob ein $a \in \Sigma^*$ und ein $b \in G$ existiert, so dass gilt: $(a, b, s'_x) \in T^1_G$ mit $ab = s_x$. Ist dies der Fall, dann gehe zu (β1). Andernfalls gehe zu (β2).

(β1) Setze $H_x = H_x \cup \{G\}$.

(β2) Für alle $e \in E$ führe folgendes durch: Teste, ob ein $a \in \Sigma^*$ existiert, so dass gilt: $(a, e, s'_x) \in T^1_{G'}$ mit $ae = s_x$ und $G' = G \cup \{e\}$. Existiert so ein a nicht, dann bleibt H_x unverändert. Existiert so ein a, dann teste, ob $G' \in \mathcal{L}_k$. Ist dies nicht der Fall, dann bleibt H_x unverändert. Ist dies der Fall, dann setze $H_x = H_x \cup \{G'\}$.

Bestimme das lexikographisch erste A' in $\mathcal{H}$ mit $\mathcal{H} = \{P_G \mid G \in H_x\}$ und gebe es aus.

Jetzt müssen wir zeigen, dass die Ausgabe von M_1 gegen die korrekte Sprache konvergiert. Zwei Voraussetzungen sind, dass M_1 immer eine Hypothese ausgibt und die Ausgabe irgendwann konvergiert. Die Lemmata auf den nächsten Seiten bestätigen dies für M_1.

Lemma 1 *$\forall x \in \mathbb{N}$ gilt: $\exists G \in H_x$ mit $G \subseteq A$.*

Beweis. Wir beweisen dies durch vollständige Induktion.

Induktionsanfang: Am Anfang gilt $H_{-1} = \{\emptyset\}$. Sei $s_0 \# s'_0$ das erste Beispiel. Da $s_0 \# s'_0 \in P_A$ gilt, existiert in der Menge E aller Suffixe von s_0 genau ein Element e, für das gilt: $(a, e, s'_0) \in T_A^1$ und $ae = s_0$. (a, e, s'_0) erfüllt die Bedingung $(\beta 2)$ und es gilt deshalb $G \in H_0$ mit $\{e\} = G$ und $G \subseteq A$.

Induktionsschritt $(x \rightarrow x + 1)$: Angenommen, es gibt ein $G \in H_x$ mit $G \subseteq A$. Sei $s_{x+1} \# s'_{x+1}$ ein Beispiel für P_A. Da $s_{x+1} \# s'_{x+1} \in P_A$ gilt, existiert in der Menge E aller Suffixe von s_{x+1} genau ein Element e, für das gilt: $(a, e, s'_{x+1}) \in T_A^1$ und $ae = s_{x+1}$. Nach Definition von T_A^1 gilt damit $e \in A$. Es gibt nun 2 Fälle:

1. $e \in G$, dann ist wegen $(\beta 1)$ $G \in H_{x+1}$.
2. $e \notin G$. Existiert ein $e' \in G$ und $a' \in \Sigma^*$ mit $(a', e', s'_{x+1}) \in T_G^1$, dann ist wegen $(\beta 1)$ $G \in H_{x+1}$. Existiert so ein e' nicht, dann gilt wegen $(\beta 2)$ $G \cup \{e\} \in H_{x+1} \wedge G \cup \{e\} \subseteq A$.

Damit wäre bewiesen, dass für alle $x \in \mathbb{N}$ gilt: $\exists G \in H_x$ mit $G \subseteq A$.

■{Ende des Beweises von Lemma 1}

Lemma 2 *$(H_x)_{x \in \mathbb{N}}$ konvergiert.*

Beweis. Wir beweisen zuerst durch vollständige Induktion, dass H_x für alle x endlich ist. Induktionsanfang: H_0 hat nur endlich viele Elemente. Induktionsschritt$(x \rightarrow x + 1)$: Angenommen, H_x hat endlich viele Elemente. Da zu H_{x+1} nur endlich viele Elemente hinzugenommen werden, ist auch H_{x+1} endlich.

Das Hinzunehmen ist durch k beschränkt. So folgt aus Königs Lemma, dass $(H_x)_{x \in \mathbb{N}}$ konvergiert.

■{Ende des Beweises von Lemma 2}

Wir haben nun bewiesen, dass die Hypothesenmenge H_x immer mindestens ein Element enthält (Lemma 1) und die Ausgabe gegen eine Hypothese konvergiert (Lemma 2). Für die nun folgende Argumentation nehmen wir an, dass H_∞ die Menge ist gegen die H_x konvergiert und P_h die Sprache der

Hypothese h ist, gegen die die Ausgabe des Lernverfahrens M_1 konvergiert.

Mit den folgenden Lemmata zeigen wir, dass in der konvergierten Hypothesenmenge die korrekte Sprache enthalten ist und dass diese Sprache P_h ist.

Lemma 3 *$\forall h' \in H_\infty$ ist $P_A \subseteq P_{h'}$.*

Beweis. Nach der Definition des Lernverfahrens M_1 müssen alle Beispiele $s_x \# s'_x$ konsistent mit den Hypothesen sein. Nehmen wir aber das Gegenteil an: Es existiert ein $h' \in H_x$ und ein $w \in \Sigma^+$, so dass gilt: $w \in P_A \setminus P_{h'}$. Dann gibt es dieses w irgendwann als Beispiel. Somit ist $P_{h'}$ nicht konsistent mit allen Beispielen und h' kann nicht in der konvergierten Hypothesenmenge H_∞ enthalten sein.

■{Ende des Beweises von Lemma 3}

Lemma 4 *Sei $G \subseteq A$. Dann gilt $P_A \not\subset P_G$.*

Beweis. Angenommen, das Gegenteil gilt: $P_A \subset P_G$. D.h. aus $w \in P_A$ folgt $w \in P_G$. Weiterhin muss mindestens ein Wort w existieren, für das gilt: $w \in P_G \wedge w \notin P_A$. Sei $w = ab\#c$ so ein Wort, d.h. $(a, b, c) \in T_G^1$ und $(a, b, c) \notin T_A^1$. Aus der Definition von T_A^1 geht hervor, dass wegen $(a, b, c) \notin T_A^1$ eine der folgenden 5 Bedingungen nicht erfüllt sein kann:

(1) $a \in \Sigma^*$

(2) $c \in \Sigma^+$

(3) $b \in A$

(4) $\neg\exists b' \in A$ mit $b' \sqsubset_p b$

(5) $\neg\exists a' \in \Sigma^*, b' \in A$ mit $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$

Bedingung (1) und (2)

Dies gilt wegen $(a, b, c) \in T_G^1$ immer.

Bedingung (3)

Aus $(a, b, c) \in T_G^1$ folgt $b \in G$. Wegen $G \subseteq A$ gilt somit auch immer $b \in A$.

Bedingung (4)

Angenommen, es existieren $b' \in A$, so dass $b' \sqsubset_p b$ gilt. Dann bilden wir aus dem kürzesten b' ein Wort $w' = a'b'\#c'$ mit $a' = \varepsilon$ und $b'c' = b$. Aufgrund unserer Wahl von b' gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $w \in P_G$ folgt, dass $b' \notin G$ gilt. Deshalb muss eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a''b''\#c'$ mit $a'' \in \Sigma^*$ und $b'' \in G$ sei diese Zerlegung. Von ihr kann man folgendes sagen: (i) Wegen $a' = \varepsilon$ gilt $a' \sqsubset_p a''$. (ii) Es gilt $b \in G$ und $b'c' = b$. Aufgrund von $a'b'c' = a''b''c'$ ist $a'b = a''b''c'$ und damit auch $a'b \sqsubseteq_p a''b''c'$. Aus (i) und (ii) folgt $(a'', b'', c') \notin T^1_G$. Es existiert also keine andere Zerlegung und es gilt $w' \notin P_G$. Dies ist ein Widerspruch zu der Annahme, dass $P_A \subset P_G$ ist und somit gibt es kein $b' \in A$ mit $b' \sqsubset_p b$.

Bedingung (5)

Angenommen, es existieren $a' \in \Sigma^*, b' \in A$, so dass $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$ gilt. Wir betrachten das kürzeste dieser a' und das kürzeste b', das mit a' die Bedingung erfüllt. Dann gilt einer der 3 folgenden Fälle:

(a) $a'b' \sqsubset_p ab$

(b) $ab \sqsubset_p a'b'$

(c) $ab = a'b'$

> **Bedingung (5.a)** Angenommen, (a) gilt. Dann bilden wir das Wort $w' = a'b'\#c'$ mit $a'b'c' = ab$. Aufgrund unserer Wahl von a' und b' gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $w \in P_G$ folgt, dass $b' \notin G$. Es muss also eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a''b''\#c'$ mit $a'' \in \Sigma^*$ und $b'' \in G$ sei diese Zerlegung. Für a'' gibt es drei Möglichkeiten: (i) $a'' \sqsubset_p a$, (ii) $a \sqsubset_p a''$ oder (iii) $a'' = a$.
>
> Angenommen, (i) gilt. Wegen $a''b''c' = ab$ gilt außerdem $a''b'' \sqsubseteq_p abc$. Damit wäre nach Definition $(a, b, c) \notin T^1_G$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.
>
> Angenommen, (ii) gilt. Wegen $ab = a''b''c'$ gilt außerdem $ab \sqsubseteq_p a''b''c'$. Damit wäre nach Definition $(a'', b'', c'') \notin T^1_G$. Dies ist ein Widerspruch zu unserer Annahme, dass $w' \in P_G$ gilt.

Angenommen, (iii) gilt. Wegen $a''b''c' = ab$ gilt $b'' \sqsubset_p b$. Damit wäre nach Definition $(a, b, c) \notin T_G^1$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.

Es existieren also keine $a' \in \Sigma^*, b' \in A$ mit $a' \sqsubset_p a$ und $a'b' \sqsubset_p ab$.

Bedingung (5.b) Angenommen, (b) gilt. Dann bilden wir das Wort $w' = b'\#c'$ mit $c' \in \Sigma^+$. Aufgrund unserer Wahl von b' ist $(\varepsilon, b', c') \in T_A^1$ und es gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $w \in P_G$ folgt, dass $b' \notin G$ gilt. Es muss also eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a''b''\#c'$ mit $a'' \in \Sigma^*$ und $b'' \in G$ sei diese Zerlegung. Aus $b' \neq b''$ folgt $|a''| \geq 1$. Wegen $a' \sqsubset_p a$ und $ab \sqsubset_p a'b'$ wissen wir, dass ein $f \in \Sigma^*$ existiert, so dass $fb \sqsubset_p b'$ und damit auch $fb \sqsubset_p a''b''$ gilt. Weiterhin gilt $a'f = a$. Für a'' gibt es drei Möglichkeiten: (i) $a'' \sqsubset_p f$, (ii) $f \sqsubset_p a''$ oder (iii) $a'' = f$.

Angenommen, (i) gilt. Aus $a'f = a$ folgt $a'a'' \sqsubset_p a$. Wegen $a'b' \sqsubseteq_p abc$ und $a''b'' = b'$ gilt $a'a''b'' \sqsubseteq_p abc$. Damit wäre nach Definition $(a, b, c) \notin T_G^1$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.

Angenommen, (ii) gilt. Aus $fb \sqsubset_p a''b''$ folgt $fb \sqsubseteq_p a''b''$. Damit wäre nach Definition $(a'', b'', c') \notin T_G^1$. Dies ist ein Widerspruch zu unserer Annahme, dass $w' \in P_G$ gilt.

Angenommen, (iii) gilt. Wegen $fb \sqsubset a''b''$ gilt $b \sqsubset_p b''$. Dann wäre aber nach Definition $(a'', b'', c') \notin T_G^1$. Dies ist ein Widerpruch zu unserer Annahme, dass $w' \in P_G$ gilt.

Es existieren also keine $a' \in \Sigma^*, b' \in A$ mit $a' \sqsubset_p a$ und $ab \sqsubset_p a'b'$.

Bedingung (5.c) Angenommen (c) gilt. Dann ist $w = ab\#c = a'b'\#c$. Aufgrund unserer Wahl von a' und b' gilt: $(a', b', c) \in T_A^1$. Dies ist aber ein Widerspruch zu der Annahme, dass $w \notin P_A$ gilt. Es gibt also keine $a' \in \Sigma^*, b' \in A$ mit $a' \sqsubset_p a$ und $ab = a'b'$.

Somit existieren keine $a' \in \Sigma^*, b' \in A$, so dass $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$ gilt. Da alle 5 Bedingungen erfüllt sind, gilt $(a, b, c) \in T_A^1$ und damit auch $w \in P_A$. Dies ist ein Widerpruch zu unserer Annahme, dass $P_A \subset P_G$ gilt. Wir haben

somit bewiesen, dass $P_A \not\subset P_G$ für $G \subseteq A$ gilt.

■{Ende des Beweises von Lemma 4}

Aus Lemma 1 und Lemma 4 folgt, dass in der Hypothesenmenge ein P_G existiert mit $P_A \not\subset P_G$. Dank Lemma 3 wissen wir, dass für diese P_G gilt: $P_A \subseteq P_G$. Daraus folgt $P_G = P_A$. Dank Lemma 3 wissen wir außerdem, dass $P_A \subseteq P_h$ gilt. Daraus folgt $P_G \subseteq P_h$. Da das Lernverfahren immer die kleinste Sprache auswählt, gilt $P_G = P_h$. Damit gilt auch $P_h = P_A$. Das Lernverfahren M_1 konvergiert also gegen die richtige Sprache und es gilt $\mathcal{P}_1(\mathcal{L}_k) \in LimTxt$.

■{Ende des Beweises von Theorem 1}

7.2 Lernbarkeitsaufgabe 2

Theorem 2 $\mathcal{P}_2(\mathcal{L}_k) \in LimTxt$.

Beweis. Als Beweis werden wir ein Lernverfahren für $\mathcal{P}_2(\mathcal{L}_k)$ angeben und zeigen, dass es die Sprache lernt. Im Folgenden sei $P_G = \{a\#bc \mid (a,b,c) \in T^2_G\}$ für eine Sprache $G \in \mathcal{L}_k$.

Sei $A \in \mathcal{L}_k$ und sei $t = (s_n \# s'_n)_{n\in\mathbb{N}}$ ein Text für P_A. Dann lernt folgende IIM die Sprache P_A, wenn sie mit t gespeist wird.

IIM M_2: Setze $H_{-1} = \{\emptyset\}$. Bei Eingabe t_x gehe wie folgt vor:

Sei $H_x = \emptyset$ und sei E die Menge aller nicht leeren Präfixe von s'_x. Sei H_{x-1} gegeben. Für alle $G \in H_{x-1}$ führe folgendes durch: Teste, ob ein $c \in \Sigma^*$ und ein $b \in G$ existiert, so dass gilt: $(s_x, b, c) \in T^2_G$ mit $bc = s'_x$. Ist dies der Fall, dann gehe zu (β1). Andernfalls gehe zu (β2).

(β1) Setze $H_x = H_x \cup \{G\}$.

(β2) Für alle $e \in E$ führe folgendes durch: Teste, ob ein $c \in \Sigma^*$ existiert, so dass gilt: $(s_x, e, c) \in T^2_{G'}$ mit $ec = s'_x$ und $G' = G \cup \{e\}$. Existiert so ein c nicht, dann bleibt H_x unverändert. Existiert so ein c, dann teste, ob $G' \in \mathcal{L}_k$. Ist dies nicht der Fall, dann bleibt H_x unverändert. Ist dies der Fall, dann setze $H_x = H_x \cup \{G'\}$.

Bestimme das lexikographisch erste A' in $\mathcal{H}$ mit $\mathcal{H} = \{P_G \mid P_G \in H_x\}$ und gebe es aus.

Im Weiteren zeigen wir, dass die Ausgabe von M_2 gegen die korrekte Sprache konvergiert. Mit den folgenden zwei Lemmata beweisen wir zuerst, dass in der Hypothesenmenge von M_2 immer mindestens ein Element enthält und die Ausgabe irgendwann konvergiert.

Lemma 5 $\forall x \in \mathbb{N}$ *gilt:* $\exists G \in H_x \wedge G \subseteq A$.

Beweis. Wir beweisen dies durch vollständige Induktion.

Induktionsanfang: Am Anfang gilt $H_{-1} = \{\emptyset\}$. Sei $s_0 \# s'_0$ das erste Beispiel. Da $s_0 \# s'_0 \in P_A$ gilt, existiert in der Menge E aller Präfixe von s'_0 genau ein Element e, für das gilt: $(s_0, e, c) \in T_A^2$ und $ec = s'_0$. Da (s_0, e, c) die Bedingung $(\beta 2)$ erfüllt, existiert ein $G \in H_0$ mit $G = \{e\}$ und $G \subseteq A$.

Induktionsschritt $(x \rightarrow x + 1)$: Angenommen, es gibt ein $G \in H_x$ mit $G \subseteq A$. Sei $s_{x+1} \# s'_{x+1}$ ein Beispiel für P_A. Da $s_{x+1} \# s'_{x+1} \in P_A$ gilt, existiert in der Menge E aller Präfixe von s'_{x+1} genau ein Element e, für das gilt: $(s_{x+1}, e, c) \in T_A^2$ und $ec = s'_{x+1}$. Nach Definition von T_A^2 gilt damit $e \in A$. Es gibt nun 2 Fälle:

1. $e \in G$, dann ist wegen $(\beta 1)$ $G \in H_{x+1}$.
2. $e \notin G$. Existiert ein $e' \in G$ und $c' \in \Sigma^*$ mit $(s_{x+1}, e', c') \in T_G^1$ und $e'c' = s'_{x+1}$, dann ist wegen $(\beta 1)$ $G \in H_{x+1}$. Existiert so ein e' nicht, dann gilt wegen $(\beta 2)$ $G \cup \{e\} \in H_{x+1} \wedge G \cup \{e\} \subseteq A$.

■{Ende des Beweises von Lemma 5}

Lemma 6 $(H_x)_{x \in \mathbb{N}}$ *konvergiert.*

Beweis. Der Beweis erfolgt analog zum Beweis von Lemma 2.

■{Ende des Beweises von Lemma 6}

Nach Lemma 5 enthält H_x immer mindestens ein Element und nach Lemma 6 konvergiert die Ausgabe. Für die nun folgende Argumentation nehmen wir an, dass H_∞ die Menge ist gegen die H_x konvergiert und P_h die Sprache der Hypothese h ist, gegen die die Ausgabe des Lernverfahrens M_2 konvergiert.

Wir zeigen jetzt, dass in der konvergierten Hypothesenmenge die zu lernende Sprache enthalten ist und dass diese Sprache P_h ist.

Lemma 7 $\forall h' \in H_\infty$ *ist* $P_A \subseteq P_{h'}$.

Beweis. Der Beweis erfolgt analog zum Beweis von Lemma 3.

■{Ende des Beweises von Lemma 7}

Lemma 8 *Sei $G \subseteq A$. Dann gilt $P_A \not\subset P_G$.*

Beweis. Angenommen, das Gegenteil gilt: $P_A \subset P_G$. D.h. aus $w \in P_A$ folgt $w \in P_G$. Weiterhin muss mindestens ein Wort w existieren, für das gilt: $w \in P_G \wedge w \notin P_A$. Sei $w = a\#bc$ so ein Wort, d.h. $(a, b, c) \in T_G^2 \wedge (a, b, c) \notin T_A^2$. Aus der Definition der Sprache T_A^2 geht hervor, dass wegen $(a, b, c) \notin T_A^2$ eine der folgenden 5 Bedingungen nicht erfüllt sein darf:

(1) $a \in \Sigma^+$

(2) $c \in \Sigma^*$

(3) $b \in A$

(4) $\neg\exists b' \in A$ mit $b' \sqsubset_p b$

(5) $\neg\exists a' \in \Sigma^*, b' \in A$ mit $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$

Bedingung (1) und (2)

Dies gilt wegen $(a, b, c) \in T_G^2$ immer.

Bedingung (3)

Aus $(a, b, c) \in T_G^2$ folgt, dass $b \in G$ gilt. Wegen $G \subseteq A$ gilt somit auch immer $b \in A$.

Bedingung (4)

Angenommen, es existieren $b' \in A$, so dass $b' \sqsubset_p b$ gilt. Dann bilden wir aus dem kürzesten b' ein Wort $w' = a'\#b'c'$ mit $a' \in \Sigma$ und $c' = \varepsilon$. Für dieses Wort gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Da aus $w \in P_G$ folgt, dass $b' \notin G$ gilt, muss eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $a'\#b''c''$ mit $b'' \in G$ und $c'' \in \Sigma^*$ sei diese Zerlegung. Wegen $c' = \varepsilon$ und $b'c' = b''c''$ gilt $b'' \sqsubset_p b'$. Damit ist aber auch $b'' \sqsubset_p b$ und $(a, b, c) \notin T_G^2$. Dies ist ein Widerspruch zu der Annahme, dass $w \notin P_G$ gilt. Es gibt also keine andere Zerlegung und es gilt $w' \notin P_G$. Somit existiert kein $b' \in A$ mit $b' \sqsubset_p b$.

Bedingung (5)

Angenommen, es existieren $a' \in \Sigma^*, b' \in A$, so dass $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$ gilt. Wir betrachten das kürzeste dieser a' und das kürzeste b', das zusammen mit a' die Bedingung erfüllt. Wir bilden das Wort $w' = a'\#b'c'$ mit $c' = \varepsilon$.

Aufgrund unserer Wahl von a' und b' gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $w \in P_G$ folgt, dass $b' \notin G$ gilt. Es muss also eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'\#b''c''$ mit $c'' \in \Sigma^*$ und $b'' \in G$ sei diese Zerlegung. Von ihr kann man folgendes sagen: (i) $a' \sqsubset_p a$ (ii) Wegen $c' = \varepsilon$ und $b''c'' = b'c'$ gilt $b'' \sqsubset_p b'$ und damit auch $a'b'' \sqsubseteq_p abc$. Aus (i) und (ii) folgt $(a, b, c) \notin T_G^2$. Dies ist ein Widerspruch zu der Annahme, dass $w \in P_G$ gilt. Es gibt also keine andere Zerlegung und es gilt $w' \notin P_G$. Somit können keine $a' \in \Sigma^*$ und $b' \in A$ mit $a' \sqsubset_p a \wedge a'b' \sqsubseteq_p abc$ existieren.

Da alle 5 Bedingungen erfüllt sind, gilt $(a, b, c) \in T_A^2$ und damit auch $w \in P_A$. Dies ist ein Widerpruch zu unserer Annahme, dass $P_A \subset P_G$ gilt. Somit gilt für alle $G \subseteq A$: $P_A \not\subset P_G$.

■{Ende des Beweises von Lemma 8}

Aus Lemma 5 und Lemma 8 folgt, dass in der Menge der Sprachen der Hypothesen ein P_G existiert mit $P_A \not\subset P_G$. Dank Lemma 7 wissen wir, dass für diese P_G gilt: $P_A \subseteq P_G$. Daraus folgt $P_G = P_A$. Dank Lemma 7 wissen wir außerdem, dass $P_A \subseteq P_h$ gilt. Daraus folgt $P_G \subseteq P_h$. Da das Lernverfahren immer die kleinste Sprache auswählt, gilt $P_G = P_h$. Damit gilt auch $P_h = P_A$. Das Lernverfahren M_2 konvergiert also gegen die richtige Sprache und es gilt $\mathcal{P}_2(\mathcal{L}_k) \in LimTxt$.

■{Ende des Beweises von Theorem 2}

7.3 Lernbarkeitsaufgabe 3

Theorem 3 $\mathcal{P}_3(\mathcal{L}_k) \in LimTxt$.

Beweis. Als Beweis werden wir ein Lernverfahren für $\mathcal{P}_3(\mathcal{L}_k)$ angeben und zeigen, dass es die Sprache lernt. Im Folgenden sei $P_G = \{a_1\#b_1a_2b_2\#c_2 \mid (a_1, b_1, a_2b_2c_2) \in T^3_{G_1},\ (a_2, b_2, c_2) \in T^3_{G_2}\}$ für die Sprachen $G_1, G_2 \in \mathcal{L}_k$ und $G = (G_1, G_2)$.

Seien $A_1,\ A_2 \in \mathcal{L}_k$, $A = (A_1, A_2)$ und sei $t = (s_n\#s'_n\#s''_n)_{n\in\mathbb{N}}$ ein Text für P_A. Dann lernt folgende IIM die Sprache P_A, wenn sie mit t gespeist wird.

IIM M_3: Setze $H_{-1} = \{(\emptyset, \emptyset)\}$. Bei Eingabe t_x gehe wie folgt vor:

Sei $H_x = \emptyset$, sei E_1 die Menge aller nicht leeren Präfixe von s'_x und E_2 die Menge aller nicht leeren Suffixe von s'_x. Sei H_{x-1} gegeben. Für alle $(G_1, G_2) \in H_{x-1}$ führe folgendes durch: Teste, ob ein $a_2 \in \Sigma^+$, ein $b_1 \in G_1$ und ein $b_2 \in G_2$ existiert, so dass gilt: $(s_x, b_1, a_2b_2s''_x) \in T^3_{G_1} \wedge (a_2, b_2, s''_x) \in T^3_{G_2}$ mit $b_1a_2b_2 = s'_x$. Ist dies der Fall, dann gehe zu (β1). Andernfalls gehe zu (β2).

(β1) Setze $H_x = H_x \cup \{(G_1, G_2)\}$.

(β2) Für alle $e_1 \in E_1$, $e_2 \in E_2$ führe folgendes durch: Teste, ob ein $a_2 \in \Sigma^+$ existiert, so dass gilt: $(s_x, e_1, a_2e_2s''_x) \in T^3_{G'_1} \wedge (a_2, e_2, s''_x) \in T^+_{G'_2}$ mit $e_1a_2e_2 = s'_x$, $G'_1 = G_1 \cup \{e_1\}$ und $G'_2 = G_2 \cup \{e_2\}$. Existiert so ein a_2 nicht, dann bleibt H_x unverändert. Existiert so ein a_2, dann teste, ob $G'_1,\ G'_2 \in \mathcal{L}_k$. Ist dies nicht der Fall, dann bleibt H_x unverändert. Ist dies der Fall, dann setze $H_x = H_x \cup \{(G'_1, G'_2)\}$.

Bestimme das lexikographisch erste A' in $\mathcal{H}$ mit $\mathcal{H} = \{P_G \mid P_G \in H_x\}$ und gebe es aus.

Im Weiteren zeigen wir, dass die Ausgabe von M_3 gegen die korrekte Sprache konvergiert. Mit den folgenden zwei Lemmata beweisen wir zuerst, dass die konvergierte Hypothesenmenge von M_3 immer mindestens ein Element enthält und die Ausgabe irgendwann konvergiert.

Lemma 9 $\forall x \in \mathbb{N}$ *gilt:* $\exists (G_1, G_2) \in H_x \wedge G_1 \subseteq A_1 \wedge G_2 \subseteq A_2$.

Beweis. Wir beweisen dies durch vollständige Induktion.

Induktionsanfang: Am Anfang gilt $H_{-1} = \{(\emptyset, \emptyset)\}$. Sei $s_0 \# s_0' \# s_0''$ das erste Beispiel. Da $s_0 \# s_0' \# s_0'' \in P_A$ gilt, existiert in der Menge E_1 aller nicht leeren Präfixe von s_0' genau ein Element e_1 und in der Menge E_2 aller nicht leeren Suffixe von s_0' genau ein Element e_2, so dass gilt: $(s_0, e_1, a_2 e_2 s_0'') \in T^3_{A_1}$ $\wedge$ $(a_2, e_2, s_0'') \in T^3_{A_2}$ und $e_1 a_2 e_2 = s_0'$. Da die Zerlegung $s_0 \# e_1 a_2 e_2 \# s_0''$ die Bedingung ($\beta 2$) erfüllt, existiert ein $(G_1, G_2) \in H_0$ mit $\{e_1\} \in G_1$,$\{e_2\} \in G_2$ und $G_1 \subseteq A_1 \wedge G_2 \subseteq A_2$.

Induktionsschritt ($x \rightarrow x + 1$): Angenommen, es gibt ein $(G_1, G_2) \in H_x$ mit $G_1 \subseteq A_1$ und $G_2 \subseteq A_2$. Sei $s_{x+1} \# s_{x+1}' \# s_{x+1}''$ ein Beispiel für P_A. Da $s_{x+1} \# s_{x+1}' \# s_{x+1}'' \in P_A$ gilt, existiert in der Menge E_1 aller nicht leeren Präfixe von s_{x+1}' genau ein Element e_1 und in der Menge E_2 aller nicht leeren Suffixe von s_{x+1}' genau ein Element e_2, so dass gilt: $(s_{x+1}, e_1, a_2 e_2 s_{x+1}'') \in T^3_{A_1}$ $\wedge$ $(a_2, e_2, s_{x+1}'') \in T^3_{A_2}$ und $e_1 a_2 e_2 = s_{x+1}'$. Nach Definition von $T^3_{A_1}$ und $T^3_{A_2}$ gilt damit $e_1 \in A_1$ und $e_2 \in A_2$. Es gibt nun 2 Fälle:

1. $e_1 \in G_1 \wedge e_2 \in G_2$, dann ist wegen ($\beta 1$) $(G_1, G_2) \in H_{x+1}$.
2. $e_1 \notin G_1 \vee e_2 \notin G_2$. Existieren ein $e_1' \in G_1$, ein $e_2' \in G_2$ und ein $a_2' \in \Sigma^+$ mit $(s_{x+1}, e_1', a_2' e_2' s_{x+1}'') \in T^3_{G_1} \wedge (a_2', e_2', s_{x+1}'') \in T^3_{G_2}$ und $e_1' a_2' e_2' = s_{x+1}'$, dann gilt wegen ($\beta 1$) $(G_1, G_2) \in H_{x+1}$. Existiert so ein e_1' oder so ein e_2' nicht, dann ist wegen ($\beta 2$) $((G_1 \cup \{e_1\}), (G_2 \cup \{e_2\})) \in H_{x+1}$ mit $G_1 \cup \{e_1\} \subseteq A_1$ und $G_2 \cup \{e_2\} \subseteq A_2$.

■{Ende des Beweises von Lemma 9}

Lemma 10 $(H_x)_{x \in \mathbb{N}}$ *konvergiert.*

Beweis. Der Beweis erfolgt analog zum Beweis von Lemma 2.

■{Ende des Beweises von Lemma 10}

Nach Lemma 9 enthält H_x immer mindestens ein Element und nach Lemma 10 konvergiert die Ausgabe. Für die nun folgende Argumentation nehmen wir an, dass H_∞ die Menge ist gegen die H_x konvergiert und P_h die Sprache der Hypothese h ist, gegen die die Ausgabe des Lernverfahrens M_3 konvergiert.

Wir zeigen jetzt, dass in der konvergierten Hypothesenmenge die zu lernende Sprache enthalten ist und dass diese Sprache genau P_h ist.

Lemma 11 *$\forall h' \in (H_x)_{x\in\mathbb{N}}$ ist $P_A \subseteq P_{h'}$.*

Beweis. Der Beweis erfolgt analog zum Beweis von Lemma 3.

■{Ende des Beweises von Lemma 11}

Lemma 12 *Sei $G_1 \subseteq A_1 \wedge G_2 \subseteq A_2$, dann gilt $P_A \not\subset P_G$.*

Beweis. Angenommen, das Gegenteil gilt: $P_A \subset P_G$. D.h. aus $w \in P_A$ folgt $w \in P_G$. Weiterhin muss mindestens ein Wort w existieren, für das gilt: $w \in P_G \wedge w \notin P_A$. Sei $w = a_1\#b_1a_2b_2\#c_2$ so ein Wort, d.h. $(a_1, b_1, a_2b_2c_2) \in T^3_{G_1} \wedge (a_2, b_2, c_2) \in T^3_{G_2}$ und eine der folgenden zwei Bedingungen ist nicht erfüllt:

(A) $(a_1, b_1, a_2b_2c_2) \in T^3_{A_1}$

(B) $(a_2, b_2, c_2) \in T^3_{A_2}$

Bedingung (A)

Angenommen, $(a_1, b_1, a_2b_2c_2) \notin T^3_{A_1}$ gilt. Aus der Definition geht hervor, dass eine der folgenden 4 Bedingungen nicht erfüllt sein darf:

(1) $a_1, a_2b_2c_2 \in \Sigma^+$

(2) $b_1 \in A_1$

(3) $\neg\exists b'_1 \in A_1$ mit $b'_1 \sqsubset_p b_1$

(4) $\neg\exists a'_1 \in \Sigma^+, b'_1 \in A_1$ mit $a'_1 \sqsubset_p a_1 \wedge a'_1b'_1 \sqsubseteq_p a_1b_1a_2b_2c_2$

Bedingung (A.1)

Dies gilt wegen $(a_1, b_1, a_2b_2c_2) \in T^3_{G_1}$ immer.

Bedingung (A.2)

Aus $(a_1, b_1, a_2b_2c_2) \in T^3_{G_1}$ folgt, dass $b_1 \in G_1$ gilt. Wegen $G_1 \subseteq A_1$ gilt somit auch immer $b_1 \in A_1$.

Bedingung (A.3)

Angenommen, es existieren $b'_1 \in A_1$, so dass $b'_1 \sqsubset_p b_1$ gilt. Dann bilden wir aus dem kürzesten b'_1 ein Wort $w' = a'_1\#b'_1a'_2b'_2\#c'_2$ mit $a'_1 \in \Sigma$. $a'_2 \in \Sigma^+$ mit $|a'_2| = 1$, $b'_2 \in A_2$ und $c'_2 \in \Sigma^+$ sind so gewählt, dass kein echter Suffix e von b'_2 existiert mit $e \in A_2$ und dass $(a'_2, b'_2, c'_2) \in T^3_{A_2}$ gilt. Aufgrund unserer Wahl von a'_1 und b'_1 ist außerdem $(a'_1, b'_1, a'_2b'_2c'_2) \in T^3_{A_1}$. Für dieses Wort gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Da aus $(a_1, b_1, a_2b_2c_2) \in T^3_{G_1}$ folgt, dass $b'_1 \notin G_1$ gilt, muss eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'_1\#b''_1a''_2b''_2\#c'_2$ mit $b''_1 \in G_1$, $a''_2 \in \Sigma^+$ und $b''_2 \in G_2$ sei diese Zerlegung. Für b''_1 gibt es zwei Möglichkeiten: (a) $b''_1 \sqsubseteq_p b'_1$ oder (b) $b'_1 \sqsubset_p b''_1$.

Angenommen, (a) gilt. Dann ist wegen $b'_1 \sqsubset_p b_1$ auch $b''_1 \sqsubset_p b_1$. Damit ist nach Definition $(a_1, b_1, a_2b_2c_2) \notin T^3_{G_1}$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.

Angenommen, (b) gilt. Wegen $|a'_2| = 1$ gilt dann $b''_2 \sqsubset_s b'_2$. Da keine echten Suffixe e von b'_2 existieren mit $e \in A_2$ und $G_2 \subseteq A_2$, ist $b''_2 \notin G_2$. Damit gilt aber $(a''_2, b''_2, c''_2) \notin T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w' \in P_G$ gilt.

Es gibt also keine andere Zerlegung und $w' \notin P_G$ gilt. Da das ein Widerspruch zu unserer Annahme ist, dass $P_A \subset P_G$ gilt, existiert kein $b'_1 \in A_1$ mit $b'_1 \sqsubset_p b_1$.

Bedingung (A.4)

Angenommen, es existieren $a'_1 \in \Sigma^+, b'_1 \in A_1$ mit $a'_1 \sqsubset_p a_1$ und $a'_1b'_1 \sqsubseteq_p a_1b_1a_2b_2c_2$. Wir betrachten das kürzeste dieser a'_1 und das kürzeste b'_1, das zusammen mit a'_1 die Bedingung erfüllt. Nun bilden wir das Wort $w' = a'_1\#b'_1a'_2b'_2\#c'_2$ mit $a_1b_1 \sqsubseteq_p a'_1b'_1a'_2$. $b'_2 \in A_2$ und $c'_2 \in \Sigma^+$ sind so gewählt, dass $(a'_2, b'_2, c'_2) \in T^3_{A_2}$ gilt. Aufgrund unserer Wahl von a'_1 und b'_1 ist außerdem $(a'_1, b'_1, a'_2b'_2c'_2) \in T^3_{A_1}$. Für dieses Wort gilt $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Da aus $(a_1, b_1, a_2b_2c_2) \in T^3_{G_1}$ folgt, dass $b'_1 \notin G_1$ gilt, muss eine andere Zerlegung für w' existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'_1\#b''_1a''_2b''_2\#c'_2$ sei so eine Zerlegung. Von ihr kann man folgendes sagen: (i) $a'_1 \sqsubset_p a_1$ (ii) $a'_1b''_1 \sqsubseteq_p a_1b_1a_2b_2c_2$. Aus (i) und (ii) folgt $(a_1, b_1, a_2b_2c_2) \notin T^3_{G_1}$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$. Es gibt also keine Zerlegung und $w' \notin P_G$ gilt. Somit existieren keine $a'_1 \in \Sigma^+, b'_1 \in A_1$ mit $a'_1 \sqsubset_p a_1$ und $a'_1b'_1 \sqsubseteq_p a_1b_1a_2b_2c_2$.

Da alle 4 Bedingungen erfüllt sind, gilt $(a_1, b_1, a_2b_2c_2) \in T^3_{A_1}$.

Bedingung (B)

Angenommen, $(a_2, b_2, c_2) \notin T^3_{A_2}$ gilt. Aus der Definition geht hervor, dass eine der folgenden 4 Bedingungen nicht erfüllt sein darf:

(1) $a_2, c_2 \in \Sigma^+$

(2) $b_2 \in A_2$

(3) $\neg\exists b'_2 \in A_2$ mit $b'_2 \sqsubset_p b_2$

(4) $\neg\exists a'_2 \in \Sigma^+, b'_2 \in A_2$ mit $a'_2 \sqsubset_p a_2 \wedge a'_2b'_2 \sqsubseteq_p a_2b_2c_2$

Bedingung (B.1)

Dies gilt wegen $(a_2, b_2, c_2) \in T^3_{G_2}$ immer.

Bedingung (B.2)

Aus $(a_2, b_2, c_2) \in T^3_{G_2}$ folgt, dass $b_2 \in G_2$ gilt. Wegen $G_2 \subseteq A_2$ gilt somit auch immer $b_2 \in A_2$.

Bedingung (B.3)

Angenommen, es exstieren $b'_2 \in A_2$, so dass $b'_2 \sqsubset_p b_2$ gilt. Dann bilden wir aus dem kürzesten b'_2 ein Wort $w' = a'_1\#b_1a'_2b'_2\#c'_2$ mit $a'_1 \in \Sigma$, $a'_2 \in \Sigma$ und $b'_2c'_2 = b_2$. Wir haben oben bewiesen, dass $(a_1, b_1, a_2b_2c_2) \in T^3_{A_1}$ gilt (siehe Abschnitt (A)). Damit ist $b_1 \in A_1$ und es existiert kein Präfix e von b_1 mit $e \in A_1$. Da $|a'_1| = 1$ gilt, ist $(a'_1, b_1, a'_2b'_2c'_2) \in T^3_{A_1}$. Aufgrund unserer Wahl von a'_2 und b'_2 gilt außerdem $(a'_2, b'_2, c'_2) \in T^3_{A_2}$. Somit ist $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Da aus $(a_2, b_2, c_2) \in T^3_{G_2}$ folgt, dass $b'_2 \notin G_2$ gilt, muss eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'_1\#b''_1a''_2b''_2\#c'_2$ mit $b''_1 \in G_1$, $a''_2 \in \Sigma^+$ und $b''_2 \in G_2$ sei diese Zerlegung. Da $b_1 \in G_1$ gilt und kein Präfix e von b_1 existiert mit $e \in G_1$, ist nach Definition von $T^3_{G_1}$ $b''_1 = b_1$ und damit $a''_2b''_2 = a'_2b'_2$. Folgendes läßt sich nun über die Zerlegung sagen: (i) Aufgrund von $|a'_2| = 1$ gilt $a'_2 \sqsubset_p a''_2$ (ii) Wegen $a''_2b''_2 = a'_2b'_2$ ist $a'_2b'_2c'_2 \sqsubseteq_p a''_2b''_2c'_2$. Außerdem ist $b'_2c'_2 = b_2$. Deshalb gilt $a'_2b_2 \sqsubseteq_p a''_2b''_2c'_2$. Aus (i) und (ii) folgt $(a''_2, b''_2, c'_2) \notin T^3_{G_2}$. Es gibt also keine Zerlegung und $w' \notin P_G$. Dies ist ein Widerspruch zu unsere Annahme, dass $P_A \subset P_G$ gilt. Somit existieren keine $b'_2 \in A_2$ mit $b'_2 \sqsubset_p b_2$.

Bedingung (B.4)

Angenommen, $\exists a'_2 \in \Sigma^+, b'_2 \in A_2$ mit $a'_2 \sqsubset_p a_2 \wedge a'_2 b'_2 \sqsubseteq_p a_2 b_2 c_2$ gilt. Wir betrachten das kürzeste dieser a'_2 und das kürzeste b'_2, das zusammen mit a'_2 die Bedingung erfüllt. Dann gilt einer der 3 folgenden Fälle:

(a) $a'_2 b'_2 \sqsubset_p a_2 b_2$

(b) $a_2 b_2 \sqsubset_p a'_2 b'_2$

(c) $a_2 b_2 = a'_2 b'_2$

Bedingung (B.4.a) Angenommen, (a) gilt. Dann bilden wir das Wort $w' = a'_1 \# b_1 a'_2 b'_2 \# c'_2$ mit $a'_1 \in \Sigma$ und $a'_2 b'_2 c'_2 = a_2 b_2$. Wir haben oben bewiesen, dass $(a_1, b_1, a_2 b_2 c_2) \in T^3_{A_1}$ gilt (siehe dazu Abschnitt (A)). Damit ist $b_1 \in A_1$ und es existiert kein Präfix e von b_1 mit $e \in A_1$. Da $|a'_1| = 1$ gilt, ist $(a'_1, b_1, a'_2 b'_2 c'_2) \in T^3_{A_1}$. Aufgrund unserer Wahl von a'_2 und b'_2 gilt außerdem $(a'_2, b'_2, c'_2) \in T^3_{A_2}$ und damit $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $(a_2, b_2, c_2) \in T^3_{G_2}$ folgt, dass $a'_2 \notin G_2$ gilt. Es muss also eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'_1 \# b''_1 a''_2 b''_2 \# c'_2$ mit $b''_1 \in G_1$, $a''_2 \in \Sigma^+$ und $b''_2 \in G_2$ sei diese Zerlegung. Da $b_1 \in G_1$ gilt und kein Präfix e von b_1 existiert mit $e \in G_1$, ist nach Definition von $T^3_{G_1}$ $b''_1 = b_1$. Dann ist $a''_2 b''_2 = a'_2 b'_2$ und damit $a''_2 b''_2 c'_2 = a_2 b_2$. Für b''_2 gibt es drei Möglichkeiten: (i) $b''_2 c'_2 = b_2$, (ii) $b''_2 c'_2 \sqsubset_s b_2$ oder (iii) $b_2 \sqsubset_s b''_2 c'_2$.

Angenommen, (i) gilt. Dann folgt daraus $b''_2 \sqsubset_p b_2$. Damit wäre nach Definition $(a_2, b_2, c_2) \notin T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.

Angenommen, (ii) gilt. Da außerdem $a''_2 b''_2 c'_2 = a_2 b_2$ gilt, ist $a_2 \sqsubset_p a''_2$ und $a_2 b_2 \sqsubseteq_p a''_2 b''_2 c'_2$. Damit wäre $(a''_2, b''_2, c'_2) \notin T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w' \in P_G$ gilt.

Angenommen, (iii) gilt. Da außerdem $a''_2 b''_2 c'_2 = a_2 b_2$ gilt, ist $a''_2 \sqsubset_p a_2$ und $a''_2 b''_2 \sqsubseteq_p a_2 b_2 c_2$. Nach Definition ist deshalb $(a_2, b_2, c_2) \notin T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$ gilt.

Es gibt also keine andere Zerlegung und $w' \notin P_G$ gilt. Dies ist ein Widerpruch zu der Annahme, dass $P_A \subset P_G$ ist. Es existieren

somit keine $a'_2 \in \Sigma^+$, $b'_2 \in A_2$ mit $a'_2 \sqsubset_p a_2$ und $a'_2 b'_2 \sqsubset_p a_2 b_2$.

Bedingung (B.4.b) Angenommen, (b) gilt. Dann bilden wir das Wort $w' = a'_1 \# b_1 a'_2 b'_2 \# c'_2$ mit $a'_1 \in \Sigma$ und $a'_2 b'_2 c'_2 = a_2 b_2 c_2$. Wir haben oben bewiesen, dass $(a_1, b_1, a_2 b_2 c_2) \in T^3_{A_1}$ gilt (siehe dazu Abschnitt (A)). Damit ist $b_1 \in A_1$ und es existiert kein Präfix e von b_1 mit $e \in A_1$. Da $|a'_1| = 1$ gilt, ist $(a'_1, b_1, a'_2 b'_2 c'_2) \in T^3_{A_1}$. Aufgrund unserer Wahl von a'_2 und b'_2 gilt außerdem $(a'_2, b'_2, c'_2) \in T^3_{A_2}$ und damit $w' \in P_A$. Angenommen, es gelte $w' \in P_G$. Aus $(a_2, b_2, c_2) \in T^3_{G_2}$ folgt, dass $b'_2 \notin G_2$ gilt. Es muss also eine andere Zerlegung existieren, die die Bedingungen der Sprache erfüllt. Angenommen, $w' = a'_1 \# b''_1 a''_2 b''_2 \# c'_2$ mit $b''_1 \in G_1$, $a''_2 \in \Sigma^+$ und $b''_2 \in G_2$ sei diese Zerlegung. Da $b_1 \in G_1$ gilt und kein Präfix e von b_1 existiert mit $e \in G_1$, ist nach Definition von $T^3_{G_1}$ $b''_1 = b_1$. Dann ist $a''_2 b''_2 = a'_2 b'_2$ und damit $a_2 b_2 \sqsubset_p a''_2 b''_2$. Für b''_2 gibt es drei Möglichkeiten: (i) $\exists z_1, z_2 \in \Sigma^+$ mit $b''_2 = z_1 b_2 z_2$, (ii) $\exists z_2 \in \Sigma^+$ mit $b''_2 = b_2 z_2$ oder (iii) $\exists z''_1, z_2 \in \Sigma^+$ mit $z''_1 b''_2 = b_2 z_2$.

Angenommen, (i) gilt. Wegen $a_2 b_2 \sqsubset_p a''_2 b''_2$ gilt dann $a''_2 \sqsubset_p a_2$. Aus $a''_2 b''_2 = a'_2 b'_2$ und $a'_2 b'_2 \sqsubseteq_p a_2 b_2 c_2$ folgt außerdem $a''_2 b''_2 \sqsubseteq_p a_2 b_2 c_2$. Damit wäre nach Definition $(a_2, b_2, c_2) \in T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w \in P_G$.

Angenommen, (ii) gilt. Dann gilt $b_2 \sqsubset_p b''_2$. Damit wäre nach Definition $(a''_2, b''_2, c'_2) \in T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w' \in P_G$.

Angenommen, (iii) gilt. Wegen $a_2 b_2 \sqsubset_p a''_2 b''_2$ gilt dann $a_2 \sqsubset_p a''_2$ und $a_2 b_2 \sqsubseteq_p a''_2 b''_2 c'_2$. Damit wäre nach Definition $(a''_2, b''_2, c'_2) \in T^3_{G_2}$, was ein Widerspruch zu unserer Annahme ist, dass $w' \in P_G$.

Es gibt also keine andere Zerlegung und $w' \notin P_G$ gilt. Dies ist ein Widerpruch zu der Annahme, dass $P_A \subset P_G$ ist. Es existieren somit keine $a'_2 \in \Sigma^+$, $b'_2 \in A_2$ mit $a'_2 \sqsubset_p a_2$ und $a_2 b_2 \sqsubset_p a'_2 b'_2$.

Bedingung (B.4.c) Angenommen, (c) gilt. Dann ist $w = a_1 \# b_1 a_2 b_2 \# c_2 = a_1 \# b_1 a'_2 b'_2 \# c_2$ und es gilt $(a'_2, b'_2, c_2) \in T^3_{A_2}$. Dies ist aber ein Widerspruch zu unserer Annahme, dass $w \notin P_A$. Es gibt also keine andere Zerlegung und $w' \notin P_G$ gilt. Dies ist ein

> Widerpruch zu der Annahme, dass $P_A \subset P_G$ ist. Es existieren somit keine $a'_2 \in \Sigma^+$, $b'_2 \in A_2$ mit $a'_2 \sqsubset_p a_2$ und $a_2 b_2 = a'_2 b'_2$.

Da alle 4 Bedingungen erfüllt sind, gilt $(a_2, b_2, c_2) \in T^3_{A_2}$.

Wegen $(a_1, b_1, a_2 b_2 c_2) \in T^3_{A_1}$ und $(a_2, b_2, c_2) \in T^3_{A_2}$ gilt $w \in P_A$. Dies ist aber ein Widerpruch zu unserer Annahme, dass $P_A \subset P_G$ gilt. Somit gilt für alle $G_1 \subseteq A_1$ und $G_2 \subseteq A_2$: $P_A \not\subset P_G$.

■{Ende des Beweises von Lemma 12}

Aus Lemma 10 und Lemma 12 folgt, dass in der Menge der Hypothesen ein P_G existiert mit $P_A \not\subset P_G$. Dank Lemma 11 wissen wir, dass für diese P_G gilt: $P_A \subseteq P_G$. Daraus folgt $P_G = P_A$. Dank Lemma 11 wissen wir außerdem, dass $P_A \subseteq P_h$ gilt. Daraus folgt $P_G \subseteq P_h$. Da das Lernverfahren immer die kleinste Sprache auswählt, gilt $P_G = P_h$. Damit gilt auch $P_h = P_A$. Das Lernverfahren M_3 konvergiert also gegen die richtige Sprache und es gilt $\mathcal{P}_3(\mathcal{L}_k) \in LimTxt$.

■{Ende des Beweises von Theorem 3}

8 Fazit und Ausblick

In unserer Studienarbeit haben wir eine neue Wrapperklasse eingeführt: den Fixed-Left-Right-Wrapper. Dabei sind wir von einer in Pseudo-Code angegebenen Extraktionsprozedur ausgegangen und haben die Extraktionsmechanismen, die der FLR-Wrapper bestimmt, formal definiert. Als Vorlage für die Prozedur diente dabei die Extraktionsprozedur des LR-Wrappers (vgl. [Kus97, Kus00]).

Der FLR-Wrapper-Klasse haben wir außerdem eine andere Klasse gegenübergestellt, die sogenannte Island-Wrapper-Klasse. Gemeinsam haben beide, dass sie Textteile extrahieren, die durch linke und rechte Begrenzer eingepackt sind. Beide Klassen sind vollständig durch ihre Begrenzersprachen definiert. Es gibt jedoch zwei große Unterschiede.

Erstens, die Wrapper definieren verschiedene Extraktionsmechanismen. Ist ein erstes Tupel mithilfe des FLR-Wrappers gefunden, dann wird im Dokument nach dem Ende dieses Tupels nach weiteren gesucht. Die extrahierten Tupel überschneiden sich also nicht in der Position im Dokument. Wird aber mithilfe des Island-Wrappers Informationen extrahiert, dann können sich die Tupel in der Position im Dokument aufgrund der Eigenschaften der Extraktionsmechanismen des Island-Wrappers überschneiden.

Zweitens, wegen den unterschiedlichen Extraktionsmechanismen werden auch unterschiedliche Bedingungen an die Begrenzer der Wrapper gestellt. Dabei sind die Bedingungen an die Begrenzer des FLR-Wrappers strenger. So darf in einem bestimmten Abschnitt vor einem Begrenzer kein Teilwort beginnen, das Element der Sprache des Begrenzers ist. Weiterhin darf kein Präfix eines Begrenzers existieren, der auch Element derselben Begrenzersprache ist. Für die Begrenzer des Island-Wrappers gelten diese Bedingungen nicht. Die einzige Bedingung ist, dass der bestimmte Abschnitt vor einem Begrenzer kein Teilwort der Sprache des Begrenzers enthält. Deshalb kann es vorkommen, dass mithilfe des Island-Wrappers mehr Tupel extrahiert werden als mithilfe des FLR-Wrappers.

Weiterhin haben wir für ein speziellen Fall gezeigt, dass der FLR-Wrapper lernbar ist und untersucht, ob sich die Unterschiede des FLR-Wrappers gegenüber dem Island-Wrapper auf die Lernbarkeit ausgewirkt haben. Für Begrenzersprachen begrenzter Kardinalität ergab sich dabei kein Unterschied. Sowohl der Island-Wrapper als auch der FLR-Wrapper sind mit diesen Begrenzersprachen lernbar.

Ob sich hinsichtlich der Lernbarkeit der Wrapper mit anderen Begrenzersprachen Unterschiede ergeben, müßte in zukünftigen Untersuchungen gezeigt werden. Da es sich bei der FLR-Wrapper-Klasse um ein theoretisches Modell handelt, stellt sich auch die Frage, wie eine Anwendung des Modells in der Praxis aussehen würde. Dafür ist es möglicherweise nötig, die Klasse anzupassen.

Literatur

[GJLT00] Grieser, Gunter, Klaus P. Jantke, Steffen Lange und Bernd Thomas: *A Unifying Approach to HTML Wrapper Representation and Learning.* In: Arikawa, S. und S. Morishita (Herausgeber): *Proc. 3rd International Conference on Discovery Science*, Band 1967 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 50–64, Berlin, 2000. Springer-Verlag.

[GL] Grieser, Gunter und Steffen Lange: *Interne Kommunikation. lernergebnisse.tex.*

[GL01] Grieser, Gunter und Steffen Lange: *Learning Approaches to Wrapper Induction.* In: Russel, I. und J. Kolen (Herausgeber): *Proc. 14th International FLAIRS Conference*, Seiten 249–253. AAAI-Press, 2001.

[Gol67] Gold, Mark E.: *Language identification in the limit.* Information and Control, 14:447–474, 1967.

[Kus97] Kushmerick, Nicholas: *Wrapper Induction for Information extraction.* Doktorarbeit, University of Washington, 1997.

[Kus00] Kushmerick, Nicholas: *Wrapper Induction: Efficiency and expressiveness.* Artificial Intelligence, 118(1-2):15–68, 2000.