Theorie des Algorithmischen Lernens
Sommersemester 2007

Teil 2.3: Lernen formaler Sprachen: Patternsprachen

Version 1.1

Gliederung der LV

Patternsprachen

Alphabet Σ und aufzählbare Menge X von ***Variablen***, $\Sigma \cap X = \emptyset$

Ein ***Pattern*** ist ein String $\pi \in (\Sigma \cup X)^+$

eine ***(non-erasing) Substitution*** σ ist eine Abbildung von $X \to \Sigma^+$

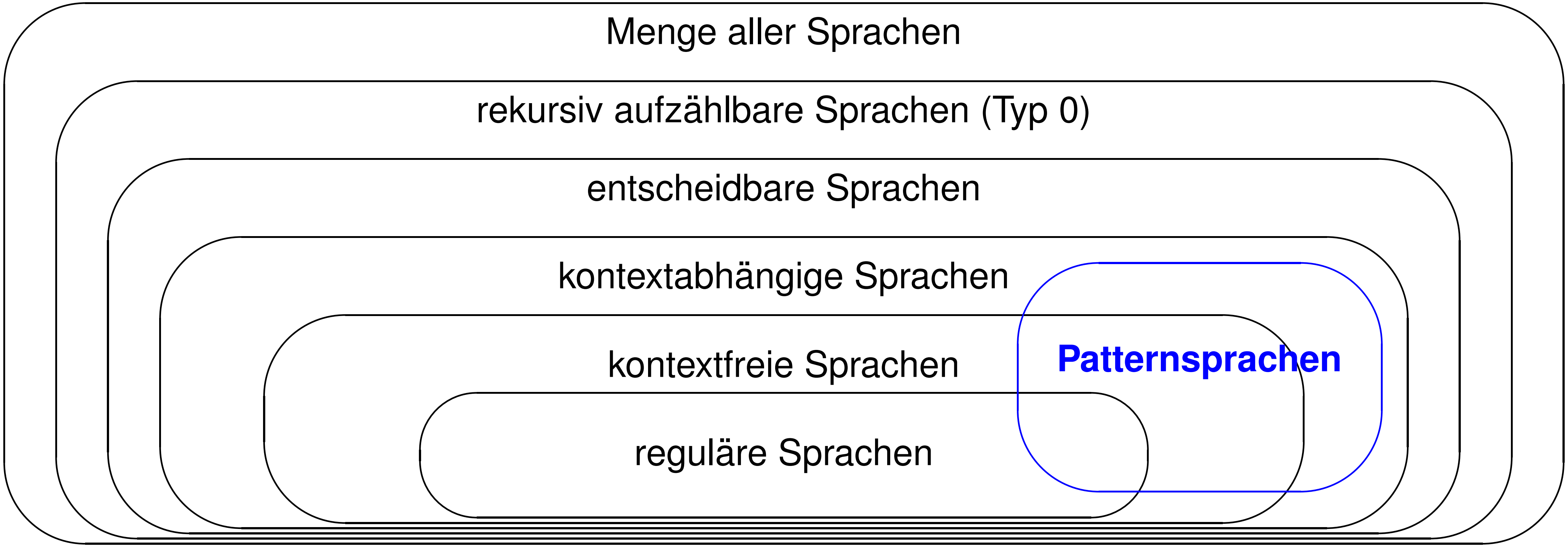
- Kanonische Erweiterung von Substitutionen auf Patterns

$\boldsymbol{L(\pi)} = \{w \mid w \in \Sigma^+$ und es gibt eine Substitution σ so daß $\sigma(\pi) = w\}$

Patternsprache: Sprache durch Pattern beschreibbar

PAT: Menge aller Patternsprachen

Einordnung in die Chomsky-Hierarchie



Offenes Problem: $(L_{pattern} \cap L_{cf}) \setminus L_{reg} = \emptyset$?

Beobachtungen

- Patternsprachen sind entweder einelementig oder unendlich
- Falls nur Substitutionen durch nichtleere Zeichenketten erlaubt sind, gibt es nur *endlich* viele Patterns (*bis auf Umbenennung der Variablen*), die w erzeugen können.
  - Begriff: ***Kanonisches Pattern***:
    - ∗ Variablen sind durchnumeriert
    - ∗ Wenn an einer Stelle die Variable x_{i+1} vorkommt, dann kommt links davon (vorher) die Variable x_i vor.
  - Falls auch Substitutionen durch leere Zeichenketten erlaubt sind, gibt es unendlich viele Patterns, die w erzeugen können.
- Eine Patternsprache ist eindeutig durch *die Menge ihrer kürzesten Wörter* bestimmt.

Ein erster Lernalgorithmus

Theorem 2.3.1:
$\mathit{PAT} \in \mathit{LimTxt}$

Übungsaufgabe: Geben Sie die Telltalemengen an.

Ein erster Lernalgorithmus

Proof.

Definition 2.3.1:
Ein Pattern π heißt ***beschreibend*** für eine Menge S von Wörtern, falls gilt:

- π ist konsistent mit S, d.h. $S \subseteq L(\pi)$.
- Es gibt kein Pattern π' mit $S \subseteq L(\pi') \subset L(\pi)$.

$M(t_x)$: Wenn $x = 0$ gehe zu (*). Ansonsten teste, ob $t_x^+ \subseteq L(M(t_{x-1}))$ gilt. Wenn ja, gib $M(t_{x-1})$ aus, sonst gehe zu (*).

(*) Berechne ein beschreibendes Pattern von t_x^+ und gib es aus.

Verifikation $\longrightarrow$ Übungsaufgabe

Ein erster Lernalgorithmus

Übungsaufgabe: Wie könnte ein Algorithmus zur Berechnung beschreibender Patterns aussehen?

Theorem 2.3.2:
Die Berechnung von beschreibenden Patterns ist NP-hart.

Der Algorithmus von Lange und Wiehagen

Beobachtung: Eine Patternsprache ist eindeutig durch *die Menge ihrer kürzesten Wörter* bestimmt.

Sei der Text $t = w_0, w_1, w_2, \ldots$ gegeben.

$M(t_0) = w_0$

$$M(t_{x+1}) = \begin{cases} M(t_x) & : |w_{x+1}| > M(t_x) \\ w_{x+1} & : |w_{x+1}| < M(t_x) \\ join(M(t_x), w_{x+1}) & : \text{sonst} \end{cases}$$

$join(\pi, w)$: For $j = 1, 2, \ldots |w|$:

- Wenn $w[j] = \pi[j]$, dann setze $\pi'[j] = \pi[j]$.
- Wenn $w[j] \neq \pi[j]$ und die Kombination $(w[j], \pi[j])$ kommt in π' bereits vor (sagen wir an Stelle j'), dann setze $\pi'[j] = \pi'[j']$.
- Wenn $w[j] \neq \pi[j]$ und die Kombination $(w[j], \pi[j])$ kommt noch nicht in π' vor, dann setze $\pi'[j]$ auf die Variable X_j.

Gib π' aus.

($w[j]$ meint den $j$en Buchstaben von w)

Diskussion

Übungsaufgabe:

- In welchen Fällen ist die Hypothese stets konsistent, an welchen stets inkonsistent?
- Arbeitet der Algorithmus richtig? $\longrightarrow$ Verifikation
- Wie ist das Laufzeitverhalten?

Diskussion

Theorem 2.3.3:
Der Algorithmus von Lange und Wiehagen lernt *PAT* ***inkrementell*** im Limes, ist nicht konsistent und arbeitet in *Polynomialzeit*.

Übungsaufgabe: Wie kann man den Algorithmus umgestalten, so daß er konsistent und trotzdem inkrementell arbeitet?

Reguläre Patterns

Definition 2.3.2:
Ein Pattern heißt ***regulär*** genau dann wenn jede Variable nur einmal auftritt.

rPAT: Menge aller regulären Patternsprachen.

Lernalgorithmus für reguläre Patternsprachen:

$M(t_x)$:
Sei w ein kürzestes Wort in t_x und k dessen Länge. Setze π auf das Pattern $X_1 X_2 \cdots X_k$.
Für $j = 1, \ldots, k$:

- Erzeuge π', indem die je Variable in π durch den $j$en Buchstaben von w ersetzt wird. Wenn $t_x^+ \subseteq L(\pi')$, dann setze π auf π'.

Gib π aus.

Reguläre Patterns

Analyse:

- Algorithmus arbeitet korrekt
  - Warum?
- Laufzeitverhalten:
  - Konstruktion von π ist linear in k
  - Test $w \in L(\pi)$ ist für jedes Wort w linear in k
  - Komplexität von M ist “kürzeste Wortlänge · Summe aller Wortlängen”

Zusammenfassung

- Patternsprachen sind aus Text lernbar, aber ineffizient.
- Beim Verzicht auf Konsistenz gewinnt man Polynomialzeitverhalten.
- Reguläre Patternsprachen sind effizient lernbar.

Changelog

- V1.1:
  - Folie 11: Algorithmus korrigiert