Ambiguität in formalen Sprachen II: Kontextfreie Sprachen

Christian Wurm

Düsseldorf, 9.5.2023

Formale Sprachen sind nicht per se ambig: jede Turingmaschine kann determinisiert werden.

Formale Sprachen sind ambig

1. Im Hinblick auf eine bestimmte Grammatik/Automat
2. Im Hinblick auf alle Grammatiken/Automaten einer bestimmten Klasse.

Wir werden beide Fälle betrachten:

1. Ambiguität von endlichen Automaten, also für reguläre Sprachen
2. Ambiguität von Sprachen im Hinblick auf alle Kontext-freien Grammatiken

Diesmal: Kontextfreie Sprachen! Wer hats gefunden? Chomsky '57, vgl. Chomsky Hierarchie, Typ 2.

Kontextfreie Sprachen

Zur Erinnerung: kontextfreie Sprachen (CFL) sind charakterisiert durch

1. Kontextfreie Grammatiken (CFG)
2. Keller Automaten (alias *pushdown* oder PDA)
3. Lineare Gleichungssysteme (etwas esoterisch)
4. Durch Homomorphismus, inversen Homomorphismus und Schnitt mit regulären Sprachen aus *einer einzigen* Sprache (gegeben ein Alphabet)
5. Kategorialgrammatiken (aber CCG ist ein sehr problematischer Spezialfall)

Normalerweise beschränkt man sich auf CFG und PDA, alles andere ist nicht so wichtig.

Kontextfreie Sprachen

Kontextfreie Grammatiken sind in der Computerlinguistik absolut zentral:

- Lange Zeit war man der Meinung dass CFG ausreichen, um alle natürlichen Sprachen zu beschreiben (bis in die 90er, siehe Shieber/Schweizerdeutsch)
- Aber auch praktisch alle Formalismen, die jenseits von CFL liegen, haben ein kontextfreies "Rückgrat":

Kontextfreie Sprachen

Aber auch praktisch alle Formalismen, die jenseits von CFL liegen, haben ein kontextfreies "Rückgrat":

- CCG, wie gesagt, und erweiterte Kategorialgrammatiken (mit Modalität)
- Minimalist Grammars bzw. Transformationsgrammatiken in der generativen Tradition Chomskies.
- Auch TAG kann man auffassen als Erweiterung von CFG: CFG ist Stufe 0 von Weirs Kontrollsprachen-Hierarchie, Tag ist Stufe 1.
- Ebenso MCFG bzw. LCFRS bilden eine Hierarchie, in der CFG auf Stufe 0 liegt.

Kontextfreie Sprachen

Kontextfreie Grammatiken sind ebenfalls in der Informatik absolut zentral:

- Die Syntax der meisten Programmiersprachen ist kontextfrei (Klammer auf – Klammer zu; die Dyk-Sprache)
- Die Semantik allerdings nicht unbedingt, insbesondere die Belegung von Variablen ist problematisch unter "lokaler" Kontextfreier Interpretation
- Viele Sprachen der Mathematik sind kontextfrei:
  - Arithetische Terme und Gleichungen
  - Formale Logik (Aussagen- und Prädikatenlogik etc.)

Abschlusseigenschaften

Kontextfreie Grammatiken sind wichtig wegen ihrer Abschlusseigenschaften.

1. Homomorphismus: $L \in CFL \Rightarrow h[L] \in CFL$
2. Inverser Homomorphismus: $L \in CFL \Rightarrow h^{-1}[L] \in CFL$
3. Vereinigung: $L, L' \in CFL \Rightarrow L \cup L' \in CFL$
4. Schnitt mit regulären Sprachen: $L \in Reg,\ L' \in CFL \Rightarrow L \cap L' \in CFL$
5. Konkatenation: $L, L' \in CFL \Rightarrow L \cdot L' \in CFL$
6. Kleene Stern: $L \in CFL \Rightarrow L^* \in CFL$
7. Quotient: $L, L' \in CFL \Rightarrow L/L', L\backslash L' \in CFL$
8. Komplement: $L \in CFL \Rightarrow \Sigma^* - L \in CFL$ NEIN!
9. Schnitt: $L, L' \in CFL \Rightarrow L \cap L' \in CFL$ NEIN!

Abschlusseigenschaften

Kontextfreie Grammatiken sind wichtig wegen ihrer Abschlusseigenschaften.

8. Komplement: $L \in CFL \Rightarrow \Sigma^* - L \in CFL$ NEIN!
9. Schnitt: $L, L' \in CFL \Rightarrow L \cap L' \in CFL$ NEIN!

Das bedeutet: CFL sind keine **Boolesche Algebra**, und zu Zwecken der Verifikation kann man sie nicht einsetzen!

Entscheidungsprobleme – Korrespondenzen

1. Leerheit entspricht Konsistenz einer Spezifikation
2. Universalität entspricht Tautologie, sprich eine Spezifikation ist uninformativ
3. Teilmenge entspricht logischer Implikation einer Spezifikation
4. Äquivalenz (mengentheoretisch) entspricht Äquivalenz (einer Spezifikation)

Entscheidungsprobleme sind also oftmals verknüpft mit reellen Problemen!

Entscheidungsprobleme

Hier eine Liste relevanter Entscheidungsprobleme.

1. Gegeben eine Charakterisierung c von $L \in CFL$, ist $L = \emptyset$?
2. Gegeben eine Charakterisierung c von $L \in CFL$, ist L universell ($= \Sigma^*$)? NEIN!
3. Gegeben c, c' für L, L', ist $L \subseteq L'$? NEIN!
4. Gegeben c, c' für L, L', ist $L = L'$? NEIN!

Die negativen Ergebnisse lassen sich leicht herleiten aus folgendem Ergebnis.

Die Mutter der Unentscheidbarkeit

Theorem (Halteproblem, Turing)

Gegeben eine Turingmaschine M, ein Wort w, ist es i.A. unentscheidbar ob M das Wort w akzeptiert bzw. nicht akzeptiert.

Theorem (CFL und RA)

Gegeben ein Turingmaschine M gibt es zwei CFL L_1, L_2 und einen Homomorphismus h so dass $L(M) = h[L_1 \cap L_2]$.

Anders gesagt: jede rekursiv aufzählbare Sprache hat die Form $h[L_1 \cap L_2]$, $L_1, L_2 \in CFL$.

Unentscheidbarkeit von CFL-Problemen

Aus diesem fundamentalen Ergebnis lassen sich eine ganze Reihe von Unentscheidbarkeits-Ergebnissen für CFL ableiten, neben den obigen:

Unentscheidbarkeit

Gegeben eine Charakterisierung c von $L \in CFL$, ist es unentscheidbar ob $L \in Reg$.

Aus diesem Grund ist es sehr interessant, **Subklassen von CFL zu finden**, die bessere Entscheidungsprobleme haben.

Intermediäre Klassen

Da für Reg alle relevanten Probleme entscheidbar sind, für CFL fast alle relevanten Probleme unentscheidbar, interessieren uns **intermediäre Klassen** (das ist ein Begriff den ich nur für diese Sitzung erfunden habe; es gibt sonst glaube ich keinen Sammelbegriff für was wir hier machen).

Intermediäre Klassen

Eine Klasse von Sprachen C ist **intermediär**, falls $Reg \subsetneq C \subsetneq CFL$.
Eine intermediäre Klasse C ist **intermediär entscheidbar**, falls ein Entscheidungsproblem, was für CFL unentscheidbar ist, für C entscheidbar ist.

CFL und Verfikation

Daraus folgt:

- Wenn ein Problem jenseits der regulären Sprachen liegt, dann wird eine effektive Verifikation unmöglich!
- Es wäre aber schön, auch mit kontext-freien Problemen effektiv zu operieren!

Intermediär entscheidbare Klassen

Intermediär entscheidbare Klassen: Überblick

Wie wir sehen werden haben Ambiguität und Determinismus eine entscheidende Rolle für intermediär entscheidbare Klassen.

Wir werden hier betrachten:

1. Transparente kontextfreie Sprachen.
2. NTS-Sprachen
3. Deterministische kontextfreie Sprachen.
4. Unambige kontextfreie Sprache.

In dieser Reihe sind die Klassen nach Inklusion geordnet.

Intermediär entscheidbare Klassen: Überblick

Wie wir sehen werden haben Ambiguität und Determinismus eine entscheidende Rolle für intermediär entscheidbare Klassen.

Wir werden hier betrachten:

1. Transparente kontextfreie Sprachen.
2. NTS-Sprachen
3. Deterministische kontextfreie Sprachen.
4. Unambige kontextfreie Sprache.

In dieser Reihe sind die Klassen nach Inklusion geordnet.

Lokale und globale Ambiguität

Wir erinnern uns an die Entscheidung zwischen lokaler und globaler Ambiguität.
In der formalen Sprachtheorie gibt es forlgende Korrespondenzen:

- Satz in natürlicher Sprache $\cong$ Wort in formaler Sprache
- Interpretation in natürlicher Sprache $\cong$ Ableitung in formaler Sprache

Lokale Ambiguität

Ambiguität ist lokal, falls ein Teil eines Satzes/Wortes mehrere Ableitungen hat, der Satz selber aber nur eine einzige.

Globale Ambiguität

Ambiguität ist global, falls ein Satz/Wort mehrere Ableitungen hat.

Intermediär entscheidbare Klassen: Überblick

Transparente Sprachen

Eine Sprache ist transparent, falls es keine lokale Ambiguität gibt. Sprich: jeder Teil jedes Wortes/Satzes kann für sich nur auf eine Art interpretiert werden.

NTS-Sprachen

Eine Sprache ist NTS, falls wir für *jede lokale* Interpretation eines Teilwortes/satzes *mindestens eine* damit konsistente *globale* Interpretation des Gesamtsatzes finden.

Die Inklusion ist damit trivial.

Intermediär entscheidbare Klassen: Überblick

NTS-Sprachen

Eine Sprache ist NTS, falls wir für jede lokale Interpretation eines Teilwortes/satzes eine damit konsistente globale Interpretation des Gesamtsatzes finden.

Deterministische Sprachen

Eine Sprache ist deterministisch, falls es, wenn wir das Wort/Satz von links nach rechts lesen, an jedem Punkt höchstens eine mögliche Interpretation gibt.

Diese Inklusion ist alles andere als trivial!

Intermediär entscheidbare Klassen: Überblick

Deterministische Sprachen

Eine Sprache ist deterministisch, falls es, wenn wir das Wort/Satz von links nach rechts lesen, an jedem Punkt höchstens eine mögliche Interpretation gibt.

Unambige Sprachen

Eine Sprache ist unambig, falls es, wenn es für jedes Wort/Satz höchstens eine mögliche Interpretation gibt.

Diese Inklusion ist wiederum geradeaus!

Intermediär entscheidbare Klassen: Überblick

Also:

$$\begin{aligned} & \text{Transparente kontextfreie Sprachen} \\ \subseteq\ & \text{NTS-Sprachen} \\ \subseteq\ & \text{Deterministische kontextfreie Sprachen.} \\ \subseteq\ & \text{Unambige kontextfreie Sprache.} \end{aligned}$$

Diese Inklusion betrifft aber nur die Sprachen, nicht die Grammatiken:

- eine Grammatik mit NTS-Eigenschaft muss keinesfalls unambig sein!
- Es gibt aber eine äquivalente unambige Grammatik!

Transparente Sprachen

Transparente Sprachen: Definition

Sei G eine CFG. Wir sagen w ist ein G-Konstituent der Kategorie N, falls $N \vdash_G^* w$.

Wir definieren den Begriff der **Transparenz** leicht informell mittels Ableitungsbäumen.

Akzidentelle Konstituenten

Sei B ein Ableitungsbaum in G für xwy, wobei w eine G-Konstituent der Kategorie N ist. Wir sagen das w **akzidentiell** ist in B, falls w *nicht* dominiert wird von einem Knoten der Kategorie N.

Akzidentielle Konstituenten: Ein Beispiel

Nimm die Sprache $L_m = \{a^n b^m : n < m\}$. Hier gibt es (in praktisch jeder Grammatik) im Wort

$$a^4 b^6$$

eine akzidentielle Konstituente

$$a^3 b^6$$

Denn dieses Wort ist in L_m, also eine Konstituente der Kategorie S!

Transparente Sprachen: Definition

Akzidentelle Konstituenten

Sei B ein Ableitungsbaum in G für xwy, wobei w eine G-Konstituent der Kategorie N ist. Wir sagen das w **akzidentiell** ist in B, falls w *nicht* dominiert wird von einem Knoten der Kategorie N.

Akzidentelle Konstituenten haben also die Eigenschaft,

- dass sie für sich betrachtet von einer Kategorie sein könnten,
- es aber im vorliegenden Baum nicht sind!

Transparente Sprachen: Definition

Transparente Grammatiken

Eine Grammatik ist **transparent**, falls sie keine **akzidentiellen Konstituenten** generiert.

Das bedeutet: wir können

- in jedem Fall und immer
- unabhängig vom Kontext

die Kategorie eines Teilwortes eindeutig bestimmen.

Transparente Sprachen: Definition

Transparente Sprachen

Eine Sprache L ist **transparent**, falls es eine transparente Grammatik G gibt so dass $L(G) = L$.

Hier sehen wir wieder den Unterschied von Sprachen und Grammatiken:

Entscheidbarkeit

1. Es ist entscheidbar, ob eine Grammatik G transparent ist.
2. Es ist allerdings nach meiner Kenntnis nicht entscheidbar, ob eine Sprache transparent ist!

Transparente CFL: Beispiele

Beispiel 1

Die Sprache $L_1 = \{w\#w^T : w \in \Sigma^*, \# \notin \Sigma\}$ ist transparent.

Beispiel 2

Die Sprache $L_2 = \{a^n b^n c^m : m, n \in \mathbb{N}\}$ ist transparent.

Beispiel 3

Die Sprache $L_3 = \{a^n b^m c^n : m, n \in \mathbb{N}\}$ ist transparent.

Beispiel 4

Die Sprache $L_4 = \{ww^T : w \in \Sigma^*\}$, ist **nicht** transparent.

Beispiel 5

Die Sprache $L_2 \cup L_3$, ist **nicht** transparent.

Transparente CFL: Beispiele

Beispiel 6

a^+ ist **nicht** transparent; also sind die transparenten Sprachen nicht intermediär!

Beweis ist nicht ganz einfach. Idee ist: jedes Teilwort von a^n ist eine Konstituente von a^n – aber einige sind zwangsläufig immer akzidentiell!

Frage

Warum ist a^+ nicht transparent, aber L_2 (was das in gewissem Sinne subsumiert) schon?

Antwort

Wir müssen verhindern, dass irgendein c^n überhaupt eine Konstituente sein kann (passende Grammatik schreiben)! Hierfür ist entscheidend dass $n > 0$!

Transparente CFL: Beispiele

Beispiel 7

Die Dyck-Sprache (über beliebiges Alphabet) ist transparent.

Beispiel 8

Praktisch alle logischen Sprachen, Termsprachen, also formalen Sprachen der Mathematik, sind transparent.

Transparente Sprachen sind also durchaus wichtig und interessant, aber nicht wirklich das was wir hier suchen.

NTS-Sprachen

NTS-Sprachen: Definition

Die NTS-Eigenschaft ist etwas lockerer als Transparenz:

Eine Grammatik G hat die NTS-Eigenschaft, falls folgendes gilt: falls

1. w eine G-Konstituente der Kategorie N ist, und
2. $S \vdash_G^* xwy$,

dann gilt auch $S \vdash_G^* xNy$.

Sprich: es ist möglich, Kategorien rein lokal ohne Kontext zuzuweisen, und wir bekommen dergestalt eine Ableitung, aber: wir bekommen nicht die einzig mögliche Ableitung!

NTS-Sprachen: Definition

Eine Grammatik G hat die NTS-Eigenschaft, falls folgendes gilt: falls

1. w eine G-Konstituente der Kategorie N ist, und
2. $S \vdash_G^* xwy$,

dann gilt auch $S \vdash_G^* xNy$.

Dann wie üblich:

NTS-Sprachen

Eine Sprache L ist NTS, falls es eine NTS-Grammatik G gibt so dass $L = L(G)$.

NTS-Sprachen: Inklusion

Das erste ist offensichtlich:

Lemma

Jede transparente Sprache ist NTS, aber nicht umgekehrt.

Eine NTS-Sprache die nicht transparent ist, ist folgende: a^+

Folgendes Ergebnis ist für uns interessanter (aber nicht einfach zu beweisen!):

Reguläre Sprachen

Jede reguläre Sprache ist NTS.

NTS-Sprachen: Inklusion

Auch folgendes Ergebnis ist alles andere als einfach zu beweisen (man generiert die Greibach Normal Form aus dem Kellerautomaten):

Inklusion

Jede NTS-Sprache ist in DCFL.

Aber:

Inklusion

Nicht jede DCFL ist NTS!

Ein Gegenbeispiel ist folgendes:

$L_7 = \{a^n b^m : n \geq m\}$

NTS-Sprachen: Inklusion

Inklusion

Nicht jede DCFL ist NTS!

Ein Gegenbeispiel ist folgendes:

$L_7 = \{a^n b^m : n \geq m\}$ ist nicht NTS –

- es finden sich hier zwangsläufig akzidentelle Konstituenten der Form Sb,
- aber $S \to Sb$ kann keine Regel sein!

Deterministische Sprachen

Kellerautomaten (Wiederholung)

Ein Kellerautomat hat die Form

$$(\Sigma, Q, q_0, F, \Gamma, \#, \delta)$$

wobei

- Σ das Eingabealphabet ist
- Q die endliche Zustandsmenge,
- $q_0 \in Q$ der Startzustand, $F \subseteq Q$ die akzeptierenden Zustände
- Γ ist das stack-Alphabet, die Symbole die auf den stack geschrieben werden
- $\# \notin \Gamma$ ist das Symbol für den Boden des stacks.
- δ ist die Übergangsrelation, wobei

$$\delta \subseteq \Gamma_{\#} \times Q \times (\Sigma \cup \epsilon) \times Q \times \Gamma^*$$

Kellerautomaten (Wiederholung)

Eine **Konfiguration** ist ein Tupel (S, q, w), wobei

- $S \in \#\Gamma^*$
- $q \in Q$
- $w \in \Sigma^*$

Ein PDA geht von Konfiguration in Konfiguration nach δ:

$$(S\gamma, q, aw) \vdash (Sw, q', w) \text{ gdw. } (\gamma, q, a, q', w)\delta$$

$\vdash^*$ ist die **reflexive transitive** Hülle von $\vdash$.
Ein PDA **akzeptiert** w, falls gilt:

$$(\#, q_0, w) \vdash^* (S, q_f, \epsilon), \text{ wobei } q_f \in F.$$

Kellerautomaten (Wiederholung)

Im allgemeinen lassen sich Kellerautomaten nicht determinisieren:

Lemma

Es gibt kontextfreie Sprachen L, für die gibt es keinen **deterministischen** Kellerautomaten K so dass $L = L(K)$.

Dieses Ergebnis liefert die Grundlage für folgende Definition:

Deterministische CFL

Eine Sprache L ist **deterministisch kontext-frei**, falls es einen deterministischen Kellerautomaten K gibt so dass $L = L(K)$. Wir nennen diese Klasse DCFL.

Deterministische CFL: Beispiele

Beispiel 1

Die Sprache $L_1 = \{w\#w^T : w \in \Sigma^*, \# \notin \Sigma\}$, ist deterministisch.

Beispiel 2

Die Sprache $L_2 = \{a^n b^n c^m : m, n \in \mathbb{N}\}$, ist deterministisch.

Beispiel 3

Die Sprache $L_3 = \{a^n b^m c^n : m, n \in \mathbb{N}\}$, ist deterministisch.

Beispiel 4

Die Sprache $L_4 = \{ww^T : w \in \Sigma^*\}$, ist **nicht** deterministisch.

Deterministische CFL: Beispiele

Beispiel 5

Die Sprache $L_2 \cup L_3$, ist **nicht** deterministisch.

Beispiel 6

Jede reguläre Sprache ist deterministisch, also $Reg \subsetneq DCFL$.

Beispiel 7

Die Dyck-Sprache (über beliebiges Alphabet) ist deterministisch.

Deterministische CFL: Ergebnisse

Für DCFL gibt es folgende wichtige Ergebnisse:

Wortproblem (besser bekannt als Parsing)

Gegeben w und $L \in DCFL$, ist es in linearer Zeit entscheidbar ob $w \in L$.

Wir erinnern uns dass für CFL i.A. die Zeitkomplexität bei $\approx O(|w|^{2.7})$ liegt, $O(|w|)$ ist also deutlich besser!

Dieses Ergebnis ist natürlich nur dann gültig, wenn unsere "Präsentation" der Sprache L deterministisch ist. Es gibt keinen Determinisierungs-Algorithmus für Kellerautomaten!

Deterministische CFL: Ergebnisse

Zentrales Theorem: Abschluss unter Schnitt

Falls $L, L' \in DCFL$ dann ist auch $L \cap L' \in DCFL$.

Das ist ein Hammer! Daraus folgt nun folgendes:

Korollar

Für $L, L' \in DCFL$ können wir entscheiden, ob $L \cap L' = \emptyset$.

Deterministische CFL: Ergebnisse

Vereinigung, Komplement

DCFL ist **nicht** abgeschlossen unter Vereinigung (Bsp. 5), also auch nicht unter Komplement (Kontraposition).

Das bedeutet insbesondere: die Probleme der

- Universalität
- Äquivalenz
- Inklusion

zweier Sprachen sind nicht unbedingt entscheidbar. Meiner Kenntnis nach sind das noch relativ prominente offene Probleme der formalen Sprachtheorie.

Unambige Sprachen

Unambige Sprachen

Determinismus ist eine lokale Eigenschaft von Automaten, Ambiguität ist eine globale Eigenschaft:

Definition

Eine CFG G ist unambig, falls es für jedes Wort $w \in L(G)$ nur eine Ableitung gibt. Eine Sprache L ist unambig, falls es eine unambige Grammatik G gibt so dass $L(G) = L$.

Unambige Sprachen: Beispiele

Beispiel 1

Die Sprache $L_1 = \{w\#w^T : w \in \Sigma^*, \# \notin \Sigma\}$ ist unambig.

Beispiel 2

Die Sprache $L_2 = \{a^n b^n c^m : m, n \in \mathbb{N}\}$ ist unambig.

Beispiel 3

Die Sprache $L_3 = \{a^n b^m c^n : m, n \in \mathbb{N}\}$ ist unambig.

Beispiel 4

Die Sprache $L_4 = \{ww^T : w \in \Sigma^*\}$, ist unambig.

Unambige Sprachen: Beispiele

Beispiel 5

Die Sprache $L_2 \cup L_3$, ist **nicht** unambig.

Beispiel 7

Die Dyck-Sprache (über beliebiges Alphabet) ist unambig.

Unambige Sprachen: Caucal

Caucal ist Graphentheoretiker; er behandelt Sprachen mit Graphen.

- Einen Graphen kann man sich vorstellen als einen unendlichen Automaten.
- Caucals Grammatiken generieren die unendlichen Zustandsgraphen von PDA!

Dementsprechend:

- Eine Sprache ist **deterministisch**, gdw. sie von einem deterministischen gelabelten Graphen erkannt wird.
- Eine Sprache ist **unambig**, gdw. es einen Graphen gibt der sie erkennt, der für jedes Wort nur einen akzeptierenden Pfad hat.

Diese Äquivalenzen sind aber nicht trivial!

Ambiguität: Inklusion

Daraus folgt natürlich folgendes:

Theorem

Jede deterministische CFL ist eine unambige CFL.

Aber nicht umgekehrt:

$L_8 = \{a^n b^{2n} c : n \in \mathbb{N}\} \cup \{a^{2n} b^n d : n \in \mathbb{N}\}$ ist unambig, aber nicht deterministisch.

- Der PDA weiss beim Lesen anfangs nicht, ob er für jedes a 2 Symbole in den Keller tun soll, oder "ein halbes" – woher auch?
- In der "Draufsicht" ist die Sache aber klar!

Ambiguität: Inklusion

Symmetrie

Wenn eine Sprache unambig ist, dann ist es offensichtlich auch ihr Spiegelbild.

- Für deterministische Sprachen gilt das **nicht**, da Determinismus entscheidend von der Richtung abhängt.
- (Es wäre also auch sinnvoll von **Co-Deterministischen** Sprachen zu sprechen, die von hinten deterministisch sind!)

Ambiguität; automatentheorisch

Lemma

Eine kontextfreie Sprache $L \subseteq \Sigma^*$ ist unambig, genau dann wenn es einen PDA gibt, der für jedes Wort $w \in \Sigma^*$ höchstens einen akzeptierenden Lauf hat

Das ergibt sich aus der engen Korrelation

$$\text{PDA} - \text{CFG}$$

Synchronisierung

Eine Graph-Grammatik weist jedem Knoten des generierten Graphen ein **level** zu. Das ist der Schritt, in dem der Knoten generiert wurde.

Synchronisierung

Eine Graph-Grammatik G wird synchronisiert von S, geschrieben $G \lhd S$, wenn es für jeden akzeptierenden Pfad p_g in G einen gleichnamigen akzeptierenden Pfad p_s in S gibt, so dass jeder Knoten an n-ter Stelle das gleiche level hat.

Synchronisierung – Automatentheoretisch

Ein Automat weist jeder Konfiguration ein *level* zu – die minimale Anzahl der Berechnungsschritte, die er braucht, um in diese Konfiguration zu kommen.

Synchronisierung

Ein (unambiger) PDA $\mathfrak{A}$ wird synchronisiert von $\mathfrak{B}$, geschrieben $\mathfrak{A} \lhd \mathfrak{B}$, wenn es für jeden akzeptierenden Lauf l_a in $\mathfrak{A}$ einen gleichnamigen akzeptierenden Lauf l_b in $\mathfrak{B}$ gibt, so dass für alle n gilt:

Die n-te Konfiguration in $\mathfrak{A}$ und die n-te Konfiguration in $\mathfrak{B}$ haben dasselbel level.

Synchronisierung – Automatentheoretisch

Bsp.:

- $\mathfrak{A}$ für $a^{2n}b^{2n}$
- $\mathfrak{B}$ für $a^n b^n$

$\mathfrak{B}$ kann $\mathfrak{A}$ synchronisieren, d.h. sein Verhalten *simulieren*.

Synchronisierung

$\mathfrak{A} \lhd \mathfrak{B}$ impliziert natürlich: $L(\mathfrak{A}) \subseteq L(\mathfrak{B})$.

Aber es bedeutet mehr: die Struktur wird simuliert. Das ist entscheidend für das folgende Ergebnis, das zentral ist für intermediäre Sprachen:

Definition

$sync(\mathfrak{B}) = \{L : L = L(\mathfrak{A})$, wobei 1. $\mathfrak{A} \lhd \mathfrak{B}$ und 2.$\mathfrak{A}$ unambig$\}$

$sync(\mathfrak{B})$ enthält also die Menge aller Sprachen aller Automaten, die $\mathfrak{B}$ simulieren kann.

Caucals Theorem

Deswegen sind die folgenden Ergebnisse so interessant:

Theorem (Caucal)

Für jeden unambigen Graphen R ist $sync(R)$ eine **Boolesche Algebra**, mit maximalem Element $L(R)$ und relativem Komplement dazu.

Man beachte, dass das konform ist mit der Tatsache, dass unambige Sprachen nicht geschlossen sind unter Vereinigung:

- Wenn die beiden Sprachen aus verschiedenen *sync*-Algebren kommen, ist ihre Vereinigung nicht unbedingt unambig!

Caucals Theorem – automatentheoretisch

Theorem (Caucal)

Für jeden unambigen PDA $\mathfrak{B}$ ist $sync(\mathfrak{B})$ eine **Boolesche Algebra**, mit maximalem Element $L(\mathfrak{B})$ und relativem Komplement dazu.

Man beachte, dass das konform ist mit der Tatsache, dass unambige Sprachen nicht geschlossen sind unter Vereinigung:

- Wenn die beiden Sprachen aus verschiedenen *sync*-Algebren kommen, ist ihre Vereinigung nicht unbedingt unambig!

Konsequenz

Korollar

Nimm an L, L' sind unambige Sprachen, und es gibt einen unambigen Graphen/PDA $\mathfrak{A}$ so dass $L, L' \in sync(\mathfrak{A})$. Dann können wir entscheiden ob $L \subseteq L'$.

Folgt aus der üblichen Methode: Schnitt und Komplement, dann Leerheit prüfen!

Literatur

- Kracht, M.:Mathematics of Language, Kapitel 2.3, 2.4
- Caucal, D.: Boolean Algebras of Unambiguous Context-Free Languages