L–Systeme

Vorlesung

Sommersemester 2001

Johannes Waldmann, Institut für Informatik
Universität Leipzig, joe@informatik.uni-leipzig.de

Kapitel 1

Überblick

In der Vorlesung betrachten wir Motivation, Grundlagen, Resultate und Anwendungen von Lindenmayer–Systemen (L–Systemen).

Wir lernen damit einen wesentlichen Zweig der Theorie der Formalen Sprachen innerhalb der Theoretischen Informatik kennen.

Ich bemühe mich, die gezeigten Definitionen und Resultate einerseits praktisch (durch Software) zu verdeutlichen, andererseits ihre Entstehungsgeschichte zu verfolgen, und schließlich auf offene Probleme hinzuweisen.

Hier folgt eine kurze Inhaltsangabe der Kapitel. Die Reihenfolge ändert sich eventuell noch.

Probieren Sie bereits jetzt, die Beispiele und Aufgaben zu bearbeiten. Darunter befinden sich einige leichte, einige schwere, einige unlösbare, und einige, deren Status bis heute offen ist.

Grundlagen

- Lindenmayers Definitionen von 2L–, 1L– und 0L–Systemen [Lin68, Lin71]
- einfache geometrische Anwendungen [PL90]
- erzeugte Sprachen, Vergleich mit bekannten Sprachfamilien
- Wachstum [KRS97]

typische Beispiele/Fragen:

- Der Morphismus $h : a \mapsto abc^2, b \mapsto bc^2, c \mapsto c$ bestimmt die Folge der Wörter

  $$w_0 = a, w_1 = h(a) = abc^2, w_2 = h^2(a) = abc^2bc^4, \dots$$

  Berechne die Folge der Wortlängen $|h^0(a)|, |h^1(a)|, |h^2(a)|, \dots$
- Welche Funktionen können so realisiert werden?

Unendliche Wörter und vermeidbare Muster

- Fixpunkte von Morphismen
- vermeidbare Muster: klassische Resultate (Thue, Morse) [Thu12, Thu06] [MH44, MH38]
- moderne Resultate (Cassaigne) [Cas94], offene Fragen

typische Beispiele/Fragen:

- Das von $\tau : a \mapsto ab, b \mapsto ba$ erzeugte Wort $abbabaabbaababba\ldots$ enthält kein Teilwort der Form xxx.
- Finde ein Verfahren, das entscheidet, ob ein Muster m über einem Alphabet Σ durch ein HD0L–Wort vermieden wird.
- Beweise: k–vermeidbar $\iff$ k–HD0L–vermeidbar.

Test- und Gleichheitsmengen

- lösbare Fälle des Postschen Korrespondenzproblems [HK97]
- Ehrenfeuchts Vermutung
- Wortgleichungen, Makanins Algorithmus [Die97]

typische Beispiele/Fragen:

- [Sal81] Die Gleicheitsmenge $E(g,h)$ zweier Morphismen g,h ist $\{w : g(w) = h(w)\}$. Bestimme $E(g,h)$ für

	a	b	c	d	e	f
g	1234	2323	4	24	32	5
h	1	23	4	42	3232	4325

- [Sal81] Für gegebenes $m,n \geq 1$, konstruiere g,h mit $E(g,h) = (a^m b^n)^*$.
- T ist eine Testmenge für L, wenn für alle Morphismen g,h gilt

$$L \subseteq E(g,h) \iff T \subseteq E(g,h)$$

  Beispiel: $\{a,b\}$ ist *keine* Testmenge für $L = (ab)^*$. (Betrachte $g : a \mapsto 12, b \mapsto 3$ und $h : a \mapsto 1, b \mapsto 23$.) Finde eine kleine Testmenge für L.
- Beweise: jede Sprache besitzt eine endliche Testmenge.

Äquivalenzen

- Sprachen– und Folgen–Äquivalenz für D0L, HD0L
- Omega–Äquivalenz

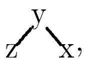
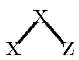
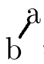
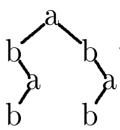
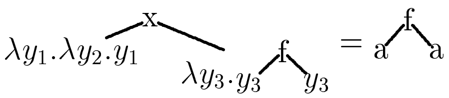
typische Beispiele/Fragen:

- [KRS97] Die D0L–Systeme
  $$S_1 = (\{a, b\}, \{a \mapsto b^2, b \mapsto a\}, b), \qquad S_2 = (\{a, b\}, \{a \mapsto b, b \mapsto a^2\}, a)$$
  erzeugen die gleiche Sprache, aber nicht die gleiche Folge.
- für die Morphismen
  $\tau : a \mapsto ab, b \mapsto ba, \quad \mu : 0 \mapsto 012, 1 \mapsto 02, 2 \mapsto 1, \quad h : 0 \mapsto abb, 1 \mapsto ab, 2 \mapsto a$
  gilt $h(\mu^\omega(0)) = \tau^\omega(a)$.
- Finde ein Entscheidungsverfahren für solche Gleichungen.

Bäume

- Baumgleichungen
- Unifizieren erster und zweiter Ordnung
- Matching höherer Ordnung

typische Beispiele/Fragen:

- Für $t_1 =$ , $t_2 =$  und die Substitution $\phi : x \mapsto$ , $y \mapsto$ , $z \mapsto$ 
  gilt $\phi(t_1) = \phi(t_2) =$  . [MR98]
- [CDG+97] Die Gleichung
  
  wird durch $x = \lambda x_1.\lambda x_2.x_1(x_2(x_1(x_1(a, t_1), t_2)), t_3)$ erfüllt.

Nicht–Uniforme Modelle

- das L–Alphabet (A,C,D,E,F,H,I,J,L,N,O,P,T,U,W)
- Anwendungen beim Modellieren komplizierterer Wachstumsprozesse [PL90, Kur99]
- P–Systeme zur Beschreibung von (bio–)molekularen Rechnern [Pau98]

Inhaltsverzeichnis

Literaturverzeichnis

[Cas94] Julien Cassaigne. *Motifs évitables et régularitiés dans le mots.* PhD thesis, Université Paris 6, 1994.

[CDG+97] Hubert Comon, Max Dauchet, Rémi Gilleron, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications.* http://www.grappa.univ-lille3.fr/tata/, 1997.

[CG95] John H. Conway and Richard K. Guy. *The Book of Numbers.* Copernicus/Springer, 1995.

[CIH84] Karel Culik II and Tero Harju. The ω sequence equivalence problem for d0l systems is decidable. *Journal ACM*, 31(2):282–298, April 1984.

[CK97] Christian Choffrut and Juhani Karhumäki. *Combinatorics of Words*, pages 329–428. Volume 1 of Rozenberg and Salomaa [RS97], 1997.

[Die97] Volker Diekert. *Makanin's Algorithm*, pages 344–391. In [Lot97a], 1997.

[HH01] Vesa Havala and Tero Harju. Some new results on post correspondence problem and its modifications. TUCS Technical Report 338, Turku Centre for Computer Science, January 2001. http://www.tucs.fi/publications/techreports/TR338.html.

[HK97] Tero Harju and Juhani Karhumäki. *Morphisms*, pages 439–510. Volume 1 of Rozenberg and Salomaa [RS97], 1997.

[KRS97] Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. *L Systems*, pages 253–328. Volume 1 of Rozenberg and Salomaa [RS97], 1997.

[Kur99] Winfriend Kurth. *Die Simulation der Baumarchitektur mit Wachstumsgrammatiken.* Wissenschaftlicher Verlag Berlin, 1999. http://www.uni-forst.gwdg.de/~wkurth/public.html.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interaction in development. *J. Theoret. Biology*, 18:280–315, 1968.

[Lin71] Aristid Lindenmayer. Developmental systems without cellular interactions, their languages and grammars. *J. Theoret. Biology*, 30:455–484, 1971.

[Lot97a] M. Lothaire. *Algebraic Combinatorics on Words.* http://www-igm.univ-mlv.fr/~berstel/Lothaire/, 1997.

[Lot97b] M. Lothaire. *Combinatorics on Words*. Cambridge Mathematical Library. Cambridge University Press, 1997.

[MH38] Marston Morse and Gustav Hedlund. Symbolic dynamics. *Amer. J. Mathematics*, LX:815–866, 1938.

[MH44] Marston Morse and Gustav Hedlund. Unending chess, symbolic dynamics and a problem in semigroups. *Duke Mathem. J.*, 11, 1944.

[MR98] Sabrina Mantaci and Antonio Restivo. Codes and equations for trees. Technical Report 194, Turku Centre for Computer Science, August 1998. http://altair.math.unipa.it/CSG/sabrina/publications.html.

[OR00] John J. O'Connor and Edmund F. Robertson. The MacTutor History of Mathematics Archive. http://www-groups.dcs.st-and.ac.uk:80/~history/, 2000.

[Pau98] Gheorghe Paun. Computing with membranes. Technical Report 208, Turku Centre for Computer Science, November 1998. http://www.tucs.fi/publications/techreports/TR208.html.

[PL90] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, Berlin, 1990.

[PWZ96] Marko Petkovsek, Herbert Wilf, and Doron Zeilberger. $A = B$. A. K. Peters, 1996. http://www.akpeters.com/book.asp?bID=100.

[RS80] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.

[RS85] G. Rozenberg and A. Salomaa, editors. *The Book of L*. Springer, Berlin, 1985.

[RS97] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*. Springer, 1997.

[Sal81] Arto Salomaa. *Jewels of Formal Language Theory*. Computer Science Press, Rockville, 1981.

[Sch95] Uwe Schöning. *Perlen der Theoretischen Informatik*. B I Wissenschaftsverlag; Mannheim, 1995.

[Thu06] Axel Thue. Über unendliche zeichenreihen. *Kra. Vidensk. Selsk. Skrifter. I. Mat.-Nat. Kl. Christiana*, (7), 1906.

[Thu12] Axel Thue. Über die gegenseitige lage gleicher teile gewisser zeichenreihen. *Kra. Vidensk. Selsk. Skrifter. I. Mat.-Nat. Kl. Christiana*, (10), 1912.

Kapitel 2

Grundlagen

2.1 Wörter

Ein *Alphabet* ist eine endliche Menge von Zeichen.

Ein *Wort* w ist eine Folge von Zeichen $(w_1, w_2, \ldots, w_n)$. Die Klammern und Kommas lassen wir weg.

Die Länge $|w|$ des Wortes $w = w_1 w_2 \ldots w_n$ ist n.

Das leere Wort heißt ϵ. (In manchen Büchern: λ, und manchmal auch: 1.) Es gilt $|\epsilon| = 0$.

Eine *Sprache* ist eine Menge von Wörtern.

Die Sprache *aller* Wörter über Σ heißt Σ^*, und $\Sigma^* \setminus \{\epsilon\}$ heißt Σ^+.

Wir lassen die Mengenklammern um Einermengen gern weg und schreiben $\Sigma^* \setminus \{\epsilon\}$.

Die Operation $\cdot$ ist das Hintereinanderschreiben zweier Wörter. Sie ist assoziativ, aber nicht invertierbar, und (für Alphabete mit mehr als einem Buchstaben) nicht kommutativ.

Deswegen ist $(\Sigma^+, \cdot)$ eine Halbgruppe, und $(\Sigma^*, \cdot, \epsilon)$ ein Monoid.

Wir übertragen die Operation $\cdot$ von Wörtern auf Sprachen durch $L_1 \cdot L_2 = \{w_1 \cdot w_2 : w_i \in L_i\}$.

2.2 Substitutionen, Morphismen

Im folgenden sind Σ, Δ Alphabete.

Eine *Substitution* von Σ nach Δ ist eine Abbildung $s : \Sigma \to 2^{\Delta^*}$. (Jedem $x \in \Sigma$ eine Wortmenge $\subseteq \Delta^*$ zuordnet.)

Die Substitution s heißt *endlich*, wenn alle Mengen $s(x)$ endlich sind.

Die Substitution s heißt *eindeutig*, wenn alle Mengen $s(x)$ genau ein Element enthalten.

Die Substitution s heißt *löschend*, wenn es ein $x \in \Sigma$ gibt mit $\epsilon \in s(x)$. Andernfalls heißt s *nicht löschend*.

Das *Bild* $s(w)$ eines Wortes $w = w_1 \dots w_n \in \Sigma^*$ unter der Substitution s ist die Sprache $s(w_1) \cdot \ldots \cdot s(w_n)$.

Das Bild $s(L)$ einer Sprache $L \subseteq \Sigma^*$ ist die Sprache $\{v : w \in L, v \in s(w)\}$.

Falls s eindeutig ist, dann ist $s(w)$ eine Einermenge. Wir identifizieren diese mit ihrem Element. Es gilt dann $s(\epsilon) = \epsilon$ und $s(v \cdot w) = s(v) \cdot s(w)$, also ist s tatsächlich ein Morphismus (vom Monoid Σ^* in das Monoid Δ^*).

2.3 0L und D0L

Definition 1 *Ein 0L–System ist ein Tripel $G = (\Sigma, s, w_0)$, wobei Σ ein endliches Alphabet ist, s eine endliche Substitution, und w_0 ein Wort $\in \Sigma^*$.*

Das Wort w_0 heißt *Axiom*.

Beispiel: $G_3 = (\{a\}, a \mapsto \{epsilon, a^2, a^5\}, a)$.

Definition 2 *Die vom 0L–System $G = (\Sigma, s, w_0)$ erzeugte Sprache $L(G)$ ist*

$$\bigcup_{i \geq 0} s^i(w_0)$$

Beispiel/Übung: Zeige $L(G_3) = a^* \setminus a^3$. Finde 0L–Systeme G_n mit $L(G_n) = a^* \setminus a^n$

Definition 3 *Eine Sprache ist eine 0L–Sprache, wenn ein 0L–System G existiert mit $L = L(G)$. Die Menge aller 0L–Sprachen heißt 0L.*

Definition 4 *Ein D0L–System ist ein 0L–System $G = (\Sigma, s, w_0)$, dessen Substitution s ein Morphismus ist.*

Übung: Betrachte die 0L–Systeme

$$G_1 = (\{a\}, a \mapsto \{a, a^2\}, a^2)$$
$$G_2 = (\{a, b\}, \{a, b\} \mapsto \{aa, ab, ba, bb\}, a)$$

Beweise: $L(G_1)$ und $L(G_2)$ sind keine D0L–Sprachen.

Die folgende Definition ist für D0L (aber nicht für OL) sinnvoll:

Definition 5 *Die vom D0L–System $G = (\Sigma, s, w_0)$ definierte* Folge *$S(G)$ ist die unendliche Liste von Wörtern*

$$(w_0, s(w_0), s^2(w_0), \ldots)$$

Beispiel/Übung: Welche Längen haben die Wörter in $S(G)$ für $G = (\{a, b, c, d\}, (a \mapsto abd^6, b \mapsto bcd^{11}, c \mapsto cd^6, d \mapsto d), a)$

2.4 Propagation

Definition 6 *Ein (D)0L–System* (Σ, s, w_0)*, heißt* propagierend*, wenn s nicht löschend ist.*

Die Menge der propagierenden 0L–Systeme heißt P0L, die der propagierenden D0L–Systeme heißt PD0L.

Diese (und folgende) Bezeichnugen für L–System–Klassen sind weitgehend modular.

Beispiel: Das D0L–System $D = (\{a, b\}, (a \mapsto ab^2a, b \mapsto \epsilon), ab^2a)$ ist kein PD0L–System.

Es gilt $L(D) = \{(ab^2a)^{2^i} : i \geq 0\}$.

Satz: Es gibt kein PD0L–System G mit $L(G) = L(D)$.

Beweis: Falls es doch ein solches $G = (\Sigma, s, w_0)$ gibt, dann muß w_0 das kürzeste Wort aus $L(D)$ sein (also ab^2a), denn s ist nicht löschend.

Weiter muß $s(w_0) = s(ab^2a) = ab^2a^2b^2a = w_1$ sein (das zweit–kürzestes Wort in $L(D)$). Dann steht $s(a)$ als Präfix und Suffix von w_1. Man probiert alle Möglichkeiten durch und sieht, daß keine funktioniert.

2.5 E0L

Die Grammatiken der Chomsky–Hierarchie erzeugen Wortmengen durch eine Kombination von zwei Mechanismen:

- *sequentielles, lokales* Anwenden von endlichen Substitutionen
- Schnitt mit einer regulären Sprache (der Terminalwörter)

Bei 0L–Systemen wenden wir die Substitutionen *parallel* (global) an. Weiterhin haben wir keine Möglichkeit, Hilfszeichen zu benutzen (in der erzeugten Sprache sind alle abgeleiteten Wörter).

Definition 7 *Ein E0L–System ist ein Tupel $G = (\Sigma, \Delta, s, w_0)$, wobei Σ und Δ disjunkte Alphabete sind, und $G' = (\Sigma \cup \Delta, s, w_0)$ ein 0L–System. Die von G erzeugte Sprache ist*

$$L(G) = L(G') \cap \Sigma^*$$

Die Σ sind Terminalzeichen, die Δ sind Nichtterminale.

Beachte, daß (im Gegensatz zu Grammatiken) durch s auch Terminale ersetzt werden.

Beispiel: Das E0L–System $S = (\{a, b, c\}, \{A, B, C, A', B', C', F\}, s, ABC)$ mit der Substition

$$s = \begin{array}{lllll} A \to AA' & A \to a & A' \to A' & A' \to a & a \to F \\ B \to BB' & B \to b & B' \to B' & B' \to b & b \to F \\ C \to CC' & C \to c & C' \to C' & C' \to c & c \to F \\ F \to F & & & & \end{array}$$

erzeugt $L(S) = \{a^n b^n c^n : n \geq 1\}$.

Wir erinnern uns: diese Sprache ist kontextsensitiv, aber nicht kontextfrei.

2.6 L und Chomsky

Zur Erinnerung: die Chomsky–Hierarchie ist

> (endliche $\subset$) reguläre $\subset$ kontextfreie $\subset$ kontextsensitive ($\subset$ entscheidbare) $\subset$ aufzählbare Sprachen

Die einfachsten Sprachen hier sind die endlichen, und schon diese sind nicht in 0L enthalten.

Beispiel/Übung: $\{a, a^3\} \notin 0L$.

Satz: kontextfreie $\subset$ E0L $\subset$ kontextsensitive Sprachen.

Beweis: aus eine kontextfreien Grammatik können wir ein E0L–System bauen, indem wir für jeden Buchstaben die Regel $x \to x$ hinzunehmen.

In der anderen Richtung müssen wir uns überlegen, wie wir E0L–Ableitungen durch eine kontextsensitive Grammatik (oder einen linear beschränkten Automaten) simulieren können.

Schwierig ist dabei nur der Fall von löschenden Morphismen. (Hausaufgabe :-)

Daß die angegebenen Inklusionen echt sind, sieht man einerseits an $CF \not\ni \{a^n b^n c^n : n \geq 1\} \in E0L$, andererseits $E0L \not\ni \{a^m b^n c^m : n \geq m \geq 1\} \in CS$.

Übung: Beweise, daß das tatsächlich keine E0L–Sprache ist.

Aber: $\{a^m b^n c^m : m \geq n \geq 1\}$ und $\{a^m b^n c^m : m \geq 1, n \geq 1\}$ sind E0L!

Kapitel 3

Unentscheidbarkeit

3.1 Beispiele

Gesucht sind Entscheidungsverfahren für bestimmte Fragen. Genauer: ein Algorithmus, der die jeweilige Frage für alle möglichen (syntaktisch korrekten) Eingaben richtig beantwortet. Die Eingabe ist also ein endliche Beschreibung einer Struktur, und der Algorithmus selbst ist auch endlich (eine Programm endlicher Länge), und er soll nach endliche vielen Rechenschritten die Antwort ausgeben.

Die Suche nach solchen Algorithmen ist ja eine wesentliche Antriebskraft der Entwicklung der Mathematik (seit einigen tausend Jahren).

Ein klassisches Beispiel aus der Mathematik ist

Hilberts zehntes Problem (1900) Eingabe: ist ein Polynom in mehreren Variablen, mit ganzzahligen Koeffizienten.

Frage: besitzt das Polynom eine ganzzahlige Nullstelle?

Nun rechnet der Mathematiker nicht immer nur mit Zahlen, sondern zum Beispiel mit (Halb–)Gruppen. Die sind z. B. durch Generatoren und Relationen dargestellt.

Wortproblem in Halbgruppen Gegeben: sind eine endliche dargestellte Halbgruppe sowie zwei Wörter (= Folgen von Generatoren).

Frage: beschreiben beide dasselbe Element?

Logische Formeln sind auch Zeichenfolgen:

Logik Gegeben: ist eine prädikatenlogische Formel,

Frage: ist diese allgemeingültig?

3.2 Algorithmen

Für die entscheibaren Probleme braucht man das nicht so sehr: man sucht eine Weile, findet den passenden Algorithmus, und schreibt ihn in irgendeiner Programmiersprache auf.

Schwierig wird es erst, wenn man trotz (jahrhunderte-)langen Suchens keinen Algorithmus findet. Das könnte ja dann daran liegen, daß überhaupt kein solcher existiert. (Darauf mußten die Mathematiker aber auch erstmal kommen, daß nicht jede Aufgabe algorithmisch lösbar ist. Vergleiche mit der Überraschung der Pythagoräer über die Aussage, daß nicht alle Zahlen rational sind.)

Richtig exakt kann man die Zusammenhänge erst durchschauen, wenn man einen formalisierten Algorithmenbegriff hat (seit < 100 Jahren). Wenn man nachweisen will, daß ein Problem unentscheidbar ist, muß man zeigen: für jeden ALgorithmus, der das Problem angeblich löst, finden wir einen Eingabe, bei der er sich verrechnet (d. h. ein falsches oder gar kein Ergebnis liefert). Also eine Aussage über die Menge *aller* Algorithmen.

3.3 Halteproblem, Diagonal–Verfahren

Das typische algorithmisch unentscheidbare Problem ist das:

Eingabe: eine natürliche Zahl x

Frage: hält das Programm mit der Nummer x an, wenn es die Eingabe x bekommt?

Hier werden Programm(text)e aufgezählt, das geht (z. B. der Länge nach, und innerhalb der Länge lexikografisch). Die vom Programm mit der Nummer x berechnete Funktion nennen wir ϕ_x, und ihren Definitionsbereich D_x. Die Frage heißt dann $x \in D_x$.

Es gibt kein Programm P, das diese Frage immer korrekt beantworten kann. Falls doch so eins existiert, dann betrachten wir das Programm

$$f(x) = \textbf{if } x \in D_x \textbf{ then } \perp \textbf{ else } 0$$

Dabei bedeutet $\perp$ irgendeine nicht haltende Berechnung.

Das Programm f kommt auch in der Aufzählung $\phi.$ vor, sagen wir, mit der Nummer y. Was passiert nun bei der Rechnung $f(y)$? Dann hält $f(y)$, genau dann, wenn wir im else–Zweig landen, also gdw. $y \notin D_y$, also, wenn $f(y)$ nicht hält.

3.4 Universalität, Simulation

Wenn ein Problem entscheidbar ist, dann liegt das an irgendeiner speziellen Eigenschaft des Problems selbst. Die muß man finden, das ist mühevoll, aber wirklich von Fall zu Fall wenig vergleichbar.

Wenn ein Problem jedoch nicht entscheidbar ist, dann hat das in den meisten Fällen einen ganz uniformen Grund: man findet in dem Problem das obige Halteproblem versteckt. Das passiert immer dann, wenn die Struktur, an die die Frage gestellt wird, in der Lage ist, die übliche Berechung von Funktionen (auf Turingmaschinen, zum Beispiel) zu simulieren.

Wir können mit einer prädikatenlogischen Formel ausdrücken, wie sich der Bandinhalt einer Turingmaschien während einer Rechnung verhält. Wir beschreiben dazu ein Gitter (eine Dimension: Bandkoordinate, andere: Zeit).

Wir können die Rechnung einer Turingmaschine auch leicht durch ein Wort-ersetzungssystem simulieren, das wir dann als definierende Relationen einer Halbgruppe auffassen.

Nicht so offensichtlich ist die Unentscheidbarkeit des zehnten Hilbertschen Problems (Beweis ca. 1960?). Wir haben ja da nur Zahlen, und keine Wörter (Zahlenfolgen). Doch! ein entscheidender Gedanke ist, daß jede Zahl eine eindutige Primfaktorzerlegung besitzt. Die Folge der Exponenten ist demnach eine Möglichkeit der Kodierung.

Es gibt auch unkonventioneller, aber trotzdem Turing-vollständige Berechnungsmodelle: *Turing-Eisenban* und das hier ([CG95])

Beginne mit 2, und multipliziere mit dem jeweils am weitesten links stehenden Bruch aus dieser Liste, für den das Ergebnis der Multiplikation eine ganze Zahl ist.

$$\frac{17}{91}\ \frac{78}{85}\ \frac{19}{51}\ \frac{23}{38}\ \frac{29}{33}\ \frac{77}{29}\ \frac{95}{23}\ \frac{77}{19}\ \frac{1}{17}\ \frac{11}{13}\ \frac{13}{11}\ \frac{15}{14}\ \frac{15}{2}\ \frac{55}{1}$$

Welche Zweierpotenzen kommen dabei vor? Hinweis: Anfang ist:

$$2 \to 15 \to 825 \to 725 \to 1925 \to 2275 \to 425 \to 390 \to \dots$$

3.5 Postsches Korrespondenzproblem

ist das wohl meistbenutzte unentscheidbare Problem innerhalb der Theorie formaler Sprachen.

Eingabe: Folge von Paaren von Wörtern $((u_1, v_1), \dots, (u_n, v_n))$

Frage: existiert eine nichtleere Indexfolge $(i_1, \dots, i_l)$, so daß

$$u_{i_1} \cdot \ldots \cdot u_{i_l} = v_{i_1} \cdot \ldots \cdot v_{i_l}$$

Beispiele, usw. siehe http://www.informatik.uni-leipzig.de/~pcp/, und jedes Grundkurslehrbuch. Ich wiederhole hier nur kurz die Beweisidee:

Man kann die Berechnung einer Turingmaschine durch ein PCP simulieren (PCP hat Lösung $\iff$ Maschine hält), indem der jeweils aktuellen Bandinhalt in der PCP–Differenz aufgehoben wird. Für jeweils eine Kopfbewegung der Maschine müssen wir den gesamten Bandinhalt umkopieren (rotieren).

Damit ist PCP-Lösbarkeit unentscheidbar (ansonsten hätten wir ein Entscheidungsverfahren für das Halteproblem).

Mit dem PCP kommen wir zum Thema der Vorlesung (Morphismen) zurück. Es hat gestattet nämlich eine alternative Formulierung:

Eingabe: Morphismen $f, g : \Sigma \to \Delta^*$
Frage: ist die Gleichheitsmenge

$$E(f, g) = \{w \in \Sigma^* : f(w) = g(w)\}$$

größer als $\{\epsilon\}$?

Über Gleichheitsmengen sagen wir später mehr.

3.6 Schwere Eigenschaften von CF–Grammatiken

Dieses Problem sieht harmlos aus:

Eingabe: eine CF–Grammatik G_1
Frage: enthält $L(G_1)$ ein Palindrom der Länge > 1?

Satz: es ist unentscheidbar.

Beweis: Reduktion vom PCP. Gegeben eine PCP–Instanz $((u_1, v_1), \ldots, (u_n, v_n))$, bauen wir die Grammatik G mit Startsymbol S und Regeln

$$S \to \#, \; \{S \to u_i S \overline{v_i} : 1 \leq i \leq n\}$$

Dabei ist # ein neues Zeichen, und $\overline{w}$ ist das Spiegelwort von w. Wenn ein Wort $w \in L(G)$ ein Palindrom ist, also $w = \overline{w}$, dann gibt es eine Indexfolge $(i_1, \ldots, i_l)$ mit $w = u_{i_1} \ldots u_{i_l} \# \overline{v_{i_l}} \ldots \overline{v_{i_1}} = \overline{w} = v_{i_1} \ldots v_{i_l} \# \ldots$, und weil nur ein # vorkommt, sind die Wörter bis dahin gleich (danach aus Symmetriegründen auch), und die Indexfolge gibt eine Lösung des PCP. Damit hätten wir ein Entscheidunsverfahren für PCP, Widerspruch.

Eingabe: zwei CF–Grammatiken G_1, G_2
Frage: Ist der Durchschnitt $L(G_1) \cap L(G_2)$ leer?

Für reguläre Grammatiken ist das natürlich entscheidbar. Ist es in CF wirklich schlimmer? ja, unentscheidbar! Wir zeigen: ein entscheidungsverfahren für obige Frage ergibt ein Entscheidungsverfahren für das Palindrom–Problem, und damit wieder für das PCP, und wir wissen schon, daß es so etwas nicht gibt.

Beweis: G_1 wählen wir wie oben, und G_2 erzeugt einfach *alle* Palindrome der Länge > 1 über dem richtigen Alphabet, und zwar durch die Regeln

$$\{S \to i\#i, S \to iSi : 1 \leq i \leq n\}$$

Dann ist $L(G_1) \cap L(G_2)$ nichtleer gdw. $L(G_1)$ ein Palindrom enthält, und damit das PCP lösbar ist.

Der Durchschnitt zweier kontextfreier Sprachen kann also bereits nicht entscheidbar (aber aufzählbar) sein.

3.7 Unentscheidbarkeit der 0L–Sprach–Äquivalenz

(Dieser Abschnitt ist aus [Sal81].)

Das ist eine Folgerung aus einer weiteren Unentscheidbarkeit für CF–Grammatiken. Wir betrachten dazu nicht die Menge der ableitbaren Wörter (aus Terminalzeichen)

$$L(G) = \{w \; in \Sigma^* : S \to^* w\},$$

sondern die Menge aller Ableitbaren Satzformen (aus Terminalzeichen $\subseteq \Sigma$ und Nichtterminalzeichen $\subseteq M$)

$$F(G) = \{w \in (M \cup \Sigma)^* : S \to^* w\}.$$

Dieses Problem ist unentscheidbar:

Eingabe: CF–Grammatiken G_1, G_2
Frage: Gilt $F(G_1) = F(G_2)$?

Beweisidee: Wir betrachten eine PCP–Instanz, gegeben durch Morphismen $g, h : \Sigma \to \Delta$ und konstruieren drei Sprachen L, L_g, L_h über dem Alphabet $\Sigma \cup \Delta \cup \#$.

Es ist $L = \Sigma^* \# \Delta^*$,

$$\begin{aligned} L_g &= L \setminus \{w\#g(\overline{w}) : w \in \Sigma^+\} \\ L_h &= L \setminus \{w\#h(\overline{w}) : w \in \Sigma^+\} \end{aligned}$$

und $L_{gh} = L_g \cup L_h$.

Offensichtlich gilt $L = L_{gh}$ gdw. $E(g, h) = \{\epsilon\}$.

Wir konstruieren kontexfreie Grammatiken G_1, G_2 mit der Eigenschaft $L(G_1) = L_{gh}, L(G_2) = L$ sowie

$$F(G_1) \setminus L(G_1) = F(G_2) \setminus L(G_2)$$

d. h. die ableitbaren Satzformen unterscheiden sich doch nur in der Menge der ableitbaren Terminalwörter. Daraus folgt dann die Behauptung.

Zur Übung bauen wir die Sprache aller Nicht–Palindrome über zwei Buchstaben $0, 1$:

$$N \to 0N0 \mid 1N1,\ N \to 0A1 \mid 1A0,\ A \to \epsilon \mid 0A \mid 1A$$

Beachte: $A \to \{0,1\}^*$.

Tatsächlich erzeugen wir im folgenden auf der linken Seite Buchstaben, aber rechts Bilder der Buchstaben unter dem jeweiligen Morphismus.

Wir die Regelmenge für G_1 besteht aus vier Teilen. Erstens

$$S_0 \to A \mid B \mid S_4 \mid \#$$

zweitens Regeln für A/g:

$$\begin{aligned} \forall x \in \Sigma : &\quad A \to xAg(x) \\ \forall x \in \Sigma, u \in \Delta^{<|g(x)|} : &\quad A \to xS_1u \\ \forall x \in \Sigma, u \in \Delta^{|g(x)|} \setminus g(x) : &\quad A \to xS_2u \\ \forall y \in \Delta : &\quad A \to S_3y \end{aligned}$$

sowie drittens den entsprechenden Regeln für B und h, und schließlich viertens

$$\begin{aligned} S_1 \to \#, &\quad \forall x \in \Sigma :\ S_1 \to xS_1 \\ S_2 \to S_3, &\quad \forall x \in \Sigma :\ S_2 \to xS_2 \\ S_3 \to \#, &\quad \forall y \in \Delta :\ S_3 \to S_3y \\ \forall x \in \Sigma : S_4 \to xS_4 & \\ \forall y \in \Delta : S_4 \to S_4y & \end{aligned}$$

und erhalten die für G_2 daraus, indem wir $S_1 \to \#$ und $S_3 \to \#$ durch $S_4 \to \#$ ersetzen.

Durch scharfes Draufgucken wird klar, daß sich $F(G_1)$ und $F(G_2)$ höchstens in Terminalwörtern unterscheiden. Deswegen können wir die Frage $F(G_1) = F(G_2)$ doch reduzieren auf $L(G_1) = L(G_2)$.

Wir sehen leicht $L(G_2) = L$, und zeigen $L(G_1) = L_{gh} = L_g \cup L_h$, nach Konstruktion.

Die Menge der ableitbaren Satzformen ist jedoch genau die vom enstsprechenden 0L–System erzeugte Sprache, und damit folgt

Satz 1 *Das Problem*

Eingabe: *zwei 0L–Systeme* G_1, G_2
Frage: $L(G_1) = L(G_2)$

ist nicht entscheidbar.

Kapitel 4

Morphismen

4.1 Gleichheitsmengen

Wir betrachten in diesem Kapitel Gleichheitsmengen von Morphismen.

Definition 8 *Für zwei Morphismen $g, h : \Sigma^* \to \Delta^*$ ist die* Gleichheitsmenge

$$E(g, h) = \{w \in \Sigma^* : g(w) = h(w)\}$$

Damit werden wir im nächsten Kapitel Äquivalenz von D0L–Folgen und –Sprachen entscheiden. Dazu benötigen wir noch diesen Begriff:

Definition 9 *Zwei Morphismen* stimmen auf L *überein, wenn $L \subseteq E(g, h)$.*

insbesondere für die Fälle, daß L eine reguläre, oder ein D0L–Sprache ist. Warum das letztere? Die D0L–Systeme $G_1 = (\Sigma, g_1, w_1)$ und $G_2 = (\Sigma, g_2, w_2)$ sind folgen–äquivalent, wenn $w_1 = w_2$ und die Morphismen g_1 und g_2 auf der Sprache $L(G_1)$ übereinstimmen.

Für jede Gleichheitsmenge gilt $\epsilon \in E(g, h)$ und: wenn $w \in E(g, h)$, dann auch $w^* \subseteq E(g, h)$. Gleichheitsmengen sind also nie leer, und wenn sie mehr als ein Element haben, dann sogar unendlich viele.

Besonders angenehm sind die Fälle, in denen $E(g, h)$ regulär ist. Das ist aber nicht immer so, siehe

$$\begin{aligned} g &: a \mapsto 0, b \mapsto 00 \\ h &: a \mapsto 00, b \mapsto 0 \end{aligned}$$

offensichtlich ist $g(w) = h(w)$ gdw. $|g(w)| = |h(w)|$, und das geschieht genau für $|w|_a = |w|_b$, also eine kontextfreie Sprache.

Es gibt sogar nicht kontextfreie Gleichheitsmengen (siehe Beispiel am Anfang des Skripts), aber es gilt (Übungsaufgabe):

Satz 2 *Jede Gleichheitsmenge $E(g, h)$ ist kontextsensitiv.*

4.2 Balance

Definition 10 *Die* Balance *eines Wortes w bezüglich zweier Morphismen (g, h) ist* $\mathsf{bal}(w) = |g(w)| - |h(w)|$.

Definition 11 *Eine Sprache $L \subseteq \Sigma^*$ hat k–beschränkte Balance bezüglich (g, h), wenn*

$$\forall u \sqsubseteq w \in L : |\mathsf{bal}(u)| \leq k$$

(hierbei: $u \sqsubseteq w$: u ist Präfix von w)

Definition 12

$$E_k(g, h) = \{w \in E(g, h) : \forall u \sqsubseteq w : |\mathsf{bal}(u)| \leq k\}$$

Offensichtlich

$$E_k(g, h) \subseteq E_{k+1}(g, h); \quad E(g, h) = \bigcup_{k \geq 0} E_k(g, h)$$

Satz 3 *Für beliebige Morphismen $g, h : \Sigma^* \to \Delta^*$ und $k \geq 0$ ist $E_k(g, h)$ eine reguläre Sprache.*

Beweis–Idee: bei beschränkter Balance gibt es nur endliche viele Differenzen. Diese nehmen wir als die Zustände eines endlichen Automaten, der $E_k(g, h)$ erkennt.

Beweis: Die Zustandsmenge des Automaten besteht aus dem Zustand s (Start– und Endzustand) sowie Zuständen mit den symbolischen Namen $+w$ und $-w$ für alle $w \in \Delta^{1...k}$.

Rest Übungsaufgabe.

Satz 4 *Für jedes Paar von Morphismen $g, h : \Sigma^* \to \Delta^*$ und reguläre Sprache $L \subseteq \Sigma^*$ gilt*

$$L \subseteq E(g, h) \Rightarrow \exists k : L \subseteq E_k(g, h)$$

Beweis–Idee: betrachte Balance entlang von Schleifen im endlichen Automaten von L.

Beweis: A ist der minimale deterministische Automat für L, er hat a Zustände.

Betrachte ein Wort $w \in L$, für das der Automat eine Schleife durchläuft, d. h. das eine Zerlegung $w = pqr$ besitzt mit $\forall i \geq 0 : pq^i r \in L$.

Dann gilt $\mathsf{bal}(q) = 0$. Wenn nicht, dann finden wir einen Exponenten j mit $\mathsf{bal}(pq^j r) \neq 0$, im Gegensatz zu $pq^j r \in L \subseteq E(g, h)$.

Deswegen finden wir eine obere Schranke für $\{|\mathsf{bal}(w)| : w \in L\}$, wenn wir allein die schleifenfreien Pfade in A betrachten. Die haben höchstens die Länge a, also erhalten wir die Schranke

$$s = a \cdot \max\{|\mathsf{bal}(x)| : x \in \Sigma\}/2$$

(Übung: Woher kommt die Zwei?)

Satz 5 *Das Problem:*

Eingabe*: zwei Morphismen* $g, h : \Sigma^* \to \Delta^*$ *und eine reguläre Sprache* $L \subseteq \Sigma^*$
Frage*: gilt* $L \subseteq E(g, h)$

ist entscheidbar.

Beweis (Variante 1, nicht effektiv): wir rechnen abwechselnd je einen Schritt der beiden Programme:

A) Zähle alle Wörter $w \in L$ auf, und teste für jedes, ob $g(w) = h(w)$. Falls nicht, dann stop mit Ausgabe $L \not\subseteq E(g, h)$.

B) Für alle $k = 0, 1, \ldots$, berechne $E_k(g, h)$, und teste $L \subseteq E_k(g, h)$. Falls ja, dann stop mit Ausgabe $L \subseteq E(g, h)$.

Eines der beiden Programme hält in endlicher Zeit.

Beweis (Variante 2, effektiv): Da die Schranke s für k in $L \subseteq E_k(g, h)$ berechenbar ist (siehe oben), genügt es auch, nur einen Schritt von Programm B auszuführen: teste, ob $L \subseteq E_s(g, h)$.

4.3 Elementare Morphismen (Definition)

Wir definieren den Begriff *elementarer Morphismus* und zeigen dann: Gleichheitsmengen zweier elementarer Morphismen sind regulär.

Daraus folgt *nicht* die Entscheidbarkeit des entsprechenden PCPs, obwohl man denken würde: Gleichheitsmenge regulär $\to$ entscheidbar, ob nur $\{\epsilon\}$ oder mehr $\to$ Lösbarkeit des PCP. Die Konstruktion (des endlichen Automaten) ist jedoch nicht effektiv, und Entscheidbarkeit von PCPs mit elementaren Morphismen ist bis heute offen. Wir gehen später auf entscheidbare Einschränkungen von PCPs ein.

Vorher zeigen wir, daß D0L–Sprach–Äquivalenz für elementare Morphismen entscheidbar ist. Den nicht–elementaren Fall reduzieren wir dann auf den elementaren.

Aber zunächst stellen wir Definition und Eigenschaften von elementaren Morphismen und Sprachen zusammen.

Definition 13 *Ein Morphismus $f : \Sigma^* \to \Delta^*$ heißt* zusammengesetzt*, wenn es ein Alphabet Γ mit $|\Gamma| < |\Sigma|$ gibt und Morphismen $g : \Sigma^* \to \Gamma^*, h : \Gamma^* \to \Delta^*$ mit $f = h \circ g$.*

Ist das nicht der Fall, heißt der Morphismus elementar.

Beispiel/Übungen:

1) Der Morphismus $f : a \mapsto 012, b \mapsto 01, c \mapsto 22$ ist nicht elementar wegen $g : a \mapsto XY, b \mapsto X, c \mapsto YY; \quad h : X \mapsto 01, Y \mapsto 2$ und $|\{X, Y\}| < |\{a, b, c\}|$.

2) Jeder Morphismus $f : \Sigma^* \to \Delta^*$ mit $|\Sigma| > |\Delta|$ ist zusammengesetzt.

3) Jeder löschende Morphismus ist zusammengesetzt.

4) Das Produkt zweier zusammengesetzter Morphismen ist zusammengesetzt.

5) Das Produkt zweier elementarer Morphismen g, h ist nicht notwendig elementar. (Finde eine Beispiel $(g, h) : \Sigma^* \to \Sigma^*$, für das $|\Sigma|$ möglichst klein ist. Geht < 6?)

Definition 14 *Eine endliche Sprache L heißt* zusammengesetzt*, wenn es eine Sprache L' mit $|L'| < |L|$ gibt mit $L \subseteq L'^*$. Ansonsten heißt die Sprache* elementar.

Beispiel: die Sprache $L = \{012, 01, 22\}$ ist zusammengesetzt, denn $L \subseteq \{01, 2\}^*$.

Satz 6 *Wenn der Morphismus $h : \Sigma^* \to \Delta^*$ elementar ist, dann auch die Sprache $\{h(x) : x \in \Sigma\}$.*

Wenn für eine injektive Abbildung $h : \Sigma \to \Delta^*$ die Menge $\{h(x) : x \in \Sigma\}$ elementar ist, dann ist h elementar.

Beweis: Wenn $L = \{h(x) : x \in \Sigma\}$ nicht elementar ist, dann gibt es $L' = \{u_1, \ldots, u_n\}$ mit $n < |\Sigma|$ und $L \subseteq L'^*$. Wir benutzen das Zwischen–Alphabet $\Gamma = \{1, \ldots, n\}$.

Bemerkung: damit können wir ein Programm implementieren, das feststellt, ob ein gegebener Morphismus elementar ist.

4.4 Elementare Morphismen (Hauptsatz)

Das weitere Ziel ist zu zeigen, daß $E(g,h)$ für elementare g,h regulär ist. Dazu benötigen wir diese Aussage:

Satz 7 *Wenn $\Sigma = \{a_1, \ldots, a_k\}$ ein Alphabet ist und $h : \Sigma^* \to \Delta^*$ ein elementare Morphismus, sowie $x, y \in \Sigma^*$ zwei Wörter mit unterscheidlichen Anfangsbuchstaben, für die $h(x) \sqsubseteq h(y)$ gilt, dann*

$$|h(x)| \leq |h(a_1)| + \ldots + |h(a_k)| - k.$$

Beispiel: $h : a \mapsto 012, b \mapsto 01, c \mapsto 20$ ist elementar.

Es gilt $h(bc) = 0120 \sqsubset 01201 = h(ab)$, und $|0120| = 4 \leq 4 = |012| + |01| + |20| - 3$.

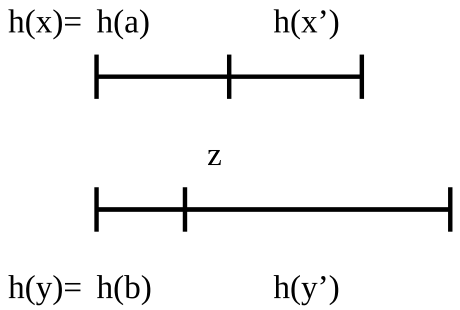
Beweis durch Induktion nach $|h(x)|$:

Induktionsanfang: $|h(x)| = 0$. Weil h elementar ist, sind alle $h(a_i) \neq \epsilon$, und deswegen $|h(a_1)| + \ldots + |h(a_k)| - k \geq 0$.

Induktionsschritt: Wer nehmen an, die Behauptung ist *für alle* h und für alle $|h(x)| \leq p$ gezeigt, und untersuchen $|h(x)| = p + 1$.

Wir bezeichnen den Anfangsbuchstaben von x mit a, so daß $x = ax'$, sowie den Anfangsbuchstaben von y mit b, so daß $y = by'$, und unterscheiden zwei Fälle:

1. $h(b) \sqsubset h(a)$



Wir erklären z durch $h(b) \cdot z = h(a)$ und einen Morphismus h' durch

$$h'(c) = \text{ wenn } c = a \text{ dann } z \text{ sonst } h(c).$$

Zur Erinnerung: $h(a) = h(b) \cdot z$, und wegen $a \neq b$ gilt $h'(b) = h(b)$. Daraus folgt $h(\Sigma) \subseteq h'(\Sigma)^*$, und deswegen ist h' elementar.

Jetzt wenden wir die Induktionsvoraussetzung auf den Morphismus h' und die Wörter $z \cdot h(x') \sqsubseteq h(y')$ an. Beide sind in $h'(\Sigma)^*$, wir müssen verifizieren, daß ihre Urbilder nicht mit den gleichen Buchstaben beginnen:

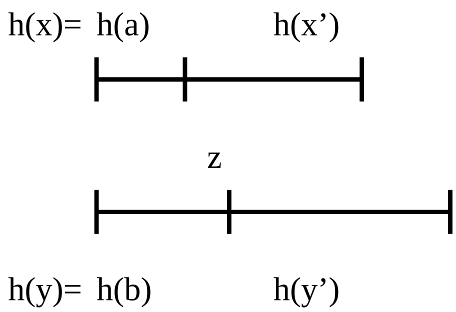
Es gilt $z \cdot h(x') \sqsubseteq h(y')$. Nun ist z nicht von der Form $h(c)$, denn sonst wäre $h(a) = h(b)z = h(b)h(c)$ und damit h nicht elementar.

Damit folgt

$$|z \cdot h(x')| \leq |h'(a_1)| + \ldots + |h'(a_k)| - k = |h(a_1)| + \ldots + |h(a_k)| - |h(a)| - k$$

und daraus durch Addition von $|h(a)|$ die Behauptung.

2. $h(a) \sqsubset h(b)$.



Wir definieren analog z durch $h(a) \cdot z = h(b)$.

(a) $x' = \epsilon$
Dann ist $h(x) = h(a) \sqsubset h(b)$. Daraus folgt $|h(a_1)| + \ldots + |h(a_k)| - k = \sum(|h(a_i)| - 1) \geq |h(b)| - 1 \geq |h(x)|$.

(b) $x' \neq \epsilon$
Wir setzen
$$h'(c) = \text{ wenn } c = b \text{ dann } z \text{ sonst } h(c),$$
und wieder ist h' elementar.

i. x' beginnt nicht mit b.
Dann ist (analog zu vorhin) die Induktionsvoraussetzung anwendbar auf h' und die Wörter $h(x') \sqsubseteq z \cdot h(y')$.

ii. x' beginnt mit b.
Dann beginnt $h(x')$ mit $h(b) = h(a) \cdot z$, und $h(y')$ beginnt mit $z = h'(b)$. Wegen $b \neq a$ ist wiederum die Induktionsvoraussetzung anwendbar, und die Behauptung folgt.

4.5 Elementare Morphismen (Balance)

Satz 8 *Wenn $g, h : \Sigma^* \to \Delta^*$ elementare Morphismen sind, dann $\exists r : E(g, h) \subseteq E_r(g, h)$. Deswegen ist $E(g, h)$ regulär.*

Beweis–Idee: wenn die Balance eines Präfixes einer Lösung zu groß wird, dann ist die Fortsetzung eindeutig, und zwar wenigstens solange, bis die Balance wieder klein wird. Es gibt aber nur endlich viele kleine Balancen, und deswegen ergibt sich die Schranke r als das Maximum über eine endliche Menge.

Lemma 1 *Für elementare Morphismen $g, h : \Sigma^* \to \Delta^*$ gibt es eine Zahl p, so daß für alle $x \in \Sigma^*$ gilt: Wenn $|\mathsf{bal}(x)| > p$, dann gibt es höchstens ein $a \in \Sigma$, für welches $\exists z \in \Sigma^* : g(xaz) = h(xaz)$.*

Beweis: Wir berechnen $p = \max(p_g, p_h)$ mit

$$\begin{aligned} p_g &= |g(a_1)| + \ldots + |g(a_k)| - k + \max\{|g(a)| : a \in \Sigma\} \\ p_h &= |h(a_1)| + \ldots + |h(a_k)| - k + \max\{|h(a)| : a \in \Sigma\} \end{aligned}$$

Wir betrachten den Fall $g(x) \cdot y = h(x)$ mit $|y| > p$. Falls es nun doch $a_1 \neq a_2$ gibt mit $g(xa_1z_1) = g(xa_2z_2) = h(xa_iz_i)$, dann konstruieren wir einen Widerspruch zum Satz 7:

Dazu suchen wir z_i' = der längste Präfix von z_i, für den $|g(az_i')| \le |y|$. Dann gilt für $i \neq j$, daß $g(a_iz_i') \sqsubseteq g(a_jz_j)$. Wir haben andererseits $|g(az_i')| > |y| - \max\{|g(a)| : a \in \Sigma\} \ge |g(a_1)| + \ldots + |g(a_k)| - k$, und das kann nicht sein.

Beweis von Satz 8: Wir betrachten Paare (y_0, b) mit $y_0 \in \Delta^{\le p}, b \in \Sigma$, und

$$\exists x \in \Sigma^*, w \in E(g, h) : xb \sqsubseteq w \wedge g(x) \cdot y_0 = h(x) \wedge \mathsf{bal}(xb) > p$$

D. h. bei Schritt von x zu xb überschreitet die Balance die Schwelle p. Beachte, daß es nur endlich viele solcher Paare gibt (vielleicht unendlich viele x, aber nur endlich viele y_0).

Nach dem Lemma gibt es für jedes Paar eine eindeutige Fortsetzung

$$g(xba_1) \cdot y_1 = h(xba_1)$$

usw. bis schließlich

$$g(xba_1 \ldots a_n) \cdot y_n = h(xba_1 \ldots a_n) \wedge |y_n| \le p$$

(Die Balance muß irgendwann wieder $\le p$ sinken, weil wir anfangs $xb \sqsubseteq w \in E(g, h)$ verlangt haben, und deswegen $\mathsf{bal}(w) = 0$.)

Wir setzen nun

$$r_g = \max\{|y_0|, |y_1|, \ldots, |y_n|\}$$

(das Maximum über die Folgen zu *allen* eben konstruierten Paaren) und bestimmen analog r_h. Die Behauptung des Satzes gilt dann für $r = \max(r_g, r_h)$.

Zusammenfassung: für elementare Morphismen g, h sind in $E(g, h)$ die Balancen beschränkt, und deswegen ist die Gleichheitsmenge regulär.

Warnung: das nützt bei der Suche nach Lösungen des entsprechenden PCPs aber *nichts*, weil wir zur Konstruktion der Schranke r bereits die Lösungen w benötigt haben.

Kapitel 5

Vermeidbare Muster

5.1 Unendliche Wörter

Satz 9 *Mit der Präfix–Ordnung* $\sqsubseteq_L$ *ist* $(\Sigma^*, \sqsubseteq_L)$ *ein unterer Halbverband.*

D. h. zu je zwei Wörtern u, v existiert eine größte untere Schranke $\inf\{u, v\}$. Es ist dies der längste gemeinsame Präfix. (Dieser kann natürlich auch leer sein.)

Definition 15 *Der* Abstand *zweier Wörter* $u, v \in \Sigma^*$ *ist*

$$\operatorname{dist}(u, v) = \text{ wenn } u = v \text{ dann } 0 \text{ sonst } 2^{-|\inf(u,v)|}$$

Satz 10 $(\Sigma^*, \operatorname{dist})$ *ist ein metrischer Raum*

Beweis (Übungsaufgabe): wir müssen die Dreiecksungleichung nachrechnen

$$\operatorname{dist}(u, w) \le \operatorname{dist}(u, v) + \operatorname{dist}(v, w)$$

Es gilt sogar

$$\operatorname{dist}(u, w) \le \max(\operatorname{dist}(u, v), \operatorname{dist}(v, w))$$

d. h. wir haben eine Ultrametrik.

Der Raum $(\Sigma^*, \operatorname{dist})$ ist nicht vollständig, die Fundamentalfolge $(1, 11, 111, \ldots)$ hat keinen Limes. Wir können den Raum aber abschließen. Die Häufungspunkte sind (einseitig) unendliche Wörter.

Definition 16 *Ein* unendliches Wort *über dem Alphabet* Σ *ist eine Abbildung* $\mathbb{N} \to \Sigma$.

Die Menge aller unendlichen Wörter über Σ *heißt* Σ^ω.

Wir setzen $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$

Beachte, daß die Metrik–Definition so bleiben kann: der längste gemeinsame Präfix von u, v ist entweder endlich, oder u ist gleich v.

Satz 11 $(\Sigma^\infty, \operatorname{dist})$ *ist ein vollständiger kompakter metrischer Raum.*

Definition 17 *Ein D0L–System* $G = (\Sigma, g, w)$ *definiert ein (unendliches) Wort* $\lim G = \lim S(G)$ *als Limes der D0L–Folge. (Falls dieser existiert.)*

5.2 Muster

Für diesen Abschnitt fixieren wir ein Alphabet Σ wie üblich sowie ein dazu disjunktes Alphabet V (von Variablen).

Definition 18 *Ein* Muster *ist ein Wort aus V^*.*

Ein Wort $w \in \Sigma^\infty$ enthält *ein Muster m, geschrieben $m \ll w$, falls es einen nicht löschenden Morphismus $h : V^* \to \Sigma^*$ gibt, für den $h(m) \sqsubseteq w$.*

Andernfalls vermeidet *w das Muster m.*

Beispiel: das Wort $w = abaababaabaababaababa$ enthält das Muster $m = x^3 = xxx$, denn für $h : x \mapsto aba$ ist $m = abaab\ aba\ aba\ aba\ baababa = abaab\ h^3(x)\ baababa$.

Definition 19 *Ein Muster m heißt k–vermeidbar, wenn es ein Alphabet Σ der Größe k sowie ein unendliches Wort $w \in \Sigma^\omega$ gibt, das m vermeidet.*

Sonst heißt das Muster k–unvermeidbar, und $R_k(m)$ ist die kleinste Zahl n, für die jedes Wort der Länge $\geq n$ das Muster m enthält.

Ein Muster m heißt unvermeidbar*, wenn es über* jedem *Alphabet unvermeidbar ist.*

Bemerkung : R wegen Ramsey.

5.3 Das Wort von Thue ...

Wir konstruieren ein Wort $t \in \{a,b\}^\omega$, das xxx über zwei Buchstaben vermeidet.

Definition 20 *Das* Thue–*Wort ist der Limes der vom D0L–System* $T = (\{a,b\}, \tau, a)$ *erzeugten Folge, wobei*

$$\tau : a \mapsto ab, b \mapsto ba$$

Das Wort fängt so an

$$t = abbabaabbaababbabaababbaabbabaabbaababba\ldots$$

Bemerkung: wir können, ähnlich wie beim Fibonacci–Wort, eine Zerlegung direkt angeben: für die Präfixe des Thue–Wortes $t_k = \tau^k(a)$ gilt

$$\begin{aligned} t_0 &= a, & u_0 &= b \\ t_{k+1} &= t_k u_k, & u_{k+1} &= u_k t_k \end{aligned}$$

Bemerkung: eine weitere äquivalente Darstellung ist: in $t = x_0 x_1 x_2 \ldots$ gilt:

$$x_k = a \iff \text{die Binärdarstellung von } k \text{ enthält eine gerade Anzahl von 1}$$

Man kann das verallgemeinern: ein unendliches Wort $w = (w_0, w_1, \ldots)$ heißt *automatisch* (zur Basis b), wenn der Buchstabe w_i auf der Stelle i durch einen endlichen Automaten berechnet werden kann, der als Eingabe die Darstellung von i (zur Basis b) bekommt.

Das Thue–Wort ist 2–automatisch.

Offensichtlich gibt es Zusammenhänge zwischen automatischen Wörtern und uniformen Morphismen (d. h. für alle $x \in \Sigma$ stimmen die Längen $|h(x)|$ überein).

5.4 ... ist kubikfrei

Das Thue–Wort ist nicht quadratfrei, aber:

Definition 21 *Ein* starkes Quadrat *ist ein Wort der Form yyc, wobei c der Anfangsbuchstabe von y ist.*

Satz 12 *t enthält keine starken Quadrate.*

Beweis: Angenommen, t enthält yyc, Wir nehmen davon das kürzeste und am weitesten links liegende Vorkommen. Es gibt dann ein k, so daß das Muster bereits in $t_k = \tau^k(a)$ liegt.

1. Weder aaa noch bbb sind Teilworte von t_k.
2. Weder $ababa$ noch $babab$ sind Faktoren von t_k.
3. Jeder Faktor von t der Länge ≥ 5 enthält aa oder bb.
4. Für die Buchstaben x_i von t gilt

   $$x_i = x_{2i} \neq x_{2i+i}.$$

   Die Buchstaben auf den geraden Plätzen sind also durch ihre unmittelbaren rechten Nachbarn eindeutig festgelegt (und die ungeraden durch die linken).
5. Jeder Faktor aa oder bb von t_k beginnt auf einer geraden Position.
6. Wenn $|y| \geq 2$, dann $|yyc| \geq 5$, und enthält damit aa oder bb. Sogar zweimal (falls in y, dann klar; falls an der Grenze yy, dann auch bei yc, und umgekehrt). Wenn es aber zweimal vorkommt, dann liegt dazwischen eine gerade Zahl von Zeichen. Was dazwischen liegt, ist aber genau die Länge von y. Deswegen brauchen wir nur Faktoren yyc zu betrachten, für die $|y|$ eine gerade Zahl ist.
7. Falls das erste y auf einer geraden Position $2i$ beginnt, dann finden wir einen Faktor $y'y'c'$ ab der Position i mit $\tau(y') = y$.
8. Falls y auf ungerader Position beginnt, dann auch.

Folgerung: t vermeidet das Muster xxx.

5.5 Quadratfreiheit

Wir bezeichen die Buchstaben des Thue–Wortes mit $t = (t_1, t_2, \dots)$, und konstruieren daraus ein unendliches Wort über dem Alphabet $\{a, b\}^2$ (der Größe vier) durch

$$s = (s_1, s_2, \dots) \quad \text{mit} \quad s_i = (t_i, t_i + 1)$$

Bei Bedarf können wir umbenennen: $1 = (a, a), 2 = (a, b), 3 = (b, a), 4 = (b, b)$.

Satz 13 *s vermeidet das Muster xx.*

Beweis: falls

$$(s_{i+1}, \dots, s_{i+d}) = (s_{i+d+1}, \dots, s_{i+2d}),$$

dann gilt

$$((t_{i+1}, t_{i+2}), \dots, (t_{i+d}, t_{i+d+1})) = ((t_{i+d+1}, t_{i+d+2}), \dots, (t_{i+2d}, t_{i+2d+1}))$$

Damit ist

$$t_{i+1} = t_{i+d+1} = t_{i+2d+1}, \quad \forall 1 \leq k \leq n : t_{i+k} = t_{i+d+k}$$

und damit haben wir ein starkes Quadrat in t im Widerspruch zum vorigen Abschnitt.

Die Alphabetgröße vier läßt sich noch reduzieren: Quadrate sind auch 3–vermeidbar.

Satz 14 *Wenn wir im Wort s jeden Buchstaben 4 durch 1 ersetzen, dann ist das entstehende Wort quadratfrei.*

Beweis: Übung. Benutze, daß in s die Teilwörter $11, 14, 41, 44$ nicht vorkommen.

Ein anderes quadratfreies Wort über $\Sigma = \{a, b, c\}$ erhält man aus dem D0L–System $G = (\Sigma, g, a)$ mit

$$g : a \mapsto abcab, b \mapsto acabcb, c \mapsto acbcacb$$

Satz 15 *Wenn $w \in \Sigma^*$ quadratfrei ist, dann auch $g(w)$.*

Beweis: Übung.

Folgerung: alle $g^k(a)$ sind quadratfrei, also auch das vom D0L–System $(\{a, b, c\}, g, a)$ erzeugte unendliche Wort.

Selbst Limites von Morphismen, die die Quadratfreiheit nicht erhalten, können quadratfrei sein. Im nächsten Abschnitt betrachten wir

$$\mu : a \to abc, b \to ac, c \to b.$$

Übung: zeige durch ein Beispiel, daß μ die Quadratfreiheit nicht erhält.

5.6 Das Wort von Morse ist quadratfrei

Das ist ein unendliches Wort m über drei Buchstaben, das das Muster xx vermeidet. Es ist der Limes der Wortfolge des D0L–Systems $M = (\{0, 1, 2\}, \mu, 0)$ mit

$$\mu \quad : \quad 0 \mapsto 012, 1 \mapsto 02, 2 \mapsto 1$$

und beginnt so

$$m = 012021012102012021020121012021012102\ldots$$

Satz 16 *m ist* quadratfrei*, d. h. vermeidet das Muster xx.*

Beweis: t ist ein homomorphes Bild von m. Wir zeigen: ein Quadrat in m ergibt dann starkes Quadrat in t, dort sind aber keine.

Lemma 2 *Für das Thue–Wort t und das Morse–Wort m:*

$$t = h(m) \quad mit \quad h : 0 \mapsto abb, 1 \mapsto ab, 2 \mapsto a$$

Beweis: Wir müssen dazu diese Diagramme verifizieren:

$$\begin{array}{ccc} 0 & \stackrel{\mu}{\to} & 012 \\ {\scriptstyle h}\downarrow & & {\scriptstyle h}\downarrow \\ abb & \stackrel{\tau}{\to} & abbaba \end{array} \qquad \begin{array}{ccc} 1 & \stackrel{\mu}{\to} & 02 \\ {\scriptstyle h}\downarrow & & {\scriptstyle h}\downarrow \\ ab & \stackrel{\tau}{\to} & abba \end{array} \qquad \begin{array}{ccc} 2 & \stackrel{\mu}{\to} & 1 \\ {\scriptstyle h}\downarrow & & {\scriptstyle h}\downarrow \\ a & \stackrel{\tau}{\to} & ab \end{array}$$

Lemma 3 *Wenn m das Muster xx enthält, dann enthält t ein starkes Quadrat.*

Beweis: Also gilt $m = uxxv$. Dann ist $t = h(u)h(x)h(x)h(v)$. Beachte, daß $h(x)$ und $h(v)$ in jedem Fall mit a beginnen.

5.7 Zimins Muster und Zimins Algorithmus

Wir betrachten jetzt globale Vermeidbarkeit. (Zur Erinnerung: “unvermeidbar” = “unvermeidbar über *jedem* Alphabet”.)

Satz 17 *Wenn $m \in V^+$ unvermeidbar ist, und die Variable x nicht in m vorkommt, dann ist mxm unvermeidbar.*

Beweis: Wir wählen $l = R_s(m)$ und betrachten ein Wort der Länge $(s^l+1)(l+1)-1 = s^l(l+1)+l$ Das können wir uns vorstellen als s^l+1 Stück von Wörtern der Länge l, die durch je einen Buchstaben getrennt sind.

$$(1, \ldots, l), (l+2, \ldots, 2l+1), \ldots, (k(l+1)+1, \ldots k(l+1)+l), \ldots$$

Da es nur s^l verschiedene Wörter in Σ^l gibt, befinden sich unter diesen s^l+1 wenigstens zwei gleiche, und diese sind durch wenigstens einen Buchstaben getrennt. Jedes der beiden enthält zudem die gleiche Instanz von m. Daraus folgt

$$R_s(m \cdot x \cdot m) \leq s^l(l+1)+l \quad \text{mit} \quad l = R_s(m)$$

Definition 22 *Die* Zimin–Muster *sind*

$$Z_0 = \epsilon \qquad Z_k = Z_{k-1} x_k Z_{k-1}$$

Beispiel: $Z_3 = x_1x_2x_1x_3x_1x_2x_1$

Satz 18 *Alle Z_k sind unvermeidbar.*

Beispiel: Offensichtlich ist $R_2(Z_0) = 0$ und $R_2(Z_1) = 1$. Nach unserem Beweis ist $R_2(Z_2) \leq 2^1(1+1)+1 = 5$, und das ist auch exakt. Weiter ist $R_2(Z_3) \leq 2^5(5+1)+5 = 197$, aber durch tatsächliches Absuchen findet man $R_2(Z_3) = 29$.

Satz 19 *Es ist entscheidbar, ob ein Muster m vermeidbar ist.*

(Siehe Lothaire: ACW)

Beweisidee: Man legt fest, welche Mustervariablen man streichen darf, so daß die (Un–)Vermeidbarkeit erhalten bleibt, und versucht, durch mehrere Schritte auf diese Weise ϵ (oder ein anderes als unvermeidbar bekanntes Muster) zu erhalten.

Aus Z_{n+1} entsteht durch Streichen von x_0 das Muster Z_n.

Für $m = xyxzxy$ löscht man x und erhält yzy, und das ist Z_2.

Man definiert dazu eine Relation $<$ auf dem Muster–Alphabet V durch $x <_1 y$, falls m das Teilwort xy enthält, und betrachtet die transitive (nicht reflexive!) Hülle $<$ von $<_1$.

Eine Menge $F \subseteq V$ heißt *frei*, wenn $\forall x, y \in F : x \not< y$.

Wir schreiben $m \to m'$, wenn m' aus m durch Löschen aller Elemente einer freien Teilmenge entsteht.

Lemma 4 *Wenn $p \to^* q$ und q unvermeidbar, dann auch p unvermeidbar.*

Lemma 5 *Wenn $p \not\to^* \epsilon$, dann ist p vermeidbar.*

5.8 Schranken und offene Fragen

Für den Fall, daß $|V| = |\Sigma| = 2$, ist die Vermeidbarkeit durch Arbeiten von Cassaigne (199?) vollständig geklärt.

Diese Muster sind unvermeidbar:

m	x	xy	x^2	x^2y	xyx	x^2yx	xy^2x	x^2y^2	$xyxy$	x^2yx^2	x^2yxy
$R_2(m)$	1	2	4	5	5	10	11	12	19	19	39

Beispiel: $x^2yxy \not\leq 10101010110011100011110000111000110010$ (Länge 38).

Alle längeren binären Muster sind über zwei Buchstaben vermeidbar. Solche unvermeidbaren Mustermengen sind immer endlich. (Siehe dazu Arbeiten von Cassaigne und Roth.)

Definition 23 *Wir bezeichnen mit $l_{n,k}$ die kleinste Zahl l, für die alle Muster der Länge l mit n Variablen k–vermeidbar sind:*

$$l_{n,k} = \min\{l \mid \forall m \in \{x_1, \ldots, x_n\}^{\leq l} : R_k(m) = \infty\}$$

Die obige Tabelle zeigt $l_{2,2} = 6$. Es ist bekannt, daß $l_{3,3} = 8$.

Satz 20 *Für $k \geq 2$ und alle n gilt: $l_{n,k} \leq l_{n,2} < \infty$.*

Definition 24 *Der* Index *eines Musters m ist die Zahl*

$$I(m) \quad = \quad \min\{k : R_k(m) = \infty\}$$

Unvermeidbare Muster haben unendlichen Index: $I(xyx) = \infty$.

Beispiele: $I(xx) = 3, I(abwbcxcaybazac) = 4$.

Muster mit größerem (endlichen) Index sind bis heute nicht bekannt.

Vermutung: wenn m vermeidbar ist, dann ist m sogar 4–vermeidbar.

Die Entscheidbarkeit des Problems

- gegeben $m \in V^+$ und $s \in \mathbb{N}^+$
- gefragt: gilt $R_s(m) = \infty$

ist bis heute offen.

Es ist beispielsweise noch nicht bewiesen, daß $xxyyzz$ über zwei Buchstaben vermeidbar ist. (Obwohl eine Backtracking–Suche, siehe nächsten Abschnitt, deutlich auf Vermeidbarkeit hinweist.)

5.9 (H)D0L–Vermeidbarkeit

Die übliche Vorgehensweise bei der Betrachtung der Vermeidbarkeit eines Musters m über einem Alphabet der fixierten Größe k ist:

1. untersuche durch Backtracking, ob m vermeidbar ist (falls nein, ist das ein endlicher Suchbaum)
2. falls die Suche zu lange dauert, dann vermute Vermeidbarkeit, und finde eine effektive Beschreibung eines unendlichen m–freien Wortes.

Das klappt oft (siehe Quadratfreiheit), aber der zweite Teil dieses Plans ist aus zwei Gründen nicht sicher:

1. es gibt noch kein Entscheidungsverfahren für die Frage: Vermeidet ein (H)D0L–System G ein Muster M?
2. es ist nicht bewiesen, daß aus k–Vermeidbarkeit bereits k–HD0L–Vermeidbarkeit folgt.

Im Gegenteil, es ist bekannt, daß es k–vermeidbare Muster gibt, die aber nicht k–D0L–vermeidbar sind:

Satz 21 *Die Muster $xyyzxyyz$ und $xyxzxyxz$ sind 2–vermeidbar, aber nicht 2–D0L–vermeidbar.*

Beweis: 1. $xyyzxyyz$ ist nicht 2–D0L–vermeidbar:

Jedes 2–D0L–Wort enthält beliebig lange Quadrate. (Quadrate sind nicht 2–vermeidbar, als entsteht irgendwann eines, und aus diesem entstehen durch den Morphismus immer längere.)

Unter diesen beliebig langen Quadraten qq betrachten wir die Wörter q. Diese enthalten selbst irgendwann ein Quadrat, und das nehmen wir als yy.

2. $xyyzxyyz$ ist 2–HD0L–vermeidbar:

Sogar $xyzxyz$ ist 2–HD0L–vermeidbar. und zwar durch das Wort $g(f^\omega(a))$ für

$$\begin{aligned} f &: a \mapsto abc, b \mapsto ac, c \mapsto b \\ g &: a \mapsto abab, b \mapsto aabb, c \mapsto baaa \end{aligned}$$

Der Anteil $f^\omega(a)$ ist also das quadratfreie Wort von Morse.

Genau besehen, ist $xyzxyz$ ein andere Name für "Quadrate der Länge ≥ 3".

Definition 25 *Ein Muster m heißt* schwach *k–vermeidbar, wenn es eine Zahl l und ein unendliches Wort w auf k Buchstaben gibt, so daß für keinen Morphismus $h : V^* \to \Sigma_k^*$ mit $\forall v \in V : |h(v)| \geq l$ gilt $h(m) \subset w$.*

Satz 22 *Jedes k–vermeidbare Muster ist schwach 2–vermeidbar.*

Beweis: Übung.

Kapitel 6

E0L

Wir haben mit E0L einen kräftigen Formalismus zur Beschreibung von Sprachen. Bei *jedem* solchen stellen sich automatisch (wenigstens) diese Fragen:

- Normalformen
- Abschluß–Eigenschaften
- Entscheidbarkeiten

Das ergibt sich ganz von selbst, wenn man Theorie formaler Sprachen betreibt. Vergleiche beispielsweise kontextfreie Grammatiken:

- Normalformen: Chomsky: $X \to Y_1Y_2|a$, Greibach: $X \to aY_1 \dots Y_n$
- CF abgeschlossen unter Vereinigung, Stern, aber nicht Durchschnitt (jedoch Schnitt mit REG), nicht unter Komplement
- Gegeben G, w, gefragt $w \in L(G)$ ist entscheidbar.

  Die Frage $L(G_1) = \emptyset$ ist entscheidbar.

  Gegeben G_1, G_2, gefragt $L(G_1) = L(G_2)$ ist nicht entscheidbar.

6.1 E0L–Synchronisation

Definition 26 *Ein E0L–System* (Σ, Δ, s, w_0) *heißt* synchronisiert, *wenn für alle* $x \in \Sigma$ *und* $w \in s^+(x)$ *gilt:* $w \notin \Sigma^*$.

Das bedeutet, daß aus Terminalzeichen x niemals ein Terminalwort abgeleitet werden kann. Das früher angegebene Beispiel für $\{a^n b^n c^n : n \geq 1\}$ war synchronisiert.

Satz 23 *Zu jedem E0L–System* $G = (\Sigma, \Delta, s, w_0)$ *gibt es ein äquivalentes synchronisiertes E0L–System* $G' = (\Sigma, \Delta', s', w_0')$

(äquivalent heißt $L(G) = L(G')$)

Beweis–Idee: wir verdoppeln das Terminal–Alphabet.

Beweis: neue Buchstaben sind $\overline{\Sigma} = \{\overline{x} : x \in \Sigma\}$ und $\overline{\Delta} = \{\overline{x} : x \in \Delta\}$ sowie ein neues Symbol F. Dann ist $\Delta' = \overline{\Delta} \cup \overline{\Sigma} \cup F$. Für jede Regel $x \to s(x)$ in G nehmen wir die Regel $\overline{x} \to \overline{s(x)}$ in G' auf, sowie die Regeln

$$\begin{aligned} \forall x \in \Sigma : \quad & \overline{x} \to x, x \to F \\ & F \to F \end{aligned}$$

Das Startwort ist $\overline{w_0}$.

Das entstehende System ist offensichtlich synchronisiert. Das F heißt Synchronisations–Symbol. Wenn es einmal auftaucht, kann es nie mehr verschwinden. Da $F \notin \Sigma$, entsteht dann daraus nie ein Ausgabewort.

Bemerke, daß für Terminal $x \in \Sigma$ jeweils nur eine einzige Regel $x \to F$ im System ist.

Übung: zeige, daß man außerdem noch erreichen kann, daß das Startwort die Länge 1 hat, und in keiner weiteren rechten Regelseite benutzt wird.

6.2 E0L = EP0L

Wir erinnern uns, daß die Modifikatoren modular wirken, also P = propagating (nichtlöschend) angewandt auf E0L ergibt EP0L: propagierende E0L–Systeme.

Satz 24 *Die Sprachklassen E0L und EP0L stimmen überein.*

Beweis–Beispiel: Gegeben ist ein System mit den Regeln $a \to aba, b \to \epsilon$. Beachte, daß das System mit nur der Regel $a \to aa$ nicht äquivalent ist, weil wir aba nicht ableiten können.

Wir bauen die Regeln

$$S \to \{a, A\}, A \to \{AA, aba\}, a \to F, b \to F, F \to F$$

mit F als Synchronisations–Zeichen.

Beweis–Idee: wir ersetzen die Buchstaben $x \in \Sigma \cup \Delta$ durch Paare (x, S), wobei $S \subseteq \Sigma \cup \Delta$ mit der Bedeutung: aus (x, S) sollen nur Wörter abgeleitet werden, die wirklich *alle* Zeichen aus S enthalten.

Beweis: TODO

Jetzt haben wir endlich eine obere Schranke für die Einordnung der L–Sprachen in die Chomsky–Hierarchie: nämlich unterhalb der kontextsensitiven Sprachen.

Folgerung 1 *E0L* $\subseteq$ *CS*

Beweis: E0L = EP0L, und ein propagierendes System kann man durch eine monotone Grammatik simulieren.

Folgerung 2 *Die Frage:*

- *Eingabe: ein E0L–System G und ein Wort w*
- *Frage: gilt $w \in L(G)$*

ist entscheidbar.

Beweis: weil die gleiche Frage sogar für CS entscheidbar ist.

6.3 Abschluß–Eigenschaften von E0L

Satz 25 *Die E0L–Sprachen sind abgeschlossen unter Vereinigung, Konkatenation, positiver Hüllenbildung und Schnitt mit regulären Sprachen.*

Das heißt: für beliebige E0L–Sprachen L_1 und L_2 und regulären Sprachen R sind auch die folgenden Sprachen in E0L: $L_1 \cup L_2$, $L_1 \cdot L_2$, L_1^+ und $L_1 \cap R$.

Beweis (Katenation): Wir können für L_i o.B.d.A. ein synchronisiertes E0L–System $(\Sigma_i, \Delta_i, s_i, w_i)$ annehmen, mit disjunkten Nicht–Terminal–Alphabeten Δ_i und Synchronisations–Zeichen F_i sowie Startsymbolen S_i.

Wir erhalten ein E0L–System für $L_1 \cdot L_2$ durch ein neues Startsymbol S, und die Regeln

$$\begin{aligned} \forall i \in \{1,2\}, x \in \Delta_i : &\quad x \to s_i(x) \\ \forall x \in \Sigma_i : &\quad x \to F \\ &\quad S \to S_1' S_2', F \to F \\ \forall i \in \{1,2\} : &\quad S_i' \to S_i', S_i' \to S_i \end{aligned}$$

Übung: erkläre den Sinn der Regeln aus der letzten Zeile!

Beweis (Vereinigung, Hülle) Übungsaufgabe. Geht genauso wie Katenation.

Beweis–Idee (Schnitt mit regulärer Sprache): Wir nehmen an, daß wir das E0L–System durch einen EP0L–System G gegeben haben.

Wir nehmen einen deterministischen Automaten A für R mit Zustandsmenge Q und setzen als Nicht–Terminial–Menge für das zu konstruierende E0L–System G_R die Menge $Q \times (\Sigma \cup \Delta) \times Q$.

Ein solches Nicht–Terminal (p, x, q) soll in G_R genau die Wörter erzeugen, die in G aus x hervorgehen *und* in A den Zustand p in den Zustand q überführen.

Beachte, daß es hierbei wesentlich ist, daß nicht $x \to^+ \epsilon$, denn dann brauchten wir Epsilon–Übergänge im Automaten, das wäre aber nicht deterministisch. (— na und?)

Folgerung 3 *Die Frage:*

- *Eingabe: ein E0L–System $G = (\Sigma, \Delta, s, w_0)$, eine reguläre Grammatik R*
- *Frage: gilt $L(G) \subseteq L(R)$*

ist entscheidbar.

Beweis: Dann ist auch die Sprache $R' = \Sigma^* \setminus R$ regulär, und der Durchschnitt $L(G) \cap R'$ ist eine E0L–Sprache. Wenn er leer ist, dann gilt $L(G) \subseteq L(R)$. Die Leere einer E0L–Sprache läßt sich aber effektiv testen.

Übungsaufgabe: wie?

Kapitel 7

D0L–Äquivalenz

Für gegebene Systeme G_1, G_2 untersuchen wir Folgen–Äquivalenz $S(G_1) = S(G_2)$ und Sprach–Äquivalenz $L(G_1) = L(G_2)$.

(Diese Kapitel folgt wörtlich dem Abschnitt III.2 aus [RS80].)

7.1 Beispiele

Für die D0L–Systeme $G_i = (\{a,b\}, h_i, aba)$ mit

$$h_1 : a \mapsto aba, b \mapsto \epsilon \qquad h_2 : a \mapsto a, b \mapsto ba^2b$$

gilt $S(G_1) = S(G_2)$.

Für die D0L–Systeme $G_i = (\{a,b\}h_i, ab)$ mit

$$h_3 : a \mapsto abb, b \mapsto aabba \qquad h_4 : a \mapsto abbaabb, b \mapsto a$$

stimmen $S(G_3)$ und $S(G_3)$ nur auf den ersten drei Elementen überein.

Gibt es eine berechenbare Funktion $s : \mathbb{N} \to \mathbb{N}$ mit der Eigenschaft: $S(G_1) = S(G_2) \iff$ die ersten $s(n)$ Folgenglieder von $S(G_1)$ und $S(G_2)$ stimmen überein (wobei $n = |\Sigma|$)?

Der hier angegebene Beweis liefert so eine Schranke nicht direkt. Es sind jedoch berechenbare Schranken bekannt, allerdings sind diese sehr schnell wachsend.

Andererseits kennt man keine Systeme G_1, G_2, die nicht folgen–äquivalent sind, aber trotzdem sehr lange übereinstimmen. Man vermutet, daß $s(n) = 2n$ als Schranke ausreicht. (Siehe das obige Beispiel: $3 < 2 \cdot 2$)

Übungs(Programmier)Aufgabe: untersuche diese Vermutung.

7.2 D0L–Äquivalenz für elementare Morphismen

Satz 26 *Das folgende Problem ist entscheidbar.*

- Eingabe*: ein D0L–System G, zwei elementare Morphismen h_1, h_2*
- Frage*: Ist $L(G) \subseteq E(h_1, h_2)$*

Beweis: Nach Satz 8 ist $E = E(h_1, h_2)$ regulär. Aber Vorsicht, der Beweis liefert keine effektive Konstrukion eines Automaten für E.

Wir kombinieren deswegen das Entscheidungsverfahren aus zwei Semi–Algorithmen:

- Wir testen für $w_0, w_1, \dots$ aus der D0L–Folge $S(G)$, ob $h(w_i) = g(w_i)$, und halten bei Ungleichheit mit Ausgabe "$L(G) \not\subseteq E(h_1, h_2)$."
- Wir bilden für $k = 0, 1, \dots$ nach Satz 3 die reguläre Grammatik R_k für die Sprache $E_k(h_1, h_2) \subseteq E$, und testen nach Folgerung 3, ob $L(G) \subseteq L(R_k)$. Falls ja, halten wir mit Ausgabe "$L(G) \subseteq E(h_1, h_2)$."

Es ist klar, daß die Ausgabe in beiden Fällen korrekt ist. Wir müssen uns davon überzeugen, daß stets einer der beiden Semi–Algorithmen terminiert. Falls $L(G) \not\subseteq E$, dann gibt es tatsächlich dafür einen Beweis w_i mit $h_1(w_i) \neq h_2(w_i)$. Falls andererseits $L(G) \subseteq E$, und E regulär, dann gibt es nach Satz 4 ein k, so daß $E \subseteq E_k(h_1, h_2)$, d. h. $E = E_k(h_1, h_2)$. Auf dieses k warten wir mit der zweiten Aufzählung. □

Satz 27 *Die zwei D0L–Systeme $G_i = (\Sigma, h_i, w_i)$ sind genau dann folgen–äquivalent, wenn $w_1 = w_2$ und die Morphismen h_1 und h_2 auf der Sprache $L(G_1)$ übereinstimmen, also $L(G_1) \subseteq E(h_1, h_2)$.* □

Mit Satz 27 erhalten wir

Folgerung 4 *Das folgende Problem ist entscheidbar.*

- Eingabe*: zwei D0L–Systeme $G_i = (\Sigma_i, h_i, w_i)$ mit elementaren Morphismen h_i*
- Frage*: Ist $S(G_1) = S(G_2)$* □

7.3 Konstruktion elementarer Morphismen

Satz 28 *Für beliebige Morphismen h_1, h_2 gibt es eine Indexfolge $(k_1, \ldots, k_n) \in \{1,2\}^*$, sowie* elementare *Morphismen p_1, p_2, und einen Morphismus f, so daß*

$$\forall i \in \{1,2\}: \quad h_i \circ h_{k_1} \circ \ldots \circ h_{k_n} = p_i \circ f$$

Beweis: Für eine beliebige Indexfolge $(k_1, \ldots, k_n)$ können wir das Produkt $h_{k_1} \circ \ldots \circ h_{k_n}$ als $g \circ f$ darstellen, mit Morphismen $\Sigma \xrightarrow{f} \Sigma' \xrightarrow{g} \Sigma$, so daß $|\Sigma'|$ minimal wird.

Weil auf jeden Fall $0 < |\Sigma'| \leq |\Sigma|$, existieren eine Folge $(k_1, \ldots, k_n)$ sowie Morphismen g, f, für die $|Sigma'|$ das globale Minimum annimmt.

Wir setzen $p_i = h_i \circ g$ und zeigen indirekt, daß beide p_i elementar sind:

Ansonsten existiert eine Zerlegung $h_i \circ g = g' \circ f'$ mit $\Sigma' \xrightarrow{f'} \Sigma'' \xrightarrow{g'} \Sigma$ und $|\Sigma''| < |\Sigma|$. Es gilt jedoch $h_i \circ h_{k_1} \circ \ldots \circ h_{k_n} = g' \circ f' \circ f$, und damit ist $\Sigma \xrightarrow{f' \circ f} \Sigma'' \xrightarrow{g'} \Sigma$ eine Zerlegung der Indexfolge $(i, k_1, \ldots, k_n) \in \{1,2\}^*$ mit kleinerem Zwischenalphabet, im Widerspruch zur Minimalität von $h_{k_1} \circ \ldots \circ h_{k_n}$. $\square$

Lemma 6 *Diese $(k_1, \ldots, k_n), p_1, p_2, f$ lassen sich effektiv konstruieren.*

Beweis: Da wir wissen, daß die gesuchten Dinge existieren, können wir sie mit einem Semi–Algorithmus suchen. Wir zählen *alle* Folgen $(k_1, \ldots, k_n) \in \{1,2\}^*$ auf und prüfen, ob es passende Morphismen f und elementare p_i gibt (das sind jeweils nur endlich viele Tests). $\square$

Übung: zwei Morphismen h_1, h_2 hernehmen, die beide nicht elementar sind, und die obige Konstruktion ausführen!

7.4 D0L–Folgen–Äquivalenz

Wir wollen die Frage $S(G_1) = S(G_2)$ entscheiden. Für D0L–Systeme mit elementaren Morphismen können wir das, und den allgemeinen Fall reduzieren wir jetzt auf den bekannten.

Lemma 7 *Für D0L–Systeme $G_i = (\Sigma, h_i, w_i)$ mit den in Satz 28 erklärten $(k_1, \ldots, k_n), p_1, p_2, f$ definieren wir $g_i = h_i \circ h_{k_1} \circ \ldots \circ h_{k_n}$, und damit die D0L–Systeme $G_i^j = (\Sigma, g_i, h_i^j(w_i))$. Dann gilt*

$$S(G_1) = S(G_2) \iff \forall 0 \le j \le n : S(G_1^j) = S(G_2^j)$$

Beweis: Wir haben $S(G_1) = S(G_2) \iff L(G_i) \subseteq E(h_1, h_2)$. Nach Konstruktion gilt $L(G_i^j) \subseteq L(G_i)$, sowie $E(h_1, h_2) \subseteq E(g_1, g_2)$. Deswegen folgt aus $S(G_1) = S(G_2)$, daß $S(G_1^j) = S(G_2^j)$.

Falls $S(G_1) \neq S(G_2)$, dann existiert ein kleinstes k mit $h_1^k(w_1) \neq h_2^k(w_2)$.

Im Falle $k \le n$ ergibt sich daraus, daß in den Systemen G_i^k bereits die Axiome nicht übereinstimmen, also $S(G_1^k) \neq S(G_2^k)$.

Andernfalls ist $k > n$, und wir bestimmen die Zahl $l \equiv k \bmod (n+1)$ mit $0 \le l \le n$. Dann folgt aber $S(G_1^l) \neq S(G_2^l)$. □

Lemma 8 *Für die D0L–Systeme $G_i^{j\prime} = (\Sigma, f p_i, f(h_i^j(w_i)))$ (mit den obigen Bezeichnungen) gilt*

$$S(G_1^j) = S(G_2^j) \iff L(G_1^{j\prime}) \subseteq E(p_1, p_2)$$

Beweis: Diagramme jagen. □

Folgerung 5 *Jede der Fragen $S(G_1^j) = S(G_2^j)$ ist entscheidbar.*

Beweis: Beide p_i sind elementar, deswegen ist $L(G_1^{j\prime}) \subseteq E(p_1, p_2)$ nach Satz 26 entscheidbar. □

Alles zusammen:

Satz 29 *Das Problem:*

- Eingabe*: zwei D0L–Systeme $G_i = (\Sigma_i, h_i, w_i)$*
- Frage*: Ist $S(G_1) = S(G_2)$*

ist entscheidbar. □

7.5 D0L–Sprach–Äquivalenz

Die Begriffe Folgen– und Sprach–Äquivalenz stimmen nicht überein.

Beispiel: Für die D0L–Systeme

$$\begin{aligned} G_1 &= (\{a,b\}, a \mapsto bb, b \mapsto a, b) \\ G_2 &= (\{a,b\}, a \mapsto b, b \mapsto aa, a) \end{aligned}$$

gilt $S(G_1) \neq S(G_2)$, aber $L(G_1) = L(G_2)$.

Trotzdem möchten wir das Entscheidungsverfahren für die Folgen–Äquivalenz bei der Sprach–Äquivalenz benutzen. Das gelingt auf naive Weise nur, wenn die Wörter in $S(G_i)$ immer länger werden, denn dann stimmen Sprach– und Folgen–Äquivalenz überein.

Die Wortlänge ist zu ungenau, wir vergleiche die Anzahlen einzelner Buchstaben:

Definition 27 *Für Wörter $v, w \in \Sigma^*$ schreiben wir $v \leq_P w$, wenn $\forall x \in \Sigma : |v|_x \leq |w|_x$.*

Beispiele: $aaba \leq_P bababa$, aber $aab \not\leq_P bab$ und $bab \not\leq_P aab$. Diese Relation ist offensichtlich eine Halbordnung.

Definition 28 *Eine Quasiordung (d. h. nicht notwendig antisymmetrisch) $\leq$ auf M heißt Wohl–Quasi–Ordung (WQO), wenn in jeder unendliche Folge $x_1, x_2, \ldots$ von Elementen aus M Indizes $i < j$ existieren mit $x_i \leq x_j$.*

Das bedeutet zweierlei: es gibt keine unendliche absteigende Folge, und es gibt keine unendliche Menge von paarweise unvergleichbaren Elementen. (Zu WQOs siehe auch meine Vorlesung Kombinatorik SS00.)

Satz 30 *Die Relation $\leq_P$ ist eine WQO.*

Beweis: Übung (Induktion nach Alphabetgröße).

Lemma 9 *Wenn eine D0L–Sprache $L(G)$ unendlich ist, dann gibt es für $S(G) = (w_0, w_1, \ldots)$ keine Indizes $i < j$, so daß $w_i \geq_P w_j$.*

Beweis: Übung.

Lemma 10 *Wenn eine D0L–Sprache $L(G)$ unendlich ist, und zwei Indizes $i < j$ mit $w_i \leq_P w_j$ besitzt, dann gilt $\forall n \geq 0 : w_{i+n} \leq_P w_{j+n}$.*

Beweis: Übung.

7.6 D0L–Sprach–Äquivalenz: Entscheidungsverfahren

Satz 31 *Das Problem:*

- Eingabe*: zwei D0L–Systeme* $G_i = (\Sigma_i, h_i, w_i)$
- Frage*: Ist* $L(G_1) = L(G_2)$

ist entscheidbar.

Beweis–Idee: Wir zerlegen $L(G_i)$ in zwei Anteile: einen endlichen (der den Anfangsmüll aufsammelt) sowie einen unendlichen, der eine disjunkte Summe von wachsenden D0L–Systemen ist, für die Sprach–Äquivalenz mit Folgen–Äquivalenz übereinstimmen.

Dann gilt $L(G_1) = L(G_2)$, falls die endlichen Anteile übereinstimmen, sowie eine Permutation existiert, die die Summanden aus den unendlichen Anteilen so einander zuordnet, daß sie folgen–äquivalent sind.

Beweis: der folgende Algorithmus entscheidet die Frage $L(G_1) = L(G_2)$:

repeat $m := 0$;
    $C := \{h_i^j(w_i) : 0 \le j < m\}$;
    for $i \in \{1, 2\}$ do
        $q_i := \min\{q : q > m \wedge \exists 1 \le p \le q_i - m : h_i^{q_i - p}(w_i) <_P h_i^{q_i}(w_i)\}$;
        $p_i := \min\{p : 1 \le p \le q_i - m, h_i^{q_i - p}(w_i) <_P h_i^{q_i}(w_i)\}$;
        $F_i := \{h_i^j(w_i) : m \le j < q_i\}$;
    if $F_1 \ne F_2 \vee p_1 \ne p_2$
    then return $L(G_1) \ne L(G_2)$;
    else for $i \in \{1, 2\}, j \in \{0, \ldots p_1 - 1\}$:
            $G_i^j := (\Sigma, h_i^{p_1}, h_i^{q_1 + j}(w_i))$;
        if $\exists$ Permutation $\pi : \{0, \ldots, p_1 - 1\} \to \{0, \ldots, p_1 - 1\}$
                mit $\forall 0 \le j < p_1 : S(G_1^j) = S(G_2^{\pi(j)})$
        then return $L(G_1) = L(G_2)$;
        else $m := q_1$;
end repeat;

Bemerkungen: die Zahlen q_i finden wir wegen Satz 30.

Die Minimalität von p_i bewirkt, daß $\forall m \le a < b \le q_i$ gilt: $h_i^a(w_i) \not<_P h_i^b(w_i)$.

Die Sprachen F_i sind endlich, das C (auch endlich) ist jeweils die Vereinigung der bisherigen F_i.

Die Systeme G_i^j haben die Eigenschaft, daß die Wortlängen (sogar die Parikh–Vektoren) streng monoton wachsen. Deswegen gilt $L(G_1^j) = L(G_2^{\pi(j)})$ genau dann, wenn $S(G_1^j) = L(G_2^{\pi(j)})$, aber diese Frage (Folgen–Äquivalenz) haben wir schon als entscheidbar nachgewiesen.

7.7 D0L–Sprach–Äquivalenz: Korrektheit

Die Ausgabe $L(G_1) \neq L(G_2)$ ist korrekt:

Nehmen wir an, daß $F_1 \neq F_2$, aber trotzdem $L(G_1) = L(G_2)$, und es gibt ein $w \in F_1 \setminus F_2$. Dann ist jedenfalls $w \in L(G_2)$, also existiert ein $i \geq q_2$ mit $w = h_2^i(w_2)$. Dann gibt es nach Lemma 10 ein $j \geq m$ mit $v = h_2^j(w_2) <_P w$.

Diese v hat auch einen Index in G_1, nämlich $v = h_1^k(w_1)$.

- Falls $k < m$, dann muß $v \in C$ liegen, also in G_2 ebenfalls einen Index $< m$ besitzen. Dann kommt v zweimal in $S(G_2)$ vor, das ist ein Widerspruch zu Lemma 9.
- Falls $m \leq k < q_1$, dann haben wir $v <_P w$ für zwei Wörter aus F_1, im Widerspruch zur Wahl von p_1.
- Falls $q_1 \leq k$, dann gibt es ein Wort $v' \in F_1$ mit $v' <_P v$, und benutzen damit das vorige Argument.

Nehmen wir an, daß $F_1 = F_2$ und o.B.d.A $p_1 < p_2$, aber trotzdem $L(G_1) = L(G_2)$. Dann wird $\{h_1^j(w_1) : j \geq q_1\}$ überdeckt von p_1 Stück Parikh–monotonen Teilfolgen. Jedes Folgenglied besitzt auch einen Index in $\{h_2^j(w_2) : j \geq q_1\}$. Diese Indizes können wir modp_2 reduzieren auf den Bereich $q_1 - p_2 \ldots q_1 - 1$. Wenn jede Teilfolge dabei nur ein Urbild besitzt, entstehen Lücken (d. h. Wörter in $L(G_2) \setminus L(G_1)$), und wenn eine Teilfolge mehr als ein Urbild hat, dann erhalten wir einen Wiederspruch zur Minimalität von p_2.

Die Ausgabe $L(G_1) = L(G_2)$ ist korrekt, denn $L(G_i) = C \cup F_i \cup \bigcup_{0 \leq j < p_i} L(G_i^j)$.

7.8 D0L–Sprach–Äquivalenz: Termination

1. Falls $L(G_1) \neq L(G_2)$, dann hält der Algorithmus mit der richtigen Antwort:

   Dann gibt es ja o.B.d.A. ein $w \in L(G_1) \setminus L(G_2)$. Weil m in jedem Schritt um wenigstens eins erhöht wird, ist dieses w irgendwann in $F_1 \setminus F_2$.

2. Falls $L(G_1) = L(G_2)$, dann hält der Algorithmus mit der richtigen Antwort.

   Beweis–Idee: Die Werte der p_i nehmen während der Programm–Ausführung monoton ab. Wir müssen nur zeigen, daß sie nicht beliebig lange gleich bleiben.

   Wir haben schon verifiziert, daß $p_1 = p_2$.

   Wenn sich (p_1, p_2) nicht geändert haben, dann muß $p_1 = p_2 = q_1 - m$ sein.

   Auch die Permutation π ist jedesmal dieselbe.

   Damit bilden wir die jeweiligen F_i bijektiv aufeinander ab.

   Tun wir das $(n + 1)$ mal, wobei n die Alphabetgröße ist, haben wir jeweils übereinstimmende Anfangsstücke der Länge $(n + 1)$ von D0L–Folgen. Daraus folgt leicht durch Matrizenmultiplikation (siehe Abschnitt Wachstum), daß dann die Folgen Parikh–äquivalent sind.

   Falls nun doch eine der Antowrten $S(G_1^j) \neq S(G_2^j)$ lautet, gibt es also zwei Wörter $w \in S(G_1^j), v \in S(G_2^j)$ die Parikh–äquivalent, aber verschieden sind. Weil jedoch $L(G_1) = L(G_2)$, muß v an anderer Stelle in G_1 vorkommen. Damit enthält aber $S(G_1)$ zwei verschiedene Wörter w, v mit gleichem Parikh–Vektor, im Widerspruch zu Lemma 9.

7.9 Verwandte und offene Fragen

Übungsausfgaben :-) g, h sind Morphismen und x, y sind Wörter. Untersuche die Entscheidbarkeit dieser Problemen:

1. Gegeben sind h, x, y.
   (a) Existiert ein n mit $h^n(x) = h^n(y)$?
   (b) Existieren m, n mit $h^m(x) = h^n(y)$?
2. Gegeben sind g, h, x, y.
   (a) Gilt $\forall n : g^n(x) = h^n(y)$?
   (b) Existiert ein m, so daß $\forall n \geq m : g^n(x) = h^n(y)$?
   (c) Existiert ein n mit $g^n(x) = h^n(y)$?
   (d) Existieren m, n mit $g^m(x) = h^n(y)$?

Welches davon ist das D0L–Äquivalenz–Problem? Die ersten beiden Problemen sind einfacher als dieses, die Entscheidbarkeit der letzten beiden ist offen (Stand 1980).

Kapitel 8

Lösbare Fälle des PCP

Regularität der Gleichheitsmenge reicht eben nicht, sondern man muß eine Repräsentation davon auch effektiv konstruieren können.

8.1 Periodische PCP

Definition 29 *Ein Morphismus* $h : \Sigma^* \to \Gamma^*$ *ist* periodisch, *wenn ein Wort* u *existiert, so daß* $h(\Sigma) \subset u^*$.

Beispiele: $h : 0 \to ab, 1 \to ab, 2 \to abab$ ist periodisch.

Alle Morphismen $h : \Sigma^* \to \Gamma^*$ mit $|\Gamma| = 1$ sind periodisch.

Satz 32 *Die Frage* $\{\epsilon\} = E(g, h)$ *ist entscheidbar, wenn (wenigstens) einer der Morphismen* g, h *periodisch ist.*

Beweis: Wir nehmen g als periodisch, also $g(\Sigma) \subseteq u^*$, und h beliebig.

Für jede Lösung w des PCP gilt $h(w) = g(w) \in u^*$. Anders gesagt, $w \in H = h^{-1}(u^*)$.

Diese Sprache H ist regulär (und effektiv konstruierbar), weil reguläre Sprachen abgeschlossen unter inversen Morphismen sind.

Welche Wörter w aus H sind nun tatsächlich Lösungen? Anstatt $g(w) = h(w)$ zu testen, reicht es wegen der Periodizität von g aus, die Wortlängen $|g(w)| = |h(w)|$ zu vergleichen.

Dazu erklären wir einen Morphismus $d : \Sigma^* \to \mathbb{Z}$ (augefaßt als Monoid unter der Addition) durch $d(x) = |g(x)| - |h(x)|$.

Die Sprache $D = d^{-1}(0)$ ist kontextfrei. (Beweis: Übung. Wie lautet eine CF–Grammatik für die Sprache $\{w : |w|_a + 2|w|_b - 3|w|_c = 0\}$?)

Weil kontextfreie Sprachen abgeschlossen sind gegen Schnitt mit regulären Sprachen, ist auch $D \cap H$ kontextfrei (und effektiv konstruierbar).

Schließlich ist zu entscheiden, ob $\emptyset = D \cap H$. Die Leere einer kontextfreien Sprache ist jedoch ebenfalls effektiv feststellbar. $\square$

8.2 Markierte PCP

Definition 30 *Ein Morphismus* $h : \Sigma^* \to \Gamma^*$ *ist* markiert*, falls* $\forall x, y \in \Sigma$ *gilt:* $|\inf(h(x), h(y)| > 0 \to x = y$

D. h. die Bilder verschiedener Buchstaben müssen bereits mit verschiedenen Anfangsbuchstaben beginnen.

Das bringt natürlich bei der Lösungssuche enorme Vorteile: die Fortsetzung ist jeweils eindeutig bestimmt. Eine Lösung finden wir so garantiert, aber wir wissen trotzdem nicht, wann wir eine erfolglose Suche abbrechen dürfen.

Dieser Abschnitt folgt [HH01], Einzelheiten kann man dort nachlesen.

Beweis–Ideen:

1. Wir bestimmen zunächst Lösungen (x, y) der Aufgaben:

   - Eingabe: Morphismen $g, h : \Sigma^* \to \Gamma^*$, Buchstabe $a \in \Gamma$
   - Gesucht: $x, y \in \Sigma^*$ mit $g(x) = h(y) \neq \epsilon$ und $g(x)$ beginnt mit a.

   Das tun wir dadurch, daß wir die jeweils kürzere Komponente verlängern. Weil die Morphismen markiert sind, gelingt das eindeutig. Da wir immer auf der kürzeren Seite anbauen, gibt es nur endliche viele mögliche Differenzen. Wir brechen ab, wenn

   (a) Eine Verlängerung nicht möglich ist

   (b) Eine Differenz zum zweiten Mal auftritt

   (c) Die Differenz ϵ ist.

   Es ist klar, daß genau einer dieser Fälle eintreten muß.

2. Nur im letzten Fall können wir das Paar (x, y) als Bestandteil einer PCP–Lösung verwenden. Wir nennen (x, y) dann einen Block für a. Es gibt endlich viele Blöcke $(x_1, y_1), \ldots$. Jede PCP–Lösung ist eine Folge von solchen Blöcken.

3. Damit haben wir das PCP (g, h) übersetzt in ein neues PCP, indem wir die eben konstruierten Blöcke als Buchstaben auffassen. Beide PCP sind äquivalent. Wir wiederholen die Konstruktion, bis wie die Lösung (oder die Unlösbarkeit) "sehen."

   Wir müssen nachweisen, daß das neue PCP in irgeneinem Sinne einfacher als das alte ist. Daraus folgt dann, daß wir die obige Schleife irgendwann verlassen. Die geeignete Bewertungsfunktion ist zunächst die Alphabetgröße (es gibt jedesmal höchstens soviele Blöcke wie vorher Buchstaben) und (bei Übereinstimmung) die Anzahl der unterschiedlichen Suffixe (siehe Literatur).

□

8.3 Binäre PCPs

Satz 33 *Für Morphismen $g, h : \Sigma \to \Gamma$ mit $|\Sigma| = 2$ ist die Frage $\{\epsilon\} = E(g, h)$ entscheidbar.*

Beweis–Idee: reduziere auf GPCP(2) für markierte Morphismen.

Wir setzen $\Sigma = \{a, b\}$. Falls die Morphismen g, h nicht markiert sind, dann haben $g(a)$ und $h(a)$ einen nichtleeren gemeinsamen Präfix p_g:

$$g(a) = p_g x_g, g(b) = p_g y_g.$$

Wir betrachten dann den Morphismus

$$g' : a \mapsto x_g p_g, b \mapsto y_g p_g.$$

Jetzt ist g' markiert!

(Falls das nicht geht, weil irgendwelche Teile leer sind, dann ist g selbt periodisch, und damit das PCP lösbar.)

Das gleiche tun wir für h und erhalten dem markierten Morphismus h'.

Damit gilt $g(w) = h(w) \iff g'(w) = h'(w)z$, falls $|p_g| \leq |p_h|$ und $p_g z = p_h$.

Nun haben wir eine markierte GPCP–Instanz. Dafür gibt es ein Lösungsverfahren (siehe [HK97]) nach der gleichen Idee wie für markiertes PCP. $\square$

Wir sehen außerdem, daß die obige Konstruktion (Abschneiden des gemeinsamen Präfixes) für PCP(3) nicht funktionert.

Kapitel 9

Wachstum

Wenn wir statt des Inhalts der Wörter, die von einem (D0)L–System erzeugt werden, nur ihre Länge betrachten, erhalten wir Zahlenfolgen, die das Wachstum beschreiben. Welche Zahlenfolgen können als Wachstumsfolgen auftreten?

Es ist sicher notwendig, nicht nur die Gesamtzahl der Buchstaben, sondern die Anzahl der Vorkommen jedes einzelnen Buchstaben zu betrachten. Damit wird ein Wort $w \in \Sigma$ beschrieben durch einen Vektor mit $|\Sigma|$ nichtnegativen Einträgen. Dieser Vektor heißt Parikh–Vektor.

Wir befinden uns damit auf einmal im Bereich der linearen Algebra, und möchten möglichst viele ihrer Resultate nachnutzen. Andererseits können wir einige Aspekte aus der Berechenbarkeit in die Matrizenrechnung tragen.

Warum funktioniert das hier, und sonst selten? Sobald wir nicht mehr mit Wörtern rechnen, sondern mit Anzahlen (von Buchstaben), ist die Monoid–Operation (aus Verkettung wird Addition) kommutativ.

(Dieses Kapitel folgt Abschnitt I.3 aus [RS80], Abschnitt 5 aus [KRS97] sowie 4.4 und 4.5 aus [HK97].)

9.1 Wachstumsfolgen

Definition 31 *Der* Parikh–Vektor *des Wortes $w \in \Sigma^*$ für ein Alphabet $\Sigma = \{x_1, \ldots, x_n\}$ ist die Folge $(|w|_{x_1}, \ldots, |w|_{x_n})$.*

Die Einträge sind nichtnegativ.

Das Wort $w = abcc$ hat den Parikh–Vektor $P(w) = (1, 1, 2)$.

Das D0L–System mit dem Morphismus

$$q : a \mapsto abc^2, b \mapsto bc^2, c \mapsto c$$

erzeugt $q(w) = abccbcccc$ mit $P(q(w)) = (1, 2, 6)$. Wir $P(q(w))$ auch direkt aus $P(w)$ bestimmen: durch Matrix–Multiplikation:

$$(1, 1, 2) \cdot \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} = (1, 2, 6)$$

im allgemeinen gilt $P(q^i(w)) = P(w) \cdot M^i$, Beispiel:

$$(1, 1, 2) \cdot \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}^{10} = (1, 1, 2) \cdot \begin{pmatrix} 1 & 10 & 110 \\ 0 & 1 & 20 \\ 0 & 0 & 1 \end{pmatrix} = (1, 11, 132)$$

Um die Wortlänge festzustellen, multipliziert man den Parikh–Vektor mit $(1, \ldots, 1)^T$:

$$|q^i(w)| = P(w) \cdot M^i \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Das charakteristische Polynom p_M einer Matrix M ist $\det(M - xI)$. Im Beispiel $p_M = (x - 1)^3$. Nach dem Cayley–Hamilton–Satz erfüllt jede Matrix ihre eigene charakteristische Gleichung: $p_M(M) = 0$. Der höchste Koeffizient ist 1 (die anderen sind ganzzahlig), also gibt es Zahlen c_i mit

$$M^n = c_1 M^{n-1} + \ldots + c_n M^0$$

(Im Beispiel $M^3 = 3M^2 - 3M + 1$.)

Wenn wir von links mit $P(q^i(w_0))$ und rechts mit $(1, \ldots, 1)^T$ multiplizieren, erhalten wir für die Wachstumsfunktion $f(i) = |w_i| = |q^i(w_0)|$ der D0L–Folge

$$\forall i \geq 0 \quad : \quad f(i + n) = c_1 f(i + n - 1) + \ldots + c_n f(i)$$

Das heißt, sie erfüllt eine lineare Rekurrenz.

$$f(i + 3) = 3f(i + 2) - 3f(i + 1) + f(0)$$

Durch draufgucken oder Übungsaufgabenlösen wissen wir schon $f(k) = k^2$. Dann können wir die Identität $(i + 3)^2 = 3(i + 2)^2 - 3(i + 1)^2 + i^2$ leicht nachrechnen.

(Über solche Identitäten im Allgemeinen siehe das ganz erstaunliche Buch [PWZ96], sowie Veranstaltungen zur Computeralgebra von Herrn Dr. Gräbe.)

9.2 $\mathbb{N}$–rationale Funktionen, D0L–Wachstum

Definition 32 *Eine Funktion $f : \mathbb{N} \to \mathbb{N}$ heißt $\mathbb{N}$–rational, wenn es eine Zahl $d \geq 1$ sowie einen Zeilenvektor r, eine $d \times d$–Matrix M und einen Spaltenvektor c, alles mit Einträgen aus $\mathbb{N}$ gibt, so daß $f(n) = r \cdot M^n \cdot c$*

Satz 34 *Eine Funktion f ist genau dann einen D0L–Wachstums–Funktionen, wenn sie eine Darstellung als $\mathbb{N}$–rationalen Funktionen besitzt, für die $c = (1, \ldots, 1)^T$ ist.*

Die Lösungen der linearen Rekurrenzen kann man immer auf bestimmte Weise darstellen:

Beispiel: Für das System $a \mapsto ab, b \mapsto a$ ist die Matrix $M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ und hat Eigenwert $r_1 = (1 + \sqrt{5})/2 =: \phi$ mit Eigenvektor $v_1 = (\phi, 1)$ und Eigenwert $r_2 = 1 - \phi$ mit Eigenvektor $v_2 = (1 - \phi, 1)$. Dann ist $(1, 0) = (1/\sqrt{5})v_1 - (1/\sqrt{5})v_2$ und $(1, 0) \cdot M^k = (1/\sqrt{5})\phi^k v_1 - (1/\sqrt{5})(1 - \phi)^k v_2$.

Die Wachstumsfunktion des D0L–Systems ist damit

$$f(k) = (1/\sqrt{5})(\phi^k(\phi + 1) - (1 - \phi)^k(2 - \phi)$$

Mit $1 + \phi = \phi^2$ folgt $f(k) = (1/\sqrt{5})(\phi^{k+2} - (1 - \phi)^{k+2})$. Beachte, daß dabei immer ganze Zahlen (Fibonacci–Zahlen) entstehen.

Dieses Beispiel ist typisch, im Allgemeinen gilt:

Satz 35 *Jede D0L–Wachstums–Funktion f besitzt eine Darstellung*

$$\forall n \geq n_0 \quad : \quad f(n) = \sum r_i^n (a_{i0} + a_{i1}n + \ldots + a_{it_i} n^{t_i})$$

Beweis: die r_i sind die Eigenwerte (mit Vielfachheit t_i) der Wachstumsmatrix. Die Koeffizienten $a_i j$ ergeben sich durch Einsetzen einiger Startwerte.

Übung: rechne ein Beispiel mit einem mehrfachen Eigenwert.

Folgerung: D0L–Wachstums–Funktionen wachsen exponentiell oder polynomiell.

Folgerung: Funktionen $\log n$ oder $2^{\sqrt{n}}$ sind nicht durch D0L–Systeme realisierbar.

Satz 36 *Das Problem:*

Eingabe*: zwei D0L–Systeme G_1, G_2*
Frage*: haben beide die gleiche Wachstumsfunktion?*

ist entscheidbar.

Beweis (kompliziert) man berechnet und vergleicht die Darstellungen nach obigem Satz.

Beweis (einfach): es genügt, die Wortlängen der ersten $|\Sigma_1| + |\Sigma_2|$ Wörter zu vergleichen.

Beachte, daß das viel viel einfacher ist als das Entscheidungsverfahren zur D0L–Folgen–Äquivalenz.

Übung: Begründe diese Schranke (d. h. wieviele Anfangswörter benötigt man, um alle Koeffizienten der Darstellung zu finden) und zeige durch ein Beispiel, daß sie scharf ist.

9.3 HD0L–Wachstum

Satz 37 *Eine Funktion f ist Wachstumsfunktion eines HD0L–Systems $\iff$ f ist eine $\mathbb{N}$–rationale Funktion.*

Wir multiplizieren dann also nicht mit $c = (1, \ldots, 1)^T$, sondern mit einem beliebigen Vektor (mit nichtnegativen Einträgen).

Insbesondere können einige Komponenten von c gleich 0 sein.

Übungsaufgabe: untersuche die Funktion

$$f(n) = (1,0,1,0) \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot (1,0,0,1)^T$$

finde ein HD0L–System, dessen Wachstumsfunktion sie ist.

9.4 $\mathbb{Z}$–rationale Funktionen

Definition 33 *Eine Funktion $f : \mathbb{Z} \to \mathbb{Z}$ heißt $\mathbb{Z}$–rational, wenn es eine Zahl $d \geq 1$ sowie einen Zeilenvektor r, eine $d \times d$–Matrix M und einen Spaltenvektor c, alles mit Einträgen aus $\mathbb{Z}$ gibt, so daß*

$$f(n) = r \cdot M^n \cdot c$$

Satz 38 *Jede $\mathbb{Z}$–rationale Funktion besitzt eine Darstellung als Differenz von zwei $\mathbb{N}$–rationalen Funktionen, die Wachstumsfunktionen von PD0L–Systemen sind.*

Übung: Beweise (wenigsten bis zum Komma).

9.5 D0L–Omega–Äquivalenz

Satz 39 *Das Problem*

Eingabe: *zwei D0L–Systeme* $G_i = (\Sigma, h_i, w_i)$ *mit* $h_i^k(w_i) \sqsubset h_i^{k+1}(w_i)$
Frage: *gilt* $h_1^\omega(w_1) = h_2^\omega(w_2)$*?*

ist entscheidbar.

Beweis–Idee [CIH84]:

Wir wollen das bekannte Argument benutzen, daß eine gewisse Sprache regulär ist, und dann zwei Semi–Algorithmen kombinieren: einerseits reguläre Grammatiken aufzählen und testen, ob sie die passenden sind (das ergibt dann die Antwort "ja"), andererseits immer längere Präfixe der Omega–Wörter generieren und testen, ob sie sich irgendwo unterscheiden (das ergibt dann die Antwort "nein").

Bei D0L–Äquivalenz hat das ja funktioniert, die Sprache war die Gleichheitsmenge von elementaren Morphismen, die wir aus den gegebenen h_i konstruiert hatten, und diese hatte beschränkte Balance.

Bei Omega–Äquivalenz nützen Gleichheitsmengen nichts, wir betrachten stattdessen Mengen

$$C(h_1, h_2) = \{w \in \Sigma^* : h_1(w) \sqsubseteq h_2(w) \vee h_2(w) \sqsubseteq h_1(w)\}$$

Allerdings kann für omega–äquivalente h_1, h_2 die Balance $|h_1(w)| - |h_2(w)|$ für $w \in C(h_1, h_2)$ unbeschränkt wachsen (weil die Limes–Wörter verschieden schnell entstehen dürfen).

Man betrachtet stattdessen die Systeme $G_{12} = (\Sigma, h_1 \cdot h_2, w)$ und $G_{21} = (\Sigma, h_2 \cdot h_1, w)$ und zeigt $G_1 =_\omega G_2 \iff G_{12} =_\omega G_{21}$ sowie daß die Balance in $C(h_1 \cdot h_2, h_2 \cdot h_1)$ beschränkt ist.

Entscheidendes Hilfsmittel dabei ist, daß die Parikh–Matrizen für G_{12} und G_{21} die gleichen Eigenwerte haben (Übung: warum?). Falls beide Systeme das gleiche Omega–Wort w erzeugen, dann gilt für die Präfixe w_n der Länge n von w:

$$\lim_{n \to \infty} \frac{|h_1 h_2(w_n)| - |h_2 h_1(w_n)|}{n} = 0,$$

d. h. die Balance steigt jedenfalls langsamer als die Länge.

Man rechnet andererseits nach, daß beide Systeme exponentiell wachsen, und leitet daraus ab, daß die Balance sogar beschränkt sein muß. (Wäre sie es nicht, dann wäre der Limes nicht 0.)

Damit ist $C(h_1 \cdot h_2, h_2 \cdot h_1)$ regulär, und der oben genannte Semi–Algorithmus funktioniert. □

9.6 Offene Fragen zu Wachstums–Funktionen

Ich gebe hier eine Liste offener Fragen aus [RS80]. Vorsicht — das ist von 1981, also bevor man sehr viel Zeit in die Lösung investiert, sollte man vielleicht doch prüfen, ob die Lösungen inzwischen gefunden wurden.

Für eine Matrix M mit ganzzahligen Einträgen bezeichnet $f(k)$ das Element oben rechts in M^k.

1. Entscheide, ob $\forall k : f(k) \geq 0$. — Äquivalent dazu
   (a) Für zwei D0L–Wachstums–Folgen $r(n), s(n)$: Entscheide, ob $\forall n : r(n) \leq s(n)$.
   (b) Das gleiche für PD0L–Wachstums–Folgen.
   (c) Für eine D0L–Wachstums–Folge $r(n)$: Entscheide, ob $\forall n : r(n) \leq r(n+1)$.
2. Entscheide, ob $\exists k : f(k) = 0$. — Äquivalent dazu
   (a) Für zwei D0L–Wachstums–Folgen $r(n), s(n)$: Entscheide, ob $\exists n : r(n) = s(n)$.
   (b) Das gleiche für PD0L–Wachstums–Folgen.
   (c) Für eine D0L–Wachstums–Folge $r(n)$: Entscheide, ob $\exists n : r(n) = r(n+1)$.

Aus Entscheidbarkeit des zweiten Problems folgt auch die Entscheidbarkeit von

1. Stimmen die Wort–Folgen von zwei HD0L–Systemen überein?
2. Unterscheiden sich zwei HD0L–Wortfolgen nur in endlich vielen Wörtern?
3. Haben zwei HD0L–Wortfolgen einen leeren oder endlichen Durchschnitt?

Beachte allerdings den kommenden Abschnitt über Ehrenfeuchts Vermutung.

9.7 Unentscheidbare lineare Algebra?

Woher kommt das eigentlich, daß einige Eigenschaften ganzzahliger Matrizen schwer (unentscheidbar) sind? Wir können Rechnungen (von Turingmaschinen) kodieren. Eine Maschinen–Konfiguration ist eine Matrix C, ein Rechnungsschritt (nach einem festen Programm) ist Multiplikation (mit einer festen Matrix P), die Folge der Konfigurationen also $C \cdot P^k$. Die Haltebedingung ist dann $(C \cdot P^k)_{13} = 0$ “oben rechts erscheint 0” oder ähnlich. Genauer:

Das Matrix–Monoid $\mathbb{Z}^{2\times 2}$ enthält ein freies Untermonoid, das erzeugt wird von

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Übung: beweise, daß das wirklich frei ist. Zeige am Beispiel, durch welches Generatorwort aus $\{A, B\}^*$ die Matrix $\begin{pmatrix} 31 & 11 \\ 14 & 5 \end{pmatrix}$ erzeugt wird.

Wir können damit das Wort–Monoid Σ^* in $\mathsf{SL}(2, \mathbb{Z})$ einbetten (für $|\Sigma| = 2$, aber das ist keine wesentliche Einschränkung), also Wörter eindeutig durch bestimmte Matrizen kodieren, wobei die Wortverkettung der Matrixmultiplikation entspricht.

Wir können sogar das Produkt $\Sigma^* \times \Sigma^* \to \mathbb{Z}^{3\times 3}$ einbetten (d. h. Paare von Wörtern als 3×3–Matrizen) durch

$$p(u, v) = \begin{pmatrix} 1 & \underline{v} & \underline{u} - \underline{v} \\ 0 & 2^{|v|} & 2^{|u|} - 2^{|v|} \\ 0 & 0 & 2^{|u|} \end{pmatrix}$$

Dabei sind $u, v \in \{0, 1\}^*$ und $\underline{u}$ bedeutet den Wert von u zur Basis 2.

Übung: nachrechnen, daß das ein Monoid–Morphismus ist, d. h.

$$\forall a, b, c, d \in \{0, 1\}^* : p(a, b) \cdot p(c, d) = p(ac, bd)$$

und daß er injektiv ist.

Satz 40 *Das folgende Problem ist nicht entscheidbar:*

Eingabe: *eine endlich generiertes Matrix–Monoid $\mathcal{M}$ mit Erzeugenden $M_i \in \mathbb{Z}^{3\times 3}$*
Frage: *existiert ein $M \in \mathcal{M}$ mit $M_{13} = 0$ (das Element oben rechts ist 0)?*

Beweis: Mit der obigen Kodierung können wir direkt PCPs verpacken. Oben rechts steht dann 0, wenn die Binärcodes des oberen und unteren Wortes übereinstimmen, d. h. das PCP eine Lösung besitzt. □

Kapitel 10

Wort– und Baumgleichungen

10.1 Hilberts zehntes Problem

(Dieser Abschnitt siehe [Sch95])

Ist die folgende Frage $D(\mathbb{Z})$ entscheidbar (D = Diophant)?

Gegeben ein Polynom p in n Variablen mit ganzen Koeffizienten,
gefragt: gibt es Nullstellen, d. h. Tupel $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ mit $p(x_1, \ldots, x_n) = 0$.

Wir können statt einer einzigen Gleichung sogar Gleichungssysteme gestatten, d. h. mit "und" verknüpfte Bedingungen. (Beweis: $f = 0 \wedge g = 0 \iff f^2 + g^2 = 0$)

Weiterhin ist auch "oder" als Verknüpfung simulierbar. (Beweis: $f = 0 \vee g = 0 \iff f \cdot g = 0$)

Tatsächlich geht es also um prädikatenlogische Formeln mit Funktionssymbolen $+, -, *$, Booleschen Operationen "und" und "oder", sowie Existenzquantoren (über ganze Zahlen).

(Wenn wir auch noch die Negation (und damit den All–Quantor) zulassen, dann folgt aus Gödels Unvollständigkeitssatz sofort die Unentscheidbarkeit.)

Das Problem $D(\mathbb{Z})$ ist äquivalent zu dem Problem $D(\mathbb{N})$, das nur nach Lösungen in $\mathbb{N}^n$ fragt.

Dazu zeigt man $D(\mathbb{Z}) \leq D(\mathbb{N})$ (simpel: man stellt jede Variable für eine ganze Zahl als Differen zweier natürlichen Zahlen dar) und $D(\mathbb{N}) \leq D(\mathbb{Z})$ (mit Trick: man stellt jede Variable für einen natürlich Zahl als Summe von vier Quadraten (ganzer Zahlen) dar).

10.2 Makanins Algorithmus

10.2.1 Definitionen

(Dieser Abschnitt siehe [Die97])

Wir fixieren ein Alphabet Σ und eine (dazu disjunkte) Variablenmenge V. Eine Wortgleichung ist ein Paar $(l,r) \in (\Sigma \cup V)^{*2}$. Eine Lösung der Gleichung ist ein Morphismus $h : \Sigma \cup V \to \Sigma^*$, der auf Σ die Identität ist, mit $h(l) = h(r)$.

Beispiel: $\Sigma = \{a,b\}, V = \{x,y,z,u\}$. Finde eine Lösung der Gleichung

$$xauzau = yzbxaaby$$

Ebenso können wir Wort–Gleichungs–Systeme betrachten.

Das Problem W ist:

Eingabe: eine endliches System S von Wortgleichungen,
Frage: besitzt S eine Lösung.

10.2.2 Geschichte

Markov und seine Schüler untersuchten (ab ca. 1950) solche Gleichungen, weil sie hofften, damit die Unentscheidbarkeit des zehnten Hilbertschen Problems zu zeigen.

Satz 41 $W \leq D(\mathbb{N})$

Beweis: Wir nehmen an, daß $\Sigma = \{a,b\}$ (das ist keine wesentliche Einschränkung). Wir ersetzen in jeder Gleichung des gegebene Systems die Buchstaben a, b sowie Variablen x_i durch

$$a = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, x_i = \begin{pmatrix} y_{i1} & y_{i2} \\ y_{i3} & y_{i4} \end{pmatrix}$$

Dann multiplizieren wir die Matrizen für jede Wortgleichung (l_i, r_i) aus und erhalten eine Gleichung (L_i, R_i) zwischen Matrizen. Die Matrix–Elemente sind Polynome mit ganzen Koeffizienten, und wir suchen nach einer Lösung in natürlichen Zahlen, die außerdem noch die Bedingungen $y_{i1}y_{i4} - y_{i2}y_{i3} = 1$ erfüllt. □

Markovs Plan erfüllte sich nicht, brachte jedoch trotzdem interessante Resultate:

1970 zeigte Y. Matijasevic (nach Vorarbeiten von J. Robinson, H. Putnam und M. Davis), daß das zehnte Hilbertsche Problem unentscheidbar ist. Der Beweis verwendet eine trickreiche Kodierung von Rechnungen von Registermaschinen als Gleichungssysteme.

1977 bewies Makanin, daß die Lösbarkeit von Wortgleichungen entscheidbar ist.

10.2.3 Quadratische Gleichungen

Ein Wort–Gleichungs–System heißt quadratisch, wenn jede Variable höchstens zweimal vorkommt. Beispiel: $abxcy = ycxba$

Es gibt ein einfach einzusehendes Lösungsverfahren für quadratische Gleichungen.

Zunächst überlegt man sich, daß es genügt, Lösungen h zu finden, für die $\forall v \in V : h(v) \neq \epsilon$. Beweis: Man kann mit solch einem Verfahren dann auch *alle* Lösungen konstruieren. Die mit $h(x_1) = \epsilon$ findet man, indem man im Gleichungssystem überalle x_1 löscht, und auf das (so verkleinerte) System den Algorithmus rekursiv anwendet.

Wir betrachten jetzt irgendeine Gleichung $l = r$ des gegebenen Systems. Sie hat die Form $x \cdots = a \cdots$ oder $x \cdots = y \cdots$. (Warum?) Dann gilt $x = a, x = az, x = y$ oder $x = yz$ für eine neue Variable z. Wir ersetzen nun überall x durch az oder yz, kürzen, und erhalten wieder ein quadratisches System, das in einem geeigneten Sinne "kleiner" ist als das ursprüngliche.

10.2.4 Existentielle Theorie der Wortverkettung

Mit einem Entscheidungsverfahren für die Lösbarkeit von Wortgleichungssystemen kann man beweisen:

Satz 42 *Es ist entscheidbar, ob eine prädikatenlogische Formel mit Konstanten, Variablen für Wörter, dem Funktionssymbol $\cdot$ (Verkettung), allen Booleschen Operatoren, und Existenz–Quantoren allgemeingültig ist.*

Zum Beweis gibt man an, wie man die Booleschen Operatoren kodiert:

Satz 43 *Das Gleichungssystem $L_1 = R_1, \ldots, L_n = R_n$ hat die selbe Lösungsmenge wie die Gleichung*

$$L_1 a \cdots L_n a L_1 b \cdots L_n b = R_1 a \cdots R_n a R_1 b \cdots R_n b$$

wobei $a \neq b$ verschiedene Konstanten sind.

Beweis: Übung. □

Satz 44 *Die Formel $L \neq R$ ist äquivalent zu der Formel*

$$\exists x, y, z : \bigvee_{a \in \Sigma} (L = Rax \vee R = Lax) \vee \bigvee_{a,b \in \Sigma, a \neq b} (L = xay \wedge R = xbz)$$

Beweis: Übung.

10.2.5 Der Exponent einer Lösung

Ein wesentlicher Schritt bei Makanins Beweis ist es, mögliche Wiederholungen in kürzesten Lösungen zu beschränken.

Definition 34 *Der Exponent eines Wortes $w \in \Sigma^*$ ist* $\exp(w) = \max\{e \in \mathbb{N} : \exists x, y, z \in \Sigma^*, p \neq \epsilon, w = xy^e z\}$

10.3 Ehrenfeuchts "Vermutung"

(Dieser Abschnitt siehe [HK97])

Satz 45 (Hilberts Basis–Satz) *Für jede Menge von Polynomen P_i über den Variablen $x_1, \ldots, x_t$ gibt es eine Zahl r, so daß für jedes P_i eine Darstellung $P_i = \sum_{k=1}^{r} Q_k P_k$ mit Polynomen Q_k existiert.*

Übung: Beispiele besorgen sowie Satz und Beweis in einem Algebra–Lehrbuch aufsuchen.

Satz 46 *Jedes (unendliche) Gleichungssystem $P_i = 0$ mit Polynomen $P_i \in \mathbb{Z}[x_1, \ldots, x_t]$ besitzt ein äquivalentes endliches Teilsystem.*

Beweis: Übung. □

Satz 47 *Jedes System von Wortgleichungen besitzt ein äquivalentes endliches Teilsystem.*

Beweis: benutze Hilberts Basis–Satz für die Kodierung in Satz 41. □

Definition 35 *Eine Menge $T \subseteq L$ heißt* Testmenge *für $L \subseteq \Sigma^*$, falls für alle Morphismen $h, g : \Sigma^* \to \Delta^*$ gilt*

$$L \subseteq E(h, g) \iff T \subseteq E(h, g)$$

Satz 48 (Ehrenfeuchts Vermutung) *Jede Sprache L besitzt eine endliche Testmenge.*

Übung: Beispiele besorgen.

Beweis: folgt aus Satz 47.

Satz 49 *Für endlichen Mengen T und L ist entscheidbar, ob T eine Testmenge für L ist*

Beweis: Die Morphismen (h, g), die auf L übereinstimmen, können wir durch ein Gleichungssystem G_L beschreiben. Mit Makanins Algorithmus (und vorschalteter Entfernung der Negation) entscheiden wir, ob $G_T \land \neg G_L$ Lösungen hat. □

In Spezialfällen kann man Testmengen effektiv konstruieren (im Allgemeinen natürlich nicht, da nicht gesagt wird, wie die unendliche Menge L repräsentiert ist).

Satz 50 *Jede reguläre Sprache L mit Minimalautomat A mit n Zuständen besitzt eine Testmenge mit Wörtern w. für die $|w| \leq 2n$.*

Beweis: falls $uv, uzv, uwv \in E(h, g)$, dann auch $uzwv \in E(h, g)$. Falls $L \not\subseteq E(h, g)$, also $\exists s \in L \setminus E(h, g)$, und A bei Eingabe w einen Zustand dreifach besucht,

$$q_0 \xrightarrow{u} q \xrightarrow{z} q \xrightarrow{w} q \xrightarrow{v} q_1$$

dann ist wenigstens eines der Wörter uv, uzv, uwv auch nicht in $E(h, g)$. Diese sind aber kürzer, und daraus folgt der Satz.

(Es gilt weiter: Für jede kontextfreie Sprache läßt sich effektiv eine Testmenge konstruieren.)

Satz 51 *Für jede D0L–Sprache läßt sich effektiv eine endliche Testmenge konstruieren.*

Beweis: Wir bezeichnen $L = L(\Sigma, f, w)$ und $L_k = \{f^i(w) : 0 \leq i \leq k\}$.

Nach Ehrenfeuchts Vermutung existiert ein t, so daß L_t eine Testmenge für L ist. Dann ist L_t auch eine Testmenge für L_{t+1}.

Die kleinste Zahl k, für die L_k Testmenge für L_{k+1} ist, kann man nach Satz 49 effektiv finden. (Durch Probieren.)

Wir müssen zeigen, daß dieses L_k eine Testmenge für L ist. (Übung.) $\square$

Satz 52 *Das D0L–(Folgen–Äquivalenz–)Problem, das morpische D0L–Problem sowie das HD0L–Problem sind entscheidbar.*

Beweis: Das Morphische D0L–Problem ist die Frage $\forall k h(f^k(w)) = g(f^k(w))$, also $f^*(w) \subseteq E(h, g)$. Wir konstruieren eine endliche Testmenge für $f^*(w)$ und rechnen die Gleichheit der Morphismen nach.

Das D0L–Problem fragt $\forall k h^k(w) = g^k(w)$. Das ist äquivalent zu $\forall k h(h^k(w)) = g(h^k(w))$, und das ist ein morphisches D0L–Problem.

Für die HD0L–Folgen $h_1(f_1^k(w_1))$ und $h_2(f_2^k(w_2))$ können wir die Alphabete Σ_i von f_i als disjunkt annehmen. Dann erweitern wir h_i auf $(\Sigma_1 \cup \Sigma_2)^*$ durch $h_i(\Sigma_i) =$