
23. Kontextsensitive Sprachen

In diesem Abschnitt betrachten wir die Klasse der kontextsensitiven Sprachen, der nach der Klasse der allgemeinen Chomsky-Sprachen größten Klasse der Chomsky-Hierarchie. Wir zeigen, dass die kontextsensitiven Sprachen genau die Sprachen vom Erweiterungstyp sind und benutzen diese Äquivalenz für eine Maschinencharakterisierung der kontextsensitiven Sprachen: Die Sprachklasse KS stimmt mit der nichtdeterministisch linearen Platzklasse $\text{NSPACE}(n)$ aus der Komplexitätstheorie überein. Insbesondere sind also – im Gegensatz zu den Typ-0-Sprachen – kontextsensitive Sprachen stets rekursiv.

Zum Nachweis der Gleichheit $\text{KS} = \text{ERW}$ führen wir zunächst eine Normierung von Grammatiken ein.

23.1 DEFINITION. Eine Grammatik $G = (N, T, P, S)$ heißt *separiert*, falls P nur Regeln der Form

$$v \rightarrow w \quad \text{mit } v \in N^+, w \in N^* \quad (\text{Umformungsregeln})$$

und

$$X \rightarrow a \quad \text{mit } X \in N, a \in T \quad (\text{Substitutionsregeln})$$

enthält.

23.2 LEMMA. Jede Grammatik G vom Typ $i \in \{0, 1, 2\}$ oder vom Erweiterungstyp lässt sich effektiv in eine äquivalente separierte Grammatik G' desselben Typs überführen.

BEWEIS. Sei $G = (N, T, P, S)$ gegeben. Die Grammatik $G' = (N', T, P', S)$ erhält man dadurch, dass man zu jedem Terminalzeichen a eine neue Variable \hat{a} einführt, in den Regeln von G alle Vorkommen von Terminalzeichen durch die entsprechenden neuen Variablen ersetzt (Umformungsregeln von G') und die Substitutionsregeln $\hat{a} \rightarrow a$ für jedes Terminalzeichen hinzufügt. Formal ist also $N' = N \cup \hat{T}$, wobei $\hat{T} = \{\hat{a} : a \in T\}$ disjunkt zu N und T gewählt ist, und

$$P' = \{\hat{v} \rightarrow \hat{w} : v \rightarrow w \in P\} \cup \{\hat{a} \rightarrow a : a \in T\},$$

wobei \hat{w} das Bild $h(w)$ von $w \in (N \cup T)^*$ unter dem von $h_0 : (N \cup T) \rightarrow N'$ mit

$$h_0(X) = X \quad \text{für } X \in N \quad \text{und} \quad h_0(a) = \hat{a} \quad \text{für } a \in T$$

erzeugten Homomorphismus $h : (N \cup T)^* \rightarrow N'^*$ ist. □

23.3 SATZ. $\text{KS} = \text{ERW}$

Da jede kontextsensitive Grammatik vom Erweiterungstyp ist, gilt – wie im letzten Abschnitt bereits festgehalten – die Inklusion $KS \subseteq ERW$ trivialerweise. Zum Nachweis der Umkehrung genügt es, folgendes Lemma zu zeigen.

23.4 LEMMA. *Jede Grammatik G vom Erweiterungstyp lässt sich effektiv in eine äquivalente kontextsensitive Grammatik G' überführen.*

BEWEIS. Nach Lemma 23.2 können wir ohne Einschränkung davon ausgehen, dass die gegebene Grammatik $G = (N, T, P, S)$ vom Erweiterungstyp separiert ist. Weiter nehmen wir zunächst an, dass G λ -frei ist. Wir müssen dann jede Umformungsregel

$$r = X_1 \dots X_m \rightarrow Y_1 \dots Y_n \quad (n \geq m \geq 1, X_1, \dots, Y_n \in N)$$

von G durch äquivalente kontextsensitive Regeln ersetzen. Ist $m > 1$ (sonst ist die Regel r schon kontextsensitiv), so führen wir hierzu neue Variablen Z_1, \dots, Z_m ein (für jede Regel sind diese Variablen verschieden zu wählen) und ersetzen r durch die kontextsensitiven Regeln

$$\begin{array}{lcl} X_1 \dots X_m & \rightarrow & Z_1 X_2 \dots X_m \\ Z_1 X_2 \dots X_m & \rightarrow & Z_1 Z_2 \dots X_m \\ & \vdots & \\ Z_1 \dots Z_{m-2} X_{m-1} X_m & \rightarrow & Z_1 \dots Z_{m-1} X_m \\ Z_1 \dots Z_{m-1} X_m & \rightarrow & Z_1 \dots Z_m Y_{m+1} \dots Y_n \\ Z_1 \dots Z_m Y_{m+1} \dots Y_n & \rightarrow & Z_1 \dots Z_{m-1} Y_m \dots Y_n \\ & \vdots & \\ Z_1 Y_2 \dots Y_n & \rightarrow & Y_1 \dots Y_n \end{array}$$

Man ersetzt also (im entsprechenden Kontext) zunächst die Variablen X_1, \dots, X_{m-1} durch Z_1, \dots, Z_{m-1} und die Variable X_m durch $Z_m Y_{m+1} \dots Y_n$. Dann werden (wieder im entsprechenden Kontext) die Hilfsvariablen Z_i in Y_i überführt.

Zum Nachweis, dass die so gewonnene Grammatik G' zu G äquivalent ist, muss man für die nichttriviale Inklusion $L(G') \subseteq L(G)$ zeigen, dass in der Herleitung eines Terminalwortes in G' jede mit $X_1 \dots X_m \rightarrow Z_1 X_2 \dots X_m$ begonnene Simulation der G -Regel r (durch Ausführung der entsprechenden G' -Regeln in der oben aufgeführten Reihenfolge) beendet wird (wir verzichten hier auf Details).

Enthält G die λ -Regel $S \rightarrow \lambda$, so kann die oben beschriebene Umformung die λ -Treue verletzen (z.B. für $X_2 = S$). Hier erhält man $G' = (N', T, P', S)$ dadurch, dass man zunächst $G'_0 = (N'_0, T, P'_0, S)$ zu $G_0 = (N, T, P - \{S \rightarrow \lambda\}, S)$ wie oben konstruiert, und hieraus

$$G' = (N'_0 \cup \{S'\}, T, P'_0 \cup \{S' \rightarrow S, S' \rightarrow \lambda\}, S')$$

durch Hinzufügen eines neuen Axioms S' , das eliminiert oder in das alte Axiom überführt werden kann, gewinnt. \square

Zur Analyse kontextsensitiver Sprachen benutzt man *nichtdeterministische linear beschränkte Automaten* (NLBA), die man als Variante nichtdeterministischer Turingmaschinen wie folgt beschreiben kann: Grob gesprochen ist ein linear beschränkter

Automat eine 1-Band-Turingmaschine, die nur die Felder besucht, auf denen zu Beginn der Rechnung die Eingabe steht. Dies wird realisiert mit Hilfe von vor und nach der Eingabe gesetzten Randmarken, die weder überschrieben noch überschritten werden dürfen. Ein NLBA M beginnt also seine Rechnung bei Eingabe $w \in T^*$ mit der Startkonfiguration

$$\begin{array}{c} \triangleright w \triangleleft \\ 0 \end{array}$$

wobei die Randmarken $\triangleright, \triangleleft$ in dem Bandalphabet von M nicht vorkommen, und 0 der Startzustand von M ist. M verhält sich dann wie eine Turingmaschine, verlässt aber den mit \triangleright und \triangleleft markierten Bereich nie. Ist also das mit \triangleright (bzw. \triangleleft) beschriftete Feld das Arbeitsfeld von M , so wird die Inschrift nicht verändert, und der Kopf läuft nach rechts (bzw. links) oder bleibt stehen. Im Folgenden nehmen wir o.B.d.A. an, dass das Programm eines NLBA M aus bedingten Anweisungen besteht, und es genau einen akzeptierenden Zustand gibt.

Mit NLBA bezeichnen wir auch die Klasse der Sprachen $L(M)$, die von einem nichtdeterministischen linear beschränkten Automaten M erkannt werden. Entsprechend ist DLBA die Klasse der von deterministischen linear beschränkten Automaten erkannte Sprachen. Aufgrund der Definition linear beschränkter Automaten lässt sich mit Hilfe des linearen Kompressionsatzes leicht zeigen:

23.5 LEMMA. *Es gilt*

$$\text{NLBA} = \text{NSPACE}(n+2) = \text{NSPACE}(O(n))$$

und

$$\text{DLBA} = \text{DSPACE}(n+2) = \text{DSPACE}(O(n)).$$

□

Der folgende Satz zeigt also, dass KS mit der nichtdeterministischen Platzklasse $\text{NSPACE}(O(n))$ zusammenfällt.

23.6 SATZ. *Es lässt sich zu jeder kontextsensitiven Grammatik G effektiv ein NLBA M mit $L(G) = L(M)$ angeben und umgekehrt. Insbesondere gilt also $\text{KS} = \text{NLBA}$.*

Wir formulieren die beiden für den Beweis erforderlichen Behauptungen als Lemmata.

23.7 LEMMA. *Jede kontextsensitive Grammatik G lässt sich effektiv in einen NLBA M mit $L(G) = L(M)$ überführen.*

BEWEIS. Von der gegebenen kontextsensitiven Grammatik $G = (N, T, P, S)$ können wir o.B.d.A. annehmen, dass sie separiert ist. Weiter gehen wir davon aus, dass G λ -frei ist. (Ist dies nicht der Fall, so muss der unten angegebene Automat so erweitert werden, dass er zunächst zusätzlich testet, ob die Eingabe leer ist.) Ein NLBA M , der die von G erzeugte Sprache akzeptiert, arbeitet wie folgt: M benutzt zwei Spuren, wobei zunächst die Eingabe w auf die obere Spur kopiert und das Axiom S auf die untere Spur geschrieben wird. Auf der unteren Spur werden dann nichtdeterministisch alle möglichen Herleitungen von G erzeugt, in denen nur Satzformen der Länge $\leq |w|$ vorkommen (diese also auf dem verfügbaren Platz abgespeichert werden können). Da G

vom Erweiterungstyp ist, erhält man so eine Herleitung von w , falls $w \in L(G)$ gilt. Nach Abschluss der Herleitung vergleicht also M die Wörter auf beiden Spuren und akzeptiert bei Gleichheit.

Im Folgenden geben wir ein Programm aus bedingten Anweisungen an, das auf diesen Ideen basiert. Der Startzustand ist 0, + der einzige akzeptierende Zustand. Die Inschriften α der oberen und β der unteren Spur eines Feldes schreiben wir als Paar (α, β) . Weiter sind $a, a' \in T, A \in N, \alpha \in T \cup N \cup \{b\}$ beliebige Buchstaben aus diesen Alphabeten, wobei $b \notin T \cup N \cup \{\triangleright, \triangleleft\}$ das Blankzeichen von M ist.

Die Überführung einer nichtleeren Eingabe $w = a_1 \dots a_n$ ($n \geq 1$) in die Spurendarstellung zu Beginn der Simulation der G -Herleitungen

$$\begin{array}{c} \underline{\triangleright} \\ 0 \end{array} a_1 \dots a_n \triangleleft \xRightarrow{*} \begin{array}{c} \triangleright \\ S \end{array} \begin{array}{c} a_1 \\ b \end{array} \dots \begin{array}{c} a_n \\ b \end{array} \begin{array}{c} \triangleleft \\ 3 \end{array}$$

leisten die Instruktionen

$$\begin{array}{cccc} 0 & \triangleright & \triangleright & R \quad 1 \\ 1 & a & (a, S) & R \quad 2 \\ 2 & a & (a, b) & R \quad 2 \\ 2 & \triangleleft & \triangleleft & S \quad 3 \end{array}$$

Zur folgenden Simulation von G ermöglicht man den Übergang

$$\begin{array}{c} \triangleright \\ S \end{array} \begin{array}{c} a_1 \\ b \end{array} \dots \begin{array}{c} a_n \\ b \end{array} \begin{array}{c} \triangleleft \\ 3 \end{array} \xRightarrow{*} \begin{array}{c} \underline{\triangleright} \\ 3 \end{array} \begin{array}{c} a_1 \\ \alpha_1 \end{array} \dots \begin{array}{c} a_n \\ \alpha_n \end{array} \triangleleft$$

für jedes Wort $\tilde{w} = \alpha_1 \dots \alpha_n$ der Länge n , für das $S \xRightarrow{*} h(\tilde{w})$ gilt, wobei $h : (T \cup N \cup \{b\})^* \rightarrow (T \cup N)^*$ der Homomorphismus ist, der alle Vorkommen von Blanks in \tilde{w} eliminiert (d.h. der von der Funktion $h' : T \cup N \cup \{b\} \rightarrow (T \cup N)^*$ mit $h'(\alpha) = \alpha$ für $\alpha \neq b$ und $h'(b) = \lambda$ induziert wird).

Hierzu benutzt man zunächst die Instruktionen

$$\begin{array}{cccc} 3 & \triangleleft & \triangleleft & L \quad 3 \\ 3 & \triangleright & \triangleright & R \quad 3 \\ 3 & (a, \alpha) & (a, \alpha) & L \quad 3 \\ 3 & (a, \alpha) & (a, \alpha) & R \quad 3 \end{array}$$

und

$$\begin{array}{cccc} 3 & (a, b) & (a, \alpha) & L \quad [\alpha] \\ 3 & (a, b) & (a, \alpha) & R \quad [\alpha] \\ [\alpha] & (a', \alpha) & (a', b) & S \quad 3 \end{array}$$

um das Arbeitsfeld beliebig zu bewegen bzw. auf der unteren Spur die Blanks beliebig zu verschieben.

Jede Umformungsregel $r = X_1 \dots X_m \rightarrow Y_1 \dots Y_n$ ($1 \leq m \leq n$) von G simuliert man dann mit den Instruktionen

$$\begin{array}{cccc} 3 & (a, X_1) & (a, Y_1) & R \quad [r, 2] \\ [r, i] & (a, X_i) & (a, Y_i) & R \quad [r, i+1] & (2 \leq i \leq m) \\ [r, i] & (a, b) & (a, Y_i) & R \quad [r, i+1] & (m < i < n) \\ [r, n] & (a, b) & (a, Y_n) & S \quad 3 \end{array}$$

und jede Substitutionsregel $r = X \rightarrow a$ durch

$$3 \quad (a', X) \quad (a', a) \quad S \quad 3$$

Nach Abschluss der Simulation von G akzeptiert M bei positivem Vergleich der beiden Spuren

$$\begin{array}{c} \triangleright \\ 3 \end{array} \begin{array}{c} a_1 \quad \dots \quad a_n \\ a_1 \quad \dots \quad a_n \end{array} \triangleleft \xrightarrow{*} \triangleright b \dots b \triangleleft \begin{array}{c} \\ + \end{array}$$

mit Hilfe der Instruktionen

$$\begin{array}{cccc} 3 & \triangleright & \triangleright & R & 4 \\ 4 & (a, a) & b & R & 4 \\ 4 & \triangleleft & \triangleleft & S & + \end{array}$$

Auf den Korrektheitsbeweis verzichten wir. □

23.8 LEMMA. *Jeder NLBA M lässt sich effektiv in eine Grammatik G vom Erweiterungstyp mit $L(M) = L(G)$ überführen.*

BEWEIS. Wir gehen wie im Beweis von Lemma 21.12 vor, in dem die Rechnungen einer Turingmaschine durch eine Chomsky-Grammatik simuliert wurden. Da bei einem NLBA Konfigurationslänge und Eingabelänge (im Wesentlichen) gleich sind, ist die Simulation hier (mit geeigneten kleineren Modifikationen) mit Hilfe einer Grammatik vom Erweiterungstyp möglich.

Sei also ein NLBA M gegeben, Σ, Γ, Z das Eingabe- bzw. Bandalphabet und die Zustandsmenge von M , wobei o.B.d.A. $Z \cap \Gamma = \emptyset$ gelte, und 0 und 1 der Start- bzw. einzige akzeptierende Zustand von M seien. Weiter können wir annehmen, dass M stets auf der linken Randmarke \triangleright stoppt.

Eine Grammatik $G = (N, \Sigma, P, S)$, die $L(M)$ erzeugt, enthält folgende Regeln. Für die endlich vielen Wörter $w \in L(M)$ der Länge ≤ 2 sichert man die Herleitbarkeit durch Regeln $S \rightarrow w$. Für die anderen Wörter aus $L(M)$ wird die Herleitbarkeit durch Regeln gesichert, die die Arbeitsweise von M beschreiben. Die Satzformen von G bestehen hierzu aus zwei Spuren. Während ein Kandidat $w = a_1 \dots a_n$ ($n \geq 3$) für ein Wort $w \in L(M)$ in der oberen Spur steht, wird in der unteren Spur eine aus der zu w gehörenden Startkonfiguration erreichbare Konfiguration stehen. Dabei schreiben wir den Zustand vor den Buchstaben auf dem Arbeitsfeld und wählen jedes Paar bestehend aus Zustand und Feldinschrift als Variable in N . Weiter verschmelzen wir die Randmarken mit dem ersten bzw. letzten Buchstaben der Bandinschrift. (Durch diese Maßnahmen stimmen Eingabe- und Konfigurationslänge exakt überein, weshalb wir verkürzende Regeln vermeiden können.) Formal transponieren wir die Spaltenvektoren, schreiben also (α, β) für α in der oberen und β in der unteren Spur eines Feldes. Das Alphabet N der syntaktischen Variablen von G wird also neben dem Axiom S und einer Hilfsvariablen U alle Paare

$$(a, \alpha), (a, \triangleright\alpha), (a, \alpha\triangleleft), (a, z\alpha), (a, z\triangleright\alpha), (a, \triangleright z\alpha), (a, z\alpha\triangleleft), (a, \alpha z\triangleleft)$$

enthalten, wobei $a \in \Sigma$, $\alpha \in \Gamma$ und $z \in Z$ gilt.

In den folgen Regeln sind jeweils $a, a' \in \Sigma$, $\alpha, \alpha', \alpha'' \in \Gamma$, $z, z' \in Z$ beliebige Buchstaben aus den genannten Alphabeten. Für jedes Wort $w = a_1 \dots a_n \in \Sigma^*$ mit $n \geq 3$ ermöglicht man zunächst

$$S \xrightarrow{*} \begin{array}{c} a_1 \quad a_2 \quad \dots \quad a_n \\ \triangleright 0a_1 \quad a_2 \quad \dots \quad a_n \triangleleft \end{array}$$

durch die Regeln

$$\begin{aligned} S &\rightarrow (a, \triangleright 0a)U \\ U &\rightarrow (a, a)U \\ U &\rightarrow (a, a\triangleleft) \end{aligned}$$

Zu jeder Instruktion $I = (z, \alpha, \alpha', B, z')$ führt man Simulationsregeln ein, die von der Kopfbewegung abhängen. Für den Fall der Linksbewegung $B = L$ (andere Fälle: Übung) sind dies

$$\begin{aligned} (a, \triangleright z\alpha) &\rightarrow (a, z' \triangleright \alpha') \\ (a', \triangleright \alpha'')(a, z\alpha) &\rightarrow (a', \triangleright z' \alpha'')(a, \alpha') \\ (a', \alpha'')(a, z\alpha) &\rightarrow (a', z' \alpha'')(a, \alpha') \\ (a', \alpha'')(a, z\alpha\triangleleft) &\rightarrow (a', z' \alpha'')(a, \alpha'\triangleleft) \end{aligned}$$

Entsprechend simuliert man Instruktionen, die die Randmarken betreffen. Z.B. entspricht der Instruktion $I = (z, \triangleright, \triangleright, R, z')$ die Regel

$$(a, z \triangleright \alpha) \rightarrow (a, \triangleright z' \alpha)$$

Mit diesen Simulationsregeln kann man mit w in der oberen Spur alle Konfigurationen (und nur diese) in der unteren Spur erzeugen, die M bei Eingabe w erreichen kann. Ist solch eine Konfiguration akzeptierend, kann man die untere Spur auflösen mit Hilfe der Regeln

$$\begin{aligned} (a, 1 \triangleright \alpha) &\rightarrow a \\ a(a', \alpha) &\rightarrow aa' \\ a(a', \alpha\triangleleft) &\rightarrow aa' \end{aligned}$$

Wir verzichten auf den Korrektheitsbeweis. □

Fassen wir die Sätze 23.3 und 23.6 sowie Lemma 23.5 zusammen, so erhalten wir

23.9 KOROLLAR. $KS = ERW = NLBA = NSPACE(O(n))$.

Insbesondere ist also jede kontextsensitive Sprache rekursiv, woraus sich die Echtheit der Inklusion $KS \subseteq CH$ ergibt:

23.10 KOROLLAR. $KS \subsetneq CH$.

BEWEIS. Das Halteproblem K ist rekursiv aufzählbar aber nicht rekursiv. Nach Satz 21.14 und Korollar 23.9 gilt also $K \in CH - KS$. □

Setzen wir eine Gödelisierung der Grammatiken voraus, so können wir (wie am Ende von Abschnitt 21 bereits für allgemeine Grammatiken gemacht) das Wortproblem

$$W_{KS} = \{\ulcorner G \urcorner, x\} : G \text{ kontextsensitiv \& } x \in L(G)\},$$

das Leerheitsproblem

$$LEER_{KS} = \{\ulcorner G \urcorner, x\} : G \text{ kontextsensitiv \& } L(G) = \emptyset\},$$

das Unendlichkeitsproblem

$$\text{INF}_{\text{KS}} = \{\ulcorner G \urcorner, x\} : G \text{ kontextsensitiv \& } L(G) \text{ unendlich}\},$$

und das Äquivalenzproblem

$$\text{ÄQU}_{\text{KS}} = \{\ulcorner G \urcorner, \ulcorner G' \urcorner\} : G, G' \text{ kontextsensitiv \& } L(G) = L(G')\}$$

für kontextsensitive Grammatiken betrachten.

Im Gegensatz zum Wortproblem W_{CH} für allgemeine Chomsky-Grammatiken ist das Wortproblem W_{KS} für kontextsensitive Grammatiken entscheidbar.

23.11 KOROLLAR. W_{KS} ist rekursiv.

BEWEISIDEE. Um $(e, x) \in W_{\text{KS}}$ zu entscheiden, berechnet man zunächst aus e die Grammatik G mit Gödelnummer e . Ist e keine Gödelnummer oder G nicht kontextsensitiv, so gilt $(e, x) \notin W_{\text{KS}}$. Sonst führt man mit Hilfe von Lemma 23.7 G effektiv in einen äquivalenten NLBA M über, für den man effektiv feststellen kann, ob er die Eingabe x akzeptiert. \square

Die anderen oben aufgeführten Probleme sind jedoch auch für KS (wie im allgemeinen Fall) unentscheidbar. Dies zeigt man mit der Reduktionsmethode mit Hilfe der folgenden Verschärfung des Projektionslemmas.

23.12 LEMMA. Jede rekursiv aufzählbare Sprache A lässt sich als Projektion einer kontextsensitiven Sprache B darstellen. Darüberhinaus lässt sich aus einem Index e für A ($A = W_e$) eine kontextsensitive Grammatik G , die B erzeugt, effektiv berechnen.

BEWEIS. Sei die rekursiv aufzählbare Sprache $A = W_e$ gegeben, d.h. A enthält alle Wörter x , für die die e -te Turingmaschine M_e stoppt. Wir definieren (wie im Beweis des Projektionslemmas)

$$B = \{(x, c_0\# \dots \#c_n) : c_0, \dots, c_n \text{ ist term. } M_e\text{-Rechnung bei Eingabe } e\}.$$

(Hierbei sind die M_e -Konfigurationen c_i geeignet durch Wörter kodiert.) Offensichtlich ist A die Projektion von B , weshalb es genügt, $B \in \text{KS}$ zu zeigen. Ob ein Wort $c_0\# \dots \#c_n$ eine terminierende M_e -Rechnung bei Eingabe x beschreibt, kann man jedoch mit einer (deterministischen) Turingmaschine ohne zusätzlichen Platzbedarf, also insbesondere durch einen NLBA N überprüfen. Dieser lässt sich dabei effektiv aus M_e gewinnen. Mit den Lemmata 23.8 und 23.4 erhalten wir aber aus N effektiv die gewünschte kontextsensitive Grammatik zur Erzeugung von B . \square

23.13 SATZ. LEER_{KS} , INF_{KS} und ÄQU_{KS} sind nicht rekursiv.

BEWEIS. Zum Nachweis der Nichtrekursivität von LEER_{KS} reduzieren wir das – nach dem Satz von Rice nichtrekursive – Leerheitsproblem

$$\text{LEER} = \{e : W_e = \emptyset\} = \{e : M_e \text{ terminiert für keine Eingabe}\}$$

für Turingmaschinen auf diese Menge. Hierzu wählen wir mit Lemma 23.12 eine rekursive Funktion f , die den Index e einer rekursiv aufzählbaren Menge $A = W_e$ auf

die Gödelnummer $e' = f(e)$ einer kontextsensitiven Grammatik $G_{e'}$ abbildet, die eine Menge B_e erzeugt, deren Projektion W_e ist. Es gilt dann

$$\begin{aligned} e \in \text{LEER} &\Leftrightarrow A = W_e \text{ leer} \\ &\Leftrightarrow B_e = L(G_{f(e)}) \text{ leer} \\ &\Leftrightarrow f(e) \in \text{LEER}_{\text{KS}} \end{aligned}$$

d.h. $\text{LEER} \leq_m \text{LEER}_{\text{KS}}$ via f .

Hierbei haben wir benutzt, dass die Menge $B = B_e$ im Beweis von Lemma 23.12 so gewählt ist, dass B_e genau dann leer ist, wenn W_e leer ist. Da es zu jedem x höchstens ein y mit $(x, y) \in B_e$ gibt (da M_e deterministisch!), gilt sogar $\|W_e\| = \|B_e\|$. Dies zeigt, dass auch $\text{INF} = \{e : W_e \text{ unendlich}\}$ auf INF_{KS} vermöge f many-one-reduzierbar und damit auch INF_{KS} nicht rekursiv ist.

Zum Nachweis der Nichtrekursivität von ÄQU_{KS} beobachtet man schließlich, dass LEER_{KS} auf ÄQU_{KS} vermöge $g(e) = (e, e_0)$ many-one-reduzierbar ist, wobei e_0 die Gödelnummer einer kontextsensitiven Grammatik G_{e_0} ist, die die leere Sprache erzeugt:

$$\begin{aligned} e \in \text{LEER}_{\text{KS}} &\Leftrightarrow L(G_e) = \emptyset \\ &\Leftrightarrow L(G_e) = L(G_{e_0}) \\ &\Leftrightarrow (e, e_0) \in \text{ÄQU}_{\text{KS}} \end{aligned}$$

□

Aus den Maschinen-Charakterisierungen der allgemeinen bzw. kontextsensitiven Chomsky-Sprachen können wir Abschlusseigenschaften von CH und KS ableiten, aus denen sich deren Ungleichheit ebenfalls ergibt.

23.14 LEMMA. *CH ist gegen Vereinigung und Durchschnitt, nicht aber gegen Komplement abgeschlossen. KS ist gegen Vereinigung, Durchschnitt und Komplement abgeschlossen.*

BEWEIS. Dies folgt aus den entsprechenden Abschlusseigenschaften von $\text{RA}(= \text{CH})$ bzw. $\text{NSPACE}(O(n))(= \text{KS})$. □

In der Theorie formaler Sprachen betrachtet man häufig noch den Abschluss unter Homomorphismen (oder genauer: homomorphe Bilder). Ist $L \subseteq \Sigma^*$ eine Sprache und $h : \Sigma^* \rightarrow T^*$ ein Homomorphismus, so heißt die Sprache $h(L) \subseteq T^*$ mit $h(L) = \{h(w) : w \in L\}$ das *homomorphe Bild* von L bzgl. h .

23.15 LEMMA. *CH, nicht jedoch KS, ist gegen homomorphe Bilder abgeschlossen.*

BEWEIS. Den Abschluss von CH gegen Homomorphismen zeigt man unter Verwendung der Identität $\text{CH} = \text{RA}$. Sei $L \subseteq \Sigma^*$ in CH, und sei $h : \Sigma^* \rightarrow T^*$ ein Homomorphismus. Da h durch seine Werte auf den einzelnen Buchstaben bestimmt ist, ist h rekursiv. Wegen

$$x \in h(L) \Leftrightarrow \exists y (x = h(y) \ \& \ y \in L)$$

folgt hieraus $h(L) \in \text{RA}$ mit Hilfe der Abschlusseigenschaften der rekursiv aufzählbaren Mengen.

Um zu zeigen, dass KS nicht gegen homomorphe Bilder abgeschlossen ist, zeigen wir, dass jede rekursiv aufzählbare Sprache, d.h. jede Chomsky-Sprache das homomorphe Bild einer kontextsensitiven Sprache ist (woraus sich die Behauptung dann wegen $\text{KS} \subsetneq \text{CH}$ ergibt). Hierzu verwenden wir Lemma 23.12 und die Beobachtung, dass man Projektionen als homomorphe Bilder auffassen kann. Sei also $L \in \text{RA} = \text{CH}$, $L \subseteq \Sigma^*$, gegeben. Nach Lemma 23.12 gibt es eine kontextsensitive Sprache L' mit

$$\forall x \in \Sigma^* (x \in L \Leftrightarrow \exists y (x\#y \in L')).$$

Hierbei benutzen wir das Hilfszeichen $\#$ zur Repräsentation von Paaren ($x\#y$ statt (x, y)), und durch eventuelle Umbenennung können wir davon ausgehen, dass y ein Wort über einem zu Σ disjunkten Alphabet T ist. Der von der Funktion $h_0 : \Sigma \cup T \cup \{\#\} \rightarrow \Sigma^*$ mit $h_0(a) = a$ für $a \in \Sigma$ und $h_0(a) = \lambda$ für $a \in T \cup \{\#\}$ induzierte Homomorphismus $h : (\Sigma \cup T \cup \{\#\})^* \rightarrow \Sigma^*$ eliminiert dann den zweiten Teil $\#y$, projiziert also das Paar $x\#y$ auf die erste Komponente. D.h. $L = h(L')$. \square