Chomsky hierarkiet af sprogklasser

Torben Mogensen

Juli 2001

I oversætterbogen [Mog01] beskrives to klasser af sprog: De regulære sprog, beskrevet med regulære udtryk og endelige automater samt de kontekstfri sprog, beskrevet med kontekstfri grammatikker. Der findes flere klasser af sprog, som kan organiseres i et hierarki baseret på udseendet af de grammatikker, der bruges til at beskrive sprogene. Dette hierarki kaldes Chomsky hierarkiet efter sprogforskeren Noam Chomsky, som først beskrev klassificeringen i 1956 [Cho56].

Vi starter med at beskrive en generaliseret form for grammatik, som beskriver den mest frie klasse af sprog. Derefter vil vi i flere trin indføre begrænsninger på grammatikkerne, for dermed at indskrænke den klasse af sprog, som grammatikkerne kan beskrive. Vi vil beskrive forskellige egenskaber ved klasserne, og se på nogle underklassificeringer af de kontekstfri sprog.

1 Grammatikker

En grammatik består formelt set af fire komponenter: $G = (N, T, P, S)$, hvor N er mængden af nonterminaler, T er mængden af terminalsymboler, P er mængden af produktioner og $S \in N$ er startsymbolet. N og T er disjunkte mængder, dvs. $N \cap T = \emptyset$. Alle mængderne er endelige, og en produktion har formen

$$\beta \to \alpha \quad \text{hvor } \beta \in (N \cup T)^+ \text{ og } \alpha \in (N \cup T)^*$$

Det sprog en grammatik beskriver, er mængden af tegnfølger, som man kan få ved at starte med startsymbolet S og gentagne gange bruge produktionerne som omskrivningsregler, indtil resultatet er en følge af terminalsymboler. Mere præcist har vi omskrivningsrelationen $\Rightarrow$ over mængden $(N \cup T)^*$, defineret ved

$$\gamma\beta\delta \Rightarrow \gamma\alpha\delta \quad \text{hvis og kun hvis } \beta \to \alpha \in P$$

hvor $\gamma, \delta \in (N \cup T)^*$.

Vi definerer relation $\Rightarrow^*$ som den refleksive og transitive afslutning af $\Rightarrow$, dvs.

$$\gamma \Rightarrow^* \gamma$$
$$\gamma \Rightarrow^* \zeta \quad \text{hvis } \gamma \Rightarrow \delta \text{ og } \delta \Rightarrow^* \zeta$$

Vi definerer nu sproget $L(G)$, som G genererer ved

$$L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$$

2 Chomsky hierarkiet

Chomsky hierarkiet omfatter klasser af sprog, der alle kan beskrives med de ovennævnte grammatikker. Men der findes sprog, der ikke kan beskrives med grammatikker overhovedet (antallet af grammatikker er lig med størrelsen af $\mathbb{N}$ (mængden af naturlige tal), men antallet af sprog er lig med størrelsen af $\mathbb{R}$ (mængden af reelle tal), altså findes der flere sprog end grammatikker). Derfor findes der udenom den mest generelle klasse i Chomsky hierarkiet (type 0 sprog) klassen af alle mulige sprog. Der findes ingen beskrivelsesmetode, som kan beskrive alle sprog, så det er ikke som sådan en brist ved grammatikker, at de heller ikke kan.

Selv om vi beskriver Chomsky hierarkiet via *grammatikkernes* udseende, er der faktisk tale om et hierarki af *sprog*klasser. Nogle af sprogklasserne kan beskrives med forskellige (men ækvivalente) klasser af grammatikker. Et sprog er et type n sprog, hvis det kan genereres af en type n grammatik. Hvis det er tilfældet, kan sproget iøvrigt genereres af uendeligt mange forskellige type n grammatikker.

Det gælder generelt, at type $n+1$ sprog også er type n sprog, da type $n+1$ er defineret som en indskrænkning af type n.

Type 0 sprog: Uindskrænkede sprog

Uindskrænkede grammatikker har, som navnet antyder, ingen begrænsninger på formen af produktionerne (udover de krav, som blev stillet i afsnit 1).

Et eksempel på et sprog af type 0, som ikke er i de mindre klasser er mængden af Turingmaskineprogrammer (se afsnit 5), der terminerer med tomt inddatabånd.

Type 1 sprog: Kontekstfølsomme sprog

Her sætter vi en begrænsning på formen af produktionerne: I en produktion $\beta \rightarrow \alpha$ skal længden af β være mindre end eller lig med længden af α, dvs. at produktionerne har formen

$$\beta \rightarrow \alpha \quad \text{hvor } \alpha, \beta \in (N \cup T)^+,\ |\beta| \leq |\alpha|$$

Navnet "kontekstfølsom" stammer fra en anden karakterisering af samme klasse. Bemærk, at højresiden af en produktion ikke må være tom (da den ikke må være kortere end venstresiden, som ikke er tom). Derfor kan kontekstfølsomme grammatikker ikke beskrive sprog, der indeholder den tomme tegnfølge. Hvis man udvider type 1 grammatikker med tomme produktioner, får man alle type 0 sprog. Man kan dog udvide kontekstfølsomme grammatikker til at omfatte sprog med den tomme tegnfølge (og intet andet ekstra) ved at tillade en tom produktion med startsymbolet på venstresiden, hvis startsymbolet ikke forekommer på højresiden af nogen produktion.

Et eksempel på et type 1 sprog, som ikke er type 2, er sproget $\{ww|w \in \{a,b\}^*\}$, dvs. sproget af alle tegnfølger (over alfabetet $\{a,b\}$), som består af to ens halvdele. En grammatik for dette sprog er

$$\begin{array}{ll} S & \rightarrow CaS \mid CbS \mid aA' \mid bB' \\ Ca & \rightarrow Aa \\ Cb & \rightarrow Bb \\ Aa & \rightarrow aA \\ Ab & \rightarrow bA \\ AA' & \rightarrow aA' \\ AB' & \rightarrow aB' \\ Ba & \rightarrow aB \\ Bb & \rightarrow bB \\ BA' & \rightarrow bA' \\ BB' & \rightarrow bB' \\ A' & \rightarrow a \\ B' & \rightarrow b \end{array}$$

Det, der sker i denne grammatik, er at S opbygger en sekvens af a'er og b'er med det samme antal C'er foran. C'erne sørger for at kopiere tegnene til enden af tegnfølgen, så resultatet bliver to kopier af den oprindelige tegnfølge. A' eller B' sørger for kopiering af det sidste tegn.

Type 2 sprog: Kontekstfri sprog

Her sætter vi yderligere den begrænsning, at venstresiden af en produktion skal være en enkelt nonterminal. Da vi stadig har begrænsningen om at højresiden ikke må være kortere end venstresiden, må den stadig ikke være tom. Produktionerne har altså formen

$$A \rightarrow \alpha \quad \text{hvor } A \in N,\ \alpha \in (N \cup T)^+$$

I [Mog01] tillades tomme højresider i kontekstfri grammatikker. Det er en relativt lille udvidelse, da man kan vise, at hvis en kontekstfri grammatik med tomme

produktioner kan generere sproget L, så findes en kontekstfri grammatik uden tomme produktioner, der genererer sproget $L\backslash\{\varepsilon\}$.

Et eksempel på et type 2 sprog, som ikke er et type 3 sprog er sproget $\{a^n b^n \mid n \in \mathbb{N}\}$, dvs. mængden af tegnfølger som starter med et antal a'er og fortsætter med det samme antal b'er. En grammatik for dette sprog er

$$\begin{aligned} S &\rightarrow ab \\ S &\rightarrow aSb \end{aligned}$$

Type 3 sprog: Regulære sprog

Her må højresiderne af produktionerne kun have to former:

$$\begin{aligned} A &\rightarrow w \quad \text{hvor } w \in T^+ \\ A &\rightarrow wB \quad \text{hvor } w \in T^*,\ B \in N \end{aligned}$$

Også her vil man nogle gange udvide med tomme produktioner, og igen vil det kun udvide sprogene med den tomme tegnfølge.

Et eksempel på et type 3 sprog, som ikke er et type 4 sprog er sproget $\{(ab)^n \mid n \in \mathbb{N}\}$, dvs. mængden af tegnfølger, der består af skiftevis a'er og b'er. En regulær grammatik for dette sprog er

$$\begin{aligned} S &\rightarrow ab \\ S &\rightarrow abS \end{aligned}$$

Type 4 sprog: Endelige sprog

Her begrænser vi produktionerne til

$$S \rightarrow w \text{ hvor } w \in T^+$$

Da startsymbolet er den eneste nonterminal, der forekommer, er $N = \{S\}$, og grammatikken er en opremsning af en endelig mængde af tegnfølger. Igen vil vi ofte udvide med tomme produktioner.

Et eksempel på et endeligt sprog er $\{aa, bb\}$. Et andet eksempel er mænden af alle gensekvenser for alle dyr på jorden. Der kan altså være tale om ret store og komplicerede mængder, bare de er endelige.

3 Egenskaber ved sprogklasserne

Det ville være uinteressant at lave en klassificering af sprog, hvis ikke de forskellige klasser havde forskellige egenskaber. De fleste af de herunder nævnte resultater findes i [AHU74] og [HU79].

Det, der interesserer os dataloger mest ved grammatikker, er syntaksanalyse, dvs. spørgsmålet om en given tegnfølge ligger i det sprog, som en grammatik genererer. Det viser sig, at for type 0 sprog er dette spørgsmål ikke altid afgørligt. Dvs. at der ikke findes nogen metode, der kan afgøre spørgsmålet for alle grammatikker og alle tegnfølger. Type 0 grammatikker er altså som sådan uegnede til at beskrive programmeringssprog (man ville ganske enkelt ikke kunne skrive en oversætter). For typerne 1-4 er syntaksanalysespørgsmålet afgørligt og beregneligt. Alle disse klasser kan altså i princippet bruges til at beskrive programmeringssprog. Men vi interesserer os ikke kun for *om* vi kan løse et problem, men også hvor lang tid vi er om det. Og her er der faktisk en stor forskel mellem de forskellige typer af grammatikker og tilhørende sprog. Tabellen her viser kompleksiteten af de bedst kendte metoder til syntaksanalyse for de forskellige klasser.

Sprogklasse	kompleksitet af $w \in L(G)$, $\|w\| = n$
Type 0	uafgørligt
Type 1	$O(2^n)$
Type 2	$O(n^{\sqrt{8}})$
Type 3	$O(n)$
Type 4	$O(n)$

Den eksponentielle køretid for syntaksanalyse af type 1 sprog gør dem uegnede til brug for programmeringssprog. Selv den næsten kubiske ($\sqrt{8} \simeq 2.8$) køretid for syntaksanalyse af kontekstfri sprog er urimelig. Man kunne derfor tro, at kun regulære sprog ville være rimelige til beskrivelse af syntaks for programmeringssprog. Der findes dog heldigvis underklasser af de kontekstfri sprog, der tillader syntaksanalyse i lineær tid. Disse underklasser bliver normalt karakteriseret ved den anvendte syntaksanalysemetode (LL eller LR, se [Mog01]), da det ikke er nemt at beskrive klasserne ved begrænsninger i produktionerne.

En anden egenskab, vi gerne vil have ved grammatikker, er entydighed. Det kan også være interessant at afgøre om to grammatikker genererer det samme sprog (at de er ækvivalente), eller om en grammatik i det hele taget genererer et ikke-tomt sprog. Disse egenskaber er dog heller ikke afgørlige for alle sprogklasser. For de forskellige klasser gælder der

Afgørlighed af forskellige egenskaber			
Sprogklasse	Entydighed	Ækvivalens	$L(G) \neq \emptyset$
Type 0	nej	nej	nej
Type 1	nej	nej	nej
Type 2	nej	?	ja
Type 3	ja	ja	ja
Type 4	ja	ja	ja

Det er endnu ikke kendt om ækvivalens af kontekstfri grammatikker er afgørligt.

Det gælder for alle klasserne at foreningsmængden af to type n sprog er et type n sprog (det er nemt at konstruere en grammatik for foreningsmængden ud fra grammatikkerne for to sprog). For de fleste klasser gælder det også, at fællesmængden af to sprog eller komplementærmængden af et sprog tilhører samme klasse som sprogene selv, men for type 2 (kontekstfri) sprog findes der tilfælde, hvor fællesmængden eller komplementærmængden ikke er et type 2 sprog, men kræver en type 1 grammatik. Et eksempel på et kontekstfrit sprog, hvor komplementærmængden ikke er kontekstfri, er sproget $\{vw \mid v,w \in \{a,b\}^*,\ |v| = |w|,\ v \neq w\}$, dvs. sproget af alle tegnfølger, der består af to forskellige halvdele. En kontekstfri grammatik, der beskriver dette sprog er

$$\begin{array}{lll} S & \rightarrow AB \mid BA \\ A & \rightarrow a \mid CAC \\ B & \rightarrow b \mid CBC \\ C & \rightarrow a \mid b \end{array}$$

A genererer sproget $\{\{a,b\}^n a \{a,b\}^n \mid n \in \mathbb{N}\}$ og B genererer $\{\{a,b\}^n b \{a,b\}^n \mid n \in \mathbb{N}\}$. Det betyder at AB genererer $\{\{a,b\}^m a \{a,b\}^m \{a,b\}^n b \{a,b\}^n \mid m,n \in \mathbb{N}\}$. Det er det samme som sproget $\{\{a,b\}^m a \{a,b\}^n \{a,b\}^m b \{a,b\}^n \mid m,n \in \mathbb{N}\}$, som er sproget af alle tegnfølger hvor den første halvdel mindst et sted indeholder et a på en position, hvor den anden halvdel har et b. BA klarer den modsatte situation, så tilsammen sikrer de, at der er mindst én forskel mellem de to halvdele.

4 Underklasser af kontekstfri sprog

Som nævnt i afsnit 3, findes der underklasser af kontekstfri sprog, som udmærker sig ved at have effektive syntaksanalysemetoder. I [Mog01] beskrives to metoder: LL syntaksanalyse og LR syntaksanalyse. Begge disse bruger deterministiske stakautomater til syntaksanalyse, og bruger derfor lineær tid til dette. Det er ikke nemt at se på en grammatik, om den tilhører en af disse klasser, så man vil normalt prøve at konstruere en syntaksanalysetabel ud fra grammatikken. Hvis det går godt, ligger grammatikken i klassen, ellers ikke.

Ligesom i Chomsky hierarkiet definerer disse klasser af grammatikker også klasser af sprog (de sprog, som kan genereres af grammatikker fra klasserne). Det kan vises at klassen LL indeholder mere end regulære sprog, men mindre end klassen LR. Alle LR grammatikker (og dermed også LL grammatikker) er entydige, men der findes også entydige grammatikker, der ikke er LR. Man kan også vise, at der findes kontekstfri sprog, som slet ikke kan genereres af entydige grammatikker. Et eksempel herpå er sproget $\{vw \mid v,w \in \{a,b\}^*,\ |v| = |w|,\ v \neq w\}$. Vi har altså

$$\text{Type } 3 \subset \text{LL} \subset \text{LR} \subset \text{entydige} \subset \text{Type } 2$$

Et helt nyt resultat er, at ækvilalens af LR grammatikker (og dermed også LL grammatikker) er afgørligt.

5 Automater

I [Mog01] konstruerer vi endelige automater (NFA'er og DFA'er) ud fra regulære udtryk, og bruger disse automater til at afgøre om en tegnfølge ligger i sproget. LL og LR syntaksanalysatorer bruger tabeller, som styrer deterministiske stakautomater.

Til hver sprogklasse i Chomsky hierarkiet hører en klasse af automater. For hvert sprog i klassen findes i den tilsvarende klasse en automat, der kan afgøre om en tegnfølge ligger i sproget, og for hver automat findes der ligeledes et sprog i den tilsvarende klasse. Man kan altså karakterisere Chomsky hierarkiet via automater i stedet for grammatikker.

Type 4 (endelige) sprog kan genkendes af endelige ikke-cycliske automater (f.eks. trier eller *decision diagrams*).

Type 3 (regulære) sprog kan genkendes af endelige automater. Disse har et bånd med inddata, kan kun læse det og kun rykke frem på båndet. I [Mog01] ser vi at ikke-determinisme ikke øger mængden af sprog, som endelige automater kan genkende, idet en NFA kan konverteres til en ekvivalent DFA. Endelige automater kan simuleres i lineær tid i inddatas størrelse.

Type 2 (kontekstfri) sprog kan genkendes af ikke-deterministiske stakautomater. Lageret i en stakautomat er en stak, hvor kun topelementet kan læses. Automaten kan lægge et nyt element på toppen af stakken eller fjerne det øverste. Udover stakken kan automaten læse inddata på et separat bånd. Men dette bånd kan der ikke skrives på, og pegepinden kan kun rykkes frem, aldrig tilbage. Ikke-determinisme betyder (som for endelige automater), at flere overgange kan defineres i en tilstand, og kun et af disse behøver at give succes. Ikke-deterministiske stakautomater kan simuleres i næsten-kubisk tid i inddatas størrelse.

For type 1 (kontekstfølsomme) sprog er automaterne ikke-deterministiske Turingmaskiner med begrænset båndlængde. Disse virker som Turingmaskiner (se nedenunder), men båndet er af samme længde som inddata, og kan ikke udvides. Ikke-determinisme har den sædvanlige betydning. Begrænsede ikke-deterministiske Turingmaskiner kan simuleres i en tid, der er eksponentiel i båndets størrelse.

For type 0 (ubegrænsede) sprog er den tilsvarende klasse af automater Turingmaskiner. En Turingmaskine har et uendeligt bånd (lager), som i starten indeholder inddata. Der er én pegepind til båndet, og denne pegepind kan kun flyttes

én position til venstre eller højre i et tidsskridt. Turingsmakinen kan læse eller skrive et symbol på båndet ud for pegepinden, og kan skifte tilstand på baggrund af det læste symbol. Turingmaskiner er ækvivalente med datamaskiner med ubegrænset lager. Vi nævnte i afsnit 3, at syntaksanalysespørgsmålet for type 0 sprog er uafgørligt. Det betyder, at for nogle sprog vil den tilsvarende Turingmaskine ikke altid terminere. Ikke-determinisme tilføjer ikke styrke til ubegrænsede Turingmaskiner.

5.1 Andre automater

Som nævnt er LL og LR syntaksanalyse baseret på *deterministiske* stakautomater. Disse kan simuleres (køres) i lineær tid, så de er velegnede til syntaksanalyse. Selv om LL og LR bruger samme type automat, er LR en større klasse. LR klassen er ækvivalent med klassen af deterministiske stakautomater, mens LL ikke kommer rundt i alle hjørner af denne.

Både de endelige automater for regulære sprog og stakautomaterne for de kontekstfri sprog kan kun rykke frem på deres inddatabånd. Selv om vi tillader endelige automater at læse begge veje, får vi ikke flere sprog ud af det, men stakautomater bliver stærkere af denne udvidelse.

Deterministiske tovejs stakautomater (2DPDA) kan genkende visse kontekstfølsomme sprog (som f.eks. $\{ww \mid w \in \{a,b\}^*\}$), men ikke alle kontekstfri sprog. De kan simuleres i tid lineær i inddatas størrelse.

Ikke-deterministiske tovejs stakautomater kan simuleres i næsten-kubisk tid, og genkender alle kontekstfri og visse (men ikke alle) kontekstfølsomme sprog.

Da deterministiske tovejs stakautomater kan simuleres i lineær tid, er de i princippet velegnede til syntaksanalyse. Der er dog ikke nogen nem måde at konstruere en deterministisk tovejs stakautomat ud fra en grammatik (hvadenten den er kontekstfri eller kontekstfølsom), så jeg kender ikke til nogen oversættere, der bruger dem til syntaksanalyse.

Man kan læse mere om automater og deres egenskaber i [HU79] og [AHU74].

5.2 Regulære grammatikker og endelige automater

I [Mog01] ses konstruktion af endelige automater ud fra regulære udtryk, og i Nils Andersens "Notat om regulære mængder" ses den omvendte konstruktion, men regulære grammatikker nævnes ikke. Det er dog let at konvertere regulære grammatikker til automater og omvendt.

Fra regulær grammatik til NFA

Vi starter med at lave en NFA tilstand t_i for hver nonterminal A_i i grammatikken. Tilstanden svarende til startsymbolet S er starttilstanden i NFA'en.

Hvis der er en produktion $A_i \to wA_j$ i grammatikken, laves der $n-1$ nye tilstande $s_1, \ldots, s_{n-1}$, hvor $n = |w|$. Der laves en overgang fra t_i til s_1 på det første tegn i w, og fra s_1 til s_2 på det andet tegn osv. Til sidst laves en overgang fra s_{n-1} til t_j på det sidste tegn i w. Hvis $|w| = 0$ laves en ε-overgang fra t_i til t_j.

Hvis der er en produktion $A_i \to w$ i grammatikken, laves der n nye tilstande $s_1, \ldots, s_n$, hvor $n = |w|$. Der laves en overgang fra t_i til s_1 på det første tegn i w, og fra s_1 til s_2 på det andet tegn osv. s_n gøres accepterende.

Fra NFA til regulær grammatik

Der laves en nonterminal A_i for hver tilstand t_i i NFA'en. Den nonterminal, der svarer til starttilstanden er startsymbolet i grammatikken.

Hvis der er en overgang fra t_i til t_j på symbolet a, laves en produktion $A_i \to aA_j$. Hvis der er en ε-overgang fra t_i til t_j laves en produktion $A_i \to A_j$.

Hvis t_i er en accept-tilstand, laves en produktion $A_i \to \varepsilon$. Her har vi brugt udvidelsen af de regulære grammatikker med tomme produktioner, men de kan trivielt fjernes (så længe sproget ikke indeholder den tomme tegnfølge).

6 Pumpelemmaerne

Hvis man vil vise at et givet sprog er regulært kan man gøre det ved at konstruere et regulært udtryk eller en automat (DFA eller NFA) for sproget. Tilsvarende kan man vise at et sprog er kontekstfrit ved at konstruere en kontekstfri grammatik for sproget. En lignende strategi kan ikke benyttes, hvis man vil vise at et sprog *ikke* er regulært eller *ikke* kontekstfrit.

Man kan ofte klare sig med "tommelfingerregler". For eksempel gælder det at, hvis genkendelse af sproget kræver ubegrænset hukommelse, så er sproget ikke regulært. På den måde kan man indse at sproget $\{a^n b^n \mid n \in \mathbb{N}\}$ ikke er regulært, fordi man skal kunne "tælle" ubegrænset for at sikre at antallet af a'er og b'er er det samme.

Kontekstfri sprog har ubegrænset "hukommelse", og indeholder f.eks. ovenstående sprog og sproget bestående af palindromer (tegnfølger, der er ens forfra og bagfra). Men sproget af tegnfølger, der består af to ens halvdele er ikke kontekstfrit og heller ikke sproget $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. Det er svært at formulere en tommelfingerregel, der dækker disse to tilfælde. Man kan udnytte, at et kontekstfrits sprogs "hukommelse" er en stak, og argumentere at man ikke kan nøjes med en stak til at genkende disse sprog, men det er ikke altid nemt.

Der findes heldigvis egenskaber ved regulære og kontekstfri sprog, som relativt nemt kan bruges til at bevise at sprog ikke er regulære henholdsvis kontekstfri. Disse egenskaber er udtrykt ved *pumpelemmaerne* for regulære og kontekstfri sprog.

6.1 Pumpelemmaet for regulære sprog

Lad L være et regulært sprog. Da findes et $n \in \mathbb{N}$ *sådan at, hvis* $z \in L$ *og* $|z| \geq n$*, så kan z skrives som uvw hvor* $|uv| \leq n$ *og* $|v| \geq 1$ *og* $uv^i w \in L$ *for alle* $i \geq 0$

Kort sagt har alle tegnfølger over en vis længde et stykke, der kan gentages vilkårligt ofte (inklusive nul gange). Det vises nemt ved at se på automater: Et regulært sprog L kan beskrives af en NFA med n tilstande. Hvis en tegnfølge z i L har mere end n tegn, vil der være en tilstand i NFA'en, som bliver besøgt mere end én gang under analysen af z. Alle de tegn, der er passeret mellem de to besøg, kan gentages vilkårligt ofte.

Hvis man vil vise, at et sprog ikke er regulært, kan man antage at det er regulært og nå frem til en modstrid, ved at vise at nogle af de tegnfølger, der fås ved at gentage (eller udelade) et stykke i midten af en tegnfølge fra sproget, ikke selv er med i sproget. Vi kan beskrive processen således: 'Fjenden' vælger et tal n. Derefter vælger vi en tegnfølge i sproget med længde mindst n. Derefter vælger fjenden en opdeling af denne i uvw, hvor $|uv| \leq n$ og $|v| \geq 1$. Så kan vi vælge et i, og vise at $uv^i w \notin L$.

Lad os tage sproget L = $\{a^m b^m \mid m \in \mathbb{N}\}$ som eksempel. Givet et vilkårligt n, kan vi kigge på tegnfølgen $a^n b^n$, som ligger i L. Hvis vi skriver det som uvw hvor $|uv| \leq n$ må v bestå udelukkende af a'er. Ifølge pumpelemmaet (med $i = 0$), er tegnfølgen uw også med i L. Men der er flere b'er end a'er i uw, hvilket er i modstrid med vores antagelse. Derfor kan L ikke være regulært.

Dette eksempel kan vises ikke-regulært bare ved henvisning til "hukommelsesreglen", men andre sprog kan vises ikke-regulære via pumpelemmaet, selv når hukommelsesreglen ikke duer.

6.2 Pumpelemmaet for kontekstfri sprog

Lad L være et kontekstfrit sprog. Da findes et $n \in \mathbb{N}$ *sådan at, hvis* $z \in L$ *og* $|z| \geq n$*, så kan z skrives som uvwxy hvor* $|vwx| \leq n$ *og* $|vx| \geq 1$ *og* $uv^i wx^i y \in L$ *for alle* $i \geq 0$

Kort sagt vil enhver tilstrækkelig lang tegnfølge indeholde to stykker, som kan gentages vilkårlig ofte (men lige mange gange).

Lad os tage sproget L = $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ som eksempel. Givet et vilkårligt n, kan vi kigge på tegnfølgen $a^n b^n c^n$, som ligger i L. Hvis vi skriver det som $uvwxy$ med $|vwx| \leq n$ kan vwx ikke indeholde både a'er, b'er og c'er (da afstanden mellem det sidste a og det første c er n). Hvis vi gentager v og x nul gange (så vi

får *uwy*) får vi altså overskud af det/de tegn, som ikke er med i *vwx*. Altså kan *uwy* ikke indeholde lige mange *a*'er, *b*'er og *c*'er og derfor tilhører *uwy* ikke sproget *L*. Altså har vi en modstrid, og derfor er *L* ikke kontekstfrit.

Et andet eksempel er sproget, der består af tegnfølger med to ens halvdele. Tegnfølgen $a^{n+1}b^{n+1}a^{n+1}b^{n+1}$ er med i dette sprog. Vi udnytter at $|vwx| \leq n$, så *uwy* vil stadig bestå af en sekvens af *a*'er, en sekvens af *b*'er og igen en sekvens af *a*'er og en sekvens af *b*'er. Dog vil nogle af disse sekvenser indeholde færre tegn end deres modpart i den anden halvdel, så *uwy* vil ikke bestå af to ens halvdele.

References

[AHU74] Alfred V. Aho, John E. Hopcroft and Jeffrey D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley 1974

[Cho56] N. Chomsky, *Three Models for the Description of Language*, IRE Transactions on Information Theory, 2:3 (1956), pp. 118-124

[HU79] J. Hopcroft og J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley 1979.

[Mog01] T. Æ. Mogensen, *Basics of Compiler Design*, DIKU 2001.