NPDA akzeptieren genau die kontextfreien Sprachen

N = nichtdeterministisch, muß in Algorithmen durch Determinismus simuliert werden

5.6. Deterministisch kontextfreie Sprachen

$L_{DPDA} \subset L_{NPDA}$ (und zwar eine echte Teilmenge)
L_{DPDA} sind die für die Praxis relevanten Sprachen, z.B. zur Sprachanalyse (Compilerbau). NPDA haben nämlich eine Asymmetrie bei den ja/nein - Antworten. Ein NPDA ρ kann zwar sagen, ein Wort w ist in der Sprache L(ρ), nicht aber, das Wort ist nicht in der Sprache.

Also ergibt sich die Erwartung, daß die Klasse der DPDA-erkennbaren Sprachen gegenüber der Komplementbildung abgeschlossen ist.

Satz S Die Klasse der deterministisch kontextfreien Sprachen L_{DPDA} ist

a) *nicht abgeschlossen* gegen Vereinigung, Durchschnitt, Homomorphismen, Konkatenation, Sternbildung, Spiegelung
b) *abgeschlossen* gegen Komplementbildung und Durchschnitt mit regulären Sprachen

Glaubhaftmachung von a)

$L = \{ww^R \mid w \in \{a,b\}^*\}$

L ist kontextfrei, linear, aber nicht deterministisch, denn det. PDAs können nicht die Wortmitte erkennen.

(1) $L_1 = \{wcw^R \mid w \in \{a,b\}^*\}$ ist deterministisch kontextfrei, denn c ist Mittensymbol. Setze nun $h : \{a,b,c\}^* \longrightarrow \{a,b\}^*$ mit h(a) = a, h(b) = b, h(c) = ε. Dann ist $h(L_1) = L \notin L_{DPDA}$. Damit ist L_{DPDA} nicht gegen Homomorphismen abgeschlossen. Auch wenn ε in der Bildmenge von h nicht zugelassen wird, gilt diese Aussage (setze im Beispiel h(c) = aa).

(2) $L_{21} = \{a^n b^n \mid n \geq 0\} \in L_{DPDA}$
$L_{22} = \{a^n b^{2n} \mid n \geq 0\} \in L_{DPDA}$
$L_2 = L_{21} \cup L_{22}$
Nach dem Kellern der as kann ein deterministischer Automat nicht entscheiden, ob er jedes a mit einem oder zwei b gleichsetzen soll. Also gilt $L_2 \notin L_{DPDA}$

(3) $L_{31} = \{a^n b^n c^m \mid n,m \geq 0\}$
$L_{32} = \{a^m b^n c^n \mid n,m \geq 0\}$
$L_3 = L_{31} \cap L_{32} = \{a^n b^n c^n \mid n \geq 0\} \notin$ CF und somit auch nicht in L_{DPDA}.

Komplementbildung b)

Benötigt wird hierzu eine Normalform von DPDAs.

Probleme bei der Komplementbildung von DPDAs:

(1) Der DPDA „friert ein“
Es gibt zu einer Konfiguration keine Folgekonfiguration, weil δ undefiniert ist. Gilt nun $(s, v, b_0 z) \in \hat{\delta}^k(s_0, w, b)$ und $v \neq \varepsilon$ und der DPDA ρ hänge in dieser Konfiguration, dann ist w nicht in der erkannten Sprache L(ρ).

Umgekehrt würde auch der Automat für die Komplementsprache in diesem Zustand hängen.

$\Rightarrow w \notin L(\rho) \wedge \overline{w \notin L(\rho)}$

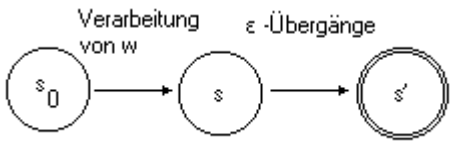
(2) Wenn ρ ein Wort w komplett abgearbeitet hat und in einem Endzustand ist, *kann* ρ noch mittels ε – Übergängen in andere Nicht - Endzustände wechseln (w ist trotzdem in L(ρ)), es kann also

$(s, \varepsilon, b_0 z) \in \hat{\delta}^k(s_0, w, b_0)$ $s \in F$ und
$(s', \varepsilon, b_0 z') \in \hat{\delta}^j(s_0, w, b_0)$ $s' \notin F$ gelten.

Der DPDA für die Komplementsprache kann ebenso rechnen, es ist
$w \in L(\rho) \wedge \overline{w \in L(\rho)}$.

Beide Probleme löst man, indem man zu einer Normalform von ρ übergeht und beim Übergang zum Komplementautomaten bestimmte Vorkehrungen trifft. *Genauer*: Zeige, daß man jeden DPDA so umbauen kann, daß er nicht einfriert. Ferner ist beim Komplementautomat nicht jeder Nicht – Endzustand von ρ ein Endzustand. Vielmehr merkt der Komplementautomat sich, daß er in einem seiner Endzustände akzeptieren könnte, wartet aber noch die ε – Übergänge von diesem Zustand ab, um zu sehen, ob ρ nicht noch akzeptiert.

ρ:



$\notin F$ $\in F \Rightarrow w \in L(\rho)$

ρ′: (s, vielleicht) → (s‘, nein)

Merke, daß ρ beim Lesen von w keinen Endzustand erreicht hat. Nach Überprüfen der ε – Übergänge wird doch ein Endzustand erreicht, also akzeptiert ρ‘ nicht.

Lemma Zu jedem deterministischen PDA $\rho = (X, \Gamma, S, \delta, s_0, b_0, F)$ gibt es einen DPDA $\rho' = (X, \Gamma', S', \delta', s'_0, b'_0, F')$ mit

(1) δ ist vollständig definiert. Dies erreicht man durch Hereinnahme eines absorbierenden Zustandes.

(2) $\forall s, s' \in S'; a \in X \cup \{\varepsilon\}; c \in \Gamma'; v \in \Gamma'^{*}$ gilt: Aus $\delta'(s, a, c) = (s', v)$ folgt $v = \varepsilon \vee v = c \vee |v| = 2$.
Dies erreicht man durch Einführen neuer Zustände. Will man mehr als zwei Zeichen auf den Keller schreiben, muß man Zwischenzustände definieren.

(3) Für alle w aus X^* gilt: ρ‘ terminiert bei Eingabe von w und ρ‘ liest das gesamte Wort w ein, das heißt:
$\forall w \in X^* \exists k \in IN_0, v \in \Gamma'^{*}, s \in S'$ mit

(i) $(s, \varepsilon, v) = \hat{\delta}'^{k}(\alpha(w))$ (w vollständig gelesen)

(ii) $\hat{\delta}'^{k+1}(\alpha(w))$ (wenn w vollständig gelesen ist, ist die Berechnung durch den Automaten zuende)

Schwierig [Hopcroft/Ullmann]: Zeige, daß es zu keinem Zustand s und zu keinem Kellersymbol $c \in \Gamma$ eine unendliche ε – Übergangsfolge gibt. Dies ist für DPDA entscheidbar

(4) $L(\rho) = L(\rho')$

Einschub: Determinismus $\longleftrightarrow$ Determiniertheit

Determinismus: Eindeutige Festlegung des nächsten Rechenschrittes (aber möglicherweise ist die Entscheidung, ihn auszuführen, willkürlich)

Determiniertheit: Funktionale Abhängigkeit zwischen Eingabe und Ausgabe

Beispiele:

	determiniert	**nicht determiniert**
deterministisch	beliebige det. Programme	write(random(0,1));
nichtdeterministisch	QuickSort	if random(0,1) then write(a) else write (b);