# Automaty a gramatiky

10

Roman Barták, KTIML

bartak@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

## Chomského normální forma

Podívejme se nyní na derivační stromy.

Jak odhadnout výšku stromu podle délky slova?

*Definice:* Říkáme, že gramatika je v *Chomského normální formě* (tvaru), jestliže všechna pravidla mají tvar:
$X \rightarrow YZ$ nebo $X \rightarrow a$, kde $a \in V_T$, $X,Y,Z \in V_N$.

K čemu je tento tvar dobrý?

Derivační strom je (skoro) binární.

Je-li maximální cesta délky k, potom terminální slovo $\leq 2^{k-1}$.

Pumping lemma pro bezkontextové jazyky

*Věta:* Ka každému bezkontextovému jazyku L existuje bezkontextová gramatika G v Chomského normální formě taková, že $L(G) = L - \{\lambda\}$.



## Převod na Chomského NF

Povoleno pouze $X \rightarrow YZ$ nebo $X \rightarrow a$, kde $a \in V_T$, $X,Y,Z \in V_N$.

*Víme:*

pravidla $A \rightarrow B$ lze odstranit (viz regulární gramatiky)

pravidla $A \rightarrow \lambda$ lze odstranit (maximálně přijdeme o $\lambda$)

*Dále:*

pravidla tvaru $X \rightarrow a$, kde $a \in V_T$ jsou v pořádku

zbývají pravidla tvaru $X \rightarrow B_1 \ldots B_k$, $k \geq 2$, $B_i \in V_N \cup V_T$

vytvoříme pravidlo $X \rightarrow C_1 \ldots C_k$, kde:

$C_i = B_i$ je-li $B_i \in V_N$

$= B'_i$ je-li $B_i \in V_T$ ($B'_i$ je nový neterminál)
+ přidáme pravidla $B'_i \rightarrow B_i$

pravidlo $X \rightarrow C_1 \ldots C_k$, $k \geq 3$ nahradíme pravidly:

$X \rightarrow C_1D_1$, $D_1 \rightarrow C_2D_2$, …, $D_{k-2} \rightarrow C_{k-1}C_k$,
$D_i$ jsou nové neterminály



## Příklad převodu na Chomského NF

*Původní gramatika*

A → B | C
B → 0B1 | 01
C → D | E
D → 1D0 | 1
E → 0E | 0

*Po odstranění X → Y*

A → 0B1 | 01 | 1D0 | 1 | 0E | 0
B → 0B1 | 01
~~C → 1D0 | 1 | 0E | 0~~
D → 1D0 | 1
E → 0E | 0

*Po nahrazení terminálů*

A → NBJ | NJ | JDN | 1 | NE | 0
B → NBJ | NJ
D → JDN | 1
E → NE | 0
N → 0
J → 1

*Chomského normální forma*

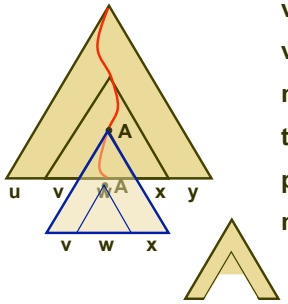
$A \rightarrow NA_1 \mid NJ \mid JA_2 \mid 1 \mid NE \mid 0$
$A_1 \rightarrow BJ$
$A_2 \rightarrow DN$
$B \rightarrow NB_1 \mid NJ$
$B_1 \rightarrow BJ$
$D \rightarrow JD_1 \mid 1$
$D_1 \rightarrow DN$
$E \rightarrow NE \mid 0$
$N \rightarrow 0$
$J \rightarrow 1$

## Pumping lemma pro bezkontexové jazyky

*Lemma o vkládání:* Nechť L je bezkontextový jazyk. Potom existují přirozená čísla p, q taková, že každé slovo z∈L, |z|>p lze psát ve tvaru z =uvwxy a platí:

1) |vwx|≤q (pumpovací část není moc dlouhá)
2) buď v ≠λ nebo x≠λ (lze psát vx≠λ )
3) ∀i≥0 $uv^iwx^iy \in L$

*Idea důkazu:*



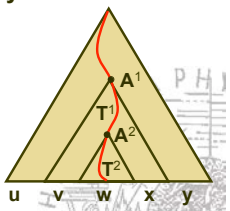

vezmeme derivační strom pro slovo z
v derivačním stromu najdeme nejdelší cestu
na této cestě najdeme dva stejné neterminály
tyto neterminály určí dva podstromy
podstromy definují rozklad slova
nyní můžeme větší podstrom posunout (i>1)
nebo nahradit menším podstromem (i=0)



## Důkaz lemma o vkládání pro BKJ

|z|>p : z =uvwxy, |vwx|≤q, vx≠λ, ∀i≥0 $uv^iwx^iy \in L$

vezmeme gramatiku v Chomského NF (slova λ nevadí)
$|V_N|=k$, položme $p = 2^{k-1}$, $q = 2^k$
$|z| > 2^{k-1}$, v libovolném derivačním stromu je cesta délky >k
na této (nejdelší) cestě musí ležet dva stejné neterminály a terminál *t*
vezmeme dvojici $A^1$, $A^2$ nejblíže k *t* (určuje podstromy $T^1$ a $T^2$)
cesta z $A^1$ do *t* je nejdelší v podstromu $T^1$ a má délku maximálně k+1
tedy slovo dané stromem $T^1$ není delší než $2^k$ (|vwx|≤q))
z $A^1$ vedou dvě cesty (ChNF), jedna do $T^2$ druhá do zbytku vx
ChNF je nevypouštějící, tedy vx≠λ
derivace slova ($A^1 \Rightarrow^* vA^2x$, $A^2 \Rightarrow^* w$)
$S \Rightarrow^* uA^1y \Rightarrow^* uvA^2xy \Rightarrow^* uvwxy$
posuneme-li $A^2$ do $A^1$ (i=0)
$S \Rightarrow^* uA^2y \Rightarrow^* uwy$
posuneme-li $A^1$ do $A^2$ (i=2,…)
$S \Rightarrow^* uA^1y \Rightarrow^* uvA^1xy \Rightarrow^* uvvA^2xxy \Rightarrow^* uvvwxxy$






## Použití lemma o vkládání

Jak ukázat, že daný jazyk není bezkontextový?

*Příklad 1:* L = $\{a^ib^ic^i \mid i \geq 1\}$ není bezkontextový jazyk sporem

zvolme n = max(p,q), potom $|a^nb^nc^n| > p$
pumpovací slovo není delší než q
tj. vždy lze pumpovat maximálně dva různé symboly
poruší se rovnost počtu symbolů - SPOR

*Příklad 2:* L = $\{a^ib^jc^k \mid 0 \leq i \leq j \leq k\}$ není bezkontextový jazyk sporem

zvolme n = max(p,q), potom $|a^nb^nc^n| > p$
pumpovací slovo není delší než q
tj. vždy lze pumpovat maximálně dva různé symboly
pokud pumpujeme *a* (případně *b*), pumpujeme nahoru
pokud pumpujeme *c* (případně *b*), pumpujeme dolu
potom i>k (a↑, c↓) nebo j>k (b↑, c↓) nebo i>j (a↑, b↓) - SPOR



## Kdy lemma o vkládání nezabere

Pozor! Lemma o vkládání je pouze implikace!

BKJ ⇒ lze pumpovat (nutná podmínka bezkontextovosti)
nejedná se o podmínku postačující

*Příklad:*

L = $\{a^ib^jc^kd^l \mid i=0 \vee j=k=l\}$ není bezkontextový jazyk
přesto lze pumpovat
i=0: $b^jc^kd^l$ lze pumpovat v libovolném písmenu
i>0: $a^ib^nc^nd^n$ lze pumpovat v části obsahující *a*

Jak na to?

- zobecnění pumping lemmatu (Ogdenovo lemma)
  pumpování vyznačených symbolů
- uzávěrové vlastnosti

## Nekonečnost bezkontextových jazyků

**Pro každý BKJ L existují přirozená čísla m, n taková, že:**
**L je nekonečný ⇔ ∃z∈L m<|z|≤n.**

***Důkaz:***

z lemmatu o vkládání máme *p* a *q*, položme: m = p, n = p+q

„⇐“

p<|z|, tedy *z* lze pumpovat ⇒ jazyk je nekonečný

„⇒ “

jazyk je nekonečný ⇒ ∃z∈L p=m < |z|

vezmeme nejkratší takové *z* a potom |z| ≤ n=p+q

*sporem:* nechť p+q < |z|, lze pumpovat dolu, tj. |z‘| < |z|
odstraňujeme část o max. velikosti q, tedy p < |z‘| - SPOR

***Rychlejší algoritmus:***

vezmeme redukovanou gramatiku G v ChNF tž. L=L(G)

uděláme orientovaný graf

vrcholy = neterminály, hrany = {(A,B), (A,C) pro (A→BC) ∈ P(G)}

hledáme orientovaný cyklus (existuje ⇒ jazyk je nekonečný)



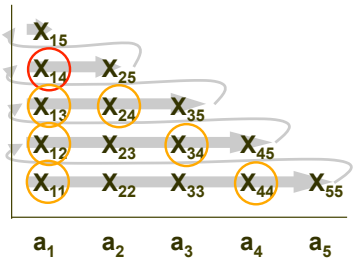
## Algoritmus CYK (Cocke, Younger, Kasami)

**Jak zjistíme příslušnost slova $a_1a_2…a_n$ do BKJ?**

vezmeme gramatiku v ChNF

a podle ní vyplníme následující tabulku (dynamické programování)

- myšlenka: $X_{i,j} = \{A \mid A \Rightarrow^* a_ia_{i+1}…a_j\}$
- začneme: $X_{i,i} = \{A \mid (A \rightarrow a_i) \in P\}$
- pokračujeme: $X_{i,j} = \{A \mid \exists k: i \le k < j \;\; B \in X_{i,k} \wedge C \in X_{k+1,j} \wedge (A \rightarrow BC) \in P\}$
- pokud $S \in X_{1,n}$, potom $a_1a_2…a_n$ patří do jazyka

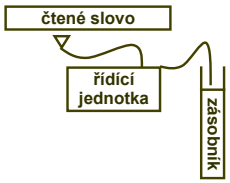



S → AB | BC
A → BA | a
B → CC | b
C → AB | a

| | | | | |
|---|---|---|---|---|
| {S,A,C} | | | | |
| - | {S,A,C} | | | |
| - | {B} | {B} | | |
| {S,A} | {B} | {S,C} | {S,A} | |
| {B} | {A,C} | {A,C} | {B} | {A,C} |
| b | a | a | b | a |



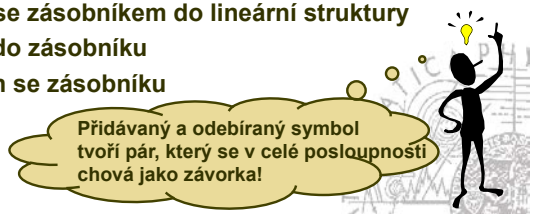
## Jak charakterizovat bezkontextové jazyky?




Pokud do zásobníku pouze přidáváme
potom si stačí pamatovat poslední symbol.
Stačí konečná paměť → konečný automat.

Potřebujeme ze zásobníku také odebírat (čtení symbolu)!
Takový proces nelze zaznamenat v konečné struktuře.
Přidávání a odebírání ale není zcela libovolné
jedná se o zásobník tj. LIFO (last-in first-out) strukturu
Roztáhněme si výpočet se zásobníkem do lineární struktury
X - symbol přidán do zásobníku
$X^{-1}$ - symbol odebrán se zásobníku

$Z\,Z^{-1}\,B\,A\,A^{-1}\,C\,C^{-1}\,B^{-1}$






## Dyckovy jazyky

**Dyckův jazyk $D_n$ je definován nad abecedou**
**$Z_n = \{a_1,a‘_1,…,a_n,a‘_n\}$ následující gramatikou:**
**$S \rightarrow \lambda \mid SS \mid a_1Sa‘_1 \mid … \mid a_nSa‘_n$.**

***Úvodní poznámky:***

- Jedná se zřejmě o jazyk bezkontextový.
- Dyckův jazyk $D_n$ popisuje správně uzávorkované výrazy s *n* druhy závorek.
- Tímto jazykem lze popisovat výpočty libovolného zásobníkového automatu.
- Pomocí Dyckova jazyka lze popsat libovolný bezkontextový jazyk.

$L = h(D \cap R)$

*Regulární jazyk*
popisuje všechny kroky výpočtu

*Dyckův jazyk*
vybírá pouze korektní výpočty

*Homomorfismus*
čistí pomocné symboly

## Dyckovy jazyky a bezkontextové jazyky

**Pro každý bezkontextový jazyk L existuje regulární jazyk R tak, že L=h(D∩R) pro vhodný Dyckův jazyk D a homomorfismus h.**

*Důkaz:*

máme zásobníkový automat přijímající L prázdným zásobníkem

BÚNO instrukce tvaru $\delta(q,a,Z) \ni (p,w)$, $|w| \leq 2$

nechť R‘ obsahuje všechny výrazy

$q^{-1}aa^{-1}Z^{-1}BAp$ pro instrukci $\delta(q,a,Z) \ni (p,AB)$

podobně pro instrukce $\delta(q,a,Z) \ni (p,A)$ a $\delta(q,a,Z) \ni (p,\lambda)$

je-li $a=\lambda$, potom dvojici $aa^{-1}$ nezařazujeme

definujme R takto $Z_0q_0R'^*Q^{-1}$

Dyckův jazyk je definován nad abecedou $X \cup X^{-1} \cup Q \cup Q^{-1} \cup Y \cup Y^{-1}$

$D \cap Z_0q_0R'^*Q^{-1}$ popisuje korektní výpočty

$Z_0\ q_0\ q_0^{-1}\ a\ a^{-1}\ Z_0^{-1}\ B\ A\ p\ \ p^{-1}\ b\ b^{-1}\ A^{-1}\ q\ \ q^{-1}\ c\ c^{-1}\ B^{-1}\ r\ \ r^{-1}$

homomorfismus h vydělí přečtené slovo, tj.

$h(a) = a$ pro vstupní (čtené) symboly

$h(y) = \lambda$ pro ostatní symboly



## Průnik bezkontextových jazyků

**Bezkontextové jazyky nejsou uzavřené na průnik.**

***Důkaz*:**

**stačí najít dva BKJ, jejichž průnik není BKJ**

$L_1 = \{a^ib^ic^j \mid 0 \leq i,j\}$ $\{S \to AC,\ A \to aAb \mid \lambda,\ C \to cC \mid \lambda\}$

$L_2 = \{a^ib^jc^j \mid 0 \leq i,j\}$ $\{S \to AB,\ A \to aA \mid \lambda,\ B \to bBc \mid \lambda\}$

$L_1 \cap L_2 = \{a^ib^ic^i \mid 0 \leq i\}$ není BKJ (víme z pumping lemmatu)

***Pozorování:***

paralelní běh dvou zásobníkových automatů

řídící jednotky umíme spojit (viz konečné automaty)

čtení umíme spojit (jeden automat může čekat)

bohužel dva zásobníky nelze obecně spojit do jednoho

dva zásobníky = Turingův stroj

= rekurzivně spočetné jazyky



## Průnik bezkontextového a regulárního jazyka

**(Deterministické) bezkontextové jazyky jsou uzavřené na průnik s regulárním jazykem.**

***Důkaz:***

**zásobníkový a konečný automat můžeme spojit**

konečný automat $A_1 = (Q_1,X,\delta_1,q_1,F_1)$

zásobníkový automat (přijímání stavem) $M_2 = (Q_2,X,Y,\delta_2,q_2,Z_0,F_2)$

nový automat $M = (Q_1 \times Q_2, X, Y, \delta, (q_1,q_2), Z_0, F_1 \times F_2)$

$((p',q'), u) \in \delta((p,q),a,Z)$ právě když

- $a \neq \lambda$: $p' \in \delta_1(p,a) \wedge (q',u) \in \delta_2(q,a,Z)$ … automaty čtou vstup
- $a = \lambda$: $(q',u) \in \delta_2(q,\lambda,Z)$ … ZA mění zásobník
  $p'=p$ … KA stojí

**zřejmě $L(M) = L(A_1) \cap L(M_2)$**

paralelní běh automatů



## Použití uzavřenosti průniku BKJ a RJ

**$L = \{a^ib^jc^kd^l \mid i=0 \vee j=k=l\}$ není bezkontextový jazyk**

***SPOREM:***

**nechť L je bezkontextový jazyk**

**$L_1 = \{ab^ic^jd^k \mid 0 \leq i,j,k\}$ je regulární jazyk**

$S \to aB,\ B \to bB \mid C,\ C \to cC \mid D,\ D \to dD \mid \lambda$

**$L \cap L_1 = \{ab^ic^id^i \mid 0 \leq i\}$ není bezkontextový jazyk - SPOR**

**L je kontextový jazyk**

$S \to B' \mid aA$

$B' \to bB' \mid C',\ C' \to cC' \mid D',\ D' \to dD' \mid \lambda$

$A \to aA \mid P$

$P \to bPCD \mid \lambda$

$DC \to CD$ $\{DC \to XC, XC \to XY, XY \to CY, CY \to CD\}$

$bC \to bc,\ cC \to cc,\ cD \to cd,\ dD \to dd$

## Sjednocení a doplněk BKJ

**Bezkontextové jazyky jsou uzavřené na sjednocení.**

použijeme gramatiky (pro ZA nedeterministický rozeskok na startu)

$L_1=L(G_1)$ $G_1 = (V_{N1},V_{T1},S_1,P_1)$

$L_2=L(G_2)$ $G_2 = (V_{N2},V_{T2},S_2,P_2)$

můžeme předpokládat, že $V_{N1} \cap V_{N2} = \varnothing$ (jinak přejmenuj)

uděláme gramatiku:

$G = (V_{N1} \cup V_{N2} \cup \{S\}, V_{T1} \cup V_{T2}, S, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\})$

zřejmě $L(G) = L(G_1) \cup L(G_2)$

„počítá“ jedna nebo druhá gramatika

**Bezkontextové jazyky nejsou uzavřené na doplněk.**

sporem podle de Morganových pravidel

$L_1 \cap L_2 = -(-L_1 \cup -L_2)$

bezkontextové jazyky nejsou uzavřené na průnik - SPOR

v ZA nestačí prohodit koncové a nekoncové stavy!



## Zrcadlový obraz, zřetězení a iterace

**Bezkontextové jazyky jsou uzavřené na zrcadlový obraz, zřetězení, iteraci a pozitivní iteraci.**

1) zrcadlový obraz $L^R = \{w^R \mid w \in L\}$

G: $X \to w^R$ (obrátíme pravou stranu pravidel)

ZA: slova do zásobníku dáváme „opačně“

2) zřetězení $L_1 . L_2$

G: $S \to S_1S_2$ (nejprve generujeme první slovo, potom druhé)

ZA: nejprve běží první automat, potom druhý

3) iterace $L^* = \cup_{i \geq 0} L^i$

G: $S' \to SS' \mid \lambda$ (opakovaně spouštíme generování slov z jazyka)

ZA: na konci výpočtu můžeme restartovat + prázdné slovo

4) pozitivní iterace $L^+ = \cup_{i \geq 1} L^i$

G: $S' \to SS' \mid S$ (opakovaně spouštíme generování slov z jazyka)

ZA: na konci výpočtu můžeme restartovat



## Substituce a homomorfismus

**Substituce** $\sigma$ **převádí slova na jazyky**

$\sigma(\lambda) = \{\lambda\}$, $\sigma(x) =$ jazyk

$\sigma(uv) = \sigma(u).\ \sigma(v)$ $\sigma: X^* \to P(Y^*)$

$\sigma(L) = \cup_{w \in L} \sigma(w)$

Třída *T* jazyků je **uzavřena na substituci**, když:

$\forall a \in X\ \sigma(a) \in T \wedge L \in T \Rightarrow \sigma(L) \in T$

**Homomorfismus** **převádí slova na slova**

$h(\lambda) = \lambda$, $h(x) =$ slovo

$h(uv) = h(u)\ .\ h(v)$ $h: X^* \to Y^*$

$h(L) = \{h(w) \mid w \in L\}$

**Inverzní homomorfismus** **převádí slova zpět**

$h^{-1}(L) = \{ w \mid h(w) \in L\}$



## Uzavřenost BKJ na substituci

**Bezkontextové jazyky jsou uzavřeny na substituci.**

intuitivně: listy v derivačním stromu generují další stromy

formálně:

máme bezkontextový jazyk $L_0$, tj. gramatiku $G_0 = (V_{N0},V_{T0},S_0,P_0)$,

pro každý terminál $a_i$ z $V_{T0}$, $\sigma(a_i)$ je bezkontextový jazyk - gramatika $G_i$

předpokládejme, že množiny neterminálů jsou navzájem disjunktní a že žádný terminál není v jiné gramatice neterminálem

definujme $G = (\cup_{i \geq 0} V_{Ni}, \cup_{i \geq 1} V_{Ti}, S_0, P)$, kde $P = \cup_{i \geq 1} P_i \cup P'$ a $P' = P_0$, kde každý terminál $a_i$ je nahrazen $S_i$

zřejmě: $L(G) = \sigma(L_0)$

*Příklad:*

$L_0 = \{a^ib^j \mid 0 \leq i \leq j\}$ $S_0 \to aS_0b \mid S_0b \mid \lambda$

$\sigma(a) = L_1 = \{c^id^i \mid 0 \leq i\}$ $S_1 \to cS_1d \mid \lambda$

$\sigma(b) = L_2 = \{c^i \mid 0 \leq i\}$ $S_2 \to cS_2 \mid \lambda$

$\sigma(L_0)$: $S_0 \to S_1S_0S_2 \mid S_0S_2 \mid \lambda$, $S_1 \to cS_1d \mid \lambda$, $S_2 \to cS_2 \mid \lambda$

**Bezkontextové jazyky jsou uzavřeny na homomorfismus.**

přímý důsledek předchozí věty (terminál nahradíme slovem)

## Inverzní homomorfismus

**Bezkontextové jazyky jsou uzavřeny na inverzní homomorfismus.**

$h^{-1}(L) = \{ w \mid h(w) \in L \}$ - máme zásobníkový automat M pro L a čteme w

**idea:**

- přečteme písmeno x a do vnitřního bufferu dáme h(x)
- simulujeme výpočet M, kdy vstup bereme z bufferu
- po vyprázdnění bufferu načteme další písmeno ze vstupu
- slovo je přijato, když je buffer prázdný a M je v koncovém stavu

**formálně:**

buffer je konečný, můžeme ho tedy modelovat ve stavu

pro L máme $M = (Q,X,Y,\delta,q_0,Z_0,F)$ (přijímání koncovým stavem)

$h: A^* \rightarrow X^*$

definujme $M' = (Q', A, Y, \delta', [q_0,\lambda], Z_0, F \times \{\lambda\})$, kde

$Q' = \{[q,u] \mid q \in Q, u \in X^*, \exists a \in A\ \exists v \in X^* : h(a) = vu\}$, u je buffer

$\delta'([q,u], \lambda, Z) = \{ ([p,u],\gamma) \mid (p,\gamma) \in \delta(q, \lambda, Z) \}$

$\cup \{ ([p,v],\gamma) \mid (p,\gamma) \in \delta(q, b, Z) \}$ … u=bv (čte buffer)

$\delta'([q, \lambda], a, Z) = \{ ([q,h(a)], Z) \}$ … naplňuje buffer



## Kvocienty s regulárním jazykem

**Bezkontextové jazyky jsou uzavřené na levý (pravý) kvocient s regulárním jazykem.**

$R\backslash L = \{ w \mid \exists u \in R\ uw \in L \}$, $L/R = \{ u \mid \exists w \in R\ uw \in L \}$

**idea:**

- ZA běží na prázdno (nečte vstup) paralelně s KA
- je-li KA v koncovém stavu, můžeme začít číst vstup

**formálně:**

konečný automat $A_1 = (Q_1,X, \delta_1, q_1, F_1)$

zásobníkový automat $M_2 = (Q_2,X,Y,\delta_2,q_2,Z_0,F_2)$ (přijímání koncovým stavem)

definujme nový automat $M = (Q', X, Y, \delta, (q_1,q_2), Z_0, F_2)$, kde

$Q' = (Q_1 \times Q_2) \cup Q_2$ (dvojice stavů pro paralelní běh ZA a KA)

$\delta'((p,q), \lambda, Z) = \{ ((p',q'),u) \mid \exists a \in X\ p' \in \delta_1(p, a) \wedge (q',u) \in \delta_2(q, a, Z) \}$

$\cup \{ ((p,q'),u) \mid (q',u) \in \delta_2(q, \lambda, Z) \}$

$\cup \{ (q,Z) \mid p \in F_1 \}$

$\delta'(q, a, Z) = \delta_2(q, a, Z)$ $a \in X \cup \{\lambda\}$, $q \in Q_2$

zřejmě $L(M) = L(A_1) \backslash L(M_2)$



## Uzávěrové vlastnosti deterministických BKJ

Rozumné programovací jazyky jsou deterministické BKJ.

Deterministické bezkontextové jazyky:

- nejsou uzavřené na průnik,
- jsou uzavřené na průnik s regulárním jazykem,
- jsou uzavřené na inverzní homomorfismus.

**Doplněk deterministického BKJ je opět deterministický BKJ!**

„prohodíme koncové a nekoncové stavy“

*potíže:*

- nemusí přečíst celé vstupní slovo
  - krok není definován (např. vyprázdnění zásobníku)
    snadno ošetříme „podložkou“ na zásobníku
  - cyklus (zásobník roste, zásobník pulsuje)
    odhalíme pomocí čítače
- po přečtení slova prochází koncové a nekoncové stavy
  stačí si pamatovat, zda prošel koncovým stavem



## Uzávěrové vlastnosti DBKJ v praxi

**DBKJ nejsou uzavřené na sjednocení (BKJ ano)!**

***Příklad:***

$L = \{a^i b^j c^k \mid i \neq j \vee j \neq k \vee i \neq k\}$ je BKJ, ale není DBKJ

sporem: nechť L je DBKJ

potom -L (doplněk) je DBKJ

$-L \cap a^*b^*c^* = \{a^i b^j c^k \mid i=j=k\}$ je DBKJ - SPOR

**DBKJ nejsou uzavřené na homomorfismus (BKJ ano)!**

***Příklad:***

$L_1 = \{a^i b^j c^k \mid i=j\}$ je DBKJ

$L_2 = \{a^i b^j c^k \mid j=k\}$ je DBKJ

$0L_1 \cup 1L_2$ je DBKJ, $1L_1 \cup 1L_2$ není DBKJ

položme h(0) = 1

h(x) = x pro ostatní symboly

$h(0L_1 \cup 1L_2) = 1L_1 \cup 1L_2$