1) Sestrojte redukovany automat nad abecedou {a,b} prijimajici slova koncici nejmene dvema znaky 'a' predchazenymi znakem 'b'. Prevedete na deterministicky automat prijimajici obracena slova. Co to je redukovany automat? Co to je ekvivalence stavu? Dokazte vsechna tvrzeni a vety, ktere jste pouzili pri tvorbe automatu.

- zrejme se mini jazyk vyhovujici regularnimu vyrazu (a+b)*ba*aa, paklize ano, pak sestrojeni NKA je trivialni:

```
       a        b
->A    A X      A B
  B    B C      X
  C    D        X
<-D    X        X
  X    X        X
```

A je vstup, D vystup, X je garbage stav

- deterministicky automat prijimajici jazyk L^R (jazyk obracenych slov) je taktez trivialni:

```
       a        b
->A    B        X
  B    C        X
  C    C        D
<-D    D        D
  X    X        X
```

A je vstup, D vystup, X je garbage stav
-pak by bylo asi vhodné předvest důkaz kleeneovy věty

2) Je dán jazyk $L = \{0^n 1^m \mid 0 \le n \le m \le 2n\}$. Zkonstruuj bezkontextovou gramatiku *G*, tž. $L = L(G)$. Definuj regulární gramatiku, bezkontextovou gramatiku. Dokaž, že pro daný jazyk nejde zkonstruovat regulární gramatika. Dokaž věty, které jsi použil. (Tzn. Nerodovu větu.)

BKG:
S->0S11 | 0S1 | λ

Důkaz že není regulární (Nerodovou větou):
Předpokládejme, že jazyk je regulární
=> existuje pravá kongruence konečného indexu m,
L je sjednocením tříd
vezmeme slova 0, 00, …, 0^{m+1}
dvě slova padnou do stejné třídy (krabičkový princip)

ex. i<j	$0^i \sim 0^j$
přidejme 1^i	$0^i 1^i \sim 0^j 1^i$ (pravá kongruence)
spor	$0^i 1^i \in L$ & $0^j 1^i \notin L$

3) Zaradit do Chomskeho chierarchie a dokazat ze tomu tak opravdu je (pripadne vytvorit gramatiky) nasledujici: $L = \{ u.u^R.u \mid$ u patri do {0,1}*}

Tu mas kontextovu gramatiku aj s odtrasovanim pre slovo u=1010. Snad pomoze. Este je nutne dodat, ze pravidla typu AB -> CD niesu kontextove a treba ich previest (vid slajdy). Dokaz je podobny tomu ako dokaz pre slovo ww tunak na fore.

Vt = {0,1}

Vn = {S,H,D,T,C,J,i,o,P,N} U { term. pri prevode AB -> CD}

Reprezentacia:
1 = T,J,i,P
0 = D,C,o,N

```
S -> 0H0D
  -> 1H1T
  -> 111
  -> 000
  -> λ

// 1H1T

H -> 00C
  -> 11J
  -> 0H0o
  -> 1H1i

// 10H0o1T
// 101H1i0o1T
// 10100C1i0o1T

o0 -> 0o
o1 -> 1o
i0 -> 0i
i1 -> 1i

// 10100C10io1T
// 10100C10i1oT
// 10100C101ioT

C0 -> 0C
C1 -> 1C
J0 -> 0J
J1 -> 1J

// 101001C01ioT
// 1010010C1ioT
// 10100101CioT

Co -> CN
Ci -> JN
Jo -> CP
Ji -> JP

No -> oN
Ni -> iN
Po -> oP
Pi -> iP

// 10100101JNoT
// 10100101JoNT
// 10100101CPNT
// 10100101CPNT

ND -> D0
NT -> T0
PD -> D1
PT -> T1

// 10100101CPT0
// 10100101CT10

CD -> 00
CT -> 10
JD -> 01
```

```
JT -> 11

// 101001011010
```

4) Zaradit do Chomskeho chierarchie a dokazat ze tomu tak opravdu je (pripadne vytvorit gramatiky) nasledujici: L = { w | w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek}

Kontextova gramatika :
S -> S' | λ
S' -> ABCS' | ABC
AB -> BA (resp. AB -> AY, AY -> XY, XY -> XA, XA -> BA)
BA -> AB (...)
AC -> CA (...)
CA -> AC (...)
CB -> BC (...)
BC -> CB (...)
A->a
B->b
C->c

Není BKJ:

Staci to rovno dokazat cez $a^n b^n c^n$ patri L => pumping lemma pro BKJ:

Dvojku vylucuje pripad $a^n b^n c^n$ kde n=max(p,q),
Predpoklad: je BJK.
Ak je BJK mozno pumpovat....
1)
vwx cely v bloku a^n alebo b^n alebo c^n
Zapumpujeme hore.
Dostanes :
$a^k b^n c^n$ alebo
$a^n b^k c^n$ alebo
$a^n b^n c^k$
kde k>n potom nemozno zapisat v tvare (w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek)
Nemozno pumpovat!

2)
vwx padne na rozhranie ab alebo bc tak, ze w je presne na rozhrani :
$a^l b^k c^n$ alebo
$a^n b^l c^k$
Zapumpujeme hore.
kde l>n alebo(OR) k>n potom nemozno zapisat v tvare (w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek)
Nemozno pumpovat!

3a) x = λ
vwx padne na rozhranie ab alebo bc tak, ze v je presne na rozhrani :
$a^l b^k c^n$ alebo
$a^n b^l c^k$
Zapunpujeme dole.
kde l<n a(AND) k<n potom nemozno zapisat v tvare (w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek)
Nemozno pumpovat!

3b) $v = \lambda$
vwx padne na rozhranie ab alebo bc tak, ze x je presne na rozhrani :
Rovnaky pripad ako predchadzajuci.
Nemozno pumpovat!

3c-1) $v,x \neq \lambda$
vwx padne na rozhranie ab alebo bc tak, ze **v** je presne na rozhrani:
$a^l b^k c^n$ alebo
$a^n b^l c^k$
Zapunpujeme dole.
kde **k<=l**<n potom nemozno zapisat v tvare (w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek)
Nemozno pumpovat!

3c-2) $v,x \neq \lambda$
vwx padne na rozhranie ab alebo bc tak, ze **x** je presne na rozhrani:
$a^l b^k c^n$ alebo
$a^n b^l c^k$
Zapunpujeme dole.
kde **l<=k**<n potom nemozno zapisat v tvare (w patri do {a,b,c}* a pocet acek = pocet becek = pocet cecek)
Nemozno pumpovat!

Nieje bezkontextovy pretoze nemozno punpovat.

5) L = {$a^i b^j c^k$ | 0<=i<j<k} - napisat kontextovu gramatiku a dokazat ze to nie je bezkontextove

- $a^i b^j c^k$ 0<=i<j<k je kontextovy
(BKJ vylucuje $a^n b^{(n+1)} c^{(n+2)}$, kde n=max(p,q)) kde p a q jsou cisla z pumping lemmatu pro BKJ

- kontextova gramatika:

```
S -> aBBCCC | aXBBCCC | bBCCC | bCC
X -> aBC
B -> BBC
C -> CC
aB -> ab
bB -> bb
bC -> bc
cC -> cc
CB -> CY
CY -> BY
BY -> BC
```

8) Zařadit jazyk {ww|w patri do {0,1}*} do Chomského hierarchie (= kontextový jazyk), dokázat, že nepatří níž, a dokázat větu, kterou jsme použili (bezkontextové pumping lemma).

Myslenka je nasledujici:

1) vygenerovat slovo w . w^r (cili slovo w a potom obracene slovo w),
2) obratit w^r, cimz ziskame $w^{r\text{^}r}$, coz je opet puvodni w a dohromady tak dostaneme w.w; toho dosahneme tak, ze z uplneho konce ww^r vzdy vysleme znak smerem doprostred, kde se zastavi.

Na uvod poznamka: potrebujeme vice zpusobu, jak zakodovat nulu a jednicku, takze nejdriv napisu malou tabulku, jaky neterminal vlastne "reperezentuje" nulu a jaky jednicku:

1 je X, J, P, C

0 je Y, N, Q, D

Jak to tedy udelat, aby vysledna gramatika byla nevypoustejici (a tim padem prevoditelna na kontextovou)? Slovo $w.w^r$ vygenerujeme nasledovne:

S -> 00 | 11 | 1AX | 0AY
A -> 1P | 0Q | 0AN | 1AJ

Tato gramatika nam vygeneruje slova nasledujiciho tvaru (priklad):

1001101PNJJNNX

V tomto slove je nejdrive retezec w zakodovany primo terminalovymi symboly:

1001101

No a potom je tam obraceny retezec zakodovany neterminalnimi symboly:

PNJJNNX

Ovsem potrebujeme nejak identifikovat, kde je konec a kde zacatek (to budeme potrebovat za chvili, abychom umeli poslat "posla" a zabit ho), takze nepouzijeme ke kodovani 0/1 jen neterminaly N/J, ale jeste P/Q (P koduje jednicku na zacatku w^r, Q koduje nulu na zacatku w^r) a X/Y (X koduje jednicku na konci w^r, Y nulu...)

Ted provedeme to, ze z praveho konce vzdycky vysleme "posla" nesouciho bud jednicku nebo nulu - samozrejme podle toho, jestli na tom pravem kraji byla jednicka nebo nula. Dostavame tedy pravidla (C koduje "posla nesouciho jednicku", D koduje "posla nesouciho nulu"):

NX -> CY
NY -> DY
JX -> CX
JY -> DX

Mozna bych to jeste okomentoval. Napriklad pravidlo:

NX -> CY

rika, ze kdyz mame na konci X (tedy zakodovanou jednicku), tak mame vyslat posla C (tedy posla "nesouciho jednicku"); pred tou posledni jednickou se nachazela nula a jakmile posel zacne uhanet smerem vlevo, tak se ta nula stane znakem, ktery je nejvice napravo, tzn. uz nebude zakodovana symbolem N, ale symbolem Y (Y koduje nulu, ktera je zcela vpravo, N koduje nulu, ktera je nekde uvnitr w^r).

Ostatni pravidla je mozne oduvodnit podobne. Jakmile jsme posla vyslali, chceme zajistit, aby nam hopsal smerem vlevo (ke stredu). To je easy:

NC -> CN
JC -> CJ
ND -> DN
JD -> DJ

Zajistili jsme tedy, aby se posel (tzn. terminal C nebo D) prohnal vlevo pres celou druhou cast naseho slova. Kdyz posel dorazi do "prostredka" (on to samozrejme prostredek bude jen na zacatku, jakmile zacneme popsanym zpusobem "prelevat" neterminaly z konce do konce prvni casti, uz to skutecny prostredek nebude), zabijeme ho a prepiseme na 0/1. Pravidla, ktera zbiji posly:

PC -> 1P

PD -> 0P
QC -> 1Q
QD -> 0Q

Asi by bylo dobre, ukazat si prubeh vypoctu (kazdy radek odpovida provedeni jedne derivace):

1001101PNJJN**NX**
1001101PNJJN**CY**
1001101PNJJ**CN**Y
1001101PNJ**CJ**NY
1001101PN**CJ**JNY
1001101P**CN**JJNY
10011011**P**NJJNY

Tak je asi videt, ze jsme takto presunuli symbol z konce druhe casti na konec prvni casti. Po presunu vsech symbolu z druhe casti takto onu cast obratime, protoze konec prvni casti je pred zacatkem druhe casti, takze posilame pismenka z konce na zacatek, jinymi slovy ^r.

Nakonec prepiseme neterminalni symboly na terminalni:

PX -> 11
PY -> 01
QX -> 10
QY -> 00

Nejake blaboly na konec o zkousce a teoreticke casti otazky s ww:

1) V jazyku generovanem touto gramatikou neni slovo lambda, ktere je trivialne rovnez slovem $w.w^r$. Samozrejme neni mozne udelat monotonni gramatiku generujici slovo lambda, takze je vhodne zminit, ze po by clovek po prevodu teto gramatiky na kontextovou pridal jeste pravidlo S -> lambda.

2) Pri dukazu, ze tento jazyk neni bezkontextovy pomoci pumping lemmatu a slova $0^n1^n0^n1^n$ nezapomente overit situaci, kdy napriklad:

v = 111..11
w = 111..11
x = 11..1100..00

(iteruje se samozrejme v a x)

Tedy situaci, kdy samotne iterovane slovo je na rozhrani nul a jednicek. Je jasne, ze takove slovo (po provedeni nejake iterace) uz neni mozne zapsat ve tvaru $0^n1^n0^n1^n$, ale to nestaci - je nutne dokazat, ze ho neni mozne zapsat ani ve tvaru ww. Jak to udelat: rozepsat si to slovo na tech hodne moc casti (nebo chvili popremyslet) a pumpovat smerem dolu.

Podrobneji:
Pro spor predpokladejme, ze je jazyk bezkontextovy. Urcite pro kazde n elementem prirozenych cisel obsahuje slovo:

$0^n1^n0^n1^n$

protoze toto slovo je ve tvaru ww. Vezmeme tedy n > p, n > q, kde p,q jsou koeficienty z pumping lemmatu pro bezkontextove jazyky. Uvedne slovo tedy jde pumpovat. Vezmeme pumpovaci usek vwx a budeme chtit dokazat, ze at uz vwx padne kamkoliv (OMG ja mam ale hlad), nikdy pumpovanim nedostaneme slovo ve tvaru ww (cili nedostaneme slovo spadajici do jazyka, coz je spor, protoze kdyby byl bezkontextovy, pumpovat by sel). Je dulezite si uvedomit, ze nestaci jen dokazat, ze tim nevznikne slovo $0^n1^n0^n1^n$ - tenhle jazyk neuvazujeme, zajima nas jazyk ww. Muzou nastat nasledujici pripady:

1) usek vwx cely spada do prvniho (resp. druheho) nepretrziteho useku nul, potom muzeme pumpovat nahoru a dostaneme slovo $0^k1^n0^n1^n$ (resp. $0^n1^n0^k1^n$ pro druhy pripad), kde $k > n$; toto uz ale neni slovo ve tvaru ww

2) usek vwx cely spada do prvniho (resp. druheho) nepretrziteho useku jednicek - resime uplne stejne jako pripad 1)

3) usek vwx je na rozhranni nul a jednicek, rekneme, ze to rozhranni je uvnitr slova w; co se stane, kdyz zacneme pumpovat? Muze to vypadat takhle:

nase slovo: 00000111110000011111
usek: vvwwxx

(toto snad kazdy chape, ma to znazornovat, ze v = 00, w = 01, x = 11). Po prvnim zapumpovani:

nase slovo: 0000000111111110000011111
usek: vvvvwwxxxx

Opet je videt, ze toto uz neni slovo ve tvaru ww (formalne bych to delal tak, ze bych to rozseknul veprostred a ukazal, ze pak musi byt v jedne z tech polovin neco jako $1^a0^b1^c$, zatimco v druhe je 0^d1^e - at uz je a,b,c,d,e cokoliv, je jasne, ze to neni ten samy retezec).

4) usek uvw se nachazi na rozhranni nul a jednicek a bohuzel se na rozhrani nul a jednicek nachazi (bez ujmy na obecnosti) jeho cast v. Muzeme si vsimnout, ze tim padem w ani x na rozhrani nul a jednicek neni, protoze $|vwx| \leq q < n$, jinymi slovy pumpovaci usek neni dostatecne dlouhy, aby obema svymi konci "dosahl" na nejake rozhrani nul a jednicek (protoze useky nul, resp. jednicek, jsou dlouhe n znaku, coz je vic nez delka pumpovaciho useku). Muze to vypadat takto:

nase slovo: 00000111110000011111
usek: vvwwxx

Po pumpovani nahoru dostavame:

nase slovo: 0000010111111110000011111
usek: vvvvwwxxxx

No, ja nevim, jestli by nekoho bavilo ukazovat, ze tohle neni ve tvaru ww, takze si zapumpujeme smerem dolu:

nase slovo: 0001110000011111
usek: ww

Kdyz na to pujdu s rafinovanou symbolikou: meli jsme slovo:

$0^n1^n0^n1^n$

ktere zapiseme nasledovne (za chvili popisu, proc je to rozdeleno zrovna takhle):

$0^p0^i1^j1^q1^{i+j}1^r0^n1^n$

kde:

$p + i = n$
$j + q + i + j + r = n$
$i + j = |v|$
$q = |w|$
$i + j = |x|$

Je mozne si overit, ze je to skutecne to same slovo (sectenim toho, co se da secist a spojenim stejnych nasledujicich posloupnosti znaku). Vyznam znazornim pomoci barvicek (omlouvam se barvoslepym ☹):

$0^p 0^i 1^j 1^q 1^{i+j} 1^r 0^n 1^n$

Cervene obarveny usek je slovo v,
zelene obarveny usek je slovo w,
modre obarveny usek je slovo x.

Ted kdyz budeme pumpovat dolu:

$0^p 1^q 1^r 0^n 1^n$

Protoze $p < n$ a zaroven $q + r < n$, tak je delka zbytku prvni casti slova ($0^p 1^q 1^r$) kratsi nez delka druhe casti slova ($0^n 1^n$), coz by nemelo prekvapit, ostatne proto se tomu rika zbytek - je to mensi. Kdyz tohle slovo rozsekneme v polovine, dostaneme dve casti. Prvni bude tvorena posloupnosti nul, pak posloupnosti jednicek, pak posloupnosti nul. Druha bude tvorena posloupnosti nul a pak posloupnosti jednicek. Rozhodne nebude mit na konci nuly. "Rozseknutim" uprostred tedy nedostaneme dve stejna slov a nejedna se tedy o slovo ve tvaru ww.

Ukazali jsme tedy, ze pumpovani slova $0^n 1^n 0^n 1^n$ vede vzdy ke slovu, ktere neni mozne zapsat ve tvaru ww. Jazyk ww tedy pro abecedu {0,1} neni bezkotextovy. (Pro abecedu {1} samozrejme je, to je jasne - je tam jen jedno pismeno.)

3) Druhou casti u teto otazky, je dokazat pumping lemma pro BKJ. Neni vhodne zapomenout na to, proc se musi na zacatku vzit nejdelsi cesta z S do terminaloveho symbolu (je to kvuli dukazu $|vwx| <= q$ a bez toho to dokazat nejde).

9) Preveďte (nejakým obecne použiteľným algoritmom) nasledujúci regulárny výraz na (nedeterministický) konečný automat: $((ab + c)^+a(bc)^* + b)^*$ Dokážte všetky tvrdenia, ktoré zaručujú, že tento prevod je možné urobiť vždy.

Rozumie sa, skonštruujte (nedeterministický) konečný automat, ktorý príjma jazyk reprezentovaný regulárnym výrazom.

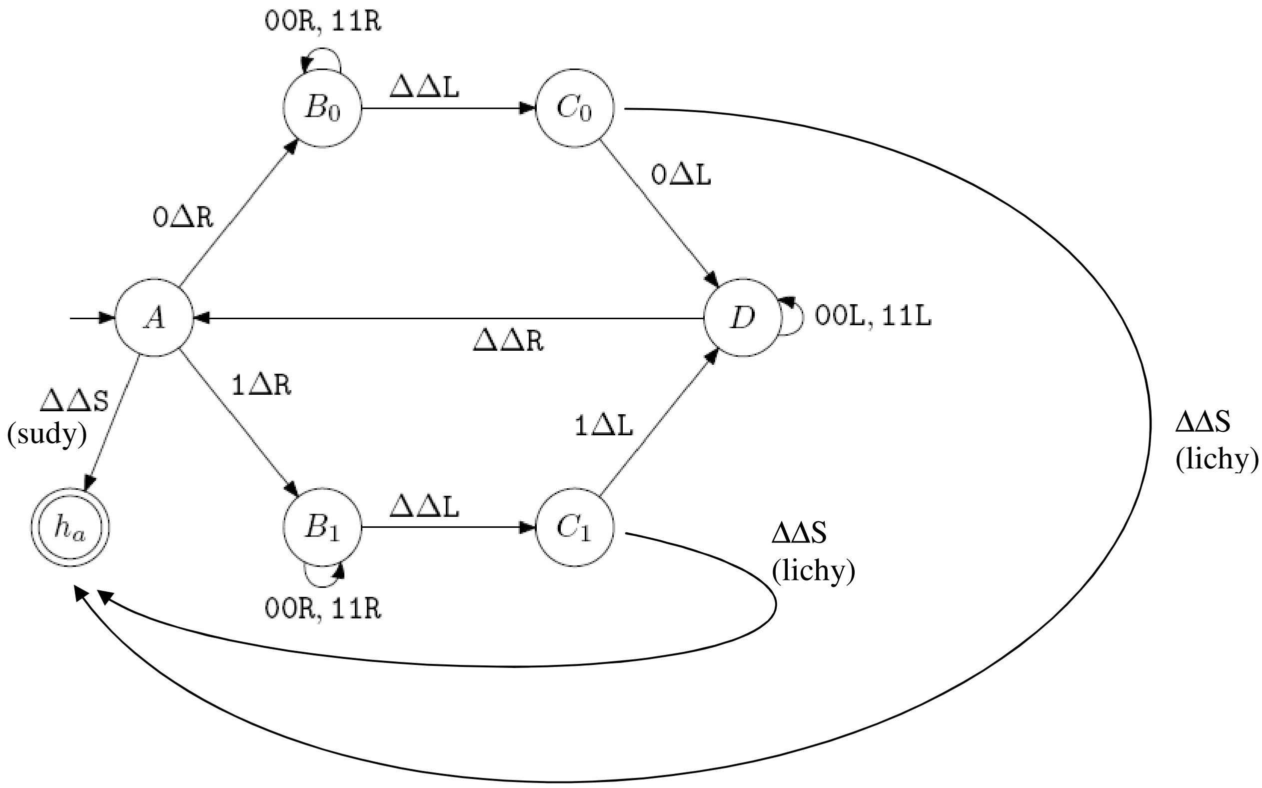
		a	b	c
->	A1		B2	
	B2	A4		C3
->	C3	A1,A4		C3
<-	A4	A1	B5,B7	C3
	B5			C6
<-	C6	A1	B5,B7	C3
<->	B7	A1	B7	C3

Hint:
1. Hodnotou regulárneho výrazu je regulárny jazyk. Uzavretosť triedy regulárnych jazykov na operácie zjednotenia, zreťazenia a iterácie.
2. Kleeneova veta (charakteristika regulárnych jazykov).

10) Udelat Turinguv stroj pro jazyk L={u | u=u^R} (neboli u je palindrom), zaradit jazyk do Chomskeho chierarchie, a dokazat tvrzeni ktere nam pomohlo rozhodnout ze ho nejde zaradit do mensi tridy.

Turinguv stroj pro u∈ {0,1}* :

Δ jsou λ, L-doleva R-doprava , format sipek je: CoPrecteme/CoZapiseme/SmerPosunu (nechtelo se mi o prepisovat)



Pamatujeme si poslední znak zmenou stavu, a jezdime sem tam:

A: pokud je string prázdný přijmout (sudá délka), jinak urči a smaž nejlevější symbol; pokud 0 -> B_0 jinak -> B_1.

B_j : pokud prvni symbol byl j; dojdi na pravy konec; -> C_j .

C_j : pokud prazdny symbol tak přijmout, jinak pokud poslední symbol nesouhlasi odmitni; jinak smaz symbol a -> D.

D: pretocime pasku na zacatek -> A.

Ten jazyk je bezkontextovy (S -> a_iSa_i | a_i | e), a pres iteracni lemma se da dokazat ze neni regularni.

11) Sestrojte zásobníkový automat, který prázdným zásobníkem přijímá jazyk všech správných uzávorkování nad abecedou {0, 1} (0 je otevírací závorka a 1 uzavírací). Lze udělat deterministicky? Definujte nedeterministický a deterministický zásobníkový automat a způsoby přijímaní slov a dokažte vztahy mezi jazyky, které generují.

ZA prazdnym zasobnikem:

δ(p,0,Z)={(p,AZ)} //kvuli tomuhle je nedeterministický

δ(p,0,A)={(p,AA)}

δ(p,1,A)={(p,λ)}

δ(p, λ,Z)={(p,λ)}

Napsal jsem mu ten automat (ten je docela jednoduchý) a že nejde udělat dereministicky, pokud má přijímat prázdným zásobníkem (protože jazyk není bezprefixový), ale koncovým stavem by to deterministicky šlo a jak (už jenom princip). Pak jsem mu sepsal ty definice a důkaz jsem napsal jenom pro to, že přijímání koncovým stavem a prázdným zásobníkem je rovnocenné z hlediska síly automatu (oba směry).

DZA koncovým stavem:

δ(p,0,Z)={(r,AZ)}

δ(r,0,A)={(r,AA)}

δ(r,1,A)={(r,λ)}

$\delta(r, \lambda, Z) = \{(q_F, Z)\}$ //skonci když precte cely slovo a je v koncovem stavu
$\delta(q_F, \lambda, Z) = \{(p, Z)\}$ // aby prijimal i prefixovy slova

Když si mě bral na ústní, říkal, že mít z testu o ten bod víc, tak mě pošle domů ihned. Potom velmi rychle proletěl můj výtvor a nakonec se jen zeptal, proč se musí vyztužit zásobník speciálním symbolem u převodu koncový stav -> prázdný zásobník (protože původní automat mohl po přečtení slova vyprázdnit zásobník a nebýt přitom v koncovém stavu, takže nový by bez výztuhy mohl přijímat něco navíc).