1.část: Najděte jednoznačnou gramatiku generující $L = \{ a^i b^i \mid i \in N \}$

$$S \rightarrow AS \mid BS \mid A \mid B$$
$$A \rightarrow aXb$$
$$B \rightarrow bYa$$
$$X \rightarrow AX \mid \lambda$$
$$Y \rightarrow BY \mid \lambda$$

Komentář:

Pomocí pravidla $S \rightarrow AS \mid BS \mid A \mid B$ můžu generovat různé permutace Áček a Béček.
Následný přepis Áčka i Béčka mi zaručuje „párovost" a, b a zároveň pomocí X, Y dovoluje rozšíření nepřímo o další neterminály / párové terminály.

Důkaz jednoznačnosti:

Dokážu pro půlku pravidel $\underline{S \rightarrow AS \mid A}$, $\underline{A \rightarrow aXb}$ a $\underline{X \rightarrow AX \mid \lambda}$
Pro druhou půlku je důkaz stejný.

Uvažme, že terminály a, b představují závorky a: (, b:).
Pak je jednoduše vidět, že pravidlo společně s generuje korektní uzávorkování, protože generuje „(X)", zatímco generuje „(X)(X)...", tedy „noří", zatímco skládá závorky vedle sebe.
Platí, že pro lib. korektně uzávorkované slovo „w" neexistuje žádný vlastní prefix řetězce „(w)", který by byl korektně uzávorkovaný. Uvažme pro spor, že je. Pak takový řetězec začnu „loupat" o uzavřené závorky. Pokud by byl prefix správně uzávorkován, měl bych řetězec oloupat do prázdného slova, avšak ve slově mi zbydou právě ty druhé strany závorek, které jsem vypustil jako suffix. Tedy existuje vždy unikátní korekt. uzávorkované slovo.
Nyní ukážu, že i pravidlo $S \rightarrow AS \mid S$ odvozuje unikátně. Pravidlo $S \rightarrow A$ totiž lze aplikovat pouze na korekt. uzávorkované slovo, aby gramatika dávala smysl.
Tudíž na slovo ve tvaru „(X)". O takovém slově však již víme, že nemá žádný prefix,

který by byl korektně uzávorkovaný. Tedy speciálně by takové slovo nešlo zpracovat pravidlem $S \rightarrow AS$, protože by to bylo ve sporu s neexistencí korektního prefixu. Tudíž každé slovo musí mít jednoznačné odvození / strom odvození.

To samé platí adekvátně pro $S \rightarrow BS / B$, $B \rightarrow bYa$, $Y \rightarrow BY$.

Tedy každé slovo má unikátní odvození, protože jsme dokázali, že žádná dvě rozdílná slova se nemohou plně shodovat ve svém odvození. $\boxed{X}$

Dokažte, že následující gramatika generuje reg. jazyk.

2)
$\circledast\ X_i \rightarrow a \quad \forall i \in \{1\_n\}$
$\circledast\ X_i \rightarrow X_j X_k \quad \forall j \geq i, \forall k > i, \forall i \in \{1\_n\text{-}1\}$

Pro zadanou obecnou gramatiku vytvoříme RegEx, který nám automaticky zaručuje regulárnost jazyka. Důkazem tedy postupně bude korektnost jednotlivých kroků přepisu pro vytvoření RegExu.

Důležitým pozorováním je, že pokud je n konečné, je i počet pravidel pro daný $X_i \rightarrow X_j X_k$ $j \geq i, k > i$ <u>konečný</u>.

Záraveň uvážíme, že $S \rightarrow X_i$ pro lib. $i \{1\_n\}$. $\rightarrow$ je také konečně mnoho.

Pak gramatické pravidlo pro X_i lze přepsat do RegExu následovně:

$$X_i \sim \left(\left(\underline{X_i (X_{i+1})^+ + X_i (X_{i+2})^+ + \dots X_i (X_n)^+}\right) + \left(\underline{X_{i+1}X_{i+1} + X_{i+1}X_{i+2} + \dots + X_{i+2}X_{i+1} + \dots + X_n X_n}\right) + \right.$$
$$\left. + \left(\underbrace{a + b + \dots}_{n \text{ neterminálů}}\right)\right)$$

Kde sekce ≋ reprezentuje pravidlo $X_i \rightarrow X_j X_k$ $j = i, k > i$. Lze nahlédnout, že použití tohoto přepisu v RegExu (díky jeho konečnosti) implikuje, že bude slovo generováno pouze tímto pravidlem a tedy nutně se musí dříve či později, daný neterminál X_i přepsat na lib. terminál a pomocí prvního pravidla $X_i \rightarrow a$, protože jinak je jen použito „prodlužovací" pravidlo $X_i \rightarrow X_i X_j$, které X_i nemění.

Tedy lze první část přepsat na:

$$\left(\underline{(a+b+\dots)(X_{i+1})^{+} + (a+b+\dots)(X_{i+2})^{+} + \dots + (a+b+\dots)(X_n)^{+}}\right)$$

kde $(a+b+\dots)$ je disjunkce n terminálů.
↳ protože 1. pravidlo $X_i \to a$, kde a je lib. terminál

tuto reprezentuje všechny pravidla $X_i \to X_j X_k$ $j \geq i, k > i$. Tam žádné opakování nelze zaměřit a tudíž nejde zaměřit ani žádné zjednodušení.

zde si lze všimnout následujícího: (příklad) $\left(a(X_j)^{+} + a\right) \sim a(X_j)^{*}$.
Tedy pokud v nahradím všechny $^{+}$ hvězdičkou, můžu celou část vypustit.

Mám tedy finální:

$$X_i \sim \left(\left(\left((a+b+\dots)(X_{i+1})^{*} + (a+b+\dots)(X_{i+2})^{*} + \dots + (a+b+\dots)(X_n)^{*}\right) + \right.\right.$$
$$\left.\left. + \left(X_{i+1}X_{i+1} + X_{i+1}X_{i+2} + \dots + X_{i+1}X_n + X_{i+2}X_{i+1} + \dots + X_nX_n\right)\right)\right)$$

Nyní si stačí všimnout, že ve výrazu se vyskytují nejmenší X_{i+1} neterminály a žádný X_i, tedy úplně stejně můžu začít přepisovat všechny X_{i+1} podle stejného algoritmu.

Jelikož je ale neterminálů jen konečně mnoho (a protože nebudu uvažovat „overflow" neterminály jako např.: X_{n+1}, které z výrazu jednoduše vypustím), tak po konečně mnoha krocích v konečně mnoha iteracích získám RegEx, který bude obsahovat jenom terminály, protože speciálně pro X_n existuje pouze $X_n \to a$.

Každý RegEx generuje reg. jazyk a zároveň RegEx generuje jazyk daný gramatikou, protože z ní přímo vychází a její pravidla jen „formalizuje" do výrazu.

Celkový RegEx tedy bude gigantický, ale protože každá iterace měla konečně mnoho kroků nad konečně dlouhým RegExem, i finální výraz bude konečný.
RegEx vytvořený iterativním přepisem pro lib. X_i je tedy důkazem regulárnosti gramatiky.