9. Bezkontextové jazyky. Bezkontextové gramatiky a zásobníkové automaty. Normální formy bezkontextových gramatik. Převod bezkontextové gramatiky na zásobníkové automaty. Použití pumping lemmatu a uzávěrových vlastností bezkontextových jazyků.

Bezkontextové jazyky a gramatiky

Bezkontextový jazyk je jazyk generovaný bezkontextovou gramatikou.
Pro připomenutí, bezkontextová gramatika je gramatika, ve které jsou všechna pravidla ve tvaru $A \rightarrow w$, kde w může být složeno z terminálů i neterminálů.

Jazyk L je bezkontextový, pokud existuje bezkontextová gramatika G taková, že $L(G) = L$
Bezkontextová gramatika (CFG) G je čtveřice (N, Σ, P, S), kde:

- N je neprázdná konečná množina neterminálních symbolů
- Σ je konečná množina terminálních symbolů taková, že $N \cap \Sigma = \emptyset$
- $P \subseteq N \times V^*$ je konečná množina pravidel ($V = N \cup \Sigma$)
- $S \in N$ je počáteční neterminál

Současně třída jazyků rozpoznávaných zásobníkovými automaty je právě třída bezkontextových jazyků. Tj. jazyk L je bezkontextový, pokud je akceptován nějakým zásobníkovým automatem (existuje zásobníkový automat M, takový, že $L(M) = L$).

Bezkontextové jazyky lze reprezentovat:

- bezkontextovou gramatikou
- zásobníkovým automatem
- rozšířeným zásobníkovým automatem

Bezkontextové jazyky mají zásadní význam v definici syntaxe programovacích jazyků, lze zapisovat např. aritmetické výrazy, programové bloky atd.

Způsob jak se taková slova z gramatik vygenerují lze zapsat např. pomocí derivačních stromů.

Derivační stromy

V gramatice je možné, aby různé derivace byly ekvivaletní v tom smyslu, že všechny tyto derivace používají stejná pravidla na stejných místech, tj. jsou aplikována na stejné výskyty neterminálů, avšak v různém pořadí. Zatímco definice takové ekvivalence je pro gramatiky typu 0 poněkud komplikovaná, lze v případě bezkontextových gramatik zavést pro třídu ekvivalentních derivací jednoduchou grafovou reprezentaci, tzv. deriva ční strom (někdy též zvanou strom odvození). Alternativně lze reprezentovat ve výše uvedeném smyslu ekvivalentní derivace jistými *kanonickými* derivacemi, v nichž aplikujeme pravidla jistým standardizovaným postupem (např. v každém kroku derivace přepisujeme ve větné formě

Derivační strom má **výsledek**, což je uspořádání listů v pořadí zleva doprava.
Ze stromu tak vznikají **levé** nebo **pravé** derivace podle toho jestli se přepisuje vždy buď nejlevější nebo nejpravější neterminál.

Odpovídající gramatika může být:

- **Jednoznačná** – pokud v jazyce neexistuje slovo, které by mělo dva nebo více derivačních stromů
- **Víceznačná** – v opačném případě

Jazyk se nazývá vnitřně víceznačný, právě když každá gramatika, která jej generuje, je víceznačná.

Příkladem může být gramatika výrazů z abecedy a, +, 1.

(1) S → S + S
(2) S → 1
(3) S → a

jejíž derivační strom vypadá takto:

```
           S
          /|\
         / | \
        /  |  \
       S  '+'  S
      /|\      |
     / | \     |
    S '+' S   'a'
    |     |
   '1'   '1'
```

tato gramatika je samozřejmě víceznačná.

Bezkontextové gramatiky se pro další použití většinou upravují do speciálních tvarů.

Normální (kanonické) tvary CFG

Redukované bezkontextové gramatiky (bez nepoužitelných symbolů)

Symbol X ∈ N ∪ Σ je nepoužitelný v CFG G = (N, Σ, P, S) právě když v G neexistuje derivace tvaru
S ⇒∗ wXy ⇒∗ wxy
pro žádné w, x, y ∈ Σ∗. Řekneme, že G je redukovaná, jestliže neobsahuje žádné nepoužitelné symboly.
Nepoužitelný symbol (terminál či neterminál) může být dvojího druhu:

- X je nepoužitelný typu I (nenormovaný) ⟸⟹ neexistuje w ∈ Σ∗ splňující X ⇒∗ w (z X nelze vyderivovat žádný terminál)

- X je nepoužitelný typu II (nedosažitelný) $\Longleftrightarrow$ neexistují $\alpha, \beta \in (N \cup \Sigma)^*$ splňující $S \Rightarrow^* \alpha X \beta$ (*X* se nevyskytuje v žádné větné formě)

Pro eliminaci nepoužitelných symbolů existuje algoritmus.
Každý neprázdný bezkontextový jazyk je generovaný nějakou redukovanou CFG.

Gramatiky bez ε-pravidel

Nejsou zde ε-pravidla, nebo je zde jen jedno pravidlo $S \to \varepsilon$, a ε není na pravé straně žádného pravidla.

Gramatiky bez jednoduchých pravidel

Nejsou zde pravidla $A \to B$. kde $A, B \in N$.

Necyklické, vlastní a rekursivne gramatiky

CFG $G = (N, \Sigma, P, S)$ se nazývá **necyklická**, právě když neexistuje $A \in N$ takový, že $A \Rightarrow^+ A$.

G se nazývá **vlastní**, právě když je bez nepoužitelných symbolů, bez ϵ-pravidel a necyklická.

Ke každému neprázdnému bezkontextovému jazyku existuje **vlastní** bezkontextová gramatika, která jej generuje

Neterminál A v CFG $G = (N, \Sigma, P, S)$ se nazývá **rekursivní** jestliže v G existuje derivace $A \Rightarrow^+ \alpha A \beta$. Je-li $\alpha = \epsilon$ resp. $\beta = \epsilon$, pak A se nazývá levorekursivní resp. pravorekursivní. CFG bez levorekursivních terminálů se nazývá nelevorekursivní. α, β jsou levé strany pravidel v *P*.

Chomského NF

Gramatika je v CNF, právě když je bez ε-pravidel a obsahuje jen pravidla

- $A \to BC, B, C$
- $A \to a$
- $S \to \varepsilon$

Každý CFL lze generovat gramatikou v CNF.

Ve zkratce algoritmus funguje tak, že pravé strany délky větší než 2 nahradí novými neterminály, a tak se tato pravá strana zkrátí. Například pravidlo X → ABC se změní na pravidla X → A<BC>, <BC> → BC (<BC> je nový neterminál). Pro pravé strany délky 2, které se skládají z terminálu a neterminálu nahradí terminál novým neterminálem. Např. X → aC se změní na X → <a>C, <a> → a.

Výhodou CNF je, že jejich uzly mají vždy dva potomky, pokud se nejedná o listy. Takové stromy, kde je největší délka *j* mají právě 2^j listů, ovšem v CNF je počet listů roven počtu jejich přímých předků, tj. mají $2^{(j-1)}$ listů.

Greibachové NF

Gramatika je v GNF, pokud je bez ε-pravidel a pokud dále obsahuje jen pravidla

- $A \to a\alpha$, kde a terminál a α je řetězec neterminálů.

Každý CFL lze generovat CFG v GNF.

Velmi zhruba naznačený postup je následující: Pokud je $L(G)$ prázdný, pak jej definujeme jako jediné pravidlo ($S \to aS$). Pokud je $L(G)$ neprázdný, odstraníme levou rekurzi a převedeme do GNF.

Použití pumping lemmatu pro CFL

Nechť L je CFL. Pak existují přirozená čísla p, q taková, že každé slovo z z L, $|z| > p$ lze psát ve tvaru $z = uvwxy$, kde

- alespoň jedno ze slov v, x je neprázdné
- $|vwx| \leq q$
- uv^iwx^iy je z L pro všechna $i \geq 0$.

Tvrzení zůstává v platnosti i když namísto konstant p, q budeme všude psát jen jedinou konstantu n.

Důkaz pro jazyk L = $\{a^jb^jc^j: j> 0\}$,že není CFL.

1. Předpokládejme, že je CFL
2. Zvolíme $p = q = n$
3. Přemýšlíme jak slovo zapsat, aby splňovalo podmínky lemmatu, ale po napumpování přestalo patřit do jazyka L.
4. Protože platí, že $|vwx| \leq q$ a $q = n$, nemůže se stát, že bychom do vwx dokázali nějakým způsobem nacpat všechny tři znaky. Tj. bude tam buď a nebo b nebo c. Nebo ab, nebo bc. Nemůže být prázdný z podmínky lemmatu.
5. Jakýmkoliv pumpováním se nám nemůže podařit přidat tři různé znaky, tj. podmínka je porušena.

Lemma o vkládání je vlastně ve tvaru implikace. Její kontrapositivní formu lze použít k důkazu, že nějaký jazyk *není* CFL. Stačí ukázat platnost následujícího tvrzení:

Pro libovolnou konstantu n existuje slovo z, $|z| > n$ takové, že pro všechna slova u, v, w, x, y splňující: $z = uvwxy$, $vx \neq \epsilon$ a $|vwx| \leq n$ existuje takové i, že $uv^iwx^iy \notin L$.

Příklad:

$L = \{wcw \mid w \in \{a,b\}*\}$

Pro libovolnou konstantu n existuje slovo $z \in L, |z| > n$: Zvolíme slovo z jako libovolné, pro důkaz vhodné, slovo z z jazyka L - $z = a^n b^n c a^n b^n$. Dále pro každé libovolné rozdělení slova z splňující následující podmínky: $z = uvwxy, \ vx \neq \epsilon, \ |vwx| \leq n$ musí existovat $i \in N_0$ takové, že $uv^iwx^iy \notin L$.

Nyní musíme takové i nalézt. Položme tedy $i = 0$ a ověříme, že $uwy \notin L$ rozebráním jednotlivých případů:

1. c není podslovo vwx. Potom ale $uwy = z'ca^nb^n$ nebo $uwy = a^nb^ncz'$, kde $|z'| < 2n$. Z toho plyne, že $uwy \notin L$.
2. c je podslovo vwx. Tuto situaci rozdělíme ještě na další dva případy:
   - c je podslovo vx: Pak ale $c \notin uwy$, z čehož plyne, že $uwy \notin L$
   - c je podslovo w: $|vwx| \leq n, \ uwy = a^nb^{n-|v|}ca^{n-|x|}b^n$. Současně $vx \neq \epsilon$ a tudíž $|v| + |x| > 0$ a tedy $uwy \notin L$.

Zásobníkové automaty

Tak jako k regulárním gramatikám existují konečné automaty, které rozpoznávají právě jazyky generované těmito gramatikami, tak i ke gramatikám bezkontextovým existují ve výše uvedeném smyslu ekvivaletní automaty – tzv. zásobníkové automaty (push-down automata – PDA). PDA si lze představit jako (nedeterministický) konečný automat s tím, že navíc obsahuje (pomocnou) paměť (a díky ní i další zdroj nedeterminismu), která pracuje jako zásobník (push-down store) a jejíž velikost není shora omezena – je tzv. *potenciálně* nekonečná v následujícím smyslu: v každém okamžiku (konečného) výpočtu je sice konečná (tj. v zásobníku je uloženo jen konečně mnoho symbolů), ale kdykoli ji můžeme rozšířit o další konečný počet paměťových míst (přidat další symboly na zásobník).

Ze vstupní pásky, na níž je zapsáno slovo nad jistou vstupní abecedou, lze pouze číst a čtecí hlava se pohybuje jen vpravo. Automat může na vrchol zásobníku ukládat symboly (opět z jisté abecedy) a takto uložené symboly může následně číst s tím, že smí číst pouze z vrcholu zásobníku: přečtený symbol je z vrcholu odstraněn (tj. systém LIFO – Last In First Out). Jinými slovy, nelze číst do hloubi zásobníku, aniž by přečtené symboly nebyly odstraněny – zásobník, u něhož naopak toto "nedestruktivní" čtení lze realizovat se v angličtině nazývá *stack*, na rozdíl od našeho *push-down store* s desktruktivním čtením. Takto intuitivně popsané zařízení nyní formalizujme.

Nedeterministický zásobníkový automat (PDA), je sedmice:

- *Q* – konečná množina stavů
- *Σ* – vstupní abeceda
- *Γ* – zásobníková abeceda
- *δ* – přechodová funkce, ze *(Q x (Σ x Γ)) →Množina konečných podmnožin (Q x Γ)*
- q_0 – počáteční stav
- Z_0 – počáteční symbol na zásobníku
- *F* – množina koncových stavů

$\delta(q_0, a, Z_0) = \{(q_0, AZ_0)\}$; Z_0 - dno zásobníku, A - vrchol zásobníku, přidal se v tomto kroku
$\delta(q_0, c, A) = \{(q_1, A)\}$ … obsah zásobníku se nezměnil, pouze se přešlo do stavu q_1
$\delta(q_1, b, A) = \{(q_1, \varepsilon)\}$ … stav se nezměnil, ze zásobníku se přečetlo A – bylo odebráno
vrchol zásobníku běžného PDA se píše vlevo

Konfigurace PDA

Konfigurací PDA M se nazve trojici (*q, w, α*), kde

- q – stav řídící jednotky
- w – nenačtená část vstupního řetězce, první symbol pod čtecí hlavou
- *α* - obsah zásobníku (vrchol zásobníku)

Vnitřní konfigurace (= částečná konfigurace) = aktuální stav a celý obsah zásobníku (vrchol se píše vlevo)
Počáteční konfigurace PDA má tvar (q_0, w, Z_0) pro slovo z abecedy *Σ,* tj. automat je v počátečním stavu q_0, na vstupní pásce je řetězec w a v zásobníku je startovací symbol Z_0.
Koncová konfigurace má tvar *(q, ε, α),* kde *q* je koncový stav a *α* je řetězec (může být i prázdný) ze zásobníkové abecedy.

Přechod PDA

Přechod PDA se definuje jako binárni relace |- (krok výpočtu), která je definována na množině konfigurací zásobníkového automatu *M*.

(q, aw, Zα) |- *(q', w, γα)*

Nachází-li se zásobníkový automat *M* ve stavu *q* a na vrcholu zásobníku je uložen symbol Z*,* pak po přečtení vstupního symbolu *a* různého od *ε* může automat přejít do stavu *q'*, přičemž se čtecí hlava posune doprava a na vrchol zásobníku se uloží po odstranění symbolu *Z* řetězec *γ*. Je-li *γ* = *ε* pak je pouze odstraněn vrchol zásobníku. Je-li *a* = *ε*, pak se neposouvá čtecí hlava a nový obsah zásobníku není určován příštím symbolem. Tento typ přechodu se nazývá ***ε*-přechodem**. *ε*-přechod může nastat i tehdy, kdy byly načteny všechny vstupní symboly.

Rozpoznávání PDA

Rozpoznávání koncovým stavem – slovo *w* je automatem přijato, jestliže existuje možnost, že po přečtení slova *w* se ocitne v koncovém stavu.
$L(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \overset{*}{\vdash} (q_f, \varepsilon, \alpha)$, kde $q_f \in F, \alpha \in \Gamma^*\}$

Rozpoznávání prázdným zásobníkem – slovo w je přijato, jestliže existuje možnost, že po zpracování slova se automat ocitne s prázdným zásobníkem. Děje se tak pomocí speciálního symbolu označujícího dno zásobníku, který PDA na začátku zapíše na zásobník.
$L_e(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, kde $q \in Q\}$.
z
Takto definovaný PDA je nedeterministický – akceptuje slovo, jestliže existuje **alespoň jeden výpočet**, který vede z počáteční konfigurace do akceptující. V případě možnosti automat „hádá správně“, protože nesprávna volba sama o sobě nemuže zpusobit zamítnutí vstupu – ten muže být zamítnut jedine tedy, pokud žádna správná volba neexistuje

Z pohledu jazyka: Jazyk rozpoznávaný koncovým stavem nebo prázdným zásobníkem. Oba způsoby jsou ekvivalentní.

Rreprezentace konfigurací

PDA $A = (\{q, r\}, \{a, b, c\}, \{Z, A\}, \delta, q, Z, \{r\})$ - akceptuje koncovým stavem
PDA $A' = (\{q, r\}, \{a, b, c\}, \{Z, A\}, \delta, q, Z, \emptyset)$ - akceptuje prázdným zásobníkem

Přechodovou funkci mají oba automaty stejnou:
$\delta(q, a, Z) = \{(q, AZ)\}$
$\delta(q, a, A) = \{(q, AA)\}$
$\delta(q, c, Z) = \{(r, Z)\}$
$\delta(q, c, A) = \{(r, A)\}$
$\delta(r, b, Z) = \{(r, \epsilon)\}$
$\delta(r, b, A) = \{(r, \epsilon)\}$

PDA sa dá ešte reprezentovať pomocou prechodového grafu (stavového diagramu)

Rozdíl mezi zásobníkovým automatem a rozšířeným zásobníkovým automatem

Zásobníkový automat má přechodovou funkci δ definovanou jako zobrazení z $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ do konečných podmnožin množiny $(Q \times \Gamma^*)$. Oproti tomu rozšířený zásobníkový automat má definovanou rozšířenou přechodovou funkci δ jako zobrazení z konečné podmnožiny množiny $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^*$ do konečných podmnožin množiny $(Q \times \Gamma^*)$.

Neformálně řečeno, rozšířený zásobníkový automat „vidí“ hlouběji do zásobníku – tj. čte 0–*n* vrchních symbolů.

Ke každému rozšířenému PDA existuje ekvivalentní (obyčejný) PDA.

Převod bezkontextové gramatiky na zásobníkové automaty

CFG a PDA jsou vzájemně převoditelné. Třída jazyků rozpoznávána PDA je právě třída bezkontextových jazyků.

<u>Nedeterministická syntaktická analýza</u>

Shora dolů:
Ke každé CFG G lze sestrojit PDA M takový, že $L(G) = L_e(M)$.

V levé derivaci je v jednom kroku odvození nahrazen (nejlevější) neterminál *A* pravou stranou *X1 . . .Xn* nějakého *A*-pravidla. V *M* této situaci odpovídá náhrada *A* na vrcholu zásobníku řetězem *X1 . . .Xn*. Terminální vstupy ze vstupu se načítají, pokud jsou na vrcholu zásobníku, akceptujeme prázdným zásobníkem, automat je jednostavový.

$M = (\{q\}, \Sigma, N \cup \Sigma, \delta, q, S, \emptyset)$, kde δ je definována:

- $\delta(q, \varepsilon, A)$ obsahuje (q, α) právě když $A \to \alpha \in P$
- $\delta(q, a, a) = \{(q, \varepsilon)\}$ pro všechna $a \in \Sigma$

Zdola nahoru:
Nechť G je libovolná CFG, pak lze zkonstruovat rozšířený PDA R takový, že L(G) = L(R).

Vrchol zásobníku píšeme vpravo. Konstruujeme rozšířený PDA, který simuluje pravou derivaci v G v obráceném pořadí.
PDA má kroky dvojího typu:
1. může kdykoli číst do zásobníku vstupní symbol,
2. (redukce) je-li na vrcholu zásobníku řetěz tvořící pravou stranu nějakého pravidla v G, může ho nahradit odpovídajícím levostranným neterminálem (a ze vstupu nic nečte).
Do koncového stavuj přejdeme, pokud je na zásobníku „dno“S. Akceptuje tedy koncovým stavem.

Nechť $G = (N, \Sigma, P, S)$.
Položme $R = (\{q, r\}, \Sigma, N \cup \Sigma \cup \{\bot\}, \delta, q, \bot, \{r\})$, kde $\bot$ je nově přidaný symbol a kde δ je definována takto:

1. $\delta(q, a, \varepsilon) = \{(q, a)\}$ pro všechna $a \in \Sigma$ *...načítání vstupu na zásobník*,
2. je-li $A \to \alpha$, pak $\delta(q, \varepsilon, \alpha)$ obsahuje (q, A) *...redukce*,
3. $\delta(q, \varepsilon, \bot S) = \{(r, \varepsilon)\}$ *...akceptování*.

zdola nahoru. Nedeterminismus se objevuje v těchto dvou základních variantách:

(1) v konfiguracích, kdy PDA má na vrcholu pravou stranu nějakého pravidla a volí zda redukovat tuto pravou stranu na odpovídající neterminál nebo zda číst ze vstupu – viz výše např. $(q, \ \perp E + T, \ *i)$, kde lze nejen provést uvedené čtení symbolu '*', ale též redukci dle $E \rightarrow E + T$). Tuto situaci nazýváme konfliktem typu čtení versus redukce;

(2) v konfiguracích, kdy PDA má na vrcholu takový řetěz, že jsou možné alespoň dvě různé redukce (tj. redukce podle různých pravidel); jako příklad může opět sloužit výše uvedená $(q, \ \perp E + T, \ *i)$, kde lze redukovat jak $E + T$ na E, tak i příponu tohoto řetězu, tj. T, na (shodou okolností opět) E. Tuto situaci nazývýme konfliktem typu redukce versus redukce.

Variantu (1) lze obecně charakterizovat existencí jak pravidla $A \rightarrow \alpha$, tak i $B \rightarrow \alpha\beta$ (spolu s existencí například $S \Rightarrow^* \gamma Az$ a $S \Rightarrow^* \gamma Bz$), kdežto pro (2) je typická existence pravidel $A \rightarrow \alpha$ a $B \rightarrow \beta\alpha$ (spolu například s $S \Rightarrow^* \gamma\beta Az$ a $S \Rightarrow^* \gamma Bz$). Současný výskyt obou typů konfliktů je samozřejmě v konfiguraci PDA též možný.

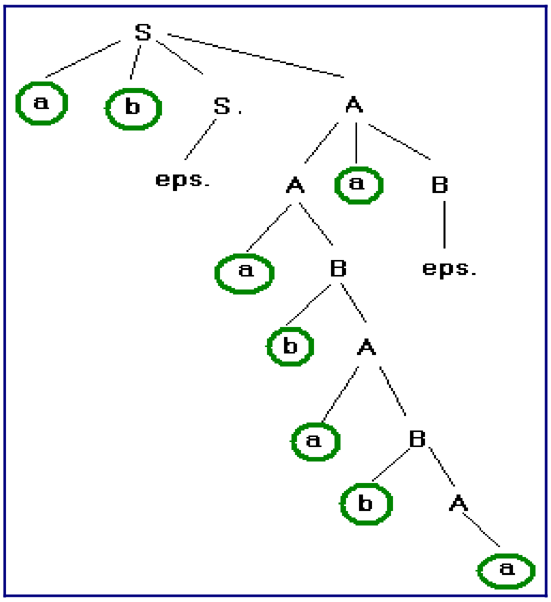
Co se nedeterminismu týče, je tedy vidět, že zatímco u analýzy shora dolů se omezuje na případ, kdy máme volit, kterou z pravých stran nahradit neterminál na vrcholu zásobníku, je u analýzy zdola nahoru jeho povaha poněkud košatější.

<u>Příklad syntaktickej analýzy:</u>

G = ({S, A, B}, {a, b}, P, S)
P =
{S → ε | abSA

A → AaB | aB | a,
B → aSS | bA | ε}

Derivační strom pro slovo w = abababaa



Automat, který provádí syntaktickou analýzu shora dolů:
A = ({q}, {a,b}, {S, A, B, a, b}, δ, q, S, $\emptyset$)
2 typy přechodových fcí:

- *levou stranu pravidel z gram. dáváme na zásobník levé strany přechodové fce PDA, pravou stranu pravidel z gram. na zásobník pravé strany*

δ(q, ε, S) = {(q, ε), (q, abSA)}
δ(q, ε, A) = {(q, AaB), (q, aB), (q, a)}
δ(q, ε, B) = { (q, aSS), (q, bA), (q, ε)}

- *„mazací" přechodové fce*

δ(q, a, a) = {(q, ε)}
δ(q, b, b) = {(q, ε)}

- **Akceptující výpočet nad slovem abababaa:**

(intuice: postupuju podle derivačního stromu, sleduju jeho rozvíjení odshora a čtu ho zleva)
vrchol zásobníku je vlevo

$(q, abababaa, S) \vdash (q, abababaa, abSA) \vdash (q, bababaa, bSA) \vdash$

$(q, ababaa, SA) \vdash (q, ababaa, A) \vdash (q, ababaa, AaB) \vdash$

$(q, ababaa, aBaB) \vdash (q, babaa, BaB) \vdash (q, babaa, bAaB) \vdash$

$(q, abaa, AaB) \vdash (q, abaa, aBaB) \vdash (q, baa, BaB) \vdash$

$(q, baa, bAaB) \vdash (q, aa, AaB) \vdash (q, aa, aaB) \vdash$

$(q, a, aB) \vdash (q, \varepsilon, B) \vdash (q, \varepsilon, \varepsilon)$
… celé slovo se přečetlo a zásobník se vyprázdnil → **automat slovo akceptuje**

Automat pro syntaktickou analýzu zdola nahoru:
$A_2 = (\{q, r\}, \{a, b\}, \{S, A, B, a, b, \perp\}, \delta, q, \perp, \{r\})$
3 typy přechodových funkcí:

- *terminály z gramatiky se předávají na zásobník*

δ(q, a, ε) = {(q, a)}
δ(q, b, ε) = {(q, b)}

- *pravou stranu pravidel z gram. dáváme na zásobník levé strany přechodové fce PDA, levou stranu pravidel z gram. na zásobník pravé strany*

δ(q, ε, abSA) = {(q, S)}
δ(q, ε, ε) = {(q, S), (q, B)} **! ! !** Nesmí být rozděleno na: δ(q, ε, ε) = {(q, S)} a δ(q, ε, ε) = {(q, B)}, nebyla by to pak funkce.
δ(q, ε, AaB) = δ(q, ε, aB) = δ(q, ε, a) = {(q, A)}
δ(q, ε, aSS) = δ(q, ε, bA) = {(q, B)}

- *přechod do koncového stavu*

δ(q, ε, $\perp$S) = {(r, ε)}

- **Akceptující výpočet nad slovem abababaa:**

(Intuice: čtu derivační strom zleva – jako když teče voda (= používám první skupinu pravidel), pokud by voda musela téct do kopce, použiju pravidlo z druhé skupiny)
vrchol zásobníku je vpravo
$(q, abababaa, \perp)|\!\!-\!\!-(q, bababaa, \perp a)|\!\!-\!\!-(q, ababaa, \perp ab)|\!\!-\!\!-$

$(q, ababaa, \perp abS)|\!\!-\!\!-(q, babaa, \perp abSa)|\!\!-\!\!-(q, abaa, \perp abSab)|\!\!-\!\!-$

$(q, baa, \perp abSaba)|\!\!-\!\!-(q, aa, \perp abSabab)|\!\!-\!\!-(q, a, \perp abSababa)|\!\!-\!\!-$

$(q, a, \perp abSababA)|\!\!-\!\!-(q, a, \perp abSabaB)|\!\!-\!\!-(q, a, \perp abSabA)|\!\!-\!\!-$

$(q, a, \perp abSaB)|\!\!-\!\!-(q, a, \perp abSA)|\!\!-\!\!-(q, \varepsilon, \perp abSAa)|\!\!-\!\!-$

$(q, \varepsilon, \perp abSAaB)|\!\!-\!\!-(q, \varepsilon, \perp abSA)|\!\!-\!\!-(q, \varepsilon, \perp S)|\!\!-\!\!-(r, \varepsilon, \varepsilon)$
… automat přečetl celé slovo a přešel do koncového stavu r, **slovo akceptuje**.

U nedeterministické analýzy automat slovo neakceptuje, pokud **žádný** z možných výpočtů není akceptující.

Efektivnost syntaktické analýzy

Nederministický PDA ⇒ nedeterministický algoritmus ⇒ exponenciální deterministický algoritmus

Řešení:
• deterministický algoritmus složitosti $O(n^3)$, kde n = |w| (algoritmus Cocke – Kasami – Younger)
• deterministické zásobníkové automaty a deterministické bezkontextové jazyky
• lineární algoritmy pro speciální třídy deterministických bezkontextových jazyků

Uzávěrové vlastnosti

Bezkontextové jazyky jsou uzavřené na operace **sjednocení**, **zřetězení**, **iteraci** a **pozitivní iteraci, průniku s regulárním jazykem.**

Nejsou uzavřeny vzhledem k operacím **průniku** a **doplňku**.

Sjednocení:

dvě gramatiky (búno. mající disjunktní množiny neterminálů) spojím dohromady, zavedu nový počáteční neterminál S, z kterého se půjde buď do S1 nebo do S2 (nové pravidla), ostatní pravidla se sjednotí.

Definujme $G = (N_1 \cup N_2 \cup \{S\},\ \Sigma_1 \cup \Sigma_2,\ P,\ S)$, kde S je nový symbol a $P = P_1 \cup P_2 \cup \{S \to S_1,\ S \to S_2\}$

Jazyk $L = L_1 \cup L_2$ je generován gramatikou G.

Zřetězení:
podobně, přidáme pravidlo S -> S1S2.

Iterace a Pozitivní iterace
Definujme $G = (N_1 \cup \{S\},\ \Sigma_1,\ P,\ S)$**, kde S je nový symbol a** $P = P_1 \cup \{S \to SS_1|\epsilon\}$

Jazyk $L = L_1^*$ je generován gramatikou G.

Definujme $G = (N_1 \cup \{S\},\ \Sigma_1,\ P,\ S)$, kde S je nový symbol a $P = P_1 \cup \{S \to SS_1|S_1\}$

Jazyk $L = L_1^+$ je generován gramatikou G.

Neuzavřenost na průnik a doplněk

$L_1 = \{a^n b^n c^m \mid m, n \geq 1\},\ L_2 = \{a^m b^n c^n \mid m, n \geq 1\}$

Kdyby byla třída bezkontextových jazyků uzavřena vzhledem k operaci průnik, pak i $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ by musel být bezkontextový, což však není.

Neuzavřenost vůči doplňku plyne z uzavřenosti třídy bezkontextových jazyků vzhledem k operaci sjednocení, neuzavřenosti na průnik a De Morganových pravidel:

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