Bezkontextové gramatiky

Definice: **BKG: $G = (N, T, P, S)$,** kde

1. N je množina neterminálních symbolů,
2. T " " terminálních " ,
3. $S \in N$ je počáteční symbol,
4. P je množina přepisovacích pravidel tvaru

$$A \rightarrow \alpha \quad , \quad \text{kde } A \in N, \quad \alpha \in (N \cup T)^*$$

Bezkontextový jazyk

Definice: **BKL: $L(G) = \{ w : S \Rightarrow^* w , \; w \in T^* \}$**

Tj. L(G) je množina řetězců derivovatelných z S

Úmluva pro zjednodušení zápisů:

a, b, c, ... představují terminální symboly

A, B, C, ... „ neterminální „

X, Y, Z, ... „ $N \cup T$

$\alpha, \beta, \gamma, ...$ „ řetězce z $N \cup T$

u, v, z, . . „ „ z terminálních symbolů

e představuje prázdný řetězec

- **DERIVACE** řetězce α je posloupnost kroků odvození α pomocí přepisovacích pravidel gramatiky

$$S = \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = \alpha$$

Dtto $S \Rightarrow^* \alpha$ pozn.: $\Rightarrow^*$ je uzávěr relace $\Rightarrow$

- **PŘÍMÁ DERIVACE** $\alpha A \beta \Rightarrow \alpha \gamma \beta$, kde $A \rightarrow \gamma \in P$
- **DERIVAČNÍ STROM** je grafickým vyjádřením derivace (struktury) řetězce. Kořenem je počáteční symbol, uzly jsou prvky $N \cup T$, listy jsou prvky T, větve z uzlu A vedou do uzlů, které zleva doprava tvoří řetězec α, který je pravou stranou pravidla $A \rightarrow \alpha$

Př. G[E]

$E \rightarrow T \mid E + T$
$T \rightarrow F \mid T * F$
$F \rightarrow (E) \mid i$

Vytvořte derivační strom a derivaci věty např. i + i * i (na tabuli)

Vztah derivace a derivačního stromu: derivaci odpovídá jeden strom, jednomu stromu odpovídá více derivací.

- KANONICKÉ DERIVACE
  - **Levá derivace** -expanduje vždy nejlevější neterminál
  - **Pravá derivace** -expanduje vždy nejpravější neterminál

Př. levá a pravá derivace věty i + i * i
na tabuli

- **VĚTNÁ FORMA**

Def.: Řetězec α se nazývá větnou formou v gramatice G, s počátečním symbolem S, platí-li:

$S \Rightarrow^* \alpha$, kde $\alpha \in (N \cup T)^*$

- **VĚTA**

Def.: Řetězec α se nazývá větou v gramatice G, s počátečním symbolem S, platí-li:

$S \Rightarrow^* \alpha$, kde $\alpha \in T^*$

- **FRÁZE**

Def.: Nechť $\lambda = \alpha \beta \gamma$ je větná forma v gramatice G. Podřetězec β se nazývá frází větné formy λ vzhledem k neterminálnímu symbolu A, platí-li

$S \Rightarrow^* \alpha A \gamma$ a $A \Rightarrow^* \beta$

Tzn.frázi tvoří listy podstromu derivačního stromu.

- **JEDNODUCHÁ FRÁZE** větné formy $\alpha A \gamma$ vzhledem k neterm. A je podřetězec β, platí-li

$S \Rightarrow^* \alpha A \gamma$ a $A \Rightarrow \beta$

- **L-FRÁZE**

je nejlevější jednoduchou frází

Př.
Najdi fráze, jednoduché fráze a l-frázi větné formy i*i+i v G[E]
(na tabuli)

Problémy analýzy při konstrukci derivačního stromu:

1. (shora dolů) Kterou z pravých stran vybrat k derivování

2. (zdola nahoru) Jak vymezit l-frázi a na co ji redukovat

řešení: -buď analýza s návratem (neefektivní složitost)

-nebo deterministická analýza (jen pro některé, druhy BKG)

Víceznačnost gramatik

Def. Věta generovaná gramatikou G je víceznačná, existují-li alespoň dva různé derivační stromy této věty.
G pak rovněž nazýváme víceznačnou.

Př. Jazyk { $a^m c a^n$; m, n >= 0 }

je generován gramatikou   S -> aS | Sa | c

-Je věta aaca jednoznačná? jak vypadá strom, je jen jeden?

-Může pro nejednoznačnou gramatiku existovat ekvivalentní jednoznačná gramatika? Může:

```
S -> aS | Z
Z -> Za | c
```

Př. G[E]   E -> E + E | E * E | i

-Jaké jsou důsledky v generovaném jazyce ?

Věta: Nutnou podmínkou jednoznačnosti gramatiky je, aby pro žádný neterminální symbol neexistovalo jak pravidlo rekurzivní zprava, tak i pravidlo rekurzivní zleva.

Problém nejednoznačnosti bezkontextověch jazyků je algoritmicky nerozhodnutelný. Tzn. je dokázáno, že nikdy nebude existovat pro takový problém algoritmus.

Př. Syntaktický tvar podmíněného příkazu:

```
S -> a S b S | a S | c
```

-Je G[S] víceznačná ?

```
S1 -> a S2 b S1 | a S1 | c
S2 -> a S2 b S2 | c
```

Gramatika je také, víceznačná, existují-li v G pro rekurzivní neterm. symbol A alespoň 2 rekurzivní pravidla, z nichž jedno je rekurzivní zprava (zleva) a má shodný prefix (postfix) rekurzivního symbolu A s druhým pravidlem.

Jazyky, které nelze generovat jednoznačnou gramatikou se nazývají inherentně nejednoznačné.

Úpravy gramatik

Odstranění zbytečných symbolů (získání redukované gramatiky)

Zbytečný je takový symbol X, který buď (1.)je-li neterminální z něj nelze generovat terminální řetězec, nebo (2.)ať je terminální či neterminální, je nedosažitelný z S.

$$\underbrace{S \Rightarrow^* \mathrlap{\overbrace{}^{1.}}w\ X\ y}_{2.} \Rightarrow^* w\ x\ y\text{, kde } w,\ x,\ y \in T^*$$

Postup při eliminaci zbytečných symbolů

1. a)Označíme všechny $X \in T$.
   b)Označíme všechny $X \in N$, pro něž existuje X-pravidlo, jehož pravá strana neobsahuje neoznačený symbol.
   c)Opakujeme krok b), dokud přibývá označených symbolů.
   d)Neoznaznačené symboly jsou zbytečné.
2. a) Označíme počáteční symbol S.
   b) Označíme všechny symboly z pravých stran pravidel s označeným levostranným symbolem.
   c)Opakujeme krok b), dokud přibývá označených symbolů.
   d)Neoznačené symboly jsou zbytečné.

! záleží na pořadí kroků 1. a 2. !

Př. G[S]: $S \rightarrow a \mid A \qquad A \rightarrow A\ B \qquad B \rightarrow b$

Odstranění prázdných pravidel

Gramatika G je bez prázdných pravidel, jestliže buď neobsahuje žádné pravidlo $A \rightarrow e$, nebo obsahuje jediné takové pravidlo tvaru $S \rightarrow e$ a S se nevyskytuje na pravé straně žádného pravidla v G.

Postup při odstranění prázdných pravidel

1. Označíme všechny symboly X, pro něž existuje pravidlo s prázdnou pravou stranou.
2. Označíme všechny symboly X, pro něž existuje pravidlo s pravou stranou obsahující pouze označené symboly.
3. Opakujeme 2 dokud přibývá označených symbolů.
4. Takto získanou množinu označíme N_e .
5. Každé pravidlo gramatiky mající na pravé straně jeden či více symbolů z N_e , nahradíme množinou pravidel vzniklých všemi možnými způsoby vypuštění v pravých stranách symbolů z N_e. Případně vznikající pravidla tvaru $X \rightarrow e$ do výsledné gramatiky nezařazujeme.
6. Obsahuje-li N_e počáteční symbol S, vytvoříme nový počáteční symbol S´ s pravidly $S' \rightarrow e$ a $S' \rightarrow S$

(Gramatika bez prázdných pravidel je nevypouštějící = nezkracující, větné formy při derivování se nezkracují)

Př. Na tabuli. Odstraňte prázdná pravidla z G[S]: S → a S b S | e

Výsledek: S´ → S | e
S → a S b S | a b S | a S b | a b

Odstranění jednoduchých pravidel

Jednoduchá pravidla mají tvar A → B , kde A, B ∈ N

Odstranění = žádný problém = nahradíme A → B všemi možnými pravidly vzniklými záměnou B za pravé strany B-pravidel
Př. Zkusme pro G[E] na tabuli

Odstranění cyklů

A =>* A implikuje existenci jednoduchých pravidel
Cyklus je evidentní nešvar. Proč?
Cykly eliminujeme odstraněním jednoduchých pravidel.

Odstranění libovolného pravidla

Nechť chceme z G odstranit pravidlo A → α B β. Musíme proto místo něj dát do G všechna pravidla tvaru A → α γ β, kde γ jsou pravé strany B pravidel.
Př. V G s pravidly A → a A A | b odstranit pravidlo A → a A A
Na tabuli:
Výsledek: A → a a A A A | a b A | b

Upravená gramatika

neobsahuje cykly, e-pravidla a zbytečné symboly

Odstranění levé rekurze

(Greibachové normální forma: ∀pravé strany začínají terminálem)

Levorekurzivní gramatiku nelze použít k analýze shora dolů

Odstranění pravidla rekurzivního zleva:

Nechť je dána BKG G = (N, T, P, S), ve které,

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

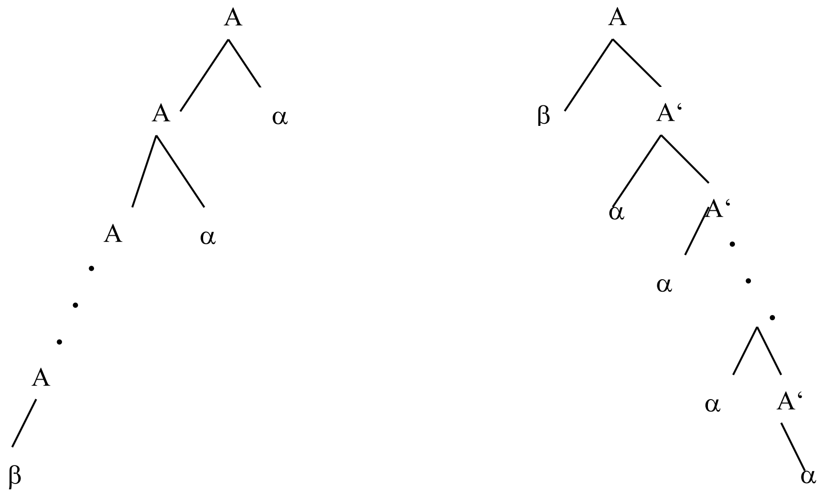
jsou všechna A pravidla v P a žádné, z β nezačíná A.

Pak G' = (N ∪ { A'}, T, P', S), kde P' obsahuje místo uvedených pravidel pravidla:

$$A \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$$
$$A' \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m \mid \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A'$$

je ekvivalentní s gramatikou G

Levou rekurzi nahradíme pravou, jak je zřejmé z obr.



Př.a) G[E] na tabuli. $E \rightarrow E + T \mid T$ **?proč nepoužijeme** $E \rightarrow T + E$

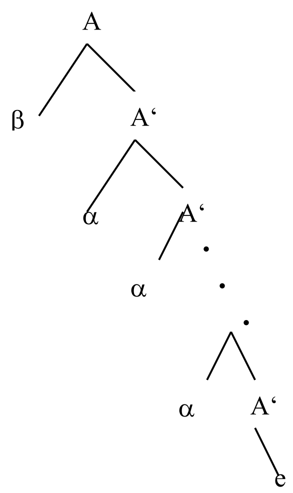
$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid i$

Alternativa odstranění s kratším výsledkem:

Ekvivalentní bude jak vidno z obr. i gramatika s pravidly

$$\boxed{\begin{aligned} A &\to \beta_1\, A' \mid \beta_2\, A' \mid \ldots \mid \beta_n\, A' \\ A' &\to \alpha_1\, A' \mid \alpha_2\, A' \mid \ldots \mid \alpha_m\, A' \mid e \end{aligned}}$$



Př.b) G[E] na tabuli.

E → E + T | T
T → T * F | F
F → (E) | i

Pro pohodlné:

Výsledek př.a)

E → T | T E′
E‵→ +T | + T E‵
T → F | F T′
T‵→ * F | * F T′
F → (E) | i

výsledek př.b)

E → T E′
E‵→ + T E‵ | e
T → F T′
T‵→ * F T′ | e
F → (E) | i

Odstranění levé rekurze (včetně nepřímé rekurze):

1. Zvolíme uspořádání na N = {A1, A2, ... An} tak, aby:
   je-li Ai → α pravidlo, jehož pravá strana začíná neterminálním symbolem Aj , pak j > i .
   Přiřaďme i = 1

2. Odstraníme přímou levou rekurzi u Ai pravidel (postup viz výše)

3. Je-li i = n, pak jsme získali výslednou G' a skonči

   Jinak přiřaď i = i + 1 ; j = 1

4. Každé, pravidlo tvaru Ai → Aj γ nahraď pravidly

   Ai → $\alpha_1 \gamma$ | $\alpha_2 \gamma$| … | $\alpha_p \gamma$, kde

   Aj → α_1 | α_2 | … | α_p jsou všechna Aj pravidla

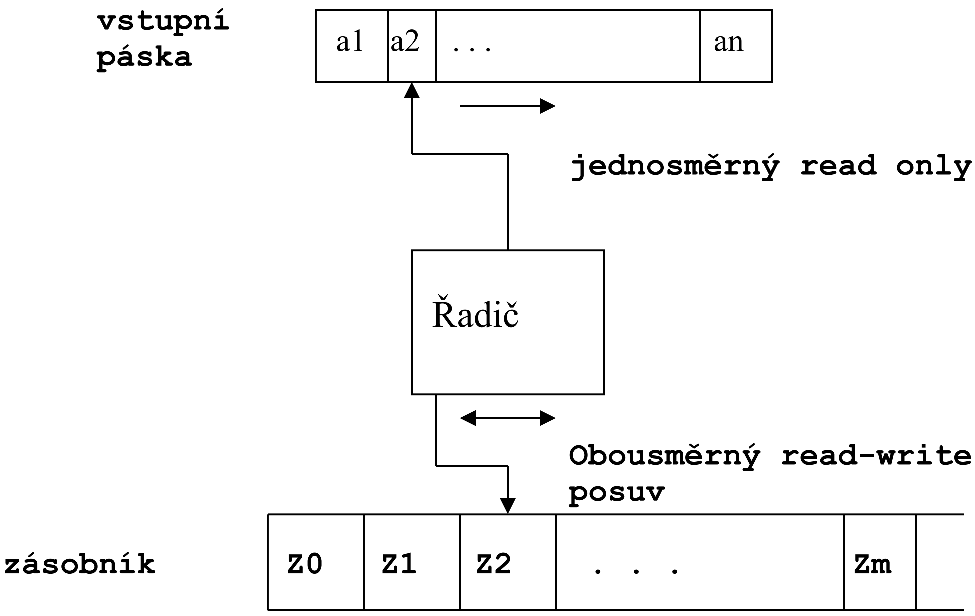
5. Je-li j = i - 1 jdi na krok 2., jinak j = j + 1 a jdi na 4.

Př. na tabuli s použitím kratší alt.: A → B C | a
B → C A | A b
C → A B | CC | a

1. Zvolíme uspořádání A_1 = A < A_2 = B < A_3 = C a přiřadíme i=1
2. Odstanime případnou levou rekurzi u A_1 pravidel(žádná tam není)
   Proto do výsledku jdou pravidla A → B C | a
3. i ≠ n, proto i=2 a j=1
4. Vnutíme uspořádání B pravidlům: B → C A | B C b | a b
5. j = i-1 proto dělej bod 2.
2. Teď z nich odstraníme přímou levou rekurzi. Do výsledku jde:
   B → C A B′ | a b B‘ B‘ → C b B‘ | e
3. i ≠ n, proto i=3 a j=1
4. Vnutíme uspořádání C pravidlům s ohledem na A:
   C → B C B | a B | CC | a
5. j ≠ i-1 proto j = 2 a dělej znovu bod 4.
4. Vnutíme uspořádání C pravidlům i vzhledem k B:
   C → C A B′C B | a b B‘ C B | a B | CC | a
5. j = i-1 proto dělej bod 2 a do výsledku půjde:
2. C → a b B‘ C B C‘| a B C‘| a C‘
   C‘→ A B′C B C‘| C C‘| e
3. Konec

Zásobníkové automaty

ZA je abstraktní model syntaktického analyzátoru BK jazyků
Obecně je:
- jednocestný,
- nedeterministický,
- s nekonečnou pamětí (zásobníkem)



Definice:

ZA $\mathscr{P} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
   1 2 3 4 5 6 7

1. Konečná množina stavů
2. Konečná vstupní abeceda
3. Konečná abeceda zásobníkových symbolů
4. Zobrazení δ: $Q \times (\Sigma \cup \{e\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$
5. Počáteční stav řadiče $q_0 \in Q$
6. Dno zásobníku $Z_0 \in \Gamma$
7. Množina koncových stavů $F \subset Q$

Konfigurace ZA $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$

Stav řadiče
Dosud nepřečtený vstup
Obsah zásobníku

Pozn.: Je to akceptační automat, ne překladový.

Přechod ZA je binární relace $\vdash$ nad množinou konfigurací, nebo její p-tou mocninou $\vdash^p$ či uzávěrem $\vdash^*$ a $\vdash^+$

$(q, aw, \alpha\beta) \vdash (p, w, \gamma\beta)$ jestliže $\delta(q, a, \alpha)$ obsahuje (p, γ), $a \in T \cup \{e\}$, $\alpha, \beta, \gamma \in \Gamma^*$

Př. na tabuli. Popsat $\mathscr{P}$ akceptující $L = \{ 0^n 1^n \}$ kde $n \geq 0$

Počáteční konfigurace ZA je (q_0, w, Z_0), kde $w \in \Sigma^*$

Interpretace zápisu přechodové fce

$\delta(q, a, \alpha) = \{ (p1, \gamma 1), (p2, \gamma 2), \ldots, (pn, \gamma n) \}$

ZA ve stavu q, se vstupním symbolem a, vrcholovým řetězcem zásobníku α, přejde do některého ze stavů pi a vrchol α nahradí příslušným řetězcem $\gamma i \in \Gamma^*$.

Přechod bez čtení vstupního symbolu (e-přechod)

$\delta(q, e, \alpha) = \{ (p1, \gamma 1), (p2, \gamma 2), \ldots, (pn, \gamma n) \}$

Def.

Rozšířený ZA (RZA) je sedmice $\mathscr{P} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

kde $\delta: Q \times (\Sigma \cup \{e\}) \times \Gamma^* \rightarrow Q \times \Gamma^*$

tj. reaguje na vrcholové řetězce zásobníku

Př. popsat $\mathscr{P}$ akceptující $L = \{ w w^R \}$ kde $w \in \{a, b\}^*$

R má význam „reverzní"

$\mathscr{P} = (\{q,p\}, \{a, b\}, \{a,b,S,Z\}, \delta, q, Z, \{p\})$

cokoliv

$\delta(q,a,e) = \{(q, a)\}$ **názornější zápis by byl $\delta(q,a,-)$** cokoliv

$\delta(q,b,e) = \{(q, b)\}$ „---------„ **$\delta(q,b,-)$**

$\delta(q,e,e) = \{(q, S)\}$ to je e-přechod, vloží střed S **$\delta(q,-,-)$**

$\delta(q,e,aSa) = \{(q, S)\}$ to je e-přechod **$\delta(q,e,aSa)$**

$\delta(q,e,bSb) = \{(q, S)\}$ to je e-přechod **$\delta(q,e,bSb)$**

$\delta(q,e,ZS) = \{(p, e)\}$

Např akceptace věty abba

$(q, abba, Z) \vdash (q, bba, Za) \vdash (q, ba, Zab) \vdash (q, ba, ZabS)$
$\vdash (q, a, ZabSb) \vdash (q, a, ZaS) \vdash (q, e, ZaSa) \vdash (q, e, ZS)$
$\vdash (p, e, e)$

Def.

Věta w jazyka může být akceptována zásobníkovým automatem $\mathscr{P} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ dvojím způsobem:

a) přechodem do koncového stavu

$L(\mathscr{P}) = \{ w: (q_0, w, Z_0) \vdash^* (q, e, \gamma), \gamma \in \Gamma^*, q \in F, w \in \Sigma^*\}$

b) s prázdným zásobníkem

$L_e(\mathscr{P}) = \{ w: (q_0, w, Z_0) \vdash^* (q, e, e), q \in Q, w \in \Sigma^*\}$

Vztah bezkontextových gramatik a zásobníkových automatů

Pro danou BKG $G = (N, T, P, S)$ můžeme sestrojit Z A $\mathscr{P}$ takový, že $L(G) = L(\mathscr{P})$. Platí to i opačně.

A.
Konstrukce ZA, který je modelem syntaktické analýzy metodou shora dolů.

$\mathscr{P} = (\{q\}, T, N \cup T, \delta, S, \emptyset)$, kde δ je definováno takto:

1. $\delta(q, \text{cokoliv}, A) = \{ (q, \alpha) : A \rightarrow \alpha \in P\}$ pro $\forall A \in N$,
2. $\delta(q, a, a) = \{ (q, e) \}$ pro $\forall a \in T$.

Pozn.: Argument „cokoliv" znamená, že se jedná o e-přechod

Operaci 1. nazýváme expanzí (nahradí na vrcholu zásobníku a tím i ve větné formě neterminální symbol některou jeho pravou stranou).

Operaci 2. nazýváme srovnáním (čteného vstupního symbolu a symbolu z vrcholu zásobníku).

Tento ZA má vrchol zásobníku vždy vlevo.

Př. Zapsat $\mathscr{P}$ pro G[E] (na tabuli)

$\mathscr{P} = (\{q\}, \{(,),+,*,a\}, \{E,T,F,(,),+,*,a\},\delta, q, E, \emptyset)$
$\delta(q, \text{cokoliv}, E) = \{(q, E+T), (q, T)\}$
$\delta(q, \text{cokoliv}, T) = \{(q, T*F), (q, F)\}$
$\delta(q, \text{cokoliv}, F) = \{(q, (E)), (q, a)\}$
$\delta(q,a',a') = \{(q, e)\}$ pro $\forall a' \in \{(,),+,*,a\}$

Např. zpracování věty a+a
Tady je vrchol zásobníku
$(q, a+a, E) \vdash (q, a+a, E + T) \vdash (q, a+a, T + T)$ to jsou expanze
$\vdash (q, a+a, F + T) \vdash (q, a+a, a + T)$ teď provedeme 2krát srovnání
$\vdash (q, +a, + T) \vdash (q, a, T)$ a opět expandujeme
$\vdash (q, a, F) \vdash (q, a, a)$ a naposledy srovnáme $\vdash (q, e, e)$

Zásobník je po přečtení vstupního řetězce prázdný, takže řetězec byl akceptován

B.
Konstrukce ZA, který je modelem syntaktické analýzy metodou zdola nahoru.

$\mathcal{P}$ = ({q, r}, T, N ∪ T ∪ {#}, δ, q, #, {r}), kde δ je definováno takto:

1. δ(q, a, cokoliv) = { (q, a) } pro ∀a ∈ T,
2. δ(q, cokoliv, α) = { (q, A) : A → α ∈ P },
3. δ(q, e, #S) = { (r, e) }.

Operaci 1. nazýváme přesun (přesun vstupního symbolu na vrchol zásobníku).

Operaci 2.nazýváme redukce (náhrada pravé strany pravidla na vrcholu zásobníku a tím i ve větné formě stranou levou).

Operace 3. je přijetí.

Tento ZA má vrchol zásobníku vpravo.
Konfiguraci budeme zapisovat ve tvaru: (stav, zásobník, vstup).
Zřetězením stavu zásobníku se zbytkem vstupu pak uvidíme jednotlivé větné formy

Př. Zapsat $\mathcal{P}$ pro G[E] (na tabuli)

$\mathcal{P}$ = ({q,r}, {(,),+,*,a}, {#,E,T,F,(,),+,*,a},δ, q, #, r)
δ(q, a',cokoliv) = { (q, a') } pro ∀a ∈ T,
δ(q, cokoliv, E+T) = { (q, E) }
δ(q, cokoliv, T) = { (q, E) }
δ(q, cokoliv, T*F) = { (q, T) }
…
δ(q, e, #E) = { (r, e) }

Např. zpracování věty a+a
Vrchol
(q, #, a+a) ├ (q, #a, +a) ├ (q, #F, +a) ├ (q, #T, +a)
├ (q, #E, +a) ├ (q, #E+, a) ├ (q, #E+a, e) ├ (q, #E+F, e)
├ (q, #E+T, e) ├ (q, #E, e) ├ (r, e, e)

ZA konstruované dle A. i B. jsou obecně nedeterministické (nepoužitelné pro SA). Pro konstrukci SA lze použít buď:

a) Deterministickou simulaci nedeterministického ZA = algoritmus syntaktické analýzy s návraty.
b) Zdokonalit konstrukci ZA tak, aby byl pro určitou třídu BKG deterministický.

Pozn.: Obsah zásobníku zřetězený se zbytkem vstupu je větnou formou.

Řešení příkladů

Př. eliminace zbytečných symbolů

G[S]: S → a | A   A → A B   B → b

Krok1.{S, B} tj. vypadne A

Krok2.{S, a}

Vysledek je S → a

Krok2.{S,A,B,a,b}

Krok1.{S, B} tj. Vypadne A

Vysledek je S → a, B → b

Př.Odstraňte prázdná pravidla z G[S]: S → a S b S | e

Ne = { S } proto modifikujeme pravidla s S na pravé straně všemi možnými vypuštěními S

Výsledek: S´ → S | e

S → a S b S | a b S | a S b | a b

Př.Odstranění jednoduchých pravidel

G[E]

E → T | E + T

T → F | T * F

F → (E) | i

Řešení   E → T * F |(E) | i | E + T

T → (E) | i | T * F

F → (E) | i

Cocke-Younger-Kasami algoritmus analýzy BK jazyků

Vychází z Chomského normální formy

Chomského normální forma (CNF)

Přepisovací pravidla jsou pouze v některém z tvarů:

1. X $\rightarrow$ Y Z
2. X $\rightarrow$ a
3. S $\rightarrow$ ε pokud jazyk obsahuje prázdný řetězec. *S* nesmí být na pravé straně žádného přepisovacího pravidla.

Algoritmus převodu na gramatiku v CNF

Vstup: upravená (vlastní) gramatika bez jednoduchých pravidel G =(N, T, P, S)

Výstup: Gramatika v CNF G'=(N', T, P', S)

1. Množina P' obsahuje všechny pravidla z P ve tvaru $A \rightarrow BC$ a $A \rightarrow a$, případně $S \rightarrow \varepsilon$.
2. Každé pravidlo tvaru $A \rightarrow X_1X_2\ldots X_k$ ($k > 2$) přepíšeme do P' jako sérii pravidel:
   $A \rightarrow X_1X_2'$
   $X_2' \rightarrow X_2X_3'$
   …
   $X_{k-1}' \rightarrow X_{k-1}X_k$
   A zavedeme nové neterminální symboly $X_2'\ldots X_{k-1}'$ (tj. pravidla mající pravou stranu delší než dva prvky rozdělíme na sérii pravidel, která mají napravo jen dva prvky)
3. Ve všech pravidlech v P' které nejsou ve tvaru $A \rightarrow a$, kde se vyskytují na pravé straně terminální symboly, nahradíme tyto terminály za nové neterminály a přidáme pravidlo přepisující tyto nové neterminály na původní terminály (tj. nepovolená pravidla s terminály na pravé straně upravíme tak, že terminál nahradíme neterminálem, který se přepisuje na původní terminál).

Př. Mějme G[S]:

S → a X b X
X → a Y | b Y | ε
Y → X | c

Odstraníme prázdná pravidla (N_e = { X, Y})
S → a X b X | a b X | a X b | a b
X → a Y | b Y | a | b
Y → X | c
Odstraníme jednoduchá pravidla
S → a X b X | a b X | a X b | a b
X → a Y | b Y | a | b
Y → a Y | b Y | a | b | c

Převedeme do CNF
S → E F | A F | E B | A B
X → A Y | B Y | a | b
Y → A Y | B Y | a | b | c
E → A X
F → B X
A → a
B → b
C → c

Idea algoritmu CYK:
Bottom Up Parsing, Dynamické programování – rozklad na subproblémy = parsing substrings
<u>Algoritmus (stručně)</u>:

Dán řetězec *s* délky *N*
Pro *k* = 1 .. *N* dělej
Pro každý <u>**subřetěz**</u> délky *k*
Urči, které neterminály jej mohou derivovat a zapiš je do tabulky.

<u>**Algoritmus CYK** (detailně)</u>:

Vstup: řetězec $w = w_1 w_2 \ldots w_n$
Metoda:

Jestliže w = ε, a S → ε je pravidlem, pak akceptuj

Pro i = 1 . . n dělej

Pro každý neterminal A dělej

Jestliže A → b je pravidlo, kde $b = w_i$

pak přidej A do buňky X(i, i)

Pro l = 2 . . n dělej

Pro i = 1 . . (n – l + 1) dělej

j = i + l – 1

Pro k = i . . (j – 1) dělej

Pro každé pravidlo tvaru A → BC dělej

Jestliže X(i,k) obsahuje B a X(k+1, j) obsahuje C
pak přidej A do buňky X(i, j)

Jestliže S je obsaženo v X (1,n) pak akceptuj, jinak odmítni.

Konstruujeme trojúhelníkovou matici X, řádky odpovídají subřetězcům:

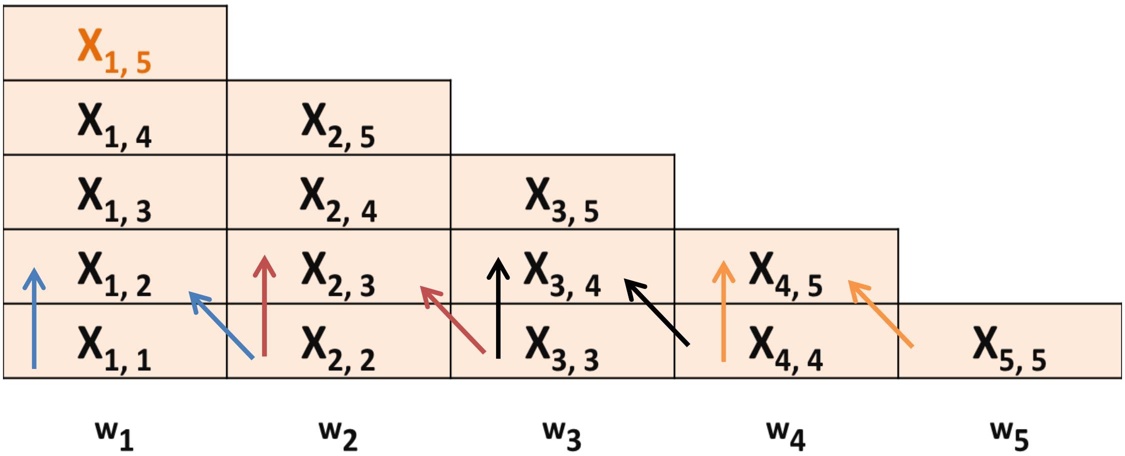
- Dolní řádek – řetězec délky 1
- Druhý odspoda - řetězec délky 2

:

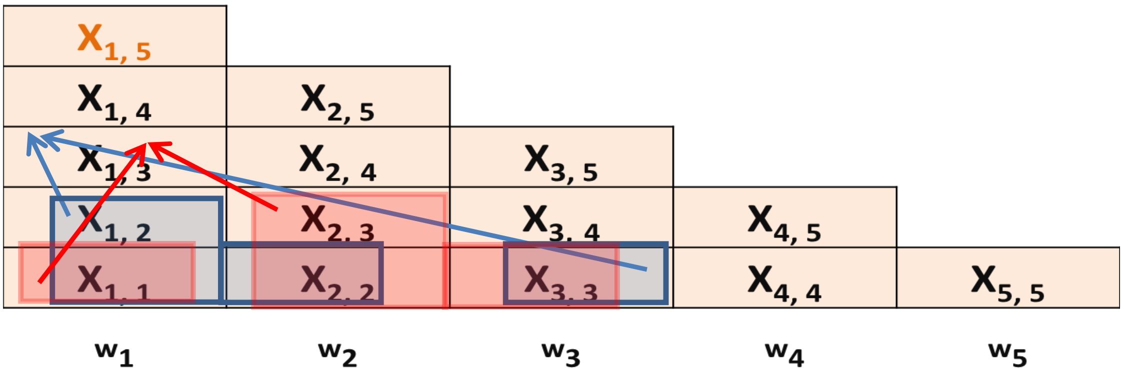
- Vrcholový – odpovídá řetězci w

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
w_1	w_2	w_3	w_4	w_5

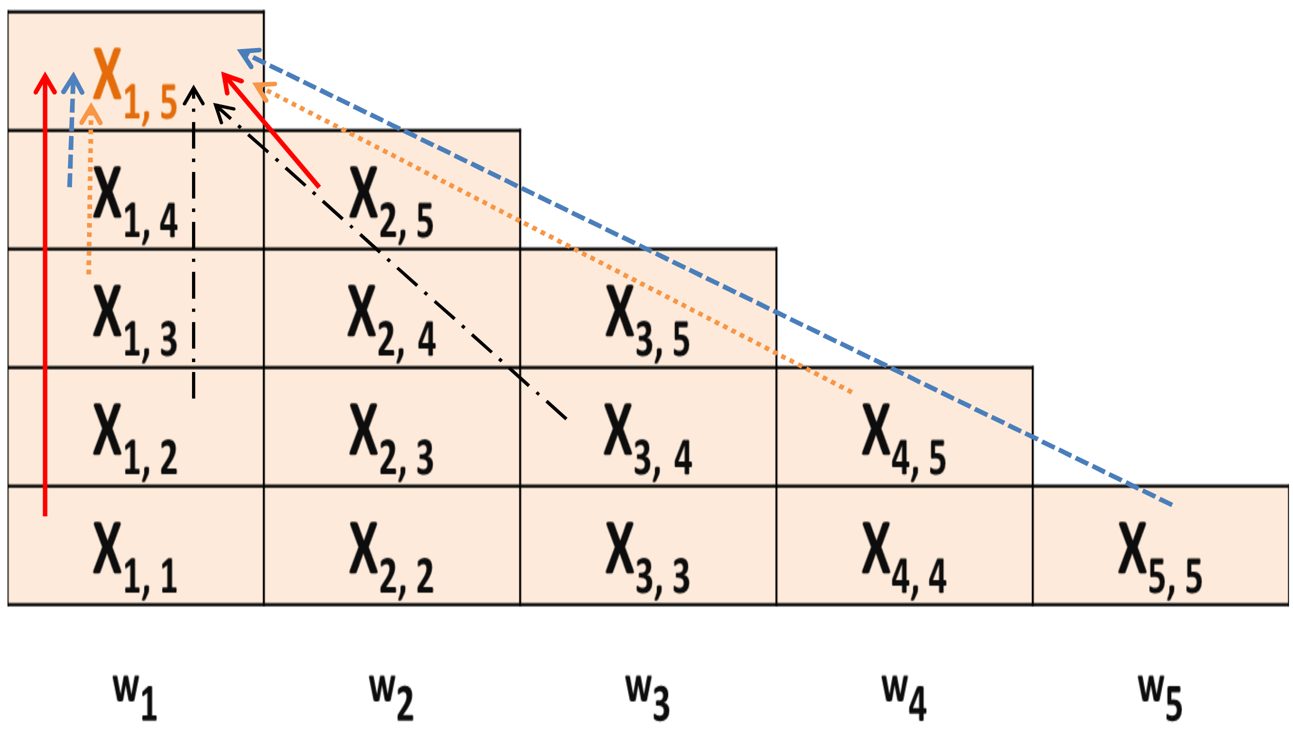
Tabulka pro w délky 5



Porovnávané dvojice při konstrukci druhé řádky



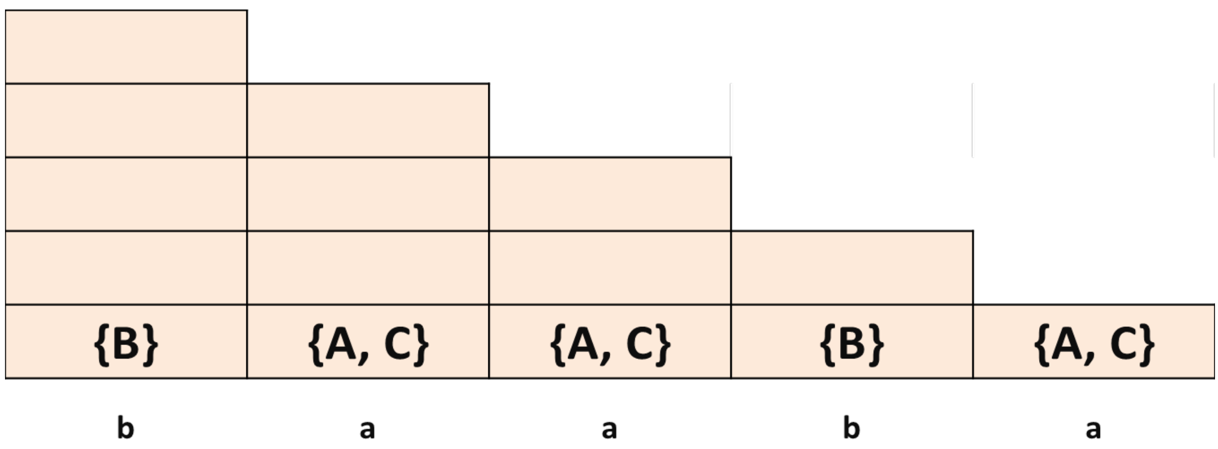
Porovnávané dvojice při konstrukci třetí řádky (prvku $X_{1,3}$)



Porovnávané dvojice při konstrukci vrcholové hodnoty

Př.

- CNF gramatika **G[S]**
  - S → AB | BC
  - A → BA | a
  - B → CC | b
  - C → AB | a
- **w** má tvar b a a b a . Je **baaba** v L(G)?
- **Konstruujeme trojůhelníkovou matici, vyplnění spodní řádky**



Dále počítáme

- $\mathbf{X_{1,2} = (X_{i,i}, X_{i+1,j}) = (X_{1,1}, X_{2,2})}$
- → {B} {A,C} = {BA, BC}
- Kroky:
  - Najdi pravidla generující BA nebo BC
  - Jsou dvě: S a A
  - $\mathbf{X_{1,2} = \{S, A\}}$

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

{S, A} 1,2				
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Dále $\mathbf{X_{2,3}}$

- $\mathbf{X_{2,3} = (X_{i,i}, X_{i+1,j}) = (X_{2,2}, X_{3,3})}$
- → {A, C} {A,C} = {AA, AC, CA, CC} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Je jen jedno: B
  - $\mathbf{X_{2,3} = \{B\}}$

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

{S, A}	{B} 2,3			
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Dále $X_{3,4}$

- $X_{3,4} = (X_{i,i}, X_{i+1,j}) = (X_{3,3}, X_{4,4})$
- ➔ {A, C} {B} = {AB, CB} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Jsou dvě: S a C
  - $X_{3,4} = \{S, C\}$

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

{S, A}	{B}	{S, C}		
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

Dále $X_{4,5}$

- $X_{4,5} = (X_{i,i}, X_{i+1,j}) = (X_{4,4}, X_{5,5})$
- ➔ {B} {A, C} = {BA, BC} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Jsou dvě: S a A
  - $X_{4,5} = \{S, A\}$

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

{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

Dále $\mathbf{X_{1,3}}$

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

- $\mathbf{X_{1,3} = (X_{i,i}, X_{i+1,j}) \ U \ (X_{i,i+1}, X_{i+2,j})}$

  $\mathbf{= (X_{1,1}, X_{2,3}) \ U \ (X_{1,2}, X_{3,3})}$
- ➔ {B}{B} **U** {S, A}{A, C}= {BB, SA, SC, AA, AC} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Nejsou žádná: S a A
  - $\mathbf{X_{1,3} = Ø}$

Ø				
{S, A}	**{B}**	**{S, C}**	**{S, A}**	
{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Další $\mathbf{X_{2,4}}$

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

- $\mathbf{X_{2,4} = (X_{i,i}, X_{i+1,j}) \ U \ (X_{i,i+1}, X_{i+2,j})}$

  $\mathbf{= (X_{2,2}, X_{3,4}) \ U \ (X_{2,3}, X_{4,4})}$
- ➔ {A, C}{S, C} **U** {B}{B}= {AS, AC, CS, CC, BB} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Je jen jedno: B
  - $\mathbf{X_{2,4} = \{B\}}$

Ø	**{B}**			
{S, A}	**{B}**	**{S, C}**	**{S, A}**	
{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Další $\mathbf{X_{3,5}}$

- $\mathbf{X_{3,5} = (X_{i,i}, X_{i+1,j})\ U\ (X_{i,i+1}, X_{i+2,j})}$

  $\mathbf{= (X_{3,3}\ , X_{4,5})\ U\ (X_{3,4}\ , X_{5,5})}$
- ➔ {A,C}{S,A} **U** {S,C}{A,C}

  = {AS, AA, CS, CA, SA, SC, CA, CC} = Y
- Kroky:
  - Najdi pravidla generující Y
  - Je jen jedno: B
  - $\mathbf{X_{3,5} = \{B\}}$

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

Ø	**{B}**	**{B}**		
{S, A}	**{B}**	**{S, C}**	**{S, A}**	
{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Další $\mathbf{X_{1,4}}$

- $\mathbf{X_{1,4} = (X_{1,1}, X_{2,4})\ U\ (X_{1,3}, X_{4,4})\ U\ (X_{1,2}, X_{3,4})}$

  ➔ {B}{B} **U** Ø {B}

  = {BB, B } = Y
- Kroky:
  - Najdi pravidla generující Y
  - Není žádné
  - $\mathbf{X_{1,4} = Ø}$

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

Ø				
Ø	**{B}**	**{B}**		
{S, A}	**{B}**	**{S, C}**	**{S, A}**	
{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Další $X_{2,5}$

- $X_{2,5}$ = ($X_{2,2}$, $X_{3,5}$) U ($X_{2,4}$, $X_{5,5}$) U ($X_{2,3}$, $X_{4,5}$)

  ➔ {AB, CB} U {BA, BC} U {BS, BA} = Y

  Kroky:
  - Najdi pravidla generující Y
  - Jsou tři
  - $X_{2,5}$ = {S, A, C }

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

Ø	{S, A, C}			
Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Poslední $X_{1,5}$

$X_{1,5}$ = ($X_{1,1}$, $X_{2,5}$) U ($X_{1,4}$, $X_{5,5}$) U ($X_{1,3}$, $X_{4,5}$) U ($X_{1,2}$, $X_{3,5}$)

➔ {BS, BA, BC} U {A, C} U {S, A} U {SB, AB} = Y

Kroky:
- Najdi pravidla generující Y
- Jsou tři
- $X_{1,5}$ = {S, A, C }

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

{S, A, C}				
Ø	{S, A, C}			
Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

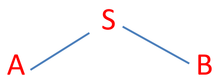
Výsledek: **<u>$X_{1,5}$ Obsahuje S, věta proto patří do jazyka</u>**

Jak získat derivační strom?

Pro získání stromu je třeba si pamatovat, jak vznikaly jeho uzly.

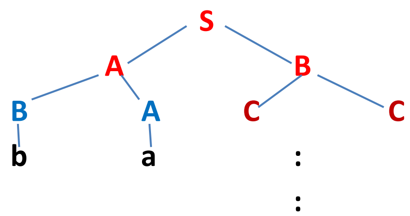
Jdeme postupně od kořene S dolů:

- S bylo získáno z A B



{ S, A, C }				
Ø	{B}	{ B }		
{ S, A	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

- A bylo získáno z B A
- B bylo získáno z C C
- B bylo získáno z b
- A bylo získáno z a
- C bylo získáno z A B
- C bylo získáno z a
- A bylo získáno z a
- B bylo získáno z b



dokreslit strom