Bezkontextové gramatiky

Def. 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

   $A \to \alpha$, kde $A \in N$, $\alpha \in (N \cup T)^*$

Bezkontextový jazyk

Def. 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, ... „ $N \cup T$

α, β, γ, ... „ 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 $A \to \gamma \in P$
- 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 $A \to \alpha$

Př. G[E]

$E \to T \mid E + T$

$T \to F \mid T * F$

$F \to (E) \mid i$

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:

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

- **VĚTA**

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

$$S \Rightarrow^* \alpha, \text{ kde } \quad \alpha \in T^*$$

- **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

$$S \Rightarrow^* \alpha A \gamma \quad \text{a} \quad A \Rightarrow^* \beta$$

Tzn.frázi tvoří listy podstromu derivačního stromu.

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

$$S \Rightarrow^* \alpha A \gamma \quad \text{a} \quad A \Rightarrow \beta$$

- **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 kubická)

-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

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.

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

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 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: $\forall$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,

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

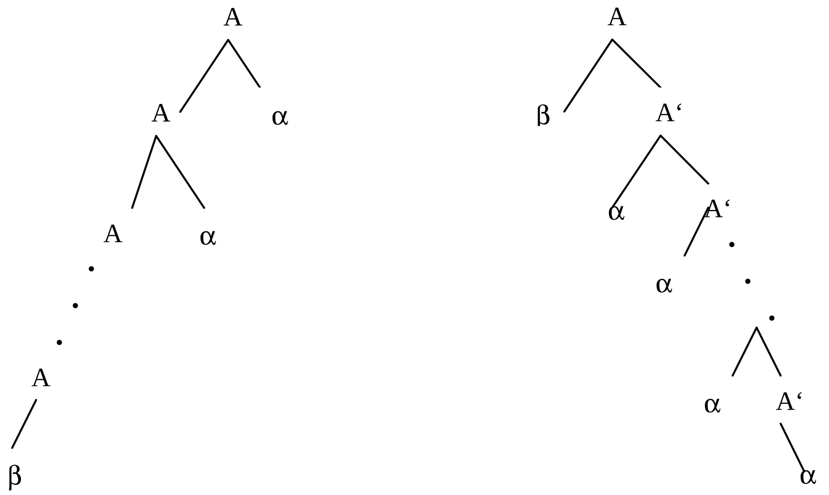
jsou všechna A pravidla v P a žádné, z β nezačíná A.

Pak G' = (N $\cup$ { 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 → E + T | T ?proč nepoužijeme E → T + E

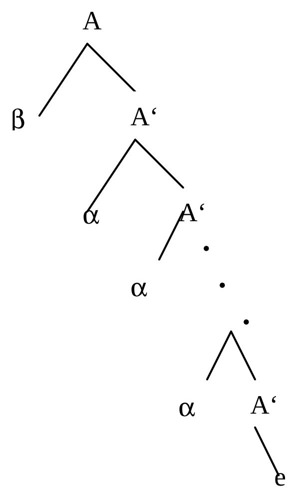
T → T * F | F

F → (E) | i

Alternativa odstranění s kratším výsledkem:

Ekvivalentní bude jak vidno z obr. i gramatika s pravidly

$$\boxed{\begin{array}{l} A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_n A' \\ A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_m A' \mid e \end{array}}$$



Př.b) G[E] na tabuli.

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

Pro pohodlné:

Výsledek př.a)

$E \rightarrow T \mid T E'$
$E' \rightarrow +T \mid + T E'$
$T \rightarrow F \mid F T'$
$T' \rightarrow * F \mid * F T'$
$F \rightarrow (E) \mid i$

výsledek př.b)

$E \rightarrow T E'$
$E' \rightarrow + T E' \mid e$
$T \rightarrow F T'$
$T' \rightarrow * F T' \mid e$
$F \rightarrow (E) \mid i$

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 .
   Přiřadme 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

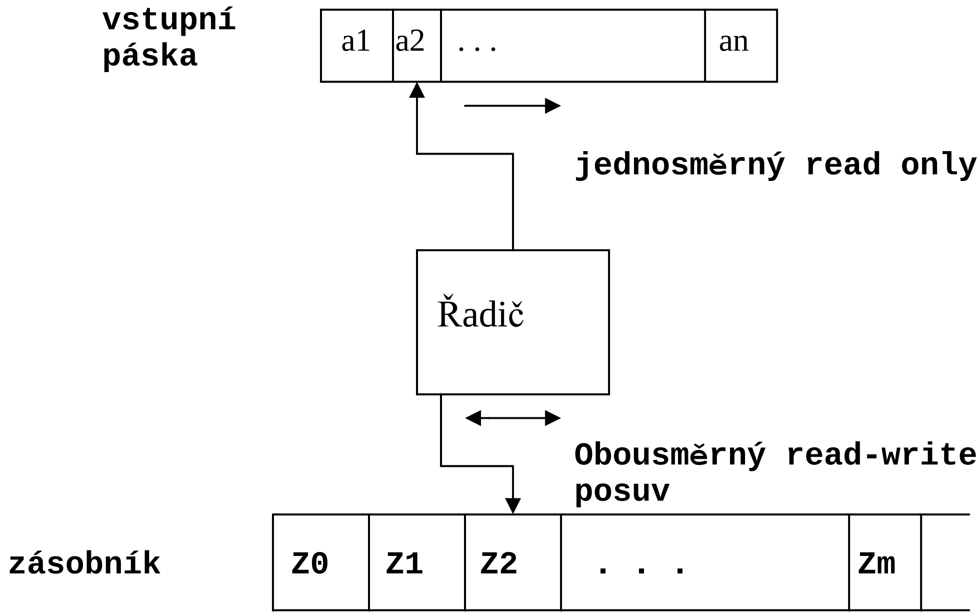
5. Je-li j = i - 1 jdi na krok 2., jinak j = j + 1 a jdi na 4.

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**
$(\ p,\ \gamma\ )$, $a \in T \cup \{e\}$, $\alpha,\ \beta,\ \gamma \in \Gamma^*$
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

$\delta(q, a, b) = \{ (p1, \gamma 1), (p2, \gamma 2), \ldots, (pn, \gamma n) \}$
ZA ve stavu q, se vstupním symbolem a, vrcholovým řetězcem zásobníku b, 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

$\delta(q,a,e) = \{(q, a)\}$
$\delta(q,b,e) = \{(q, b)\}$
$\delta(q,e,e) = \{(q, S)\}$ to je e-přechod, vloží střed S
$\delta(q,e,aSa) = \{(q, S)\}$ to je e-přechod
$\delta(q,e,bSb) = \{(q, S)\}$ to je e-přechod
$\delta(q,e,ZS) = \{(p, e)\}$

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 ∪ T, δ, S, Ø), kde δ je definováno takto:

1. δ(q, e, A) = { (q, α): A → α ∊ P} pro ∀ A ∊ N,
2. δ(q, a, a) = { (q, e) } pro ∀a ∊ T.

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},δ, q, E, Ø)
δ(q, e, E) = {(q, E+T), (q, T)}
δ(q, e, T) = {(q, T*F), (q, F)}
δ(q, e, F) = {(q, (E)), (q, a)}
δ(q,a‘,a‘) = {(q, e)} pro ∀a’ ∊ {(,),+,*,a}

Např. zpracování věty a+a
Tady je vrchol zásobníku
(q, a+a, E) ├ (q, a+a, E + T) ├ (q, a+a, T + T) to jsou expanze
├ (q, a+a, F + T) ├ (q, a+a, a + T) teď provedeme 2krát srovnání
├ (q, +a, + T) ├ (q, a, T) a opět expandujeme
├ (q, a, F) ├ (q, a, a) a naposledy srovnáme ├ (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, e) = { (q, a) } pro ∀a ∊ T,
2. δ(q, e, 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', e) = { (q, a') } pro ∀a ∊ T,
δ(q, e, E+T) = { (q, E) }
δ(q, e, T) = { (q, E) }
δ(q, e, 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.