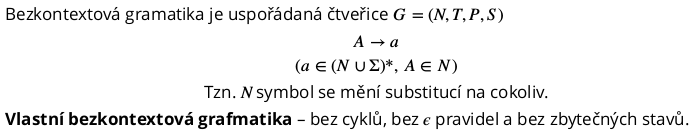
**BI-SPOL-3 Bezkontextové jazyky: Bezkontextové gramatiky, zásobníkové automaty a jejich varianty. Modely syntaktické analýzy bezkontextových jazyků**

BI-AAG

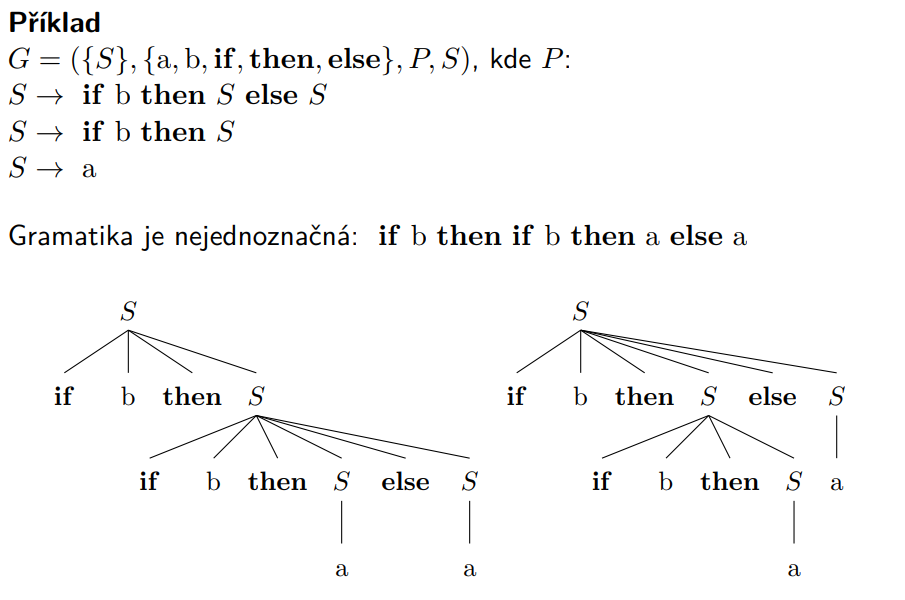
**Bezkontextové jazyky**

* lze generovat bezkontextovou gramatikou nebo přijmout nedeterministickým zásobníkovým automatem

### Bezkontextové gramatiky



* dokáží popsat převážnou většinu syntaktických struktur programovacích jazyků
* nejednoznačná, jestli existuje věta w ∈ L(G) taková, že je výsledkem nejméně dvou různých derivačních stromů, jinak je jednoznačná
* ryze zbytečné uzly nejsou schopny vygenerovat větu
* nedostupné uzly



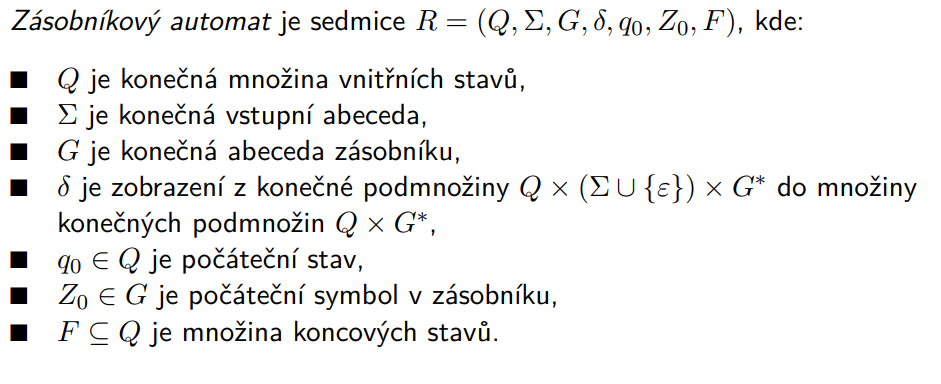
**Derivační strom**

* grafické zobrazení syntaktické struktury větné formy, kde kořenem je počáteční neterminál, vnitřní uzly jsou neterminály, listy naopak terminály

Chomského normální tvar BG

* používá se v algoritmu CYK, který dokáže určit jestli daný řetězec patří do jazyka
* nemá jednoduchá pravidla (např. A→B)
* je vlastní gramatika:
  + nesmí mít cyklus (jednoduchá pravidla a ε-přechody)
  + nemá ε-pravidla
  + nemá zbytečné stavy (ryze zbytečné + nedostupné)

### Zásobníkové automaty



* ze stavu Q , na vstup Σ nebo ε čtu ze zásobníku, jdu do stavu a přidávám do zásobníku (tzn. mám nějaký zásobník, kde si můžu něco ukládat)
* podle hodnoty na zásobníku se mohou definovat přechody v daném stavu (0, in|out)
* v počáteční konfiguraci je zásobník prázdný ((q0, w, Z0), w ∈Σ∗)
* koncový stav se liší dle typu

**Typy ZA**

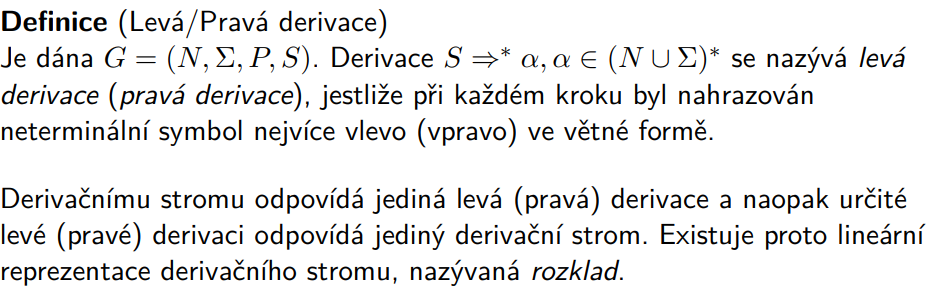
* přijímání prázdným zásobníkem (věta zpracována, zásobník je prázdný a nachází se v jakémkoliv stavu) - (kdekoliv, ε, ε)
* přijímání koncovým stavem (věta zpracována, zásobník nemusí být prázdný a nachází se v koncovém stavu) - (qF, ε, cokoliv)
* deterministický/nedeterministický

### Syntaktická analýza

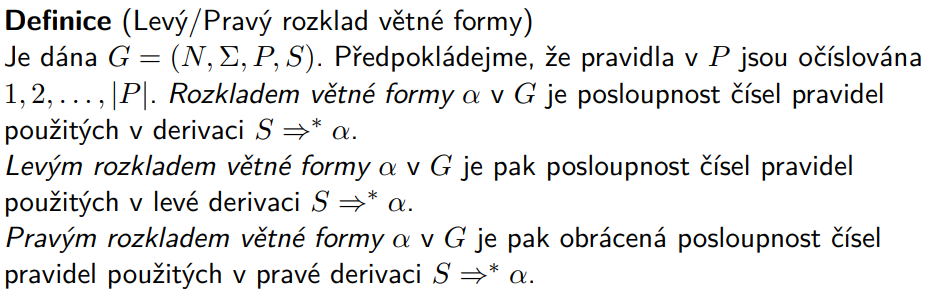
* vytváří NZA
* levý a pravý rozklad



**Derivace**

****

**Rozklad**



**Levý rozklad a levá derivace**

* Při expanzi se nahrazuje neterminál nejvíc vlevo.
* Pravidla zapisujeme v pořadí – posloupnost 1,3,2 je levý rozklad.



**Shora dolů**

* pro nalezení levého rozkladu
* strom, který vychází z počátečního stavu S, expanduje a dostáváme řetězec
* vrchol zásobníku vlevo
* fáze: expanze - expanze neterminálu v zásobníku dle přechodu

srovnání - na vrcholu zásobníku je stejná sekvence terminálů, jako má slovo (začínají stejně), takže můžeme srovnat, že odebereme stejné terminály

**Pravý rozklad a pravá derivace**

* Při expanzi se nahrazuje neterminál nejvíc vpravo.
* Pravidla zapisujeme v opačném pořadí, aby vznikl pravý rozklad – posloupnost 1,3,2 je pravý rozklad .Protože postupujeme vlastně od hotového řetězce a "tipujeme"

pravidla, kterými se můžeme dostat do S.



**Zdola nahoru**

* pro nalezení pravého rozkladu
* vycházíme z listů, které spojujeme do pravidel, na základě kterých mohly vzniknout (tzn. tvoří se strom od listů po kořen S)
* vrchol zásobníku vpravo, proto to tam dáváme obráceně vlastně
* fáze: přesun - přesouvá jeden nejlevější terminál na zásobník

redukce - zásobník má posloupnost terminálů, které můžeme zredukovat

dle pravidel

* slovo přijato, až na zásobníku zůstane |S