SMR

Strukturální metody rozpoznávání

KKY/SMR

KATEDRA KYBERNETIKY

Doc. Ing. Miloš Železný, Ph.D.

UK509 (377 63 2548)

E-mail: zelezny@kky.zcu.cz

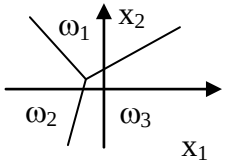
OBSAH

1 ZÁKLADNÍ POJMY A METODY ROZPOZNÁVÁNÍ

1.1 Příznakové metody

Obrazy charakterizovány vektorem naměřených hodnot – příznaků $X = [x_1,\ldots,x_N]$

Každému obrazu odpovídá bod v N-dimenzionálním euklidovském prostoru. Vyjdeme-li z předpokladu, že body odpovídající obrazům stejné třídy budou ležet blízko sebe, princip úlohy klasifikace spočívá v rozdělení prostoru na tolik částí, kolik je tříd.



Postupy používané v rámci příznakových metod vycházejí z těchto teorií:

- ***Statistická klasifikace*** – příznaky jsou považovány za náhodné proměnné. Předpokládáme, že pro každou třídu ω_j je známá N-rozměrná hustota psti $p(x/\omega_j)$ a apriorní pst $P(\omega_j)$. Jedná se o statistický rozhodovací problém, přičemž cílem je minimalizovat pst chybné klasifikace.

  Nejznámější metody jsou založeny na Bayesovském pravidle – mluví se o tzv. Bayesovské klasifikaci. V praxi obvykle nejsou známy parametry hustot pstí jednotlivých tříd a k dispozici je pouze znalost počtu tříd a omezená množina vzorků, reprezentovaných vektory naměřených hodnot příznaků, tzv. trénovací množina. V těchto případech se využívá různých estimačních a samoučících se algoritmů.

- ***Shluková analýza*** – nepředpokládá apriorní znalost o třídách (shlucích), do nichž jsou jednotlivé obrazy zařazovány. Shlukování představuje neparametrický samoučící se proces. Hledáme takový rozklad množiny obrazů do jednotlivých shluků, který je optimální vzhledem k jistému kritériu optimality, které obvykle oceňuje některou z těchto vlastností:

  - vzájemnou podobnost obrazů uvnitř shluků
  - míru separace shluků
  - homogenitu rozložení obrazů uvnitř shluků

  Shlukovací algoritmy dělíme:

  - hierarchické
    - aglomerativní (1)
    - divizní (2)
  - nehierarchické
    - optimalizační (3)
    - analýzy módů (4)

  (1): vycházíme od jednotlivých obrazů, které postupně shlukujeme
  (2): vycházíme od celé množiny, kterou postupně dělíme
  (3): optimalizace kriteria
  (4): definuje shluky na základě existence a polohy módů pstní fce a vycházejí z pstního pojetí

- ***Diskriminační fce*** – metody využívající diskriminačních fcí nepředpokládají znalost rozložení. Známe počet tříd a máme k dispozici trénovací množinu se známou klasifikací – $X_1..X_n$. V učícím se procesu hledáme parametry zvolené diskr. fce analýzou trénovací množiny.

- ***Fuzzy množiny*** – metody založené na využití fuzzy množin představují alternativu k pravděpodobnostnímu přístupu. Tyto metody dávají často lepší výsledky v případě, kdy schází apriorní znalost a pravděpodobnostní přístup nemůže být použit.

1.2 Strukturální metody, syntaktické, lingvistické

Jednotlivé obrazy jsou reprezentovány pomocí množiny základních popisných elementů – tzv. **primitiv**, jejich vlastností a vztahů mezi nimi tak, že vzniklé popisy vystihují jejich podstatné strukturální vlastnosti.

Využívá se analogie mezi **strukturou** obrazů a **syntaxí**, resp. **gramatikou jazyka**. Obrazy se skládají z jednodušších částí podobně jako se věty skládají ze slov, slova ze slabik atd. Výhoda strukt. přístupu k rozpoznávání je v tom, že kromě klasifikace do tříd umožňuje i získání popisu struktury analyzovaného obrazu – jeho částí a vztahů mezi nimi.

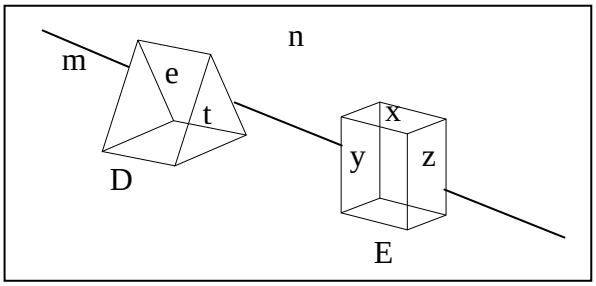
Obě metody (příznakové i strukturální) mají své výhody i nevýhody a spíše se vzájemně doplňují, než si konkurují. Proto se v praxi používá obou – strukturálních metod pro získání strukt. popisu celku a příznakových většinou pro rozpoznání jednotlivých primitiv.

1.3 Motivace strukturálního přístupu rozpoznávání

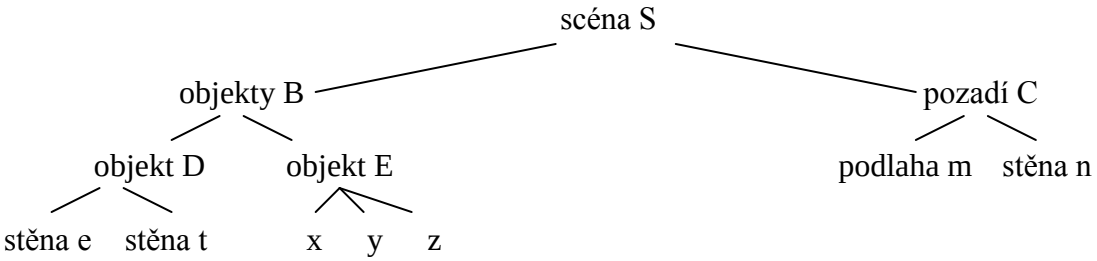
- V některých úlohách (např. identifikace otisků prstů nebo tváří, rozpoznávání spojité řeči či čínských znaků) je možných popisů tolik, že je neúnosné považovat každý popis za definici jedné třídy. V takových úlohách je požadavek rozpoznání splněn spíše nalezením vhodného popisu každého analyzovaného obrazu než vyřešením úlohy klasifikace do tříd.
- V případě velmi složitých obrazů (např. analýza scény) může být informace o jejich struktuře natolik podstatná, že bez jejího využití nemůže být dosaženo žádoucích výsledků. Navíc počet příznaků v takových úlohách by byl neúnosný pro příznakové metody.
- Pro „inteligentní“ chování (rozhodování) robota je nezbytným předpokladem popis analyzovaného obrazu ve vhodném jazyce a proto strukt. metody rozpoznávání nacházejí uplatnění i v rámci jiných úloh UI – např. strojové vnímání.

Př. 1.1

scéna S



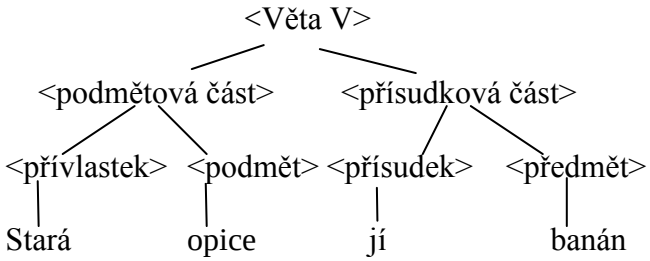
hierarchický popis



Př. 1.2

Věta: „Stará opice jí banán.“

Hierarchický popis:



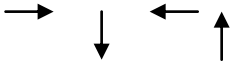
Celý obraz je složen z jednodušších částí a z ještě jednodušších podobně jako věta z jednotlivých částí a ty ze slov. Právě reprezentaci takové hierarchické strukturální informace a způsob jejího využití pro rozpoznávání zahrnuje strukturální přístup k rozpoznávání.

Nejjednodušší strukturální elementy obrazu jsou nazývány **primitiva**. (v př. 1.1 „stěna e“,...)

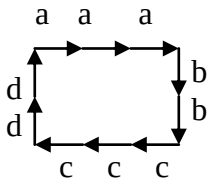
Strukturální informaci doplňují informace o **relacích** (vztazích) mezi primitivy. Relace mohou být reprezentovány buď pomocí logických nebo matematických operací, nebo relačním grafem.

Př. 1.3

primitiva: a b c d



obdélník



jediná relace – zřetězení

označíme +

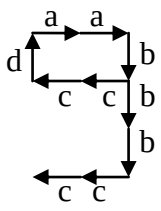
x + y ... x je zřetězeno s y

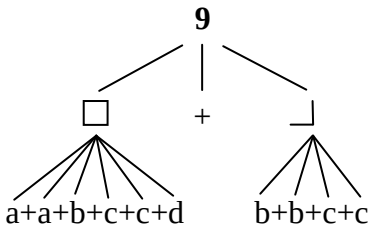
popis obdélníka:

a + a + a + b + b + c + c + c + d + d

jediná použitá relace - zřetězení ⇒ možno vynechat její symbol ⇒ aaabbcccdd

Př. 1.4

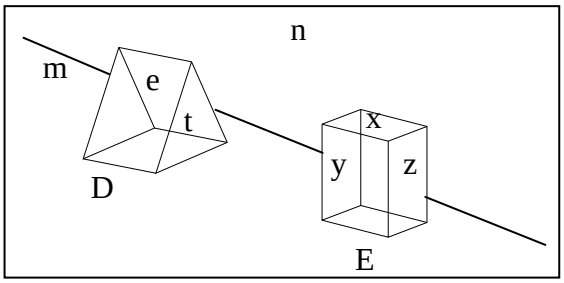




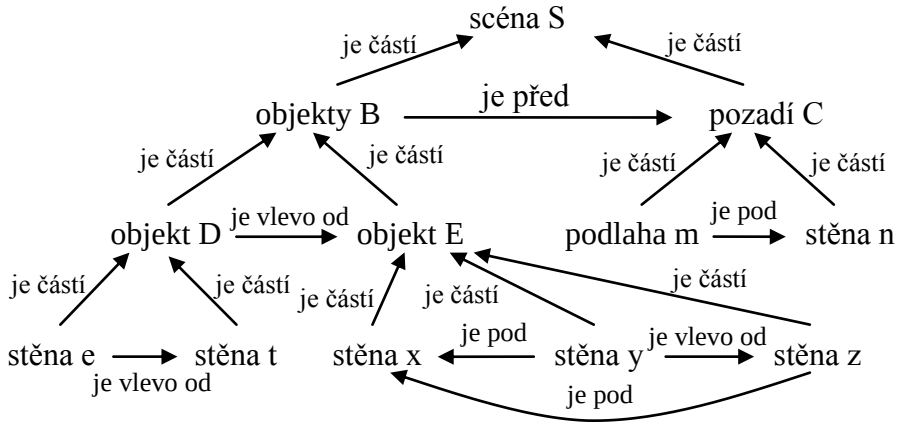
ccdaabbbcc .. pro vhodně zvolený počáteční bod

Jiná možnost – použít relační graf, kde každá hrana mezi 2 uzly představuje relaci mezi 2 částmi obrazu.

Př. 1.5



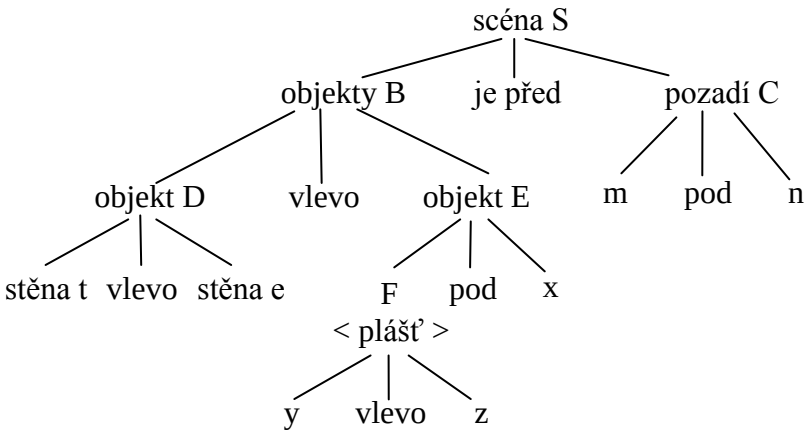
Popis pomocí relačního grafu může vypadat např. takto:



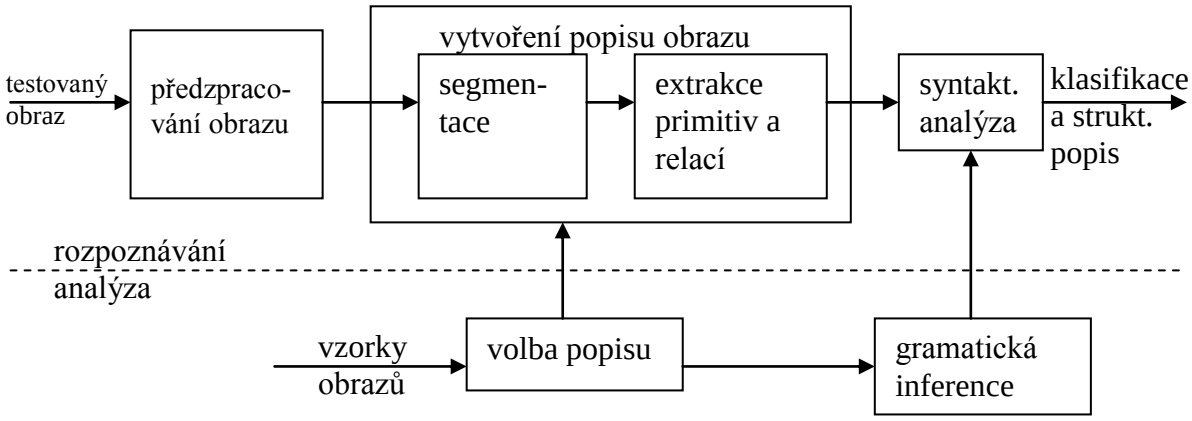
Popis grafem je bohatší, ale popis stromovou strukturou umožňuje použití výhodných postupů, které se opírají o výsledky tzv. **teorie formálních jazyků**. Tyto popisy umožňují reprezentaci často obrovského množství různých struktur obrazů v jedné třídě pouze pomocí konečné množiny vhodně zvolených primitiv a jistých syntaktických pravidel. Proto se obvykle snažíme pomocí vhodných úprav převést relační graf na stromovou strukturu.

Př. 1.6 Scéna S z př. 1.5 – Převod na stromovou strukturu

... Převod na stromovou strukturu.



Systém strukturálního rozpoznávání – blokové schéma



- ❖ předzpracování
  např. vzorkování, kódování, aproximace, filtrace, vyhlazování dat ...
- ❖ vytváření popisu
  - segmentování obrazu, vyhledávání primitiv a relací mezi nimi
  - výsledkem je popis obrazu – řetězec, strom, graf
- ❖ syntaktická analýza
  - generuje rozhodnutí, zda analyzovaný obraz patří do třídy charakterizované danou gramatikou
  - pokud ano – je získán úplný strukturální popis analyzovaného obrazu
- ❖ volba popisu

primitiva a relace musí
- být mnohem snadněji rozpoznatelné než strukturálně složité výchozí obrazy
- vystihovat přirozené a výrazné strukturální elementy rozpoznávaných obrazů

2 TEORIE FORMÁLNÍCH JAZYKŮ

- ♦ prvotně snaha o pochopení přirozených jazyků
- ♦ zkoumání programovacích jazyků z teoretického hlediska
- ♦ hlavní teoretický základ pro strukturální metody

2.1 Jazyky a gramatiky

2.1.1 Gramatiky

V .. konečná množina (abeceda) symbolů

V^+ .. množina všech konečných neprázdných posloupností (řetězců) utvořených z prvků množiny (symbolů abecedy) V

λ .. prázdný řetězec

$V^* = V^+ \cup \{\lambda\}$

Zřetězení řetězců (slov) φ a θ $\quad \varphi,\theta \in V^*$

$\varphi = a_1\, a_2 \,..\, a_n$, $\quad \theta = b_1\, b_2 \,..\, b_m$, pak $\quad \varphi\theta = a_1\, a_2 \,..\, a_n\, b_1\, b_2 \,..\, b_m$

n-násobné zřetězení řetězce φ

$\varphi^n \quad \varphi^1 = \varphi \quad \varphi^2 = \varphi\varphi \quad \varphi^{i+1} = \varphi^i\, \varphi \quad \varphi^0 = \lambda$

Délka řetězce θ

$|\theta| \quad |\, b_1\, b_2 \,..\, b_m| = m \quad |\lambda| = 0$

- Definice: ***Gramatika G*** *je čtveřice* $\boldsymbol{G = (V_N, V_T, P, S)}$*, kde*

  $\boldsymbol{V_N}$ *.. množina neterminálních symbolů*
  $\boldsymbol{V_T}$ *.. množina terminálních symbolů (primitiva)*
  P *.. množina přepisovacích pravidel*
  S *.. počáteční symbol gramatiky* $S \in V_N$

$V = V_N \cup V_T$

neterminální symboly	–	velká písmena	S, A, X, ..
terminální symboly	–	malá písmena	a, b, y, ..
celé řetězce	–	řecká písmena	ω, α, ..

Př. 2.1. Gramatika z příkladu 1.2.

S = < Věta V >

V_N = { <věta V>,<podm.část>,<přís.část>,<podmět>,<přísudek>,<přívlastek>,<předmět>}

V_T = {opice, žirafa, jí, stará, mladá, opice, ...}

P: <věta V> → <podm.č.> <přís.č.>
<podm.č.> → <podmět>
<podm.č.> → <přívlastek> <podmět>
<přís.č.> → <přísudek>
<přís.č.> → <přísudek> <předmět>
<podmět> → opice
<podmět> → žirafa
.....

Př. 2.2. Gramatika z příkladu 1.1.

V_N = {S, B, C, D, E}

V_T = {m, n, e, t, x, y, z}

P: S → B C
B → D E
C → m n
D → e t
E → x y z

S ... reprezentuje celý obraz

V_N ... její prvky reprezentují strukturálně jednodušší části

V_T ... její prvky reprezentují jednotlivá primitiva

Terminologie

- řetězec δ *přímo* generuje ω, nebo řetězec ω lze *přímo* odvodit z δ pokud:

$$\delta = \gamma_1\ \alpha\ \gamma_2 \qquad \omega = \gamma_1\ \beta\ \gamma_2 \qquad \alpha \to \beta \in P \qquad \delta, \gamma_1, \gamma_2, \alpha, \beta \in V^* \quad \alpha \notin V_T{}^*$$

značíme: $\boxed{\delta \Rightarrow \omega}$

- δ generuje ω, nebo ω lze odvodit z δ, pokud existuje posloupnost řetězců $\alpha_1,..., \alpha_n$ taková $\delta = \alpha_1$; $\omega = \alpha_n$

$$\alpha_i \Rightarrow \alpha_{i+1} \qquad i = 1,...,n-1$$

- posloupnost řetězců nazýváme **derivace** neboli odvození řetězce ω z δ, značíme:

$$\delta \overset{*}{\Rightarrow} \omega$$

Popis způsobu derivace řetězce:

- ❖ Zápis posloupnosti čísel pravidel postupně aplikovaných v derivaci
- ❖ Derivační strom

Př. 2.3. G = (V_N, V_T, P, S)

V_N = {S, A, B}
V_T = {a, b}

P: (1) S → A B (3) A → a (5) B → b
(2) A → a A (4) B → b B

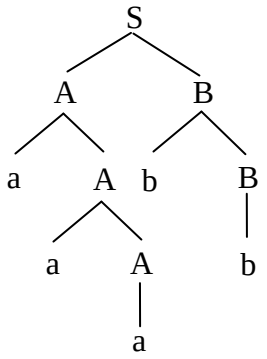
ω = a a a b b

derivace: $S \overset{*}{\Rightarrow} \omega$

$$S \overset{(1)}{\Rightarrow} A\ B \overset{(4)}{\Rightarrow} A\ b\ B \overset{(2)}{\Rightarrow} a\ A\ b\ B \overset{(5)}{\Rightarrow} a\ A\ b\ b \overset{(2)}{\Rightarrow} a\ a\ A\ b\ b \overset{(3)}{\Rightarrow} a\ a\ a\ b\ b$$

(1, 4, 2, 5, 2, 3) ... čísla pravidel

Derivační strom



2.1.2 Rozdělení gramatik podle Chomského hierarchie

Rozdělení řetězcových gramatik podle tvaru jejich pravidel:

- **Typ 0 – Gramatiky bez omezení**
- **Typ 1 – Kontextové gramatiky**

  pravidla omezena na tvar:

  $\gamma_1 A \gamma_2 \rightarrow \gamma_1 \beta \gamma_2$ $\quad A \in V_N$

  γ_1, γ_2 – kontext $\quad \beta, \gamma_1, \gamma_2 \in V^*$

- **Typ 2 – Bezkontextové gramatiky**

  pravidla omezená na tvar

  $A \rightarrow \beta$ $\quad A \in V_N$, $\beta \in V^*$

- **Typ 3 – Regulární gramatiky**

  pravidla omezena na tvar

  $A \rightarrow a\,B$ $\quad A, B \in V_N$

  $A \rightarrow b$ $\quad a, b \in V_T$

Regulární gramatiky jsou speciálním případem bezkontextových, bezkontextové speciálním případem kontextových a kontextové speciálním případem gramatik typu 0 ... **hierarchie.**

2.1.3 Levá derivace

Definice: Nechť G je **bezkontextová** gramatika. Řekneme, že řetězec α lze přepsat na β podle gramatiky G **levým přepsáním**, jestliže:

$\exists A \rightarrow \gamma \in P$ takové, že $\alpha = \delta\, A\, \varphi \quad \beta = \delta\, \gamma\, \varphi$

$A \in V_N$ $\qquad G = (V_N, V_T, P, S)$

$\gamma \in V^+$

$\delta \in V_T^*$

$\varphi \in V^*$

Odvození $\alpha_0 \Rightarrow \alpha_1 \Rightarrow ... \Rightarrow \alpha_n$ se nazývá **levou derivací,** právě když každé přepsání (odvození) $\alpha_i \Rightarrow \alpha_{i+1}$ i = 0,1,..., n-1, vzniká levým přepsáním.

Věta: Nechť $G = (V_N, V_T, P, S)$ je bezkontextová gramatika $A \overset{*}{\Rightarrow} \omega; A \in V_N, \omega \in V_T^*$ potom ω lze z A odvodit levou derivací.

Důkaz: Při přepisování řetězce u bezkontextové gramatiky nezávisí přepsání neterminálního symbolu na sousedních symbolech (řetězcích). Jestliže např. α_1 Y $\alpha_2 \overset{*}{\Rightarrow} \omega$, pak ω lze psát ve tvaru:

$\omega = \beta_1 \gamma \beta_2 \qquad \alpha_1 \overset{*}{\Rightarrow} \beta_1 \qquad \alpha_2 \overset{*}{\Rightarrow} \beta_2 \qquad Y \overset{*}{\Rightarrow} \gamma$

protože se části α_1, α_2, Y přepisují nezávisle na sobě, je možno přeskupit aplikaci pravidel tak, že se nejdříve přepisuje první netermální symbol zleva. Podobně lze zavést i **pravou derivaci**.

Př. 2.4. Aplikaci pravidel v př. 2.3. lze přeskupit takto:

$$S \overset{(1)}{\Rightarrow} A\ B \overset{(2)}{\Rightarrow} a\ A\ B \overset{(2)}{\Rightarrow} a\ a\ A\ B \overset{(3)}{\Rightarrow} a\ a\ a\ B \overset{(4)}{\Rightarrow} a\ a\ a\ b\ B \overset{(5)}{\Rightarrow} a\ a\ a\ b\ b$$

tedy (1, 2, 2, 3, 4, 5) – levá derivace

2.1.4 Víceznačnost gramatik

Definice: Mějme $G = (V_N, V_T, P, S)$; existuje-li řetězec ω takový, že existuje více než jedna levá derivace $S \overset{*}{\Rightarrow} \omega$, pak gramatika G je **víceznačná**.

Pozn.: Někdy je možné víceznačnou gramatiku transformovat na jinou, která generuje stejná slova (jazyk), ale není víceznačná.

Př. 2.5. $G = (V_N, V_T, P, S) \qquad V_N = \{S, X, Y\} \qquad V_T = \{x, y, z, +\}$

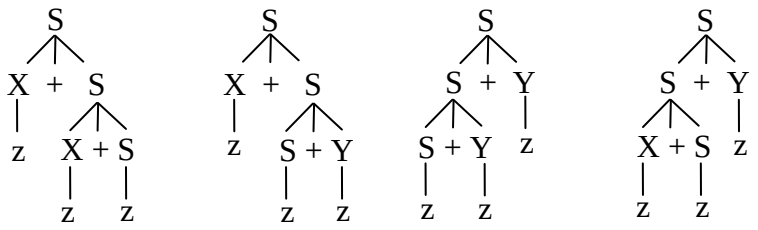
P: (1) S → X + S (3) S → z (5) Y → y (7) Y → z
(2) S → S + Y (4) X → x (6) X → z

ω = z + z + z

$$S \overset{(1)}{\Rightarrow} X + S \overset{(6)}{\Rightarrow} z + S \overset{(1)}{\Rightarrow} z + X + S \overset{(6)}{\Rightarrow} z + z + S \overset{(3)}{\Rightarrow} z + z + z \quad (1, 6, 1, 6, 3)$$

$$S \overset{(1)}{\Rightarrow} X + S \overset{(6)}{\Rightarrow} z + S \overset{(2)}{\Rightarrow} z + S + Y \overset{(3)}{\Rightarrow} z + z + Y \overset{(7)}{\Rightarrow} z + z + z \quad (1, 6, 2, 3, 7)$$

$$S \overset{(2)}{\Rightarrow} S + Y \overset{(2)}{\Rightarrow} S + Y + Y \overset{(3)}{\Rightarrow} z + Y + Y \overset{(7)}{\Rightarrow} z + z + Y \overset{(7)}{\Rightarrow} z + z + z \quad (2, 2, 3, 7, 7); (2, 1, 6, 3, 7)$$



⇒ Gramatika G je víceznačná.

Existuje ekvivalentní jednoznačná gramatika, např.:

S → x + S S → z S → z + S + z
S → S + y S → z + z

2.1.5 Jazyky

Definice: Mějme gramatiku $G = (V_N, V_T, P, S)$. **Jazyk** generovaný touto gramatikou definujeme: $L(G) = \left\{ \omega \,\middle|\, \omega \in V_T^*, S \overset{*}{\Rightarrow} \omega \right\}$

Jazyk generovaný regulární gramatikou nazýváme **Regulární jazyk (L_3)**
Jazyk generovaný bezkontextovou gramatikou nazýváme **Bezkontextový jazyk (L_2)**
Jazyk generovaný kontextovou gramatikou nazýváme **Kontextový jazyk (L_1)**
Jazyk generovaný gramatikou typu 0 nazýváme **Jazyk typu 0 (L_0)**

Pro třídy jazyků L_0,..., L_3 platí: $L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$

Př. 2.6. $G = (V_N, V_T, P, S)$ $V_N = \{S, X, Y\}$ $V_T = \{c, d\}$

P: S → d X Y → d
S → c Y X → c S
Y → d S X → d X X
Y → c Y Y X → c

$L(G) = \left\{ \omega \,\middle|\, \omega \in V_T^*, S \overset{*}{\Rightarrow} \omega \right\}$ = { ω | $n_c = n_d$ } = { cd, dc, cdcd, ccdd, ddcc, ...}

Pozn.: Transformace gramatik – někdy lze gramatiku jedné třídy transformovat na gramatiku vyšší třídy.

2.2 Jazyky a automaty

Jednou z možností, jak určit, zda daný řetězec patří do určitého jazyka, je použití rozpoznávacího stroje (automatu), který bude rozpoznávat (přijímat) pouze slova tohoto jazyka. Pro každou třídu gramatik existuje typ automatu, který přijímá její slova.

2.2.1 Regulární jazyky a konečné automaty

Definice: **Deterministický konečný automat** je pětice $A = (W, Q, \delta, q_0, F)$, kde:

W ... konečná množina vstupních symbolů
Q ... konečná množina stavů
δ ... přechodová funkce (zobrazení Q x W → Q)
q_0 ... počáteční stav: $q_0 \in Q$
F ... množina koncových stavů $F \subseteq Q$

Př. 2.7. Uvažujme $A = (W, Q, \delta, q_0, F)$

W = { x, y} Q = { q_0, q_1, q_2, q_3} F = { q_0 }

δ: $\delta(q_0, x) = q_1$ $\delta(q_2, x) = q_3$
$\delta(q_0, y) = q_2$ $\delta(q_2, y) = q_0$
$\delta(q_1, x) = q_0$ $\delta(q_3, x) = q_2$
$\delta(q_1, y) = q_3$ $\delta(q_3, y) = q_1$

Reprezentace automatu:

- ***Tabulkou***

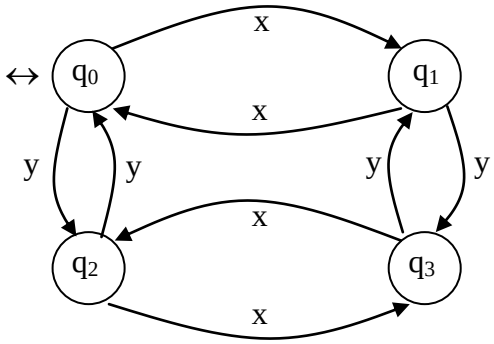
vst.sym. / stav	x	y
↔ q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1

→ počáteční stav

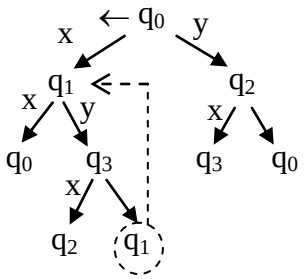
← koncový stav

↔ zároveň počáteční i koncový stav

- ***Stavovým diagramem***



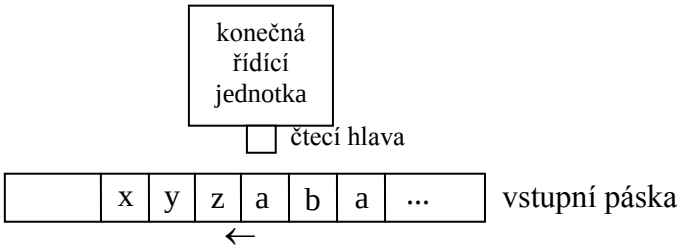
- ***Stavovým stromem***



... dostaneme-li se do listu (např. q_1), musíme vyhledat q_1 ve stromu a pokračujeme. Strom není totiž nekonečný.

... počáteční stav není třeba vyznačovat, protože je jím kořen stromu.

Znázornění funkce konečného automatu



Na počátku:

- automat ve stavu q_0
- snímá levý krajní symbol ze vstupní pásky

Při každém kroku:

- přečte symbol **a** ze vstupní pásky a v závislosti na tomto symbolu **a** a aktuálním stavu q přejde do nového stavu (definovaného δ(q, a))
- Přesune čtecí hlavu o 1 políčko doprava

Pokud je po posledním kroku (po přečtení posledního symbolu) automat v koncovém stavu ($q \in F$), řetězec, který byl přečten ze vstupní pásky, je přijat.

Definice: Zobecněná přechodová funkce

$A = (W, Q, \delta, q_0, F)$

$\delta^*: Q \times W^* \rightarrow Q$

platí:

1. $\delta^*(q, \lambda) = q \qquad \forall q \in Q$ $\qquad \forall q \in Q$
2. $\delta^*(q, \omega a) = \delta (\delta^*(q, \omega), a)$ $\qquad \omega \in W^*$, $a \in W$

nebo

$\delta^*(q, a\omega) = \delta^*(\delta(q, a), \omega)$

Jazyk rozpoznávaný (přijímaný) automatem

$$T(A) = \left\{\omega \middle| \ \delta^*(q_0, \omega) \in F, \omega \in W^*\right\}$$

Řetězec $\omega \in W^*$ je přijímán automatem A, právě když $\delta^*(q_0, \omega) \in F$, resp. $\omega \in T(A)$. Automat přijímá všechny řetězce, které ho převedou z počátečního stavu do jednoho z koncových stavů.

Ekvivalence automatů a gramatik

Dva automaty / gramatiky jsou ekvivalentní, pokud přijímají / generují stejný jazyk.

Definice: Nedeterministický konečný automat je pětice $A = (W, Q, \delta, q_0, F)$,

W, Q, q_0, F ... stejná jako u deterministického automatu.

δ ... přechodová funkce $Q \times W \rightarrow 2^Q$

V každém kroku automat sejme symbol **a** ze vstupní pásky, zvolí si jeden ze stavů z množiny stavů $\delta(q, a) = \{q_1, ..., q_e\}$, přejde do něj a posune čtecí hlavu o 1 políčko doprava.

Věta: Mějme **nedeterministický automat (konečný)** $A = (W, Q, \delta, q_0, F)$, který přijímá jazyk L. Pak **existuje deterministický konečný automat** $\widetilde{A} = \left(\widetilde{W}, \widetilde{Q}, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F}\right)$, který také přijímá jazyk L.

Platí: $\widetilde{W} = W$

$\widetilde{Q} = 2^Q$

$\widetilde{F} = \left\{\widetilde{q} \in \widetilde{Q} \middle| \ \exists q \in \widetilde{q} : q \in F\right\}$

stav $\widetilde{q}$ označíme $[q_1, q_2, ..., q_e] \in \widetilde{Q}; \quad q_1, q_2, ..., q_e \in Q$

počáteční stav $\widetilde{q}_0 = [q_0]$

$\widetilde{\delta}([q_1, q_2, ..., q_e], a) = [p_1, p_2, ..., p_j]$ právě tehdy když:

$$\delta(\{q_1, q_2, ..., q_e\}, a) = \bigcup_{k=1}^{e} \delta(q_k, a) = \{p_1, p_2, ..., p_j\}$$

Věta: Mějme G = (V_N, V_T, P, S) regulární gramatiku, která generuje jazyk L. Existuje nedeterministický konečný automat A = (W, Q, δ, q_0, F), který přijímá jazyk L:

$$T(A) = L(G)$$

Platí:

$W = V_T$ ---- koncový stav

$Q = V_N \cup \{M\}$

$q_0 = S$

$F = \{M\}$

$M \in \delta(X,a) \Leftarrow (X \to a,\ X \in V_N,\ a \in V_T) \in P$

$Y \in \delta(X,a) \Leftarrow (X \to aY;\ X,Y \in V_N,\ a \in V_T) \in P$

$\delta(M,a) = \emptyset \qquad \forall a \in V_T$

} definice δ

Věta: Nechť A = (W, Q, δ, q_0, F) je konečný automat, který přijímá jazyk L. Existuje regulární gramatika G = (V_N, V_T, P, S), která generuje jazyk L, tedy L(G) = T(A)

Platí:

$V_N = Q$

$V_T = W$

$S = q_0$

$X \to aY \in P \Leftarrow \delta(X,a) = Y \quad X,Y \in Q, a \in W$

$X \to a \quad \in P \Leftarrow \delta(X,a) = Y \wedge Y \in F; \quad X,Y \in Q, a \in W$

Př. 2.8.: G = (V_N, V_T, P, S)

V_N = {S, D} P: S → c D D → c D

V_T = {c, d} S → d S D → c

- Zkonstruujeme nedeterministický konečný automat:

  A = (W, Q, δ, q_0, F) tak, aby T(A) = L(G):

  W = V_T = {c, d}
  Q = $V_N \cup$ {M} = {S, D, M}
  q_0 = S
  F = {M}
  δ: δ(S, c) = {D}
  δ(S, d) = {S}
  δ(D, c) = {D, M}
  δ(D, d) = Ø
  δ(M, c) = δ(M, d) = Ø

- Zkonstruujeme deterministický konečný automat:

  $\widetilde{A} = (\widetilde{W}, \widetilde{Q}, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F})$ takový, že $T(\widetilde{A}) = T(A)$

  $\widetilde{A}$ je ekvivalentní s A. **pozn.:** [stav]

$\tilde{W} = W = \{c, d\}$

$\tilde{Q} = \{\emptyset, [S], [D], [M], [S,D], [S,M], [M,D], [S,D,M]\}$

$\tilde{q}_0 = [S]$

$\tilde{F} = \{[M], [S,M], [D,M], [S,D,M]\}$

$\tilde{\delta}: \tilde{\delta}([S], c) = [D] \qquad \tilde{\delta}([D, M], c) = [D,M]$

$\tilde{\delta}([S], d) = [S] \qquad \tilde{\delta}([D, M], d) = \emptyset$

$\tilde{\delta}([D], c) = [D, M]$

$\tilde{\delta}([D], d) = \emptyset$

Stavy [M], [S, D], [S, M], [S, D, M] nejsou využity (není do nich definován přechod) a mohou být z $\tilde{Q}$ i z $\tilde{F}$ vypuštěny.

- Sestrojíme regulární gramatiku
  $\tilde{G} = (\tilde{V}_N, \tilde{V}_T, \tilde{P}, \tilde{S})$ takovou, že $L(\tilde{G}) = T(\tilde{A})$

$\tilde{V}_N = \tilde{Q} = \{[S], [D], [D,M]\}$

$\tilde{V}_T = \tilde{W} = \{c, d\}$

$\tilde{S} = [S]$

$\tilde{P}$:

[S] → c [D]	[D] → c
[S] → d [S]	[D, M] → c [D, M]
[D] → c [D, M]	[D, M] → c

Protože ze symbolů [D] a [D, M] lze odvodit stejnou množinu řetězců: *c [D, M]* a *c*, lze tyto symboly ztotožnit. Pokud vynecháme závorky bude $\tilde{P}$ vypadat takto:

S → c D D → c D
S → d S D → c

- Výsledkem je tedy stejná gramatika, ze které jsme vyšli $\tilde{G} = G$.
- Příklad ukazuje, že při transformaci nedeterministického konečného automatu na deterministický nemusíme vždy získat minimální realizaci.

2.2.2 Bezkontextové jazyky a zásobníkové automaty

Věta: **Chomského normální forma**

Libovolný bezkontextový jazyk může být generován gramatikou $G = (V_N, V_T, P, S)$, ve které všechna pravidla jsou ve tvaru:

A → B C A, B C ∈ V_N

A → a a ∈ V_T

Př. 2.9.:

Převod na Chomského normální formu. Mějme bezkontextovou gramatiku $G = (V_N, V_T, P, S)$:

V_N = {S, A, B, C}

V_T = {a, b, c, d}

P: (1) S → A A B C (4) B → b B b
(2) A → a b a b (5) B → C
(3) C → c d (6) C → S

- Odstraníme pravidla typu $X \rightarrow Y$

  místo (5) $B \rightarrow C$: (7) $B \rightarrow c\ d$

  (8) $B \rightarrow S$

  (6) $C \rightarrow S$: (9) $C \rightarrow A\ A\ B\ C$

  (8) $B \rightarrow S$: (10) $B \rightarrow A\ A\ B\ C$

- Odstraníme pravidla typu $X \rightarrow a_1\ a_2\ ...\ a_n$, kde $\exists\ a_i,\ a_i \in V_T$, $n>1$

Pro každý symbol $x \in V_T$ (terminální symbol) zavedeme neterminální symbol Z_x

- ❖ Na pravé straně všech pravidel nahradíme x symbolem Z_x
- ❖ Přidáme pravidla:

$Z_X \rightarrow x$

(2) $\Rightarrow$ (2a) $A \rightarrow Z_a\ Z_b\ Z_a\ Z_b$

(3) $\Rightarrow$ (3a) $C \rightarrow Z_c\ Z_d$

(4) $\Rightarrow$ (4a) $B \rightarrow Z_b\ B\ Z_b$

(7) $\Rightarrow$ (7a) $B \rightarrow Z_c\ Z_d$

(10) $Z_a \rightarrow a$

(11) $Z_b \rightarrow b$

(12) $Z_c \rightarrow c$

(13) $Z_d \rightarrow d$

- Pravidla, která mají pravou stranu délky 2 (neterminální symboly), již vyhovují Chomského normální formě.
- Totéž platí pro pravidla s pravou stranou délky 1 (terminální symboly)
- Ostatní pravidla nahradíme soustavou pravidel:

$S \rightarrow A\ Y_1$; $Y_1 \rightarrow A\ Y_2$; $Y_2 \rightarrow B\ C$

$C \rightarrow A\ Y_1$

$B \rightarrow A\ Y_1$

$A \rightarrow Z_a\ Y_3$; $Y_3 \rightarrow Z_b\ Y_4$; $Y_4 \rightarrow Z_a\ Z_b$

$B \rightarrow Z_b\ Y_5$; $Y_5 \rightarrow B\ Z_b$

Přepíšeme:

$Z_a \rightarrow a$	$C \rightarrow Z_c\ Z_d$	$C \rightarrow A\ Y_1$
$Z_b \rightarrow b$	$B \rightarrow Z_c\ Z_d$	$B \rightarrow A\ Y_1$
$Z_c \rightarrow c$	$S \rightarrow A\ Y_1$	$A \rightarrow Z_c\ Y_3$
$Z_d \rightarrow d$	$Y_1 \rightarrow A\ Y_2$	$Y_3 \rightarrow Z_b\ Y_4$
	$Y_2 \rightarrow B\ C$	$Y_4 \rightarrow Z_a\ Z_b$
		$B \rightarrow Z_b\ Y_5$
		$Y_5 \rightarrow B\ Z_b$

Výsledná gramatika je v Chomského normální formě a je ekvivalentní s původní gramatikou.

Definice: **Nedeterministický zásobníkový automat M:**

$M = (W, Q, \Gamma, \delta, q_0, Z_0, F)$

W ... konečná množina vstupních symbolů

Q ... konečná množina stavů

Γ ... konečná množina symbolů zásobníku

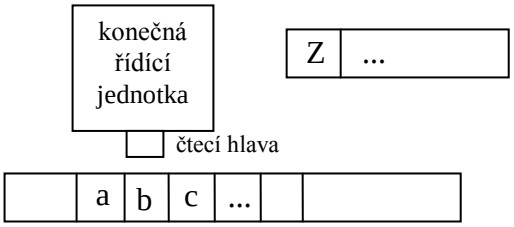
q_0 ... počáteční stav

Z_0 ... počáteční symbol zásobníku $Z_0 \in \Gamma$

F ... konečná množina koncových stavů $F \subseteq Q$

δ ... zobrazení $Q \times (W \cup \{\lambda\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$

Znázornění funkce:



$\delta (q, a, Z) = \{ (q_1, \gamma_1), (q_2, \gamma_2), ..., (q_m, \gamma_m) \}$

$q, q_1, ..., q_m \in Q$ $\quad Z \in \Gamma$

$a \in W$ $\quad \gamma_1, \gamma_2, ..., \gamma_m \in \Gamma^*$

Automat ve stavu *q* se symbolem *a* na vstupní pásce a symbolem *Z* na vrcholu zásobníku přejde do jednoho ze stavů q_i; i = 1,..., n a symbol *Z* nahradí řetězcem γ_i tak, že jeho levý krajní symbol bude umístěn na vrcholu zásobníku.

$\delta(q, \lambda, Z) = \{ (q_1, \gamma_1), (q_2, \gamma_2), ..., (q_m, \gamma_m) \}$

$q, q_1, q_2, ..., q_m \in Q$ $\quad Z \in \Gamma$ $\quad \gamma_1, \gamma_2, ..., \gamma_n \in \Gamma^*$

Automat ve stavu *q* se symbolem *Z* na vrcholu zásobníku přejde **nezávisle** na symbolu na vstupní pásce (aniž by ho přečetl) do jednoho ze stavů q_i, i = 1,..., n a symbol *Z* nahradí řetězcem γ_i tak, že jeho levý krajní symbol bude na vrcholu zásobníku.

Celkový stav automatu (q, γ)

q ... stav automatu, $q \in Q$

γ ... řetězec uložený v zásobníku, $\gamma \in \Gamma^*$

Zápis:

$$a: \quad (q, Z\gamma) \rightarrow (\tilde{q}, \beta\gamma)$$

$a \in (W \cup \{\lambda\})$ $\quad (\tilde{q}, \beta) \in \delta(q, a, Z)$

$\gamma, \beta \in \Gamma^*$ $\quad q, \tilde{q} \in Q$

$Z \in \Gamma$

Znamená, že podle pravidel δ může vstupní symbol *a* způsobit, že automat přejde z celkového stavu (q, Zγ) do celkového stavu $(\tilde{q}, \beta\gamma)$.

Jestliže platí:

$$a_i : (q_i, \gamma_i) \to (q_{i+1}, \gamma_{i+1}), \quad i = 1,...,n$$

$$\left.\begin{array}{l} a_i \in (W \cup \{\lambda\}) \\ q_i \in Q \\ \gamma_i \in \Gamma * \end{array}\right\} \forall \text{i, i = 1,..., n+1}$$

Pak píšeme $a_1a_{2...}a_n$: $(q_1, \gamma_1) \overset{*}{\to} (q_{n+1}, \gamma_{n+1})$ a říkáme, že řetězec $a_1a_2...a_n$ převádí automat z celkového stavu (q_1, γ_1) do celkového stavu (q_{n+1}, γ_{n+1}).

2 způsoby přijímání jazyka zásobníkovým automatem:

- **Jazyk přijímán koncovým stavem**

  $$T(M) = \left\{ \omega \mid \omega : (q_0, Z_0) \overset{*}{\to} (q, \gamma), \gamma \in \Gamma^*, q \in F \right\}$$

  Koncovým stavem jsou přijímány takové řetězce, které převedou automat z počátečního stavu do jednoho z koncových stavů.

- **Jazyk přijímán prázdným zásobníkem**

  $$N(M) = \left\{ \omega \mid \omega : (q_0, Z_0) \overset{*}{\to} (q, \lambda), q \in Q \right\}$$

  Prázdným zásobníkem jsou přijímány takové řetězce, které způsobí vyprázdnění zásobníku. V tomto případě je množina koncových stavů nepodstatná a obvykle volíme F = Ø.

Věta: Nechť L je libovolný bezkontextový jazyk, platí:
L = T(M_1) právě tehdy, když L = N(M_2), kde M_1 a M_2 jsou zásobníkové automaty. Oba způsoby přijímání jazyka jsou ekvivalentní.

Př. 2.10. Příklad automatu, který přijímá jazyk prázdným zásobníkem.

$$N(M) = \left\{ \omega\, x\, \omega^T \mid \omega \in \{a,b\}^* \right\}$$ (abcd)^T = dcba

$\omega = x_1\, x_2 \ldots x_m \Rightarrow \omega^T = x_m, x_{m-1}, \ldots, x_1$

M = (W, Q, Γ, δ, q_0, Z_0, F)

W = {a, b, x} $q_0 = q_1$ F = Ø
Q = {q_1, q_2} Z_0 = J
Γ = {J, K, L}

δ: δ(q_1, b, J) = { (q_1, K J) }
δ(q_1, b, K) = { (q_1, K K) }
δ(q_1, b, L) = { (q_1, K L) }
δ(q_1, a, J) = { (q_1, L J) }
δ(q_1, a, K) = { (q_1, L K) }
δ(q_1, a, L) = { (q_1, L L) }
δ(q_1, x, J) = { (q_2, J) }
δ(q_1, x, K) = { (q_2, K) }
δ(q_1, x, L) = { (q_2, L) }
δ(q_2, b, K) = { (q_2, λ) }
δ(q_2, a, L) = { (q_2, λ) }
δ(q_2, λ, J) = { (q_2, λ) }

Vstupní řetězec (ω = bba): bbaxabb

$$(q_1 J) \overset{b}{\rightarrow} (q_1, K\ J) \overset{b}{\rightarrow} (q_1, K\ K\ J) \overset{a}{\rightarrow} (q_1, L\ K\ K\ J) \overset{x}{\rightarrow} (q_2, L\ K\ K\ J) \overset{a}{\rightarrow} (q_2, K\ K\ J) \overset{b}{\rightarrow}$$

$$\overset{b}{\rightarrow} (q_2, K\ J) \overset{b}{\rightarrow} (q_2, J) \overset{\lambda}{\rightarrow} \underline{\underline{(q_2, \lambda)}} \text{ ...přijato.}$$

Věta: Ke každému bezkontextovému jazyku L existuje nedeterministický zásobníkový automat M (s jediným stavem) takový, že přijímá tento jazyk prázdným zásobníkem, tj. L = N(M).

Věta: Ke každému zásobníkovému automatu M lze sestrojit bezkontextovou gramatiku tak, že: L(G) = N(M).

Př. 2.11. G = (V_N, V_T, P, S)

V_N = {S, A, B, C}

V_T = {a, b, c, d}

P: (1) S → A A B C (4) B → b B b

(2) A → a b a b (5) B → c

(3) C → c d (6) C → S

Odpovídající zásobníkový automat lze sestavit na základě následujících úvah:

pravidlo X → ω $\hat{=}$ δ(q, λ, X) ∋ (q, ω)

a pro x ∈ V_T $\hat{=}$ δ(q, x, x) ∋ (q, λ)

↑ ↑

do těchto množin patří tento prvek

V zásobníku v podstatě provádím aplikaci pravidel, dokud je na vrcholu neterminál, a když se tam objeví terminál, porovnáme ho se symbolem na vstupním pásce a v případě rovnosti ho přečteme ze vstupní pásky a odstraníme z vrcholu zásobníku.

V našem případě δ

Pro pravidla:

S → A A B C δ(q, λ, S) = { (q, A A B C) }

A → a b a b δ(q, λ, A) = { (q, a b a b) }

B → b B b, B → c } δ(q, λ, B) = { (q, b B b), (q, c) }

C → c d, C → S } δ(q, λ, C) = { (q, c d), (q, S) }

$$\delta(q,a,a) = \{(q,\lambda)\};\quad \delta(q,b,b) = \{(q,\lambda)\};\quad \delta(q,c,c) = \{(q,\lambda)\};\quad \delta(q,d,d) = \{(q,\lambda)\}$$

M = (W, Q, Γ, δ, q_0, Z_0, F)

W = {a, b, c, d}

Q = {q}

Γ = {S, A, B, C, a, b, c, d}

q_0 = q

Z_0 = S

F = Ø

Všimněte si, že automat vystačí s 1 stavem.

Věta: Ke každému zásobníkovému automatu lze sestrojit ekvivalentní zásobníkový automat **s 1 stavem.**

Odpovídající bezkontextovou gramatiku k zásobníkovému automatu lze nalézt tak, že nejdříve sestrojíme ekvivalentní zásobníkový automat s jedním stavem a pak využijeme uvedeného vztahu mezi δ zásobníkového automatu a pravidly gramatiky.

Pozn.: instrukci $\delta(q, \lambda, A) = \{(q, abab)\}$ můžeme díky instrukci $\delta(q, a, a) = \{(q, \lambda)\}$ nahradit takto: $\delta(q, a, A) = \{(q, bab)\}$

Instrukce typu $\delta(q, x, x) = \{(q, \lambda)\}$ lze z δ vynechat jen v případě, že se symbol *x* nevyskytuje nikde na pravé straně (po převodu).

Příklad provedení této úpravy u automatu z předchozího příkladu

δ: $\delta(q, \lambda, S) = \{(q, A\ A\ B\ C)\}$
$\delta(q, a, A) = \{(q, bab)\}$
$\delta(q, c, B) = \{(q, \lambda)\}$
$\delta(q, b, B) = \{(q, B\ b)\}$
$\delta(q, \lambda, C) = \{(q, S)\}$
$\delta(q, c, C) = \{(q, d)\}$
$\delta(q, a, a) = \{(q, \lambda)\}$
$\delta(q, b, b) = \{(q, \lambda)\}$
$\delta(q, d, d) = \{(q, \lambda)\}$

$\Gamma = \{S, A, B, C, a, b, d\}$

Na rozdíl od konečných automatů nelze obecně ke každému nedeterministickému zásobníkovému automatu sestrojit odpovídající deterministický zásobníkový automat.

2.2.3 Jazyky typu 0 a Turingovy stroje

Na rozdíl od konečného automatu, má Turingův stroj čtecí a záznamovou hlavu, takže může symboly na pásce číst i přepisovat. Dalším rozdílem je možnost pohybovat čtecí a záznamovou hlavou na obě strany.

Definice: Turingův stroj je šestice:

$T = (W, Q, I, \delta, q_0, F)$

Q ... konečná množina stavů

I ... konečná množina symbolů pásky (jedním z nich je prázdný symbol B)

W ... konečná množina vstupních symbolů, $W \subseteq I$ (neobsahuje B)

F ... konečná množina koncových stavů $F \subseteq Q$

q_0 ... počáteční stav

δ ... zobrazení $Q \times I \rightarrow Q \times (I-\{B\}) \times \{-1, 1\}$

Zobrazení δ - přechodová funkce nemusí být definována pro určité argumenty

Konfigurace Turingova stroje

(q, α, i)

q ... stav $q \in Q$

α ... řetězec na neprázdné části pásky $\alpha \in (I - \{B\})^*$

i ... pozice hlavy

Elementární kroky Turingova stroje

- Jestliže $\delta(q, A_i) = (p, X, 1)$ $i \in \langle 1, n \rangle$
  $(q, A_1A_2...A_{i-1}\,A_i\,A_{i+1}...A_n, i) \rightarrow (p, A_1A_2...A_{i-1}\,X\,A_{i+1}...A_n, i+1)$
  pak Turingův stroj zapíše symbol X do i-tého políčka (přepíše A_i), přejde ze stavu *q* do stavu *p* a posune hlavu o 1 políčko doprava.
- Jestliže $\delta(q, A_i) = (p, X, -1)$ $i \in \langle 2, n \rangle$
  $(q, A_1A_2...A_{i-1}\,A_i\,A_{i+1}...A_n, i) \rightarrow (p, A_1A_2...A_{i-1}\,X\,A_{i+1}...A_n, i-1)$
  tj. posunu hlavu o 1 políčko doleva
- Jestliže $\delta(q, B) = (p, X, 1)$ $i = n+1$
  $(q, A_1A_2...A_n, n+1) \rightarrow (p, A_1A_2...A_n\,X, n+2)$
  přepíšeme prázdný symbol B na X. n:= n+1 (n++)
- Jestliže $\delta(q, B) = (p, X, -1)$ $i = n+1$
  $(q, A_1A_2...A_n, n+1) \rightarrow (p, A_1A_2...A_n\,X, n)$ n:= n+1 (n++)
- Jestliže jsou dvě konfigurace spojeny konečným počtem těchto elementárních kroků, používáme značení $\overset{*}{\rightarrow}$

$$(q, \alpha, i) \overset{*}{\rightarrow} (p, \beta, j)$$

Definice: Jazyk přijímaný Turingovým strojem:

$$L(T) = \left\{ \omega \mid \omega \in W^* \wedge (q_0, \omega, 1) \overset{*}{\rightarrow} (q, \alpha, i), q \in F, \alpha \in I^*, i \in \langle 1, n \rangle \right\}$$

Všechna slova (řetězce) patřící do jazyka přijímaného Turingovým strojem převádí Turingův stroj do koncového stavu, z něhož není další přechod definován – Turingův stroj se zastaví. Pro jiná slova je obecně možné, že se Turingův stroj vůbec nezastaví.

Př. 2.12. Jazyk $L = \{x^n y^n \mid n \geq 1\}$

$T = (W, Q, I, \delta, q_0, F)$

$W = \{x, y\}$
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$I = \{x, y, B, C, D\}$
$F = \{q_5\}$

δ:
(1) $\delta(q_0, x) = (q_1, D, 1)$
(2) $\delta(q_1, x) = (q_1, x, 1)$
(3) $\delta(q_2, C) = (q_2, C, -1)$
(4) $\delta(q_3, C) = (q_3, C, 1)$
(5) $\delta(q_4, x) = (q_4, x, -1)$
(6) $\delta(q_1, C) = (q_1, C, 1)$
(7) $\delta(q_2, D) = (q_3, D, 1)$
(8) $\delta(q_3, B) = (q_5, C, 1)$
(9) $\delta(q_4, D) = (q_0, D, 1)$
(10) $\delta(q_1, y) = (q_2, C, -1)$
(11) $\delta(q_2, x) = (q_4, x, -1)$

např. pro ω = x x y y je posloupnost operací:

$$(q_0, x\ x\ y\ y, 1) \overset{1}{\rightarrow} (q_1, D\ x\ y\ y, 2) \overset{2}{\rightarrow} (q_1, D\ x\ y\ y, 3) \overset{10}{\rightarrow} (q_2, D\ x\ C\ y, 2) \overset{11}{\rightarrow} (q_4, D\ x\ C\ y, 1) \overset{9}{\rightarrow}$$

$$\overset{9}{\rightarrow} (q_0, D\ x\ C\ y, 2) \overset{1}{\rightarrow} (q_1, D\ D\ C\ y, 3) \overset{6}{\rightarrow} (q_1, D\ D\ C\ y, 4) \overset{10}{\rightarrow} (q_2, D\ D\ C\ C, 3) \overset{3}{\rightarrow}$$

$$(q_2, D\ D\ C\ C, 2) \overset{7}{\rightarrow} (q_3, D\ D\ C\ C, 3) \overset{4}{\rightarrow} (q_3, D\ D\ C\ C, 4) \overset{4}{\rightarrow} (q_3, D\ D\ C\ C, 5) \overset{8}{\rightarrow}$$

$$\overset{8}{\rightarrow} (q_5, D\ D\ C\ C\ C, 6)$$

Věta: Jestliže je L generovaný gramatikou typu 0, potom L je také přijímaný nějakým Turingovým strojem.

Věta: Jestliže L je přijímaný Turingovým strojem, potom je také generovatelný nějakou gramatikou typu 0.

Definice: Turingův stroj **rozhoduje** jazyk L nad abecedou W, jestliže:

$L = L(T),\ L \subseteq W^*$

a pro každé $\omega \in W^*$ se T-stroj zastaví.

Existují jazyky rozpoznatelné T-strojem, které nejsou rozhodnutelné žádným T-strojem.

Definice: Nedeterministický Turingův stroj: $T = (W, Q, I, \delta, q_0, F)$

Q, W, I, F, q_0 ... stejné jako u definice T-stroje.

δ - zobrazení $Q \times I \rightarrow 2^{Q \times (I - \{B\}) \times \{-1,1\}}$

Věta: Je-li jazyk L přijímaný nějakým nedeterministickým Turingovým strojem, pak je přijímaný i nějakým deterministickým Turingovým strojem.

2.2.4 Kontextové jazyky a nedeterministické Turingovy stroje s lineárním prostorem

Definice: Nedeterministický T-stroj s lineárním prostorem:

$M = (W, Q, I, \delta, q_0, F)$

Q ... konečná množina stavů

I ... konečná množina symbolů pásky

W ... konečná množina vstupních symbolů $W \subseteq I$

q_0 ... počáteční stav $q_0 \in Q$

F ... množina koncových stavů $F \subseteq Q$

δ ... zobrazení $Q \times I \rightarrow 2^{Q \times I \times \{-1;\ 1\}}$

W obsahuje dva speciální symboly m_l, m_r – levý a pravý mezník řetězce na vstupní pásce – zabraňují přechodu hlavy mimo vstupní řetězec.

Definice: Jazyk přijímaný nedeterministickým T-strojem s lineárním prostorem:

$$L(M) = \left\{ \omega \mid \omega \in (W - \{m_l, m_r\})^*; (q_0, m_l\ \omega\ m_r, 1) \overset{*}{\rightarrow} (q, \alpha, i), q \in F, \alpha \in I^*, i \in <1, n> \right\}$$

Věta: Jestliže je L kontextový jazyk, pak je přijímán nějakým nedeterministickým T-strojem s lineárním prostorem.

Věta: Jestliže je L přijímán nějakým nedeterministickým T-strojem s lineárním prostorem, pak je generovatelný nějakou kontextovou gramatikou.

3 GRAMATIKY PRO POPIS OBRAZŮ

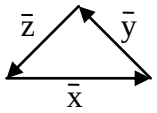
3.1 Kritéria výběru primitiv

***Primitiva*:**

- ❖ Žádaný popis obrazů.
- ❖ Snadno detekovatelná a rozpoznatelná.

Př. 3.1. *Odlišení rovnostranných trojúhelníků od ostatních obrazů.*

množina primitiv



x̄ ... horizontální segment
ȳ ... segment se sklonem 120°
z̄ ... segment se sklonem -120°

segmenty x̄, ȳ, z̄ mají stejnou délku

množina všech rovnostranných trojúhelníků je reprezentována řetězcem x̄ ȳ z̄.

Odlišení rovnostranných trojúhelníků různých velikostí → přiřadíme segmentům x̃,ỹ,z̃ jednotku délky a množinu rovnostranných trojúhelníků popíšeme jazykem:

$L = \{\tilde{x}^n \tilde{y}^n \tilde{z}^n \mid n = 1, 2,...\}$

Př. 3.2. *matematické výrazy*

$G = (V_N, V_T, P, S)$

V_N = { <výraz>, <operátor>, <operand>, <X>, <Y>, <Z>, <plus>, <mínus>, <krát>, <děleno>, <mezera>}

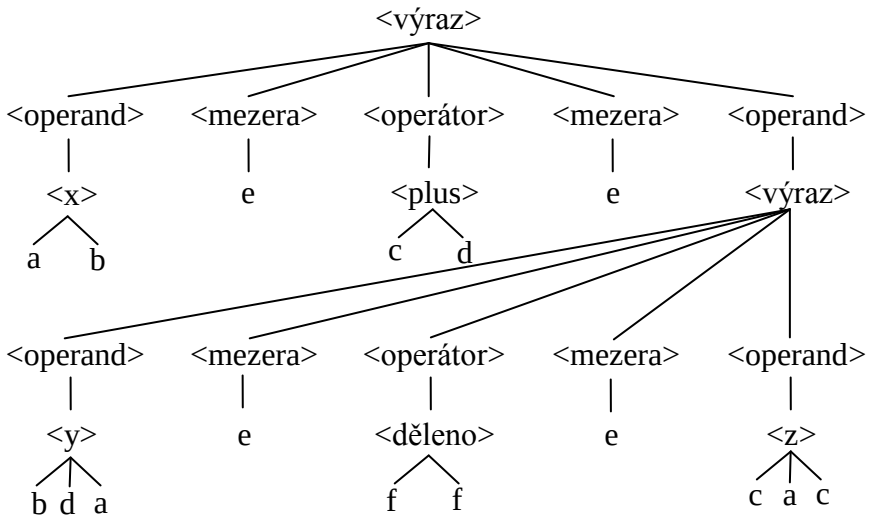
V_T = {a, b, c, d, e, f} a / b \ c - d | e ␣ f .

S = <výraz>

P: <výraz> → <operand> <operátor> <operand>
<operand> → <X>
<operand> → <Y>
<operand> → <Z>
<operand> → <výraz>
<operátor> → <plus>
<operátor> → <mínus>
<operátor> → <krát>
<operátor> → <děleno>
<X> → a b
<Y> → b d a
<Z> → c a c
<plus> → c d
<mínus> → c
<krát> → f
<děleno> → a
<děleno> → f f
<výraz> → <operand> <mezera> <operátor> <mezera> <operand>
<mezera> → e

Obraz: X + Y : Z je reprezentován řetězcem: ω = abecdebdaeffecac

Odpovídající derivační strom:



→ získaný strukturální popis je příliš složitý.

V_N = {V, D, R}

V_T = {x, y, z, +, -, ., :, /}

S = V

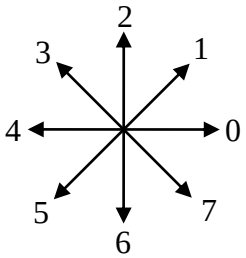
P: V → D R D R → +
D → V R → -
D → x R → .
D → y R → :
D → z R → /

ω = x + y : z

Př. 3.3. Popis čárových obrazců, rozhraní a obrysů ve snímcích – **Freemanův řetězcový kód** – každému segmentu je přiřazeno číslo v závislosti na jeho sklonu.

Výhody:

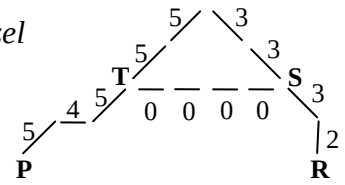
- Otáčení obrazu o násobek 45°
- Možnost měření délky křivky
- Určení průsečíků



viz ZDO

Obraz písmene A:

řetězec	*relativní směr*	*počáteční uzel*	*koncový uzel*
23	2	R	S
3355	4	S	T
0000	0	T	S
545	5	T	P



Př. 3.4. Popis křivek pomocí jejich tvarových vlastností.

- *Primitiva* – úseky, kde má křivka přibližně stejný tvar (část přímky, paraboly, kružnice,...)
- *Segmentační a aproximační algoritmy* – nejjednodušší, nejrychlejší, nejpoužívanější i po částech lineární aproximace.

Analýza EEG (záznamy aktivity lidského mozku) – Giese a kol.

Obraz: 4 křivky po 100 sekundách (4 kanály)
každý sekundový úsek → primitivum
17 příznaků
shluková analýza → 7 tříd primitiv
reprezentace obrazu: 4 řetězce po 100 primitivech.

3.2 Jednodimenzionální gramatiky

Po výběru primitiv – další problém: konstrukce gramatiky

Ideál: odvození gramatiky inferenčním strojem (konstrukce inferenčních strojů omezena na speciální případy)

Volba jazyka (složitější jazyk):

- Silnější prostředek k vyjadřování
- Vyšší nároky na složitost rozpoznávacího automatu
- Delší doba potřebná pro klasifikaci
- Narůstá algoritmická nerozhodnutelnost vlastností daného jazyka

Kompromis mezi vyjadřovací sílou jazyka a efektivností jeho analýzy je určován řešitelem v závislosti na řešené úloze.

Př. 3.6. Porovnání vyjadřovací síly u různých typů gramatik. Zkonstruujeme gramatiku pro regulární jazyk.

$$L = \left\{ x^n y^n z^n \mid n = 1, 2, 3 \right\}$$

- ***Regulární gramatika***

$G_1 = (V_N, V_T, P, S)$

$V_N = \{S, D_1, D_2, E_{10}, E_{20}, E_{21}, E_{31}, E_{32}, F_1, F_2, F_3\}$
$V_T = \{x, y, z\}$

P:	$S \to x\ D_1$	$E_{10} \to y\ F_1$	$F_1 \to z$		
	$S \to x\ E_{10}$	$E_{20} \to y\ E_{21}$	$F_2 \to z\ F_1$		
	$D_1 \to x\ D_2$	$E_{21} \to y\ F_2$	$F_3 \to z\ F_2$		
	$D_1 \to x\ E_{20}$	$E_{30} \to y\ E_{31}$		$\lvert V_N \rvert = 12$	$\lvert P \rvert = 14$
	$D_2 \to x\ E_{30}$	$E_{32} \to y\ F_3$	$E_{31} \to y\ E_{32}$		

- ***Bezkontextová gramatika***

$G_2 = (V_N, V_T, P, S)$

$V_N = \{S, D_1, D_2, E_1, E_2, E_3, F\}$
$V_T = \{x, y, z\}$

P:	$S \to x\ D_1\ F$	$D_1 \to x\ D_2\ F$	$E_2 \to y\ E_1$		
	$D_1 \to y$	$D_2 \to x\ E_3\ F$	$E_1 \to y$		
	$D_1 \to x\ E_2\ F$	$E_3 \to y\ E_2$	$F \to z$	$\lvert V_N \rvert = 7$	$\lvert P \rvert = 9$

Gramatiky s řízeným přepisováním

- Zvyšují generativní sílu určitého typu gramatiky.
- Např. pro generování kontextového jazyka můžeme použít i bezkontextovou gramatiku s řízeným přepisováním a pro rozpoznávání potom upravený zásobníkový automat s řízenou volbou instrukcí (namísto nedeterministického Turingova stroje s lineárním prostorem).

Př. 3.7. Jazyk $L = \{x^n y^n z^n \mid n = 1, 2,...\}$

- ***Bezkontextová programová gramatika***

$G = (V_N, V_T, P, S, J)$
$V_N = \{S, E, F\}$
$V_T = \{x, y, z\}$
$J = \{1, 2, 3, 4, 5\}$ ... množina čísel pravidel

P:	*č. pravidla*	*pravidlo*	*úspěch*	*neúspěch*	
	1	$S \rightarrow x\ E$	{2, 3}	Ø	Ø...konec.
	2	$E \rightarrow x\ E\ E$	{2, 3}	Ø	
	3	$E \rightarrow F$	{4}	{5}	
	4	$F \rightarrow y\ F$	{3}	Ø	
	5	$F \rightarrow z$	{5}	Ø	

Generování v programové gramatice probíhá takto: Je-li nějaké pravidlo úspěšně aplikováno, je možné v dalším kroku volit pouze pravidlo s číslem z množiny uvedené ve sloupci ***úspěch***, jinak je možné volit pouze pravidla z množiny ***neúspěch***.

Např. řetězec: x x x y y y z z z

$$S \overset{(1)\,u'}{\rightarrow} x\,E \overset{(2)\,u'}{\rightarrow} x\,x\,E\,E \overset{(2)\,u'}{\rightarrow} x\,x\,x\,E\,E\,E \overset{(3)\,u'}{\rightarrow} x\,x\,x\,F\,E\,E \overset{(4)\,u'}{\rightarrow} x\,x\,x\,y\,F\,E\,E \overset{(3)\,u'}{\rightarrow}$$

$$\overset{(3)\,u'}{\rightarrow} x\,x\,x\,y\,F\,F\,E \overset{(4)\,u'}{\rightarrow} x\,x\,x\,y\,y\,F\,F\,E \overset{(3)\,u'}{\rightarrow} x\,x\,x\,y\,y\,F\,F\,F \overset{(4)\,u'}{\rightarrow} x\,x\,x\,y\,y\,y\,F\,F\,F \overset{(3)\,N}{\rightarrow}$$

$$\overset{(5)\,u'}{\rightarrow} x\,x\,x\,y\,y\,y\,z\,F\,F \overset{(5)\,u'}{\rightarrow} x\,x\,x\,y\,y\,y\,z\,z\,F \overset{(5)\,u'}{\rightarrow} x\,x\,x\,y\,y\,y\,z\,z\,z$$

- ***Maticová gramatika***

$G = (V_N, V_T, \overline{P}, S)$
$V_N = \{S, D, E, F\}$ — matice pravidel
$V_T = \{x, y, z\}$

$\overline{P}$: (1) <$S \rightarrow D\ E\ F$>

(2) <$D \rightarrow x\ D$; $E \rightarrow y\ E$; $F \rightarrow z\ F$>

(3) <$D \rightarrow x$; $E \rightarrow y$; $F \rightarrow z$>

U maticových gramatik jsou zadávány posloupnosti (matice) pravidel, které se musí při odvozování použít současně.

Např. ω = x x x y y y z z z

$$S \overset{(1)}{\rightarrow} D\,E\,F \overset{(2)}{\rightarrow} x\,D\,y\,E\,z\,F \overset{(2)}{\rightarrow} x\,x\,D\,y\,y\,E\,z\,z\,F \overset{(3)}{\rightarrow} x\,x\,x\,y\,y\,y\,z\,z\,z$$

Př. 3.8. Rozpoznávání vrcholů v křivkách, založené na strukturálním přístupu – Horowitz.

Křivka reprezentována množinou bodů:

$\{ (x_i, y_i), i = 1, 2,..., n \}$ $\quad x_{i+1} > x_i \quad \forall i = 1, 2,..., n-1$

První diference d_i, i = 1, 2,..., n-1:

$d_i = (y_{i+1} - y_i) / (x_{i+1} - x_i)$

Ke každé dvojici bodů $[(x_i, y_i), (x_{i+1}, y_{i+1})]$ přiřadíme symbol ω takto:

$d_i > 0$ $\quad \omega_i = p$ (rostoucí úsek)
$d_i < 0$ $\quad \omega_i = n$ (klesající úsek)
$d_i = 0$ $\quad \omega_i = 0$ (konstantní úsek)

Nyní můžeme křivku reprezentovat řetězcem $\omega = \omega_1 \omega_2 \omega_3 ... \omega_{n-1}$

Kladný vrchol K

levá strana $\quad L = \{p\} \cup \{ p (\{p\} \cup \{0\})^* p \}$

pravá strana $\quad P = \{n\} \cup \{ n (\{n\} \cup \{0\})^* n \}$

vrcholek $\quad V = 0^*$ (libovolný počet *n* a nul)

$K = L \; V \; P$

Obdobně záporný vrchol Z

$Z = P \; V \; L$

Křivka: x vrcholů, x = 0, 1,...

Gramatika:

L: $\langle rost_1 \rangle \rightarrow p$
$\langle rost_1 \rangle \rightarrow \langle rost_1 \rangle \; p$
$\langle rost_1 \rangle \rightarrow \langle rost_2 \rangle \; p$
$\langle rost_2 \rangle \rightarrow \langle rost_1 \rangle \; 0$
$\langle rost_2 \rangle \rightarrow \langle rost_2 \rangle \; 0$

P: $\langle kles_1 \rangle \rightarrow n$
$\langle kles_1 \rangle \rightarrow \langle kles_1 \rangle \; n$
$\langle kles_1 \rangle \rightarrow \langle kles_2 \rangle \; n$
$\langle kles_2 \rangle \rightarrow \langle kles_1 \rangle \; 0$
$\langle kles_2 \rangle \rightarrow \langle kles_2 \rangle \; 0$

V: $\langle nula \rangle \rightarrow 0$
$\langle nula \rangle \rightarrow \langle nula \rangle \; 0$

K: $\langle vrchol^+ \rangle \rightarrow \langle rost_1 \rangle \; \langle kles_1 \rangle$
$\langle vrchol^+ \rangle \rightarrow \langle rost_1 \rangle \; \langle nula \rangle \; \langle kles_1 \rangle$

Z: $\langle vrchol^- \rangle \rightarrow \langle kles_1 \rangle \; \langle rost_1 \rangle$
$\langle vrchol^- \rangle \rightarrow \langle kles_1 \rangle \; \langle nula \rangle \; \langle rost_1 \rangle$

křivka: $\langle vrchol^- \rangle \rightarrow \langle vrchol^+ \rangle \; \langle rost_1 \rangle$
$\langle vrchol^- \rangle \rightarrow \langle vrchol^+ \rangle \; \langle nula \rangle \; \langle rost_1 \rangle$
$\langle vrchol^+ \rangle \rightarrow \langle vrchol^- \rangle \; \langle kles_1 \rangle$
$\langle vrchol^+ \rangle \rightarrow \langle vrchol^- \rangle \; \langle nula \rangle \; \langle kles_1 \rangle$

hlavní část křivky: $\langle průběh \rangle \rightarrow \langle rost_1 \rangle$
$\langle průběh \rangle \rightarrow \langle kles_1 \rangle$
$\langle průběh \rangle \rightarrow \langle vrchol^+ \rangle$
$\langle průběh \rangle \rightarrow \langle vrchol^- \rangle$

celá křivka: <w> → <nula>
<w> → <nula> <průběh>
<w> → <průběh> <nula>
<w> → <nula> <průběh> <nula>

3.3 Vícedimenzionální gramatiky

Nevýhoda jednodimenzionálních gramatik: používají pouze operace zřetězení.

- Získaný popis 2D a 3D objektů bývá těžkopádný
- Použití vícedimenzionální gramatiky

Definice: Pavučinová gramatika G

$G = (V_N, V_T, P, S)$

V_N ... Množina neterminálních symbolů

V_T ... Množina terminálních symbolů

S ... Množina počátečních pavučin

P ... Množina pavučinových přepisovacích pravidel ve tvaru: $\alpha \rightarrow \beta$, E

α, β ... Pavučiny

E ... Je množina jistých logických funkcí, určujících způsob vložení β místo α v dosud vytvořené pavučině.

Pavučinová gramatika generuje orientované grafy, jejichž uzly jsou terminální symboly.

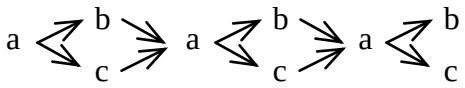
Př. 3.9. Pavučinová gramatika $G = (V_N, V_T, P, S)$

$V_N = \{A\}$; $V_T = \{a, b, c\}$; $S = \{A\}$

P: (1) $A \rightarrow a \begin{matrix} b \\ c \end{matrix}$ $E = \{ (p, a) \mid (p, A) \}$

(2) $A \rightarrow a \begin{matrix} b \\ c \end{matrix} A$ $E = \{ (p, a) \mid (p, A) \}$

Aplikace pravidel 2, 2, 1



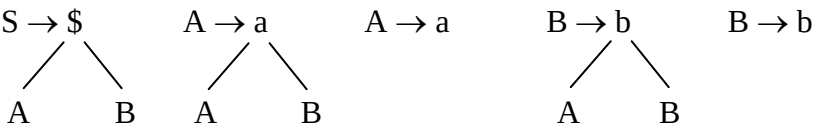
V případě, že V_T obsahuje jediný symbol → všechny uzly mají stejné označení, které tedy nemusíme zapisovat. Pavučinu tedy můžeme považovat za graf. Tento typ pavučinových gramatik je někdy nazýván **gramatikami grafů.**

Rozšířením jednodimenzionálního zřetězení na vícedimenzionální vznikají stromy. Může-li být struktura popsána stromem, může být její popis generován nějakou stromovou gramatikou a přijímán nějakým stromovým automatem.

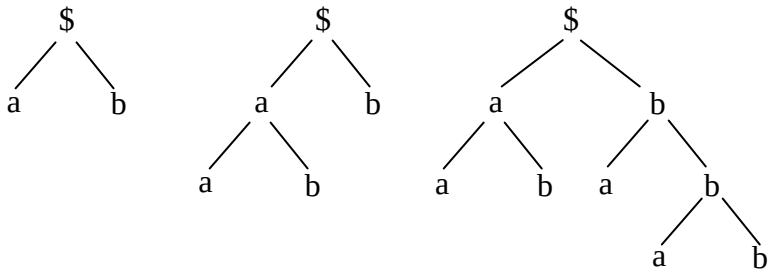
Př. 3.10. Stromová gramatika $G = (V_N, V_T, P, S)$

$V_N = \{S, A, B\}$ $V_T = \{a, b, \$\}$

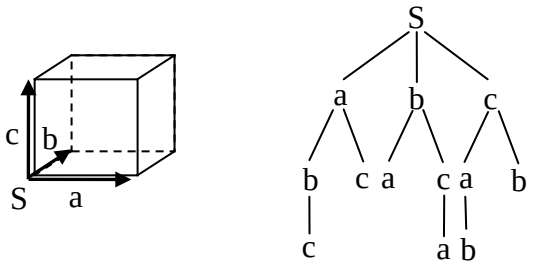
P: $S \rightarrow \$$ $A \rightarrow a$ $A \rightarrow a$ $B \rightarrow b$ $B \rightarrow b$



Příklady generovaných stromů:



Př. 3.11. Popis krychle stromem



4 SYNTAKTICKÁ ANALÝZA

4.1 Definice

Předpokládejme třídy obrazů ω_i, i = 1, 2,..., m, každá třída ω_i je charakterizována gramatikou G_i.

$L(G_l) \cap L(G_k) = \emptyset$ $\forall l$, k = 1,..., m $l \neq k$

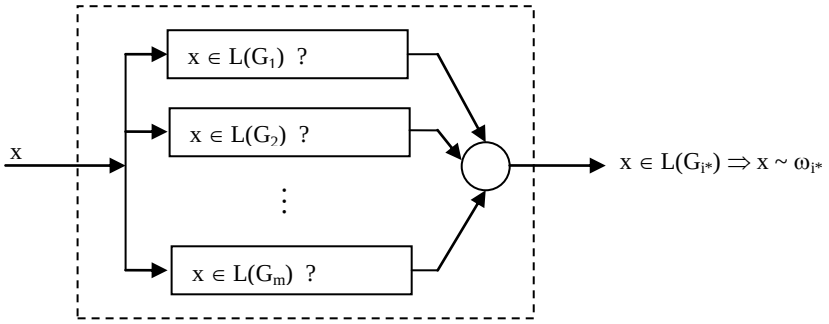
Úloha klasifikace:

$\exists$ i*: $x \in L(G_{i*})$ $i* \in \langle 1, m \rangle$?

Syntaktická analýza

Proces, který rozhoduje o náležitosti řetězce do jazyka generovaného gramatikou.

Blokový diagram syntaktického klasifikátoru:



V případě, že existuje G_{i*} taková, že $x \in L(G_{i*})$, klasifikujeme x do třídy ω_{i*}.

❖ ***Regulární gramatika***

Stačí zkonstruovat odpovídající deterministický konečný automat, který má obvykle snadnou softwarovou i hardwarovou realizaci.

❖ ***Bezkontextová gramatika***

Obecně stačí zkonstruovat nedeterministický zásobníkový automat.

❖ ***Kontextová gramatika***

„Rozumné“ syntaktické analýzy lze dosáhnout pomocí bezkontextové gramatiky s řízeným přepisováním.

Syntaktická analýza bezkontextových jazyků

♦ ***Shora – dolů (top – down parsing)***

Vycházíme z počátečního symbolu gramatiky a snažíme se postupně vygenerovat analyzovaný řetězec.

♦ ***Zdola – nahoru (bottom – up parsing)***

Vycházíme z analyzovaného řetězce a snažíme se ho postupně redukovat na počáteční symbol.

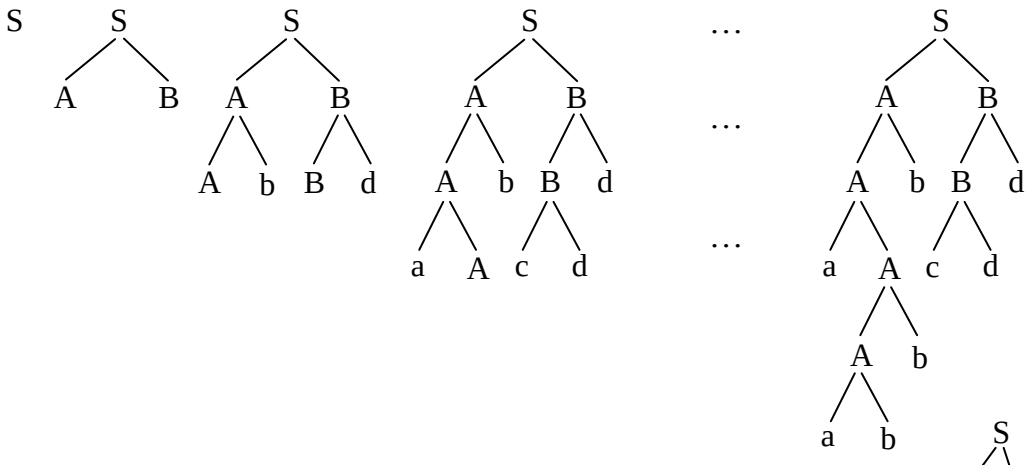
Př. 4.1. $G = (V_N, V_T, P, S)$

$V_N = \{S, A, B\}$ $V_T = \{a, b, c, d\}$

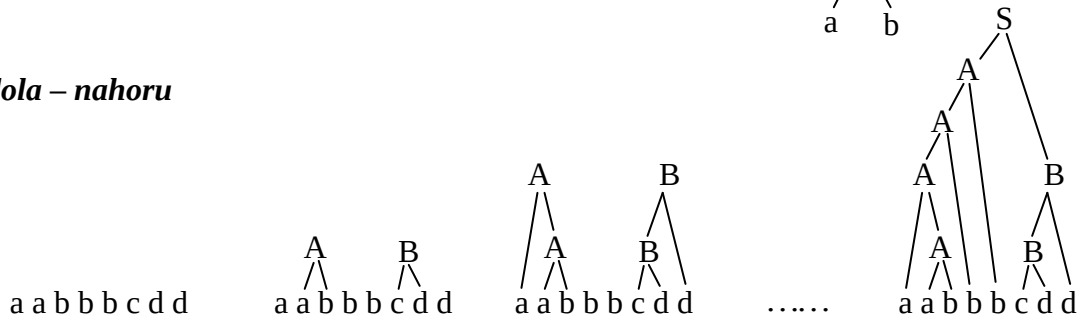
P: $S \to A\ B$, $A \to a\ b$, $A \to a\ A$, $A \to A\ b$, $B \to c\ d$, $B \to c\ B$, $B \to B\ d$

ω = aabbbcdd

♦ ***Shora – dolů***



♦ ***Zdola – nahoru***



4.2 Problém volby pravidel

V určitém stavu syntaktické analýzy může nastat situace, kdy máme možnost volit z více pravidel:

$X \to \omega_1$
$X \to \omega_2$
$\vdots$

Jak určit, které má být použito?

- ***Backtracking***

V případě, že po volbě jednoho pravidla dojdeme k neúspěchu, musíme se vrátit k poslednímu pravidlu, kde byla možnost volby a zvolíme další pravidlo. Tento způsob je však algoritmicky náročný.

- ***Prohledávání do šířky***

V tomto případě provádíme všechny možné rozbory analyzovaného řetězce najednou.

Využití heuristické informace

Omezíme volbu pravidel pomocí podmínek založených např. na znalosti:

- ***Délky řetězce*** – přesáhne-li délka průběžně odvozovaného řetězce délku řetězce analyzovaného, můžeme ho prohlásit za „***neúspěch***“, protože z každého neterminálního symbolu lze odvodit minimálně jeden terminální symbol. (Platí za předpokladu, že gramatika neobsahuje pravidla typu $X \to \lambda$)
- ***Výskytu terminálního symbolu*** – Aplikaci pravidla můžeme zamítnout, pokud by vedlo ke generování terminálního symbolu, který se v analyzovaném řetězci vůbec nevyskytuje.
- ***Levých krajních terminálních symbolů*** – všech řetězců, které lze z daného neterminálního symbolu odvodit.

Př. 4.3. $G = (V_N, V_T, P, S)$

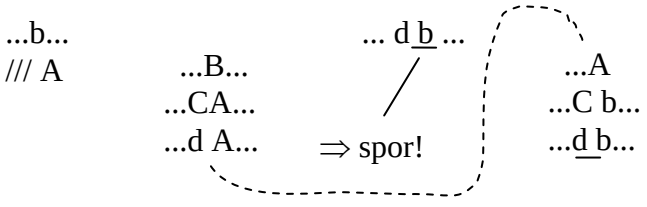
$V_N = \{S, A, B, C\}$

$V_T = \{a, b, c, d\}$

P: $S \to A\ B$, $A \to A\ B$, $A \to a\ B$, $A \to C\ b$, $B \to B\ d$, $B \to d\ B$, $B \to C\ A$, $C \to C\ d$, $C \to d$

	a	b	c	d
S	Ano	Ne	Ne	Ano
A	Ano	Ne	Ne	Ano
B	Ne	Ne	Ne	Ano
C	Ne	Ne	Ne	Ano

Vytvořili jsme tabulku určující, které terminální symboly lze odvodit z neterminálních symbolů jako první zleva.



4.3 Syntaktická analýza shora – dolů

Vycházíme z počátečního symbolu a snažíme se vygenerovat analyzovaný řetězec. Dosud vygenerovaný řetězec ukládáme do zásobníku. Vždy, když se na vrcholu zásobníku objeví terminální symbol, porovná se s aktuálním vstupním symbolem analyzovaného řetězce. V případě souhlasu se terminální symbol z vrcholu zásobníku odstraní. V opačném případě se vrátím tak daleko, kde lze zvolit jiné pravidlo (backtracking).

Př. 4.4 $G = (V_N, V_T, P, S)$ $V_N = \{S, T, I\}$ $V_T = \{a, b, c, f, g\}$

P: (1) $S \to T$ (5) $I \to a$
(2) $S \to T\ f\ S$ (6) $I \to b$
(3) $T \to I\ g\ T$ (7) $I \to c$
(4) $T \to I$

Analyzovaný řetězec: ω = afbgc

Průběh syntaktické analýzy (pomocí backtrackingu):

krok	obsah zásobníku	čtený symbol	vstupní řetězec	pravidlo	č.	další možnost	návrat na krok
1	S		a f b g c	$S \to T$	(1)	(2)	
2	T		a f b g c	$T \to I\ g\ T$	(3)	(4)	
3	I g T		a f b g c	$I \to a$	(5)	(6),(7)	
4	a g T	a	f b g c				
5	g T	f	b g c				3
3	I g T		a f b g c	$I \to b$	(6)	(7)	
6	b g T	a	f b g c				3
3	I g T		a f b g c	$I \to c$	(7)		
7	c g T	a	f b g c				2
2	T		a f b g c	$T \to I$	(4)		
8	I		a f b g c	$I \to a$	(5)	(6), (7)	
9	a	a	f b g c				
10	-	f	b g c				8
8	I		a f b g c	$I \to b$	(6)	(7)	
11	b	a	f b g c				8
8	I		a f b g c	$I \to c$	(7)		
12	c	a	f b g c				1
1	S		a f b g c	$S \to T\ f\ S$	(2)		
13	T f S		a f b g c	$T \to I\ g\ T$	(3)	(4)	
14	I g T f S		a f b g c	$I \to a$	(5)	(6), (7)	
15	a g T f S	a	f b g c				
16	g T f S	f	b g c				14
14	I g T f S		a f b g c	$I \to b$	(6)	(7)	
17	b g T f S	a	f b g c				14
14	I g T f S		a f b g c	$I \to c$	(7)		
18	c g T f S	a	f b g c				13
13	T f S		a f b g c	$T \to I$	(4)		
19	I f S		a f b g c	$I \to a$	(5)	(6), (7)	
20	a f S	a	f b g c				
21	f S	f	b g c				
22	S		b g c	$S \to T$	(1)	(2)	

pokrač.

krok	obsah zásobníku	čtený symbol	vstupní řetězec	pravidlo	č.	další možnost	návrat na krok
23	T		b g c	T → I g T	(3)	(4)	
24	I g T		b g c	I → a	(5)	(6), (7)	
25	a g T	b	g c				24
24	I g T		b g c	I → b	(6)	(7)	
26	b g T	b	g c				
27	g T	g	c				
28	T		c	T → I g T	(3)	(4)	
29	I g T		c	I → a	(5)	(6), (7)	
30	a g T	c					29
29	I g T		c	I → b	(6)	(7)	
31	b g T	c					29
29	I g T		c	I → c	(7)		
32	c g T	c					
33	g T	-	-				28
28	T		c	T → I	(4)		
34	I		c	I → a	(5)	(6), (7)	
35	a	c					34
34	I		c	I → b	(6)	(7)	
36	b	c					34
34	I		c	I → c	(7)		
37	c	c					
38	-		-				

řetězec je přijat

Realizace: Postup kroků uložit do zásobníku.

4.4 Syntaktická analýza zdola – nahoru

Postupujeme od analyzovaného řetězce směrem k počátečnímu symbolu. Analýza začíná s prázdným zásobníkem. V případě úspěšného přijetí řetězce zůstane v zásobníku pouze počáteční symbol.

Př. 4.5. Stejná gramatika jako v Př.4.4. i stejný analyzovaný řetězec ω.

obsah zásobníku	vstupní symbol	pravidlo	č.pravidla	instrukce automatu
-	a	-	-	načtení
a	-	I → a	(5)	redukce
I	-	T → I	(4)	redukce
T	f			načtení
T f	b			načtení
T f b	-	I → b	(6)	redukce
T f I	g			načtení
T f I g	c			načtení
T f I g c	-	I → c	(7)	redukce
T f I g I	-	T → I	(4)	redukce
T f I g T	-	T → I g T	(3)	redukce
T f T	-	S → T	(1)	redukce
T f S	-	S → T f S	(2)	redukce
S	-	-	-	-

řetězec je přijat

Pozn.: zapsány jsou pouze kroky bez navrácení !

4.5 Dva efektivní algoritmy pro syntaktickou analýzu

Čas potřebný k syntaktické analýze může v nejhorších případech narůstat až exponenciálně s délkou řetězce.
Existují algoritmy, které zaručují kratší čas, potřebný k syntaktické analýze.

4.5.1 Cocke – Younger – Kasami algoritmus

Tento algoritmus zaručuje, že čas potřebný k syntaktické analýze je úměrný pouze třetí mocnině délky řetězce. Uskutečňuje syntaktickou analýzu zdola – nahoru.

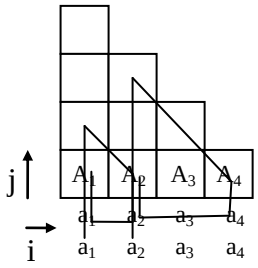
Algoritmus CYK

Vstup:

- Bezkontextová gramatika $G = (V_N, V_T, P, S)$ v Chomského normální formě.
- Vstupní analyzovaný řetězec $\omega = a_1\, a_2 \ldots a_n$

Výstup:

- Tabulka T pro analyzovaný řetězec taková, že t_{ij} obsahuje $A \in V_N$ právě tehdy, když:



$$A \overset{*}{\Rightarrow} a_i\ a_{i+1} \ldots a_{i+j-1}$$

j ... délka odvoditelného řetězce

- ***Krok 1***

  Položíme $t_{i,1} = \{A \mid A \to a_i \in P\}$ i = 1, 2,..., n

  Pozn.: platí $A \overset{*}{\Rightarrow} a_i$

- ***Krok 2***

  Opakuj pro j = 2, 3,..., n
  i = 1, 2,..., n-j+1

  - Předpokládáme, že $t_{i,k}$ bylo stanoveno (naplněno) pro všechna i = 1,..., n-k+1, k = 1,..., j.
  - $t_{ij} = \{A \mid A \to B\, C \in P, \exists\, k \in \langle 1, j): B \in t_{ik}, C \in t_{i+k,j-k}\}$

  Z faktu, že **k < j** a **j-k < j** vyplývá, že:

  $$B \overset{*}{\Rightarrow} a_i\ a_{i+1} \ldots a_{i+k-1}$$

  $$C \overset{*}{\Rightarrow} a_{i+k}\ a_{i+k+1} \ldots a_{i+j-1}$$

  a tedy $A \overset{*}{\Rightarrow} a_i\ a_{i+1} \ldots a_{i+j-1}$

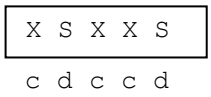
- ***Krok 3***

  Jestliže $S \in t_{1n}$, pak řetězec je přijat, tj. $\omega \in L(G)$.

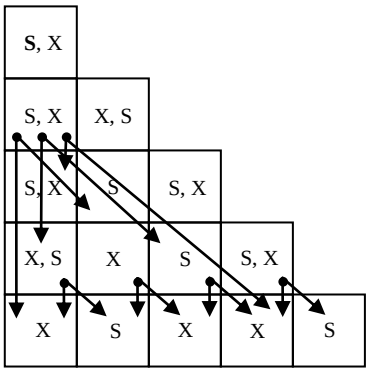
Př. 4.6. G = (V_N, V_T, P, S) v Chomského NF.
V_N = {S, X}
V_T = {c, d}

P: $S \rightarrow X\ X$ $X \rightarrow S\ X$ $S \rightarrow d$
$S \rightarrow X\ S$ $X \rightarrow X\ S$ $X \rightarrow c$ ω = c d c c d

Řádek 1:



Řádek 2, 3, 4, 5:



4.5.2 Earleyho algoritmus

Uskutečňuje syntaktickou analýzu shora – dolů. Provádí všechny možné způsoby analýzy současně takovým způsobem, že často může zkombinovat již získané částečné výsledky. Čas potřebný k syntaktické analýze je úměrný třetí mocnině délky řetězce. Není-li gramatika víceznačná, je čas potřebný k analýze úměrný dokonce jen druhé mocnině délky řetězce.

Algoritmus

Vstup:

- Bezkontextová gramatika G = (V_N, V_T, P, S)
- Analyzovaný řetězec $\omega = a_1\ a_2\ ...\ a_n$

Výstup:

- Seznamy $I_0\ I_1\ ...\ I_n$ pro analyzovaný řetězec

➢ ***Krok 1:***

číslo seznamu, kde byla položka zavedena

Zkonstruujeme seznam I_0

- ❖ Pro každé pravidlo $S \rightarrow \alpha \in P$ přidáme do I_0 položku $[S \rightarrow . \alpha, 0]$
- ❖ Provádíme tak dlouho, dokud lze do I_0 přidat položku,
  pokud je v I_0 položka $[A \rightarrow . B \beta, 0]$
  přidáme do I_0 pro každé pravidlo $B \rightarrow \gamma$ položku $[B \rightarrow . \gamma, 0]$
  (≈ rozvedení neterminálů)

➢ ***Krok 2:***

Zkonstruujeme seznamy $I_1, I_2, ..., I_n$

pro i = 1, 2, ..., n:

- ❖ ***Operace porovnání***

Pro ∀ položky z I_{i-1} ve tvaru $[B \rightarrow \alpha . a \beta, j]$ takovou,

že a = a_i přidáme do I_i položku: [B → α a . β, j]

Provádíme následující body kroku 2 tak dlouho, dokud do I_i lze přidat nějakou položku.

❖ ***Operace kompletace***

Pro každou položku typu [A → γ . , j] v I_i prohledáme seznam I_j a najdeme položky typu: [B → α . A β, k]. Pro každou takovou položku přidáme do I_i položku: [B → α A . β, k].

❖ ***Operace predikce***

Pro každou položku typu: [B → α . C β, j] v I_i přidáme do I_i pro všechna C → γ položky [C → . γ, i].

➢ ***Krok 3:***

Pokud je v I_n položka [S → α . , 0], pak řetězec je přijat, tj. ω ∈ L(G).

Př. 4.7. G = (V_N, V_T, P, S)

V_N = {S, T, F} V_T = {a, +, *, (,) }

P: (1) S → S + T (4) S → T
(2) T → T * F (5) T → F
(3) F → (S) (6) F → a

ω = a * a

Postup syntaktické analýzy pomocí ***Earleyho algortimu***:

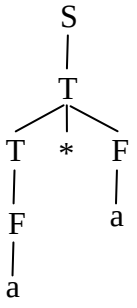
➢ I_0:
- [S → . S + T, 0]
  [S → . T, 0]
- [T → . T * F, 0]
  [T → . F, 0]
  [F → . (S), 0]
  [F → . a, 0]

➢ I_1:
- [F → a . , 0]
- [T → F . , 0]
  [S → T . , 0]
  [T → T . * F, 0]
  [S → S . + T, 0]
- nelze

➢ I_2:
- [T → T * . F, 0]
- nelze
- [F → . (S), 2]
  [F → . a, 2]

➢ I_3:
- [F → a . , 2]
- [T → T * F . , 0]
  [S → T . , 0]
  [T → T . * F, 0]
  [S → S . + T, 0]

Od poslední položky [S → T . , 0] lze vystopovat posloupnost položek, které vedly k přidání dané položky do seznamu.



Z posloupnosti:

[S → T . , 0] v I_3 – [T → T * F . , 0] v I_3 – [F → a . , 2] v I_3 –
- [T → T . * F, 0] v I_1 – [T → F . , 0] v I_1 – [F → a . , 0] v I_1

Lze odvodit derivaci řetězce ω:

$$S \overset{(4)}{\Rightarrow} T \overset{(2)}{\Rightarrow} T*F \overset{(5)}{\Rightarrow} F*F \overset{(6)}{\Rightarrow} a*F \overset{(6)}{\Rightarrow} a*a$$

5 VYUŽITÍ STOCHASTICKÝCH JAZYKŮ PRO ROZPOZNÁVÁNÍ

5.1 Stochastický jazyk a gramatika

V praxi často platí $L(G_i) \cap L(G_j) \neq \emptyset$, $i \neq j$, tj. dochází k překrývání jazyků popisujících třídy Ω_i a Ω_j.

Příčiny:

- Strukturální podobnost obrazů z různých tříd.
- Nejrůznější šumy a poruchy.
- Nepřesnost v procesu předzpracování obrazů, detekce primitiv a relací.
- Nepřesnost při konstrukci či inferenci gramatik, charakterizujících jednotlivé třídy obrazů.

Analogie s příznakovými metodami, kde obrazy patřící do různých tříd mohou být charakterizovány stejným vektorem příznaků, ale s různou pravděpodobností. Pro řešení tohoto problému v rámci strukturálního přístupu musíme gramatiku doplnit o pravděpodobnost výskytu každého slova v daném jazyce. K tomu využíváme **stochastických jazyků a gramatik.**

Definice: **Stochastická gramatika** $G_S = (V_N, V_T, P_S, S)$

V_N ... množina neterminálních symbolů

V_T ... množina terminálních symbolů

S ... startovací symbol

P_S ... množina stochastických přepisovacích pravidel ve tvaru:

$$\alpha_i \overset{p_{ij}}{\rightarrow} \beta_{ij} \qquad \begin{matrix} i = 1, 2, \ldots k \\ j = 1, 2, \ldots, n_i \end{matrix}$$

$\alpha_i, \beta_{ij} \in V^* \quad V = V_N \cup V_T$

$\alpha_i \notin V_T^*$

p_{ij} ... pravděpodobnost spojená s aplikací tohoto pravidla $p_{ij} \in (0, 1>$

$$\sum_{j=1}^{n_i} p_{ij} = 1 \qquad \forall i = 1, 2, \ldots, k$$

k ... počet různých levých stran pravidel

n_i ... počet pravidel s levou stranou α_i

Pravděpodobnost spojená s odvozením

Existuje-li posloupnost řetězců $\omega_1, \omega_2, \ldots, \omega_{n+1}$ taková, že $\varphi = \omega_1$, $\gamma = \omega_{n+1}$ a $\omega_i \overset{p_i}{\Rightarrow} \omega_{i+1}$ pro i = 1, 2,..., n, potom z φ lze odvodit γ s pravděpodobností $p = \prod_{i=1}^{n} p_i$. To zapíšeme takto:

$$\boxed{\varphi \overset{p}{\Rightarrow} \gamma}$$

Stochastický jazyk je pak definován takto:

$L(G_S) = \{ (\omega, p(\omega)) \mid \omega \in V_T^*, S \overset{p(\omega)}{\Rightarrow} \omega \}$... platí pro jednoznačnou gramatiku.

$L(G_S) = \{ (\omega, p(\omega)) \mid \omega \in V_T^*, S \overset{p_j}{\Rightarrow} \omega, j = 1, 2,..., k \quad p(\omega) = \sum_{j=1}^{k} p_j \}$

k ... počet odlišných derivací řetězce ω
p_j ... pravděpodobnost spojená s j-tou derivací

$L(G_S)$ může být charakterizován dvojicí (L, p).

L ... charakteristický jazyk $L(G_S)$
p ... pravděpodobnostní rozložení definované nad L

- Platí-li $\sum_{\omega \in L} p(\omega) = 1$ říkáme, že stochastická gramatika je **konzistentní.**

Př. 5.1. Regulární gramatika $G = (V_N, V_T, P, S)$

$V_N = \{S, A, B\}$

$V_T = \{a, b\}$

P: $S \to a\ A$, $A \to b\ B$, $A \to a$, $B \to b$, $B \to a\ S$

$L = \{ (a\ b\ a)^n\ a\ a, (a\ b\ a)^n\ a\ b\ b, n = 0, 1,...\}$

Stochastická gramatika vznikne doplněním pravděpodobností k jednotlivým pravidlům.

$G_S = (V_N, V_T, P_S, S)$

P_S: $S \overset{1}{\to} a\ A$, $A \overset{0,7}{\to} b\ B$, $A \overset{0,3}{\to} a$, $B \overset{0,4}{\to} b$, $B \overset{0,6}{\to} a\ S$

Příklad derivace:

$S \Rightarrow a\ A \Rightarrow a\ b\ B \Rightarrow a\ b\ a\ S \Rightarrow a\ b\ a\ a\ A \Rightarrow a\ b\ a\ a\ b\ B \Rightarrow a\ b\ a\ a\ b\ b$

$p(a\ b\ a\ a\ b\ b) = 1 \cdot 0{,}7 \cdot 0{,}6 \cdot 1 \cdot 0{,}7 \cdot 0{,}4 = 0{,}1176$

$L_S = L(G_S)$

$$\sum_{\omega \in L_S} p(\omega) = \sum_{n=0}^{\infty} (0{,}3 + 0{,}28) \cdot (0{,}42)^n = 1$$

Generovaný řetězec ω	***p(ω)***
a a	0,3
a b b	0,28
$(a\ b\ a)^n\ a\ a$	$0{,}3 \cdot (0{,}42)^n$
$(a\ b\ a)^n\ a\ b\ b$	$0{,}28 \cdot (0{,}42)^n$

5.2 Využití stochastických gramatik při klasifikaci

Výhody využití stochastických gramatik při klasifikaci:

- Lze vyjádřit skutečnost, že některé obrazy se v určité třídě vyskytují častěji než jinde.
- Je možno přiřadit „nechtěným“ řetězcům nereprezentujícím žádné obrazy malé hodnoty pravděpodobnosti.
- Lze řešit problém víceznačnosti tím, že jestliže má řetězec reprezentující obraz více různých derivačních stromů, můžeme za úplný strukturální popis považovat ten nejpravděpodobnější.

Rozpoznávání podle stochastického jazyka probíhá takto:

- Provedeme syntaktickou analýzu řetězce ω podle gramatik všech tříd Ω_j, j = 1, 2,..., m.
- Tím vlastně získáme pravděpodobnosti $p(\omega \mid \Omega_j) = p(\omega \mid G_j)$.
- Na základě znalostí apriorních pravděpodobností výskytu tříd $p(\Omega_j)$, j = 1, 2,..., m můžeme použít klasifikátor, založený na ***Bayesovském*** klasifikačním pravidle.
- Apriorní pravděpodobnost, že řetězec ω reprezentující neznámý obraz byl generován stochastickou gramatikou G_j, tj. aposteriorní pravděpodobnost, s jakou analyzovaný řetězec patří do třídy Ω_j:

$$P(G_j \mid \omega) = \frac{p(\omega \mid G_j) \cdot P(G_j)}{p(\omega)} = \frac{p(\omega \mid G_j) \cdot P(G_j)}{\sum_{i=1}^{m} p(\omega \mid G_i) \cdot P(G_i)}$$

$P(G_j) = P(\Omega_j)$ ... apriorní pravděpodobnost výskytu třídy Ω_j.

$P(\omega \mid G_j) = P(\omega \mid \Omega_j)$ ... pravděpodobnost generování řetězce ω gramatikou G_j, která reprezentuje třídu Ω_j.

$P(\omega)$ ... pravděpodobnost výskytu obrazu reprezentovaného řetězcem ω v řešené úloze vůbec.

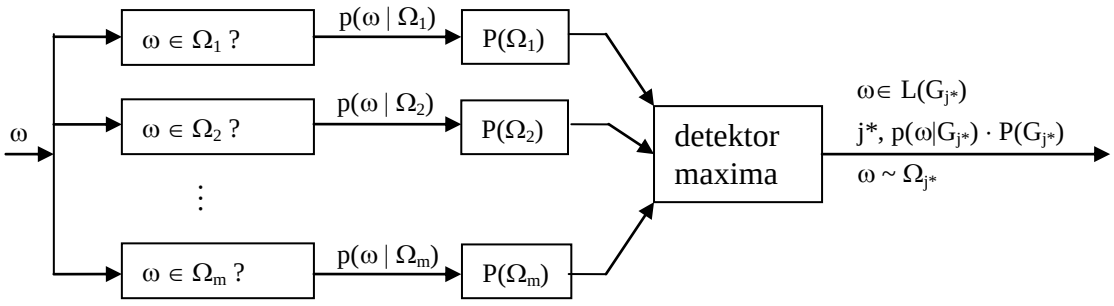
- Klasifikujeme na základě principu maximální pravděpodobnosti:

$$\omega \sim \Omega_{j*}: \; P(G_{j*} \mid \omega) = \max_{j=1,2,\ldots,m} \{P(G_j \mid \omega)\} = \max_{j=1,2,\ldots,m} \left\{ \frac{P(\omega \mid G_j) \cdot P(G_j)}{P(\omega)} \right\}$$

- Protože hodnota $P(\omega)$ nemá vliv na výsledek klasifikace, můžeme vztah upravit na:

$$\omega \sim \Omega_{j*}: \; p(\omega \mid G_{j*}) \cdot P(G_{j*}) = \max_{j=1,2,\ldots,m} \left\{ p(\omega \mid G_j) \cdot P(G_j) \right\}$$

- Blokové schéma stochastického strukturálního klasifikátoru:



5.3 Stanovení pravděpodobností pravidel

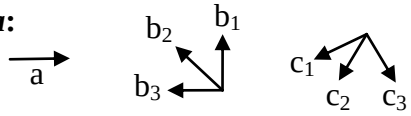
Máme-li k dispozici trénovací množinu řetězců a jim odpovídající pravděpodobnosti (četnosti), je za určitých předpokladů možné vypočítat nebo odhadnout pravděpodobnosti jednotlivých pravidel stochastické gramatiky tak, aby generovala řetězce s těmito pravděpodobnostmi.

5.3.1 Stochastické regulární gramatiky

Př. 5.2. Stochastická regulární gramatika $G_S = (V_N, V_T, P_S, S)$

$V_N = \{S, A_1, A_2, A_3, A_4\}$ $V_T = \{a, b_1, b_2, b_3, c_1, c_2, c_3\}$

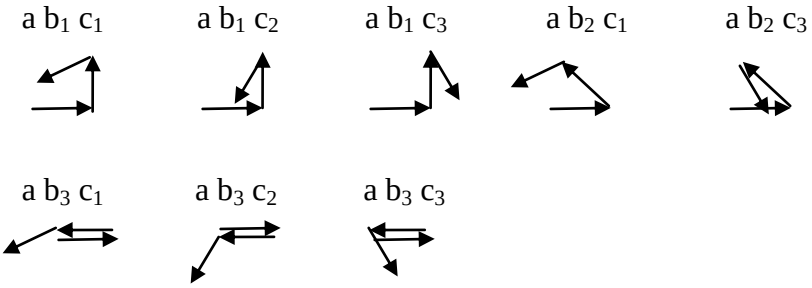
***Primitiva*:**



Generované trojúhelníky

Rovnostranný trojúhelník:  a b_2 c_2

a jeho deformace:



Pravděpodobnosti jednotlivých řetězců mohou být odhadnuty na základě dlouhodobého pozorování četnosti jejich výskytu.

ω	p(ω)
ω_1 = a b_1 c_1	1/36
ω_2 = a b_1 c_2	2/36
ω_3 = a b_1 c_3	3/36
ω_4 = a b_2 c_1	1/36
ω_5 = a b_2 c_2	21/36
ω_6 = a b_2 c_3	2/36
ω_7 = a b_3 c_1	3/36
ω_8 = a b_3 c_2	2/36
ω_9 = a b_3 c_3	1/36

P: (1) $S \xrightarrow{p_1} a\ A_1$

(2) $A_1 \xrightarrow{p_2} b_1\ A_2$

(3) $A_1 \xrightarrow{p_3} b_2\ A_3$

(4) $A_1 \xrightarrow{p_4} b_3\ A_4$

(5) $A_2 \xrightarrow{p_5} c_1$

(6) $A_2 \xrightarrow{p_6} c_2$

(7) $A_2 \xrightarrow{p_7} c_3$

(8) $A_3 \xrightarrow{p_8} c_1$

(9) $A_3 \xrightarrow{p_9} c_2$

(10) $A_3 \xrightarrow{p_{10}} c_3$

(11) $A_4 \xrightarrow{p_{11}} c_1$

(12) $A_4 \xrightarrow{p_{12}} c_2$

(13) $A_4 \xrightarrow{p_{13}} c_3$

Normalizační podmínky:

$p_1 = 1$

$p_2 + p_3 + p_4 = 1$

$p_5 + p_6 + p_7 = 1$

$p_8 + p_9 + p_{10} = 1$

$p_{11} + p_{12} + p_{13} = 1$

Pro generování řetězce ω_1 musíme použít pravidlo (1), (2), (5), takže $p(\omega_1) = p_1 \cdot p_2 \cdot p_5$
Obdobně:

$p(\omega_2) = p_1\, p_2\, p_6$ $p(\omega_5) = p_1\, p_3\, p_9$ $p(\omega_7) = p_1\, p_4\, p_{11}$
$p(\omega_3) = p_1\, p_2\, p_7$ $p(\omega_6) = p_1\, p_3\, p_{10}$ $p(\omega_8) = p_1\, p_4\, p_{12}$
$p(\omega_4) = p_1\, p_3\, p_8$ $p(\omega_9) = p_1\, p_4\, p_{13}$

Z rovnic:

$p_1 = 1$

$$p(\omega_1) + p(\omega_2) + p(\omega_3) = p_1 p_2 \cdot \underbrace{\left(p_5 + p_6 + p_7\right)}_{1} = p_2 = \frac{1+2+3}{36} = \frac{6}{36} = \frac{1}{6}$$

$$p_1 \cdot p_2 \cdot p_5 = \frac{1}{36} = 1 \cdot \frac{1}{6} p_5$$

$$p_5 = \frac{6}{36} = \frac{1}{6}; \quad p_6 = \frac{2}{36} \cdot \frac{6}{1} = \frac{2}{6} = \frac{1}{3}; \quad p_7 = \frac{3}{36} \cdot \frac{6}{1} = \frac{3}{6} = \frac{1}{2}$$

Obdobně získáme:

$p_3 = 2/3$ $p_8 = 1/24$ $p_9 = 7/8$ $p_{10} = 1/12$
$p_4 = 1/6$ $p_{11} = 1/2$ $p_{12} = 1/3$ $p_{13} = 1/6$

Stochastická gramatika s takto stanovenými pravděpodobnostmi jednotlivých pravidel bude generovat řetězce s pravděpodobnostmi uvedenými v tabulce.

5.3.2 Stochastické bezkontextové gramatiky

Předpokládáme trénovací množinu $M_t = \{ (\omega_1, f_1), (\omega_2, f_2),..., (\omega_t, f_t)\}$, kde f_k, k = 1, 2,..., t je odhadnutá pravděpodobnost výskytu řetězce ω_k, popř. pravděpodobnost subjektivně určená řešitelem, chce-li např. generování některých řetězců potlačit. Platí:

$$M_t \subseteq L(G)$$

ale většinou: $M_t \subset L(G)$ ... často $|L(G)| \to \infty$

Bezkontextová gramatika G:

$A_i \overset{p_{ij}}{\to} \gamma_{ij}$ $A_i \in V_N$ $\gamma_{ij} \in V^+$

$p_{ij} = ?$ ... odhad označíme $\hat{p}_{ij}$

Provedeme syntaktickou analýzu všech vzorků řetězců ω_k z trénovací množiny M_t a pro každý řetězec ω_k zjistíme absolutní četnosti $N_{ij}(\omega_k)$ výskytu pravidel $A_i \to \gamma_{ij}$ použitých při jeho derivaci. Očekávaná absolutní četnost n_{ij} výskytu pravidla $A_i \to \gamma_{ij}$ při syntaktické analýze všech vzorků z M_t je:

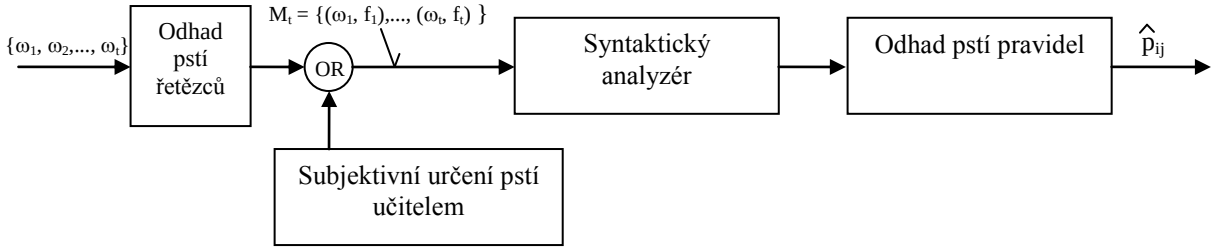
$$n_{ij} = \sum_{\omega_k \in M_t} f_k N_{ij}(\omega_k)$$

Maximálně pravděpodobný odhad pro p_{ij} vypočteme podle vztahu:

$$\hat{p}_{ij} = \frac{n_{ij}}{\sum_l n_{il}} \quad \forall l : A_i \to \gamma_{il}$$

$\hat{p}_{ij}$ konverguje k p_{ij}, pokud M_t konverguje k L(G) a odhady f_k konvergují ke skutečným hodnotám.

Blokový diagram systému pro odvození pravidel



Ve složitějším případě je známo, že řetězce jsou generovány bezkontextovými gramatikami:

$$G_q = (V_{Nq}, V_{Tq}, P_q, S_q) \qquad q = 1, 2,..., m,$$

kde *m* je počet tříd.

Trénovací množina $M_t \subseteq L$, kde $L = L(G_1) \cup L(G_2) \cup ... \cup L(G_m)$

Pro řetězce z M_t je známa aposteriorní pravděpodobnost $P(G_q \mid \omega_k)$, že řetězec ω_k byl generován gramatikou G_q pro $k = 1,2,..., l$ a $q = 1, 2,..., m$ platí:

$$\sum_{q=1}^{m} P(G_q \mid \omega_k) = 1$$

Nejdříve vypočteme:

$$n_{qij} = \sum_{\omega_k \in M_t} f_k \cdot P(G_q \mid \omega_k) \cdot N_{qij}(\omega_k)$$

f_k ... odhadnutá pravděpodobnost (relativní četnost) výskytu obrazu reprezentovaného řetězcem ω_k.

$N_{q\,ij}(\omega_k)$ ... absolutní četnost výskytu pravidla $A_i \to \gamma_{ij}$ v derivaci řetězce ω_k podle gramatiky G_q.

$n_{q\,ij}$ ... očekávaná absolutní četnost výskytu pravidla $A_i \to \gamma_{ij}$ z P_q v derivacích všech řetězců $\omega_k \in M_t$ podle gramatiky G_q.

Potom maximálně pravděpodobný odhad pravděpodobnosti $\hat{p}_{q\,ij}$ příslušející pravidlu $A_i \to \gamma_{ij}$ z P_q vypočteme podle vztahu:

$$\hat{p}_{qij} = \frac{n_{qij}}{\sum_{l} n_{qil}} \quad \forall l : A_i \to \gamma_{il} \in P_q$$

$\hat{p}_{q_{ij}}$ konverguje k p_{qij} za předpokladu, že s rostoucím *t* M_t konverguje k L a odhadované četnosti f_k konvergují ke skutečným pravděpodobnostem $p(\omega_k)$, pro které platí:

$$p(\omega_k) = \sum_{q=1}^{m} p(\omega_k \mid G_q) \cdot P(G_q)$$

jestliže:

$$\sum_{q=1}^{m} P(G_q) = 1$$

Př. 5.3. $G_S = (V_N, V_T, P_S, S)$ P_S: $S \xrightarrow{p_{11}} a\ X\ S$ $X \xrightarrow{p_{21}} d$

$V_N = \{S, X\}$ $S \xrightarrow{p_{12}} c\ X$ $X \xrightarrow{p_{22}} b\ X$

$V_T = \{a, b, c, d\}$

K dispozici je trénovací množina 100 řetězců, pro které určíme absolutní četnost.

ω_h	Absolutní četnost	Relativní četnost
a d c d	9	0,09
c d	77	0,77
a d c b d	2	0,02
c b d	6	0,06
a b d a d c b d	1	0,01
a b d c d	2	0,02
a b d a d a d c d	1	0,01
a d a d c d	2	0,02
	$\Sigma = 100$	$\Sigma = 1$

Zjistíme výskyty jednotlivých pravidel:

	počet výskytů pravidla				
ω	**S → a X S**	**S → c X**	**X → d**	**X → b X**	**četnost**
adcd	1	1	2	0	9
cd	0	1	1	0	77
adcbd	1	1	2	1	2
cbd	0	1	1	1	6
abdadcbd	2	1	3	3	1
abdcd	1	1	2	1	2
abdadadcd	3	1	4	1	1
addacd	2	1	3	0	2
	22	100	122	14	
	22/122	100/122	122/136	14/136	

f_k

$N_{ii}(\omega)$

$\hat{p}_{ij}$ n_{ij}

Pro celou trénovací množinu.

$S \to a\ X\ S$ $n_{ij} = \sum_k f_k \cdot N_{ij}(\omega_k)$ $n_{11} = 22$

$S \to c\ X$ $n_{12} = 100$

$X \to d$ $n_{21} = 122$

$X \to b\ X$ $n_{22} = 14$

Odhady pravděpodobností jednotlivých pravidel pak vypočteme takto:

$$\hat{p}_{11} = \frac{n_{11}}{\sum_{l=1}^{2} n_{1l}} = \frac{22}{22+100} \approx 0{,}18$$

$$\hat{p}_{12} = \frac{100}{122} \approx 0{,}82 \qquad \hat{p}_{22} = \frac{14}{136} \approx 0{,}10$$

$$\hat{p}_{21} = \frac{122}{136} \approx 0{,}90$$

5.4 Syntaktická analýza stochastických jazyků

Informaci o pravděpodobnostech spojených s jednotlivými pravidly gramatiky můžeme využít v procesu syntaktické analýzy. Využitím speciálních algoritmů lze snížit průměrný počet kroků syntaktické analýzy a tak celý její proces urychlit. Mluvíme o tzv. ***stochastické syntaktické analýze.***

V případě regulárních jazyků postačí zkonstruovat odpovídající, tzv. ***stochastický konečný automat.***

Definice: Stochastický konečný automat je pětice $A_S = (W, Q, M, \alpha_0, F)$

W ... konečná množina vstupních symbolů

Q ... konečná množina stavů ($|Q| = n$)

M ... zobrazení množiny W do množiny matic (n x n) stochastických stavových přechodů

α_0 ... n-dimenzionální řádkový vektor počátečního rozložení stavů

F ... množina koncových stavů ($F \subseteq Q$)

$M(a) = [p_{ij}(a)] \qquad a \in W$

$p_{ij}(a)$ ... pravděpodobnost přechodu ze stavu q_i do stavu q_j při sejmutí symbolu *a* ze vstupní pásky.

V každém kroku musí stochastický konečný automat přejít ze stavu q_i do nějakého (i stejného) stavu q_j, tj. platí:

$$\sum_{j=1}^{n} p_{ij} = 1 \qquad \text{pro } i = 1, 2, ..., n$$

Definiční obor zobrazení M rozšíříme z W na W* následovně:

$M(\lambda) = I$ (identická matice n x n)

$M(a_1\, a_2 \ldots a_k) = M(a_1)\, M(a_2) \ldots M(a_k)$

Jazyk přijímaný stochastickým konečným automatem A_S:

$T(A_S) = \{ (\omega, p(\omega)) \mid \omega \in W^*, p(\omega) = \alpha_0\, M(\omega)\, \alpha_F > 0 \}$

α_F ... n-dimenzionální sloupcový vektor, jehož i-tá složka je rovna 1, pokud $q_i \in F$ a je rovna 0, pokud $q_i \notin F$.

Př. 5.4. Stochastický konečný automat: $A_S = (W, Q, M, \alpha_0, F)$

$W = \{a, b\}$

$Q = \{S, A, B, T, R\}$

$\alpha_0 = [1, 0, 0, 0, 0]$

$\alpha_F = [0, 0, 0, 1, 0]^T$

$F = \{T\}$

T ~ přijímáno
R ~ nepřijímáno

$$S \xrightarrow{1} a\ A$$
$$A \xrightarrow{0,3} a$$
$$B \xrightarrow{0,6} a\ S$$
$$A \xrightarrow{0,7} b\ B$$
$$B \xrightarrow{0,4} b$$

M:

$$M(a) = \begin{array}{c} \\ S \\ A \\ B \\ T \\ R \end{array} \begin{array}{c} \begin{array}{ccccc} S & A & B & T & R \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}3 & 0{,}7 \\ 0{,}6 & 0 & 0 & 0 & 0{,}4 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad M(b) = \begin{array}{c} \\ S \\ A \\ B \\ T \\ R \end{array} \begin{array}{c} \begin{array}{ccccc} S & A & B & T & R \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0{,}7 & 0 & 0 & 0{,}3 \\ 0 & 0 & 0{,}4 & 0 & 0{,}6 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Automat přijímá řetězec a těmito pravděpodobnostmi:

$$p(\omega) = \alpha_0\, M(\omega)\, \alpha_F = \begin{cases} 0{,}42^n \cdot 0{,}3 & \omega = (a\ b\ a)^n\ a\ a \quad n = 0, 1,... \\ 0{,}42^n \cdot 0{,}28 & \omega = (a\ b\ a)^n\ a\ b\ b \quad n = 0, 1,... \end{cases}$$

např. pro ω = a b b

$$p(\omega) = \underbrace{[1\ 0\ 0\ 0\ 0]}_{[0\ 1\ 0\ 0\ 0]} \cdot M(a) \cdot \underbrace{M(b)}_{[0\ 0\ 0{,}7\ 0\ 0{,}3]} \cdot \underbrace{M(b)}_{[0\ 0\ 0\ 0{,}28\ 0{,}72]} \cdot \alpha_F = \underbrace{0{,}28}_{0{,}28}$$

Stochastický konečný automat z př.5.4. přijímá jazyk generovaný stochastickou regulární gramatikou z př.5.1.

Platí:

- $W = V_T$
- $Q = V_N \cup \{T, R\}$

T ... stav přijetí

R ... stav odmítnutí

- α_0 má jedinou složku rovnou 1 v místě odpovídajícím počátečnímu symbolu S
- množina koncových stavů F má jediný prvek – stav T
- α_F má jedinou složku rovnou 1 v místě odpovídajícím stavu T
- pokud P_S obsahuje pravidlo $X_i \overset{p_{ij}}{\rightarrow} x_l X_j$, pak matice $M(x_l)$ má v i-tém řádku a j-tém sloupci prvek p_{ij}.
- Pokud P_S obsahuje pravidlo $X_i \overset{p_{il}}{\rightarrow} x_l$, pak matice $M(x_l)$ má v i-tém řádku na pozici sloupce odpovídajícího stavu T prvek p_{il}.
- Prvky matice $M(x_i)$ $\forall x_i \in W$ ve sloupci odpovídajícím stavu R mají takovou hodnotu, aby součty prvků v řádcích všech matic byly rovny 1.

Ke každé stochastické regulární gramatice lze sestrojit odpovídající stochastický konečný automat.

6 ŘEŠENÍ VLIVU SYNTAKTICKÝCH DEFORMACÍ

Vlivem nejrůznějších poruch, šumů a nepřesností v procesu předzpracování dochází k deformacím ve struktuře rozpoznávaných obrazů. Existují metody, které umožňují rozpoznávat neúplné, poškozené či jinak strukturálně deformované obrazy.

6.1 Template matching

Srovnávání neznámého obrazu se vzorovými obrazy tříd. Prakticky nikdy nedochází k úplné shodě. Proto je třeba zavést vhodnou míru odlišnosti dvou obrazů, podobně jako je tomu u příznakových metod.

Definice: Transformace

pro $\alpha, \beta \in V_T^*$ $\quad T: V_T^* \rightarrow V_T^*$

$\beta = T(\alpha)$

- ***Záměna (substituce) symbolu***

$$\varphi\, a\, \omega \xrightarrow{T_S} \varphi\, b\, \omega \qquad \text{pro a, b} \in V_T;\quad a \neq b;\ \varphi, \omega \in V_T^*$$

- ***Vynechání (vypuštění) symbolu***

$$\varphi\, a\, \omega \xrightarrow{T_D} \varphi\, \omega \qquad \text{pro a} \in V_T;\quad \varphi, \omega \in V_T^*$$

- ***Vložení symbolu***

$$\varphi\, \omega \xrightarrow{T_I} \varphi\, a\, \omega \qquad \text{pro a} \in V_T;\quad \varphi, \omega \in V_T^*$$

Definice: Levenshteinova vzdálenost

$d^L(\alpha, \beta)$ je definována jako nejmenší počet transformací potřebných k odvození β z α.

Např.: α = c b a b d b b $\qquad \beta$ = c b b a b b d b

$$\alpha = \text{c b a b d b b} \xrightarrow{T_I} \text{c b b a b d b b} \xrightarrow{T_I} \text{c b b a b b d b b} \xrightarrow{T_D} \beta$$

$$\underline{\underline{d^L(\alpha, \beta) = 3}}$$

Definice: Váhová Levenshteinova vzdálenost

$$d^W(\alpha, \beta) = \min_{j=1,2,\ldots,J} (w_s\, n_{sj} + w_D\, n_{Dj} + w_I\, n_{Ij})$$

J ...počet různých odvození

n_{sj} ... počet substitucí v j-tém odvození — w_s ... váha substituce

n_{Dj} ... počet vypuštění v j-tém odvození — w_D ... váha vypuštění

n_{Ij} ... počet vložení v j-tém odvození — w_I ... váha vložení

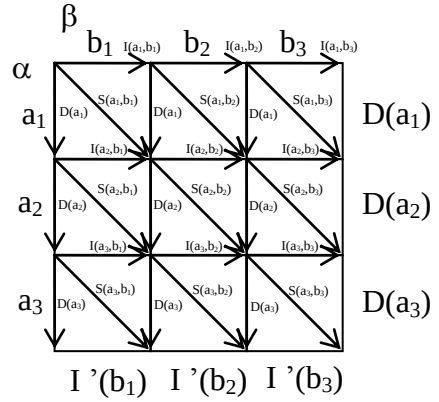
Definice: Váhová Levenshteinova vzdálenost, která splňuje axiomy metriky a respektuje skutečnost, že různé typy transformací jsou obvykle pro různá primitiva (terminální symboly) různě pravděpodobné.

- $\alpha\ a\ \beta \xrightarrow{T_S, S(a,b)} \alpha\ b\ \beta$; $a, b \in V_T \quad a \neq b$
  S(a,b) ... váha (cena) substituce symbolu b za a.
- $\alpha\ a\ \beta \xrightarrow{T_D, D(a)} \alpha\ \beta \quad a \in V_T$
  D(a) ... váha (cena) vynechání symbolu a.
- $\alpha\ a\ \beta \xrightarrow{T_I, I(a,b)} \alpha\ b\ a\ \beta \quad a, b \in V_T$
  I(a, b) ... váha (cena) vložení symbolu b před a.
- $\alpha \xrightarrow{T_I', I'(a)} \alpha\ a \quad a \in V_T$
  I'(a) ... váha (cena) vložení symbolu *a* na konec řetězce

Tato vzdálenost je nazývána ***váhová metrika:***

$$d^W(\alpha, \beta) = \min_j W_j$$

Grafické znázornění:



Váhová metrika je minimum součtu vah v dané cestě přes všechny průchody grafem.

Definice: Stochastický model

- $\alpha\ a\ \beta \xrightarrow{T_S, q_S(b|a)} \alpha\ b\ \beta \qquad a, b \in V_T$
  $q_S(b \mid a)$ ... pravděpodobnost substituce *b* za *a*.
- $\alpha\ a\ \beta \xrightarrow{T_D, q_D(a)} \alpha\ \beta \qquad a \in V_T$
  $q_D(a)$ ... pst vynechání symbolu *a*.
- $\alpha\ a\ \beta \xrightarrow{T_I,\ q_I(b|a)} \alpha\ b\ a\ \beta \qquad a, b \in V_T$
  $q_I(b \mid a)$ ... pst vložení *b* před *a*.
- $\alpha \xrightarrow{T_I',\ q_I'(a)} \alpha\ a$
  $q_{I'}(a)$ ... pst vložení *a* na konec řetězce.

Deformační psti jsou konzistentní, pokud platí:

$$\sum_{b \in V_T} q_S(b \mid a) + q_D(a) + \sum_{b \in V_T} q_I(b \mid a) = 1 \qquad \forall a \in V_T$$

***Pravděpodobnost deformace* a *na* α**

$q(\alpha \mid a) = q_D(a)$ pro $\alpha = \lambda$

$\max \{ q_S(b \mid a), \; q_I(b \mid a) \cdot q_D(a) \}$ pro $\alpha = b$

$q_I(b_1 \mid a) \cdot q_I(b_2 \mid a) \cdot \ldots q_I(b_{l-1} \mid a) \cdot \max\{ q_S(b_l \mid a), \; q_I(b_l \mid a) \cdot q_D(a) \}$ pro $\alpha = b_1 \ldots b_l$, $l > 1$

Pravděpodobnost vložení* α *na konec řetězce

$q_{I'}(\alpha) = 1 - q_{I'}$ pro $\alpha = \lambda$

$(1 - q_{I'}) \cdot q_{I'}(a_1) \cdot \ldots \cdot q_{I'}(a_l)$ pro $\alpha = a_1 .. a_l$

$q_{I'} = \sum_{a \in V_T} q_{I'}(a)$ $q_{I'}$ ... pst vložení libovolného symbolu na konec řetězce

$(1 - q_{I'})$ ... pst, že na konec řetězce nebude vložen žádný symbol.

***Pravděpodobnost deformace řetězce* β *na řetězec* α**

$$q(\alpha \mid \beta) = \max_i \left\{ \prod_{j=1}^{n-1} q\left(\alpha_j^i \mid a_j\right) \right\} \cdot q_{I'}\left(\alpha_n^i\right)$$

$\alpha_1^i \, \alpha_2^i \ldots \alpha_n^i$ ... i-tý možný způsob rozkladu α na *n* řetězců.

Např.: $\alpha = b_1 \, b_2 \, b_3 \, b_4 \, b_5 \, b_6$ $\beta = a_1 \, a_2 \, a_3 \, a_4$

možné rozklady (některé):

α_1	α_2	α_3	α_4
$b_1 \, b_2$	λ	$b_3 \, b_4 \, b_5$	b_6
λ	$b_1 \, b_2$	$b_3 \, b_4$	$b_5 \, b_6$
$b_1 \, b_2 \, b_3$	b_4	$b_5 \, b_6$	λ
⋮	⋮	⋮	⋮

6.2 Syntaktická analýza s opravou chyb

Třída obrazů je charakterizována gramatikou G. Mějme řetězec ω popisující strukturálně deformovaný obraz této třídy. Podrobíme-li nyní řetězec ω syntaktické analýze, s největší pravděpodobností zjistíme, že $\omega \notin L(G)$ a obraz nebude do dané třídy klasifikován. Cílem syntaktické analýzy s opravou chyb je nalézt nedeformovaný řetězec φ, $\varphi \in L(G)$ takový, že jeho vzdálenost k ω je ze všech řetězců jazyka L(G) nejmenší.

$$d(\omega, \varphi) = \min_{\psi} \{ d(\omega, \psi) \mid \psi \in L(G) \}$$

Metoda nalezení takového řetězce spočívá v rozšíření původní gramatiky o chybová – deformační pravidla.

Algoritmus konstrukce rozšířené gramatiky

Vstup: Bezkontextová gramatika $G = (V_N, V_T, P, S)$.
Výstup: Rozšířená gramatika $G' = (V_N', V_T', P', S')$, kde P' je množina váhových pravidel.

- ***Krok 1:***

  $V_N' = V_N \cup \{S'\} \cup \{E_b \mid b \in V_T\}$

  $V_T \subseteq V_T'$

- ***Krok 2:***

  Je-li v P pravidlo:

  $A \rightarrow \alpha_0\, b_1\, \alpha_1\, b_2 \ldots \alpha_{m-1}\, b_m\, \alpha_m \quad m \geq 0$

  $\alpha_l \in V_N^* \ \wedge\ b_i \in V_T \quad i = 1, 2, \ldots, m,\ l = 0, 1, \ldots, m$

  Potom do P' přidej pravidlo:

  $A \rightarrow \alpha_0\, E_{b1}\, \alpha_1\, E_{b2} \ldots \alpha_{m-1}\, E_{bm}\, \alpha_m$ s váhou 0.

- ***Krok3:***

  Do P' přidej následující pravidla s vahou odpovídající zvolené vzdálenosti (Levenschteinova vzdálenost, váhová Levenschtein.vzd. w, váhová metrika W).

pravidlo	***L***	***w***	***W***	***pro***
a) $S' \rightarrow S$	0	0	0	
b) $S' \rightarrow S\ a$	1	w_I	I'(a)	$a \in V_T'$
c) $E_a \rightarrow a$	0	0	0	$a \in V_T$
d) $E_a \rightarrow b$	1	w_S	S(a, b)	$a \in V_T, b \in V_T'\ a \neq b$
e) $E_a \rightarrow \lambda$	1	w_D	D(a)	$a \in V_T$
f) $E_a \rightarrow b\ E_a$	1	w_I	I(a, b)	$a \in V_T, b \in V_T'$

Pozn.: Pravidla typu b) d) e) f) jsou nazývána **deformačními pravidly**.

S rozšířenou gramatikou pracuje **Syntaktický analyzátor s opravou chyb**, který vyhledává takovou deformaci řetězce ω, která je spojena s nejmenším součtem vah chybových (deformačních) pravidel.

Př. 6.1. Konstrukce rozšířené gramatiky

$G = (V_N, V_T, P, S)$

$V_N = \{S, T, F\}$

$V_T = \{a, +, *, (,)\}$

P: $S \to S + T$ $\quad S \to T$

$T \to T * F$ $\quad T \to F$

$F \to (S)$ $\quad F \to a$

Provedeme rozšíření gramatiky:

- ***Krok 1:*** $V_N' = \{S, T, F, S', E_a, E_+, E_*, E_(, E_)\}$

  $V_T' = V_T$

- ***Krok 2:*** $S \to S\ E_+\ T$ $\quad S \to T$

  $T \to T\ E_*\ F$ $\quad T \to F$

  $F \to E_(\ S\ E_)$ $\quad F \to E_a$

- ***Krok 3:***

a) $S' \to S$

b) $S' \xrightarrow{1} S\ a$

$S' \xrightarrow{1,5} S\ +$

$S' \xrightarrow{2} S\ *$

$S' \xrightarrow{1} S\ (\ \ S' \xrightarrow{1} S\ )$

c) $E_a \to a$

$E_+ \to +$

$E_* \to *$

$E_(\to (\ \ E_) \to)$

d) $E_a \xrightarrow{1} +\ \ E_a \xrightarrow{1} *\ \ E_a \xrightarrow{2} (\ \ E_a \xrightarrow{2})$

$E_+ \xrightarrow{1} a\ \ E_+ \xrightarrow{1,5} *\ \ E_+ \xrightarrow{2} (\ \ E_+ \xrightarrow{2})$

$E_* \xrightarrow{1} a\ \ E_* \xrightarrow{1,5} +\ \ E_* \xrightarrow{2} (\ \ E_* \xrightarrow{2})$

$E_(\xrightarrow{2} a\ \ E_(\xrightarrow{2} +\ \ E_(\xrightarrow{1} *\ \ E_(\xrightarrow{0,5})$

$E_) \xrightarrow{2} a\ \ E_) \xrightarrow{2} +\ \ E_) \xrightarrow{1} *\ \ E_) \xrightarrow{0,5} (\ \ E_) \xrightarrow{1})$

e) $E_a \xrightarrow{0,5} \lambda\ \ E_* \xrightarrow{2} \lambda\ \ E_+ \xrightarrow{2} \lambda\ \ E_(\xrightarrow{1} \lambda\ \ E_) \xrightarrow{1} \lambda$

f) $E_a \xrightarrow{1} a\ E_a\ \ E_a \xrightarrow{2} +\ E_a\ \ E_a \xrightarrow{2} *\ E_a\ \ E_a \xrightarrow{2} (\ E_a\ \ E_a \xrightarrow{2})\ E_a$

$E_+ \xrightarrow{1} a\ E_+\ \ E_+ \xrightarrow{1} +\ E_+\ \ E_+ \xrightarrow{2} *\ E_+\ \ E_+ \xrightarrow{2} (\ E_+\ \ E_+ \xrightarrow{2})\ E_+$

$E_* \xrightarrow{1} a\ E_+\ \ E_+ \xrightarrow{2} +\ E_+\ \ E_+ \xrightarrow{1,5} *\ E_*\ \ E_+ \xrightarrow{2} (\ E_*\ \ E_* \xrightarrow{2})\ E_*$

$E_(\xrightarrow{2} a\ E(\ \ E_(\xrightarrow{1,5} +\ E_(\ \ E_(\xrightarrow{2} *\ E_(\ \ E_(\xrightarrow{1} (\ E_(\ \ E_(\xrightarrow{1} E($

$E_) \xrightarrow{2} a\ E_)\ \ E_) \xrightarrow{1,5} +\ E_)\ \ E_) \xrightarrow{2} *\ E_)\ \ E_) \xrightarrow{1} (\ E_)\ \ E_) \xrightarrow{1} E_)$

Zatímco G obsahuje 6 pravidel.

G' obsahuje 67 pravidel navíc.

G' je víceznačná gramatika, takže syntaktická analýza podle ní bude složitější. Většinou se pro syntaktickou analýzu s opravou chyb používá modifikovaný Earleyho algoritmus, který

oproti původnímu navíc akumuluje příslušné váhy pravidel, použité při derivaci deformovaného řetězce podle gramatiky G'.

Modifikovaný Earlyho algoritmus

Vstup:

- Rozšířená gramatika G'
- Vstupní řetězec $\omega = b_1 b_2 .. b_m$

Výstup:

- Seznamy $I_0, I_1, ..., I_m$ pro řetězec ω.
- Řetězec φ představující nejlepší opravu ω ve smyslu minima vzdálenosti.
- Vzdálenost d(ω, φ).

- ***Krok 1:*** Zkonstruujeme I_0.

  A. Pro každé pravidlo $S' \xrightarrow{x} \alpha \in P'$ přidáme do I_0 položku $[S' \rightarrow . \alpha, 0, x]$.

  B. Prováděj tak dlouho, dokud lze do I_0 přidávat položky. Pokud je v I_0 položka:

  $[A \rightarrow . B \beta, 0, y]$, přidej pro všechna pravidla $B \xrightarrow{z} \gamma$ položku: $[B \rightarrow . \gamma, 0, z]$ do I_0.

- ***Krok 2:*** Opakuj pro j = 1, 2,..., m

  A. Pro každou položku v I_{j-1} ve tvaru $[B \rightarrow \alpha . a \beta, i, x]$ takovou, že $a = b_j$, přidej do I_j položku $[B \rightarrow \alpha a . \beta, i, x]$

  Prováděj B a C tak dlouho, dokud lze do I_j přidat nějakou položku.

  B. Jestliže je položka $[A \rightarrow \alpha . , i, x]$ v I_j a položka $[B \rightarrow \beta . A \gamma, k, y]$ v I_i, pak:

  - Neexistuje-li, přidáme novou položku: $[B \rightarrow \beta A . \gamma, k, x + y]$.
  - Existuje-li již položka ve tvaru: $[B \rightarrow \beta A . \gamma, k, z]$ v I_j, pak pokud: $x + y < z$ nahradíme u této položky hodnotu *z* hodnotou $x + y$.

  C. Pro každou položku typu $[A \rightarrow \alpha . B \beta, i, x]$ v I_j přidáme pro všechna pravidla $B \xrightarrow{y} \gamma$ položky $[B \rightarrow . \gamma, j, y]$.

- ***Krok 3:***

  Pokud je položka $[S' \rightarrow \alpha . , 0, x]$ v I_m, pak řetězec ω je přijat s vahou:

  $$\underline{d(\omega, \varphi) = x}$$

Derivační strom řetězce ω lze získat podobným způsobem jako u původního Earlyho algoritmu. Řetězec φ (resp. jeho derivační strom) získáme vynecháním všech deformačních pravidel z derivace řetězce ω.

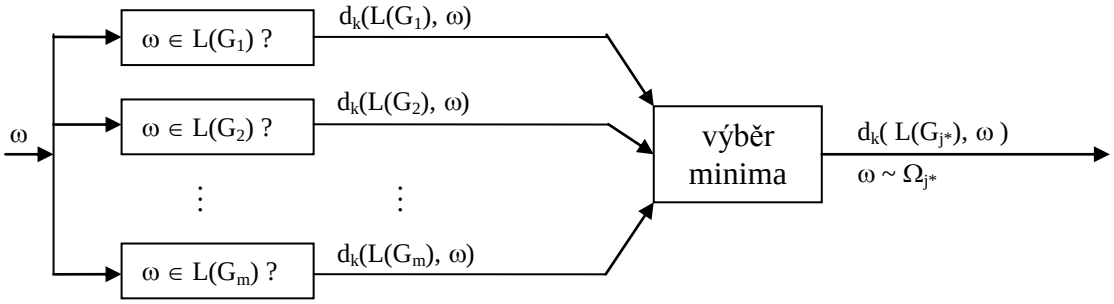
Definice: Vzdálenost mezi řetězcem a jazykem

ω ... řetězec L(G) ... daný jazyk

$$d(L(G), \omega) = \min_{\varphi \in L(G)} d(\varphi, \omega)$$

- Vzdálenost mezi řetězcem ω a jeho nejlepší opravou φ (metoda nejbližšího souseda – NN)
- Lze zobecnit na kNN: $d_k(L(G), \omega) = \min_{\varphi \in L(G)} \sum_{i=1}^{K} \frac{1}{K} d(\varphi_i, \omega)$

Blokové schéma klasifikátoru pro syntakticky deformované obrazy pracujícího na principu kNN.



6.3 Syntaktická analýza s opravou chyb pro stochastickou gramatiku

Použití zejména v případech, kdy se obrazy uvnitř tříd vyskytují s různou pravděpodobností.
Pro analyzovaný řetězec ω hledáme nejlepší opravu φ takovou, že platí:

$$q(\omega \mid \varphi)\, p(\varphi) = \max_{\Psi \in L(G_S)} \{ q(\omega \mid \psi) \cdot p(\psi) \}$$

p(ψ) ... pravděpodobnost generování řetězce ψ gramatikou G_S.

q(ω | ψ) ... pst deformace obrazu reprezentovaného řetězcem ψ na obraz reprezentovaný řetězcem ω.

Algoritmus konstrukce rozšířené stochastické gramatiky

Vstup: Stochastická bezkontextová gramatika $G_S = (V_N, V_T, P_S, S)$
Výstup: Rozšířená stochastická gramatika $G_S' = (V_N', V_T', P_S', S')$

- ***Krok 1:*** $V_N' = V_N \cup \{S'\} \cup \{E_a \mid a \in V_T\}$
- ***Krok 2:*** $V_T \subseteq V_T'$
- ***Krok 3:*** Jestliže v P_S je pravidlo $A \overset{p}{\rightarrow} \alpha_0\, b_1\, \alpha_1\, b_2 \ldots b_m\, \alpha_m$, $m \geq 0$ takové, že $\alpha_l \in V_N^*$ a $b_i \in V_T$, $l = 0, 1, \ldots, m$; $i = 1, 2, \ldots, m$; potom do P_S' přidám pravidlo:

  $$A \overset{p}{\rightarrow} \alpha_0\, E_{b1}\, \alpha_1\, E_{b2} \ldots E_{bm}\, \alpha_m$$

- ***Krok 4:*** Do P_S' přidáme následující pravidlo:
  - **A.** $S' \overset{1-q_I'}{\rightarrow} S$ $\qquad q_I' = \sum_{a \in V_T'} q_I'(a)$
  - **B.** $S' \overset{q_I'(a)}{\rightarrow} S\, a$ $\qquad \forall a \in V_T'$
- ***Krok 5:*** Pro všechna $a \in V_T$ přidáme do P_S' pravidla:
  - **A.** $E_a \overset{q_S(a \mid a)}{\rightarrow} a$
  - **B.** $E_a \overset{q_S(b \mid a)}{\rightarrow} b$ $\qquad$ pro $b \in V_T'$, $b \neq a$
  - **C.** $E_a \overset{q_D(a)}{\rightarrow} \lambda$
  - **D.** $E_a \overset{q_I(b \mid a)}{\rightarrow} b\, E_a$ $\qquad$ pro $b \in V_T'$

Stochastický jazyk generovaný rozšířenou gramatikou G_S':

$$L(G_S') = \left\{ (\omega, p(\omega)) \;\middle|\; \omega \in V_T^{*}{}',\ p(\omega) = \sum_{\varphi \in L(G_S)} \sum_{i=1}^{r} q^{(i)}(\omega \mid \varphi) \cdot p(\varphi) \right\}$$

r ... počet odlišných posloupností transformací řetězce φ na ω

$q^{(i)}(\omega \mid \varphi)$ ... pst spojená s i-tou posloupností, i = 1, 2,..., r

$p(\varphi)$ ... pst generování nedeformovaného řetězce φ nerozšířenou stochastickou gramatikou G_S

K nalezení nejlepší opravy deformovaného vstupního řetězce ve smyslu maxima psti lze využít modifikovaného Earleyho algoritmu.

Modifikovaný Earleyho algoritmus pro stochastický model

Vstup:

- Rozšířená stochastická gramatika $G_S' = (V_N', V_T', P_S', S')$
- Vstupní řetězec $\omega = b_1 b_2 ... b_m \in V_T'^*$

Výstup:

- Seznamy $I_0, I_1,..., I_m$ pro řetězec ω
- Hodnota deformační psti $q(\omega \mid \varphi)\, p(\varphi)$, kde φ představuje nejlepší opravu řetězce ω ve smyslu maxima psti.

- ***Krok 1:*** Konstrukce I_0

  A. Pro každé pravidlo $S' \xrightarrow{q} \alpha$ přidáme položku: $[S' \rightarrow . \alpha, 0, q]$ do I_0

  Provádíme tak dlouho, dokud lze přidat do I_0 nějakou položku:

  B. Pokud je v I_0 položka typu $[A \rightarrow . B \beta, 0, p]$, přidáme pro každé pravidlo $B \xrightarrow{q} \gamma$ položku $[B \rightarrow . \gamma, 0, q]$.

- ***Krok 2:*** Opakuj pro j = 1, 2,..., m

  A. Pro každou položku z I_{j-1} ve tvaru:
  $[B \rightarrow \alpha . a \beta, i, p]$ takovou, že $a = b_j$, přidáme položku $[B \rightarrow \alpha a . \beta, i, p]$ do I_j.
  Provádíme B. a C. tak dlouho dokud lze do I_j přidat nějakou položku.

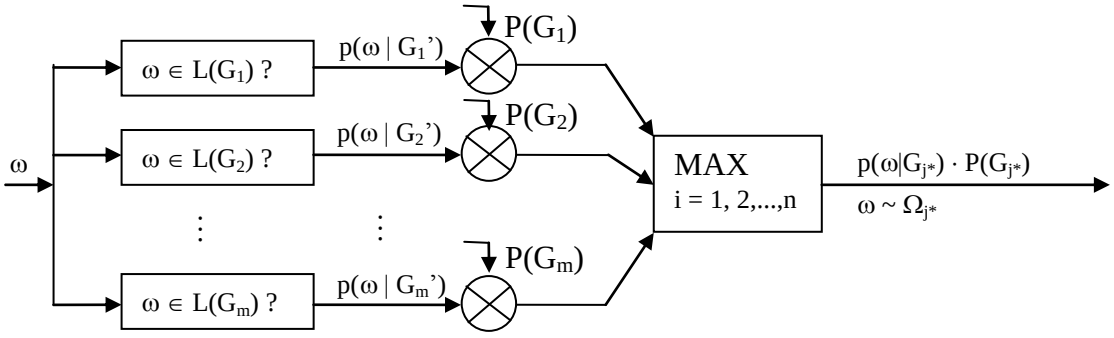
  B. Jestliže je položka $[A \rightarrow \alpha . , i, p]$ v I_j a položka $[B \rightarrow \beta . A \gamma, k, q]$ v I_i, pak:
  - Pokud dosud neexistuje přidáme novou položku: $[B \rightarrow \beta A . \gamma, k, p\ q]$
  - Pokud již existuje položka $[B \rightarrow \beta A . \gamma, k, r]$, pak pokud $p \cdot q > r$, nahradíme r hodnotou $p \cdot q$.

  C. Pro každou položku typu $[A \rightarrow \alpha . B \beta, i, p]$ v I_j přidáme pro všechna pravidla $B \xrightarrow{q} \gamma$ položky $[B \rightarrow . \gamma, j, q]$ do I_j.

- ***Krok 3:***

  Pokud je položka $[S' \rightarrow \alpha . , 0, q]$ v I_m, pak řetězec je přijat s pstí q. V případě, že G_S je jednoznačná, pak q odpovídá psti $q(\omega \mid \varphi) \cdot p(\varphi)$, jinak odpovídá psti $q(\omega \mid \varphi) \cdot p_j(\varphi)$, kde $p_j(\varphi)$ je pst nejvíce pravděpodobné derivace řetězec φ.

Blokové schéma Bayesovského syntaktického analyzátoru



V případech, kdy je k dispozici rozsáhlá a dostatečně reprezentativní trénovací množina, zahrnující i zašuměné a deformované obrazy, potom stochastická gramatika zkonstruovaná na jejím základě může dostatečně spolehlivě reprezentovat všechny v úvahu připadající struktury obrazů a jim odpovídající četnosti výskytu. V tomto případě obvykle postačí zkonstruovat stochastický syntaktický analyzátor bez opravy chyb.

Naopak, je-li trénovací množina málo reprezentativní, potom zkonstruovaná gramatika necharakterizuje možné (i deformované) struktury obrazů dostatečně. Důsledkem je, že syntaktický analyzátor podle takové gramatiky bez opravy chyb velmi často deformované obrazy této třídy nepřijímá. V takových případech je obvykle nezbytné použít syntaktický analyzátor s opravou chyb.

Syntaktické analyzátory s opravou chyb jsou sice schopny rozpoznat i značně zašuměné a deformované obrazy, ale vyžadují výrazně delší dobu zpracování na rozdíl od syntaktických analyzátorů bez opravy chyb. Proto se často využívá paralelních metod zpracování nebo speciálních algoritmů stochastické syntaktické analýzy.

7 SHLUKOVÁ ANALÝZA PRO SYNTAKTICKÉ OBRAZY

Díky zavedení vzdálenosti mezi syntaktickými obrazy je možné upravit většinu nejznámějších metod shlukové analýzy pro využití ve strukturálním přístupu k rozpoznávání.

Předpokládáme třídy Ω_i, i = 1, 2,..., m, reprezentované množinami řetězců:

$\Omega_i = \{\omega_1^i, \omega_2^i, ..., \omega_n^i\}$. Neznámý obraz reprezentovaný řetězcem φ zařadíme do třídy Ω_j, jestliže:

$$\min_{i=1,2,...n_j*} d\left(\omega_i^{j*}, \varphi\right) = \min_{j=1,2,...m} \left\{ \min_{i=1,2,...n_j} d\left(\omega_i^j, \varphi\right) \right\}$$

Vzdálenost mezi řetězci lze určit např. na základě Levenshteinovy vzdálenosti, kterou je možné vypočítat podle algoritmu dynamického programování.

Algoritmus výpočtu Levenshteinovy vzdálenosti

Vstup: Dva řetězce $\omega = a_1\ a_2\ ...\ a_n$; $\varphi = b_1\ b_2\ ...\ b_m$

Výstup: $d(\omega, \varphi)$

- ***Krok 1:*** $D(0,0) = 0$
- ***Krok 2:*** pro i = 1, 2,..., n
  $D(i, 0) = D((i - 1), 0) + 1$
  pro j = 1, 2,..., m
  $D(0, j) = D(0, (j - 1)) + 1$
- ***Krok 3:*** pro i = 1, 2,..., n; j = 1, 2,..., m
  jestliže $a_i = b_j$, pak:
  $e_1 = D(i - 1, j - 1)$
  jinak:
  $e_1 = D(i - 1, j - 1) + 1$
  $e_2 = D(i - 1, j) + 1$
  $e_3 = D(i, j - 1) + 1$
  $D(i, j) = \min(e_1, e_2, e_3)$
- ***Krok 4:*** $d(\omega, \varphi) = D(n, m)$

Tento algoritmus lze snadno upravit pro výpočet váhové Levenshteinovy vzdálenosti nebo váhové metriky.

Klasifikaci na základě nejbližšího souseda (NN) lze zobecnit na kNN:

označíme: $\widetilde{\Omega}_i = \{\widetilde{\omega}_1^i, \widetilde{\omega}_2^i, \ldots, \widetilde{\omega}_{n_i}^i\}$

Množinu Ω_i, i = 1, 2,..., m přeuspořádáme tak, že platí:

pokud $k < l$ pak $d(\widetilde{\omega}_k^i, \varphi) \le d(\widetilde{\omega}_l^i, \varphi)$

Neznámý řetězec φ pak zařadíme do třídy Ω_{i*} pokud:

$$\sum_{j=1}^{K} \frac{1}{K} \cdot d(\widetilde{\omega}_j^{i*}, \varphi) = \min_{i=1,2,\ldots m} \left\{ \sum_{j=1}^{K} \frac{1}{K} \cdot d(\widetilde{\omega}_j^{i}, \varphi) \right\}$$

Jednoduchý shlukovací algoritmus

Vstup:

- Množina vzorků $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$
- Prahová konstanta *t*

Výstup:

- Rozdělení Ω do *m* shluků $\Omega_1, \Omega_2, ..., \Omega_m$, $m \le n$, *m* předem neznáme, závisí na *t*.

- ***Krok 1:*** Zařadíme ω_1 do Ω_1.
  j = 1 (počet zařazených obrazů)
  m = 1 (počet shluků)
  $n_1 = 1$
- ***Krok 2:*** pro j = 2, 3,..., n
  $$D_{i^*} = \min_{i=1,2,..m} \left\{ \min_{l=1,2,...,n_i} d\left(\omega_l^i, \omega_j\right) \right\}$$
  jestliže:
  $D_i^* \le t$ zařadíme ω_j do Ω_{i^*}; $\omega_{n_{i^*}+1}^{i^*} = \omega_j$; $n_{i^*}++$
  $D_i^* > t$ vytvoříme novou třídu Ω_{m+1}, $\omega_1^{m+1} = \omega_j$; $m++$; $n_m=1$

Zavedeme tzv. **β-metriku** pro řetězec ω_j^i ve shluku Ω_i takto:

$$\beta_j^i = \frac{1}{n_i} \cdot \sum_{l=1}^{n_i} d\left(\omega_j^i, \omega_l^i\right)$$

Středem shluku Ω_i je pak takový řetězec $\omega_{j^*}^i$, pro který platí:

$$\beta_{j^*}^i = \min_{j=1,2,...,n_i} \left\{\beta_j^i\right\}$$

Řetězec $\omega_{j^*}^i$ nazýváme **reprezentantem shluku** Ω_i a budeme ho značit $\mathbf{A_i}$.

Algoritmus centroidní metody shlukování

Vstup: Množina vzorků $\Omega = \{\omega_1, \omega_2,..., \omega_n\}$.
Výstup: Rozdělení Ω do *m* shluků $\Omega_1, \Omega_2,..., \Omega_m$.

- ***Krok 1:*** Zvolíme *m* libovolných prvků z Ω jako reprezentanty *m* shluků a označíme je $A_1, A_2,..., A_m$.
- ***Krok 2:*** Pro všechna *i*, $\omega_i \in \Omega$, zařadíme ω_i do shluku Ω_j^*, jestliže:
  $$d\left(A_{j^*}, \omega_i\right) = \min_{j=1,2,...,m} \left\{d\left(A_j, \omega_i\right)\right\}$$
- ***Krok 3:*** Pro všechny shluky Ω_j, j = 1, 2,..., m vypočteme nové reprezentanty shluků A_j
- ***Krok 4:*** Jestliže nedošlo ke změně žádného reprezentanta A_j, j = 1, 2,..., m,
  pak: **konec** algoritmu,
  jinak: pokračujeme **bodem 2**.

Tyto algoritmy vedou pouze k rozdělení množiny řetězců na shluky reprezentované zase množinami řetězců, což umožňuje rozpoznávání pouze pomocí metody *template matching*. Abychom mohli při rozpoznávání získat navíc strukturální popis, je třeba jednotlivé shluky reprezentovat gramatikami.

Algoritmus

Vstup:

- Množina syntaktických obrazů $\Omega = \{\omega_1, \omega_2,..., \omega_n\}$.
- Neznámý obraz reprezentovaný řetězcem φ.

Výstup:

- ❖ Klasifikace neznámého obrazu
- ❖ Úplný strukturální popis derivačním stromem

- ➢ ***Krok 1:*** Vhodným shlukovacím algoritmem rozdělíme Ω na *m* shluků Ω_i, i = 1,..., m

  $\Omega_i = \{ \omega_1^i, \omega_2^i, \ldots \omega_{n_i}^{\ i} \}$

- ➢ ***Krok 2:*** Odvodíme *m* gramatik $G_1, G_2, ..., G_m$ charakterizující shluky $\Omega_1, \Omega_2, ..., \Omega_m$.
- ➢ ***Krok 3:*** Zkonstruujeme rozšířené gramatiky G_1‘, G_2‘,..., G_m‘ a odpovídající syntaktické analyzátory s opravou chyb.
- ➢ ***Krok 4:*** Vypočteme všechny vzdálenosti $d(L(G_i), \varphi)$ pro *i = 1, 2,..., m* a určíme takové *i**, pro které platí:

  $$d\left(L(G_{i*}'), \varphi\right) = \min_{i=1,2,\ldots,m} \left\{d\left(L(G_i'), \varphi\right)\right\}$$

- ➢ ***Krok 5:*** Řetězec φ zařadíme do třídy Ω_{i*}.

Výsledkem není jen nalezení minimální vzdálenosti mezi řetězcem φ a jazykem $L(G_{i*})$ a tedy klasifikace do třídy Ω_i*, ale současně i nalezení nejlepší opravy řetězce φ v jazyce $L(G_{i*})$ a úplný strukturální popis klasifikovaného obrazu.

Algoritmus pro shlukování postupně získávaných syntaktických obrazů

Vstup:

- ❖ Postupně získáváme reprezentace neznámých obrazů ve formě řetězců primitiv $\omega_1\omega_{2...}\ \omega_n$.
- ❖ Zvolený parametr: práh *t*.

Výstup:

- ❖ Klasifikace řetězců ω_i; i = 1,2,..., n do *m* shluků, včetně získání jejich úplných strukturálních popisů a gramatik $G^{(k)}$, k = 1,2,...,m charakterizujících jednotlivé shluky.

- ➢ ***Krok 1:*** Na základě ω_1 zkonstruujeme gramatiku $G_1^{(1)}$ tak, aby $\{\omega_1\} \subseteq L(G_1^{(1)})$.

- ➢ ***Krok 2:*** Pro $G_1^{(1)}$ zkonstruujeme rozšířenou gramatiku a syntaktický analyzátor s opravou chyb $E_1^{(1)}$.
- ➢ ***Krok 3:*** Pro i = 2, 3,..., n najdeme minimální vzdálenost:

  $$D = d\left(L(G_{n_{k*}}^{k*}), \omega_i\right) = \min_{k=1,2,\ldots,j} \left\{d\left(L(G_{n_k}^{(k)}), \omega_i\right)\right\}$$

  (a) Je-li $D \le t$, zařadíme ω_i do shluku Ω_k* a zkonstruujeme gramatiku $G_{n_k*+1}^{(k*)}$:

  - ❖ Zkonstruujeme rozšířenou gramatiku a syntaktický analyzátor s opravou chyb.
  - ❖ Položíme n_{k*}++.

  (b) Je-li $D > t$, založíme nový shluk Ω_{j+1}:

  - ❖ Zkonstruujeme gramatiku $G_1^{(j+1)}$ na základě ω_i
  - ❖ Zkonstruujeme rozšířenou gramatiku a syntaktický analyzátor s opravou chyb.
  - ❖ Položíme: j++; $n_j = 1$

Výsledkem shlukovacího procesu je *m* shluků, charakterizovaných gramatikami $G_{n_1}^{(1)}, G_{n_2}^{(2)}, ..., G_{n_m}^{(m)}$. Syntaktickou analýzou bez opravy chyb lze získat strukturální popisy jednotlivých obrazů reprezentovaných řetězci $\omega_1, \omega_2, ..., \omega_n$.

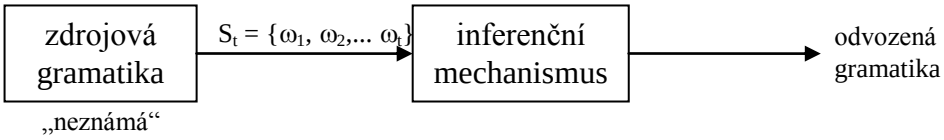
Problém volby prahu *t* zatím zůstává nedořešenou otázkou. Jeho hodnotu lze někdy určit s pomocí trénovací množiny obrazů se známou klasifikací. V případě *m* tříd charakterizovaných gramatikami $G^{(1)}, G^{(2)}, ..., G^{(m)}$ by *t* mělo být zvoleno tak, aby:

$$t < \min_{k,l=1,2,...,m} \left\{ d\left(L(G^{(l)}), \omega^{(k)}\right) \right\} \qquad k \neq l$$

kde $\omega^{(k)}$ značí řetězec, který patří do třídy Ω_k. Pokud tato informace není dostupná, musíme práh *t* zvolit experimentálně. Vede-li např. zvolená hodnota prahu *t* ke vzniku většího počtu tříd, než požadujeme, je třeba *t* zvětšit a naopak.

8 INFERENCE GRAMATIK

Gramatická inference: Úloha odvození gramatiky na základě množiny vzorků (řetězců).



Cílem gramatické inference je odvodit syntaktická pravidla neznámé gramatiky na základě konečné množiny vzorků S_t s udanou informací, zda patří, či nepatří do jazyka generovaného gramatikou. Trénovací množinu tedy můžeme rozdělit na množinu prvků, které do L(G) patří:

$$S^+ = \{\omega_1^+, \omega_2^+, ..., \omega_{t+}^+\} \qquad \omega_i^+ \in L(G);\ i = 1, 2, ..., t^+$$

a množinu prvků, které do L(G) nepatří:

$$S^- = \{\omega_1^- \omega_2^- ..., \omega_{t-}^-\} \qquad \omega_i^- \notin L(G);\ i = 1, 2, ..., t^- \qquad t^- + t^+ = t$$

Množina $S^+ = \{\omega_1^+, \omega_2^+, ..., \omega_{t+}^+\}$ je **strukturálně úplná**, jestliže každé pravidlo z neznámé zdrojové gramatiky je použito při generování alespoň jednoho řetězce z S^+ (tzn. v S^+ jsou použita všechna pravidla).

Odvozená gramatika G_t je **kompatibilní** s množinou S_t, jestliže:

$$\omega \in L(G_t) \ \ \forall \omega \in S^+ \wedge \varphi \notin L(G_t) \ \ \forall \varphi \in S^-$$

Neexistuje obecná metoda, která by umožňovala na základě zadané trénovací množiny automaticky odvodit gramatiku, která by byla vhodnou aproximací gramatiky hledané. Řešení úlohy gramatické inference je navíc obvykle nejednoznačné.

8.1 Inference regulárních gramatik

Budeme vycházet z těchto předpokladů:

- Odvozená gramatika je regulární
- Množina S_t je konečná
- Množina S^+ je strukturálně úplná
- Pro odvozenou gramatiku by mělo platit: $S^+ \subseteq L(G)$, $S^- \subseteq \overline{L(G)}$

Algoritmus inference kanonické regulární gramatiky

Vstup: $S^+ = \{\omega_1, \omega_2, ..., \omega_t\}$

Výstup: Regulární gramatika $G_c = (V_{NC}, V_{TC}, P_C, S)$ v kanonické formě.

- ***Krok 1:*** Najdeme všechny různé terminální symboly ze všech řetězců $\omega \in S^+$ a jejich množinu označíme V_{TC}.
- ***Krok 2:*** Pro každý řetězec $\omega_i = a_{i1}, a_{i2},..., a_{in}$, $\omega_i \in S^+$, přidáme do P_C pravidla:
  $S \rightarrow a_{i1}\ Z_{i1}$ $\quad Z_{i1} \rightarrow a_{i2}\ Z_{i2}$ .... $Z_{i\ n-2} \rightarrow a_{i\ n-1}\ Z_{i\ n-1}$ $\quad Z_{i\ n-1} \rightarrow a_{i\ n}$
  kde každé Z_{ij} ; j = 1, 2,..., n-1 představuje nový **neterminální symbol** $Z_{ij} \in V_{NC}$

Př. 8.1. $G = (V_N, V_T, P, S)$

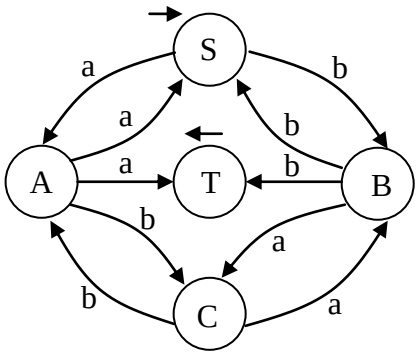
$V_N = \{S, A, B, C\}$

$V_T = \{a, b\}$

P: $S \rightarrow a\ A$ $\quad A \rightarrow a$ $\quad B \rightarrow b$ $\quad C \rightarrow a\ B$

$S \rightarrow b\ B$ $\quad A \rightarrow a\ S$ $\quad B \rightarrow b\ S$ $\quad C \rightarrow b\ A$

$A \rightarrow b\ C$ $\quad B \rightarrow a\ C$



$S^+ = \{abab, bbaa, baba, aabb, aa\}$

Získaná kanonická regulární gramatika vypadá takto:

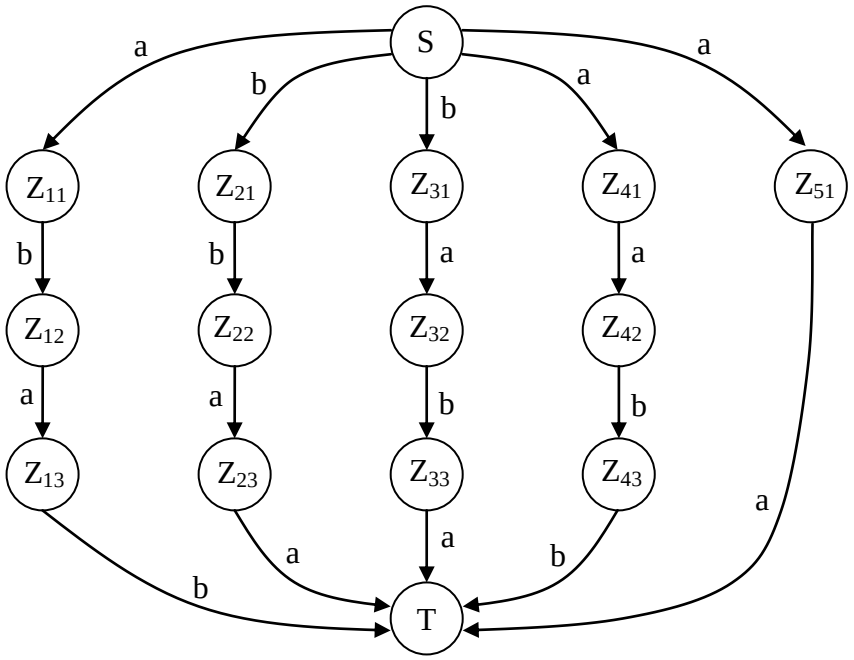
$G_c = (V_{NC}, V_{TC}, P_C, S)$

$V_{TC} = \{a, b\}$

$V_{NC} = \{S, Z_{11}, Z_{12}, Z_{13}, Z_{21}, Z_{22}, Z_{23}, Z_{31}, Z_{32}, Z_{33}, Z_{41}, Z_{42}, Z_{43}, Z_{51}\}$

P_C:

$S \rightarrow a\ Z_{11}$	$Z_{11} \rightarrow b\ Z_{12}$	$Z_{12} \rightarrow a\ Z_{13}$	$Z_{13} \rightarrow b$
$S \rightarrow b\ Z_{21}$	$Z_{21} \rightarrow b\ Z_{22}$	$Z_{22} \rightarrow a\ Z_{23}$	$Z_{23} \rightarrow a$
$S \rightarrow b\ Z_{31}$	$Z_{31} \rightarrow a\ Z_{32}$	$Z_{32} \rightarrow b\ Z_{33}$	$Z_{33} \rightarrow a$
$S \rightarrow a\ Z_{41}$	$Z_{41} \rightarrow a\ Z_{42}$	$Z_{42} \rightarrow b\ Z_{43}$	$Z_{43} \rightarrow b$
$S \rightarrow a\ Z_{51}$	$Z_{51} \rightarrow a$		



G_C popisuje jiný jazyk $L(G_C) \subseteq L(G)$. Snadné odvození kanonické gramatiky je vyváženo příliš velkým počtem neterminálních symbolů a pravidel. **Některé neterminální symboly jsou totiž ekvivalentní. Problém je v tom, že nevíme které.**

Naším cílem je nalézt skupiny takových vzájemně ekvivalentních symbolů, protože takové skupiny odpovídají neterminálním symbolům neznámé zdrojové gramatiky a můžeme je potom nahradit jednotlivými neterminálními symboly a výrazně tak redukovat přílišný počet neterminálních symbolů a pravidel gramatiky.

Jeden z možných rozkladů V_{NC}:

$V_{ND} = \{(S, Z_{22}, Z_{42}), (Z_{11}, Z_{23}, Z_{33}, Z_{41}, Z_{51}), (Z_{13}, Z_{21}, Z_{31}, Z_{43}), (Z_{12}, Z_{22})\ \} = \{B_0, B_1, B_2, B_3\}$

Jinými slovy: Naším cílem je nalézt takovou relaci ekvivalence na množině V_{NC}, která ji rozdělí do r+1 disjunktních podmnožin B_j tak, aby platilo:

$$B_j \cap B_k = 0 \quad j \neq k \qquad \wedge \qquad \bigcup_{j=0}^{r} B_j = V_{NC} \quad j, k = 0, 1, ..., r \qquad S \in B_0$$

Definice: Odvozená gramatika

$G_D = (V_{ND}, V_{TD}, P_D, B_O)$ je gramatika získaná z G_C těmito kroky.

- $V_{TD} = V_{TC}$
- V_{ND} odpovídá disjunktním podmnožinám vzniklým z V_{NC}
- B_O odpovídá té podmnožině vzniklé rozkladem V_{NC}, která obsahuje S
- P_D je určená takto:
  - jestliže $Z_\alpha, Z_\beta \in V_{NC}$, $a \in V_{TD}$

    $Z_\alpha \rightarrow a\ Z_\beta \quad \in P_C$

    $Z_\alpha \in B_i \quad a \quad Z_\beta \in B_j$

    potom do P_D přidáme pravidlo $B_i \rightarrow a\ B_j$
  - jestliže $Z_\alpha \in V_{NC}$, $a \in V_{TD}$

    $Z_\alpha \rightarrow a \quad \in P_C$

    $Z_\alpha \quad \in B_i$

    potom do P_D přidáme pravidlo $B_i \rightarrow a$

Kdybychom zvolili výše uvedený rozklad V_{NC} z př. 8.1., pak by P_D vypadalo takto:

$B_0 \rightarrow a\ B_1$	$B_1 \rightarrow a$	$B_2 \rightarrow b$	$B_3 \rightarrow a\ B_2$
$B_0 \rightarrow b\ B_2$	$B_1 \rightarrow a\ B_0$	$B_2 \rightarrow b\ B_0$	$B_3 \rightarrow b\ B_1$
	$B_1 \rightarrow b\ B_3$	$B_2 \rightarrow a\ B_3$	

V tomto případě je G_D ekvivalentní s původní gramatikou G.

Věta: Mějme:

- Regulární zdrojovou gramatiku G.
- $S^+ \subseteq L(G)$ je strukturálně úplná.
- G_C je kanonická regulární gramatika získaná z S^+.

Existuje odvozená gramatika G_D vzniklá takovým rozkladem V_{NC}, že G_D je ekvivalentní s G.

Věta: Mějme G_D odvozenou gramatiku z G_C vzniklé na základě S^+. Potom platí: $S^+ \subseteq L(G_D)$

Otázkou zůstává, jak zvolit rozklad V_{NC}.

Jestliže $| V_{NC} | = k+1$, pak existuje E_{k+1} různých rozkladů V_{NC}, kde:

$$E_0 = 1, \quad E_{k+1} = \sum_{j=0}^{k} \binom{k}{j} E_j$$

Minimálně jedna z E_{k+1} možných G_D splňuje podmínku $S^+ \subseteq L(G_D)$, $S^- \subseteq \overline{L(G_D)}$, neboť jedním z rozkladů je i G_C.

Obvykle však existuje více odvozených gramatik, které tuto podmínku splňují. Vyčerpávající prohledávání všech možností je prakticky nemožné, neboť E_{k+1} rychle narůstá s rostoucím k. Proto se využívá speciálních postupů, které se zaměřují na vyhledávání pouze některých rozkladů.

Jednou z možností je odvození kanonické regulární gramatiky formálních derivací G_{CD}.

Formální derivace množiny řetězců A pro symbol *a*, $a \in V_T$ je definováno takto:

$D_\lambda A = A$

$D_a A = \{\omega \mid a\,\omega \in A, \omega \in V_T^*\}$

Algoritmus inference kanonické gramatiky formálních derivací

Vstup: Množina řetězců S^+

Výstup: Gramatika G_{CD}

- ***Krok 1:*** Vytvoříme množinu $U = \{U_1, U_2, ..., U_T\}$ různých formálních derivací z S^+, které nejsou rovné λ či Ø.

  Přitom $U_1 = D_\lambda S^+ = S^+$
- ***Krok 2:*** Startovací symbol $S = U_1$
- ***Krok 3:*** Určíme množinu terminálních symbolů $V_{T\,CD}$ z řetězců trénovací množiny S^+
- ***Krok 4:*** $V_N = U$
- ***Krok 5:*** Do množiny pravidel pro všechna $a \in V_T$; $U_i, U_j \in V_N$ pravidla:

  $U_i \rightarrow a\,U_j$, pokud $D_a U_i = U_j$

  $U_i \rightarrow a$, pokud $D_a U_i = \{\lambda\}$

Př. 8.2.

$S^+ = \{abab, baba, aaabba, bbabab, aaabab, bbbaba\}$

Kanonická gramatika by obsahovala 27 neterminálních symbolů. Zkonstruujeme kanonickou gramatiku formálních derivací G_{CD}.

- ***Krok 1:***

$D_\lambda S^+ = S^+ = U_1$
$D_a U_1 = \{bab, aabba, aabab\} = U_2$
$D_b U_1 = \{aba, babab, bbaba\} = U_3$
$D_a U_2 = \{abba, abab\} = U_4$
$D_b U_2 = \{ab\} = U_5$
$D_a U_3 = \{ba\} = U_6$
$D_b U_3 = \{abab, baba\} = U_7$
$D_a U_4 = \{bba, bab\} = U_8$
$D_b U_4 = Ø$
$D_a U_5 = \{b\} = U_9$
$D_b U_5 = Ø$
$D_a U_6 = Ø$
$D_b U_6 = \{a\} = U_{10}$
$D_a U_7 = \{bab\} = U_{11}$
$D_b U_7 = \{aba\} = U_{12}$
$D_a U_8 = Ø$
$D_b U_8 = \{ba, ab\} = U_{13}$
$D_a U_9 = Ø$
$D_b U_9 = \{\lambda\}$
$D_a U_{10} = \{\lambda\}$
$D_b U_{10} = Ø$
$D_a U_{11} = Ø$
$D_b U_{11} = \{ab\} = U_5$
$D_a U_{12} = \{ba\} = U_6$
$D_b U_{12} = Ø$
$D_a U_{13} = \{b\} = U_9$
$D_b U_{13} = \{a\} = U_{10}$

Ø...nelze derivovat
{λ}...došli jsme na konec řetězce

- ***Krok 2:*** $S = U_1$
- ***Krok 3:*** $V_T = \{a, b\}$
- ***Krok 4:*** $V_N = \{U_1, U_2, ..., U_{13}\}$
- ***Krok 5:***

$U_1 \rightarrow a\ U_2$	$U_4 \rightarrow a\ U_8$	$U_9 \rightarrow b$
$U_1 \rightarrow b\ U_3$	$U_5 \rightarrow a\ U_9$	$U_{10} \rightarrow a$
$U_2 \rightarrow a\ U_4$	$U_6 \rightarrow b\ U_{10}$	$U_{11} \rightarrow b\ U_5$
$U_2 \rightarrow b\ U_5$	$U_7 \rightarrow a\ U_{11}$	$U_{12} \rightarrow a\ U_6$
$U_3 \rightarrow a\ U_6$	$U_7 \rightarrow b\ U_{12}$	$U_{13} \rightarrow a\ U_9$
$U_3 \rightarrow b\ U_7$	$U_8 \rightarrow b\ U_{13}$	$U_{13} \rightarrow b\ U_{10}$

Kanonická regulární gramatika formálních derivací obsahuje pouze 13 neterminálních symbolů a generuje právě množinu S^+.

Obrazy strukturálně velmi podobné obrazům S^+ jsou syntaktickou analýzou podle kanonické gramatiky odmítány. Protože syntaktická analýza s opravou chyb je časově velmi náročná, je žádoucí, aby vytvořená gramatika do jisté míry „zobecňovala“ strukturu obrazů z S^+.

Toho je možné dosáhnout např. pomocí **gramatické inference založené na „k-koncích“ řetězců** z kanonické gramatiky formálních derivací.

Nechť $\omega = a_1\ a_2\ ...\ a_n$ je řetězec $a_i \in V_T$, i = 1,..., n. **Množina „k-konců“ řetězců množiny S^+** vzhledem k řetězci ω je určena vztahem:

$$g(\omega, S^+, k) = \{\varphi \mid \varphi \in D_\omega\ S^+, \mid \varphi \mid \leq k\}$$

Uvažujme dvě různé množiny U_i a U_j z gramatiky formálních derivací G_{CD}, pro které platí:

$$U_i = D_{\varphi_i} S^+ \qquad U_j = D_{\varphi_j} S^+$$

kde $\varphi_i, \varphi_j \in V_T^*$. O množinách U_i a U_j řekneme, že jsou **„k-koncově“ ekvivalentní**, platí-li:

$$g(\varphi_i, S^+, k) = g(\varphi_j, S^+, k)$$

S pomocí takto zavedené relace ekvivalence je možné z kanonické gramatiky formálních derivací získat tzv. **odvozenou gramatiku G_{Dk}.** K dispozici však není žádná vhodná metoda pro volbu k.

Př. 8.3. Pokračujeme v příkladě 8.2.

Různé množiny „k-konců“ řetězců S^+:

	k = 5	**k = 3**	**k = 2**	**k = 1**
U_1	{abab, baba}	0	0	0
U_2	{bab, aabba, aabab}	{bab}	0	0
U_3	{aba, babab, bbaba }	{aba}	0	0
U_4	{abab, abba}	0	0	0
U_5	{ab}	{ab}	{ab}	0
U_6	{ba}	{ba}	{ba}	0
U_7	{abab, baba}	0	0	0
U_8	{bba, bab}	{bba, bab}	0	0
U_9	{b}	{b}	{b}	{b}
U_{10}	{a}	{a}	{a}	{a}
U_{11}	{bab}	{bab}	0	0
U_{12}	{aba}	{aba}	0	0
U_{13}	{ba, ab}	{ba, ab}	{ba, ab}	0

Získáme následující třídy ekvivalence:

		počet tříd:
k = 5:	$\{U_1, U_7\}, \{U_2\}, \{U_3\},..., \{U_6\}, \{U_8\},..., \{U_{13}\}$	12
k = 3:	$\{U_2, U_{11}\}, \{U_3, U_{12}\}, \{U_1, U_4, U_7\}, \{U_5\}, \{U_6\}, \{U_8\}, \{U_9\}, \{U_{10}\}, \{U_{13}\}$	9
k = 2:	$\{U_1, U_2, U_3, U_4, U_7, U_8, U_{11}, U_{12}\}, \{U_5\}, \{U_6\}, \{U_9\}, \{U_{10}\}, \{U_{13}\}$	6
k = 1:	$\{ U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_{11}, U_{12}, U_{13}\}, \{U_9\}, \{U_{10}\}$	3

k=3:

$U_1 \to a\ U_2$	$U_1 \to a\ U_8$	$U_9 \to b$
$U_1 \to b\ U_3$	$U_1 \to b\ U_9$	$U_{10} \to a$
$U_2 \to a\ U_1$	$U_6 \to b\ U_{10}$	
$U_2 \to b\ U_5$		
$U_3 \to a\ U_6$		$U_{13} \to a\ U_9$
$U_3 \to b\ U_1$	$U_8 \to b\ U_{13}$	$U_{13} \to b\ U_{10}$

k=2:

$U_1 \to a\ U_1$		$U_9 \to b$
$U_1 \to b\ U_1$	$U_1 \to b\ U_9$	$U_{10} \to a$
	$U_6 \to b\ U_{10}$	
$U_1 \to b\ U_5$		
$U_1 \to a\ U_6$		$U_{13} \to a\ U_9$
	$U_1 \to b\ U_{13}$	$U_{13} \to b\ U_{10}$