

Předmět KIV/TI - přednáška 2

Úvod do teorie jazyků

Ing. Václav Vais, Ph.D.

vais@kiv.zcu.cz

Základní pojmy z teorie jazyků

- **Abeceda** = konečná neprázdná množina. Prvky abecedy se nazývají **písmena**.
- **Řetězec (slovo)** = libovolná „konečná posloupnost“ (tj. uspořádaná n-tice) písmen abecedy.

Př.: Abeceda $\Sigma = \{a, b, c\}$

Příklady řetězců: ab , $aaabaaab$, aba , ϵ [= prázdný řetězec, empty]

Jiný hojně používaný symbol pro prázdný řetězec : λ

- **Uzávěr abecedy** Σ^+ = množina všech neprázdných řetězců vytvořených z písmen abecedy Σ .

Značení uzávěru abecedy: Σ^+

Uzávěr abecedy je nekonečná množina (délka řetězce není omezená).

Uzávěr abecedy je spočetná množina.

Základní pojmy z teorie jazyků

- Iterace abecedy Σ = množina všech řetězců vytvořených z písmen abecedy Σ .

Značení iterace abecedy: Σ^*

Zřejmé: $\Sigma^* = \Sigma^+ \cup \{e\}$

Př.: Abeceda $A = \{0, 1\}$ počet prvků abecedy $n = 2$

$A^+ = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \dots, 1111, \dots\}$

$2 = n$

$4 = n^2$

$8 = n^3$

$16 = n^4 \dots\dots$

$A^* = \{e, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \dots, 1111, \dots\}$

Operace nad řetězci

u a v jsou řetězce nad abecedou Σ , obecně různé délky, tedy

$$u = a_1 a_2 \dots a_n \qquad v = b_1 b_2 \dots b_m \qquad \forall i \quad a_i, b_i \in \Sigma$$

- **Zřetězení řetězců:** $\Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

$$u \cdot v = a_1 a_2 \dots a_n b_1 b_2 \dots b_m \quad [\text{zřetězení řetězců } u \text{ a } v]$$

Operace zřetězení není komutativní, tedy obecně neplatí $u \cdot v = v \cdot u$.

Řetězení s prázdným řetězcem: $e \cdot u = u \cdot e = u \quad \forall u \in \Sigma^*$

Operace nad řetězci

u je řetězec nad abecedou Σ , tedy

$$u = a_1 a_2 \dots a_n \quad \forall i \quad a_i \in \Sigma$$

- **Mocnina řetězce:** $\Sigma^* \times \mathbb{N}_0 \rightarrow \Sigma^*$

$$u^0 = e$$

$$u^1 = u$$

$$u^2 = u \cdot u$$

.....

$$u^r = u^{r-1} \cdot u = u \cdot u^{r-1}$$

Operace nad řetězcí

u a v jsou řetězce nad abecedou Σ , obecně různé délky, tedy

$$u = a_1 a_2 \dots a_n \qquad v = b_1 b_2 \dots b_m \qquad \forall i \quad a_i, b_i \in \Sigma$$

- **Reverze (obrácení) řetězce:** $\Sigma^* \rightarrow \Sigma^*$

$$u^R = a_n a_{n-1} \dots a_2 a_1$$

- **Délka řetězce:** $\Sigma^* \rightarrow \mathbb{N}_0$

$$|u| = n$$

$$|u \cdot v| = n + m$$

$$|e| = 0$$

$$|u^r| = r \cdot n$$

Nejobecnější chápání pojmu jazyk

Σ je abeceda, tedy konečná neprázdná množina nějakých symbolů (písmen)

- **Jazyk L nad abecedou Σ** = libovolná množina řetězců nad abecedou Σ , tedy $L \subseteq \Sigma^*$.

Toto je nejjednodušší přístup definice formálního jazyka, na jazyk pohlíží jako na množinu řetězců nad výchozí abecedou (a „neklade na řetězce z množiny žádné podmínky“).

Obecně: množinu lze definovat výčtem nebo společnou vlastností jejich řetězců.

Výše uvedená definice jazyka není v rozporu s chápáním přirozených ani programovacích jazyků.

Postupně budeme pojem **jazyk** precizovat, seznámíme se s přístupy které umožní popisovat jazyky efektivnějšími způsoby.

Jazyky chápeme jako množiny, lze nad nimi tedy definovat stejné operace jako nad množinami.

Operace nad jazyky

L_1 a L_2 jsou jazyky nad abecedou Σ , tedy $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Sigma^*$

- **Sjednocení jazyků:** $L = L_1 \cup L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \vee w \in L_2)\}$$

- **Průnik jazyků:** $L = L_1 \cap L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \wedge w \in L_2)\}$$

- **Doplňěk jazyka:** $L = \overline{L_1}$

$$L = \{w \mid w \in \Sigma^* \wedge w \notin L_1\}$$

Operace nad jazyky

L_1 a L_2 jsou jazyky nad abecedou Σ , tedy $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Sigma^*$

- **Rozdíl jazyků:** $L = L_1 / L_2$

$$L = \{w \mid w \in \Sigma^* \wedge (w \in L_1 \wedge w \notin L_2)\}$$

- **Zřetězení jazyků:** $L = L_1 \cdot L_2 = L_1 L_2$

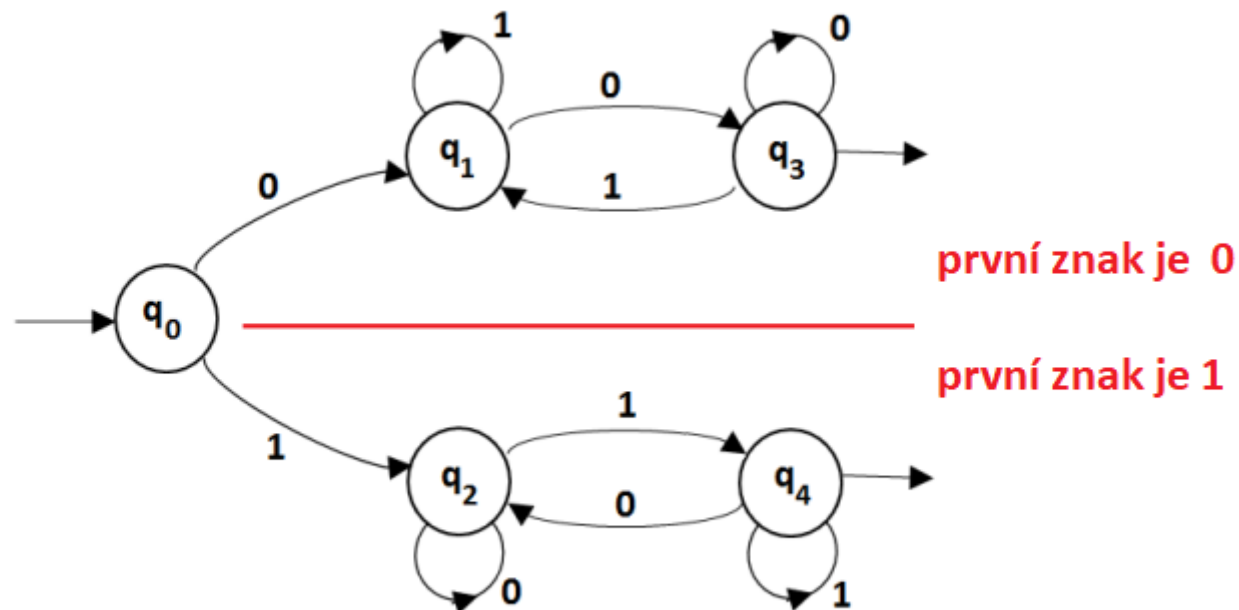
$$L = \{w \mid w \in \Sigma^* \wedge (w = u \cdot v \wedge u \in L_1 \wedge v \in L_2)\}$$

Př.: $A = \{a, b, c\}$, $B = \{ba, bb\}$, $AB = \{aba, bba, cba, abb, bbb, cbb\}$

Motivační příklad 1

- Nad abecedou $\{0,1\}$ je dán jazyk

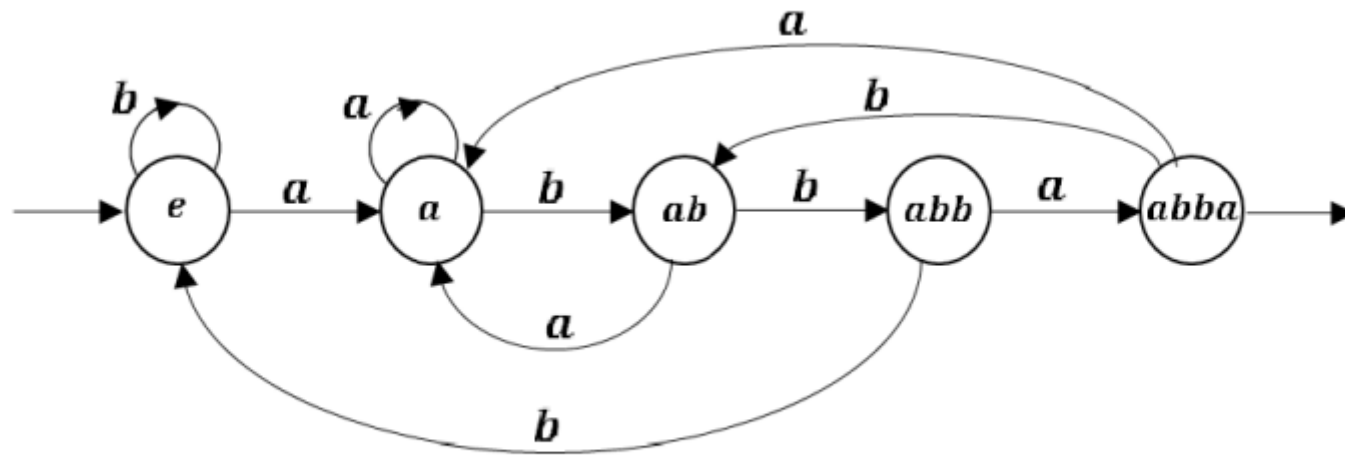
$$L = \{w \mid w \in \{0,1\}^* \wedge |w| \geq 2 \wedge w \text{ začíná a končí stejným symbolem}\}$$



Motivační příklad 2

- Nad abecedou $\{a,b\}$ je dán jazyk

$$L = \{w \mid w \in \{a,b\}^* \wedge w \text{ končí podřetězcem } abba\}$$



Motivační příklad 3

- Nad abecedou $\{0,1\}$ je dán jazyk $L = \{w \mid w = 0^n 1^n \wedge n \geq 1\}$

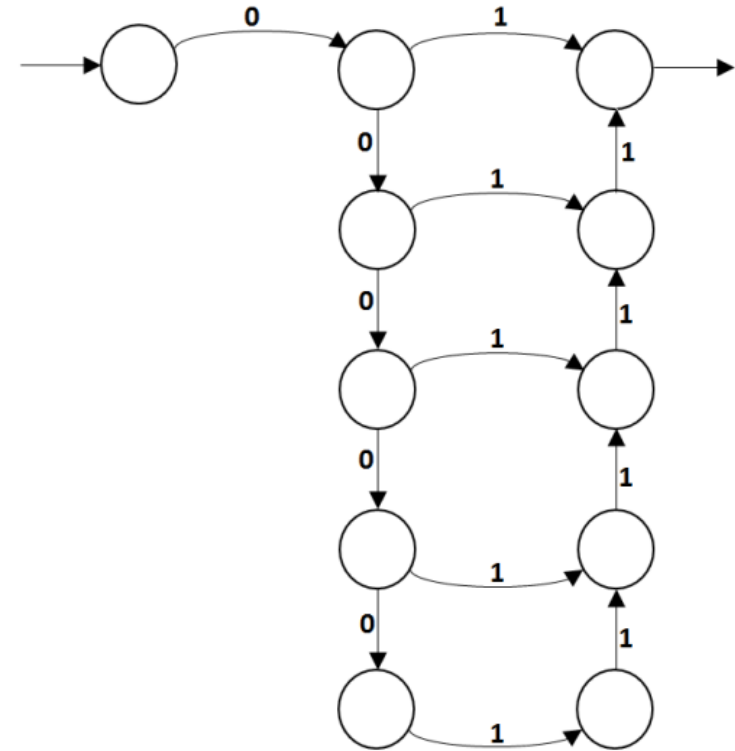
Jak vypadají řetězce tohoto jazyka?

$$L = \{01, 0011, 000111, 00001111, \dots\}$$

- Automat si prostřednictvím svých stavů musí „pamatovat“
(přinejmenším) počet nul na začátku řetězce
- Kolik musí mít automat stavů? Přinejmenším tolik, kolik nul může být na začátku řetězce (tj. n)
- n není shora omezeno \Rightarrow počet stavů KA nelze shora omezit \Rightarrow
na syntaktickou analýzu tohoto jazyka nestačí konečný automat

Motivační příklad 3

- Konečným automatem lze syntaktickou analýzu řešit pouze v případě, že bude n shora omezeno.
- Ilustrace: $L = \{w \mid w = 0^n 1^n \wedge 1 \leq n \leq 5\}$
- Zjednodušená forma přechodového grafu



- Nezakreslené hrany u rozpoznávacího automatu interpretujeme tak, že vedou do nezakresleného chybového stavu (ten je nekonečný a absorpční)

Závěr k motivačním příkladům

- Existují jazyky, které lze rozpoznávat konečným automatem, tj.
 - zpracováním řetězce $w \in L$ automat přejde do koncového stavu
 - zpracováním řetězce $w \notin L$ automat přejde do nekoncevého stavu
- Existují jazyky, které rozpoznávat konečným automatem nelze.
- Co rozhoduje o tom, zda na rozpoznání jazyka stačí konečný automat? Jak to poznáme?
- Ze slovního, respektive množinového popisu jazyka to zřejmé není, **je nutné pojem jazyk specifikovat precizněji**

Způsoby popisu (obecného) formálního jazyka

- Množinový *(již jsem poznali)*
 - výčtem
 - společnou vlastností řetězců
- Akceptační *(tj. automatem, který jazyk rozpoznává; automat nemusí být konečný, existují i jiné výpočetně silnější automatové modely)*
- Generativní *(tj. pravidly pro vytváření řetězců, tedy **gramatikou**)*

Akceptační x generativní popis jazyka

- Každý rozpoznávací automat (tedy i konečný rozpoznávací automat) jednoznačně definuje jazyk jako množinu všech řetězců, které automat převedou z počátečního stavu do některého ze stavů koncových.
- Myšlenka generativního popisu: pomocí formálních pravidel popsat „správné řetězce“ (řetězce, které patří do jazyka). Analogie gramatických pravidel v přirozených jazycích.

Příklad gramatiky

- Př.: Gramatika pro generování jednoduchých vět nad omezenou množinou anglických slov. Pravidla pro vytváření řetězců jazyka (**přepisovací pravidla**):

Pravidlo 1 :	<věta>	→ <podmětná část><přísudková část>
Pravidlo 2 :	<podmětná část>	→ <podmět>
Pravidlo 3 :	<podmětná část>	→ <přívlastek><podmět>
Pravidlo 4 :	<přísudková část>	→ <přísudek>
Pravidlo 5 :	<přísudková část>	→ <přísudek><předmět>
Pravidlo 6 :	<podmět>	→ John
Pravidlo 7 :	<podmět>	→ Mary
Pravidlo 8 :	<přívlastek>	→ our
Pravidlo 9 :	<přívlastek>	→ your
Pravidlo 10 :	<přísudek>	→ drinks
Pravidlo 11 :	<přísudek>	→ buys
Pravidlo 12 :	<předmět>	→ tea
Pravidlo 13 :	<předmět>	→ gin

Příklad gramatiky

- Př.: Gramatika pro generování jednoduchých vět nad omezenou množinou anglických slov.

Příklady řetězců, které lze pravidly odvodit:

John drinks tea

Mary buys gin

our john buys tea

Kolik řetězců nám tato pravidla umožňují odvodit? $3 * 2 * 2 * 3 = 36$ řetězců.

Příklad gramatiky

- Př.: Gramatika pro generování jednoduchých vět nad omezenou množinou anglických slov.

Jak se řetězce odvozují?

vychází se od (pomocného) symbolu <věta>

postupně nahrazujeme pomocné symboly jejich rozvoji definovanými

odvozovacími pravidly

odvozování končí tehdy, když už v odvozeném řetězci jsou pouze anglická slova a není tam žádný pomocný symbol (český název větného členu)

Příklad gramatiky

Příklad odvození řetězce (přesný význam operátoru \Rightarrow bude objasněn později)

Pravidlo 1 :	<věta>	→	<podmětná část><přísudková část>
Pravidlo 2 :	<podmětná část>	→	<podmět>
Pravidlo 3 :	<podmětná část>	→	<přívlastek><podmět>
Pravidlo 4 :	<přísudková část>	→	<přísudek>
Pravidlo 5 :	<přísudková část>	→	<přísudek><předmět>
Pravidlo 6 :	<podmět>	→	John
Pravidlo 7 :	<podmět>	→	Mary
Pravidlo 8 :	<přívlastek>	→	our
Pravidlo 9 :	<přívlastek>	→	your
Pravidlo 10 :	<přísudek>	→	drinks
Pravidlo 11 :	<přísudek>	→	buys
Pravidlo 12 :	<předmět>	→	tea
Pravidlo 13 :	<předmět>	→	gin

$\begin{aligned} &<\text{věta}> \stackrel{1}{\Rightarrow} <\text{podmětná část}> <\text{přísudková část}> \stackrel{3}{\Rightarrow} \\ &\Rightarrow <\text{přívlastek}> <\text{podmět}> <\text{přísudková část}> \stackrel{8}{\Rightarrow} \text{our} <\text{podmět}> <\text{přísudková část}> \stackrel{7}{\Rightarrow} \\ &\Rightarrow \text{our Mary} <\text{přísudková část}> \stackrel{5}{\Rightarrow} \text{our Mary} <\text{přísudek}> <\text{předmět}> \stackrel{10}{\Rightarrow} \\ &\Rightarrow \text{our Mary drinks} <\text{předmět}> \stackrel{13}{\Rightarrow} \text{our Mary drinks gin} \end{aligned}$

Závěry z příkladu gramatiky

- Jaké objekty se vyskytovaly v přepisovacích pravidlech, pomocí kterých jsme generovali řetězec?
 - pomocné symboly (české názvy větných členů, v ostrých závorkách <>)
 - jedním z pomocných symbolů byl počáteční symbol (<věta>)
 - cílové symboly; jen ty se mohly vyskytovat ve finálním řetězci (anglická slova a jména)

Definice gramatiky

- Gramatika je uspořádaná čtveřice $G = (N, T, S, P)$, kde

N množina neterminálních symbolů

T množina terminálních symbolů

$$N \cap T = \emptyset$$

$S \in N$ počáteční symbol

P množina přepisovacích pravidel ve tvaru $\alpha \rightarrow \beta$

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

$\beta \in (N \cup T)^*$

Definice gramatiky

- Vysvětlení ke tvaru přepisovacích pravidel $\alpha \rightarrow \beta$:

$$\alpha \in (N \cup T)^* N (N \cup T)^*$$

(na levé straně musí být alespoň jeden neterminální symbol)

$$\beta \in (N \cup T)^*$$

(na pravé straně může být i prázdný řetězec)

- Běžné konvence:

neterminální symboly: <podmět>, <identifikátor>, ... nebo S, A, B, C,

terminální symboly: 0,1,2, nebo a, b, c, d, e, f,

řetězce: $\alpha, \beta, \gamma, \delta$, nebo u, v, w, x, y, z

Syntaktický diagram

- syntaktický diagram = způsob reprezentace (některých) gramatik v grafické podobě
- použit např. k popisu programovacího jazyka Pascal

Příklad:

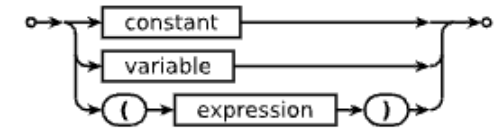
$\langle \text{factor} \rangle \rightarrow \langle \text{constant} \rangle \mid \langle \text{variable} \rangle \mid (\langle \text{expression} \rangle)$

$\langle \text{variable} \rangle \rightarrow x \mid y \mid z$

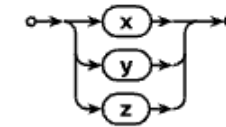
$\langle \text{constant} \rangle \rightarrow \langle \text{digit} \rangle \mid \langle \text{digit} \rangle \langle \text{constant} \rangle$

$\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

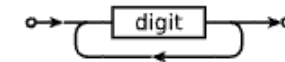
factor:



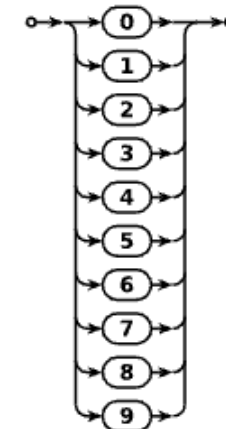
variable:



constant:



digit:



Jak gramatika generuje jazyk?

- Gramatika G generuje jazyk $L(G)$ jako množinu všech terminálních řetězců, které lze v gramatice odvodit z počátečního symbolu
[verbální formulace, ne zcela přesná]
- Přesnější formulace používá pojmy **přímé přepsání** a **přepsání**.
- Nechť existují řetězce $w \in (N \cup T)^* N (N \cup T)^*$ a $z \in (N \cup T)^*$
Řetězec w lze **přímo přepsat** na řetězec z právě tehdy, když existují řetězce $x_1, x_2, u, v \in (N \cup T)^*$ takové, že

$$w = x_1 u x_2 \wedge z = x_1 v x_2 \wedge u \rightarrow v \in P$$

Značení: $w \Rightarrow z$

Jak gramatika generuje jazyk?

- Definice přímého přepsání zcela beze slov:

$$w \Rightarrow z \stackrel{\text{def}}{\iff} \exists x_1, x_2, u, v \in (N \cup T)^*: w = x_1 u x_2 \wedge z = x_1 v x_2 \wedge u \rightarrow v \in P$$

- Nechť existují řetězce $w \in (N \cup T)^* N (N \cup T)^*$ a $z \in (N \cup T)^*$

Řetězec w lze **přepsat** na řetězec z právě tehdy, když existují řetězce

řetězce $w_0, w_1, \dots, w_n \in (N \cup T)^*$ takové, že

$$w = w_0 \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_n = z$$

Tato sekvence přímých přepsání se nazývá odvozením (derivací) slova z

ze slova w . Délka tohoto odvození je n . Značení: $w \xRightarrow{*} z$

Ilustrace přímého přepsání

our <podmět> <přísudková část> \Rightarrow **our John** <přísudková část>

Proč?

V P je pravidlo 6: <podmět> \rightarrow **John**

<podmět> \rightarrow **John** $\in P$, tedy $u = \text{<podmět>}$, $v = \text{John}$

$w = \text{our } \text{<podmět> } \text{<přísudková část>}$

$x_1 \quad u \quad x_2$

\Rightarrow

our John <přísudková část> = z

$x_1 \quad v \quad x_2$

x_1 levý kontext, x_2 pravý kontext

Ilustrace přímého přepsání

$\langle \text{přísudek} \rangle \Rightarrow \text{buys}$

Proč?

V P je pravidlo 11: $\langle \text{přísudek} \rangle \rightarrow \text{buys}$

$\langle \text{přísudek} \rangle \rightarrow \text{buys} \in P$, tedy $u = \langle \text{přísudek} \rangle$, $v = \text{buys}$

$$\begin{array}{ccc} w & = & \langle \text{přísudek} \rangle \\ & & x_1 \quad u \quad x_2 \\ & & \Rightarrow \\ & & \text{buys} \quad = \quad z \\ & & x_1 \quad v \quad x_2 \end{array}$$

Levý i pravý kontextový řetězec jsou prázdné.

Jak odvodit řetězec?

- Vyjdeme od počátečního symbolu
- V dosud odvozeném řetězci najdeme levou stranu některého přepisovacího pravidla.
- Řetězec přepíšeme tak, že tuto levou stranu nahradíme odpovídající pravou stranou, přičemž levý i pravý kontext zůstane zachován.
- Takto postupujeme dokud nedojdeme k řetězci, který se skládá pouze z terminálních symbolů.

Definice jazyka generovaného gramatikou

- Jazykem generovaným gramatikou G rozumíme množinu všech terminálních řetězců, které lze v gramatice odvodit z počátečního symbolu (již jsme slyšeli).

$$L(G) = \{ w \mid w \in T^* \wedge S \xRightarrow{*} w \}$$

Příklad gramatiky

$$G = (N, T, S, P)$$

Zjednodušený zápis gramatiky: $S \rightarrow 0S1 \mid 01$

$$N = \{S\}, \quad T = \{0,1\}, \quad S, \quad P = \{S \rightarrow 0S1, S \rightarrow 01\}$$

Odvozování řetězců: $S \Rightarrow 01$

$$S \Rightarrow 0S1 \Rightarrow 0011$$

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$$

.....

Zřejmě $L(G) = \{0^n 1^n ; n \geq 1\}$ (již jsme poznali, nestačí na něj KA)