

Automaty a jazyky

Studijní materiál ke státnicím (I1)

Jakub Klesa

Pokryté předměty: NTIN071 (Automaty a gramatiky)

Obsah

Úvod	1
1 Regulární jazyky (gramatiky, DFA/NFA, regulární výrazy)	2
1.1 Deterministický konečný automat (DFA)	2
1.2 Nedeterministický konečný automat (NFA)	4
1.3 Regulární výrazy a Kleeneho věta	6
1.4 Regulární gramatiky (typ 3)	7
1.5 Iterační (pumpovací) lemma a uzávěrové vlastnosti	7
2 Bezkontextové jazyky (gramatiky, zásobníkový automat)	9
2.1 Bezkontextová gramatika (CFG)	9
2.2 Chomského normální forma	10
2.3 Zásobníkový automat (PDA)	10
2.4 Ekvivalence CFG $\leftrightarrow$ PDA a převody	11
3 Rekurzivně spočetné jazyky (Turingův stroj, nerozhodnutelnost)	13
3.1 Turingův stroj	13
3.2 Gramatiky typu 0	14
3.3 Algoritmicky nerozhodnutelné problémy	14
4 Chomského hierarchie a zařazování jazyků	15
4.1 Klasifikace gramatik a hierarchie	15
4.2 Zařazení konkrétního jazyka	16
Shrnutí	16
Zdroje	18

Úvod

Tento materiál pokrývá okruh **I1 – Automaty a jazyky** (předmět NTIN071 *Automaty a gramatiky*). Výklad stojí na dvou zdrojích z MFF UK na úrovni kurzu pro informatiky: na *slidech k přednášce M. Vomlelové*, z nichž bereme přesný rozsah, značení a konvence NTIN071, a na souvislém textu M. Mareše *Úvod do automatů*, o který se opíráme u prózy, intuice a důkazů u regulárních jazyků, bezkontextových gramatik a Turingových strojů. Značení a hloubka odpovídají kurzu, ne obecné teorii vyčíslitelnosti.

Role důkazů. Požadavky okruhu kladou důraz na *konstrukce*: navrhnout konečný automat, bezkontextovou gramatiku nebo zásobníkový automat pro zadaný jazyk, provést standardní převod (podmnožinová konstrukce, gramatika $\leftrightarrow$ automat, gramatika $\rightarrow$ zásobníkový automat) a zařadit konkrétní jazyk do

Chomského hierarchie. Pojmenované věty proto uvádíme se zněním, intuicí a u klíčových převodů s *konstrukcí*, kterou je dobré umět provést na konkrétním vstupu. Plné formální důkazy (Kleeneho věta, ekvivalence modelů, nerozhodnutelnost) uvádíme jen v náčrtu a označujeme jako *nepovinné*; podstatné je porozumět jejich myšlence. Do výkladu je zařazeno všech sedm reálných otázek SZZ k okruhu z let 2023–2026 jako řešené příklady; jsou i v samostatném dokumentu *Příklady otázek*. Použité zdroje jsou uvedeny na konci.

Značení. *Abeceda* Σ je konečná neprázdná množina *znaků*. *Slovo* (řetězec) je konečná posloupnost znaků; množinu všech slov nad Σ značíme Σ^* , *prázdné slovo* (délky 0) značíme λ (některé texty píší ε). Délka slova w je $|w|$, počet výskytů znaku a ve slově je $|w|_a$. *Zřetězení* slov u, v píšeme uv , mocninu w^k , otočení w^R . Slovo u je *prefix* (*suffix*) slova w , je-li $w = uv$ (resp. $w = vu$) pro nějaké v ; je *pod slovem* (vždy souvislým, tj. faktorem), je-li $w = xuy$ pro nějaká x, y . *Jazyk* je libovolná podmnožina Σ^* . Proměnné (neterminály) gramatik píšeme velkými písmeny A, B, S , terminály malými a, b ; $\mathcal{P}$ je množina pravidel. Potenciální množinu množiny stavů značíme 2^Q , její konečné podmnožiny $\mathcal{P}_{\text{FIN}}(\cdot)$. Třídy jazyků Chomského hierarchie značíme $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$ (regulární, bezkontextové, kontextové, rekursivně spočetné).

1 Regulární jazyky (gramatiky, DFA/NFA, regulární výrazy)

Regulární jazyky jsou nejmenší a nejlépe ovladatelná třída. Popíšeme je čtyřmi navzájem ekvivalentními způsoby: *deterministickým a nedeterministickým konečným automatem, regulárním výrazem a regulární (pravou lineární) gramatikou*. Že jde skutečně o tutéž třídu, je obsahem Kleeneho věty a převodů, které tvoří jádro této sekce.

1.1 Deterministický konečný automat (DFA)

Definice – Deterministický konečný automat

Deterministický konečný automat (DFA) je pětice $A = (Q, \Sigma, \delta, q_0, F)$, kde

- Q je konečná neprázdná množina *stavů*,
- Σ je konečná neprázdná *vstupní abeceda*,
- $\delta : Q \times \Sigma \rightarrow Q$ je *přechodová funkce*,
- $q_0 \in Q$ je *počáteční stav*,
- $F \subseteq Q$ je množina *přijímajících* (koncových) stavů.

 **Intuice.** Automat čte vstup zleva doprava, znak po znaku, a drží si jediný „aktuální stav“ – konečnou paměť. Přechodová funkce říká, kam se ze stavu posunout po přečtení znaku. Po dočtení vstupu se podívá, zda skončil v přijímajícím stavu. Determinismus znamená, že z každého stavu vede pro každý znak *právě jeden* přechod; výpočet je tak vstupem jednoznačně určen.

Abychom mohli mluvit o přijetí celého slova, rozšíříme δ ze znaků na slova.

Definice – Rozšířená přechodová funkce, přijímaný jazyk

Rozšířená přechodová funkce $\delta^* : Q \times \Sigma^* \rightarrow Q$ je dána induktivně:

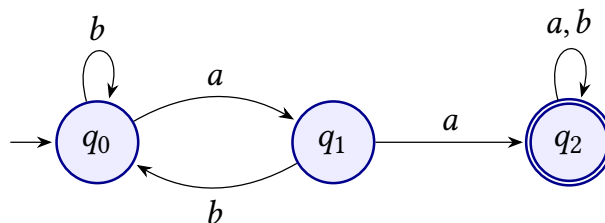
$$\delta^*(q, \lambda) = q, \quad \delta^*(q, wx) = \delta(\delta^*(q, w), x) \quad (q \in Q, w \in \Sigma^*, x \in \Sigma).$$

Slovo w je *přijímáno*, pokud $\delta^*(q_0, w) \in F$. *Jazyk přijímaný automatem* A je

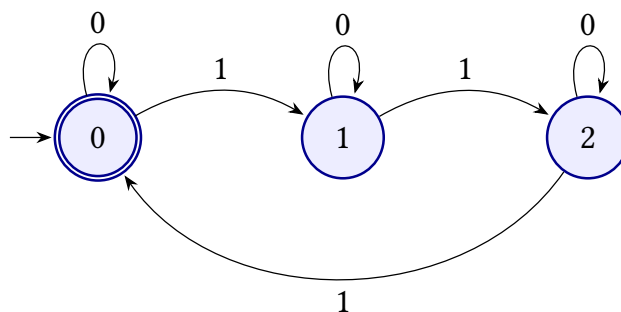
$$L(A) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}.$$

Jazyk L je *regulární*, existuje-li DFA A s $L = L(A)$.

V obrázcích kreslíme stav jako kroužek, počáteční stav značíme šipkou „odnikud“, přijímající stavy dvojitým okrajem. Hrana z p do q označená x znamená $\delta(p, x) = q$.



obecné schéma DFA (konvence kreslení) nad $\Sigma = \{a, b\}$. **Kroužek** = stav; **vstupní šipka** „odnikud“ = počáteční stav (q_0); **dvojitý kroužek** = přijímající stav (q_2); **šipka** mezi stavy s návěštím x = přechod $\delta(p, x) = q$. U DFA vede z každého stavu právě jeden přechod na každý symbol abecedy (každý stav je „totální“): zde q_0 má hranu na a i na b , q_1 má hranu na a i na b a q_2 má smyčku a, b (jeden přechod značí obě hodnoty). Výpočet je proto vstupem jednoznačně určen.



DFA nad $\Sigma = \{0, 1\}$ přijímající $\{w \mid |w|_1 \equiv 0 \pmod{3}\}$, tj. slova s počtem jedniček dělitelným třemi. Stav i znamená „dosud přečteno i jedniček mod 3“; znak 0 zbytek nemění (smyčky), znak 1 zvyšuje o 1. Počáteční i jediný přijímající stav je 0. Např. při čtení 0110 projdeme stavy $0 \rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 2$ (poslední 0 stav nemění), takže $\delta^*(0, 0110) = 2 \notin F$ a slovo 0110 přijato *není* ($|0110|_1 = 2$).

i Poznámka. Determinismus dává okamžitě **rozhodovací algoritmus**: slovo délky n otestujeme v čase $O(n)$ prostým odsimulováním přechodů. To je hlavní praktická přednost konečných automatů (vyhledávání vzorů, lexikální analýza).

Příklad — SZZ léto 2025 č. 1: Deterministické konečné automaty

(Reálná otázka SZZ: léto 2025, č. 1 – Deterministické konečné automaty (jazyk DFA, rozšířená přechodová funkce))

Zadání.

1. Uvedte definici jazyka $L(A)$ přijímaného daným DFA $A = (Q, \Sigma, \delta, q_0, F)$ včetně definice rozšířené přechodové funkce δ^* .
2. Nechť $S \subseteq \Sigma^*$ je neprázdná konečná množina slov, Q je množina všech prefixů slov z S (vč. λ a slov z S) a $A = (Q, \Sigma, \delta, \lambda, S)$. Definujte δ tak, aby A přijímal jazyk $\{w \mid w \text{ obsahuje nějaké } s \in S \text{ jako podslovo}\}$.
3. Sestrojte DFA přijímající $\{w \in \{0, 1\}^* \mid w \text{ obsahuje } 010 \text{ nebo končí na } 10\}$.

Řešení.

(1) Přesně definice výše: $\delta^*(q, \lambda) = q$, $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$ a $L(A) = \{w \mid \delta^*(q_0, w) \in F\}$.

(2) Stav je nejdelší sufix dosud přečteného slova, který je v Q (tj. je prefixem nějakého $s \in S$ nebo přímo nějakým s). Jakmile se objeví celé $s \in S$, zůstaneme v něm napořád (podslovo už ze slova

nezmizí). Pro $w \in Q$, $x \in \Sigma$:

$$\delta(w, x) = \begin{cases} w, & w \in S \text{ (absorbující – podslovo už nalezeno),} \\ s, & w \notin S \text{ a } wx \text{ končí na nějaké } s \in S \text{ (vezmi nejkratší),} \\ v, & \text{jinak; } v = \text{nejdelší sufix slova } wx \text{ ležící v } Q. \end{cases}$$

Přijímající stavy jsou $F = S$. Indukcí podle $|w|$: po přečtení w je automat ve stavu „nejdelší sufix w , který je v Q “, případně v absorbujícím s , pokud w obsahuje nějaké $s \in S$.

(3) Zde je $S = \{010, 10\}$, ale s tím rozdílem, že „010“ se hledá jako *podslovo* (absorbující), kdežto „10“ se vyžaduje jen *na konci*. Stavy odpovídají relevantním sufixům $Q = \{\lambda, 0, 1, 01, 10, 010\}$; jediný absorbující přijímající stav je 010, druhý přijímající stav 10 je „obyčejný“ (značí, že právě teď slovo končí na 10, ale ještě jsme neviděli 010). Přejchodová funkce:

stav	$\delta(\cdot, 0)$	$\delta(\cdot, 1)$
λ	0	1
0	0	01
1	10	1
01	010	1
10	0	01
010	010	010

Počáteční stav λ , přijímající $F = \{10, 010\}$. Že automat přijímá přesně $\{w \mid w \text{ obsahuje } 010 \text{ nebo končí na } 10\}$, jsme ověřili vyčerpáním všech slov do délky 12 (Python). Klíč: ze stavu 010 se už nedostaneme (podslovo nalezeno), kdežto ze stavu 10 po přečtení dalšího znaku „spadneme“ (na 0 do stavu 0, na 1 do 01), takže přijetí přes 10 vyžaduje skutečný konec na 10.

Stavový diagram (oříznutý z originálu) je v příklady/.

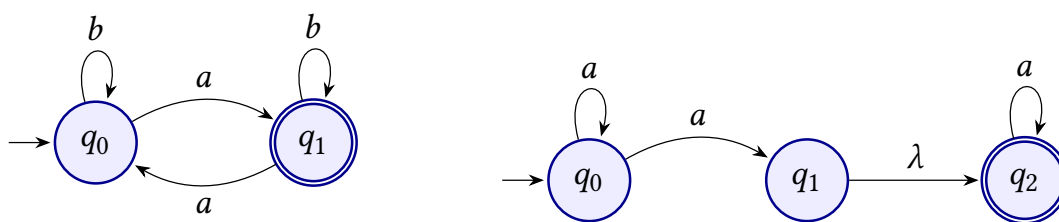
1.2 Nedeterministický konečný automat (NFA)

Při konstrukci je často pohodlnější dovolit z jednoho stavu *více* (nebo *žádný*) přechod pro tentýž znak a „uhodnout“ správnou cestu. To vede k nedeterminismu.

Definice – Nedeterministický konečný automat

Nedeterministický konečný automat (NFA) je pětice $A = (Q, \Sigma, \delta, Q_0, F)$, kde $\delta : Q \times \Sigma \rightarrow 2^Q$ přiřadí dvojici (stav, znak) množinu možných následujících stavů, $Q_0 \subseteq Q$ je množina počátečních stavů a ostatní složky jsou jako u DFA. Výpočet je libovolná posloupnost stavů $s_0, \dots, s_{|w|}$ s $s_0 \in Q_0$ a $s_{i+1} \in \delta(s_i, w[i])$. Slovo je *přijato*, existuje-li *aspoň jeden* výpočet končící v F .

Intuice. NFA přijme slovo, pokud *existuje* úspěšná cesta – jako bychom směli vždy zvolit tu šťastnou možnost (nebo paralelně sledovat všechny). To usnadňuje návrh (jeden „hlídač“ na každou alternativu), ale nezvyšuje sílu: nedeterminismus jde vždy odstranit.



DFA versus NFA (nedeterminismus). *Vlevo DFA*: z každého stavu vede na každý symbol *právě jeden* přechod (z q_0 vede na a jediná hrana do q_1), výpočet je jednoznačný. *Vpravo NFA*: ze stavu q_0 vedou na týž symbol a dva přechody (smýčka do q_0 a hrana do q_1) – automat si „vybírá“; navíc je tu λ -přechod $q_1 \rightarrow q_2$, který *nečte žádný symbol*. NFA tedy z jednoho stavu může mít na symbol více přechodů (nebo žádný) a slovo *přijme*, pokud existuje *aspoň jedna* úspěšná větev výpočtu končící v přijímajícím stavu.

★ Věta – Podmnožinová konstrukce

Ke každému NFA A existuje DFA A' s $L(A') = L(A)$. Tudiž jazyk je rozpoznatelný NFA právě tehdy, je-li regulární.

Důkaz (nepovinný – konstrukce). Stav DFA bude celá *množina* stavů, v nichž se NFA může nacházet. Položíme $A' = (2^Q, \Sigma, \delta', Q_0, F')$, kde

$$\delta'(S, x) = \bigcup_{s \in S} \delta(s, x), \quad F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}.$$

Indukcí podle délky slova platí $\delta'^*(Q_0, w) = \{\delta\text{-dosažitelné stavy NFA po } w\}$, takže A' přijme w právě když nějaký výpočet NFA končí v F . Stačí vzít *dosažitelné* stavy (typicky jich je málo, ne všech $2^{|Q|}$). $\square$

i Poznámka. [λ -přechody] Při skládání automatů (sjednocení, zřetězení, iterace) se hodí přechody, které *nečtou žádný znak* – tzv. λ -přechody. λ -NFA má $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$. I ty jdou odstranit: stavy zachováme a přechodovou funkci i počáteční stavy „uzavřeme“ na λ -okolí (λ -uzávěr stavu = množina stavů dosažitelných po samých λ -hranách). Proto i λ -NFA rozpoznávají právě regulární jazyky.

Následující reálná otázka prochází celý okruh „regulární výraz $\rightarrow$ NFA $\rightarrow$ DFA $\rightarrow$ doplněk“ a je dobrým souhrnem této a předchozí podsektce.

🔪 Příklad – SZZ podzim 2024 č. 1: Doplněk regulárního výrazu

(Reálná otázka SZZ: podzim 2024, č. 1 – Doplněk regulárního výrazu (NFA/DFA, podmnožinová konstrukce, doplněk))

Zadání.

Pro $R = ((a + b)(a + b))^*ab$ nad $\Sigma = \{a, b\}$ a $L = L(R)$:

1. Sestrojte (co nejmenší) NFA A pro L .
2. Podmnožinovou konstrukcí převeďte A na DFA B .
3. Z B sestrojte DFA C pro *doplněk* $\bar{L}$.

Řešení.

Výraz R popisuje slova *sudé délky končící na ab* (před závěrečným ab stojí libovolný *sudý* počet znaků).

(1) **NFA A .** Stavy $\{q_0, q_1, q_2, q_F\}$: q_0 je „začátek dvojice / odhad konce“, q_1 „uprostřed dvojice“, q_2 „uhodli jsme závěrečné a “, q_F přijímá. Nedeterministicky buď spotřebujeme dvojici $(a+b)(a+b)$ a vrátíme se do q_0 , nebo zahájíme závěrečné ab :

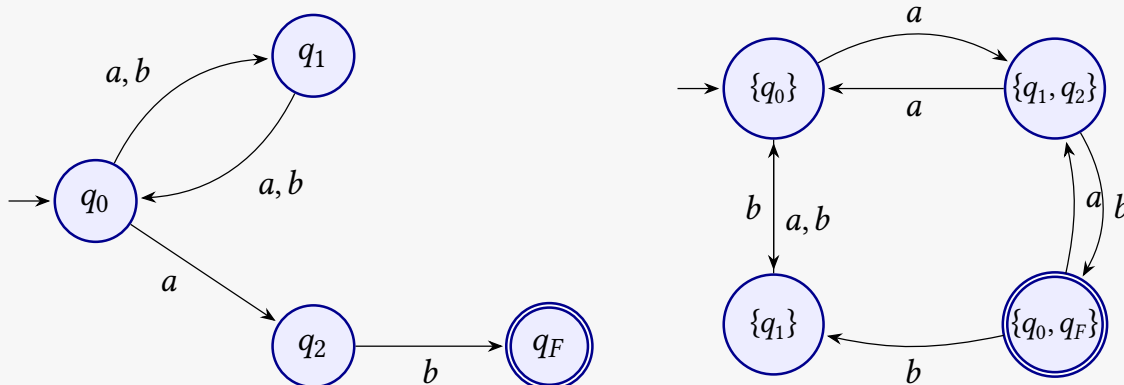
$$\delta(q_0, a) = \{q_1, q_2\}, \quad \delta(q_0, b) = \{q_1\}, \quad \delta(q_1, a) = \delta(q_1, b) = \{q_0\}, \quad \delta(q_2, b) = \{q_F\}.$$

(2) **DFA B (podmnožinová konstrukce).** Dosažitelné množiny stavů (počáteční $\{q_0\}$):

stav B	a	b
$\{q_0\}$	$\{q_1, q_2\}$	$\{q_1\}$
$\{q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_F\}$
$\{q_1\}$	$\{q_0\}$	$\{q_0\}$
$\{q_0, q_F\}^*$	$\{q_1, q_2\}$	$\{q_1\}$

Jediný přijímající stav je $\{q_0, q_F\}$ (obsahuje q_F). Přechodová funkce je *totální* (každý stav má oba přechody), takže nepřibyl žádný stav „žumpa“.

(3) **DFA C pro doplněk.** U DFA s totální δ stačí prohodit přijímající a nepřijímající stavy: C má tytéž stavy a přechody jako B, ale $F_C = \{\{q_0\}, \{q_1, q_2\}, \{q_1\}\}$. (Kdyby δ totální nebyla, museli bychom napřed doplnit žumpu, jinak by doplněk vynechal slova, na kterých B „uvízne“.)



vlevo nejmenší NFA A pro $((a + b)(a + b))^*ab$ (4 stavy; méně jich být nemůže: pro dvojice $(x_i, y_i) \in \{(\lambda, ab), (a, b), (ab, \lambda), (b, aab)\}$ je vždy $x_i y_i \in L$, ale pro $i \neq j$ do L nepatří $x_i y_j$ nebo $x_j y_i$, takže stavy po přečtení x_i na přijímajících výpočtech musí být navzájem různé), vpravo DFA B z podmnožinové konstrukce (4 dosažitelné stavy, δ totální). DFA C pro $\bar{L}$ vznikne z B záměnou přijímajících stavů (přijímající budou $\{q_0\}, \{q_1, q_2\}, \{q_1\}$). Správnost NFA i konstrukce ověřena vyčerpáním slov do délky 10. Diagramy všech tří automatů (oříznuté z originálu) jsou v příklady/.

1.3 Regulární výrazy a Kleeneho věta

Definice – Regulární výraz

Regulární výrazy nad Σ a jimi popsané jazyky definujeme induktivně:

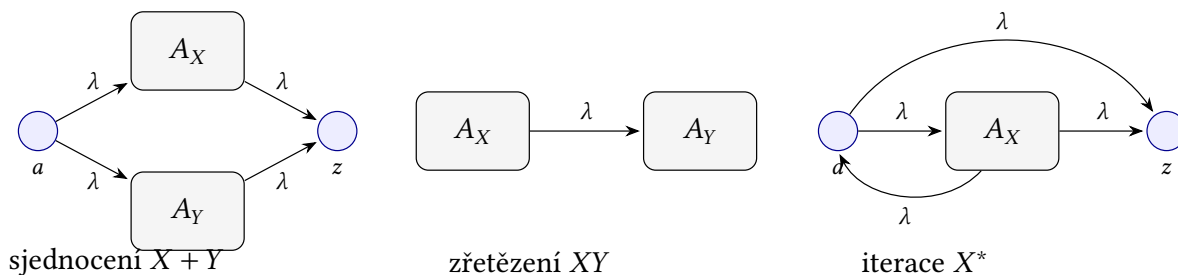
$$\emptyset (L = \emptyset), \quad \lambda (L = \{\lambda\}), \quad x (L = \{x\}, x \in \Sigma),$$

a pro výrazy X, Y : $X + Y$ (sjednocení $L(X) \cup L(Y)$), XY (zřetězení $L(X) \cdot L(Y)$), X^* (iterace $L(X)^*$).
Zkratky: $X^+ = XX^*$, $X? = X + \lambda$.

★ Věta – Kleeneho

Jazyk je popsatečný regulárním výrazem právě tehdy, je-li regulární.

Intuice. Směr „výraz $\Rightarrow$ regulární“ je stavebnice: pro každou operaci sestojíme λ -NFA z dílčích automatů (viz schéma níže). Opačný směr (*eliminace stavů*) z DFA postupně odstraňuje stavy a hrany ohodnocuje regulárními výrazy, až zbude jediná hrana mezi počátečním a koncovým stavem – její ohodnocení je hledaný výraz.



Kleeneho „stavebnice“ regulárních jazyků pomocí λ -NFA. A_X, A_Y jsou automaty pro X, Y s jediným počátečním a jediným koncovým stavem. *Sjednocení*: nový start a cíl, λ -hrany do/z obou. *Zřetězení*: λ -hrana z konce A_X na začátek A_Y . *Iterace*: nový start/cíl, λ -hrany start $\rightarrow$ cíl (přijetí λ), start $\rightarrow A_X$, konec $A_X \rightarrow$ start (opakování) a konec $A_X \rightarrow$ cíl.

i Poznámka. Konkrétně se *eliminace stavů* provádí na GNFA (hrany ohodnocené regulárními výrazy): automat doplníme o nový počáteční stav (λ -hrana do původního startu, bez vstupních hran) a nový koncový stav (λ -hrany ze všech koncových, bez výstupních) a pak po jednom rušíme vnitřní stavy. Odstraníme-li q , pro každou dvojici hran $p \rightarrow q$ a $q \rightarrow r$ přeznačíme hranu $p \rightarrow r$ na

$$R(p, r) + R(p, q) R(q, q)^* R(q, r),$$

kde $R(q, q)$ je smyčka na q (chybí-li, je $R(q, q)^* = \lambda$) a $R(p, r)$ je dosavadní přímá hrana ($\emptyset$, není-li žádná); smyčka rušeného stavu se tak promítne jako hvězdička. Např. pro DFA výše (počet jedniček dělitelný třemi) dá odstranění stavů 1 a 2 výraz $(0 + 10^*10^*1)^*$, ekvivalentně $(0^*10^*10^*1)^*0^*$.

1.4 Regulární gramatiky (typ 3)

Čtvrtý ekvivalentní popis je *generativní*: pravidla, jimiž z počátečního symbolu „vyrobíme“ slovo.

Definice – Pravá lineární gramatika, typ 3

Pravá lineární gramatika (typ 3) je $G = (V, T, \mathcal{P}, S)$, kde V jsou neterminály, T terminály, $S \in V$ počáteční a $\mathcal{P}$ obsahuje jen pravidla tvaru $A \rightarrow wB$ a $A \rightarrow w$ ($A, B \in V, w \in T^*$). *Derivace* přepisuje neterminály podle pravidel; jazyk $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$ je množina slov odvoditelných ze S .

★ Věta – Regulární gramatiky = konečné automaty

Třída jazyků generovaných pravými lineárními gramatikami je přesně třída regulárních jazyků.

💡 Intuice. Soulad je doslovný: *neterminál* = *stav*, *pravidlo* = *přechod*. Z DFA $A = (Q, \Sigma, \delta, q_0, F)$ uděláme gramatiku s $V = Q, S = q_0$ a pravidly $X \rightarrow aY$ pro $\delta(X, a) = Y$ a $X \rightarrow \lambda$ pro $X \in F$. Opačně sestrojíme λ -NFA: pravidlo $A \rightarrow a_1 \dots a_k B$ rozložíme přes nové pomocné stavy na řetěz přechodů z A do B (pro $k = 0$ je to λ -přechod), pravidlo $A \rightarrow a_1 \dots a_k$ vede takovým řetězem do nového koncového stavu. Každé slovo derivace (až na výsledné terminální) obsahuje právě jeden neterminál, a to úplně vpravo – přesně jako „aktuální stav“ při čtení.

i Poznámka. *Levá lineární gramatika* ($A \rightarrow Bw$) generuje rovněž jen regulární jazyky (otočením). Pozor: obecná *lineární gramatika* ($A \rightarrow wBv$) už umí i neregulární jazyk $\{0^n 1^n\}$ – proto patří mezi bezkontextové, ne regulární.

1.5 Iterační (pumpovací) lemma a uzávěrové vlastnosti

Jak ukázat, že jazyk *není* regulární? Konečný automat má konečně mnoho stavů, takže na dlouhém slově se nějaký stav musí zopakovat – a úsek mezi opakováními lze libovolně „napumpovat“.

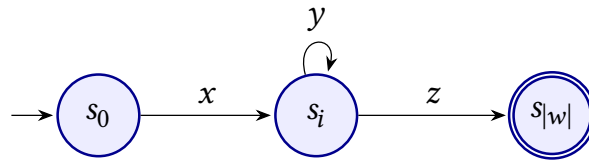
★ Věta – Iterační lemma pro regulární jazyky

Pro každý regulární jazyk L existuje $n \in \mathbb{N}$ tak, že každé $w \in L$ s $|w| \geq n$ lze rozložit na $w = xyz$, kde

$$y \neq \lambda, \quad |xy| \leq n, \quad xy^k z \in L \text{ pro všechna } k \geq 0.$$

Důkaz (nepovinný – princip opakování stavu). Za n vezmeme počet stavů DFA pro L . Při čtení $w, |w| \geq n$, projde automat posloupností $s_0, s_1, \dots, s_{|w|}$, v níž se mezi prvními $n + 1$ stavy nějaký zopakuje: $s_i = s_j$,

$i < j \leq n$. Pro $w = w_1 \cdots w_m$ položíme $x = w_1 \cdots w_i$, $y = w_{i+1} \cdots w_j$, $z = w_{j+1} \cdots w_m$. Smyčka z s_i do s_j čte y , takže ji lze projít k -krát (i nulakrát) se stejným koncovým stavem; proto $xy^kz \in L$. $\square$



jádro iteračního lemmatu. Po přečtení prefixu x je automat ve stavu $s_i = s_j$; úsek y tvoří *smyčku* v tomto stavu. Smyčku lze projít libovolně-krát, proto $xy^kz \in L$ pro každé $k \geq 0$. Naopak: jazyk, který takovou „napumpovatelnost“ porušuje (např. $\{0^m 1^m\}$ – napumpováním y z prvních nul rozbijeme rovnost počtů), *není* regulární.

Intuice. Lemma se používá jako *nutná podmínka*: chceme-li dokázat neregularitu, zvolíme vhodné dlouhé slovo $w \in L$ a ukážeme, že *žádný* rozklad xyz s danými vlastnostmi nepřežije napumpování. Klasika: $\{0^m 1^m\}$, $\{ww \mid w \in \{0, 1\}^*\}$, $\{0^p \mid p \text{ prvočíslo}\}$.

Regulární jazyky jsou *uzavřené* na všechny obvyklé operace. Nejužitečnější konstrukcí je *součin automatů* pro průnik (a po malé úpravě i sjednocení).

★ Věta – Uzávěrové vlastnosti

Jsou-li X, Y regulární, jsou regulární i $X \cup Y$, $X \cap Y$, $\overline{X} = \Sigma^* \setminus X$, $X \cdot Y$, X^* a X^R .

Důkaz (nepovinný – součin pro průnik). Pro $X = L(A_1)$, $Y = L(A_2)$ sestojíme *součin* $A_1 \times A_2$ se stavy $Q_1 \times Q_2$, přechody $\delta((s_1, s_2), x) = (\delta_1(s_1, x), \delta_2(s_2, x))$, počátkem (q_{01}, q_{02}) a $F = F_1 \times F_2$. Automat běží paralelně a přijme, právě když přijmou oba: $L = X \cap Y$. Pro sjednocení vezmeme $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$, doplněk dá záměna F za $Q \setminus F$ (u DFA s totální δ), zřetězení a iterace plynou z Kleeneho stavebnice a otočení X^R z obrácení všech hran automatu (počáteční a koncové stavy si vymění role, vznikne NFA). $\square$

Konstrukce DFA může být i *parametrická* (automat závislý na vstupním parametru). Ukázkou je následující otázka, kde stavy nesou *zbytky modulo p*.

🔗 Příklad – SZZ jaro 2026 č. 1: Porovnání modulárních hashů

(Reálná otázka SZZ: jaro 2026, č. 1 – Porovnání modulárních hashů (definice DFA, konstrukce DFA))

Zadání.

Pro prvočíslo $p \geq 2$ a $w \in \{0, 1\}^*$ buď $h(w) = \left(\sum_{i=1}^{|w|} w_i 2^{|w|-i}\right) \bmod p$ (hodnota binárního zápisu w modulo p , $h(\lambda) = 0$). Uvažme $L_p = \{u\#v \mid u, v \in \{0, 1\}^*, h(u) = h(v)\}$ nad $\Sigma = \{0, 1, \#\}$. (1) Definujte DFA. (2) Popište konstrukci DFA A_p pro L_p podle p . (3) Nakreslete A_2 .

Řešení.

(1) Pětice $A = (Q, \Sigma, \delta, q_0, F)$ jako v definici výše (zde δ může být parciální – nedefinované přechody vedou do implicitní žumpy).

(2) Čteme-li binární číslo zleva, platí Hornerovo schéma: přečtením dalšího bitu b se zbytek i změní na $(2i + b) \bmod p$. Stav si pamatuje zbytek právě čtené části:

- $Q = [p] \cup [p] \times [p]$, kde $[p] = \{0, \dots, p-1\}$. Stav i znamená „čteme u , zatím $h(u') = i$ “; stav (i, j) znamená „už jsme přečetli $\#$, $h(u) = i$ a zatím $h(v') = j$ “.
- $q_0 = 0$, $F = \{(i, i) \mid i \in [p]\}$.
- $\delta(i, 0) = 2i \bmod p$, $\delta(i, 1) = (2i + 1) \bmod p$, $\delta(i, \#) = (i, 0)$;
- $\delta((i, j), 0) = (i, 2j \bmod p)$, $\delta((i, j), 1) = (i, (2j + 1) \bmod p)$.

Druhé $\#$ nemá přechod – vede do žumpy, takže A_p přijme jen slova tvaru $u\#v$ s $h(u) = h(v)$. Správnost ověřena vyčerpáním slov do délky 8 nad $\{0, 1, \#\}$ pro $p = 2, 3$ a 5 (Python).

(3) Pro $p = 2$ je $[2] = \{0, 1\}$, stavy $\{0, 1, (0, 0), (0, 1), (1, 0), (1, 1)\}$, přijímající $(0, 0), (1, 1)$:

stav	0	1	#
0	0	1	(0, 0)
1	0	1	(1, 0)
(0, 0)*	(0, 0)	(0, 1)	–
(0, 1)	(0, 0)	(0, 1)	–
(1, 0)	(1, 0)	(1, 1)	–
(1, 1)*	(1, 0)	(1, 1)	–

(Před # jsou jen stavy 0, 1; po # jen dvojice. Vlevo ve dvojici $(i, \cdot)$ zůstává $h(u)$, vpravo se aktualizuje $h(v')$.)

Stavový diagram A_2 (oříznutý z originálu) je v příklady/.

2 Bezkontextové jazyky (gramatiky, zásobníkový automat)

Bezkontextové jazyky popisují vnořené, „závorkové“ struktury (aritmetické výrazy, syntaxe programovacích jazyků), které regulární jazyky neumějí. Popíšeme je *bezkontextovou gramatikou* a ekvivalentním *zásobníkovým automatem*; zásobník dává automatu neomezenou (ale jen zásobníkově přístupnou) paměť.

2.1 Bezkontextová gramatika (CFG)

Definice – Bezkontextová gramatika

Bezkontextová gramatika (CFG) je čtveřice $G = (V, T, \mathcal{P}, S)$, kde V je konečná množina *neterminálů* (proměnných), T konečná neprázdná množina *terminálů* ($V \cap T = \emptyset$), $S \in V$ *počáteční symbol* a $\mathcal{P}$ konečná množina *pravidel* tvaru

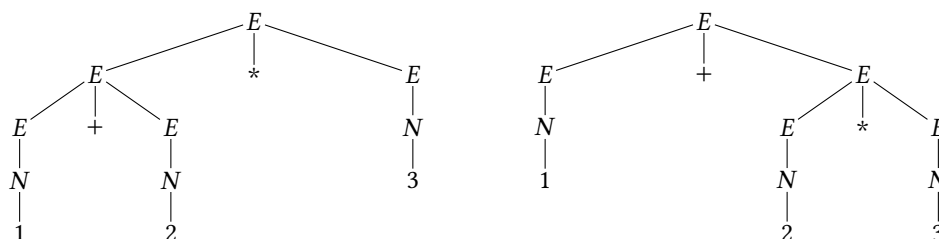
$$A \rightarrow \omega, \quad A \in V, \omega \in (V \cup T)^*.$$

Píšeme $\alpha \Rightarrow \beta$, lze-li α přepsat na β jedním pravidlem (nahrazením jednoho neterminálu); $\Rightarrow^*$ je její reflexivně-transitivní uzávěr (*derivace*). *Jazyk generovaný gramatikou* je $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.

Intuice. „Bezkontextová“ znamená, že neterminál A přepíšeme *bez ohledu na okolí* – levá strana pravidla je vždy jen A . Derivaci lze proto zachytit *derivačním (syntaktickým) stromem*: kořen je S , vnitřní uzly jsou neterminály, děti uzlu A odpovídají pravé straně použitého pravidla $A \rightarrow \omega$, listy zleva doprava dají vygenerované slovo.

Definice – Víceznačnost

Gramatika je *víceznačná* (nejednoznačná), existuje-li slovo s *dvěma různými* derivačními stromy. Některé bezkontextové jazyky jsou *podstatně víceznačné* – nemají žádnou jednoznačnou gramatiku.



dva derivační stromy slova $1+2*3$ pro víceznačnou gramatiku $E \rightarrow E+E \mid E * E \mid N, N \rightarrow 1 \mid 2 \mid 3$. Vlevo strom čte výraz jako $(1+2)*3$, vpravo jako $1+(2*3)$ – dvě *různá* ohodnocení téhož slova. Odstranění víceznačnosti (priorita, asociativita) vyžaduje pomocné neterminály, např. $E \rightarrow E+T \mid T, T \rightarrow T*N \mid N$.

2.2 Chomského normální forma

Pro mnoho úvah (i pro rozhodování příslušnosti slova) se hodí gramatika s co nejjednoduššími pravidly.

Definice – Chomského normální forma

CFG je v *Chomského normální formě* (ChNF), obsahuje-li jen pravidla tvaru $A \rightarrow BC$ a $A \rightarrow a$ (kde $A, B, C \in V$, $a \in T$), případně $S \rightarrow \lambda$ – a pak se S nevyskytuje na pravé straně žádného pravidla.

★ Věta – Převod do ChNF

Ke každé bezkontextové gramatice existuje ekvivalentní gramatika v Chomského normální formě.

Intuice. Převod je posloupnost úprav, z nichž každá odstraní jeden zakázaný typ pravidla a zachová jazyk: (1) odstíníme S z pravých stran; (2) terminály v delších pravidlech nahradíme novými neterminály $T_a \rightarrow a$; (3) dlouhé pravé strany rozdělíme na dvojice novými neterminály; (4) odstraníme λ -pravidla (přes *nulovatelné* neterminály, z nichž lze odvodit λ) a (5) jednotková pravidla $A \rightarrow B$ (přes jejich tranzitivní uzávěr).

Poznámka. ChNF dává rozhodovací algoritmus pro příslušnost slova: **CYK** (dynamické programování přes podslova) rozhodne $w \in L(G)$ v čase $O(n^3)$. Bezkontextové jazyky mají také *iterační lemma* (rozklad $w = uvxyz$ s $|vxy| \leq n$, $|vy| > 0$ a $uv^kxy^kz \in L$), kterým se dokazuje, že např. $\{a^n b^n c^n\}$ není bezkontextový. Třída je uzavřená na sjednocení, zřetězení, iteraci a otočení, *ne* na průnik a doplněk.

2.3 Zásobníkový automat (PDA)

Definice – Zásobníkový automat

Zásobníkový automat (PDA) je sedmice $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- Q konečná množina stavů, Σ vstupní abeceda, Γ zásobníková abeceda,
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \mathcal{P}_{\text{FIN}}(Q \times \Gamma^*)$ přechodová funkce,
- $q_0 \in Q$ počáteční stav, $Z_0 \in \Gamma$ počáteční zásobníkový symbol, $F \subseteq Q$ přijímající stavy.

Přechod $(q, \gamma) \in \delta(p, a, X)$ znamená: ve stavu p přečti a (nebo nečti nic, je-li $a = \lambda$), vrchol zásobníku X nahraď řetězcem γ a přejdi do q . (Γ^* jsou všechny konečné řetězce nad zásobníkovou abecedou, tj. možný nový vrchol; index FIN žádá *konečnou* množinu přechodů z dané konfigurace.)

Definice – Situace, krok, přijímaný jazyk

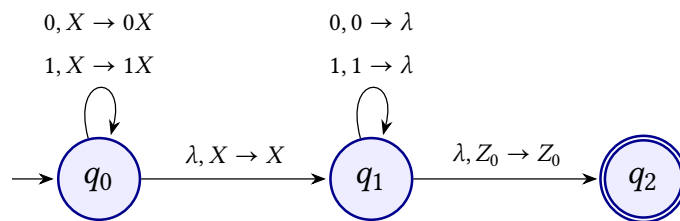
Situace (ID) je trojice (q, w, γ) : stav, zbývající vstup, obsah zásobníku (vrchol vlevo). Krok $\vdash$: je-li $(q, \alpha) \in \delta(p, a, X)$, pak $(p, aw, X\beta) \vdash (q, w, \alpha\beta)$; $\vdash^*$ je jeho uzávěr. Pro PDA definujeme dva módy přijímání:

$$L(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \lambda, \alpha), q \in F\} \quad (\text{koncovým stavem}),$$

$$N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \lambda, \lambda)\} \quad (\text{prázdným zásobníkem}).$$

Oba módy přijímají tutéž třídu (lze mezi nimi převádět); při přijímání prázdným zásobníkem je F nepodstatné a vynechává se.

Intuice. PDA je λ -NFA, ke kterému jsme přidali *zásobník*: nedeterminismus „hádá“ (např. kde je střed slova), zásobník pamatuje neomezeně mnoho informace, ale jen z vrcholu (LIFO). Právě tato omezená paměť dělá z PDA přesný protějšek bezkontextových gramatik. **Deterministické** PDA jsou slabší – přijímají jen vlastní podtřídu bezkontextových jazyků.



PDA přijímající $L_{ww^R} = \{ww^R \mid w \in \{0,1\}^*\}$ koncovým stavem. Hrana $a, X \rightarrow \alpha$ znamená $(q, \alpha) \in \delta(p, a, X)$; X je libovolný vrchol ($X \in \{0, 1, Z_0\}$). V q_0 automat ukládá čtené znaky na zásobník; λ -přechodem do q_1 uhodne střed; v q_1 odmazává shodné znaky z vrcholu; vidí-li Z_0 (zásobník vyčerpán až na dno), λ -přechodem přijme. Nedeterminismus zajistí, že se uhodne správný střed pro každé ww^R .

📌 Příklad – SZZ podzim 2023 č. 2: Zásobníkový automat

(Reálná otázka SZZ: podzim 2023, č. 2 – Zásobníkový automat (definice + konstrukce PDA))

Zadání.

(1) Uveďte definici PDA. (2) Sestrojte PDA A přijímající právě $w \in \{a, b\}^*$ s více a než b a zároveň lichým počtem a : $L(A) = \{w \mid |w|_a > |w|_b \text{ \& } |w|_a \text{ liché}\}$.

Řešení.

(1) Sedmice $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ jako v definici výše.

(2) Na zásobníku udržujeme *přebytek*: rozdíl $|w|_a - |w|_b$. Je-li kladný, leží na zásobníku tolik symbolů A (nad Z_0); je-li záporný, tolik B . Čtené a buď odmaže B (přebytek b klesá), nebo přidá A ; čtené b symetricky. *Paritu* počtu a držíme ve stavu: q_0 (sudě), q_1 (liše). Stavů $\{q_0, q_1, q_f\}$, $\Gamma = \{Z_0, A, B\}$, počátek q_0 , $F = \{q_f\}$. Přechody (čtení a překlápí $q_0 \leftrightarrow q_1$):

$$\begin{aligned} \delta(q_i, a, Z_0) &= \{(q_{1-i}, AZ_0)\}, \quad \delta(q_i, a, A) = \{(q_{1-i}, AA)\}, \quad \delta(q_i, a, B) = \{(q_{1-i}, \lambda)\}, \\ \delta(q_i, b, Z_0) &= \{(q_i, BZ_0)\}, \quad \delta(q_i, b, B) = \{(q_i, BB)\}, \quad \delta(q_i, b, A) = \{(q_i, \lambda)\} \quad (i \in \{0, 1\}), \\ \delta(q_1, \lambda, A) &= \{(q_f, A)\} \quad (\text{přijmi: lichý počet } a \text{ a vrchol } A, \text{ tj. přebytek } a). \end{aligned}$$

Přijímáme *koncovým stavem*: do q_f se dostaneme jen z q_1 (lichý počet a) a jen je-li na vrcholu A , což nastane právě když $|w|_a > |w|_b$. Správnost ověřena vyčerpáním slov do délky 11 (Python). (Originál uvádí jinou (rovněž 3-stavovou) variantu, její diagram je v příklady/; za částečné řešení se uznával i automat jen pro „více a než b “.)

2.4 Ekvivalence CFG $\leftrightarrow$ PDA a převody

★ Věta – CFG = PDA

Jazyk je bezkontextový (generovaný CFG) právě tehdy, je-li přijímaný nějakým (nedeterministickým) zásobníkovým automatem.

💡 **Intuice.** Standardní směr $\text{CFG} \rightarrow \text{PDA}$ dá *jednostavový* automat přijímající *prázdným zásobníkem*, který na zásobníku simuluje *levou derivaci* (v každém kroku se přepisuje nejlevější neterminál): zásobník drží dosud neodvozený zbytek věty. Pro $P = (\{q_0\}, T, V \cup T, \delta, q_0, S)$:

$$\delta(q_0, \lambda, A) = \{(q_0, \omega) \mid (A \rightarrow \omega) \in \mathcal{P}\} \text{ (rozepiš neterminál)}, \quad \delta(q_0, a, a) = \{(q_0, \lambda)\} \text{ (odečti terminál)}.$$

Automat tak střídá „rozepsání neterminálu na vrcholu“ a „spárování terminálu se vstupem“; přijme prázdným zásobníkem právě slova generovaná G .

Příklad – SZZ léto 2023 č. 2: Převod gramatiky na automat

(Reálná otázka SZZ: léto 2023, č. 2 – Převod bezkontextové gramatiky na automat (CFG → zásobníkový automat))

Zadání.

(1) Definice CFG. (2) Sestrojte CFG G pro $L = \{a^i b^j \mid i, j \geq 0, 2i = 3j\}$. (3) Převedte G na PDA přijímající prázdným zásobníkem.

Řešení.

(1) Čtveřice $G = (V, T, \mathcal{P}, S)$ jako v definici výše.

(2) Z $2i = 3j$ plyne $i = 3k, j = 2k$ pro $k \geq 0$, tj. $L = \{a^{3k} b^{2k} \mid k \geq 0\}$. Generuje ho

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aaaSbb \mid \lambda\}, S).$$

(Ověřeno vyčerpáním do délky 12: $L(G)$ se přesně shoduje s $\{a^i b^j : 2i = 3j\}$.)

(3) Jednostavový PDA $P = (\{q_0\}, \{a, b\}, \{a, b, S\}, \delta, q_0, S)$ přijímající prázdným zásobníkem:

$$\delta(q_0, \lambda, S) = \{(q_0, aaaSbb), (q_0, \lambda)\}, \quad \delta(q_0, a, a) = \{(q_0, \lambda)\}, \quad \delta(q_0, b, b) = \{(q_0, \lambda)\}.$$

První skupina rozepisuje S podle pravidel, druhá páruje terminály se vstupem.

(V originálním PDF tato otázka nemá náčrt řešení; řešení je vlastní, ověřené numericky.)

Stejnou „repräsentaci dvěma způsoby“ (gramatika i automat) žádá i následující otázka.

Příklad – SZZ jaro 2024 č. 1: Repräsentace bezkontextového jazyka

(Reálná otázka SZZ: jaro 2024, č. 1 – Repräsentace bezkontextového jazyka (CFG, zásobníkový automat))

Zadání.

Pro $L = \{w\#s^R \mid w, s \in \{0, 1\}^*, s \text{ je podslovo } w\}$ nad $\{0, 1, \#\}$: (1) definice CFG a PDA, (2) CFG pro L , (3) PDA pro L .

Řešení.

(1) Definice CFG a PDA jsou výše.

(2) Slovo má tvar $w\#s^R$, kde $w = p u q$ a $s = u$ je souvislé podslovo. Necháme A generovat libovolné slovo nad $\{0, 1\}$ (prefix p a suffix q), a párová pravidla X budují u vlevo i u^R vpravo kolem $\#$:

$$G = (\{S, X, A\}, \{0, 1, \#\}, \mathcal{P}, S), \quad \mathcal{P} = \{S \rightarrow AX, X \rightarrow 0X0 \mid 1X1 \mid A\#, A \rightarrow 0A \mid 1A \mid \lambda\}.$$

($S \Rightarrow AX \Rightarrow p(u A \# u^R)$, tedy $w = p u q$ a za $\#$ stojí u^R .) Ověřeno vyčerpáním do délky 9.

(3) Standardním převodem G na jednostavový PDA $(\{q_0\}, \{0, 1, \#\}, \{0, 1, \#, S, X, A\}, \delta, q_0, S)$ přijímající prázdným zásobníkem, kde přechody odpovídají pravidlům a čtení písmen:

$$\delta = \{((q_0, \lambda, H), \beta) \mid H \rightarrow \beta \in \mathcal{P}\} \cup \{((q_0, a, a), \lambda) \mid a \in \{0, 1, \#\}\}.$$

(Alternativně přímo: čteme w a ukládáme znaky na zásobník; po $\#$ nejprve λ -přechody smažeme libovolný počet symbolů (odhad konce u), pak párujeme s^R se zásobníkem a nakonec ho vyprázdníme.)

Poslední otázka kombinuje bezkontextový a regulární jazyk. *Průnik* bezkontextového a regulárního jazyka je opět bezkontextový – získáme ho *součinem* PDA a DFA (stavy nesou dvojici, zásobník zůstává jeden).

Příklad – SZZ jaro 2025 č. 1: Průnik bezkontextového a regulárního jazyka

(Reálná otázka SZZ: jaro 2025, č. 1 – Průnik bezkontextového a regulárního jazyka (PDA, $\text{CFG} \cap \text{DFA}$))

Zadání.

$L_1 = L(G)$ pro $G = (\{S\}, \{0, 1\}, \{S \rightarrow SS \mid 0S1 \mid \lambda\}, S)$ (vyvážené závorky: 0 otevírá, 1 zavírá), $L_2 = L(A)$ pro daný DFA A se stavy $\{q_0, q_1, q_2, q_3\}$, který přijímá ve všech stavech ($F = \{q_0, q_1, q_2, q_3\}$) a má přechody $\delta_A(q_0, 1) = q_0$, $\delta_A(q_0, 0) = q_1$, $\delta_A(q_1, 0) = q_1$, $\delta_A(q_1, 1) = q_2$, $\delta_A(q_2, 0) = q_3$, $\delta_A(q_2, 1) = q_0$, $\delta_A(q_3, 0) = q_1$; přechod $\delta_A(q_3, 1)$ chybí. Sestrojte PDA pro $L = L_1 \cap L_2$.

Řešení.

Použijeme *součinovou konstrukci*: přijímání DFA A simulujeme stavem, generování G zásobníkem (jako u převodu CFG na jednostavový PDA). Výsledný PDA $P = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, S\}, \delta_P, q_0, S)$ přijímá prázdným zásobníkem:

$$\delta_P(q_i, a, a) = \{(\delta_A(q_i, a), \lambda)\} \quad a \in \{0, 1\} \text{ (čti terminál, posuň DFA),}$$

$$\delta_P(q_i, \lambda, S) = \{(q_i, SS), (q_i, 0S1), (q_i, \lambda)\} \quad \text{(krok derivace } G, \text{ stav DFA beze změny).}$$

PDA vyprázdní zásobník právě tehdy, když $S \Rightarrow^* w$, tj. $w \in L_1$; jeho stavová složka přitom sleduje $\delta_A^*(q_0, w)$. Tato součinná konstrukce dokazuje, že *průnik bezkontextového a regulárního jazyka je bezkontextový*. Je *generická* v δ_A : konkrétní přechody DFA do ní vstupují jen symbolicky.

Proč stačí přijímat prázdným zásobníkem. Stav q_3 znamená „slovo zatím končí na 010“ a chybí mu přechod na 1, takže A na slově s podslovem 0101 uvízne. Protože jsou všechny stavy přijímající, je $L_2 = \{w \mid w \text{ neobsahuje } 0101\}$ (ne celé $\{0, 1\}^*$) a L tvoří vyvážená slova bez podslova 0101; např. $0011 \in L$, ale $0101 \in L_1 \setminus L_2$. Chybějící přechod DFA se v P projeví tak, že P nemůže přečíst další znak a výpočet skončí neúspěchem; jinak je každý dosažený stav DFA přijímající, takže kontrola koncového stavu není potřeba. (Obecně, kdyby $F_A \neq Q_A$, vložíme pod S nové dno Z_0 a přidáme přechody $\delta_P(q, \lambda, Z_0) = \{(q, \lambda)\}$ jen pro $q \in F_A$.) Správnost ověřena vyčerpáním slov do délky 12 (Python).

Stavový diagram DFA A je v příklady/.

3 Rekurzivně spočetné jazyky (Turingův stroj, nerozhodnutelnost)

Nejširší třída Chomského hierarchie odpovídá *nejobecnějšímu* modelu výpočtu – Turingovu stroji – a *nejobecnějším* gramatikám (typ 0). Zde už narazíme na hranici algoritmické řešitelnosti: existují přesně definované problémy, které *žádný* algoritmus nerozhodne.

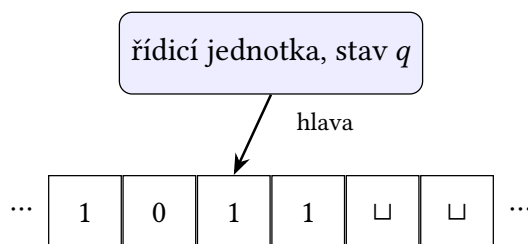
3.1 Turingův stroj

Definice – Turingův stroj

Turingův stroj (TM) je sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$, kde

- Q konečná množina stavů, $q_0 \in Q$ počáteční, $F \subseteq Q$ přijímající stavy,
- Σ vstupní abeceda, $\Gamma \supseteq \Sigma$ pásková abeceda, $\sqcup \in \Gamma \setminus \Sigma$ prázdný symbol (mezera),
- $\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ přechodová funkce (může být parciální).

Stroj má oboustranně nekonečnou pásku; v každém kroku přečte znak pod hlavou, podle δ jej přepíše, změní stav a posune hlavu doleva/doprava. *Konfigurace* popisuje stav, obsah pásky a polohu hlavy.



Turingův stroj. Konečná *řídící jednotka* (v jednom z konečně mnoha stavů) ovládá *hlavu* pohybující se po oboustranně nekonečné pásce. Na rozdíl od konečného automatu smí znak pod hlavou *přepsat* a pohybovat se *oběma směry* – to dává neomezenou pracovní paměť. Na počátku je na pásce vstup, jinde mezery $\sqcup$; hlava stojí na prvním znaku, jednotka ve stavu q_0 .

Definice – Přijímaný a rozhodovaný jazyk; R a RE

Stroj M *přijme* slovo w , dojde-li jeho výpočet do přijímacího stavu (jinak slovo odmítne, nebo se nezastaví). *Jazyk přijímaný* M je $L(M) = \{w \mid M \text{ přijme } w\}$.

- Jazyk je *rekurzivně spočetný* (částečně rozhodnutelný), třída RE, je-li $L = L(M)$ pro nějaký TM (na slovech mimo L se stroj smí *zacyklit*).
- Jazyk je *rozhodnutelný* (rekurzivní), třída R, existuje-li TM, který se na *každém* vstupu zastaví a přijímá L .

 **Intuice.** Rozdíl R vs. RE je v garanci zastavení. *Rozhodující* stroj vždy odpoví ano/ne. *Přijímací* stroj řekne „ano“ na slovech jazyka, ale na slovech mimo jazyk se může točit donekonečna – nikdy se nedozvíme „ne“. Platí $R \subseteq RE$ (rozhodující stroj je speciálním případem přijímacího); uvidíme, že inkluze je *ostrá*.

3.2 Gramatiky typu 0

Definice – Gramatika typu 0

Gramatika typu 0 (obecná) má pravidla $\alpha \rightarrow \beta$, kde $\alpha, \beta \in (V \cup T)^*$ a α obsahuje aspoň jeden neterminál (levá strana smí být libovolné slovo, ne jen jediný neterminál jako u CFG).

★ Věta – Typ 0 = RE

Třída jazyků generovaných gramatikami typu 0 je přesně třída rekurzivně spočetných jazyků: $\mathcal{L}_0 = RE$.

 **Intuice.** Gramatika typu 0 a Turingův stroj jsou dvě strany téže mince. Gramatikou lze *simulovat* výpočet stroje (pravidla přepisují konfiguraci na následující), a naopak stroj umí *vyjmenovávat* všechna slova odvoditelná gramatikou a porovnávat je se vstupem. Proto „generovatelné gramatikou“ = „přijímané strojem“ = rekurzivně spočetné.

3.3 Algoritmicky nerozhodnutelné problémy

Že některé jazyky nejsou rozhodnutelné (ba ani rekurzivně spočetné), plyne z *diagonálního* argumentu. Turingovy stroje (nad $\{0, 1\}$) lze *zakódovat* do slov; slovo α kóduje stroj M_α .

Definice – Univerzální a diagonální jazyk

Univerzální jazyk $L_u = \{\langle \alpha, \beta \rangle \mid \beta \in L(M_\alpha)\}$ obsahuje dvojice „kód stroje, vstup“, kde stroj vstup přijme. *Diagonální jazyk* $L_d = \{\alpha \mid \alpha \notin L(M_\alpha)\}$ obsahuje kódy strojů, které *nepřijmou samy sebe*.

★ Věta – Nerozhodnutelnost

L_u je rekurzivně spočetný, ale *není* rozhodnutelný. L_d není ani rekurzivně spočetný. Inkluze $R \subsetneq RE$ je tedy *ostrá*.

Důkaz (nepovinný – diagonalizace). L_u přijímá *univerzální stroj* (UTM), který odsimuluje M_α na β –

proto $L_u \in \text{RE}$. Kdyby byl L_d rekurzivně spočetný, přijímal by ho nějaký M_γ , tedy $L_d = L(M_\gamma)$. Pak ale

$$\gamma \in L_d \iff \gamma \notin L(M_\gamma) \iff \gamma \notin L_d,$$

spor. Takže $L_d \notin \text{RE}$. Odtud $L_u \notin \text{R}$: platí $\alpha \in L_d \iff \langle \alpha, \alpha \rangle \notin L_u$, takže kdyby byl L_u rozhodnutelný, rozhodli bychom i L_d (spustit rozhodovací stroj pro L_u na $\langle \alpha, \alpha \rangle$ a odpověď obrátit) – spor. $\square$

i Poznámka. [Problém zastavení a Postova věta] Týž argument dává *problém zastavení* (halting problem): neexistuje algoritmus, který pro libovolný program a vstup rozhodne, zda se výpočet zastaví. Užitečné kritérium dává *Postova věta*: jazyk L je rozhodnutelný právě tehdy, když L i jeho doplněk $\bar{L}$ jsou rekurzivně spočetné. (Proto L_u , jehož doplněk není RE, rozhodnutelný být nemůže.)

4 Chomského hierarchie a zařazování jazyků

Předchozí tři sekce probraly tři ze čtyř pater *Chomského hierarchie*. Teď je uspořádáme do jednoho obrázku a doplníme chybějící patro – *kontextové* (typ 1) jazyky – a ukážeme, jak *zařadit konkrétní jazyk*.

4.1 Klasifikace gramatik a hierarchie

Definice – Chomského hierarchie

Gramatiky (a jimi generované třídy jazyků) klasifikujeme podle tvaru pravidel:

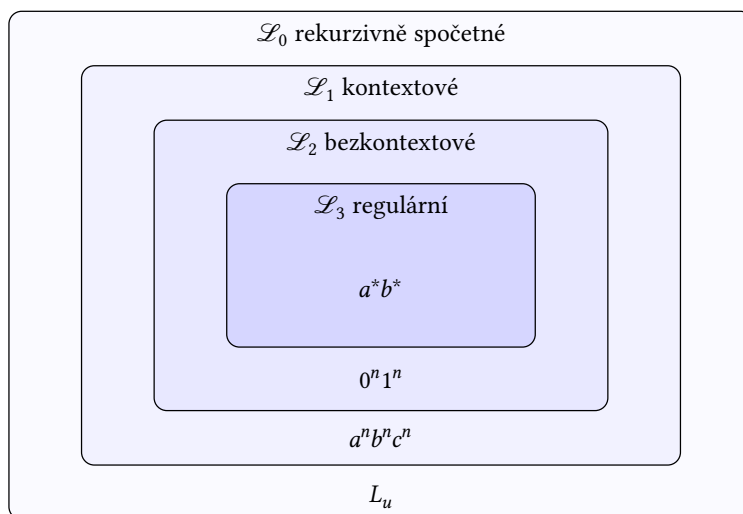
- **typ 0** (rekurzivně spočetné $\mathcal{L}_0$): $\alpha \rightarrow \beta$, α obsahuje neterminál;
- **typ 1** (kontextové $\mathcal{L}_1$): $\gamma A \beta \rightarrow \gamma \omega \beta$ s $\omega \neq \lambda$ (přepis A v kontextu γ, β ; výjimkou je $S \rightarrow \lambda$, pak se S nevyskytuje vpravo). Ekvivalentně *monotónní* (nezkracující) pravidla $|\alpha| \leq |\beta|$;
- **typ 2** (bezkontextové $\mathcal{L}_2$): $A \rightarrow \omega$;
- **typ 3** (regulární $\mathcal{L}_3$): $A \rightarrow wB$, $A \rightarrow w$.

★ Věta – Uspořádání hierarchie

Třídy tvoří ostrou hierarchii

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0,$$

a navíc $\mathcal{L}_2 \subsetneq \text{R}$, $\mathcal{L}_1 \subsetneq \text{R} \subsetneq \text{RE} = \mathcal{L}_0$. Každému patru odpovídá typ automatu.



Chomského hierarchie. Každá třída je vlastní podmnožinou třídy o patro výš (vnořené obdélníky). Příklady svědčící o ostrosti: $0^n 1^n$ je bezkontextový, ne regulární; $a^n b^n c^n$ je kontextový, ne bezkontextový; univerzální jazyk L_u je rekurzivně spočetný, ne rozhodnutelný (a R leží mezi $\mathcal{L}_1$ a $\mathcal{L}_0$). Soulad automatů, gramatik a jazyků shrnuje tabulka níže.

typ	gramatika	automat	uzávěr na operace
3	pravá lineární	konečný automat (DFA/NFA)	vše ($\cup, \cap, \neg, \cdot, *, ^R$)
2	bezkontextová	zásobníkový automat (PDA)	$\cup, \cdot, *, ^R$ (ne $\cap$, ne $\neg$)
1	kontextová (monotónní)	lineárně omezený automat (LBA)	vše
0	obecná (typ 0)	Turingův stroj	$\cup, \cap, \cdot, *, ^R$ (ne $\neg$)

Lineárně omezený automat (LBA) je Turingův stroj, který smí použít jen pásku délky vstupu (ohraničenou zarážkami).

4.2 Zařazení konkrétního jazyka

Požadavky okruhu zahrnují *zařazení konkrétního jazyka do hierarchie* se zdůvodněním. Postup má dvě strany:

💡 Intuice. Horní zařazení (do třídy patří): sestroj automat nebo gramatiku příslušného typu – tím dokážeš, že jazyk *do třídy patří*. **Dolní mez (do nižší třídy nepatří):** použij *iterační lemma* (regulární resp. bezkontextové) a najdi slovo, které napumpováním vypadne z jazyka – tím dokážeš, že jazyk *do nižší třídy nepatří*.

Příklad – Zařazení tří typických jazyků

Zadání. Zařadte do Chomského hierarchie: (a) $\{a^n b^n \mid n \geq 0\}$, (b) $\{a^n b^n c^n \mid n \geq 0\}$, (c) $\{ww \mid w \in \{0, 1\}^*\}$.

Řešení.

(a) *Bezkontextový, ne regulární* ($\mathcal{L}_2 \setminus \mathcal{L}_3$). Gramatika $S \rightarrow aSb \mid \lambda$ jej generuje (horní zařazení). Neregularitu dá iterační lemma: pro $w = a^n b^n$ leží napumpovaná část y v úvodních a , takže $a^{n+|y|} b^n \notin L$.

(b) *Kontextový, ne bezkontextový* ($\mathcal{L}_1 \setminus \mathcal{L}_2$). Bezkontextové iterační lemma na $a^n b^n c^n$ ukáže, že napumpování rozbije rovnost počtů (úsek vxy délky $\leq n$ nemůže obsahovat všechna tři písmena). Patří ale do $\mathcal{L}_1$, generuje ho monotónní gramatika

$$S_0 \rightarrow S \mid \lambda, \quad S \rightarrow aSBC \mid aBC, \quad CB \rightarrow BC, \quad aB \rightarrow ab, \quad bB \rightarrow bb, \quad bC \rightarrow bc, \quad cC \rightarrow cc$$

(pravidlo $CB \rightarrow BC$ přesune všechna B před C , zbylá pravidla je pak zleva doprava přepíšou na terminály; ověřeno vyčerpáním do délky 12).

(c) *Kontextový, ne bezkontextový* ($\mathcal{L}_1 \setminus \mathcal{L}_2$). „Kopírovací“ jazyk $\{ww\}$ rovněž odmítne bezkontextové iterační lemma (slovo $0^n 1^n 0^n 1^n$). Do $\mathcal{L}_1$ patří, protože ho rozhodne lineárně omezený automat: ověří sudou délku, označováním znaků z obou konců najde střed a obě poloviny porovná znak po znaku, vše jen v prostoru vstupu. (Pozor: $\{ww^R\}$ – palindromy sudé délky – naopak *bezkontextový* je (viz zásobníkový automat výše); rozdíl je v pořadí druhé kopie.)

Shrnutí

📌 Poznámka. Klíčové vlastnosti a vzorce (přehled k SZZ). Stručný výčet klíčových definic, vět a vzorců okruhu I1. Zkratky: DFA = deterministický, NFA = nedeterministický konečný automat, CFG = bezkontextová gramatika, PDA = zásobníkový automat, TM = Turingův stroj, CFL = bezkontextový jazyk.

Regulární jazyky – modely a ekvivalence.

- **DFA** ($(Q, \Sigma, \delta, q_0, F)$): $\delta : Q \times \Sigma \rightarrow Q$ totální (nebo parciální s implicitním mrtvým stavem); rozšíření $\delta^* : Q \times \Sigma^* \rightarrow Q$ indukci – $\delta^*(q, \lambda) = q$, $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$; jazyk $L(A) = \{w \mid \delta^*(q_0, w) \in F\}$.

- **NFA** $(Q, \Sigma, \delta, Q_0, F)$: $Q_0 \subseteq Q$ množina počátečních stavů, $\delta : Q \times \Sigma \rightarrow 2^Q$; přijme, *existuje-li* výpočet do přijímajícího stavu. (λ -NFA navíc dovoluje přechody na λ ; ekvivalentní NFA.)
- **Ekvivalence čtyř modelů**: DFA = NFA (λ -NFA) = regulární výrazy = pravé lineární gramatiky (typ 3); vše popisuje třídu $\mathcal{L}_3$.
- **Regulární výraz** nad Σ : základ $\emptyset$, λ , $a \in \Sigma$; operace $+$ (sjednocení), $\cdot$ (zřetězení), $*$ (iterace); značení $R^+ \equiv RR^*$.
- **Pravá lineární gramatika** (typ 3): pravidla tvaru $A \rightarrow wB$ nebo $A \rightarrow w$; neterminál = stav (přechod $A \xrightarrow{a} B$ odpovídá pravidlu $A \rightarrow aB$).

Klíčové převody u regulárních jazyků.

- **Podmnožinová konstrukce** (NFA $\rightarrow$ DFA): stavy DFA jsou podmnožiny stavů NFA; $\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a)$; počáteční stav Q_0 (množina počátečních stavů NFA); přijímající stavy – podmnožiny obsahující stav NFA z F . Může exponenciálně nafouknout stavový prostor (2^n stavů v nejhorším případě).
- **Kleeneho věta**: L je regulární $\iff L$ je popsateľný regulárním výrazem; výraz $\rightarrow \lambda$ -NFA Kleeneho stavebnicí (indukcí podle výrazu), automat $\rightarrow$ výraz eliminací stavů.
- **Doplňek** $\bar{L}$: totalizuj DFA (přidej mrtvý stav pro chybějící přechody), zaměň přijímající a nepřijímající stavy; výsledek přijímá právě $\bar{L}$. (Jen pro DFA – NFA neumí doplněk přímým záměněním stavů.)
- **Součin DFA** (průnik): pro $L_1 \cap L_2$ sestav DFA se stavy $Q_1 \times Q_2$, $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$, přijímající stavy $F_1 \times F_2$.

Iterační lemma pro regulární jazyky (důkaz neregularity).

- Pro každý regulární jazyk L existuje $n \geq 1$: každé $w \in L$ s $|w| \geq n$ lze rozložit $w = xyz$ tak, že $y \neq \lambda$, $|xy| \leq n$, a $\forall k \geq 0 : xy^kz \in L$.
- **Použití (dolní mez)**: zvolíme $w \in L$ s $|w| \geq n$, ukážeme, že pro *každé* rozložení xyz ($y \neq \lambda$, $|xy| \leq n$) existuje k takové, že $xy^kz \notin L$ – spor s lemmatem.
- Typické slovo: $a^n b^n$ (část y leží v úvodních a , napumpování poruší rovnost počtů).

Uzávěrové vlastnosti regulárních jazyků.

- Uzavřeno na: $\cup$, $\cap$, $-$ (doplňek), $\cdot$ (zřetězení), $*$, ^{R} (otočení).

Bezkontextové jazyky – CFG a normální formy.

- **CFG** $(V, T, \mathcal{P}, S)$: V neterminály, T terminály, $S \in V$ startovní symbol, $\mathcal{P}$ pravidla $A \rightarrow \omega$ ($A \in V$, $\omega \in (V \cup T)^*$). Jazyk $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$.
- Derivace $\Rightarrow$, $\Rightarrow^*$; **derivační strom** (parse tree); **víceznačnost** – slovo má ≥ 2 různé derivační stromy; nejednoznačný jazyk – každá CFG jej generuje víceznačně.
- **Chomského normální forma (ChNF)**: pravidla jen $A \rightarrow BC$ nebo $A \rightarrow a$ (povolena $S \rightarrow \lambda$ je-li $\lambda \in L$); každá CFG lze převést do ChNF.
- **CYK** (Cocke–Younger–Kasami): dynamické programování na trojúhelníkové tabulce; $O(n^3)$; rozhoduje $w \in L(G)$ pro CFG v ChNF.

Zásobníkový automat (PDA) – definice a módy přijímání.

- **PDA** je sedmice $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$: Γ zásobníková abeceda, $Z_0 \in \Gamma$ počáteční zásobníkový symbol, $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \mathcal{P}_{\text{FIN}}(Q \times \Gamma^*)$.
- **Situace** (q, w, γ) : aktuální stav, nezpracovaný vstup, obsah zásobníku (vrchol vlevo).
- **Přijímání koncovým stavem** $L(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q_f, \lambda, \gamma), q_f \in F\}$; **prázdným zásobníkem** $N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \lambda, \lambda)\}$. Obě třídy jsou stejné: $L(P) = N(P')$ pro vhodné P' a naopak.
- PDA je obecně nedeterministický; deterministické PDA (DPDA) přijímají striktní podtřídu CFL.

Ekvivalence CFG a PDA; průnik s regulárním jazykem.

- **CFG = PDA**: ke každé CFG existuje ekvivalentní PDA a naopak; třída CFL = třída jazyků přijímaných PDA.
- **CFG $\rightarrow$ jednostavový PDA** (přijímání prázdným zásobníkem): jeden stav q_0 ; pro každé pravidlo $A \rightarrow \alpha$: $\delta(q_0, \lambda, A) \ni (q_0, \alpha)$; pro každý terminál a : $\delta(q_0, a, a) \ni (q_0, \lambda)$. Simuluje levou derivaci.

- **Průnik $CFL \cap REG \subseteq CFL$:** součin $PDA \times DFA$ – stavy $Q_P \times Q_D$, čtení symbolu posunuje oba automaty souběžně; zásobník zůstává zásobníkem PDA. Konstruktivní důkaz a základ SZZ otázky jaro 2025.

Iterační lemma pro bezkontextové jazyky (důkaz, že jazyk není CFL).

- Pro každý CFL L existuje $n \geq 1$: každé $w \in L$ s $|w| \geq n$ lze rozložit $w = uvxyz$ tak, že $vy \neq \lambda$, $|vxy| \leq n$, a $\forall k \geq 0 : uv^kxy^kz \in L$.
- **Použití:** pumping na $a^n b^n c^n - |vxy| \leq n$ neumožňuje zahrnout všechna tři písmena; napumpování poruší rovnost počtů.
- Pozor: $\{ww^R\}$ (palindromy sudé délky) je bezkontextový – PDA uloží první polovinu na zásobník, střed nedeterministicky uhadne a druhou polovinu porovná se zásobníkem.

Uzávěrové vlastnosti bezkontextových jazyků.

- Uzavřeno na: $\cup, \cdot, *, ^R$; průnik s regulárním (viz výše).
- **Není** uzavřeno na: $\cap$ (průnik dvou CFL nemusí být CFL – protipříkladem je $\{a^n b^n c^n\} = \{a^n b^n c^m\} \cap \{a^m b^n c^n\}$, průnik dvou CFL, který podle iteračního lematu CFL není), $-$ (doplňek).

Rekurzivně spočetné jazyky a Turingův stroj.

- **TM** sedmice $(Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$: $\Sigma \subset \Gamma$ vstupní abeceda, $\sqcup \in \Gamma \setminus \Sigma$ blank; $\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ (parciální).
- **R** (rekurzivní = rozhodnutelné): TM se vždy zastaví (přijme, nebo se zastaví mimo F , tj. odmítne). **RE** (rekurzivně spočetné): TM přijme nebo cyklí. $R \subseteq RE$.
- Gramatika **typu 0** (neomezená) = RE: každý RE jazyk je generovatelný gramatikou bez omezení tvaru pravidel a naopak.

Nerozhodnutelnost.

- **Diagonální jazyk** $L_d = \{\gamma_i \mid \gamma_i \notin L(M_i)\} \notin RE$; důkaz diagonalizací.
- **Univerzální jazyk** $L_u = \{\langle M, w \rangle \mid M \text{ přijme } w\} \in RE \setminus R$; univerzální TM simuluje M na w – přijme, ale nemůže vždy odmítnout.
- **Problém zastavení** (halting): neexistuje algoritmus rozhodující pro libovolný program a vstup, zda výpočet skončí; nerozhodnutelnost plyne z nerozhodnutelnosti L_u .
- **Postova věta:** $L \in R \iff L \in RE \text{ a } \bar{L} \in RE$. (Tedy L_u není rozhodnutelný, protože $\bar{L}_u \notin RE$.)

Chomského hierarchie – přehled.

- Ostrá hierarchie: $\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$; odpovídající automaty: $FA \subsetneq PDA \subsetneq LBA \subsetneq TM$.
- Tabulka souladu: typ 3 (pravá lin. gram., FA, uzavřeno na vše), typ 2 (CFG, PDA, uzavřeno na $\cup, \cdot, ^R$, ne $\cap$), typ 1 (kontextová, LBA, uzavřeno na vše), typ 0 (obecná, TM, uzavřeno na $\cup \cap \cdot, ^R$, ne $-$).
- **Vzorové jazyky:** $a^*b^* \in \mathcal{L}_3$; $\{a^n b^n\} \in \mathcal{L}_2 \setminus \mathcal{L}_3$; $\{ww^R\} \in \mathcal{L}_2$ (PDA: push první polovina, pop druhá); $\{a^n b^n c^n\}, \{ww\} \in \mathcal{L}_1 \setminus \mathcal{L}_2$; $L_u \in RE \setminus R$; $L_d \notin RE$.
- **Zařazení jazyka:** horní mez – sestavit automat nebo gramatiku příslušného typu; dolní mez – aplikovat iterační lemma (regulární resp. bezkontextové) a najít slovo, které napumpováním vypadne z jazyka.

Zdroje

- M. Vomlelová: *Automaty a gramatiky*, slidy k přednášce NTIN071, MFF UK, verze 2021 (primární zdroj rozsahu a značení: DFA/NFA, regulární výrazy, gramatiky, zásobníkový automat, Turingův stroj, nerozhodnutelnost, Chomského hierarchie). Stránka přednášející a kopie slidů ve státnicových materiálech T. Slámy:
<https://ktiml.mff.cuni.cz/~marta/>
<https://slama.dev/assets/priprava-na-statnice-mff-uk/autogramy/slidy.pdf>
- M. Mareš: *Úvod do automatů*, studijní text, MFF UK (souvislý výklad regulárních a bezkontextových jazyků, Turingových strojů a nerozhodnutelnosti).
<https://mj.ucw.cz/vyuka/automaty/automaty.pdf>

- J. E. Hopcroft, R. Motwani, J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, 3. vyd., Addison-Wesley, 2006 (učebnice, ze které vychází přednáška).
- MFF UK: požadavky ke státní závěrečné zkoušce (Informatika, Bc.) a zadání otázek z minulých termínů.
<https://www.mff.cuni.cz/cs/studenti/bakalarske-studium/statni-zaverecne-zkousky/bakalarske-statni-zkousky-studijniho-programu-informatika>