Kapitola 1

Úvod

V přednášce se zaměříme hlavně na konečný popis obecně nekonečných množin řetězců symbolů dané množiny A. Prvkům množiny A budeme říkat písmena, řetězcům (konečným posloupnostem) písmen budeme říkat slova, množinám slov pak jazyky. Zaměříme se na dva základní typy popisu: rozpoznávání a generování. Pro rozpoznávání využijeme pojem automatu, pro generování pak pojem gramatiky.

Zhruba řečeno, automat postupně zpracovává vstupní slovo a na základě stavu, do kterého se dostane po přečtení celého slova, rozhodne, zda slovo bylo přijato (do jazyka patří) nebo nebylo přijato (do jazyka nepatří). S formální definicí automatu se setkáme v první části.

Naproti tomu gramatika je vlastně soupis pravidel, jak „vygenerovat“ všechna slova jazyka. Jako příklad jazyka, pro který máme gramatiku, která jej generuje, uveďme např. správně utvořené algebraické výrazy, nebo správně napsané programy v daném programovacím jazyce. Gramatikám (jako silnějšímu nástroji než je jen konečný automat) se věnujeme v druhé polovině přednášky.

Dříve než přistoupíme ke studiu konečných automatů, připomeneme základní pojmy týkající se jazyků.

1.1 Jazyky

1.1.1 Abeceda. Konečnou neprázdnou množinu Σ budeme nazývat *abecedou*. Prvky množiny Σ nazýváme symboly, písmeny apod.

1.1.2 Slovo nad abecedou. Pro danou abecedu Σ *slovo nad* Σ je libovolná konečná posloupnost prvků abecedy Σ. Tedy např. pro $\Sigma = \{a, b\}$ jsou aab, b, $bbaba$ slova nad Σ.

Prázdné slovo, značíme je ε, je posloupnost, která neobsahuje ani jeden symbol. □

1.1.3 Délka slova. Je dáno slovo nad abecedou Σ. *Délka slova* je rovna délce posloupnosti, tj. počtu symbolů, které se ve slově nacházejí. Délku slova u značíme $|u|$. □

Tedy, délka slova aab je rovna 3, délka slova b je 1, délka prázdného slova ε je 0.

V dalším textu používáme ještě následující značení: pro slovo u nad abecedou Σ a $c \in \Sigma$ symbol $|u|_c$ označuje počet výskytů symbolu c ve slově u. Tedy např. pro $u = aab$ je $|u|_a = 2$, $|u|_b = 1$.

1.1.4 Zřetězení slov. Je dána abeceda Σ. Pro dvě slova u, v nad abecedou Σ definujeme operaci *zřetězení* takto: Je-li $u = a_1 a_2 \dots a_n$ a $v = b_1 b_2 \dots b_k$, pak

$$u \cdot v = a_1 a_2 \dots a_n b_1 b_2 \dots b_k.$$

Často znak pro operaci zřetězení vynecháváme, píšeme tedy uv místo přesnějšího $u \cdot v$.

1.1.5 Tvrzení. Zřetězení slov je asociativní operace na množině všech slov nad danou abecedou. □

1.1.6 Poznámka. Platí, že každé slovo $u = a_1 a_2 \dots a_n$ nad abecedou Σ, vzniklo (postupným) zřetězením slov délky 1, tj.

$$u = a_1 \cdot a_2 \cdot \ldots \cdot a_n.$$

Proto se někdy místo slovo nad abecedou používá termín *řetězec*.

1.1.7 Množiny $\Sigma^\star$ a Σ^+. Označíme $\Sigma^\star$ množinu všech slov nad abecedou Σ. (Speciálně prázdné slovo patří do $\Sigma^\star$.) Množina $\Sigma^\star$ spolu s operací zřetězení tvoří monoid, jehož neutrálním prvkem je prázdné slovo ε.

Označíme Σ^+ množinu všech neprázdných slov nad abecedou Σ. (Tj. $\Sigma^+ = \Sigma^\star \setminus \{\varepsilon\}$.) Množina Σ^+ spolu s operací zřetězení tvoří pologrupu. □

1.1.8 Vlastnosti operace zřetězení na $\Sigma^\star$ a Σ^+.

1. Zřetězení slov není komutativní. Např. pro $u = aab$ a $v = b$ je $uv = aabb$, ale $vu = baab$.
2. Pro libovolná slova u a v nad stejnou abecedou platí:

$$|uv| = |u| + |v|.$$

1.1.9 Mocniny. Je-li u slovo nad abecedou Σ, pak definujeme mocniny slova u takto:

$$u^0 = \varepsilon, \qquad u^{i+1} = uu^i \text{ pro každé } i.$$

□

1.1.10 Podslovo, prefix, sufix. Je dáno slovo u. Řekneme, že slovo w je *podslovem* slova u, jestliže existují slova x, y taková, že

$$u = xwy.$$

Slova x, y mohou být i prázdná. Speciálně, slovo u je podslovem sebe sama; ano $u = \varepsilon u \varepsilon$.

Jestliže

$$u = wy,$$

řekneme, že slovo w je *prefix slova* u.

Jestliže

$$u = xw,$$

řekneme, že slovo w je *sufix slova* u.

Jestliže neprázdné slovo w, $w \neq u$, je podslovo (resp. prefix, resp. sufix) slova u, pak w se nazývá *vlastní podslovo, (resp. vlastní prefix, resp. vlastní sufix)* slova u. □

1.1.11 Jazyk nad abecedou. Je dána abeceda Σ. *Jazyk* L nad abecedou Σ je libovolná množina slov, tj. $L \subseteq \Sigma^\star$. □

1.1.12 Poznámka. Je-li Σ abeceda, pak množina všech slov $\Sigma^\star$ je spočetná. Jazyků, jako podmnožin spočetné množiny, je víc – nespočetně mnoho.

1.1.13 Délka slova. Je dáno slovo nad abecedou Σ. *Délka slova* je rovna délce posloupnosti, tj. počtu symbolů, které se ve slově nacházejí. Délku slova u značíme $|u|$. □

Tedy, délka slova aab je rovna 3, délka slova b je 1, délka prázdného slova ε je 0.

V dalším textu budeme používat ještě následující značení: pro slovo u nad abecedou Σ a $c \in \Sigma$ symbol $|u|_c$ je počet výskytů symbolu c ve slově u. Tedy např. pro $u = aab$ je $|u|_a = 2$, $|u|_b = 1$.

1.1.14 Zřetězení slov. Je dána abeceda Σ. Pro dvě slova u, v nad abecedou Σ definujeme operaci *zřetězení* takto: Je-li $u = a_1 a_2 \dots a_n$ a $v = b_1 b_2 \dots b_k$, pak

$$u \cdot v = a_1 a_2 \dots a_n b_1 b_2 \dots b_k.$$

Často znak pro operaci zřetězení vynecháváme; píšeme tedy uv místo přesnějšího $u \cdot v$.

1.1.15 Tvrzení. Zřetězení slov je asociativní operace na množině všech slov nad danou abecedou. □

1.1.16 Poznámka. Platí, že každé slovo $u = a_1 a_2 \dots a_n$ nad abecedou Σ, vzniklo (postupným) zřetězením slov délky 1, tj.

$$u = a_1 \cdot a_2 \cdot \ldots \cdot a_n.$$

Proto se někdy místo slovo nad abecedou používá termín *řetězec*.

1.1.17 Množiny $\Sigma^\star$ a Σ^+. Označíme $\Sigma^\star$ množinu všech slov nad abecedou Σ. (Speciálně prázdné slovo patří do $\Sigma^\star$.) Množina $\Sigma^\star$ spolu s operací zřetězení tvoří monoid, jehož neutrálním prvkem je prázdné slovo ε.

Označíme Σ^+ množinu všech neprázdných slov nad abecedou Σ. (Tj. $\Sigma^+ = \Sigma^\star \setminus \{\varepsilon\}$.) Množina Σ^+ spolu s operací zřetězení tvoří pologrupu. □

1.1.18 Vlastnosti operace zřetězení na $\Sigma^\star$ a Σ^+.

1. Zřetězení slov není komutativní. Např. pro $u = aab$ a $v = b$ je $uv = aabb$, ale $vu = baab$.
2. Pro libovolná slova u a v nad stejnou abecedou platí:

$$|uv| = |u| + |v|.$$

1.1.19 Mocniny. Je-li u slovo nad abecedou Σ, pak definujeme mocniny slova u takto:

$$u^0 = \varepsilon, \qquad u^{i+1} = u\,u^i \text{ pro každé } i.$$

□

1.1.20 Podslovo, prefix, sufix. Je dáno slovo u. Řekneme, že slovo w je *podslovem* slova u, jestliže existují slova x, y taková, že

$$u = xwy.$$

Slova x, y mohou být i prázdná. Speciálně, slovo u je podslovem sebe sama; ano $u = \varepsilon u \varepsilon$.

Jestliže

$$u = w\,y,$$

řekneme, že slovo w je *prefix slova* u.

Jestliže

$$u = x\,w,$$

řekneme, že slovo w je *sufix slova* u.

Jestliže neprázdné slovo w, $w \neq u$, je podslovo (resp. prefix, resp. sufix) slova u, pak w se nazývá *vlastní podslovo, (resp. vlastní prefix, resp. vlastní sufix)* slova u. □

1.1.21 Jazyk nad abecedou. Je dána abeceda Σ. *Jazyk* L nad abecedou Σ je libovolná množina slov, tj. $L \subseteq \Sigma^\star$. □

1.1.22 Poznámka. Je-li Σ abeceda, pak množina všech slov $\Sigma^\star$ je spočetná. Jazyků, jako podmnožin spočetné množiny, je víc – nespočetně mnoho.

Kapitola 2

Konečné automaty

2.1 Deterministické konečné automaty

Konečné automaty se používají v různých oborech. Jako příklady můžeme uvést překladače, dále se používají při zpracování přirozeného jazyka, při návrzích hardwaru, i dalších. M zde nejprve vysvětlíme konečný automat neformálně, poté ukážeme několik příkladů a teprve poté zavedeme formální definici konečného automatu.

2.1.1 Konečný automat je abstraktní zařízení, které si pamatuje svůj stav a reaguje na vnější podněty změnami svého stavu, přičemž všech možných stavů i všech možných podnětů je konečně mnoho. Některé druhy automatů navíc produkují výstup.

Ze všech zařízení, která jsou schopna si něco pamatovat (tj. u nichž budoucí činnost může být ovlivněna historií), jsou konečné automaty nejjednodušší možné.

Jedním z cílů teorie automatů je zjistit, jaké jsou hranice toho, co konečné automaty dovedou a co už naopak je mimo jejich možnosti.

2.1.2 Množina stavů. Konečný automat se v každém okamžiku nachází v přesně jednom *stavu*, který je prvkem konečné množiny stavů (obvykle značené písmenem Q). Podstatné je, že kromě svého stavu si konečný automat vůbec nic nepamatuje (jiný slovy: nepamatuje si, jak se do tohoto stavu dostal).

Budoucí činnost automatu, tedy to, jak bude automat reagovat na příští vstupní podněty, obecně závisí na dosavadní historii, tedy na tom, co se s automatem dělo v minulosti, ale informace o celé předchozí historii je v konečném automatu zkoncentrována do jediného stavu, ve kterém se teď automat nachází.

2.1.3 Vstupní abeceda. Podněty, na které konečný automat reaguje, formalizujeme jako tzv. vstupní symboly (též vstupní písmena), které jsou prvky tzv. vstupní abecedy (obvykle značené Σ).

Při praktickém použití může být vstupním symbolem např. znak přečtený ze vstupního souboru, příchod paketu v počítačové síti, nebo stisk tlačítka nějakého fyzického zařízení.

2.1.4 Změny stavů. V deterministickém konečném automatu je změna stavu jednoznačně určena dosavadním stavem a vstupním symbolem, což lze formalizovat tzv. *přechodovou funkcí* $\delta : Q \times \Sigma \to Q$, která každému stavu $q \in Q$ a každému vstupnímu symbolu $x \in \Sigma$ přiřazuje nový stav.

Změna stavu se děje v okamžiku příchodu vstupního symbolu. Mezi příchody vstupních symbolů a jimi vyvolanými změnami stavů se v automatu nic neděje.

2.1.5 Výstup. Výstup konečného automatu může záviset na dvojici stav (ve kterém se automat nachází) a vstup (podnět, na který automat reaguje). V takovém případě mluvíme o *Mealyho automatu*. Nebo výstup závisí pouze na stavu, ve kterém se automat nachází — pak mluvíme o

Mooreově automatu. Jestliže výstupem Mooreova automatu je „stav je koncový“ nebo „stav není koncový“, můžeme rozdělit stavy na „koncové“, tj. „přijímající“, a „nekoncové“, tj. „nepřijímající“, a hledat takové posloupnosti podnětů, po kterých automat skončí v koncovém stavu. Poslednímu typu konečného automatu budeme říkat deterministický konečný automat a zkracovat to na DFA (Deterministic Finite Automaton). Ve starší literatuře najdete i pojem „akceptor“.

2.1.6 Příklad 1 – automat na kávu. Uvažujme zjednodušený příklad automatu na kávu. Automat přijímá mince 1 Kč, 2 Kč a 5 Kč. Automat vydává jediný druh kávy, káva stojí 7 Kč. Automat na tlačítko s vrátí nevyužité peníze. Tento příklad uvedeme podrobněji.

Položíme $Q = \{0,1,2,3,4,5,6\}$, $\Sigma = \{1,2,5,s\}$, $Y = \{K,0,1,2,3,4,5,6\}$, přechodová a výstupní funkce jsou dány následující tabulkou:

	1	2	5	s
0	$1/0$	$2/0$	$5/0$	$0/0$
1	$2/0$	$3/0$	$6/0$	$0/1$
2	$3/0$	$4/0$	$0/K$	$0/2$
3	$4/0$	$5/0$	$1/K$	$0/3$
4	$5/0$	$6/0$	$2/K$	$0/4$
5	$6/0$	$0/K$	$3/K$	$0/5$
6	$0/K$	$1/K$	$4/K$	$0/6$

V prvním sloupci jsou stavy, ve kterých se automat může nacházet, v prvním řádku jsou vstupní symboly. V řádku odpovídajícím stavu q a sloupci se vstupem a je dvojice (nový stav, výstup). (K znamená kávu, číslo udává vrácené peníze).

2.1.7 Příklad 2 – posuvný registr. Stavem je uspořádaná k-tice naposled přečtených symbolů. Přečtením dalšího symbolu přejdeme do nové k-tice, tj. do nového stavu.

Na přednášce ukážeme posuvný registr, který realizuje celočíselné dělení čtyřmi daného čísla v binárním zápise čteném počínaje nejvyšším řádem.

2.1.8 Příklad 3. Zjistit, zda se v daném slovu u (textu) vyskytuje podslovo aab (šablona), lze konečným automatem.

V tomto příkladě máme 4 stavy

- q_0 buď jsme ještě nečetli žádný symbol nebo poslední čtený symbol nebyl a a přečtená část neobsahovala slovo aab jako podslovo;
- q_1 přečtená část končí jedním a, ale ne dvěma a a přečtená část neobsahuje slovo aab jako podslovo;
- q_2 přečtená část končí aa a neobsahuje slovo aab jako podslovo;
- q_3 přečtená část slova obsahuje podslovo aab.

Stav q_3 je koncovým (přijímajícím) stavem.

2.1.9 Základní typy automatů. Obecně rozlišujeme čtyři typy automatů: Mealyho automat, Mooreův automat, deterministický konečný automat DFA (akceptor) a automat bez výstupu. (Dále se budeme zabývat hlavně deterministickými konečnými automaty – DFA.)

Mealyho automat je šestice $(Q,\Sigma,Y,\delta,q_0,\lambda)$. kde

- Q je konečná množina stavů,
- Σ je konečná množina vstupních symbolů,
- Y je konečná množina výstupních symbolů,
- q_0 je počáteční stav,
- δ je přechodová funkce, tj. zobrazení $\delta\colon Q\times\Sigma\to Q$,
- λ je výstupní funkce, tj. zobrazení $\lambda\colon Q\times\Sigma\to Y$.

□

Příkladem Mealyho automatu je např. automat na kávu 2.1.6.

Mooreův automat je šestice $(Q, \Sigma, Y, \delta, q_0, \beta)$. kde Q, Σ, Y, δ a q_0 mají stejný význam v případě Mealyho automatu a β je značkovací funkce, tj. zobrazení $\beta: Q \to Y$. □
Příkladem Mooreova automatu je např. posuvný registr 2.1.7.

DFA, též akceptor, je je pětice $(Q, \Sigma, \delta, q_0, F)$, kde Q, Σ, δ a q_0 mají stejný význam jako v případě Mooreova automatu a $F \subseteq Q$ je množina koncových (též přijímajících) stavů. □
Příkladem DFA je např. automat z 2.1.8.

Poznámka. DFA je vlastně Mooreův automat, kde množina výstupních symbolů má dva prvky, totiž $Y = \{0, 1\}$, a proto značkovací funkci β nahrazujeme množinou těch stavů, kterým značkovací funkce přiřazuje 1.

Automat bez výstupu je „společnou částí“ všech výše uvedených automatů; tj. jedná se o čtveřici (Q, Σ, δ, q_0). □

2.1.10 Stavový diagram. Kromě tabulky můžeme konečný automat zadat též stavovým diagramem.

Je dán konečný automat s množinou stavů Q, množinou vstupních symbolů Σ a přechodovou funkcí δ. *Stavovým diagramem* nazýváme orientovaný ohodnocený graf, jehož vrcholy jsou stavy automatu (tj. $V = Q$) a orientovaná hrana vede z vrcholu q do vrcholu p právě tehdy, když $\delta(q, a) = p$. Taková hrana je ohodnocena a, je-li automat DFA nebo Mooreův; a je ohodnocená dvojicí $a/\lambda(q, a)$, je-li automat Mealyho.

Jestliže se jedná o Mooreův automat, vrcholy stavového diagramu jsou navíc ohodnoceny značkovací funkcí β. Pro akceptor, tj. DFA, označujeme pouze množinu koncových stavů, a to buď šipkou mířící ze stavu ven nebo jiným označením stavů, které patří do množiny F. Počáteční stav q_0 je označován šipkou mířící do něj.

2.1.11 Rozšířená přechodová funkce popisuje chování automatu ne jen nad jedním vstupním symbolem, ale nad celým vstupním slovem. Definujeme ji (jako řadu pojmů teorie automatů) induktivně.

Definice. Je dán automat s množinou stavů Q, vstupní abecedou Σ a přechodovou funkcí δ). *Rozšířená přechodová funkce* $\delta^\star: Q \times \Sigma^\star \to Q$ je definovaná induktivně takto:

1. $\delta^\star(q, \varepsilon) = q$, pro všechna $q \in Q$,
2. $\delta^\star(q, ua) = \delta(\delta^\star(q, u), a)$, pro všechna $q \in Q$, $a \in \Sigma$, $u \in \Sigma^\star$.

□

2.1.12 Tvrzení. Je dán automat (Q, Σ, δ). Potom pro každý stav $q \in Q$ a každá slova $u, v \in \Sigma^\star$ platí

$$\delta^\star(q, uv) = \delta^\star(\delta^\star(q, u), v).$$

□

Toto tvrzení se dokáže např. indukcí podle $n = |u| + |v|$.

Neformální zdůvodnění. Představte si práci konečného automatu ve stavu q nad slovem $u = a_1 a_2 \dots a_k$ jako sled ve stavovém diagramu, který začíná ve stavu (vrcholu) q a je ohodnocen postupně a_1, a_2, …, a_k. Tento sled končí ve stavu $\delta^\star(q, a_1 \dots, a_k)$.

Přesněji: Označme $q = p_0$ (což je $\delta^\star(q, \varepsilon)$), $p_1 = \delta(p_0, a_1)$ (což je $\delta^\star(q, a_1)$), $p_2 = \delta(p_1, a_2)$ (což je $\delta^\star(q, a_1 a_2)$), …, $p_k = \delta(p_{k-1}, a_k)$ (což je $\delta^\star(q, a_1 \dots a_k)$). Pak $p_0, p_1, \dots, p_k$ je sled, který začíná ve stavu q, je označen slovem a_1, a_2, …, a_k a končí ve stavu $\delta^\star(q, a_1 \dots, a_k)$.

Nyní je tvrzení 2.1.12 zřejmé, jedná se totiž o „navázání“ dvou sledů – sledu z q do $\delta^\star(q, u)$ a sledu z $\delta^\star(q, u)$ do $\delta^\star(q, uv)$.

2.1.13 Poznámka. Na konečný automat se můžeme dívat ještě následujícím způsobem: Je dána vstupní páska, která na začátku obsahuje vstupní slovo. Dále je dána řídící jednotka, která pomocí čtecí hlavy čte (postupně odleva doprava) vstupní symboly. Automat na základě stavu, ve kterém se nachází řídící jednotka, a na základě vstupního symbolu, který čte hlava, změní stav a posune čtecí hlavu po vstupní pásce o jedno pole doprava. Jestliže v okamžiku, kdy čtecí hlava „přečte“ celé vstupní slovo (hlava se dostane za poslední symbol vstupního slova), je řídící jednotka v koncovém stavu, automat slovo přijme, v opačném případě automat slovo nepřijme.

2.1.14 Jazyk přijímaný konečným automatem. Je dán DFA $M = (Q, \Sigma, \delta, q_0, F)$. Řekneme, že slovo $u \in \Sigma^\star$ je *přijímáno* automatem M, jestliže

$$\delta^\star(q_0, u) \in F.$$

Množina všech slov, které automat přijímá, se nazývá *jazyk přijímaný* M, značíme ji $L(M)$. Tedy,

$$L(M) = \{w\,;\, \delta^\star(q_0, w) \in F\}.$$

□

2.1.15 Regulární jazyky jsou jazyky, které jsou přijímány některým DFA.

V minulé přednášce jsme zavedli regulární jazyky jako ty jazyky L, pro které existuje konečný automat (DFA) M, který přijímá jazyk L. Nejdříve si ukážeme jednu vlastnost, kterou každý regulární jazyk má (pumping lemma), potom vlastnost, která regulární jazyky plně charakterizuje (Nerodova věta).

Začneme **nutnou** podmínku pro to, aby daný jazyk byl regulární. Tvrzení se také nazývá lemma o vkládání. Poznamenejme, že tato podmínka není postačující.

2.1.16 Pumping lemma pro regulární jazyky. Pro každý regulární jazyk L nad abecedou Σ existuje přirozené číslo n s touto vlastností:

Každé slovo $u \in L$, které je delší než n, lze rozdělit na tři slova $u = xwy$ tak, že

1. $|xw| \leq n$,
2. $w \neq \varepsilon$
3. a pro každé přirozené číslo $i = 0, 1, \ldots$ platí $xw^i y \in L$.

□

Dříve než pumping lemma dokážeme, ukažme, jak se dá toto tvrzení využít. Z toho, že se jedná pouze o nutnou podmínku, vyplývá, že je možno jej využít pouze pro důkaz, že nějaký jazyk **není** regulární.

2.1.17 Fakt. Jazyk $L = \{0^m\, 1^m\,;\, m \geq 0\}$ není regulární jazyk. □

Zdůvodnění. Kdyby L byl regulární jazyk, muselo by existovat přirozené číslo n s vlastnostmi 1 až 3 z pumping lemmatu 2.1.16. Ukážeme, že to vede ke sporu.

Předpokládejme, že takové číslo n existuje. Vezmeme slovo $u = 0^n\, 1^n$, které je délky $2n$, což je víc než n. Podle pumping lemmatu lze u rozložit na tři slova $u = xwy$ s vlastnostmi 1 až 3. Ukážeme, že to není možné.

Víme, že xw je prefix slova u a $|xw| \leq n$, proto se slovo xw skládá ze samých 0. Navíc $w \neq \varepsilon$, musí proto $w = 0^k$ po nějaké $k \geq 1$. Pak ale slovo $xw^2y = 0^{n+k}1^n$ nemá stejný počet 0 i 1, tj. neleží v jazyce L, ale podle pumping lemmatu by v L ležet mělo. Odvodili jsme spor; chyba byla v tom, že jsme předpokládali, že L je regulární. jazyk.

2.1.18 Myšlenka důkazu pumping lemmatu. Uvažujme libovolný regulární jazyk L. Protože L je regulární, existuje DFA $M = (Q, \Sigma, \delta, q_0, F)$, který tento jazyk přijímá (tj. $L = L(M)$).

Označme n počet jeho stavů. Vezměme libovolné slovo $u \in L$ délky větší než n. Sled ve stavovém diagramu, který odpovídá práci automatu nad slovem u, musí obsahovat cyklus (má více nebo stejně hran jako je počet (stavů), tudíž to nemůže být cesta). Označme

- x slovo, které odpovídá té části sledu, než poprvé vstoupíme do prvního cyklu,
- w slovo, které odpovídá jednomu průchodu tímto cyklem, a
- y slovo, které odpovídá zbylé části sledu.

Není těžké se přesvědčit, že slova x, w, y splňují všechny vlastnosti z pumping lemmatu.

Poznámka. Podmínka z pumping lemmatu je pouze nutná, není postačující. Následující věta plně charakterizuje regulární jazyky, tj. jazyky, které jsou přijímány konečným automatem.

2.1.19 Nerodova věta. Je dán jazyk L nad abecedou Σ. Pak L je regulární jazyk právě tehdy, když existuje ekvivalence T na množině všech slov $\Sigma^\star$ taková, že

1. L je sjednocení některých tříd ekvivalence T.
2. Jestliže pro nějaká slova $u, v \in \Sigma^\star$ platí $u\ T\ v$, pak pro každé slovo $w \in \Sigma^\star$ platí také $uw\ T\ vw$.
3. T má pouze konečně mnoho tříd ekvivalence.

□

Poznamenejme, že druhá podmínka vlastně říká, že ekvivalence T je pravá kongruence monoidu $(\Sigma^\star, \cdot, \varepsilon)$.

Myšlenka důkazu. Jestliže je jazyk L regulární, pak existuje DFA $M = (Q, \Sigma, \delta, q_0, F)$, takový, že $L = L(M)$. Definujme relaci T na $\Sigma^\star$ takto:

$$u\ T\ v \quad \text{právě tehdy, když} \quad \delta^\star(q_0, u) = \delta^\star(q_0, v).$$

Takto definovaná relace splňuje všechny podmínky Nerodovy věty.

Předpokládejme, že pro jazyk L existuje ekvivalence T splňující všechny podmínky z Nerodovy věty. Označme $[u]_T$ třídu ekvivalence T obsahující slovo u. Definujme DFA $M = (Q, \Sigma, \delta, q_0, F)$ takto:

$$Q = \{[u]_T\,;\, u \in \Sigma^\star\},\ \ q_0 = [\varepsilon]_T,\ F = \{[u]_T\,;\, [u]_T \subseteq L\};$$
$$\delta([u]_T, a) = [ua]_T \quad \text{pro každé } a \in \Sigma.$$

Není těžké ukázat, že DFA M přijímá jazyk L. □

2.1.20 Poznámka. Nerodova věta se také dá použít k tomu, abychom ukázali, že některý jazyk není regulární. Ukážeme si to na příkladu.

Fakt. Jazyk $L = \{0^m\, 1^m\,;\, m \geq 0\}$ není regulární jazyk.

Zdůvodnění. Předpokládejme, že jazyk L je regulární. Pak existuje ekvivalence T s vlastnostmi z Nerodovy věty 2.1.19. Uvažujme slova

$$0, 0^2, 0^3, 0^4, \ldots, 0^n, \ldots$$

Těchto slov je nekonečně mnoho a ekvivalence T má pouze konečně mnoho tříd, proto musejí existovat přirozená čísla i, k, $i \neq k$, taková, že $0^i\ T\ 0^k$.

Protože ekvivalence T splňuje druhou podmínku z Nerodovy věty, musí platit $0^i w\ T\ 0^k w$ pro každé binární slovo w. Zvolme $w = 1^i$. pak dostáváme

$$0^i 1^i\ T\ 0^k 1^i.$$

Ovšem slovo $0^i 1^i$ patří do jazyka L, kdežto slovo $0^k 1^i$ do jazyka L nepatří. To je ale v rozporu s první podmínkou z Nerodovy věty; totiž, že jazyk L je sjednocením některých tříd ekvivalence T. Proto jazyk L není regulární. □

2.1.21 Jak dokázat, že daný automat přijímá daný jazyk. Obvykle není problém vymyslet si pro daný regulární jazyk L *nějaký* automat, problém je *dokázat*, že funguje, tj. že přijímá správný jazyk.

Vyzkoušením činnosti automatu na několika příkladech lze prokázat, že automat funguje špatně (přijímá, co nemá, nebo nepřijímá, co má), ale nelze už tak snadno prokázat, že na všech nekonečně mnoha slovech funguje dobře.

Jednou z možností, jak dokázat, že automat přijímá daný jazyk, je využití Nerodovy věty — metoda invariantů.

Každému stavu q přiřadíme přesný popis všech slov, které převedou počáteční stav q_0 do stavu q. Tím popíšeme jednotlivou třídu ekvivalence T z Nerodovy věty. To se většinou dělá pomocí formule $I_q(u)$ (kde slovo u je proměnná); a tuto formuli nazveme *invariantem* pro stav q. Abychom opravdu dokázali, že náš automat pracuje správně, je třeba, aby:

1. Prázdné slovo musí splňovat invariant I_{q_0} počátečního stavu q_0.
2. Každé slovo u nad vstupní abecedou musí splňovat právě jeden invariant (tj. každé slovo u leží v právě jedné třídě ekvivalence T).
3. Pro každý stav q a každý vstupní symbol x platí implikace *Jestliže slovo u odpovídá invariantu $I_q(u)$, pak slovo ux odpovídá invariantu $I_{\delta(q,x)}(ux)$.*

   Tj.
   $$I_q(u) \Rightarrow I_{\delta(q,x)}(ux) \quad .$$
4. Pro každé slovo nad vstupní abecedou platí, že $u \in L$ právě tehdy, když u splňuje invariant pro některý koncový stav $q \in F$ (tj. platí $I_q(u)$ pro nějaký koncový stav q).

Vymyslet invarianty nemusí být úplně snadné, ale jsou-li již vymyšleny, pak ověření výše uvedených podmínek je již rutinní záležitostí.

2.1.22 Ekvivalentní automaty. Pro jeden regulární jazyk může existovat více DFA, které tento jazyk přijímají. Nejlepší by bylo, kdybychom byli schopni najít ten „nejjednodušší“ automat mezi nimi. Pro deterministické automaty to je možné – ukážeme postup, jak najít redukovaný automat k danému DFA a ukážeme, že jestliže dva automaty přijímají stejný jazyk, pak k nim redukované automaty se liší pouze přejmenováním stavů (jsou isomorfní). Dokonce i přejmenování stavů (isomorfismus) se dá najít jednoduchým algoritmem. Dříve než redukované automaty zavedeme, uvedeme pojem ekvivalentních automatů.

Definice. Řekneme, že dva automaty M_1 a M_2 jsou *ekvivalentní*, jestliže přijímají stejný jazyk, tj. jestliže $L(M_1) = L(M_2)$. □

V dalším textu ukážeme, jak k danému DFA najít automat, který je „co nejjednodušší“ a který přijímá stejný jazyk. To bude právě redukovaný automat.

2.1.23 V minulé přednášce jsme si zavedli pojem ekvivalentních automatů. Nyní ukážeme, jak k danému automatu sestrojit „nejmenší“ automat, který je s ním ekvivalentní.

Postup rozdělíme na dva kroky: 1) Nejprve z automatu odstraníme nedosažitelné stavy; 2) pak stavy, které „nerozlišíme“ nějakým vstupním slovem, prohlásíme za ekvivalentní — tj. „za stejné“.

2.1.24 Dosažitelné stavy. Je dán DFA $M = (Q, \Sigma, \delta, q_0, F)$. Řekneme, že stav $q \in Q$ je *dosažitelný*, jestliže existuje slovo $u \in \Sigma^\star$ takové, že $\delta^\star(q_0, u) = q$. □

Jinými slovy, stav q je dosažitelný, jestliže je dosažitelný z počátečního stavu q_0 ve stavovém diagramu automatu M (tj. z q_0 vede do q orientovaný sled).

Je zřejmé, že stavy, které nejsou dosažitelné, nemají vliv na jazyk, který daný automat přijímá.

2.1.25 Postup nalezení dosažitelných stavů. Množinu dosažitelných stavů najdeme např. následujícím postupem, který je vlastně prohledávání do šířky stavového diagramu.

$Q_0 := \{q_0\}$
`repeat`
$\quad Q_{i+1} := Q_i \cup \{\delta(q,a)\,;\, q \in Q_i, a \in \Sigma\}$
`until` $Q_{i+1} = Q_i$.
`return` $Q' = Q_i$

2.1.26 Ekvivalence stavů $\sim$. Máme dán DFA $M = (Q, \Sigma, \delta, q_0, F)$. Řekneme, že dva stavy $p, q \in Q$ jsou *ekvivalentní*, jestliže pro každé slovo $u \in \Sigma^\star$ platí

$$\delta^\star(p, u) \in F \quad \text{právě tehdy, když} \quad \delta^\star(q, u) \in F.$$

Fakt, že dva stavy p a q jsou ekvivalentní, zapisujeme $p \sim q$. □

2.1.27 Redukovaný automat. Nyní přistoupíme k definici redukovaného automatu.

Definice. Je dán DFA $M = (Q, \Sigma, \delta, q_0, F)$. Řekneme, že M je *redukovaný*, jestliže nemá nedosažitelné stavy a žádné jeho dva různé stavy nejsou ekvivalentní. □

Jinými slovy, jestliže ekvivalence $\sim$ je identická ekvivalence na množině stavů Q.

2.1.28 Konstrukce ekvivalence $\sim$. Na množině všech stavů Q postupně konstruujeme ekvivalence $\sim_i$, kde $i = 0, 1, \ldots$, takto:

a) $p \sim_0 q$ právě tehdy, když buď $p, q \in F$ nebo $p, q \notin F$.
b) Dokud $\sim_{i+1} \neq \sim_i$ konstruujeme $\sim_{i+1}$ takto
$p \sim_{i+1} q$ právě tehdy, když $p \sim_i q$ a pro každé $a \in \Sigma$ máme $\delta(p, a) \sim_i \delta(q, a)$.
c) Položíme $\sim := \sim_i$ pro $\sim_{i+1} = \sim_i$.

Správnost tohoto postupu vyplyne z následujících dvou tvrzení.

2.1.29 Tvrzení. Platí

$$\sim_0 \supseteq \sim_1 \supseteq \ldots \supseteq \sim_i \supseteq \ldots$$

Navíc, existuje k takové, že $\sim_k$ je rovna $\sim_{k+1}$. Pak pro každé $j \geq 1$ platí $\sim_k = \sim_{k+j}$. □

Důkaz. Z postupu je vidět, že ekvivalence postupně stále zjemňujeme, proto musíme po konečně mnoha krocích se zjemňováním skončit – nejjemnější ekvivalence je identická ekvivalence. Existuje tedy k takové, že $\sim_k$ je rovna $\sim_{k+1}$.

Jednoduchou matematickou indukcí s použitím definice relace $\sim_{k+1}$ se dokáže, že jestliže $\sim_k$ je rovna $\sim_{k+1}$, pak $\sim_k$ je rovna $\sim_{k+j}$ pro každé $j \geq 1$.

2.1.30 Tvrzení. Pro relace $\sim_i$ z 2.1.28 platí $p \sim_i q$ právě tehdy, když pro každé slovo u délky menší nebo rovné i je $\delta^\star(p, u) \in F$ iff $\delta^\star(q, u) \in F$. □

Důkaz vyplývá z konstrukce ekvivalencí $\sim_i$ za použití matematické indukce.

Základní krok: Jediné slovo, které má délku $i \leq 0$, je prázdné slovo ε. Dále platí $\delta^\star(q, \varepsilon) = q$, $\delta^\star(p, \varepsilon) = p$; proto tvrzení vyplývá z definice relace $\sim_0$.

Indukční krok: Mějme stavy p, q. Předpokládejme, že platí: $p \sim_i q$ právě tehdy, když pro každé slovo u s $|u| \leq i$ máme $\delta^\star(p, u) \in F$ iff $\delta^\star(q, u) \in F$.

Musíme ukázat dvě implikace. Nejprve předpokládejme, že $p \sim_{i+1} q$. Uvažujme libovolné slovo w s $|w| \leq i + 1$. Jestliže $|w| \leq i$, pak tvrzení vyplývá z indukčního předpokladu (víme, že $p \sim_i q$). Zvolme w s $|w| = i + 1$, tj. $w = au$ pro $|u| = i$. Máme $\delta^\star(p, w) = \delta^\star(\delta(p, a), u)$ a také $\delta^\star(q, w) = \delta^\star(\delta(q, a), u)$. Z definice relace $\sim_{i+1}$ víme, že platí i $\delta(p, a) \sim_i \delta(q, a)$. Proto z indukčního předpokladu (protože $|u| = i$) platí $\delta^\star(p, w) \in F$ právě tehdy, když $\delta^\star(q, w) \in F$.

Předpokládejme, že pro každé w s $|w| \leq i+1$ platí $\delta^\star(p, w) \in F$ právě tehdy, když $\delta^\star(q, w) \in F$. Pak z indukčního předpokladu víme, že $p \sim_i q$ (ano, všechna slova délky nejvýše i jsou také slova délky nejvýše $i + 1$). Kdyby nemělo platit $\delta(p, a) \sim_i \delta(q, a)$ pro nějaké $a \in \Sigma$, muselo by existovat slovo v s $|v| \leq i$ takové, že jeden ze stavů $\delta^\star(\delta(p, a), v)$ a $\delta^\star(\delta(q, a), v)$ patří do F a druhý do F nepatří. To ale není možné, neboť $\delta^\star(\delta(p, a), v) = \delta^\star(p, av)$ a $\delta^\star(\delta(q, a), v) = \delta^\star(q, av)$ a $|av| = |v| + 1 \leq i + 1$. Proto $p \sim_{i+1} q$.

Poznámka. Ekvivalence $\sim_i$ v praxi konstruujeme tak, že konstruujeme odpovídající rozklady R_i množiny stavů Q na třídy ekvivalence $\sim_i$.

2.1.31 Algoritmus redukce. Slovně můžeme redukovaný automat M_1, který konstruujeme, popsat takto: za stavy vezmeme třídy ekvivalence $\sim$; počáteční stav je třída, ve které leží původní počáteční stav q_0; přechodová funkce „pracuje“ na třídách (což je možné vzhledem k vlastnosti ekvivalence $\sim$), a množina koncových stavů je množina těch tříd, ve kterých leží alespoň jeden koncový stav původního automatu.

Formálněji: Je dán DFA $M = (Q, \Sigma, \delta, q_0, F)$.

1. Zkonstruujeme množinu Q' všech dosažitelných stavů automatu M podle postupu 2.1.25, tím dostanete automat $M' = (Q', \Sigma, \delta, q_0, F \cap Q')$.
2. Podle 2.1.28 zkonstruujeme rozklady R_i ekvivalencí $\sim_i$ pro DFA M'. Končíme tehdy, když $\sim_i = \sim_{i+1}$. Položíme $\sim = \sim_i$.
3. Označme $[q]_\sim$ třídu ekvivalence $\sim$ obsahující stav q. Vytvoříme DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, kde
   - $Q_1 := \{[q]_\sim \,;\, q \in Q'\}$ (tj. stavy M_1 jsou třídy ekvivalence $\sim$),
   - $q_1 := [q_0]_\sim$ (tj. počáteční stav je třída obsahující q_0),
   - $\delta_1([q]_\sim, a) = [\delta(q, a)]_\sim$,
   - $F_1 = \{[q]_\sim \,;\, F \cap [q]_\sim \neq \emptyset\}$.

□

2.1.32 Příklad. K DFA automatu M, který je dán následující tabulkou, najděte redukovaný automat M_1.

		a	b
$\to$	1	2	3
	2	2	4
$\leftarrow$	3	3	5
	4	2	7
$\leftarrow$	5	6	3
$\leftarrow$	6	6	6
	7	7	4
	8	2	3
$\leftarrow$	9	9	4

Řešení. Nejprve najdeme všechny dosažitelné stavy automatu M. Jsou to stavy $\{1, 2, 3, 4, 5, 6, 7\}$. Tedy $Q' = \{1, 2, 3, 4, 5, 6, 7\}$, $F' = \{3, 5, 6\}$.

Automat M' je dán tabulkou:

		a	b
$\to$	1	2	3
	2	2	4
$\leftarrow$	3	3	5
	4	2	7
$\leftarrow$	5	6	3
$\leftarrow$	6	6	6
	7	7	4

Podle definice ekvivalence $\sim_0$ je rozklad R_0 tvořen dvěma třídami:

$$O = \{1, 2, 4, 7\} \quad F = \{3, 5, 6\}.$$

Platí

$$\delta(1, a) = 2,\ \delta(2, a) = 2,\ \delta(4, a) = 2,\ \delta(7, a) = 7,$$

tj. všechny výsledky leží ve třídě O.

$$\delta(1, b) = 3,\ \delta(2, b) = 4,\ \delta(4, b) = 7,\ \delta(7, b) = 4,$$

tj. $\delta(1, b) \in F$ a $\delta(2, b), \delta(4, b), \delta(7, b) \in O$. Proto musíme množinu O rozdělit na dvě podmnožiny: $\{1\}$ a $\{2, 4, 7\}$.

Dále

$$\delta(3, a) = 3 \in F,\ \delta(5, a) = 6 \in F,\ \delta(6, a) = 6 \in F,$$

$$\delta(3, b) = 5 \in F,\ \delta(5, b) = 3 \in F,\ \delta(6, b) = 6 \in F.$$

Proto množinu F nedělíme.

Rozklad odpovídající ekvivalenci $\sim_1$ je

$$A = \{1\},\ O = \{2,4,7\},\ F = \{3,5,6\}.$$

Výpočet zahrneme do tabulky

		a	b	$\sim_0$	a	b	$\sim_1$
$\to$	1	2	3	O	O	F	A
	2	2	4	O	O	O	O
$\leftarrow$	3	3	5	F	F	F	F
	4	2	7	O	O	O	O
$\leftarrow$	5	6	3	F	F	F	F
$\leftarrow$	6	6	6	F	F	F	F
	7	7	4	O	O	O	O

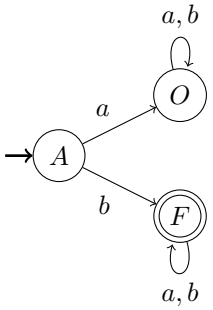
Analogicky vytváříme ekvivalence $\sim_2$. Výpočet již zkrátíme jen do tabulky.

		a	b	$\sim_0$	a	b	$\sim_1$	a	b	$\sim_2$
$\to$	1	2	3	O	O	F	A	O	F	A
	2	2	4	O	O	O	O	O	O	O
$\leftarrow$	3	3	5	F	F	F	F	F	F	F
	4	2	7	O	O	O	O	O	O	O
$\leftarrow$	5	6	3	F	F	F	F	F	F	F
$\leftarrow$	6	6	6	F	F	F	F	F	F	F
	7	7	4	O	O	O	O	O	O	O

Z tabulky vyplývá, že $\sim_1 = \sim_2$. Proto $\sim_1 = \sim$ je hledaná ekvivalence. Máme tedy tři třídy ekvivalence, a to A, O a F. Redukovaný automat M_1 má tři stavy a je dán tabulkou

		a	b
$\to$	A	O	F
	O	O	O
$\leftarrow$	F	F	F

Stavový diagram automatu je na následujícím obrázku.



Není těžké nahlédnout, že automat M_1 přijímá všechna slova jazyka $L = \{bu\,;\, u \in \{a,b\}^\star\}$.

2.1.33 Věta. Automat M i k němu redukovaný automat M_1 přijímají stejný jazyk, tj. jsou ekvivalentní. □

Myšlenka důkazu této věty vyplývá z konstrukce redukovaného automatu. Nejprve jsme odstranili nedosažitelné stavy, které se nemohou vyskytnout při přijetí žádného slova. Následně jsme pouze „slepovali“ stavy a to tak, že jestliže jsme se po orientovaném sledu z počátečního stavu označeném slovem u dostali do koncového stavu v původním automatu, tak jsme se v redukovaném automatu z počátečního stavu po sledu označeném stejným slovem dostali také do koncového astavu, a naopak.

Navíc je dobré si uvědomit, že $[q]_\sigma \cap F \neq \emptyset$ právě tehdy, když $[q]_\sim \subseteq F$. To vyplývá z faktu, že $\sim$ je zjemnění $\sim_0$.

2.1.34 Isomorfní automaty. Neformálně řečeno, dva konečné automaty jsou isomorfní, jestliže se liší pouze v pojmenování stavů. Přesněji

Definice. Jsou dány DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ a $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Řekneme, že M_1 a M_2 jsou *isomorfní*, jestliže existuje bijekce φ množiny stavů Q_1 na množinu stavů Q_2 taková, že

- $\varphi(q_1) = q_2$ (počáteční stavy si odpovídají),
- $\varphi(F_1) = F_2$ (koncové stavy si odpovídají),
- $\varphi(\delta_1(q, a)) = \delta_2(\varphi(q), a)$ pro každé $q \in Q_1$ a $a \in \Sigma$ (je jedno, zda nejprve provedete přechodovou funkci a pak stav zobrazíte nebo nejprve stav zobrazíte a pak provedete přechodovou funkci).

□

2.1.35 Věta. Dva DFA M_1 a M_2 přijímají stejný jazyk (tj. jsou ekvivalentní) právě tehdy, když jejich odpovídající redukované automaty jsou isomorfní.

Myšlenka důkazu. Jestliže jsou redukované automaty k automatům M_1 a M_2 isomorfní, pak jsou automaty M_1 a M_2 ekvivalentní. To vyplývá z věty 2.1.33.

Předpokládejme, že automaty M_1 a M_2 jsou ekvivalentní. Utvořme k nim redukované automaty $\overline{M_1}$ a $\overline{M_2}$; označme $\overline{M_1} = (Q_1, \Sigma, \delta_1, q_1, F_1)$ a $\overline{M_2} = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Vytvoříme přiřazení φ stavů Q_1 stavům Q_2 takto:

- Počáteční stavy si odpovídají, tj.
$$\varphi(q_1) = q_2.$$
- Protože všechny stavy M_1 jsou dosažitelné (a také všechny stavy M_2 jsou dosažitelné), pro každé $q \in Q_1$ existuje slovo w tak, že $\delta_1^\star(q_1, w) = q$. Položíme
$$\varphi(\delta_1^\star(q_1, w)) = \delta_2^\star(q_2, w).$$
Není těžké se přesvědčit, že φ je opravdu bijekce množiny Q_1 na Q_2 splňující všechny vlastnosti z definice isomorfismu.

□

2.2 Nedeterministické konečné automaty

Konečný deterministický automat má tu vlastnost, že v každém stavu reaguje na každé písmeno přesně jedním způsobem (přechodová funkce byla zobrazení z množiny $Q \times \Sigma$ do Q). Nedeterministické automaty se od deterministických liší tím, že jsme-li ve stavu q a čteme vstupní písmeno a, můžeme přejít do několika (a také žádného) stavů. V této sekci ukážeme, že ke každému nedeterministickému automatu existuje deterministický konečný automat, který přijímá stejný jazyk. Vzhledem k tomu, že nedeterministické automaty se snáze navrhují, zavedení nedeterministických automatů nám umožní zjednodušit postup při návrhu konečného automatu přijímajícího daný jazyk.

2.2.1 Nedeterministický automat. Shrňme, čím se nedeterministický konečný automat liší od deterministického automatu:

1. NFA může mít víc počátečních stavů, ne jen jeden,
2. každému stavu q a každému vstupnímu písmenu a přiřazuje přechodová funkce δ několik stavů (i žádný stav); tj. $\delta(q, a)$ je množina stavů (třeba i prázdná).

Definice. *Nedeterministický konečný automat*, zkráceně NFA, je pětice $(Q, \Sigma, \delta, I, F)$, kde

- Q je konečná neprázdná množina stavů,
- Σ je konečná neprázdná množina vstupů,
- δ je přechodová funkce, kde
$$\delta\colon Q \times \Sigma \to \mathcal{P}(Q),$$
($\mathcal{P}(Q)$ je množina všech podmnožin množiny Q).
- $I \subseteq Q$ je množina počátečních stavů,
- $F \subseteq Q$ je množina koncových stavů.

□

2.2.2 Stavový diagram NFA. Obdobně jako pro deterministické automaty můžeme i pro nedeterministické definovat stavový diagram. Rozdíl mezi stavovým diagramem NFA a DFA je jen v tom, že ve stavovém diagramu nedeterministického automatu mohou existovat stavy, ze kterých vychází několik orientovaných hran ohodnocených stejným písmenem i stavy, ze kterých nevychází žádná hrana ohodnocená některým písmenem.

Definice. Je dán nedeterministický automat $M = (Q, \Sigma, \delta, I, F)$. *Stavový diagram* automatu M je orientovaný ohodnocený graf, jehož množina vrcholů je množina stavů Q, z vrcholu q do vrcholu p vede hrana ohodnocená vstupním symbolem a právě tehdy, když $p \in \delta(q, a)$. Počáteční i koncové stavy jsou označeny stejně jako u stavového diagramu deterministického automatu; tj. do každého počátečního stavu a z každého koncového stavu vede šipka. □

2.2.3 Rozšířená přechodová funkce NFA. Dříve než zadefinujeme rozšířenou přechodovou funkci, zavedeme následující značení. Pro danou množinu stavů $X \subseteq Q$ a $a \in \Sigma$ položíme

$$\delta(X, a) = \bigcup \{\delta(q, a)\,;\, q \in X\}.$$

Definice. Je dán NFA $(Q, \Sigma, \delta, I, F)$. *Rozšířená přechodová funkce* $\delta^\star\colon Q \times \Sigma^\star \to \mathcal{P}(Q)$ je definovaná induktivně takto:

1. $\delta^\star(q, \varepsilon) = \{q\}$, pro všechna $q \in Q$,
2. $\delta^\star(q, ua) = \delta(\delta^\star(q, u), a)$ pro všechna $q \in Q$, $a \in \Sigma$, $u \in \Sigma^\star$.

□

2.2.4 Pro lepší pochopení práce nedeterministických automatů uveďme jednoduché pozorování (je analogické pozorování pro DFA).

Pozorování. Je dán NFA $M = (Q, \Sigma, \delta, I, F)$. Pak pro každý stav q a každé slovo $w \in \Sigma^\star$ jsou následující dvě vlastnosti ekvivalentní:

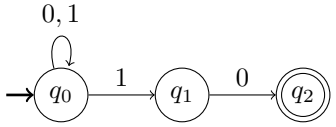
1. Existuje orientovaný sled ve stavovém diagramu z q do p označený slovem w.
2. $p \in \delta^\star(q, w)$.

Jinými slovy: množina $\delta^\star(p, w)$ je množina všech stavů p, do kterých vede orientovaný sled z vrcholu p ohodnocený slovem w. □

2.2.5 Příklad. Je dán NFA tabulkou

		0	1
→	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	q_1	$\{q_2\}$	$\emptyset$
←	q_2	$\emptyset$	$\emptyset$

se stavovým diagramem



Rozšířená přechodová funkce $\delta^\star$ pracuje nad slovem 0110 takto:

1. $\delta^\star(q_0, 0) = \{q_0\}$;
2. $\delta^\star(q_0, 01) = \delta(\{q_0\}, 1) = \{q_0, q_1\}$;
3. $\delta^\star(q_0, 011) = \delta(\{q_0\}, 1) \cup \delta(\{q_1\}, 1) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$;
4. $\delta^\star(q_0, 0110) = \delta(\{q_0\}, 0) \cup \delta(\{q_1\}, 0) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.

2.2.6 Slovo přijímané NFA, jazyk přijímaný NFA. Neformálně řečeno slovo w je přijímáno NFA právě tehdy, když ve stavovém diagramu existuje orientovaný sled označený slovem w, který začíná v některém z počátečních stavů automatu a končí v některém z koncových stavů. Jazyk se pak skládá ze všech slov, které NFA přijímá. Formálně

Definice. Řekneme, že NFA $M = (Q, \Sigma, \delta, I, F)$ *přijímá* slovo $w \in \Sigma^\star$ jestliže $\delta^\star(I, w) \cap F \neq \emptyset$; jinými slovy pro nějaký z počátečních stavů $q_0 \in I$ platí $\delta^\star(q_0, w) \cap F \neq \emptyset$.

Jazyk $L(M)$ *přijímaný* NFA M se skládá z právě všech slov, které jsou přijímány M. □

2.2.7 Každý deterministický automat $M = (Q, \Sigma, \delta, q_0, F)$ můžeme považovat za nedeterministický. Stačí říci, že množina počátečních stavů obsahuje jen q_0 a u přechodové funkce $\delta(q, a) = p$ ztotožnit stav p s jednoprvkovou množinou $\{p\}$. Následující konstrukce, tzv. podmnožinová konstrukce, ukazuje, že nedeterministické automaty přijímají pouze regulární jazyky.

Tvrzení. Ke každému nedeterministickému automatu M existuje deterministický automat $\widehat{M}$, který přijímá stejný jazyk; tj.

$$L(M) = L(\widehat{M}).$$

□

Způsob, kterým se DFA $\widehat{M}$ konstruuje, se nazývá *podmnožinová konstrukce*.

2.2.8 Podmnožinová konstrukce. Je dán nedeterministický automat $M = (Q, \Sigma, \delta, I, F)$. Definujeme deterministický automat $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, q_0, \widehat{F})$ takto:

- $\widehat{Q}$ je množina všech podmnožin množiny Q (tj. $\widehat{Q} = \mathcal{P}(Q)$);
- $q_0 = I$;
- $\widehat{F}$ je množina všech těch podmnožin Q, které obsahují aspoň jeden koncový stav M, (tj. $\widehat{F} = \{X \subseteq Q\,;\, X \cap F \neq \emptyset\}$);
- $\widehat{\delta}(X, a)$ je podmnožina všech stavů, do kterých vede hrana označená písmenem a z některého stavu množiny X (tj. $\widehat{\delta}(X, a) = \bigcup\{\delta(q, a)\,;\, q \in X\} = \delta(X, a)$.)

2.2.9 Modifikace podmnožinové konstrukce — hledání pouze dosažitelných stavů. Obdobně jako v 2.1.25 se jedná o modifikaci prohledávání stavového diagramu do šířky. Je dán nedeterministický automat $M = (Q, \Sigma, I, \delta, F)$.

1. $Q := \{I\}$; $A := \{I\}$;
2. `if` $A \neq \emptyset$ `do` $B := \emptyset$;
   `for all` $X \in A$ `do`
   `for all` $a \in \Sigma$ `do` $\widehat{\delta}(X, a) := \delta(X, a)$;
   `if` $\delta(X, a) \notin Q$ `then` $B := B \cup \{\delta(X, a)\}$;
3. `if` $B \neq \emptyset$ `do` $Q := Q \cup B$; $A := B$ `go to 2`
4. $\widehat{q_0} := I$; $\widehat{Q} := Q$; $\widehat{F} := \{X \in \widehat{Q}\,;\, X \cap F \neq \emptyset\}$;
5. `return` $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{q_0}, \widehat{\delta}, \widehat{F})$

2.2.10 Věta. Jazyk L je přijímán nějakým nedeterministickým automatem právě tehdy, když existuje deterministický automat M_1 takový, že

$$L = L(M_1).$$

□

Jinými slovy, automaty NFA i DFA přijímají stejnou třídu jazyků, a to regulární jazyky.

Myšlenka důkazu: Jedna implikace vyplývá z pozorování 2.2.7.

Druhá implikace vyplývá z podmnožinové konstrukce 2.2.8. Indukcí podle délky slova $u \in \Sigma^\star$ se dá dokázat: Pro každou podmnožinu X a každé slovo $u \in \Sigma^\star$ platí

$$\widehat{\delta^\star}(X, u) = \bigcup\{\delta^\star(q, u)\,;\, q \in X\}.$$

Tedy z definice DFA $\widehat{M}$ dostáváme

$$L(M) = L(\widehat{M}).$$

Proto $\widehat{M}$ vytvořený postupem 2.2.9, tj. podmnožinovou konstrukcí, je hledaný DFA M_1.

2.2.11 Poznámka. Poznamenejme, že počet stavů DFA $\widehat{M}$ z předchozí věty může vzrůst exponenciálně oproti počtu stavů nedeterministického automatu M.

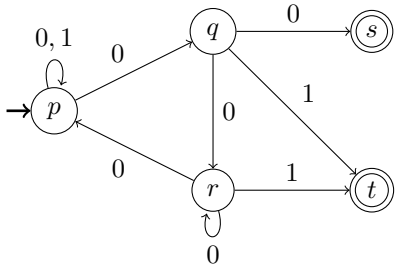
Poznamenejme, že počet stavů DFA $\widehat{M}$ z předchozí věty může vzrůst exponenciálně oproti počtu stavů nedeterministického automatu M.

Postup si ukážeme na příkladě.

2.2.12 Příklad. Je dán NFA M tabulkou

		0	1
$\rightarrow$	p	$\{p,q\}$	$\{p\}$
	q	$\{r,s\}$	$\{t\}$
	r	$\{p,r\}$	$\{t\}$
$\leftarrow$	s	$\emptyset$	$\emptyset$
$\leftarrow$	t	$\emptyset$	$\emptyset$

se stavovým diagramem



Najděte DFA $\widehat{M}$, který přijímá stejný jazyk.

Řešení. DFA vytvoříme pomocí postupu 2.2.9. Na začátku práce algoritmu máme $I = \{p\}$, $Q = \{\{p\}\}$, $A = \{\{p\}\}$.

Množinu stavů Q i přechodovou funkci $\widehat{\delta}$ konstruujeme takto:

- Po prvním průchodu bodem 2 dostáváme:

$$\widehat{\delta}(\{p\},0) := \delta(p,0) = \{p,q\}, \quad \widehat{\delta}(\{p\},1) := \delta(p,1) = \{p\} \text{ a proto } B := \{\{p,q\}\}.$$

  V kroku 3

$$Q := \{\{p\},\{p,q\}\} \quad \text{a} \quad A := \{\{p,q\}\}.$$

- Po druhém průchodu bodem 2 dostáváme:

$$\widehat{\delta}(\{p,q\},0) := \delta(p,0) \cup \delta(q,0) = \{p,q,r,s\}$$

$$\widehat{\delta}(\{p,q\},1) := \delta(p,1) \cup \delta(q,1) = \{p,t\} \quad \text{a} \quad B := \{\{p,q,r,s\},\{p,t\}\}.$$

  V kroku 3

$$Q := \{\{p\},\{p,q\},\{p,q,r,s\},\{p,t\}\} \quad \text{a} \quad A := \{\{p,q,r,s\},\{p,t\}\}.$$

- Po třetím průchodu bodem 2 dostáváme:

$$\widehat{\delta}(\{p,q,r,s\},0)=\{p,q,r,s\} \quad \text{a} \quad \widehat{\delta}(\{p,q,r,s\},1)=\{p,t\},$$

$$\widehat{\delta}(\{p,t\},0)=\{p,q\}, \;\; \widehat{\delta}(\{p,t\},1)=\{p\} \text{ a } B:=\emptyset.$$

V kroku 4 pak definujeme:

$$\widehat{Q}:=\{\{p\},\{p,q\},\{p,q,r,s\},\{p,t\}\}, \; \widehat{q_0}:=\{p\}, \; \widehat{F}:=\{\{p,q,r,s\},\{p,t\}\}.$$

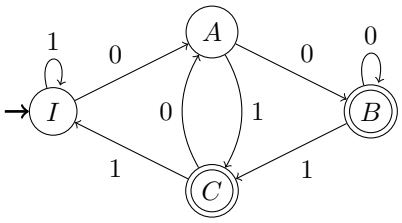
Předchozí postup je znázorněn v následující tabulce:

		0	1
$\to$	$\{p\}$	$\{p,q\}$	$\{p\}$
	$\{p,q\}$	$\{p,q,r,s\}$	$\{p,t\}$
$\leftarrow$	$\{p,q,r,s\}$	$\{p,q,r,s\}$	$\{p,t\}$
$\leftarrow$	$\{p,t\}$	$\{p,q\}$	$\{p\}$

Označíme-li $I=\{p\}$, $A=\{p,q\}$, $B=\{p,q,r,s\}$ a $C=\{p,t\}$, dostaneme následující tabulku.

		0	1
$\to$	I	A	I
	A	B	C
$\leftarrow$	B	B	C
$\leftarrow$	C	A	I

Stavový diagram je



Ze stavového diagramu není těžké nahlédnout, že jazyk přijímaný automatem M (i $\widehat{M}$) se skládá ze všech binárních slov, které končí buď 01 nebo 00, tj. ze všech binárních slov, které mají jako předposlední symbol 0.

2.3 Nedeterministické automaty s ε přechody

2.3.1 Nedeterministický automat s ε přechody (ve zkratce ε-NFA) se liší od obyčejného NFA tím, že mezi některými stavy může automat přejít, aniž by četl nějaký vstupní symbol.

Definice. ε-NFA je pětice $M=(Q,\Sigma,\delta,I,F)$, kde Q, Σ, I a F mají stejný význam jako u nedeterministického automatu, viz 2.2.1, a přechodová funkce δ je zobrazení

$$\delta\colon Q\times(\Sigma\cup\{\varepsilon\})\to\mathcal{P}(Q).$$

Stavový diagram nějakého ε-NFA definujeme takto: Vrcholy jsou stavy automatu, z vrcholu q do vrcholu p vede hrana označená ε právě tehdy, když $p\in\delta(q,\varepsilon)$, a hrana označená $a\in\Sigma$ právě tehdy, když $p\in\delta(q,a)$. □

2.3.2 Jazyk přijímaný ε-NFA (neformálně). Zde je výhodné se na práci ε-NFA nad slovem w dívat jako na množinu sledů, které začínají v některém počátečním stavu a jsou označené slovem w (kdekoli ve sledu může být hrana označená prázdným slovem ε). Slovo w je přijímáno ε-NFA, jestliže některý z těchto sledů končí v některém koncovém stavu.

K formální definici využijeme pojem rozšířené přechodové fuknce.

Rozšířená přechodová funkce (neformálně). Pro každé slovo u a stav q je $\delta^\star(q,u)$ definováno jako množina těch stavů p, do kterých ve stavovém diagramu vede z q orientovaný sled označený u. K formálně přesnější definici potřebujeme pojem ε-uzávěru.

2.3.3 ε-uzávěr. Neformálně, množina ε-UZ(X) je množina všech stavů, do kterých vede sled z některého stavu $q \in X$, jehož všechny hrany jsou označeny ε.

Definice. Je dán NFA s ε přechody $M = (Q, \Sigma, \delta, I, F)$. Pro množinu stavů X definujeme *ε-uzávěr* ε-UZ(X) indukcí takto:
- $X \subseteq \varepsilon$-UZ(X);
- je-li $p \in \varepsilon$-UZ(X), pak $\delta(p,\varepsilon) \subseteq \varepsilon$-UZ$(X)$.

□

Má-li množina X jediný stav q, píšeme ε-UZ(q) místo ε-UZ$(\{q\})$.

2.3.4 Rozšířená přechodová funkce. Definice. *Rozšířená přechodová funkce* $\delta^\star$ je definována indukcí takto:

- $\delta^\star(q,\varepsilon) = \varepsilon$-UZ$(q)$;
- $\delta^\star(q,ua) = \bigcup\{\varepsilon\text{-UZ}(\delta(p,a)) \mid p \in \delta^\star(q,u)\}$, pro $a \in \Sigma$, $u \in \Sigma^\star$.

□

2.3.5 Jazyk přijímaný ε-NFA. Definice. Slovo u je *přijímáno ε-NFA*, jestliže existuje počáteční stav q a koncový stav p takový, že $p \in \delta^\star(q,u)$. *Jazyk $L(M)$ přijímaný ε-NFA M* je množina všech slov přijímaných automatem M. □

2.3.6 Uvědomte si, že každý DFA je speciálním případem NFA a každý NFA je speciálním případem ε-NFA. Proto každý regulární jazyk je přijímán některým ε-NFA. Následující věta říká, že nedeterministické automaty s ε přechody nepřijímají „nic navíc."

Věta. Jazyk přijímaný libovolným ε-NFA je regulární. □

Nástin důkazu: Větu dokážeme modifikací podmnožinové konstrukce, která ke každému NFA zkonstruuje DFA přijímající stejný jazyk. Rozdíl je v tom, že množina stavů DFA se v případě ε-NFA neskládá ze všech podmnožin stavů automatu, ale jen z těch podmnožin, které jsou ε uzavřené. Přesněji:

Máme ε-NFA $M = (Q, \Sigma, \delta, I, F)$, který přijímá jazyk L. Označme Q_1 množinu všech podmnožin X množiny Q, které jsou ε uzavřené, tj. pro které platí ε-UZ$(X) = X$. Popíšeme variantu podmnožinové konstrukce, jejímž výsledkem bude DFA přijímající jazyk L.

Označme Q_1 množinu všech ε uzavřených podmnožin množiny Q.

Definujeme:

- $\delta_1(X,a) = \bigcup\{\varepsilon\text{-UZ}(\delta(q,a)) \mid q \in X\}$.
- $I_1 = \varepsilon$-UZ(I).
- $F_1 = \{X \in Q_1 \mid X \cap F \neq \emptyset\}$.

Pak $M_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$ je deterministický automat, který přijímá jazyk L.

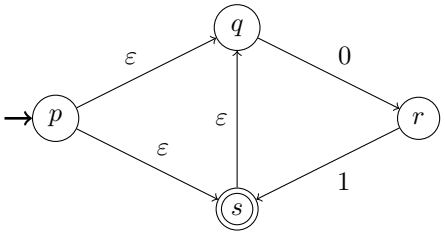
Poznámka. Opět se stačí zabývat jen dosažitelnými ε=uzavřenými množinami; tj. řídit se postupem analogickým 2.2.9. Uvědomte si, že sjednocením ε-uzavřených množin je opět ε-uzavřená množina.

Postup si ukážeme na následujícím příkladu.

2.3.7 Příklad. Je dán ε-NFA tabulkou

		ε	0	1
$\to$	p	$\{q,s\}$	$\emptyset$	$\emptyset$
	q	$\emptyset$	$\{r\}$	$\emptyset$
	r	$\emptyset$	$\emptyset$	$\{s\}$
$\leftarrow$	s	$\{q\}$	$\emptyset$	$\emptyset$

se stavovým diagramem



Najděte DFA $\widehat{M}$, který přijímá stejný jazyk a zredukujte ho.

Řešení. DFA vytvoříme pomocí postupu, který je analogický postupu 2.2.9. Máme:

$$\varepsilon\text{-UZ}(\{p\}) = \{p,q,s\},\ \varepsilon\text{-UZ}(\{q\}) = \{q\},\ \varepsilon\text{-UZ}(\{r\}) = \{r\}\ \varepsilon\text{-UZ}(\{s\}) = \{q,s\}.$$

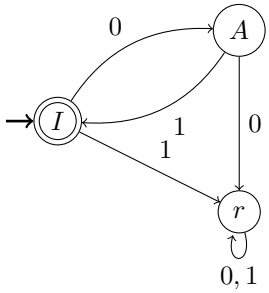
Proto počáteční stav $\widehat{M}$ je ε-UZ$(\{p\}) = \{p,q,s\}$. Další postup si znázorníme rovnou do tabulky.

		0	1
$\leftrightarrow$	$\{p,q,s\}$	$\{r\}$	$\emptyset$
	$\{r\}$	$\emptyset$	$\{q,s\}$
$\leftarrow$	$\{q,s\}$	$\{r\}$	$\emptyset$
	$\emptyset$	$\emptyset$	$\emptyset$

Po redukci a při označení $I = \{p,q,s\}$, $A = \{r\}$ a $O = \emptyset$, dostaneme výsledný DFA:

		0	1
$\leftrightarrow$	I	A	O
	A	O	I
	O	O	O

se stavovým diagramem



Není těžké nahlédnout, že $L(\widehat{M}) = \{(01)^n \mid n \geq 0\}$.

2.4 Uzávěrové vlastnosti regulárních jazyků

2.4.1 Operace s jazyky. V dalším textu si nejprve připomeneme množinové operace s jazyky a pak zavedeme ještě další operace specifické pro jazyky.

Jsou-li L_1 a L_2 dva jazyky nad stejnou abecedou, pak můžeme vytvořit jejich sjednocení $L_1 \cup L_2$, průnik $L_1 \cap L_2$ a doplněk jazyka L_1, kterým je jazyk $\overline{L_1} = \Sigma^\star - L_1$ (tj. $w \in \overline{L_1}$ právě tehdy, když $w \notin L_1$). □

2.4.2 Zřetězení jazyků. Jsou dány jazyky L_1 a L_2 nad abecedou Σ. *Zřetězení* jazyků L_1 a L_2 je jazyk L_1L_2 definovaný

$$L_1L_2 = \{uv \mid u \in L_1, v \in L_2\}.$$

□

2.4.3 Kleeneho operace $\star$. Je dán jazyk L nad abecedou Σ. Definujeme $L^0 = \{\varepsilon\}$, $L^{i+1} = L^iL$ pro $i \geq 0$. Jazyk $L^\star$ je definován takto

$$L^\star = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \ldots = \bigcup_{i=0}^{\infty} L^i.$$

□

Poznamenejme, že operaci $\star$ se též říká *Kleeneho operátor*.

2.4.4 Reverze, též obrácení. Je dán jazyk L nad abecedou Σ. Definujeme reverzi (obrácení) jazyka L takto

$$L^R = \{w^R \mid w \in L\}.$$

□

Připomeňme, že na příklad $a_1a_2 \ldots a_{n-1}a_n^R = a_na_{n-1} \ldots a_2a_1$.

2.4.5 Poznámka. Uvědomte si, že pro jazyky L_1 a L_2 obecně **neplatí**:

1. $(L_1 \cup L_2)^\star = L_1^\star \cup L_2^\star$.
2. $(L_1 \cap L_2)^\star = L_1^\star \cap L_2^\star$.
3. $L_1^\star L_2^\star = (L_1L_2)^\star$.

2.4.6 Věta. Třída regulárních jazyků je uzavřena na následující operace: 1) sjednocení, 2) doplněk, 3) průnik, 4) zřetězení, 5) Kleeneho operaci $\star$ a 6) reverzi.

Přesněji, jestliže L, L_1 a L_2 jsou regulární jazyky, pak také $L_1 \cup L_2$, $\overline{L}$, $L_1 \cap L_2$, L_1L_2, $L^\star$ a L^R jsou regulární jazyky. □

Nástin důkazu: Pro L, L_1 a L_2 uvažujme DFA, které je přijímají; označme je po řadě $M = (Q, \Sigma, \delta, q_0, F)$, $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ a $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Z jejich stavových diagramů vyrobíme stavové diagramy ε-NFA pro požadované jazyky. Předpokládáme, že množiny stavů Q_1 a Q_2 jsou disjunktní (uvědomte si, že bez tohoto předpokladu by naše konstrukce nebyly správné).

- Pro sjednocení jazyků stačí příslušné stavové diagramy položit vedle sebe. Přesněji NFA $\overline{M} = (Q_1 \cup Q_2, \Sigma, \overline{\delta}, \{q_1, q_2\}, F_1 \cup F_2)$, kde pro $q \in Q_1$ je $\overline{\delta}(q, a) = \delta_1(q, a)$ a pro $q \in Q_2$ je $\overline{\delta}(q, a) = \delta_2(q, a)$. Takto definovaný NFA přijímá jazyk $L_1 \cup L_2$.

- Pro doplněk $\overline{L}$ stačí v M přehodit koncové a nekoncové stavy, tj. koncové stavy DFA přijímajícího $\overline{L}$ je množina $Q \setminus F$. (Uvědomte si, že k tomu, aby stačilo přejmenovat koncové a nekoncové stavy, je nutné, aby automat byl deterministický; pro nedeterministický automat to udělat nemůžeme.)

- Protože průnik $L_1 \cap L_2$ je roven doplňku sjednocení doplňků, tj. $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, tvrzení vyplývá z předchozích dvou bodů.

- Pro zřetězení jazyků L_1 a L_2 položíme automaty M_1 a M_2 za sebe a z každého koncového stavu automatu M_1 vedeme ε přechod do počátečního stavu automatu M_2 (tj. pro $q \in F_1$ přidáme přechod $\overline{\delta}(q, \varepsilon) = \{q_2\}$.) Výsledný automat je pak $(Q_1 \cup Q_2, \Sigma, \overline{\delta}, \{q_1\}, F_2)$.

- Pro Kleeneho operaci $\star$, tj. pro jazyk $L^\star$, vezmeme automat M, k množině Q přidáme nový počáteční stav p_0, který bude současně novým koncovým stavem, a k přechodům automatu M přidáme tyto ε přechody: z nového počátečního stavu p_0 a z každého koncového stavu $q \in F$ přidáme ε přechod do původního počátečního stavu q_0 (všechny koncové stavy automatu M zůstanou koncové). Tím dostaneme ε-NFA, který přijímá $L^\star$.
- Pro reverzi jazyka L stačí v automatu M obrátit šipky a zaměnit počáteční a koncové stavy. (Uvědomte si, že se z DFA většinou stane NFA.)

2.4.7 Poznámka – součinová konstrukce. Fakt, že průnik a sjednocení dvou regulárních jazyků je opět regulární jazyk, se dá dokázat také pomocí součinové konstrukce. Důvody, proč zde součinovou konstrukci uvádíme, jsou dva. První je to, že pomocí ní se dá přímo z DFA pro jazyky L_1 a L_2 zkonstruovat DFA přijímající jazyk $L_1 \cap L_2$. Druhý je fakt, že se jedná o základní konstrukci, kterou využijeme i později při studiu zásobníkových automatů.

Součinová konstrukce Jsou dány DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ a $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ takové, že $L_1 = L(M_1)$ a $L_2 = L(M_2)$. Definujme DFA $M = (Q, \Sigma, \delta, q_0, F)$ takto:

- $Q = Q_1 \times Q_2$,
- $q_0 = (q_1, q_2)$,
- $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$,
- $F = F_1 \times F_2$.

Není těžké ukázat, že $L(M) = L_1 \cap L_2$. □

Výše uvedenou součinovou konstrukci jsme mohli využít i pro konstrukci DFA M', který přijímá jazyk $L(M') = L_1 \cup L_2$. Jediný rozdíl mezi M a M' je v množině koncových stavů – M' má množinu koncových stavů rovnu $F' = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

2.5 Regulární výrazy

2.5.1 Regulární jazyky jsme definovali jako ty jazyky, které jsou přijímány konečnými automaty. Ukázali jsme, že nezáleží na tom, zda jsou deterministické nebo nedeterministické. Regulární výrazy je další popis regulárních jazyků – viz 2.5.4 a 2.5.5. (Byly to právě regulární výrazy, které daly jméno třídě jazyků přijímaných konečnými automaty.)

2.5.2 Regulární výrazy nad abecedou.

Definice. Je dána abeceda Σ. Množina všech regulárních výrazů nad Σ je definována induktivně takto:

- $\emptyset$ je regulární výraz,
- ε je regulární výraz,
- $\mathbf{a}$ je regulární výraz pro každé písmeno $a \in \Sigma$,
- jsou-li $\mathbf{r_1}$ a $\mathbf{r_2}$ regulární výrazy, pak $(\mathbf{r_1} + \mathbf{r_2})$, $\mathbf{r_1 r_2}$ a $\mathbf{r_1^\star}$ jsou také regulární výrazy.

□

2.5.3 Jazyk reprezentující regulární výraz. Každý regulární výraz nad abecedou Σ reprezentuje jazyk nad abecedou Σ a to takto:

- Regulární výraz $\emptyset$ reprezentuje jazyk $\emptyset$.
- Regulární výraz ε reprezentuje jazyk $\{\varepsilon\}$.
- Je-li $a \in \Sigma$, pak regulární výraz $\mathbf{a}$ reprezentuje jazyk $\{a\}$.
- Jestliže regulární výraz $\mathbf{r_1}$ reprezentuje jazyk L_1 a regulární výraz $\mathbf{r_2}$ reprezentuje jazyk L_2, pak regulární výraz $(\mathbf{r_1} + \mathbf{r_2})$ reprezentuje jazyk $L_1 \cup L_2$ a regulární výraz $\mathbf{r_1 r_2}$ reprezentuje jazyk $L_1 L_2$.
- Jestliže regulární výraz $\mathbf{r}$ reprezentuje jazyk L, pak regulární výraz $\mathbf{r}^\star$ reprezentuje jazyk $L^\star$.

□

2.5.4 Věta. Každý jazyk reprezentovaný regulárním výrazem je regulární. □

Důkaz: Regulární výrazy $\emptyset$, ε, $\mathbf{a}$ (pro $a \in \Sigma$) reprezentují po řadě jazyky $\emptyset$, $\{\varepsilon\}$, $\{a\}$. Všechny tyto jazyky jsou regulární.

O třídě regulárních jazyků víme, že je uzavřena na sjednocení, zřetězení a Kleeneho operaci $\star$. To znamená, že jsou-li jazyky reprezentované regulárními výrazy $\mathbf{r}$, $\mathbf{r_1}$ a $\mathbf{r_2}$ regulární, pak jsou regulární i jazyky reprezentované regulárními výrazy $(\mathbf{r_1} + \mathbf{r_2})$, $\mathbf{r_1}\,\mathbf{r_2}$ a $\mathbf{r}^\star$.

2.5.5 Kleeneho věta. Každý jazyk přijímaný konečným automatem je možné reprezentovat regulárním výrazem. □

Důkaz: Je dán DFA $M = (Q, \Sigma, \delta, q_0, F)$, který přijímá jazyk L. Pro jednoduchost označme množinu stavů $Q = \{1, \ldots, n\}$ a počáteční stav $q_0 = 1$. Pro $k = 0, 1, \ldots, n$ definujeme množiny slov $R_{i,j}^{(k)}$ takto:

$R_{i,j}^{(k)}$ je množina těch slov w, že $\delta^\star(i, w) = j$ a sled z i do j má vnitřní stavy jen z $\{1, \ldots, k\}$.

Zřejmě platí:

$$\text{pro } i \neq j \text{ je } R_{i,j}^{(0)} = \{a\,;\, \delta(i,a) = j\};\ \text{pro } i = j \text{ je } R_{i,i}^{(0)} = \{\varepsilon\} \cup \{a\,;\, \delta(i,a) = j\}.$$

To platí pro to, že sledy, které nemají vnitřní vrchol, jsou buď sledy o jedné hraně, nebo triviální sledy.

V obou případech se jedná o konečnou množinu. Proto umíme množinu $R_{i,j}^{(0)}$ reprezentovat regulárním výrazem.

Předpokládejme, že všechny množiny slov $R_{i,j}^{(k)}$ pro dané k umíme reprezentovat regulárním výrazem $\mathbf{r}_{i,j}^k$. Pak pro množinu slov $R_{i,j}^{(k+1)}$ platí:

$$R_{i,j}^{(k+1)} = R_{i,j}^{(k)} \cup R_{i,k+1}^{(k)}\,(R_{k+1,k+1}^{(k)})^\star\,R_{k+1,j}^{(k)}.$$

Proto množinu $R_{i,j}^{(k+1)}$ lze reprezentovat regulárním výrazem $\mathbf{r}_{i,j}^k + \mathbf{r}_{i,k+1}^k\,(\mathbf{r}_{k+1,k+1}^k)^\star\,\mathbf{r}_{k+1,j}^k$, což je opět regulární výraz.

Navíc, jazyk L je sjednocení všech množin $R_{1,j}^{(n)}$ pro $j \in F$. Proto jazyk L je reprezentován regulárním výrazem $\sum_{j\in F} \mathbf{r}_{1,j}^n$.

2.5.6 Ekvivalentní regulární výrazy.

Definice. Řekneme, že dva regulární výrazy $\mathbf{r}$ a $\mathbf{q}$ jsou *ekvivalentní*, jestliže jimi reprezentované jazyky jsou stejné. Fakt, že regulární výrazy $\mathbf{r}$ a $\mathbf{q}$ jsou ekvivalentní zapisujeme $\mathbf{r} \models\!\models \mathbf{q}$. □

2.5.7 Některé ekvivalentní regulární výrazy. Jsou-li $\mathbf{r}$, $\mathbf{p}$ a $\mathbf{q}$ regulární výrazy, pak

1. $\mathbf{p} + \mathbf{q} \models\!\models \mathbf{q} + \mathbf{p}$,
2. $(\mathbf{p} + \mathbf{q})\mathbf{r} \models\!\models \mathbf{pr} + \mathbf{qr}$, $\mathbf{r}(\mathbf{p} + \mathbf{q}) \models\!\models \mathbf{r}\,\mathbf{p} + \mathbf{r}\,\mathbf{q}$,
3. $(\mathbf{r}^\star)^\star = \mathbf{r}^\star$, $(\varepsilon + \mathbf{r})\,(\varepsilon + \mathbf{r})^\star \models\!\models \mathbf{r}^\star$,
4. $(\mathbf{p} + \mathbf{q})^\star \models\!\models (\mathbf{p}^\star\mathbf{q}^\star)^\star \models\!\models (\mathbf{p}^\star + \mathbf{q})^\star \models\!\models (\mathbf{p}^\star\mathbf{q})^\star\mathbf{p}^\star$,
5. $\mathbf{r}^\star \models\!\models \varepsilon + \mathbf{r}\mathbf{r}^\star$,
6. $\mathbf{r}\mathbf{r}^\star \models\!\models \mathbf{r}^\star\mathbf{r}$,
7. $(\mathbf{p}\,\mathbf{q})^\star \models\!\models \varepsilon + \mathbf{p}\,(\mathbf{q}\,\mathbf{p})^\star\,\mathbf{q}$,
8. $(\mathbf{p}\,\mathbf{q})^\star\,\mathbf{p} \models\!\models \mathbf{p}\,(\mathbf{q}\,\mathbf{p})^\star$,
9. $\mathbf{r}\,\emptyset \models\!\models \emptyset \models\!\models \emptyset\,\mathbf{r}$.

Poznámka. Pomocí výše uvedených vztahů není vždy jednoduché zjistit, zda dva regulární výrazy jsou ekvivalentní. Algoritmicky se daný problém řeší např. tak, že ke každému regulárnímu výrazu $\mathbf{r}$ a $\mathbf{q}$ najdeme konečný deterministický automat, které přijímají jazyky reprezentované těmito regulárními výrazy. Získané automaty následně zredukujeme. Jsou-li redukované automaty isomorfní, jsou regulární výrazy $\mathbf{r}$ a $\mathbf{q}$ ekvivalentní, jinak ekvivalentní nejsou.

2.5.8 Konstrukce konečného automatu k regulárnímu výrazu. Je dán regulární výraz $\mathbf{r}$. Sestrojit k němu konečný deterministický automat, který přijímá jazyk $L_{\mathbf{r}}$ reprezentovaný regulárním výrazem $\mathbf{r}$, můžeme dvěma způsoby:

I. Pomocí důkazu Kleeneho věty. Regulární výraz $\mathbf{r}$ rozdělíme na několik jednoduchých podvýrazů, pro které umíme jednoduše sestrojit DFA. Nyní pomocí metod z důkazu věty 2.4.6 sestrojíme nedeterministický automat M_1 (je-li to nutné, tak s ε přechody), který přijímá jazyk $L_{\mathbf{r}}$. Podmnožinovou konstrukcí k němu sestrojíme deterministický automat M_2 přijímající stejný jazyk. Je-li potřeba, M_2 redukujeme a tím dostaneme hledaný automat.

II. Přímou metodou. Jednotlivé vstupní symboly regulárního výrazu $\mathbf{r}$ očíslujeme. Dále zjistíme

1. všechny očíslované symboly, kterými může některé slovo jazyka $L_{\mathbf{r}}$ začínat;
2. všechny dvojice očíslovaných symbolů, které mohou po sobě následovat v některém slově jazyka $L_{\mathbf{r}}$;
3. všechny očíslované symboly, kterými může některé slovo jazyka $L_{\mathbf{r}}$ končit;
4. zda prázdné ε slovo leží v jazyce $L_{\mathbf{r}}$.

Stavy nedeterministického automatu $\overline{M}$ jsou všechny očíslované symboly $\mathbf{r}$ a počáteční stav s. Stavový diagram automatu $\overline{M}$ dostaneme takto:

Ze stavu s vede hrana do každého stavu a_i, $a \in \Sigma$, který je vyjmenován v bodě 1. Hrana je v tomto případě označena symbolem a.

Ze stavu a_i do stavu b_j $(a, b \in \Sigma)$ vede hrana označená b právě tehdy, když dvojice $a_i b_j$ byla vyjmenovaná v bodě 2.

Koncové stavy jsou všechny a_k, které jsou vyjmenované v bodě 3. Jestliže prázdné slovo ε patří do jazyka $L_{\mathbf{r}}$, je s nejen počátečním ale i koncovým stavem.

DFA přijímající jazyk $L_{\mathbf{r}}$ dostaneme podmnožinovou konstrukcí z automatu $\overline{M}$ a případnou redukcí.

Obě metody ukážeme na příkladě.

2.5.9 Příklad. Pro regulární výraz

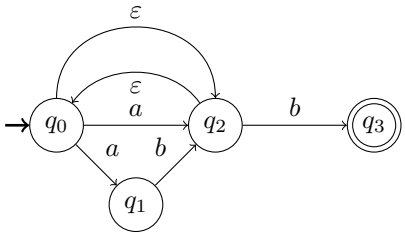
$$\mathbf{r} = (\mathbf{a} + \mathbf{ab})^\star \mathbf{b}.$$

Řešení.

I. metoda. Regulární výraz $\mathbf{r}$ rozdělíme na $\mathbf{r}_1 = (\mathbf{a} + \mathbf{ab})$, $\mathbf{r}_2 = \mathbf{b}$. Pak

$$\mathbf{r} = \mathbf{r}_1^\star \mathbf{r_2}.$$

Pak jeden ε-NFApřijímající jazyk $L_{\mathbf{r}}$ je dán stavovým diagramem



a tabulkou

		ε	a	b
$\rightarrow$	q_0	$\{q_2\}$	$\{q_1, q_2\}$	$\emptyset$
	q_1	$\emptyset$	$\emptyset$	$\{q_2\}$
	q_2	$\{q_0\}$	$\emptyset$	$\{q_3\}$
$\leftarrow$	q_3	$\emptyset$	$\emptyset$	$\emptyset$

Víme, že $\varepsilon - UZ(q_0) = \{q_0, q_2\} = \varepsilon - UZ(q_2)$, $\varepsilon - UZ(q_1) = \{q_1\}$ a $\varepsilon - UZ(q_3) = \{q_3\}$.

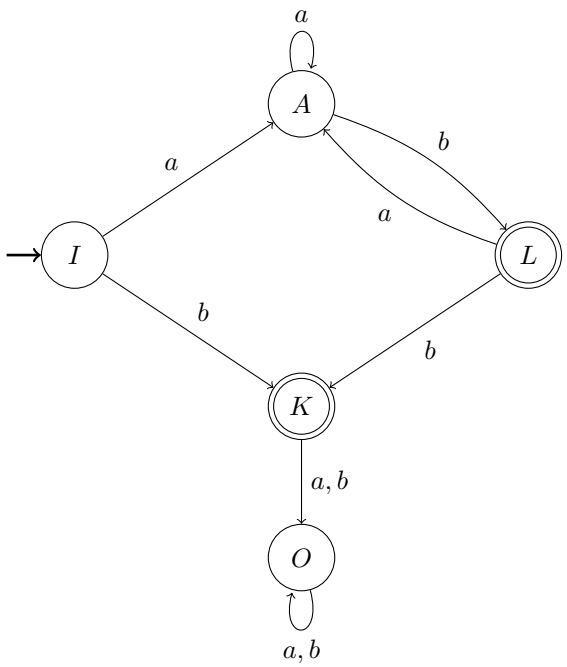
Proto podmnožinovou konstrukcí dostaneme DFA:

		a	b
$\to$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_3\}$
	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2, q_3\}$
$\leftarrow$	$\{q_3\}$	$\emptyset$	$\emptyset$
$\leftarrow$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2\}$	$\{q_3\}$
	$\emptyset$	$\emptyset$	$\emptyset$

Automat je redukovaný a po přejmenování stavů má výsledný DFA tabulku:

		a	b
$\to$	I	A	K
	A	A	L
$\leftarrow$	K	O	O
$\leftarrow$	L	A	K
	O	O	O

a stavový diagram

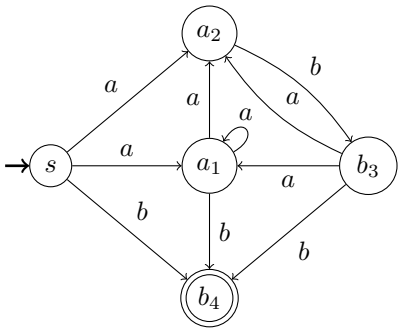


II. metoda. Očíslujeme jednotlivé výskyty vstupních symbolů:

$$(a_1 + a_2 b_3)^\star b_4.$$

1. Slovo z jazyka $L_{\mathbf{r}}$ může začínat: a_1, a_2 nebo b_4.
2. Po sobě mohou následovat (ve slově z $L_{\mathbf{r}}$): a_1a_1, a_1a_2, a_1b_4, a_2b_3, b_3a_1, b_3a_2 a b_3b_4.
3. Slovo z $L_{\mathbf{r}}$ končí b_4.
4. Prázdné slovo neleží v $L_{\mathbf{r}}$.

Zkonstruujeme nedeterministický automat podle II. metody. Ten má stavový diagram



Podmnožinovou konstrukcí dostaneme automat s tabulkou

		a	b
$\rightarrow$	s	$\{a_1, a_2\}$	$\{b_4\}$
	$\{a_1, a_2\}$	$\{a_1, a_2\}$	$\{b_3, b_4\}$
$\leftarrow$	$\{b_4\}$	$\emptyset$	$\emptyset$
$\leftarrow$	$\{b_3, b_4\}$	$\{a_1, a_2\}$	$\{b_4\}$
	$\emptyset$	$\emptyset$	$\emptyset$

Automat už je redukovaný a po přejmenování stavů dostáváme stejný automat jako metodou I.

2.5.10 Regulární výraz reprezentující jazyk $L(M)$. Je dán deterministický automat M. Regulární výraz reprezentující jazyk $L(M)$ dostaneme buď postupem z důkazu Kleeneovy věty nebo úpravami stavového diagramu. Druhý způsob popíšeme:

Ke stavovému diagramu přidáme dva stavy; a to nový počáteční stav *start* a nový koncový stav fin. Dále přidáme hrany označené prázdným slovem ε ze stavu *start* do počátečního stavu automatu a z každého koncového stavu do stavu fin. Označení hran chápeme jako regulární výrazy.

S takto vzniklým orientovaným grafem provádíme úpravy:

a) (*Odstranění paralelních hran.*) Jsou-li v grafu dvě paralelní hrany označené regulárními výrazy $\mathbf{r_1}$ a $\mathbf{r_2}$, nahradíme je jednou hranou označenou regulárním výrazem $(\mathbf{r_1} + \mathbf{r_2})$.

b) (*Odstranění smyčky.*) Je-li v grafu smyčka ve vrcholu q označená regulárním výrazem $\mathbf{r}$, pak smyčku odstraníme a označení $\mathbf{s}$ libovolné hrany začínající v q změníme na $\mathbf{r}^\star\mathbf{s}$.

c) (*Odstranění vrcholu q.*) Jestliže v q není smyčka, odstraníme jej tímto postupem: Každou dvojici hran e_1 a e_2, kde e_1 končí ve vrchollu q a e_2 začíná ve vrcholu q, tj. $KV(e_1) = q = PV(e_2)$, nahradíme hranou e s $PV(e) = PV(e_1)$ a $KV(e) = KV(e_2)$. Je-li hrana e_1 označena regulárním výrazem $\mathbf{r}$ a hrana e_2 regulárním výrazem $\mathbf{s}$, je hrana e označena regulárním výrazem $\mathbf{r}\,\mathbf{s}$.

Končíme tehdy, když graf má jen dva vrcholy, a to *start* a fin, a jedinou hranu, a to ze *start* do fin. Označení této hrany je hledaný regulární výraz.

Poznamenejme, že vytvořený regulární výraz obecně závisí na pořadí, ve kterém odstraňujeme stavy (vrcholy). Můžeme proto dostat různé regulární výrazy, ale tyto výrazy jsou ekvivalentní.

2.6 Praktické použití regulárních výrazů.

Teorie regulárních jazyků patří k úhelným kamenům informatiky. Praxe často bývá méně učesaná než teorie a s regulárními jazyky to dopadlo podobně. Proto praktickému použití regulárních jazyků, regulárních výrazů a konečných automatů věnujeme samostatnou sekci.[1]

[1] Tuto sekci napsal Jiří Demel.

2.6.1 Kde lze potkat regulární jazyk. Nejčastější použití regulárních jazyků není ani moc nápadné. Ve většině prakticky používaných počítačových jazyků lze rozlišit části (podřetězce), tzv. lexikální elementy, též tokeny, které lze pokládat za slova z regulárního jazyka. Např. v běžných programovacích jazycích to jsou zápisy konstant, zápisy identifikátorů, klíčová slova, relační znaménka apod. Program, který daný jazyk analyzuje (syntaktický analyzátor, parser), obvykle obsahuje proceduru, které se říká lexikální analyzátor a která funguje jako konečný automat. Lexikální analyzátor při každém svém vyvolání přečte ze vstupního souboru nejdelší část, která tvoří jeden lexikální element a volajícímu programu vrátí jeho typ a hodnotu. Zbytek syntaktického analyzátoru může díky tomu být jednodušší, neboť pracuje už jen s lexikálními elementy.

Další významné použití regulárních jazyků je pro uživatele viditelnější – je to vyhledávání výskytů slov daných regulárním výrazem.

Program grep opisuje na výstup ty řádky vstupního souboru, které obsahují podřetězec popsaný regulárním výrazem.

Mnohé editory dovedou při operacích „vyhledej“, popř. „vyhledej a nahraď“ hledat podle daného regulárního výrazu. Toto byla mimochodem nejstarší aplikace regulárních výrazů.

V některých programovacích jazycích (Perl, AWK, Ruby) je vyhledávání (a popř. náhrada) vzoru daného regulárním výrazem součástí jazyka. V mnoha dalších jazycích jsou k dispozici knihovny.

Vyhledávání vzorů se také využívá v antivirech a v antispamových filtrech.

2.6.2 Zápis regulárních výrazů. Formální algebraický zápis regulárních výrazů, jak byl definován v 2.5.2, je vhodný pro teoretické úvahy. Pro praktické účely se používá syntax, která zejména zjednodušuje zápis množin znaků a dovoluje flexibilně vyjádřit počty opakování.

- Regulární výraz `.` (tečka) reprezentuje libovolný znak (přesněji: jazyk tvořený všemi jednoznakovými slovy).
- Pro zápis obecnější množiny znaků se používá konstrukce s hranatými závorkami. Příklady:
  - `[abc]` reprezentuje libovolný jeden ze znaků `a`, `b`, `c`.
  - `[a-f]` reprezentuje libovolný jeden ze znaků `a` až `f`.
  - `[0-9]` reprezentuje libovolnou číslici.
  - `[^abc]` reprezentuje libovolný znak jiný než `a`, `b` a `c`.
  - `[^a-f]` reprezentuje libovolný znak jiný než `a` až `f`.
- Znak `|` („svislítko“, v UNIXu roura) značí alternativu, tedy operaci, která se v algebraickém zápisu značí $+$.
- Hvězdička (Kleeneho operace) se zapisuje v řádce, nikoli jako exponent.
- Znak `+` označuje jedno nebo více opakování předcházejícího. Tedy `r+` $= \mathbf{rr}*$.
- Znak `?` označuje žádný nebo jeden výskyt předchozího. Tedy `r?` označuje to, co by se v algebraickém zápise vyjádřilo jako $(\varepsilon + \mathbf{r})$.
- Pro obecnější počty opakování předchozího znaku nebo výrazu v závorkách se používá konstrukce se složenými závorkami. Příklady:
  - `r{3}` značí přesně 3 opakování,
  - `r{3,}` značí 3 nebo více opakování,
  - `r{3,5}` značí 3 až 5 opakování.
- Znak `^` na začátku výrazu značí začátek textu nebo řádky.
- Znak `$` na konci výrazu značí konec textu nebo řádky.
- Některé znaky (např. `[]^$.*`) mají v praktických regulárních výrazech speciální význam popsaný výše. Potřebujeme-li některý z těchto znaků použít ve výrazu jako obyčejný znak, je třeba před tento znak zapsat obrácené lomítko, tj. takzvaně znak escapovat. Např. pro vyhledání doménového jména `fel.cvut.cz` je třeba escapovat tečky, tedy správný výraz je `fel\.cvut\.cz`, jinak by výrazu vyhovovalo také nesprávné `felxcvutbcz`.
- Některé speciální znaky (např. `{}()+?`) se bohužel v praxi používají nejednotným způsobem. V některých implementacích je nutno tyto znaky escapovat, aby měly speciální význam, jinde je naopak nutno escapovat, aby fungovaly jako obyčejné znaky bez speciálního významu.

Praktické regulární výrazy se nejčastěji používají pro hledání podslova v prohledávaném textu. Tedy např. výraz `sen` bude nalezen v textu `nadnesený`. Chceme-li hledat `sen` samotný, musíme použít výraz `^sen$`. Výraz reprezentující prázdné slovo je `^$`.

Výše uvedený popis praktických regulárních výrazů zdaleka není úplný. Pro zevrubnější poučení doporučuji text Pavla Satrapy `http://www.nti.tul.cz/~satrapa/docs/regvyr/`, kde je i tabulka odlišností syntaxe regulárních výrazů v různých implementacích.

2.6.3 Implementace regulárních výrazů jsou zhruba tří typů.

Simulace DFA je nejrychlejší. Je-li DFA, který rozpoznává jazyk daný regulárním výrazem, již sestrojen, pak samotné prohledávání vyžaduje čas $O(t)$, kde t je délka prohledávaného textu. Konstrukce DFA k danému regulárnímu výrazu ovšem vyžaduje nějaký čas a paměť, je to tedy něco jako investice, která se ne vždy vyplatí. Pro hledání v rozsáhlých textech je simulace DFA nejrychlejší ze všech tří možností.

- **Stav reprezentován návěštím v programu.** Program přečte znak a podle toho, co přečetl, pomocí příkazů `if` a `case` skočí na návěští, které reprezentuje následující stav DFA. Vhodné, pokud je regulární jazyk předem znám. Používá se v lexikálních analyzátorech.
- **Stav reprezentován obsahem proměnné.** Přechodová funkce může být dána ve formě dvourozměrného pole, kde přečtený znak a dosavadní stav se použijí jako indexy a prvek pole obsahuje nový stav.

Simulace NFA probíhá tak, že vždy po přečtení znaku ze vstupu vypočteme množinu všech stavů, do nichž se NFA tímto znakem může dostat. Tedy z předchozí množiny „aktivních“ stavů počítáme novou množinu. Automat sice je nedeterministický, ale díky konečnému počtu stavů lze snadno paralelně sledovat všechny možné větve výpočtu. Není tedy třeba žádný bactracking.[2] Simulace NFA pracuje v čase $O(qt)$, kde q je počet stavů NFA a t je délka prohledávaného textu. Vlastní prohledávání je tedy pomalejší než v případě DFA, ale počáteční konstrukce NFA je jednodušší a rychlejší než konstrukce DFA.

Existuje varianta, kdy během simulace činnosti NFA je postupně vytvářen DFA.

Rekurzivní hledání (backtracking) nevyužívá konečné automaty, namísto toho porovnává prohledávaný text přímo s regulárním výrazem a když nenajde možnost pokračovat, vrací se a zkouší porovnat prohledávaný text s další částí regulárního výrazu. Na některých regulární výrazech pracuje backtracking značně neefektivně. Např. `(.*):(.*):(.*):(.*):(.*)` pro hledání pěti údajů oddělených čtyřmi dvojtečkami nebo je-li hledán výraz `a?a?a?a?a?aaaaa` v textu `aaaaa`.

Výhodou rekurzivního hledání je rychlejší start, ale zejména možnost rozšířit množinu vyhledávaných slov za hranice regulárních jazyků pomocí tzv. zpětných referencí. Každý pár kulatých závorek, kromě toho, že seskupuje to, co je uvnitř, zároveň vymezuje část vstupního textu, která se shodovala s vnitřkem závorky. Na takto zapamatovanou část vstupního textu se pak lze odvolat zápisem `\`*`číslo`*. Tedy např. ve výrazu `(.*)\1` zastupuje `\1` text, který byl reprezentován výrazem `(.*)`. Výraz `(.*)\1` tedy reprezentuje jazyk tvořený všemi zdvojenými slovy, např. `bubu` nebo `holahola`. Tento jazyk není regulární, není dokonce ani bezkontextový.

Historicky první implementace regulárních výrazů (popsána v článku z roku 1968) byla založena na paralelní simulaci NFA. Jejím autorem byl Ken Thomson, jeden z otců UNIXu. Snad proto se to v UNIXu regulárními výrazy jen hemží. Otcové UNIXu teorii regulárních jazyků a konečných automatů dobře znali. První open source implementace však byla založena na backtrackingu.

[2] Pozor, v jinak velmi pěkném textu Pavla Satrapy `http://www.nti.tul.cz/~satrapa/docs/regvyr/` jsou pojmy nedeterministický, deterministický a rekurzivní použity poněkud nešťastně: slovem deterministický je označena simulace NFA, nedeterministický stroj znamená rekurzivní hledání a možnost použít DFA tam vůbec není zmíněna.

Díky otevřenosti se rozšířila mimo jiné do Perlu a odtud skrze PCRE (Perl Compatible Regular Expressions) do knihoven řady dalších jazyků.

Detailní popis včetně příkladu implementace a včetně historických podrobností lze najít v článku `http://swtch.com/~rsc/regexp/regexp1.html`

2.7 Další uzávěrové vlastnosti třídy regulárních jazyků

Z předchozích přednášek víme, že třída regulárních jazyků je uzavřena na sjednocení, průnik, doplněk, zřetězení, Kleeneho operaci $\star$ a reverzi. Ukážeme ještě další operace s jazyky, na které je třída regulárních jazyků uzavřena.

2.7.1 Homomorfismus. Jsou dány dvě abecedy Σ, Γ a zobrazení h, které každému písmenu $a \in \Sigma$ přiřadí slovo $h(a)$ nad abecedou Γ.

Zobrazení h rozšíříme na zobrazení, které každému slovu $u \in \Sigma^\star$ přiřazuje slovo nad Γ takto:

- $h(\varepsilon) = \varepsilon$,
- $h(u\,a) = h(u)\,h(a)$.

Poznamenejme, že se jedná o rozšíření zobrazení $h\colon \Sigma \to \Gamma^\star$ na $h\colon \Sigma^\star \to \Gamma^\star$ tak, že obraz slova je zřetězením obrazů jednotlivých písmen slova.

Definice. *Obraz jazyka L* (nad Σ) *v homomorfismu h* je

$$h(L) = \{h(w) \mid w \in L\}.$$

□

Příklad: Uvažujme abecedy $\Sigma = \{0,1\}$, $\Gamma = \{a,b\}$ a zobrazení $h(0) = ab^2$, $h(1) = bab$. Pak $h(010) = ab^2babab^2 = ab^2(ba)^2b^2$. Homomorfní obraz jazyka $L = \{10^k \mid k \geq 0\}$ je $h(L) = \{bab(ab^2)^k \mid k \geq 0\}$.

2.7.2 Substituce. Obecnější pojem než homomorfismus je tzv. substituce. Jsou dány dvě abecedy Σ, Γ a zobrazení σ, které každému písmenu $a \in \Sigma$ přiřadí jazyk nad abecedou Γ.

Analogicky jako pro homomorfismus zobrazení σ rozšíříme na zobrazení, které každému slovu $u \in \Sigma^\star$ přiřazuje jazyk nad Γ takto:

- $\sigma(\varepsilon) = \{\varepsilon\}$,
- $\sigma(u\,a) = \sigma(u)\,\sigma(a)$.

I zde se jedná o rozšíření zobrazení σ takové, že obrazem slova (zřetězení písmen) je zřetězení obrazů písmen.

Definice. *Obraz jazyka L* (nad Σ) *v substituci σ* je

$$\sigma(L) = \bigcup \{\sigma(w) \mid w \in L\}.$$

□

Příklad: Uvažujme abecedy $\Sigma = \{0,1\}$, $\Gamma = \{a,b\}$ a zobrazení $\sigma(0) = L_1 = \{a^n \mid n \geq 0\}$, $\sigma(1) = L_2 = \{b^n \mid n \geq 0\}$. Pak $\sigma(01) = L_1\,L_2 = \{a^nb^m \mid n,m \geq 0\}$.

2.7.3 Inverzní homomorfismus. Je dán homomorfismus h, $h\colon \Sigma^\star \to \Gamma^\star$. Pak *inverzní homomorfismus* h^{-1} je $h^{-1}\colon \Gamma^\star \to \Sigma^\star$, kde $h^{-1}(L) = \{u \in \Sigma^\star \mid h(u) \in L\}$. □

Příklad. Uvažujme jazyk L nad abecedou $\Gamma = \{0,1\}$ popsaný regulárním výrazem $(\mathbf{00} + \mathbf{1})^\star$ a homomorfismus h určený $h(a) = 01$ a $h(b) = 10$.

Pak $h^{-1}(L)$ je jazyk nad abecedou $\Sigma = \{a,b\}$ popsaný regulárním výrazem $(\mathbf{ba})^\star$. (Ověřte si.)

2.7.4 Věta. Je dána substituce σ z $\Sigma^\star$ do jazyků nad abecedou Γ. Jestliže každý jazyk $\sigma(a)$ je regulární a jestliže L je regulární jazyk nad abecedou Σ, pak jazyk $\sigma(L)$ je regulární jazyk nad abecedou Γ. □

Myšlenka důkazu. Předpokládejme, že jazyk L je popsán regulárním výrazem $\mathbf{r}$ a každý z jazyků $\sigma(a)$ regulárním výrazem $\mathbf{r_a}$ pro každé $a \in \Sigma$. Pak regulární výraz pro jazyk $\sigma(L)$ dostaneme tak, že za každé $\mathbf{a}$, $a \in \Sigma$, v regulárním výrazu $\mathbf{r}$ „dosadíme“ výraz $\mathbf{r}_a$.

Důsledek. Je-li h homomorfismus a L regulární jazyk, pak jazyk $h(L)$ je také regulární jazyk. □

Důkaz. Tvrzení vyplývá z předchozí věty, protože jednoprvkový jazyk je vždy regulární a navíc homomorfismus je zvláštní případ substituce.

2.7.5 Věta. Jestliže h je homomorfismus, $h\colon \Sigma^\star \to \Gamma^\star$ a L je regulární jazyk nad abecedou Γ, pak inversní obraz $h^{-1}(L)$ je také regulární (nad Σ).

Myšlenka důkazu. Máme konečný automat $M_1 = (Q_1, \Gamma, \delta_1, q_1, F_1)$, který přijímá jazyk L. Zkonstruujeme konečný automat M, který přijme $h^{-1}(L)$. M má stejnou množinu stavů, stejný počáteční stav i stejnou množinu koncových stavů. Jeho přechodová funkce δ je definována: přechodová funkce M zobrazí stav p při vstupním symbolu a do takového stavu, kam v M_1 přejde stav p při přečtení slova $h(a)$.

Přesněji pro $a \in \Sigma$, $p \in Q$ je

$$\delta(p, a) = \delta_1^\star(p, h(a)).$$

Není těžké ukázat, že M přijme přesně ta slova $w \in \Sigma^\star$, pro která M_1 přijme $h(w)$.

2.8 Algoritmická řešitelnost úloh pro regulární jazyky

Pro následující otázky týkající se konečných automatů a jimi přijímaných jazyků existují algoritmy, které dají správnou odpověď.

1. Pro daný konečný automat M (ať deterministický nebo nedeterministický) a slovo $w \in \Sigma^\star$ rozhodnout, zda $w \in L(M)$.

   Zdůvodnění. Jedná se o jednoduché zjištění, zda ve stavovém diagramu existuje sled označený slovem w, který začíná v počátečním stavu a končí v koncovém stavu.

2. Pro daný konečný automat M (ať deterministický nebo nedeterministický) rozhodnout, zda $L(M) \neq \emptyset$.

   Zdůvodnění. Jedná se o jednoduché zjištění, zda ve stavovém diagramu je některý koncový stav dosažitelný z počátečního stavu.

3. Pro daný konečný automat M rozhodnout, zda $L(M) = \Sigma^\star$.

   Zdůvodnění. Stačí vytvořit odpovídající DFA přijímající jazyk $L(M)$ a tento DFA zredukovat. $L(M) = \Sigma^\star$ právě tehdy, když se redukovaný automat skládá z jediného stavu, který je současně počáteční i koncový.

4. Pro dva konečné automaty M_1 a M_2 rozhodnout, zda $L(M_1) = L(M_2)$.

   Zdůvodnění. Stačí zredukovat oba automaty a (v tomto případě jednoduchým algoritmem) zjistit, zda jsou isomorfní.

Pro „obecnější“ třídy jazyků už některé z výše položených otázek algoritmicky řešitelné nejsou, jak uvidíme později.

2.8.1 Tvrzení. Je dán deterministický konečný automat M s n stavy. Pak

1. Jazyk $L(M)$ je neprázdný právě tehdy, když M přijímá slovo w délky $|w| < n$.
2. Jazyk $L(M)$ je nekonečný právě tehdy, když M přijímá slovo v délky $n \leq |v| < 2n$.

□

Důkaz. 1. Jestliže M přijímá slovo w délky $|w| < n$, pak je $L(M)$ jistě neprázdný.

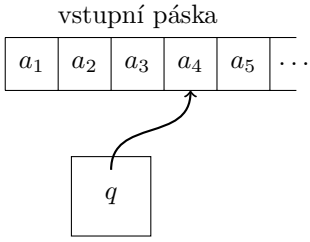
Předpokládejme, že $L(M)$ je neprázdný. Pak existuje orientovaný sled ve stavovém diagramu označený w, který začíná v počátečním stavu a končí v koncovém stavu. Každý orientovaný sled obsahuje orientovanou cestu se stejným počátečním i koncovým vrcholem. Označme u slovo, které označuje tuto orientovanou cestu. Pak $u \in L(M)$ a délka u je nejvýše $n-1$ (ano, každá cesta v grafu s n vrcholy má nejvýše $n-1$ hran).

2. Předpokládejme, že jazyk $L(M)$ je nekonečný. Pak obsahuje slovo w délky větší než $n-1$. Jestliže $|w| < 2n$, máme hledané slovo. V opačném případě uvažujme orientovaný sled z počátečního stavu do koncového stavu označený w. Tento sled musí obsahovat cyklus. Označme C_1 první cyklus na tomto sledu. Odstraňme tento cyklus a dostaneme orientovaný sled kratší délky, který má stejný počáteční i koncový vrchol. Jestliže tento sled má délku menší než $2n$, slovo, které tento sled označuje, je hledané slovo v. Jestliže je stále příliš dlouhé, pokračujeme s odstraněním (třeba) prvního cyklu. Jakmile má slovo délku menší než $2n$, postup ukončíme. Protože každý cyklus má délku maximálně n, tímto postupem dostaneme slovo délky maximálně $n-1+n$, tj. menší než $2n$ a větší než n.

Jestliže existuje $v \in L(M)$ s $n \leq |v| < 2n$, pak orientovaný sled určený tímto slovem obsahuje cyklus. Položme x slovo, které označuje tento cyklus. Pak $v = wxy$ a pro každé $i = 2, 3, \ldots$ platí $wx^iy \in L(M)$, proto je $L(M)$ nekonečný.

2.9 Dvousměrné automaty

Pouze informativně uvedeme ještě jeden typ konečných automatů, který, podobně jako nedeterministické automaty, přijímá pouze regulární jazyky. Pro lepší názornost připomeneme ještě jeden způsob, jak se dívat na konečné automaty. Tento způsob byl již zmíněn v první přednášce a je znázorněn na následujícím obrázku. (Poznamenejme, že ho využijeme později při zavedení zásobníkových automatů.)



Pro deterministické automaty víme, že jejich chování je určeno přechodovou funkcí δ, která každému stavu q a každému vstupnímu symbolu a přiřazuje následující stav $\delta(q, a)$. Práce DFA M nad slovem $w = a_1a_2 \ldots a_k$ spočívá v tom, že M ve stavu q_0 přečte první symbol a_1, přejde do stavu $q_1 = \delta(q_0, a_1)$, ve stavu q_1 přečte symbol a_2, přejde do stavu $q_2 = \delta(q_1, a_2) = \delta^\star(q_0, a_1a_2)$, atd. až do té doby, než "přečte" celé slovo $w = a_1a_2 \ldots a_k$.

Můžeme si proto představit DFA jako zařízení, které se skládá z řídící jednotky nacházející se v jednom ze stavů, pásky, na které je zapsáno vstupní slovo $w = a_1a_2 \ldots a_k$ a hlavy, která v každém okamžiku čte obsah jednoho políčka pásky. Je-li automat ve stavu q a čte vstupní symbol a_i, tak se řídící jednotka přesune do stavu $p = \delta(q, a_i)$ a hlava se posune o jedno políčko doprava a tudíž čte vstupní symbol a_{i+1}. Zda je slovo w přijato se rozhodne podle toho, zda řídící jednotka po přečtení celého slova skončí v některém koncovém stavu nebo ne. Obdobně si můžeme představit takto i práci NFA.

Dvousměrné automaty se od DFA liší tím, že hlava se po pásce nemusí pohybovat jen doprava, ale může též doleva (přitom ale nemění obsah pásky).

2.9.1 Dvousměrné automaty. Dvousměrný deterministický automat je pětice $(Q, \Sigma, \delta, q_0, F)$, kde Q, Σ, q_0 a F mají stejný význam jako pro deterministické konečné automaty a přechodová funkce δ je zobrazení

$$\delta\colon Q \times \Sigma \longrightarrow Q \times \{R, L\},$$

kde

- $\delta(q, a) = (p, R)$ znamená, že po přečtení vstupního písmene a ve stavu q se automat přesune do stavu p a hlava se posune na vstupní pásce o jedno políčko doprava;
- $\delta(q, a) = (p, L)$ znamená, že po přečtení vstupního písmene a ve stavu q se automat přesune do stavu p a hlava se posune na vstupní pásce o jedno políčko doleva.

Slovo w je přijato dvousměrným automatem právě tehdy, když automat začne práci v počátečním stavu q_0, hlava čte první písmeno vstupního slova w a automat při práci opustí pravý okraj vstupního slova a je při tom v některém z koncových stavů. □

2.9.2 Věta. Dvousměrné automaty přijímají pouze regulární jazyky. □

Větu nedokazujeme, důkaz je technicky náročný. Jeho myšlenka spočívá v tom, že se ukáže, že ke každému dvousměrnému automatu existuje DFA, který přijímá stejný jazyk; jedna z možností, jak to udělat, je použít Nerodovu větu.

Poznamenejme ale, že má-li dvousměrný automat n stavů, pak jemu odpovídající deterministický automat může mít až n^n.

Kapitola 3

Gramatiky

V předcházejících kapitolách jsme se zabývali konečnými automaty a jazyky, které tyto automaty přijímají. Nyní zavedeme nový „nástroj“, který nám umožní popsat větší třídy jazyků než jen regulární jazyky. Tímto nástrojem bude gramatika.

3.1 Hierarchie gramatik

Nejprve uvedeme příklad:

Příklad. V programovacích jazycích se často vyskytují definice podobné definici typu číslo v Backus-Naurově formě:

- ⟨číslo⟩ ::= ⟨číslo bez zn.⟩ | + ⟨číslo bez zn.⟩ | − ⟨číslo bez zn.⟩
- ⟨číslo bez zn.⟩ ::= ⟨číslice⟩ | ⟨číslo bez zn.⟩⟨číslice⟩
- ⟨číslice⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Jedná se o speciální příklad gramatiky: Symbolům ⟨číslo⟩, ⟨číslo bez zn.⟩ a ⟨číslice⟩ říkáme neterminály a symbolům $\{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ terminály. Cílem je podle „pravidel“ výše „vygenerovat“ čísla, ta představují terminální slova.

3.1.1 Definice. *Gramatika* je uspořádaná čtveřice $\mathcal{G} = (N, \Sigma, S, P)$, kde

- N je konečná množina tzv. *neterminálů*;
- Σ je konečná neprázdná množina tzv. *terminálů*, platí $N \cap \Sigma = \emptyset$;
- $S \in N$ je *startovací symbol*;
- P je konečná množina pravidel typu $\alpha \to \beta$, kde α a β jsou slova nad $N \cup \Sigma$ taková, že α obsahuje alespoň jeden neterminál.

□

V našem případě je $S =$ ⟨číslo⟩, $A =$ ⟨číslo bez zn.⟩ a $B =$ ⟨číslice⟩ a jedná se o gramatiku, kde

$$N = \{S, A, B\}, \quad \Sigma = \{+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

a pravidla P jsou

- $S \to A \mid +A \mid -A$
- $A \to B \mid BA$
- $B \to 0 \mid 1 \mid \ldots \mid 9$.

Použili jsme zkrácený zápis 15 pravidel

- $S \to A, \quad S \to +A, \quad S \to -A,$
- $A \to B, \quad A \to BA,$
- $B \to 0, \quad B \to 1, \quad \ldots, \quad B \to 9.$

3.1.2 Přímé odvození. Zhruba řečeno, ze slova γ přímo odvodíme nějaké slovo δ jestliže ve slově γ najdeme některý výskyt podslova α, které tvoří levou stranu pravidla $\alpha \to \beta$ z P. Slovo δ navíc dostaneme tak, že zvolený výskyt α (v γ) nahradíme slovem β (tj. pravou stranou pravidla). Formálně:

Definice. Je dána gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Řekneme, že δ se *přímo odvodí* (též *přímo přepíše*) z γ, jestliže existuje v P pravidlo $\alpha \to \beta$ a slova $\varphi, \psi \in (N \cup \Sigma)^\star$ taková, že $\gamma = \varphi\,\alpha\,\psi$ a $\delta = \varphi\,\beta\,\psi$.

Tento fakt zapisujeme $\gamma \Rightarrow_{\mathcal{G}} \delta$. □

Například: máme přímé odvození $+BA \Rightarrow +1A$, protože jsme ve slově $+BA$ použili pravidlo $B \to 1$ a slovem 1 jsme nahradili neterminál B.

3.1.3 Odvození, derivace. Odvození je nyní posloupnost přímých odvození. Jedná se vlastně o reflexivní a tranzitivní uzávěr relace $\Rightarrow_{\mathcal{G}}$. Formálně:

Definice. Je dána gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Řekneme, že δ se *odvodí* z γ, jestliže

- buď $\gamma = \delta$,
- nebo existuje posloupnost přímých odvození

$$\gamma = \gamma_1 \;\Rightarrow_{\mathcal{G}}\; \gamma_2 \;\Rightarrow_{\mathcal{G}} \ldots \Rightarrow_{\mathcal{G}}\; \gamma_k = \delta.$$

Tento fakt značíme $\gamma \Rightarrow^\star_{\mathcal{G}} \delta$ a této konečné posloupnosti říkáme *derivace*. □

3.1.4 Jazyk generovaný gramatikou. Definice. Řekneme, že slovo $w \in \Sigma^\star$ je *generováno* gramatikou $\mathcal{G}$, jestliže existuje derivace $S \;\Rightarrow^\star_{\mathcal{G}}\; w$.

Jazyk $L(\mathcal{G})$ *generovaný* gramatikou $\mathcal{G}$ se skládá ze všech slov generovaných gramatikou $\mathcal{G}$, tj.

$$L(\mathcal{G}) = \{w \in \Sigma^\star \mid S \;\Rightarrow^\star_{\mathcal{G}}\; w\}.$$

□

3.1.5 Konvence.

- Neterminály značíme obvykle velkými písmeny $A, B, X, Y, \ldots$.
- Terminály značíme obvykle malými písmeny ze začátku abecedy $a, b, c, d, \ldots$.
- Slova z $(N \cup \Sigma)^\star$ obvykle značíme řeckými písmeny α, β, ...
- Terminální slova, tj. slova z $\Sigma^\star$, značíme malými písmeny z konce abecedy $u, w, x, y, \ldots$.

Obvykle v textu vynecháváme jméno gramatiky, je-li z kontextu jasné, o jakou gramatiku se jedná. Píšeme proto $\Rightarrow$ a $\Rightarrow^\star$ místo $\Rightarrow_{\mathcal{G}}$ a $\Rightarrow^\star_{\mathcal{G}}$.

3.1.6 Chomského hierarchie. Podle podmínek, které klademe na pravidla dané gramatiky, rozlišujeme gramatiky a jimi generované jazyky na:

- *Gramatiky typu 0* jsou gramatiky tak, jak jsme je zavedli v odstavci 3.1.1. Jazyky generované gramatikami typu 0 se nazývají *jazyky typu 0*.
- *Gramatiky typu 1*, též *kontextové gramatiky*, jsou takové gramatiky, kde každé pravidlo v P je tvaru

  $$\alpha\,A\,\beta \to \alpha\,\gamma\,\beta,$$

  kde $\alpha, \beta, \gamma \in (N \cup \Sigma)^\star$, A je neterminál a $\gamma \neq \varepsilon$. Jedinou výjimku tvoří pravidlo $S \to \varepsilon$, pak se ale S nevyskytuje na pravé straně žádného pravidla.

  Jazyky generované gramatikami typu 1 se nazývají *jazyky typu 1*, též *kontextové jazyky*.
- *Gramatiky typu 2*, též *bezkontextové gramatiky*, (což zkracujeme na CF gramatiky) jsou takové gramatiky, kde každé pravidlo v P je tvaru

  $$A \to \gamma,$$

  kde $\gamma \in (N \cup \Sigma)^\star$ a A je neterminál.

  Jazyky generované gramatikami typu 2 se nazývají *bezkontextové jazyky* nebo *jazyky typu 2*.

- *Gramatiky typu 3* neboli *regulární gramatiky* (též *pravé lineární gramatiky*) jsou takové gramatiky, kde každé pravidlo v P je tvaru

$$A \to w\,B,\ A \to w,$$

kde A, B jsou neterminály a w je terminální slovo.

Jazyky generované gramatikami typu 3 se nazývají *regulární jazyky* nebo *jazyky typu 3.*

□

Poznamenejme, že regulární jazyky již byly definovány jako ty jazyky, které jsou přijímány konečnými automaty — později ukážeme, že je to správně, totiž, že každý jazyk typu 3 je přijímán konečným automatem, a naopak, každý regulární jazyk je generován gramatikou typu 3.

3.1.7 Nevypouštěcí gramatiky. Je zřejmé, že každý jazyka typu 1 je také jazyk typu 0. Obdobně, každý jazyk typu 3 je jazyk typu 2. Pro bezkontextové jazyky (jazyky typu 2) již není zřejmé, že jsou také kontextové (typu 1). Je to proto, že kontextové gramatiky mají omezení na pravidla, kde pravá strana je prázdné slovo.

Navíc, v dalším textu ukážeme, že i v jiných případech (Chomského normální tvar, pumping lemma pro bezkontextové jazyky, algoritmus CYK pro rozhodnutí, zda dané slovo je generováno gramatikou) je potřeba zakázat pravidla s pravou stranou ε. K tomu se hodí pojem nevypouštěcí gramatiky – jedná se o gramatiku, která má pouze pravidle s pravou stranou délky aspoň 1.

Definice. Gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ nazveme *nevypouštěcí*, jestliže neobsahuje žádné pravidlo typu $\alpha \to \varepsilon$. □

3.1.8 Tvrzení. Ke každé bezkontextové gramatice $\mathcal{G}$ existuje nevypouštěcí gramatika $\mathcal{G}_1$ taková, že

$$L(\mathcal{G}_1) = L(\mathcal{G}) \setminus \{\varepsilon\}.$$

Nástin důkazu: Pro danou bezkontextovou gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ nejprve najdeme množinu všech neterminálů U, ze kterých je možné vygenerovat prázdné slovo. To uděláme tak, že postupně vytváříme množiny $U_0, U_1, \ldots$ kde:

Do množiny U_0 dáme všechny neterminály, které se přímo přepíší na prázdné slovo, tj.

$$U_0 = \{X \mid X \to \varepsilon \in P\},$$

Máme-li zkonstruovánu množinu U_i, pak k množině U_i přidáme všechny neterminály, pro které existuje pravidlo, jehož pravá strana je tvořena jen neterminály z množiny U_i; tj.

$$U_{i+1} = U_i \cup \{X \mid X \to \alpha \in P, \text{ pro } \alpha \in U_i^\star\},$$

Protože

$$U_0 \subseteq U_1 \subseteq \ldots \subseteq U_i \subseteq \ldots N, \quad \text{a } N \text{ je konečná},$$

existuje k takové, že $U_k = U_{k+1}$. Pak $U = U_k$ je hledaná množina neterminálů.

Nyní definujeme pravidla P_1 gramatiky $G_1 = (N, \Sigma, S, P_1)$ takto: Je-li $X \to \alpha$ pravidlo z P a $\alpha \neq \varepsilon$, najdeme v α všechny výskyty neterminálů z množiny U; tj přepíšeme pravidlo na tvar

$$X \to \beta_1 X_1 \beta_2 X_2 \ldots X_{k-1} \beta_{k-1} X_k \beta_k,$$

kde $X_i \in U$ a slova β_i již neobsahují neterminály z množiny U.

Do P_1 dáme všechna pravidla $X \to \gamma$, kde γ vzniklo z α vynecháním některých (třeba i žádného nebo všech) neterminálů z množiny U.

Pravidla $X \to \varepsilon$ z P do pravidel P_1 nedáme.

O gramatice $G_1 = (N, \Sigma, S, P_1)$ se dokáže $L(\mathcal{G}) \setminus \{\varepsilon\}$, tj. že generuje všechna neprázdná slova jazyka $L(\mathcal{G})$.

3.1.9 Příklad. Je dána gramatika $\mathcal{G} = (\{A, S\}, \{a, b, c\}, S, P)$ pravidly

$$S \to aSc \,|\, A \quad A \to bAc \,|\, \varepsilon.$$

Ke gramatice $\mathcal{G}$ najdeme nevypouštěcí gramatiku $\mathcal{G}_1$, která generuje jazyk $L(\mathcal{G}) \setminus \{\varepsilon\}$ a kontextovou gramatiku $\mathcal{G}_2$, pro kterou $L(\mathcal{G}_2) = L(\mathcal{G})$.

Řešení: Použijeme postup z důkazu věty 3.1.8.

Množina U_0 je rovna $\{A\}$, dále množina $U_1 = U_0 \cup \{S\}$, protože $S \to A$ je pravidlo gramatiky $\mathcal{G}$ a $A \in U_0$. Proto $U = \{A, S\}$ (což je množina všech neterminálů). (Uvědomte si, že v gramatice $\mathcal{G}$ je možné z S vygenerovat ε a to derivací $S \Rightarrow A \Rightarrow \varepsilon$.)

Hledaná gramatika $\mathcal{G}_1$ má pravidla:

$$S \to aSc \,|\, ac \,|\, A \quad A \to bAc \,|\, bc.$$

Všimněte si, že $L(\mathcal{G}_1) = \{a^i b^j c^{i+j} \mid i + j > 0\}$, kdežto $L(\mathcal{G}) = \{a^i b^j c^{i+j} \mid i + j \geq 0\}$.

Kontextová gramatika $\mathcal{G}_2$ je gramatika $(N \cup \{S_1\}, \Sigma, S_1, P_2)$, kde pravidla jsou

$$S_1 \to S \,|\, \varepsilon,\ S \to aSc \,|\, ac \,|\, A,\ A \to bAc \,|\, bc.$$

□

3.1.10 Důsledek. Označme $\mathcal{L}_i$ třídu jazyků typu i. Pak platí:

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0.$$

□

Důkaz. Abychom mohli tvrdit, ze každý bezkontextový jazyk je také kontextový, musíme být schopni ke každé bezkontextové gramatice $\mathcal{G}$ sestrojit gramatiku $\mathcal{G}_2$, která je buď nevypouštěcí (když $\mathcal{G}$ negeneruje prázdné slovo) nebo jediné pravidlo s pravou stranou ε je pravidlo, které má na levé straně startovací symbol a tento startovací symbol se již neobjevuje na pravé straně žádného pravidla.

Jestliže tedy $\mathcal{G}$ generuje prázdné slovo, kontextovou gramatiku zkonstruujeme takto: K nevypouštěcí gramatice $\mathcal{G}_1$ přidáme nový startovací symbol S_1 a k pravidlům P_1 přidáme ještě dvě pravidla

$$S_1 \to \varepsilon, \quad S_1 \to S.$$

Gramatika $\mathcal{G}_2$ generuje jazyk $L(\mathcal{G})$ a splňuje podmínky kontextové gramatiky. (Uvědomte si, že se jedná o postup, který jsme použili v předchozím příkladě.)

3.1.11 V tomto odstavci ukážeme, že jméno regulární jazyky není zavádějící; tj. že jazyk je přijímán konečným automatem právě tehdy, když je generován gramatikou typu 3.

Tvrzení 1. Každý regulární jazyk je generován některou gramatikou typu 3, tj. některou regulární gramatikou.

Přesněji: je-li L regulární jazyk, pak existuje regulární gramatika $\mathcal{G}$ taková, že $L = L(\mathcal{G})$. □

Nástin důkazu: Je-li L regulární jazyk, pak existuje DFA M, který L přijímá. Definujme gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ takto:

- Množina neterminálů N je množina stavů automatu M,
- startovací symbol S je počáteční stav automatu M,
- P obsahuje pravidla dvou typů:
  1. $q \to ap$ pro $\delta(q, a) = p$,
  2. $q \to \varepsilon$ pro všechny koncové stavy q.

Není těžké ukázat, že takto definovaná gramatika generuje jazyk $L(M)$.

Tvrzení 2. Gramatiky typu 3, tj. regulární gramatiky, přijímají pouze regulární jazyky.

Přesněji, je-li $\mathcal{G}$ gramatika typu 3, pak jazyk $L(\mathcal{G})$ je regulární. □

Nástin důkazu: Ke gramatice s pravidly $X \to wY$ a $X \to w$ pro $w \in \Sigma^\star$ nejprve najdeme gramatiku generující stejný jazyk, která má pravidla pouze typu $X \to aY$ a $X \to \varepsilon$, kde $a \in \Sigma \cup \{\varepsilon\}$. Toho dosáhneme snadno přidáním nových neterminálů (např. pravidlo $X \to abY$ nahradíme pravidly $X \to aZ$ a $Z \to bY$; neterminál Z musí být nový).

K takovéto gramatice sestrojíme automat (obecně NFA s ε-přechody) takto:

- Množina stavů je množina neterminálů.
- Počáteční stav je startovací symbol.
- $Y \in \delta(X, a)$ právě tehdy, když $X \to aY$ je pravidlo.
- Koncové stavy jsou ty neterminály, pro které je $X \to \varepsilon$ pravidlo.

Není těžké ukázat, že takto definovaný automat přijímá jazyk $L(\mathcal{G})$.

3.2 Bezkontextové gramatiky

Připomeňme, že bezkontextová gramatika (CF gramatika) je gramatika $\mathcal{G} = (N, \Sigma, S, P)$, která obsahuje pouze pravidla typu

$$A \to \gamma, \quad \text{kde } \gamma \in (N \cup \Sigma)^\star \text{ a } A \text{ je neterminál.}$$

Dále připomeňme, že ke každé CF gramatice $\mathcal{G}$ existuje nevypouštěcí CF gramatika $\mathcal{G}_1$, která generuje všechna neprázdná slova z $L(\mathcal{G})$, tj. platí

$$L(\mathcal{G}_1) = L(\mathcal{G}) \setminus \{\varepsilon\}.$$

3.2.1 Následující tvrzení je sice jednoduché, ale velmi podstatné pro studium bezkontextových gramatik. Velmi neformálně řečeno, říká, že jednotlivé části odvození jsou nezávislé a „do sebe nezasahují“. Máme-li odvodit nějaké slovo γ ze slova $\alpha A \beta$, kde A je neterminál, pak se γ skládá ze tří částí: to, co je odvozeno z α, následované slovem odvozeným z A, končícím slovem odvozeným z β. Přesněji:

Tvrzení. Máme dánu bezkontextovou gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ a v ní derivaci

$$S \Rightarrow_{\mathcal{G}}^\star \alpha A \beta \Rightarrow_{\mathcal{G}}^\star \gamma,$$

pro $\alpha, \beta, \gamma \in (\Sigma \cup N)^\star$ a $A \in N$.

Pak existují slova $\varphi, \psi, \mu \in (\Sigma \cup N)^\star$ taková, že

$$\gamma = \varphi\psi\mu, \quad \text{kde} \quad \alpha \Rightarrow_{\mathcal{G}}^\star \varphi, \quad A \Rightarrow_{\mathcal{G}}^\star \psi, \quad \beta \Rightarrow_{\mathcal{G}}^\star \mu.$$

3.2.2 Levá derivace, levé odvození. Předchozí tvrzení říká, že pro bezkontextovou gramatiku můžeme pravidla přehazovat, jediné, co musíme ohlídat, je, aby pravidlo bylo možné použít. Můžeme proto pravidla uspořádat (přehodit) tak, abychom přepisovali vždy ten nejvíc levý neterminál. A takové derivaci říkáme levá derivace.

Definice. Přímé odvození se nazývá *levé*, jestliže se přepisuje ten neterminál, který je nejvíc „vlevo“, tj. $u\,A\,\beta \Rightarrow_{\mathcal{G}} u\,\delta\,\beta$, kde $u \in \Sigma^\star$ a $A \to \delta$ je pravidlo gramatiky.

Derivace (odvození) se nazývá *levá derivace* (*levé odvození*), jestliže se skládá pouze z levých přímých odvození. □

Obdobně definujeme pravé přímé odvození a pravou derivaci.

Tvrzení. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Pak pro každou derivaci $S \Rightarrow_{\mathcal{G}}^\star w$ existuje levá derivace terminálního w z S taková, že používá stejná pravidla jako původní derivace (pouze možná v jiném pořadí). □

3.2.3 Derivační strom (parse tree). Derivace v bezkontextové gramatice si můžeme představit také jako stromy, budeme jim říkat derivační stromy. Výhoda derivačních stromů je v tom, že v nich „není zachyceno“ pořadí použití jednotlivých pravidel, jenom jejich struktura.

Definice. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$. *Derivační strom* (též *parse tree*) je kořenový strom, takový, že:

1. Každý vrchol, který není list, je ohodnocen neterminálem.
2. Každý list je ohodnocen terminálem, neterminálem nebo prázdným slovem ε. V případě, že je list ohodnocen prázdným slovem ε, je to jediný následník (svého předchůdce).
3. Jestliže některý vrchol, který není list, je ohodnocen neterminálem A a má následníky (v pořadí zleva doprava) $X_1, X_2, \ldots, X_k$, $X_i \in N \cup \Sigma$, pak $A \to X_1 X_2 \ldots X_k$ je pravidlo gramatiky $\mathcal{G}$.

Řekneme, že derivační strom *dává*, nebo *má za výsledek* slovo γ, jestliže γ je ohodnocení listů derivačního stromu (čteno zleva doprava). □

3.2.4 Je zřejmé, že pro každou derivaci existuje derivační strom, který derivaci odpovídá. Není již pravda, že pro dvě různé derivace jsou derivační stromy také různé; ano, pro dvě derivace, které se liší pouze pořadím použití jednotlivých pravidel, je derivační strom stejný. Ovšem jednoznačnost přece jenom existuje, ale je mezi derivačními stromy a levými derivacemi (pravými derivacemi).

Tvrzení.

1. Pro každou derivaci $S \Rightarrow_{\mathcal{G}}^{\star} w$ existuje derivační strom s výsledkem w.
2. Ke každému derivačnímu stromu s výsledkem w existuje aspoň jedna derivace $S \Rightarrow_{\mathcal{G}}^{\star} w$ (takových derivací může být více).
3. Ke každému derivačnímu stromu s výsledkem w existuje právě jedna levá (právě jedna pravá) derivace slova w z S.

3.2.5 Jednoznačné a víceznačné bezkontextové gramatiky. Problému, jak ke slovu generovanému danou CF gramatikou najít jeho derivační strom, se budeme podrobněji věnovat v dalším textu. Pro rekonstrukci daného derivačního stromu je výhodné, když takový derivační strom existuje pouze jediný. A to je případ tzv. jednoznačné gramatiky.

Definice. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Řekneme, že $\mathcal{G}$ je *jednoznačná*, jestliže pro každé slovo w generované gramatikou $\mathcal{G}$ existuje jediný derivační strom s výsledkem w (tj. existuje jediná levá derivace w z S).

V opačném případě mluvíme o *víceznačné* gramatice. □

3.2.6 Víceznačný jazyk. Je zřejmé, že pro jeden bezkontextový jazyk mohou existovat dvě CF gramatiky a to tak, že jedna je víceznačná a druhá je jednoznačná. To jen říká, že v prvním případě jsme si nevybrali tu nejlepší CF gramatiku. Existují ale bezkontextové jazyky, pro které **neexistuje** jednoznačná gramatika.

Definice. Bezkontextový jazyk L se nazývá *víceznačný* (též *podstatně víceznačný*), jestliže **každá** bezkontextová gramatika, která ho generuje, je víceznačná. □

Například jazyk $L = \{a^i b^j c^k d^l \mid i = j \text{ nebo } k = l\}$ je víceznačný.

3.2.7 Redukovaná bezkontextová gramatika. Podobně jako jsme zavedli redukovaný automat, zavedeme též redukovanou CF gramatiku. A to jako gramatiku, která „nemá zbytečné neterminály“. Zbytečné neterminály jsou dvou typů; jsou to jednak neterminály, ze kterých nelze vygenerovat terminální slovo, jednak neterminály, na které „se nedostaneme“ ze startovacího symbolu. Nebude však platit, že každý bezkontextový jazyk má jedinou redukovanou CF gramatiku, která ho generuje. Jinými slovy, existují dvě různé gramatiky (i např. s jiným počtem neterminálů), které generují ten samý jazyk.

Definice. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$, která generuje aspoň jedno slovo. Řekneme, že $\mathcal{G}$ je *redukovaná*, jestliže splňuje tyto dvě podmínky:

1. Ke každému neterminálu A existuje aspoň jedno terminální slovo w takové, že $A \Rightarrow_{\mathcal{G}}^{\star} w$.
2. Ke každému neterminálu A existují slova $\alpha, \beta \in (N \cup \Sigma)^{\star}$ tak, že

$$S \Rightarrow_{\mathcal{G}}^{\star} \alpha A \beta.$$

□

V příštím odstavci uvedeme postup, který pro každý neprázdný bezkontextový jazyk zajišťuje aspoň jednu redukovanou gramatiku, která ho generuje.

3.2.8 Tvrzení. Ke každé bezkontextové gramatice $\mathcal{G} = (N, \Sigma, S, P)$, pro kterou $L(\mathcal{G}) \neq \emptyset$, existuje redukovaná gramatika $\mathcal{G}_1$ taková, že $L(\mathcal{G}_1) = L(\mathcal{G})$. □

Tvrzení dokážeme algoritmem, který pro danou CF gramatiku rozhodne, zda generuje aspoň jedno slovo; a v případě, že ano, najde redukovanou gramatiku generující stejný jazyk.

Algoritmus redukce CFG. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$.

1. Sestrojíme množinu $V = \{A \mid A \in N, A \Rightarrow_{\mathcal{G}}^{\star} w, w \in \Sigma^{\star}\}$; postupujeme indukcí:

   Nejprve položíme

$$V_1 = \{A \mid \text{existuje } w \in \Sigma^{\star} \text{ takové, že } A \to w \in P\}.$$

   Pak konstruujeme

$$V_{i+1} = V_i \cup \{A \mid \text{existuje } \alpha \in (V_i \cup \Sigma)^{\star} \text{ tak, že } A \to \alpha \in P\}.$$

   Platí

$$V_1 \subseteq V_2 \subseteq \ \ldots \ \subseteq N.$$

   Protože N je konečná, existuje n takové, že $V_n = V_{n+1}$. Položíme $V = V_n$.

   Jestliže $S \notin V$, pak $L(\mathcal{G}) = \emptyset$ a redukovaná gramatika ke gramatice $\mathcal{G}$ neexistuje.

   V opačném případě definujeme $\mathcal{G}' = (V, \Sigma, S, P')$: do P' dáme ta pravidla z P, která obsahují pouze neterminály z množiny V (a to jek na levé, tak na pravé straně).

2. Pro gramatiku $\mathcal{G}' = (V, \Sigma, S, P')$ sestrojíme (opět indukcí) množinu

$$U = \{A \mid A \in V, \text{ existují } \alpha, \beta \in (V \cup \Sigma)^{\star} \text{ tak, že } S \Rightarrow_{\mathcal{G}}^{\star} \alpha A \beta\}.$$

   Nejprve položíme

$$U_0 = \{S\},$$

   pak konstruujeme

$$U_{i+1} = U_i \cup \{A \mid \text{existují } B \in U_i, \alpha, \beta \in (V \cup \Sigma)^{\star} \text{ tak, že } B \to \alpha A \beta \in P'\}.$$

   Tj. do množiny U_{i+1} dáme všechny neterminály, které se objevují na pravé straně některého pravidla z P, kde na levé straně pravidla jsou neterminály pouze z U_i.

   Platí

$$U_0 \subseteq U_1 \subseteq U_2 \subseteq \ \ldots \ \subseteq V.$$

   Proto existuje m takové, že $U_m = U_{m+1}$. Položíme $U = U_m$.

   Hledaná gramatika je gramatika $\mathcal{G}_1 = (U, \Sigma, S, P_1)$, kde P_1 je množina všech pravidel z P' (a tedy i z P), které obsahují neterminály pouze z množiny U.

Platí: gramatika $\mathcal{G}_1 = (U, \Sigma, S, P_1)$ je redukovaná a generuje stejný jazyk jako původní gramatika $\mathcal{G} = (N, \Sigma, S, P)$.

3.2.9 Poznámky.

- Uvědomte si, že redukovaná CF gramatika „nemá zbytečné neterminály“.
- Je obtížné ke dvěma bezkontextovým gramatikám zjistit, zda generují stejný jazyk. Redukce gramatik nám k rozhodnutí nepomůže.
- Kroky předchozího postupu nelze zaměnit. Kdybychom nejprve hledali množinu neterminálů U a pak teprve z ní vybírali ty neterminály, ze kterých je možné odvodit terminální slovo, výsledná gramatika by nemusela splňovat druhou podmínku z 3.2.7 a tudíž být redukovaná.

3.2.10 Chomského normální tvar. Jedním z důležitých typů CF gramatik, je gramatika v Chomského normálním tvaru. Je to CF gramatika, kde pravá strana pravidla je buď terminální písmeno nebo dvojice neterminálů. Gramatiky v Chomského normálním tvaru budeme využívat jednak pro důkaz Pumping lemmatu pro bezkontextové jazyky, jednak při polynomiálním algoritmu, který pro danou gramatiku a dané terminální slovo rozhodne, zda je slovo generováno.

Definice. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Řekneme, že gramatika $\mathcal{G}$ je v *Chomského normálním tvaru*, jestliže má pouze pravidla tvaru

$$A \to BC,\ A \to a \quad \text{kde} \quad A, B, C \in N,\ a \in \Sigma.$$

□

Poznamenejme, že derivační strom v CF gramatice v Chomského normálním tvaru je vždy binární strom a to takový, že předchůdce listu má vždy jen jednoho následníka.

3.2.11 Věta. Pro každou bezkontextovou gramatiku $\mathcal{G}$ existuje bezkontextová gramatika $\mathcal{G}'$ v Chomského normálním tvaru taková, že

$$L(\mathcal{G}') = L(\mathcal{G}) \setminus \{\varepsilon\}.$$

□

Větu dokážeme postupem, který pro danou CF gramatiky $\mathcal{G}$ sestrojí CF gramatiku $\mathcal{G}'$ v Chomského normálním tvaru generující stejný jazyk.

Postup nalezení ekvivalentní gramatiky v Chomského normálním tvaru.

1. Pro gramatiku $\mathcal{G}$ nejprve sestrojíme nevypouštěcí gramatiku $\mathcal{G}_1$, která generuje všechna neprázdná slova z jazyka $L(\mathcal{G})$ (viz 3.1.8).
2. Následně v gramatice $\mathcal{G}_1$ odstraníme pravidla typu $A \to B$, kde $A, B \in N$, takto: Pro každou dvojici neterminálů A, C takovou, že

   $$A \Rightarrow X_1 \Rightarrow X_2 \Rightarrow \ldots \Rightarrow C,$$

   a pro každé pravidlo $C \to \alpha$ do gramatiky přidáme pravidlo $A \to \alpha$.
3. Dále:
   a) pro každý terminál $a \in \Sigma$ zavedeme nový neterminál X_a a přidáme pravidlo $X_a \to a$,
   b) pro každé pravidlo $A \to \alpha$, kde $|\alpha| \geq 2$, nahradíme v pravé straně pravidla každý terminál b neterminálem X_b.
4. Nyní stačí nahradit pravidla typu

   $$A \to B_1 B_2 \ldots B_k,\ k \geq 3$$

   pravidly s novými neterminály $C_1, C_2, \ldots, C_{k-2}$:

   $$A \to B_1 C_1,\ C_1 \to B_2 C_2,\ \ldots, C_{k-2} \to B_{k-1} B_k.$$

Tím dostaneme hledanou gramatiku $\mathcal{G}'$ v Chomského normálním tvaru.

3.2.12 Následující tvrzení využijeme při zdůvodnění pumping lemmatu pro bezkontextové jazyky.

Tvrzení. Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$ v Chomského normálním tvaru a dále derivační strom s výsledkem $w \in \Sigma^\star$.

Jestliže nejdelší orientovaná cesta v tomto derivačním stromu má délku n, pak pro délku slova w platí

$$|w| \leq 2^{n-1}.$$

□

Nástin důkazu: Pro $n = 1$ se jedná o použití jediného pravidla a proto je w rovno jedinému písmenu $w = b$, $b \in \Sigma$.

Předpokládejme, že $n > 1$. „Nejhorší“ případ nastane, když v derivačním stromě mají všechny vrcholy kromě předposlední hladiny vždy dva následníky. Pak ale předposlední hladina má 2^{n-1} vrcholů a tudíž slovo w je délky 2^{n-1}.

3.2.13 Pumping lemma pro bezkontextové gramatiky. Pro každý bezkontextový jazyk L existuje kladné přirozené číslo m takové, že každé slovo $z \in L$ délky alespoň m lze rozdělit na pět částí $z = uvwxy$ tak, že

1. $|vwx| \leq m$, (tj. prostřední část není příliš dlouhá),
2. $vx \neq \varepsilon$ (tj. aspoň jedno ze slov v, x není prázdné),
3. pro všechna $i \geq 0$ platí $uv^iwx^iy \in L$, (tj. v a x se dají současně do slova z „napumpovat“ či odebrat a stále dostaneme slovo z jazyka L).

□

Myšlenka důkazu: Protože jazyk L je bezkontextový, existuje bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$ v Chomského normálním tvaru, která generuje všechna neprázdná slova jazyka L. Označme k počet neterminálů gramatiky $\mathcal{G}$ (tj. $k = |N|$). Položíme $m = 2^k$.

Vezmeme libovolné slovo $z \in L$ délky $|z| \geq m$. Z tvrzení 3.2.12 víme, že derivační strom T s výsledkem z obsahuje orientovanou cestu délky aspoň $k + 1$. Označme vrcholy na této cestě $S, A_1, \ldots, A_k, b$. Protože gramatika $\mathcal{G}$ obsahuje pouze k různých neterminálů, musí se v této cestě aspoň jeden neterminál vyskytovat dvakrát. Vezmeme první dva výskyty téhož neterminálu „od konce“; tj. najdeme i, j takové různé indexy, že $i < j$, $A_i = A_j$ a všechny neterminály $A_{i+1}, \ldots, A_k$ jsou různé. Označme T_i, resp. T_j podstromy derivačního stromu s kořenem A_i, resp. A_j. Dále označme w výsledek stromu T_j. Výsledek stromu T_i obsahuje w jako podslovo, tj. je tvaru vwx. Protože $i < j$, aspoň jedno ze slov v, x není prázdné, tudíž w je vlastní podslovo slova vwx. Protože strom T_i má jen dva neterminály stejné, je délka slova vwx nejvýše m.

Z vlastností derivačních stromů nyní vyplývá, že $z = uvwxy$ (slova u a y mohou obě být prázdná).

Derivační strom pro slovo $uv^0wx^0y = uwy$ dostaneme tak, že ve stromu T nahradíme podstrom T_i podstromem T_j.

Derivační strom pro slovo uv^2wx^2y dostaneme tak, že ve stromu T nahradíme podstrom T_j podstromem T_i, atd.

3.2.14 Využití Pumping lemmatu pro bezkontextové gramatiky. Ukážeme, že jazyk $L = \{0^n1^n2^n \mid n \geq 0\}$ není bezkontextový.

Zdůvodnění: Předpokládejme, že jazyk L je bezkontextový. Pak existuje kladné číslo m z Pumping lemmatu. V jazyce L leží slovo $z = 0^m1^m2^m$, které má délku $3m > m$. Podle Pumping lemmatu existují slova u, v, w, x, y taková, že

$$0^m1^m2^m = uvwxy, \;\; |vx| > 0, \;\; |vwx| \leq m \;\text{ a }\; uv^iwx^iy \in L \text{ pro } i \geq 0.$$

Ukážeme, že slovo $uv^0wx^0y = uwy \notin L$. To bude hledaný spor.

Podmínka $|vwx| \leq m$ znamená, že slovo vwx buď neobsahuje písmeno 2 nebo neobsahuje písmeno 0.

Jestliže vwx neobsahuje písmeno 2, pak slova v, x obsahují pouze 0 nebo 1, tedy nejvýše dvě ze tří písmen $0, 1, 2$.

Jestliže vwx neobsahuje písmeno 0, pak slova v, x obsahují pouze 1 nebo 2, tedy opět nejvýše dvě ze tří písmen $0, 1, 2$.

To ale znamená, že v obou případech nemůže slovo uwy (tj. slovo z, ze kterého jsme vypustili slova v a x) obsahovat stejný počet všech tří písmen $0, 1, 2$.

3.3 Algoritmus CYK

CYK je algoritmus, který pro danou bezkontextovou gramatiku $\mathcal{G}$ v Chomského normálním tvaru a pro dané terminální slovo w rozhodne, zda $w \in L(\mathcal{G})$ a to v čase úměrném k^3, kde k je délka slova w.

3.3.1 CYK. Označme $\mathcal{G} = (N, \Sigma, S, P)$ a $w = a_1 a_2 \dots a_k$. Postupně vytváříme množiny $X_{i,j}$ pro $1 \le i \le j \le k$, kde

$$X_{i,j} = \{A \in N \mid A \Rightarrow_{\mathcal{G}}^{\star} a_i a_{i+1} \dots a_j\}, \quad \text{tj. z } A \text{ vygenerujeme slovo } a_i \dots a_j.$$

Platí

$$A \in X_{i,i} \quad \text{právě tehdy, když} \quad A \to a_i \in P.$$

Navíc

$$X_{1,k} = \{A \in N \mid A \Rightarrow_{\mathcal{G}}^{\star} a_1 a_2 \dots a_k = w\}.$$

Dále si uvědomte, že $A \Rightarrow_{\mathcal{G}}^{\star} a_i a_{i+1} \dots a_j$ právě tehdy, když existují neterminály B, C takové, že

$$\begin{array}{lllll} A \to BC \in P, \text{ kde} & \text{buď} & B \Rightarrow_{\mathcal{G}}^{\star} a_i & \text{a} & C \Rightarrow_{\mathcal{G}}^{\star} a_{i+1} \dots a_j \\ & \text{nebo} & B \Rightarrow_{\mathcal{G}}^{\star} a_i a_{i+1} & \text{a} & C \Rightarrow_{\mathcal{G}}^{\star} a_{i+2} \dots a_j \\ & \text{nebo} & B \Rightarrow_{\mathcal{G}}^{\star} a_i a_{i+1} a_{i+2} & \text{a} & C \Rightarrow_{\mathcal{G}}^{\star} a_{i+3} \dots a_j \\ & & \dots & & \dots \\ & \text{nebo} & B \Rightarrow_{\mathcal{G}}^{\star} a_i \dots a_{j-1} & \text{a} & C \Rightarrow_{\mathcal{G}}^{\star} a_j \end{array}$$

Předpokládejme, že máme zkonstruovány všechny množiny $X_{p,q}$, kde $q - p < n$. Pak množiny $X_{i,j}$ pro $j - i = n$ utvoříme takto:

$$\begin{array}{llll} A \in X_{i,j} \text{ iff } \exists A \to BC \in P \text{ tak, že} & \text{buď} & B \in X_{i,i} & \text{a} \quad C \in X_{i+1,j} \\ & \text{nebo} & B \in X_{i,i+1} & \text{a} \quad C \in X_{i+2,j} \\ & \text{nebo} & B \in X_{i,i+2} & \text{a} \quad C \in X_{i+3,j} \\ & & \dots & \dots \\ & \text{nebo} & B \in X_{i,j-1} & \text{a} \quad C \in X_{j,j} \end{array}$$

Začínáme tedy konstrukcí k množin $X_{i,i}$, $i = 1, 2, \dots, k$, následuje pak $k - 1$ množin $X_{i,i+1}$, $i = 1, 2, \dots, k-1$, $k - 2$ množin $X_{i,i+2}$, $i = 1, 2, \dots, k-2$, atd. dvě množiny $X_{1,k-1}$, $X_{2,k}$ a nakonec jedna množina $X_{1,k}$ a to podle následujícího postupu:

$$X_{i,j} = \{A \in N \mid \exists A \to BC \in P \text{ tak, že } B \in X_{i,i+m}, C \in X_{i+m+1,j}\}.$$

Platí $w \in L(\mathcal{G})$ právě tehdy, když $S \in X_{1,k}$.

3.3.2 Příklad. Je dána gramatika $\mathcal{G}$ pravidly

$$\begin{array}{l} S \to AB \mid BC \\ A \to BA \mid a \\ B \to CC \mid b \\ C \to AB \mid a \end{array}$$

Pomocí algoritmu CYK rozhodněte, zda slovo *aabab* je generováno bezkontextovou gramatikou $\mathcal{G}$.

Řešení: Konstrukci množin $X_{i,j}$ pro $1 \leq i \leq j \leq 5$ si znázorníme do tabulky. Tabulka bude mít 5 řádků a 5 sloupců, kde vyplněných bude jen ta část, která se nenachází „nad diagonálou“. Poslední řádek obsahuje pět množin $X_{1,1}$, $X_{2,2}$, $X_{3,3}$, $X_{4,4}$ a $X_{5,5}$. Předposlední řádek obsahuje čtyři množiny $X_{1,2}$, $X_{2,3}$, $X_{3,4}$ a $X_{4,5}$. Řádek, který je třetí od zdola (a také shora) obsahuje tři množiny $X_{1,3}$, $X_{2,4}$ a $X_{3,5}$. Řádek, který je čtvrtý od odzdola (a druhý shora) obsahuje dvě množiny $X_{1,4}$ a $X_{2,5}$. Nejvyšší řádek obsahuje jednu množinu $X_{1,5}$. V našem případě jsou neterminály množin X_{ij} v následující tabulce:

S, C				
S, A, C	B			
B	B	S, C		
B	S, C	S, A	S, C	
A, C	A, C	B	A, C	B
a	a	b	a	b

Z předchozí tabulky také můžeme odvodit derivace slova $aabab$ gramatikou $\mathcal{G}$. Jedna z takových derivací je např. tato:

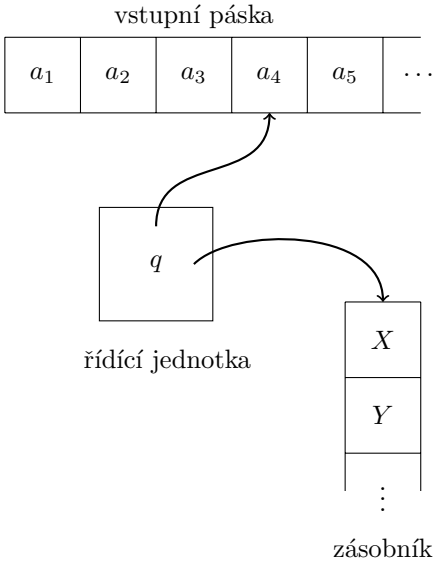
$$S \Rightarrow AB \Rightarrow aB \Rightarrow aCC \Rightarrow aaC \Rightarrow aaAB \Rightarrow aaBAB \Rightarrow aabAB \Rightarrow aabaB \Rightarrow aabab.$$

3.3.3 Poznámka. Algoritmus CYK pracuje v čase úměrném k^3, kde k je délka terminálního slova w. Velikost gramatiky v Chomského normálním tvaru považujeme za konstantu.

Jedná se o algoritmus ve stylu dynamického programování. V případě speciálních bezkontextových gramatik je možné sestrojit algoritmus pracující v lineárním čase.

3.4 Zásobníkové automaty

Regulární jazyky jsou jazyky, které (mimo jiné) jsou přijímány konečnými automaty. Ukážeme, že pro bezkontextové jazyky (třída jazyků, která obsahuje regulární jazyky, ale ještě i další) také existují automaty, které přijímají právě je. Jedná se o obecnější typ automatu; liší se od konečných automatů tím, že mají navíc dodatečnou „paměť“ ve formě zásobníku.



3.4.1 Zásobníkový automat. Zhruba řečeno, zásobníkový automat se skládá z řídící jednotky, která je v jednom z konečně mnoha možných stavů, ze vstupní pásky se čtecí hlavou a ze zásobníku. Na základě toho, v jakém stavu se automat nachází, co hlava čte na vstupní pásce a jaký symbol je na vrcholu zásobníku, automat udělá akci: Přejde do nového stavu, posune čtecí hlavu o jedno políčko doprava nebo stojí (to v případě, že automat reagoval na prázdné slovo) a zásobníkový symbol, který je na vrcholu, nahradí zásobníkovým slovem daným přechodovou funkcí. Formálně:

Definice. *Zásobníkový automat* je sedmice $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde

- Q je konečná množina stavů,
- Σ je konečná neprázdná množina vstupních symbolů,
- Γ je konečná množina zásobníkových symbolů,
- δ přiřazuje každé trojici (q, a, X), $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$, konečnou množinu dvojic (p, α), kde $p \in Q$ a $\alpha \in \Gamma^\star$. Formálně:

$$\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_f(Q \times \Gamma^\star).$$

  ($\mathcal{P}_f(A)$ značí množinu všech konečných podmnožin množiny A.)
- $q_0 \in Q$ je počáteční stav,
- $Z_0 \in \Gamma$ je počáteční zásobníkový symbol a
- $F \subseteq Q$ je množina koncových (přijímajících) stavů.

□

Uvědomte si, že zásobníkový automat tak, jak byl definován, je nedeterministický.

Situace zásobníkového automatu. Je dán zásobníkový automat $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Jestliže je zásobníkový automat ve stavu q, na pásce je ještě nepřečtené slovo u a obsah zásobníku je slovo γ, pak tuto *situaci* popisujeme trojicí (q, u, γ), kde první symbol slova γ je vrchol zásobníku. □

Jeden krok práce zásobníkového automatu – relace $\vdash_A$.

Je dán zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, který je v situaci $(q, au, X\gamma)$, kde $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$. Jestliže existuje $(p, \alpha) \in \delta(q, a, X)$, pak A přejde do situace $(p, u, \alpha\gamma)$.

Tedy

$$(q, au, X\gamma) \vdash_A (p, u, \alpha\gamma) \quad \text{právě tehdy, když} \quad (p, \alpha) \in \delta(q, a, X).$$

□

Relace $\vdash_A^\star$. Jeden krok zásobníkového automatu rozšíříme na konečný počet kroků. Automat A *přejde ze situace S do situace S'*, píšeme $S \vdash_A^\star S'$, právě tehdy, když buď $S = S'$ nebo existuje konečný počet situací $S_1, S_2, \ldots, S_n$ takových, že

$$S \vdash_A S_1,\ S_1 \vdash_A S_2\ \ldots,\ S_n \vdash_A S'.$$

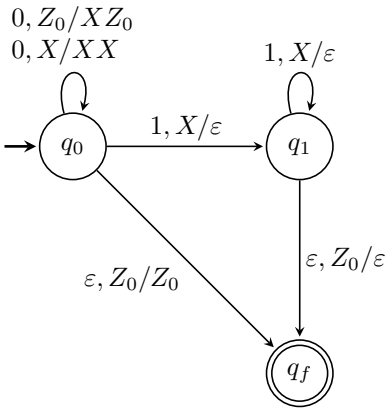
O $S \vdash_A^\star S'$ také mluvíme jako o *výpočtu* nad situací S. □

3.4.2 Stavový diagram zásobníkového automatu. Obdobně jako u konečných automatů můžeme přechodovou funkci zásobníkového automatu zadat tabulkou nebo stavovým diagramem. V tabulce máme tolik sloupců, kolik je dvojic $a \in \Sigma \cup \{\varepsilon\}$, zásobníkový symbol. U stavového diagramu máme hranu ze stavu q do stavu p ohodnocenou $a, X/\gamma$ právě tehdy, když $(p, \gamma) \in \delta(q, a, X)$, kde $a \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$. Ukážeme si oba způsoby na následujícím příkladě.

Příklad. Je dán zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde $Q = \{q_0, q_1, q_f\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{Z_0, X\}$, jehož přechodová funkce je daná tabulkou

	$(0, Z_0)$	$(0, X)$	$(1, Z_0)$	$(1, X)$	(ε, Z_0)	(ε, X)
$\to q_0$	(q_0, XZ_0)	(q_0, XX)	–	(q_1, ε)	(q_f, Z_0)	–
q_1	–	–	–	(q_1, ε)	(q_f, ε)	–
$\leftarrow q_f$	–	–	–	–	–	–

a odpovídající stavový diagram je:



3.4.3 Jazyky přijímané zásobníkovým automatem.

Pro daný zásobníkový automat rozlišujeme dva jazyky jím přijímané. Kromě jazyka přijímaného koncovým stavem (jedná se o obdobu jazyka přijímaného konečným automatem) je to ještě jazyk přijímaný prázdným zásobníkem. Zde jsou definice těchto pojmů.

Definice. Je dán zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. *Jazyk přijímaný koncovým stavem* $L(A)$, je definován takto:

$$L(A) = \{u \in \Sigma^\star \mid (q_0, u, Z_0) \vdash_A^\star (p, \varepsilon, \gamma), p \in F\}.$$

□

Zhruba řečeno, zásobníkový automat začne v počáteční situaci, tj. v počátečním stavu q_0, na vstupní pásce má slovo u a na zásobníku pouze počáteční zásobníkový symbol Z_0. Slovo je přijato právě tehdy, když existuje výpočet, který přečte u a automat je v některém koncovém stavu. To, zda je současně vyprázdněn zásobník nebo není, nehraje roli.

Definice. Je dán zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. *Jazyk přijímaný prázdným zásobníkem* $N(A)$ je definován takto:

$$N(A) = \{u \in \Sigma^\star \mid (q_0, u, Z_0) \vdash_A^\star (p, \varepsilon, \varepsilon), p \in Q\}.$$

□

Zhruba řečeno, zásobníkový automat začne v počátečním stavu q_0, na vstupní pásce má slovo u a na zásobníku pouze počáteční zásobníkový symbol Z_0 (počáteční situace). Slovo je přijato právě tehdy, když najdeme výpočet, který po přečtení slova u má prázdný zásobník.

Protože při přijímání prázdným zásobníkem nezáleží na množině koncových stavů (nepotřebujeme je),vynecháváme množinu koncových stavů a zásobníkový automat v takovém případě je pouze šestice $(Q, \Sigma, \Gamma, \delta, q_0, Z_0)$.

3.4.4 Příklad z 3.4.2 – pokračování. Ukažme si práci zásobníkového automatu nad slovem $0^3 1^3$. Zásobníkový automat začíná v situaci $(q_), 0^3 1^3, Z_0)$ a pokračuje

$$(q_0, 0^3 1^3, Z_0) \vdash (q_0, 0^2 1^3, X Z_0) \vdash^2 (q_0, 1^3, X^3 Z_0) \vdash (q_1, 1^2, X^2 Z_0) \vdash^2 (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, \varepsilon).$$

To znamená, že slovo $0^3 1^3$ je přijato jak koncovým stavem, tak prázdným zásobníkem.

Není těžké ukázat, že jazyk přijímaný koncovým stavem je

$$L(A) = \{0^n 1^n \mid n \geq 0\},$$

kdežto jazyk přijímaný prázdným zásobníkem je

$$N(A) = \{0^n 1^n \mid n \geq 1\}.$$

Ano, prázdné slovo ε není přijato prázdným zásobmíkem.

Je jasné, že zásobníkové automaty představují rozšíření konečných automatů. Ano, každý DFA, či NFA, či ε-NFA můžeme chápat jako zvláštní případ zásobníkového automatu. Např. pro NFA má zásobníkový automat jediný zásobníkový symbol a to počáteční a při práci se zásobník nemění.

Pro každý zásobníkový automat jsme zavedli dva jazyky – jazyk $L(A)$ obsahující slova přijímané koncovým stavem, a jazyk $N(A)$ obsahující slova přijímaná prázdným zásobníkem. V následujících dvou tvrzeních si ukážeme, že třída všech jazyků přijímaných koncovým stavem nějakého zásobníkového automatu je stejná jako třída všech jazyků přijímaných prázdným zásobníkem nějakého (většinou jiného) zásobníkového automatu. Jak uvidíme později, toto platí pouze pro **nedeterministické** zásobníkové automaty.

3.4.5 Tvrzení. Je-li L jazyk přijímaný zásobníkovým automatem A prázdným zásobníkem, pak existuje zásobníkový automat B, který přijímá jazyk L koncovým stavem; tj.

$$N(A) = L(B).$$

□

Nástin důkazu: Máme zásobníkový automat $A = (Q_A, \Sigma, \Gamma_A, \delta_A, q_A, Z_A)$. K němu sestrojíme zásobníkový automat $B = (Q_B, \Sigma, \Gamma_B, \delta_B, q_B, Z_B, F_B)$ takto:

- K množině stavů Q_A přidáme dva nové stavy — nový počáteční stav q_B a koncový stav q_f; tj. $Q_B = Q_A \cup \{q_B, q_f\}$, $q_B, q_f \notin Q_A$.
- K množině zásobníkových symbolů Γ_A přidáme nový počáteční zásobníkový symbol; tj. $\Gamma_B = \Gamma_A \cup \{Z_B\}$, $Z_B \notin \Gamma_A$.
- δ_B dostaneme tak, že k δ_A přidáme následující přechody:
  - $\delta_B(q_B, \varepsilon, Z_B) = \{(q_A, Z_A Z_B)\}$, (to znamená, že na začátku práce aniž bychom četli vstupní symbol přejdeme do původního počátečního stavu a zásobník změníme tak, že na vrcholu je „starý“ počáteční zásobníkový symbol a pod ním „nový“ počáteční zásobníkový symbol).
  - Pro každé $p \in Q_A$ položíme $\delta_B(p, \varepsilon, Z_B) = \{(q_f, Z_B)\}$, (to znamená, že vyprázdní-li „starý“ zásobníkový automat svůj zásobník, přejde „nový“ zásobníkový automat do koncového stavu).

Dá se dokázat, že zásobníkový automat A přijme slovo u prázdným zásobníkem právě tehdy, když ho přijme zásobníkový automat B koncovým stavem.

3.4.6 Tvrzení. Je-li L jazyk přijímaný zásobníkovým automatem A koncovým stavem, pak existuje zásobníkový automat B, který přijímá jazyk L prázdným zásobníkem; tj.

$$L(A) = N(B).$$

□

Nástin důkazu: Označme zásobníkový automat $A = (Q_A, \Sigma, \Gamma_A, \delta_A, q_A, Z_A, F_A)$. K němu sestrojíme zásobníkový automat $B = (Q_B, \Sigma, \Gamma_B, \delta_B, q_B, Z_B)$ takto:

- K množině stavů Q_A přidáme dva nové stavy — nový počáteční stav q_B a stav q_M (který bude sloužit k vyprázdnění zásobníku automatu B); tj. $Q_B = Q_A \cup \{q_B, q_M\}$, $q_B, q_M \notin Q_A$.
- K zásobníkovým symbolům přidáme nový počáteční zásobníkový symbol; tj. $\Gamma_B = \Gamma_A \cup \{Z_B\}$, $Z_B \notin \Gamma_A$;
- δ_B dostaneme tak, že k δ_A přidáme následující přechody:
  - $\delta_B(q_B, \varepsilon, Z_B) = \{(q_A, Z_A Z_B)\}$, (význam je stejný jako v konstrukci důkazu z 3.4.5).
  - pro každé $q \in F_A$, $Y \in \Gamma_B$ přidáme $\delta_B(q, \varepsilon, Y) = \{(q_M, \varepsilon)\}$ a $\delta_B(q_M, \varepsilon, Y) = \{(q_M, \varepsilon)\}$, (to znamená, že přečteme-li celé vstupní slovo a jsme v koncovém stavu zásobníkového automatu A, zásobníkový automat B vymaže obsah svého zásobníku).

Dá se dokázat, že zásobníkový automat A přijme slovo u koncovým stavem právě tehdy, když ho přijme zásobníkový automat B prázdným zásobníkem.

3.4.7 Vztah mezi jazyky generovanými CF gramatikou a jazyky přijímanými zásobníkovými automaty popisují následující věty. První věta říká, že bezkontextové jazyky jsou přijímány zásobníkovými automaty.

Věta. Ke každé bezkontextové gramatice $\mathcal{G} = (N, \Sigma, S, P)$ existuje zásobníkový automat A takový, že

$$L(\mathcal{G}) = N(A).$$

□

Nástin důkazu: Je dána bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$. Zkonstruujeme zásobníkový automat s jedním stavem q takto:

- $Q_A = \{q\}$, $q_0 = q$,
- $\Gamma_A = N \cup \Sigma$,
- $Z_0 = S$,
- $\delta_A(q, \varepsilon, X) = \{(q, \alpha) \mid X \to \alpha \in P, X \in N\}$,
- $\delta_A(q, a, a) = \{(q, \varepsilon)\}$, pro $a \in \Sigma$.

Zhruba řečeno, je-li na vrcholu zásobníku automatu A neterminál X, nahradíme ho v zásobníku pravou stranou některého pravidla gramatiky $\mathcal{G}$ s levou stranou X (a přitom nečteme vstupní symbol, tudíž neposunujeme hlavu na vstupní pásce). Je-li na vrcholu zásobníku terminál $a \in \Sigma$, tak v případě, že a je též na vstupu, odstraníme ho z vrcholu zásobníku a hlavu na vstupní pásce posuneme o jedno políčko doprava. Jestliže se terminální písmeno na vrcholu zásobníku nerovná prvnímu čtenému symbolu, automat se neúspěšně zastaví.

Platí, že zásobníkový automat A přijme slovo $u \in \Sigma^\star$ prázdným zásobníkem právě tehdy, když je slovo u vygenerováno gramatikou $\mathcal{G}$.

3.4.8 Poznámka. Zásobníkový automat konstruovaný v důkazu tvrzení 3.4.7 je vlastně formalizací tzv. analýzy shora. Zhruba řečeno, máme slovo a chceme vědět, zda je generováno CF gramatikou $\mathcal{G}$ nebo ne. Automat vlastně „zkouší“, která pravidla odvození může obsahovat a kontroluje, zda se takto dostane k vygenerování slova nebo nedostane. Více o analýze shora uvedeme později.

3.4.9 Příklad. Je dána bezkontextová gramatika $\mathcal{G}$ pravidly:

$$S \to aSa \mid A$$
$$A \to bA \mid \varepsilon$$

Řešení. Zkonstruujme zásobníkový automat A, který přijímá jazyk $L(\mathcal{G})$. A má jeden stav, tj. $Q = \{q\}$, který je počátečním stavem. Dále množina zásobníkových symbolů je $\Gamma = \{a, b, S, A\}$, počáteční zásobníkový stav je S, a přechodová funkce δ je definovaná

$$\delta(q, \varepsilon, S) = \{(q, aSa), (q, A)\}, \quad \delta(q, A) = \{(q, bA), (q, \varepsilon)\}, \quad \delta(q, a, a) = \{(q, \varepsilon)\}, \quad \delta(q, b, b) = \{(q, \varepsilon)\},$$

ve všech ostatních případech přechod není definován, tj. $\delta(q, c, X) = \emptyset$.

3.4.10 Následující věta ukazuje, že třída jazyků přijímaných některým zásobníkovým automatem je třída bezkontextových jazyků.

Věta. Ke každému zásobníkovému automatu A existuje bezkontextová gramatika $\mathcal{G}$ taková, že

$$N(A) = L(\mathcal{G}).$$

□

Důkaz věty 3.4.10 (jedná se o opačnou implikaci k větě 3.4.7) je obtížnější. Je třeba ho rozdělit do dvou kroků. Nejprve se dokáže, že pro každý zásobníkový automat A existuje zásobníkový automat B s jedním stavem takový, že $N(A) = N(B)$.

V druhém kroku se pro zásobníkový automat B s jedním stavem vytvoří bezkontextová gramatika $\mathcal{G}$, která generuje stejná slova jako zásobníkový automat B přijme prázdným zásobníkem. Jedná se vlastně o opačný postup jako v důkazu věty z 3.4.7.

3.4.11 Deterministický zásobníkový automat. Na rozdíl od konečných automatů pro zásobníkové automaty platí, že v definici zásobníkového automatu bylo podstatné, že jsme ho definovali jako nedeterministický. Je to proto, že nedeterministické zásobníkové automaty přijímají větší třídu jazyků než deterministické. Dříve než se o tom přesvědčíme, zadefinujeme pojem deterministického zásobníkového automatu.

Definice. Je dán zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Řekneme, že A je *deterministický zásobníkový automat*, jestliže splňuje následující dvě podmínky:

- Pro každé $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ a $X \in \Gamma$ je $\delta(q, a, X)$ nejvýše jednoprvková (tj. $|\delta(q, a, X)| \leq 1$).
- Jestliže pro nějaké $q \in Q$ a $X \in \Gamma$ je $\delta(q, \varepsilon, X)$ neprázdné, pak pro každé $a \in \Sigma$ je $\delta(q, a, X)$ prázdná množina.

□

Uvědomte si, že předchozí dvě podmínky zajišťují, že v každém okamžiku máme vždy nejvýše jednu možnost, jak pokračovat. Ano, druhá podmínka říká, že si nikdy zásobníkový automat nemůže vybrat zda bude pokračovat „bez čtení vstupního symbolu“ nebo „se čtením vstupního symbolu“.

3.4.12 Jazyky přijímané deterministickým zásobníkovým automatem. Stejně jako u (nedeterministických) zásobníkových automatů rozlišujeme i u deterministických zásobníkových automatů příjímání koncovým stavem a přijímání prázdným zásobníkem. Tj. pro daný deterministický zásobníkový automat $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je

$$L(A) = \{u \,|\, (q_0, u, Z_0) \vdash_A^\star (p, \varepsilon, \gamma),\ p \in F\}$$

$$N(A) = \{u \,|\, (q_0, u, Z_0) \vdash_A^\star (p, \varepsilon, \varepsilon)\}.$$

□

Na rozdíl od nedeterministických zásobníkových automatů, pro deterministické zásobníkové automaty neplatí, že přijímání prázdným zásobníkem a přijímání koncovým stavem „je totéž“.

Definice. Jazyk L nazveme *deterministický*, jestliže existuje deterministický zásobníkový automat, který L přijímá koncovým stavem. □

Definice. Deterministický jazyk L nazveme *bezprefixový*, jestliže existuje deterministický zásobníkový automat, který ho přijímá prázdným zásobníkem. □

Název bezprefixový jazyk pochází z faktu, že jazyk přijímaný deterministickým zásobníkovým automatem prázdným zásobníkem nemůže obsahovat současně dvě různá slova u, v, kdy jedno je prefixem druhého.

3.4.13 Tvrzení. Každý jazyk přijímaný deterministickým zásobníkovým automatem prázdným zásobníkem je také přijímán (nějakým) deterministickým zásobníkovým automatem koncovým stavem.

Jinými slovy: Pro každý deterministický zásobníkový automat A existuje deterministický zásobníkový automat B takový, že

$$N(A) = L(B).$$

□

Důkaz je obdobný jako důkazu věty 3.4.5; jestliže vycházíme z deterministického zásobníkového automatu A, pak i automat B je deterministický.

Obdoba tvrzení 3.4.6 pro deterministické zásobníkové automaty neplatí. Konstrukce zásobníkového automatu B z automatu A totiž vede na nedeterministický zásobníkový automat. Navíc, jestliže deterministický zásobníkový automat A přijme slovo u prázdným zásobníkem, pak nemůže prázdným zásobníkem přijmout žádné slovo uv, kde $v \neq \varepsilon$; tato vlastnost ale neplatí pro všechny jazyky přijímané deterministickým zásobníkovým automatem koncovým stavem.

3.4.14 Vztah mezi třídami regulárních jazyků, bezprefixových jazyků a deterministických jazyků.

- Třída deterministických jazyků **obsahuje** třídu bezprefixových jazyků. To dokazuje tvrzení 3.4.13.
- Třída deterministických jazyků **obsahuje** třídu regulárních jazyků. Ano, na každý deterministický konečný automat se můžeme dívat jako na deterministický zásobníkový automat, kde na obsahu zásobníku nezáleží.
- Třída bezprefixových jazyků **neobsahuje** třídu regulárních jazyků. Ano, např. jazyk $\{a, aa\}$ je regulární (je konečný), ale není bezprefixový.
- Třída regulárních jazyků **neobsahuje** třídu bezprefixových jazyků. Ano, např. jazyk $L = \{wcw^R \mid w \in \{a,b\}^\star\}$ je bezprefixový, ale není regulární. (Jako cvičení sestrojte deterministický zásobníkový automat, který přijímá jazyk L prázdným zásobníkem.)

3.5 Uzávěrové vlastnosti bezkontextových jazyků

3.5.1 Tvrzení. Bezkontextové jazyky jsou uzavřeny na sjednocení.

To znamená, jsou-li L_1 a L_2 dva bezkontextové jazyky, pak také jazyk $L_1 \cup L_2$ je bezkontextový. □

Zdůvodnění: Jazyky L_1, L_2 jsou bezkontextové, proto existují bezkontextové gramatiky $\mathcal{G}_1 = (N_1, \Sigma, S_1, P_1)$ a $\mathcal{G}_2 = (N_2, \Sigma, S_2, P_2)$ takové, že $\mathcal{G}_1$ generuje L_1 a $\mathcal{G}_2$ generuje L_2. Přejmenujeme neterminály gramatiky $\mathcal{G}_2$ tak, aby gramatiky neměly žádný společný neterminál (tj. $N_1 \cap N_2 = \emptyset$).

Gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ generující jazyk $L_1 \cup L_2$ vytvoříme takto:

a) $N = N_1 \cup N_2 \cup \{S\}$, kde S je nový startovací symbol ($S \notin N_1 \cup N_2$).

b) $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$, tj. k pravidlům gramatik $\mathcal{G}_1$ a $\mathcal{G}_2$ přidáme dvě nová pravidla $S \to S_1$ a $S \to S_2$.

Není těžké ukázat, že gramatika $\mathcal{G}$ generuje jazyk $L_1 \cup L_2$.

3.5.2 Tvrzení. Bezkontextové jazyky jsou uzavřeny na zřetězení.

To znamená, jsou-li L_1 a L_2 dva bezkontextové jazyky, pak také jazyk $L_1 L_2$ je bezkontextový. □

Zdůvodnění: Jazyky L_1, L_2 jsou bezkontextové, proto existují bezkontextové gramatiky $\mathcal{G}_1 = (N_1, \Sigma, S_1, P_1)$ a $\mathcal{G}_2 = (N_2, \Sigma, S_2, P_2)$ takové, že $\mathcal{G}_1$ generuje L_1 a $\mathcal{G}_2$ generuje L_2. Přejmenujeme neterminály gramatiky $\mathcal{G}_2$ tak, aby gramatiky neměly žádný společný neterminál (tj. $N_1 \cap N_2 = \emptyset$).

Gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ generující jazyk $L = L_1 L_2$ vytvoříme takto:

a) $N = N_1 \cup N_2 \cup \{S\}$, kde S je nový startovací symbol ($S \notin N_1 \cup N_2$).

b) $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$, tj. k pravidlům gramatik $\mathcal{G}_1$ a $\mathcal{G}_2$ přidáme nové pravidlo $S \to S_1 S_2$.

Není těžké ukázat, že gramatika $\mathcal{G}$ generuje jazyk $L_1 L_2$.

3.5.3 Tvrzení. Bezkontextové jazyky jsou uzavřeny na Kleeneho operaci $\star$. □

To znamená, je-li L bezkontextový jazyk, pak také jazyk $L^\star$ je bezkontextový.

Zdůvodnění: Jazyk L je bezkontextový, proto existuje bezkontextová gramatika $\mathcal{G}_1 = (N_1, \Sigma, S_1, P_1)$ taková, že $L = L(\mathcal{G}_1)$. Gramatiku $\mathcal{G} = (N, \Sigma, S, P)$ generující jazyk $L^\star$ vytvoříme takto:

a) $N = N_1 \cup \{S\}$, kde S je nový startovací symbol ($S \notin N_1$).

b) $P = P_1 \cup \{S \to S_1 S, S \to \varepsilon\}$, tj. k pravidlům gramatik $\mathcal{G}_1$ a $\mathcal{G}_2$ přidáme dvě nová pravidla $S \to S_1 S$, $S \to \varepsilon$.

Není těžké ukázat, že gramatika $\mathcal{G}$ generuje jazyk $L^\star$.

3.5.4 Tvrzení. Bezkontextové jazyky **nejsou** uzavřeny na průnik.

To znamená, jsou-li L_1 a L_2 dva bezkontextové jazyky, pak jazyk $L_1 \cap L_2$ nemusí být bezkontextový. □

Zdůvodnění: Uvažujme jazyk $L_1 = \{0^k 1^n 2^n \mid k, n \geq 1\}$ a jazyk $L_2 = \{0^k 1^k 2^n \mid k, n \geq 1\}$. Oba tyto jazyky jsou bezkontextové, ale jejich průnik je jazyk $L = \{0^n 1^n 2^n \mid n \geq 1\}$, o kterém víme, že není bezkontextový (viz 3.2.14).

3.5.5 Tvrzení. Bezkontextové jazyky **nejsou** uzavřeny na doplněk.

To znamená, je-li L bezkontextový jazyk, pak jeho doplněk $\overline{L}$ nemusí být bezkontextový. □

Zdůvodnění: Kdyby třída bezkontextových jazyků byla uzavřena na doplňky, byla by uzavřena i na průniky, protože

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}},$$

což víme, že neplatí.

3.5.6 Tvrzení. Třída bezkontextových jazyků je uzavřena na průniky s regulárními jazyky.

To znamená, je-li L bezkontextový jazyk a R regulární jazyk, pak jazyk $L \cap R$ je bezkontextový. □

Nástin důkazu: Jazyk L je bezkontextový, tudíž existuje zásobníkový automat, označme ho $A = (Q_1, \Sigma, \Gamma, \delta_1, q_1, Z_1, F_1)$, takový, že A přijímá L koncovým stavem.

Jazyk R je regulární, tudíž existuje deterministický konečný automat $M = (Q_2, \Sigma, \delta_2, q_2, F_2)$, který přijímá jazyk R.

Zkonstruujeme zásobníkový automat B který přijímá jazyk $L \cap R$ koncovým stavem. (Všimněte si, že se v podstatě jedná o součinovou konstrukci, kterou známe z konečných automatů.)

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

- $Q = Q_1 \times Q_2$,
- $q_0 = (q_1, q_2)$,
- $Z_0 = Z_1$,
- $F = F_1 \times F_2$,
- přechodová funkce δ je definována takto:

$$\delta((q,p),a,X) = \{((r,s),\gamma) \mid (r,\gamma) \in \delta_1(q,a,X),\ s \in \delta_2^\star(p,a)\},$$

kde $q \in Q_1, p \in Q_2, a \in \Sigma \cup \{\varepsilon\}, X \in \Gamma$.

Jinými slovy: $((r,s),\gamma) \in \delta((q,p),a,Y)$ právě tehdy, když

a) buď $a \in \Sigma$, $(r,\gamma) \in \delta_1(q,a,Y)$ a $s = \delta_2(p,a)$,
b) nebo $a = \varepsilon$, $(r,\gamma) \in \delta_1(q,\varepsilon,Y)$ a $s = p$.

Uvědomte si, že B se v prvé složce chová jako zásobníkový automat A, ve druhé složce jako deterministický konečný automat M.

Není těžké nahlédnout, že zásobníkový automat B opravdu přijímá jazyk $L \cap R$.

3.5.7 Tvrzení. Bezkontextové jazyky jsou uzavřeny na reverzi.

To znamená, je-li L bezkontextový jazyk, pak také jazyk L^R je bezkontextový. □

Zdůvodnění: Jazyk L je bezkontextový, proto existuje bezkontextová gramatika $\mathcal{G} = (N, \Sigma, S, P)$, která ho generuje. Gramatika $\mathcal{G}_R$, která generuje jazyk L^R je (N, Σ, S, P_R), kde

$$A \to \alpha \in P_R \quad \text{iff} \quad A \to \alpha^R \in P.$$

Není těžké dokázat, že gramatika $\mathcal{G}$ generuje jazyk L^R.

3.5.8 Tvrzení. Třída bezkontextových jazyků je uzavřena na substituce.

Přesněji: Máme dány dvě abecedy Σ a Δ a substituci σ, která každému písmenu $a \in \Sigma$ přiřadí bezkontextový jazyk $\sigma(a) = L_a$ nad abecedou Δ. Je-li L_0 bezkontextový jazyk nad abecedou Σ, pak jeho obraz $\sigma(L_0)$ v substituci σ je také bezkontextový jazyk nad abecedou Δ. (Substituce byla zavedena v 2.7.2.) □

Nástin důkazu: Máme dánu CF gramatiku $\mathcal{G}_0 = (N_0, \Sigma, S_0, P_0)$ takovou, že $L_0 = L(\mathcal{G}_0)$ a dále pro každé $a \in \Sigma$ bezkontextovou gramatiku $\mathcal{G}_a = (N_a, \Delta, S_a, P_a)$ takovou, že $L_a = L(\mathcal{G}_a)$. Předpokládejme, že množiny neterminálů všech gramatik jsou po dvou disjunktní.

Zkonstruujeme CF gramatiku $\mathcal{G} = (N, \Delta, S_0, P)$ takto:

a) Množina neterminálu gramatiky $\mathcal{G}$ se skládá ze všech neterminálů $\mathcal{G}_0$ a $\mathcal{G}_a$, $a \in \Sigma$. Formálně $N = N_0 \cup \bigcup\{N_a \mid a \in \Sigma\}$.

b) Pravidla gramatiky $\mathcal{G}$ se skládají ze všech pravidel gramatik $\mathcal{G}_a$ a pravidla $\mathcal{G}_0$ jsou upravena tak, že v pravých stranách je každý terminál $a \in \Sigma$ je nahrazen neterminálem S_a. Přesněji $P = \bigcup\{P_a \mid a \in \Sigma\} \cup P'$, kde pravidla P' jsou tvořena všemi pravidly z P_0, ve kterých jsme terminál $a \in \Sigma$ nahradili neterminálem S_a.

Není těžké dokázat, že gramatika $\mathcal{G}$ generuje jazyk $\sigma(L_0)$.

3.5.9 Důsledek: Třída bezkontextových jazyků je uzavřena na homomorfismy. □

Protože každý homomorfismus je zvláštním případem substituce a protože každý jazyk obsahující jedno slovo je bezkontextový, vyplývá toto tvrzení z 3.5.8.

Uvedeme ještě jedno tvrzení, ale již bez důkazu.

3.5.10 Tvrzení. Třída bezkontextových jazyků je uzavřena na inversní homomorfní obrazy.

Přesněji: Je-li h homomorfismus abecedy Σ do $\Delta^\star$ a je-li L bezkontextový jazyk nad abecedou Δ, pak $h^{-1}(L)$ je bezkontextový jazyk nad abecedou Σ. □

3.5.11 Dyckovy jazyky. Na závěr si přiblížíme, co „dělá jazyk bezkontextovým"; jinými slovy, co je všem bezkontextovým jazykům společné. Jak uvidíme z následujícího tvrzení, bezkontextové jazyky (které nejsou regulární) v sobě obsahují „správné uzávorkování" a právě to popisují Dyckovy jazyky.

Definice. Bezkontextový jazyk L se nazývá *Dyckův jazyk*, jestliže je generován gramatikou $\mathcal{G} = (\{S\}, \Sigma, S, P)$, kde $\Sigma = \{a_1, a_1', a_2, a_2', \ldots, a_n, a_n'\}$ a P:

$$S \to \varepsilon \mid SS \mid a_1 S a_1' \mid \ldots \mid a_n S a_n'.$$

Jedná se vlastně o jazyk „správně uzávorkovaných výrazů", kde máme n různých otvíracích závorek a_i a zavíracích závorek a_i'.

3.5.12 Tvrzení. Pro každý bezkontextový jazyk L existuje regulární jazyk R, Dyckův jazyk D a homomorfismus h takový, že $L = h(R \cap D)$. □

3.5.13 Greibachové normální forma (tvar). Řekneme, že CF gramatika $\mathcal{G}$ je v *Greibachové normální formě* (v *Greibachové normálním tvaru*), jestliže všechna pravidla mají tvar:

$$A \to a\alpha, \quad \text{kde } a \in \Sigma \text{ a } \alpha \in N^\star.$$

To znamená, že pravá strana každého pravidla začíná jedním terminálem a po něm následuje několik (třeba i žádný) neterminálů.

Uvědomte si, že pro gramatiku v Greibachové normální formě derivace slova délky n má n kroků. Také je zřejmé, že gramatika v Greibachové normálním tvaru vygeneruje pouze slova délky aspoň 1.

3.5.14 Věta. Ke každému bezkontextovému jazyku L existuje CF gramatika v Greibachové normálním tvaru $\mathcal{G}$ taková, že

$$L(\mathcal{G}) = L \setminus \{\varepsilon\}.$$

□

Tuto větu dokážeme později, nejprve ukážeme několik kroků, které se při převodu CF gramatiky na Greibachové normální tvar používají. Jedná se o spojení pravidel a odstranění levých rekurzí. O levé rekurzi mluvíme, jestliže gramatika má pravidlo tvaru $A \to A\alpha$, kde $\alpha \in (N \cup \Sigma)^\star$.

3.5.15 Spojení pravidel. Máme CF gramatiku $\mathcal{G}$ a v ní pravidlo $A \to \alpha B \beta$. Jestliže

$$B \to \gamma_1 \,|\, \ldots \,|\, \gamma_k$$

jsou všechna pravidla s levou stranou B, pak nahrazením pravidla $A \to \alpha B \beta$ pravidly

$$A \to \alpha\gamma_1\beta \,|\, \ldots, \,|\, \alpha\gamma_k\beta$$

dostaneme gramatiku, která generuje stejný jazyk jako $\mathcal{G}$.

3.5.16 Odstranění levé rekurze. Předpokládejme, že v CF gramatice $\mathcal{G}$ máme pravidla

$$A \to A\alpha_1,\ A \to A\alpha_2,\ \ldots,\ A \to A\alpha_k$$

a pravidla

$$A \to \beta_1,\ A \to \beta_2,\ \ldots,\ A \to \beta_m$$

jsou všechna ostatní pravidla (tj. β_i nezačíná neterminálem A).

Pak tato pravidla nahradíme pravidly

$$A \to \beta_i,\ A \to \beta_i Z,\ Z \to \alpha_j,\ Z \to \alpha_j Z,\ \ i = 1, \ldots m,\ \ j = 1, \ldots, k;$$

kde Z je nový neterminál.

Tímto nahrazením dostaneme gramatiku generující stejný jazyk, která již nemá pro neterminál A levou rekurzi.

3.5.17 Postup nalezení gramatiky v Greibachové normálním tvaru. Máme nevypouštěcí CF gramatiku $\mathcal{G}$.

- Očíslujeme její neterminály, tj. $N = \{A_1, A_2, \ldots, A_n\}$.
- Postupným spojováním pravidel a odstraňováním levé rekurze získáme pravidla, kde pravá strana začíná terminálem nebo jsou tvaru $A_i \to A_j\alpha$ pro $i < j$.
- Postupným spojováním pravidel (od n do 1) dosáhneme toho, aby pravidla s levou stranou A_i měla tvar $A_i \to a\alpha$, kde $a \in \Sigma$.
- Zajistíme, aby pravidla obsahující nové neterminály (vzniklé při odstraňování levé rekurze) také měla požadovaný tvar.
- Jestliže nyní máme terminální symbol uvnitř pravé strany některého pravidla, zavedeme za něj nový neterminál.

Postup si ukážeme na příkladě.

3.5.18 Příklad. Je dána CF gramatika $\mathcal{G}$ pravidly

$$E \to E + T \,|\, T$$
$$T \to T * F \,|\, F$$
$$F \to (E) \,|\, a$$

Gramatika je nevypouštěcí, proto nejprve odstraníme levou rekurzi pro neterminál E:

Přidáme nový neterminál Z_E a zavedeme pravidla

$$E \to T \,|\, T\, Z_E$$
$$Z_E \to +T \,|\, +T\, Z_E$$

Obdobně pro neterminál T

$$T \to F \,|\, F\, Z_T$$
$$Z_T \to *F \,|\, *F\, Z_T$$

Pro neterminál F levou rekurzi nemáme.

Slučováním odstraníme pravidla $T \to F \,|\, F\, Z_T$ [uvědomte si, že máme pravidla $F \to (E) \,|\, a$]

$$T \to (E) \,|\, a \,|\, (E)\, Z_T \,|\, aZ_T$$

Podobně nyní odstraníme pravidla $E \to T\, Z_E \,|\, T$

$$E \to (E) \,|\, a \,|\, (E)\, Z_T \,|\, aZ_T \,|\, (E)Z_E \,|\, aZ_E \,|\, (E)\, Z_T Z_E \,|\, aZ_T Z_E$$

Nyní zavedeme ještě nový neterminál X, který použijeme k tomu, abychom odstranili výskyt terminálu) v pravidlech. Tím dostaneme gramatiku v Greibachové normálním tvaru:

$$E \to (EX \,|\, a \,|\, (EX\, Z_T \,|\, aZ_T \,|\, (EXZ_E \,|\, aZ_E \,|\, (EXZ_TZ_E \,|\, aZ_TZ_E$$
$$Z_E \to +T \,|\, +T\, Z_E$$
$$T \to (EX \,|\, a \,|\, (EXZ_T \,|\, aZ_T$$
$$Z_T \to *F \,|\, *F\, Z_T$$
$$F \to (EX \,|\, a$$
$$X \to)$$

3.6 Analýza shora

V jedné z minulých přednášek jsme si ukázali algoritmus CYK, který pro danou gramatiku v Chomského normálním tvaru a pro dané slovo rozhodne, zda je slovo gramatikou generováno. Velmi často je potřeba vědět nejen, zda je slovo vygenerováno, ale znát i posloupnost pravidel gramatiky, která byla použita v levé derivaci daného terminálního slova. A právě k tomuto úkolu se často používá tak zvaná analýza shora. Pro některé CF gramatiky je tato analýza jednoduchá, pro jiné je složitější – používá „backtracking“.

3.6.1 Analýza shora. Již víme, viz. věta 3.4.7, že ke každé CF gramatice $\mathcal{G}$ můžeme sestrojit zásobníkový automat A takový, že jazyk přijímaný A prázdným zásobníkem je jazyk $L(\mathcal{G})$. Přitom zásobníkový automat

- má-li na vrcholu zásobníku neterminál A nahradí ho na zásobníku některou pravou stranou α pravidla $A \to \alpha$;
- má-li na vrcholu zásobníku terminál a a tento terminál se shoduje s čteným vstupem slova, odstraníme a z vrcholu zásobníku a přejdeme na vstupu na další písmeno;
- má-li na vrcholu zásobníku terminál a a tento terminál se neshoduje s čteným symbolem slova, automat se neúspěšně zastaví.

Ukážeme si postup na příkladě.

3.6.2 Příklad. Je dána CF gramatika $\mathcal{G}$ pravidly:

$$S \to aSa, S \to bSb, S \to c.$$

Tato gramatika generuje jazyk $L = \{wcw^R \mid w \in \{a,b\}^\star\}$. Očíslujme jednotlivá pravidla

1. $S \to aSa$
2. $S \to bSb$
3. $S \to c$

Derivace slova $bacab \in L(\mathcal{G})$ je

$$S \Rightarrow bSb \Rightarrow baSab \Rightarrow bacab,$$

kde jsme nejprve použili pravidlo 2, pak pravidlo 1 a na konec pravidlo 3.

V odpovídajícím zásobníkovém automatu postupujeme takto:

$$(q, bacab, S) \vdash (q, bacab, bSb) \vdash (q, acab, Sb) \vdash (q, acab, aSab) \vdash (q, cab, Sab) \vdash$$

$$\vdash (q, cab, cab) \vdash (q, ab, ab) \vdash (q, b, b) \vdash (q, \varepsilon, \varepsilon).$$

Jednotlivé kroky byly:

1. Přepis neterminálu na vrcholu zásobníku podle některého pravidla gramatiky.
2. V případě, že na vrcholu zásobníku byl terminální symbol shodující se s právě čteným symbolem na vstupu, tzv. „krácení“, tj. přečtení symbolu na vstupu a odstranění vrcholu zásobníku.

Výsledkem analýzy je pak posloupnost čísel pravidel v pořadí v jakém byla použita. Tedy v našem případě je to posloupnost 2, 1, 3.

Uvědomte si, že kdybychom použili jiná pravidla v práci zásobníkového automatu, výpočet by skončil neúspěšně.

K tomu, abychom správně utvořili přijímající výpočet zásobníkového automatu, potřebujeme znát výsledek a to danou posloupnost čísel pravidel. Je-li jasné ze vstupního slova (písmene, které bude první čtelno), že existuje pouze jediné pravidlo, které může vést k úspěchu, je analýza snadná.

3.6.3 Jednoduché $LL(1)$ gramatiky. Bezkontextová gramatika $\mathcal{G}$ se nazývá *jednoduchá* $LL(1)$ gramatika, jestliže pro každé její pravidlo platí:

1. Pravá strana každého pravidla začíná terminálním symbolem.
2. Jestliže dvě pravidla mají stejnou levou stranu, pak se liší terminálním symbolem, kterým začíná pravá strana pravidla.

Pro jednoduché $LL(1)$ gramatiky je lehké provádět analýzu shora. Máme-li na vstupu slovo $w = a_1a_2\ldots a_k$, pak jako první má smysl použít pouze pravidlo $S \to \alpha$, pro které α začíná terminálem a_1 a takové pravidlo je nejvýše jedno. Po zkrácení všech terminálů, které se shodovaly (alespoň a_1 bylo možné zkrátit) se může stát:

1. slovo přijmeme (vyprázdnili jsme zásobník a současně přečetli celé slovo),
2. na vrcholu zásobníku byl jiný terminální symbol než byl čten na vstupu nebo jsme přečetli celé slovo a nevyprázdnili zásobník — v tomto případě automat neúspěšně skončí,
3. na vrcholu zásobníku je neterminální symbol A a na vstupu je symbol a_i.

V případě 3. z definice jednoduché $LL(1)$ gramatiky víme, že existuje nejvýše jedno pravidlo $A \to \beta$, kde a_i je první symbol β. Tedy opět máme nejvýše jednu možnost, jak postupovat, abychom se dostali slovo w vygenerovali.

Právě popsaný postup formalizuje pojem $LL(1)$ *analyzátoru*.

3.6.4 Příklad. Je dána bezkontextová gramatika $\mathcal{G}$ z příkladu 3.6.2, tj.

1. $S \to aSa$
2. $S \to bSb$
3. $S \to c$

Její $LL(1)$ analyzátor je následující tabulka:

	a	b	c	ε
S	$aSa, 1$	$bSb, 2$	$c, 3$	chyba
a	krátit	chyba	chyba	chyba
b	chyba	krátit	chyba	chyba
c	chyba	chyba	krátit	chyba
ε	chyba	chyba	chyba	přijmout

3.6.5 $LL(1)$ **analyzátor** je v podstatě tabulka, kde řádky odpovídají terminálům, neterminálům a prázdnému slovu, sloupce odpovídají terminálům a prázdnému slovu.

V tabulce je:

- V řádku označeném neterminálem X a sloupci označeném terminálem a pravá strana pravidla $X \to a\alpha \in P$ a číslo tohoto pravidla, v případě, že takové pravidlo existuje.
- V řádku označeném terminálem a a sloupci označeném stejným terminálem a je krátit (tj. z čteného slova odstraníme terminál a).
- V řádku označeném ε a sloupci označeném ε je přijmout (výpočet úspěšně skončil a posloupnost pravidel udává levou derivaci vstupního slova).
- Ve všech ostatních případech je v tabulce chyba (výpočet skončil neúspěšně, nezvolili jsme správné pořadí pravidel nebo slovo není gramatikou generováno).

3.6.6 $LL(1)$ **gramatiky.** V některých případech můžeme $LL(1)$ analyzátor vytvořit i pro gramatiky, které nejsou jednoduché $LL(1)$ gramatiky. Stačí, aby platilo, že v případě, kdy čteme na vstupu terminál a a v levé derivaci máme přepsat neterminál X, tak existuje nejvýše jedno pravidlo, které může vést k vygenerování daného slova.

Poznat $LL(1)$ gramatiku a vytvořit její $LL(1)$ analyzátor je složitější než v případě jednoduché $LL(1)$ gramatiky. Používají se k tomu následující pojmy.

Označme $\mathbf{Fi}(\alpha)$ množinu všech terminálních symbolů a na něž začíná některé terminální slovo generované z α. V případě, že z α se dá vygenerovat prázdné slovo, ε také patří do $\mathbf{Fi}(\alpha)$.

$LL(1)$ gramatika je zhruba řečeno gramatika splňující:

1. Jestliže v gramatice $\mathcal{G}$ existují dvě pravidla $A \to \alpha$ a $A \to \beta$ se stejnou levou stranou, pak pro každé dvě levé derivace $S \Rightarrow^\star uA\mu$ a $S \Rightarrow^\star vA\chi$, $u, v \in \Sigma^\star$, $\mu, \chi \in (N \cup \Sigma)^\star$ platí $\mathbf{Fi}(\alpha\mu) \cap \mathbf{Fi}(\beta\chi) = \emptyset$.
2. Jestliže navíc $\mathbf{Fi}(A)$ obsahuje ε, pak $\mathbf{Fi}(\mu)$ je také disjunktní s $\mathbf{Fi}(\alpha\mu)$ (a tedy i $\mathbf{Fi}(\beta\chi)$).

(Tyto podmínky zajišťují, že podle vstupního symbolu jednoznačně poznáme, které pravidlo je možné použít.)

3.6.7 Úlohy algoritmicky řešitelné pro bezkontextové gramatiky Pro bezkontextové gramatiky jsou řešitelné následující problémy:

a) Otázka zda daná bezkontextová gramatika generuje neprázdný jazyk.

Zde stačí provést první krok algoritmu redukce gramatiky. Jazyk generovaný gramatikou je neprázdný právě tehdy, když množina neterminálů, ze kterých lze vygenerovat terminální slovo, obsahuje startovací symbol.

b) Pro danou bezkontextovou gramatiku $\mathcal{G}$ a slovo w zjistit, zda $w \in L(\mathcal{G})$.

Zde stačí na příklad převést gramatiku do Chomského normálního tvaru a algoritmus CYK rozhodne, zda slovo je generováno nebo ne.

V další uvidíme, že pro řadu dalších otázek, např. zda dvě bezkontextové gramatiky generují alespoň jedno společné slovo, algoritmus, který by je vyřešil, neexistuje.

3.6.8 Kontextové gramatiky, kontextové jazyky Víme, že jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$ není bezkontextový. Existuje však pro něj kontextová gramatika, která jej generuje (gramatiku si ukážeme na přednášce).

I kontextové jazyky mají svůj nástroj v podobě „zobecněného“ automatu, který přijímá právě je. Jedná se o specielní případ Turingova stroje. Pojem Turingova stroje si přiblížíme v následující kapitole.

Kapitola 4

Turingovy stroje

4.1 Turingův stroj

V kapitole zabývající se gramatikami jsme uvedli Chomského hierarchii na klasifikaci gramatik a jimi generovaných jazyků. Jednalo se o (viz **??**

1. Gramatiky typu 0 a jazyky typu O; nejobecnější gramatiky a jim odpovídající jazyky.
2. Gramatiky typu 1 a jazyky typu 1, též kontextové gramatiky a kontextové jazyky.
3. Gramatiky typu 2 a jazyky typu 2; též bezkontextové gramatiky a bezkontextové jazyky.
4. Gramatiky typu 3 a jazyky typu 3; též regulární (či levé lineární) gramatiky a regulární (či levé lineární) jazyky.

Pro regulární jazyky existují deterministické konečné automaty, které přijímají právě regulární jazyky. Pro bezkontextové jazyky existují zásobníkové automaty, které přijímají právě bezkontextové jazyky. Turingovy stroje jsou „stroje“, které přijímají právě jazyky typu 0.

Nejprve uvedeme Turingův stroj neformálně a teprve potom uvedeme jeho formální definici.

4.1.1 Turingův stroj Turingův stroj si můžeme představit takto: skládá se

- z řídící jednotky, která se může nacházet v jednom z konečně mnoha stavů,
- potenciálně nekonečné pásky rozdělené na jednotlivá pole a
- hlavy, která umožňuje číst obsah polí a přepisovat obsah polí pásky.

Na základě informace X, která je přečtena na pásce, a na základě stavu q, ve kterém se nachází řídící jednotka Turingova stroje, se řídící jednotka přesune do stavu p, pole pásky přepíše na Y a hlava se přesune buď doprava nebo doleva, nebo nemůže pokračovat a zastaví svou práci (tato akce je popsána tzv. přechodovou funkcí).

4.1.2 Formální definice. Turingův stroj je sedmice $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, kde

- Q je konečná množina stavů,
- Σ je konečná množina vstupních symbolů,
- Γ je konečná množina páskových symbolů, přitom $\Sigma \subset \Gamma$,
- B je prázdný symbol (též nazývaný *blank*), jedná se o páskový symbol, který není vstupním symbolem, (tj. $B \in \Gamma \setminus \Sigma$),
- δ je přechodová funkce, tj. parciální zobrazení z množiny $Q \times \Gamma$ do množiny $(Q \setminus F) \times \Gamma \times \{L, R\}$, (zde L znamená pohyb hlavy o jedno pole doleva, R znamená pohyb hlavy o jedno pole doprava),
- $q_0 \in Q$ je počáteční stav a
- $F \subseteq Q$ je množina koncových stavů.

4.1.3 Konfigurace. Konfiguraci Turingova stroje plně popisuje obsah pásky, pozice hlavy na pásce a stav, ve kterém se nachází řídící jednotka. Jestliže na pásce jsou v k polích symboly $X_1\,X_2\,\ldots\,X_k$, všechna pole s větším i menším číslem již obsahují pouze B, řídící jednotka je ve stavu q a hlava čte symbol X_i, tak danou konfiguraci zapisujeme

$$X_1\,X_2\,\ldots\,X_{i-1}\,q\,X_i\,X_{i+1}\,\ldots\,X_k.$$

4.1.4 Počátek práce Turingova stroje. Na začátku práce se Turingův stroj nachází v počátečním stavu q_0, na pásce má na n polích vstupní slovo $a_1\,a_2\,\ldots\,a_n$ ($a_i \in \Sigma$), ostatní pole obsahují blank B a hlava čte symbol a_1. Tedy formálně je počáteční konfigurace $q_0\,a_1\,\ldots\,a_n$.

4.1.5 Krok Turingova stroje. Předpokládejme, že se Turingův stroj nachází v konfiguraci $X_1\,X_2\,\ldots\,X_{i-1}\,q\,X_i\,\ldots\,X_k$. Pak v jednom kroku udělá následující:

Jestliže $\delta(q, X_i) = (p, Y, R)$, stroj se přesune do stavu p, na pásku napíše symbol Y (místo X_i) a hlavu posune o jedno pole doprava. Formálně to zapisujeme:

$$X_1\,X_2\,\ldots\,X_{i-1}\,q\,X_i\,\ldots\,X_k \vdash X_1\,X_2\,\ldots\,X_{i-1}\,Y\,p\,X_{i+1}\,\ldots\,X_k.$$

Jestliže je $i = k$ a Turingův stroj se má posunout hlavu doprava, pak

$$X_1\,X_2\,\ldots\,q\,X_k \vdash X_1\,X_2\,\ldots\,X_{k-1}\,Y\,p\,B.$$

Jestliže $\delta(q, X_i) = (p, Y, L)$, stroj napíše na pásku Y (místo X_i) a posune hlavu o jedno pole doleva. Formálně to zapisujeme:

$$X_1\,X_2\,\ldots\,X_{i-1}\,q\,X_i\,\ldots\,X_k \vdash X_1\,X_2\,\ldots\,X_{i-2}\,p\,X_{i-1}\,Y\,X_{i+1}\,\ldots\,X_k.$$

Jestliže je $i = 1$ a Turingův stroj se má posunout hlavu doleva, pak

$$q\,X_1\,X_2\,\ldots\,X_k \vdash p\,B\,Y\,X_2\,\ldots\,X_k.$$

4.1.6 Výpočet Turingova stroje je posloupnost jeho kroků, která začíná v počáteční konfiguraci. Tedy jedná se o reflexivní a tranzitivní uzávěr $\vdash^\star$ relace $\vdash$ (na množině všech konfigurací daného Turingova stroje).

Říkáme, že se *Turingův stroj zastavil* v případě, kdy nemůže udělat další krok.

Poznámka. Je zřejmé, že se Turingův stroj nemusí vždy nad vstupem zastavit. Může se zacyklit, ale také může pracovat do nekonečna, aniž by se zacyklil – např. může stále na konec použité pásky připisovat symbol. Na druhé straně se v koncovém stavu vždy zastaví. V takovém případě říkáme, že se Turingův stroj zastaví *úspěšně*, zastaví-li se v nekoncovém stavu, říkáme, že se zastavil *neúspěšně*.

4.1.7 Jazyk přijímaný Turingovým strojem. Vstupní slovo $w \in \Sigma^\star$ je *přijato* Turingovým strojem právě tehdy, když se Turingův stroj při práci na slově w dostane do koncového stavu. Tedy formálně: slovo $w \in \Sigma^\star$ je *přijato Turingovým strojem* právě tehdy, když

$$q_0\,w \vdash^\star \alpha q \beta \text{ pro nějaké } q \in F \text{ a } \alpha, \beta \in \Gamma^\star.$$

Množina slov $w \in \Sigma^\star$, které Turingův stroj přijímá, se nazývá *jazyk přijímaný* Turingovým strojem M a značíme ho $L(M)$.

4.1.8 Poznámka. Existuje několik základních modelů Turingova stroje. My jsme si vybrali ten, který má nekonečnou pásku „na obě strany“, posouvá hlavu vždy o jedno políčko a v koncovém stavu již nemůže provádět další krok.

Jiné varianty jsou:

- pásau s pevným levým okrajem. Pak je na začátku práce Turingova stroje vstupní slovo napsáno vždy na prvních k polích pásky a hlava čte první pole pásky. V tomto modelu je trochu obtížnější návrh strojů, protože je potřeba dát pozor, aby Turingův stroj nepřekročil levý okraj — pak by se neúspěšně zastavil.

- Turingův stroj může kromě posunu hlavy doprava nebo doleva nechat hlavu stát. To znamená, že po provedení kroku čte hlava stejné pole jako v předchozím kroku.
- Přechodová funkce může být definována i pro koncový stav a některý páskový symbol.

Všechny tyto modifikace ale neznamenají žádný rozdíl v tom, jaké jazyky jsou Turingovým strojem přijaty.

4.1.9 Věta. Jestliže jazyk L je přijímán Turingovým strojem, pak k němu existuje gramatika typu 0, která ho generuje.

Ke každé gramatice $\mathcal{G}$ typu 0 existuje Turingův stroj, který přijímá jazyk $L(\mathcal{G})$. □

Důkaz této věty neuvádíme.

4.1.10 Nedeterministický Turingův stroj se od deterministického Turingova stroje liší v tom, že přechodová funkce neurčuje nutně jen nejvýše jeden přechod, ale takových přechodů může být více.

Formálně je rozdíl pouze v definici δ a to

$$\delta\colon (Q \setminus F) \times \Gamma \longrightarrow \mathcal{P}_f(Q \times \Gamma \times \{L, R\}).$$

Platí věta, že ke každému nedeterministickému Turingovu stroji M existuje deterministický Turingův stroj M_1 tak, že $L(M) = L(M_1)$. To znamená, že nedeterministické Turingovy stroje přijímají také jen jazyky typu 0.

4.1.11 Předchozí věta nám říká, že Turingovy stroje jsou „výpočetním modelem“ pro jazyky typu 0, tj. jazyky generované tou nejobecnější gramatikou. Již víme, že „výpočetním modelem“ regulárních jazyků jsou konečné automaty, „výpočetním modelem“ bezkontextových jazyků pak zásobníkové automaty.

Obdobný model pro kontextové jazyky (tj. jazyky typu 1) jsou lineárně omezené automaty. Jedná se v podstatě o Turingův stroj, kde ovšem je možné používat pouze pole pásky, ve kterých se nacházelo vstupní slovo. Jakékoli překročení ať doprava nebo doleva znamená neúspěšné zastavení.

4.2 Nerozhodnutelnost

V této sekci si ukážeme, že existují jazyky / úlohy, pro které neexistuje Turingův stroj, který by je rozhodoval (a tudíž ani algoritmus, který by je řešil).

4.2.1 Rekursivně spočetné jazyky. Řekneme, že jazyk L je *rekursivně spočetný*, jestliže existuje Turingův stroj M, který tento jazyk přijímá.

Jinými slovy, M se pro každé slovo w, které patří do L, úspěšně zastaví a pro slovo w, které nepatří do L, se buď zastaví neúspěšně nebo se nezastaví vůbec.

4.2.2 Rekursivní jazyky. Řekneme, že jazyk L je *rekursivní*, jestliže existuje Turingův stroj M, který nejen přijímá jazyk L, ale na slovech, které nepatří do jazyka se zastaví neúspěšně. V takovém případě říkáme, že Turingův stroj jazyk *rozhoduje*.

4.2.3 Definice. Jazykům, které nejsou rekursivní, říkáme, že jsou *nerozhodnutelné* nebo *algoritmicky neřešitelné*.

4.2.4 Věta. Problém zastavení Turingova stroje (tzv. halting problem) je algoritmicky neřešitelný; tj. jeho jazyk je nerozhodnutelný. □

Důkaz je založen na několika konstrukcích/pozorování:

- Nejprve se zkonstruuje univerzální Turingův stroj, který umí simulovat práci libovolného Turingova stroje nad libovolným slovem.

- Kdyby tento univerzální Turingův stroj uměl rozhodnout, zda se Turingův stroj zastaví nad nějakým vstupem, tak bychom také měli Turingův stroj M, který by rozhodl, že se Turingův stroj nad slovem **nezastaví**.
- Kdybychom nyní pustili M na vstup, který je kódem M, dostaneme spor — M by se měl zastavit právě tehdy, když se nezastaví.

4.2.5 Postův korespondenční problém (PCP). Jsou dány dva seznamy neprázdných slov A, B nad danou abecedou Σ.

$$A = (w_1, w_2, \ldots, w_k), \quad B = (x_1, x_2, \ldots, x_k),$$

kde $w_i, x_i \in \Sigma^+$, $i = 1, 2, \ldots, k$. Řekneme, že dvojice A, B *má řešení*, jestliže existuje posloupnost $i_1, i_2, \ldots, i_r$ indexů, tj $i_j \in \{1, 2, \ldots, k\}$, taková, že

$$w_{i_1}\, w_{i_2}\, \ldots\, w_{i_r} \;=\; x_{i_1}\, x_{i_2}\, \ldots\, x_{i_r}.$$

Otázka: Má daná instance řešení?

4.2.6 Příklady.

1. Jsou dány seznamy

	1	2	3	4	5
A	011	0	101	1010	010
B	1101	00	01	00	0

   Tato instance má řešení, např. $2, 1, 5$ je

$$w_2\, w_1\, w_5 = 0011010 = x_2\, x_1\, x_1\, x_4\, x_1\, x_5.$$

   Dalšími řešeními jsou např. $2, 1, 1, 3, 5$ nebo $2, 1, 1, 4, 1, 5$.

2. Jsou dány seznamy

	1	2	3	4	5
A	11	0	101	1010	010
B	101	00	01	00	0

   Tato instance nemá řešení.

4.2.7 Věta. Kdyby byl algoritmicky řešitelný Postův korespondenční problém, byl by algoritmicky řešitelný i problém zastavení Turingova stroje. □

Větu nedokazujeme.

4.2.8 Další algoritmicky neřešitelné úlohy:

1. Pro dané bezkontextové gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$ rozhodnout, zda obě generují aspoň jedno stejné slovo, tj. zda $L(\mathcal{G}_1) \cap L(\mathcal{G}_2) \neq \emptyset$.
2. Pro dané bezkontextové gramatiky $\mathcal{G}_1$ a $\mathcal{G}_2$ rozhodnout, zda přijímají stejný jazyk, tj. zda $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.
3. Pro danou bezkontextovou gramatiku $\mathcal{G}_1$ rozhodnout, zda je víceznačná.
4. Pro danou bezkontextovou gramatiku $\mathcal{G}_1$ rozhodnout, zda přijímá všechna slova, tj. zda $L(\mathcal{G}_1) = \Sigma^\star$.
5. Pro danou bezkontextovou gramatiku $\mathcal{G}_1$ a regulární jazyk R rozhodnout, zda $R \subseteq L(\mathcal{G}_1)$.

4.2.9 Připomeňme, že jsou algoritmicky řešitelné následující úlohy:

1. Zda daná bezkontextová gramatika $\mathcal{G}$ generuje alespoň jedno slovo; tj. zda $L(\mathcal{G}) \neq \emptyset$.
2. Pro danou bezkontextovou gramatiku $\mathcal{G}$ a dané slovo w rozhodnout, zda $\mathcal{G}$ generuje w, tj. zda $w \in L(\mathcal{G})$.
3. Pro danou bezkontextovou gramatiku $\mathcal{G}$ a regulární jazyk R rozhodnout, zda $L(\mathcal{G}) \subseteq R$.