AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Gramatyki regularne

Teoria automatów i języków formalnych

Dr inż. Janusz Majewski
Katedra Informatyki

Gramatyki regularne

G = < V,Σ,P,S > jest gramatyką prawostronnie liniową, jeśli jej produkcje mają postać:

$$\left.\begin{array}{l}(i)\ U \to xW \\ (ii)\ U \to x\end{array}\right\}$$ x∈Σ* U,W∈V

G = < V,Σ,P,S > jest gramatyką prawostronnie regularną, jeśli jej produkcje mają postać:

$$\left.\begin{array}{l}(i)\ U \to aW \\ (ii)\ U \to a\end{array}\right\}$$ a∈Σ U,W∈V

oraz (iii) jeśli (S→ε)∈P to S nie występuje w prawych stronach żadnej produkcji.

(Często w definicji gramatyki regularnej pomija się warunek (iii) dotyczący niewystępowania S w prawych stronach produkcji – dopuszcza się za to produkcje U→ε ; U∈V)

Analogicznie określa się :

-gramatyki lewostronnie liniowe

$$\left.\begin{array}{l}U \to Wx \\ U \to x\end{array}\right\}$$ x ∈Σ* U,W∈V

-gramatyki lewostronnie regularne

$$\left.\begin{array}{l}U \to Wa \\ U \to a\end{array}\right\}$$ a∈Σ U,W∈V

Szkic algorytmu przekształcania gramatyki prawostronnie liniowej w prawostronnie regularną

Wejście: $G = < V,\Sigma,P,S > \in G_{PLN}$

Wyjście: $G' = < V,\Sigma,P,S > \in G_{PRG}$ taka, że $L(G) = L(G')$

Metoda:

$P' := P; \quad V' := V;$

for $(A \to \alpha) \in P$ do

begin

if $\alpha = xB$ and $x = x_1...x_n \in \Sigma^*$ and $|x| \geq 2$ then

begin

$C := \{ A \to x_1C_1 ; C_1 \to x_2C_2 ; ... ; C_{n-1} \to x_nB \};$

$P' := P' \setminus \{ A \to xB \} \cup C ;$

$V' := V' \cup \{ C_1,...,C_{n-1} \} ;$

end;

if $\alpha = x$ and $x = x_1...x_n \in \Sigma^*$ and $|x| \geq 2$ then

begin

$C := \{ A \to x_1C_1 ; C_1 \to x_2C_2 ; ... ; C_{n-}1 \to x_nB \};$

$P' := P' \setminus \{ A \to x \} \cup C ;$

$V' := V' \cup \{ C_1,...,C_{n-1} \} ;$

end;

end;

Szkic algorytmu przekształcania gramatyki prawostronnie liniowej w prawostronnie regularną

- Usunąć ε - produkcje (w razie potrzeby) ;
- Usunąć reguły łańcuchowe ;
- Usunąć symbol początkowy z prawych stron produkcji (w razie potrzeby);

/* algorytm usuwania symbolu początkowego będzie podany później */

Przykład:

$A \to abB$	$\mathbf{A \to aA_1}$ $\mathbf{A_1 \to bB}$		$A \to aA_1$ $A_1 \to bB$	$A \to aA_1$ $A_1 \to bB$
$A \to ba$	$\mathbf{A \to bA_2}$ $\mathbf{A_2 \to a}$		$A \to bA_2$ $A_2 \to a$	$A \to bA_2$ $A_2 \to a$
$B \to b$	$B \to b$		$B \to b$	$B \to b$
$B \to A$	$B \to A$		$B \to A$	$\Rightarrow^{**}$ $\mathbf{B \to aA_1}$ $\mathbf{B \to bA_2}$
$B \to \varepsilon$	$B \to \varepsilon$	$\overset{*}{\Rightarrow}$	$\mathbf{A_1 \to b}$	$A_1 \to b$
Gramatyka prawostronnie liniowa	* - usunięcie ε-produkcji		** - usunięcie produkcji łańcuchowych	Gramatyka prawostronnie regularna

Przekształcenie gramatyki lewostronnie regularnej w prawostronnie regularną

Wejście: $G = < V,\Sigma,P,S > \in \mathcal{G}_{LRG}$; G – nie zawiera symbolu początkowego S w prawych stronach produkcji

Wyjście: $G' = < V',\Sigma,P',S' > \in \mathcal{G}_{PRG}$ taka, że $L(G') = L(G)$

Metoda:

$P' := \emptyset$;

$V' := V \cup \{S'\} - \{S\}$

for $(A \rightarrow a) \in P : a \in \Sigma$ do

    if A=S

        then $P' := P' \cup \{S' \rightarrow a\}$

        else $P' := P' \cup \{S' \rightarrow aA\}$;

for $(A \rightarrow Ba) \in P : B \in V , a \in \Sigma$ do

    if A=S

        then $P' := P' \cup \{B \rightarrow a\}$

        else $P' := P' \cup \{B \rightarrow aA\}$;

Przekształcenie gramatyki lewostronnie regularnej w prawostronnie regularną

Przykład:

$G=<\{S,A\},\{a,b\},P,S>$ $G'=<\{S',A\},\{a,b\},P',S'>$

$S \rightarrow a$
$S \rightarrow Ab$
$A \rightarrow a$
$A \rightarrow Ab$

$S' \rightarrow a$
$A \rightarrow b$
$S' \rightarrow aA$
$A \rightarrow bA$

Usuwanie produkcji końcowych (kosztem wprowadzenia ε-produkcji)

Produkcje końcowe: $U \to a : U \in V , a \in \Sigma$

Wejście: $G = < V,\Sigma,P,S > \in G_{PRG}$

Wyjście: $G' = < V',\Sigma,P',S >$ - bez produkcji końcowych, taka że $L(G') = L(G)$

Metoda:

$V' := V;$

$P' := P;$

<u>for</u> $(A \to x) \in P$ <u>do</u>

<u>if</u> $x \in \Sigma$ <u>then</u>

<u>begin</u>

$V' := V' \cup \{ A_x \};$

$P' := P' \cup \{ A \to xA_x , A_x \to \varepsilon \}$

$- \{ A \to x \};$

<u>end</u>;

Usuwanie produkcji końcowych (kosztem wprowadzenia ε-produkcji)

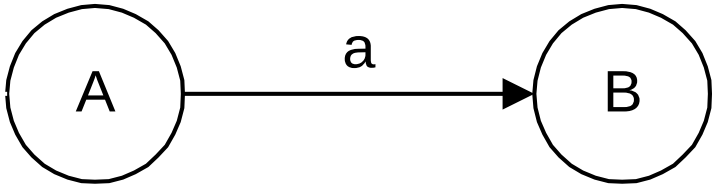
Przykład:

G = < {S,A,B,C,R,Q}, {a,b}, P, S >

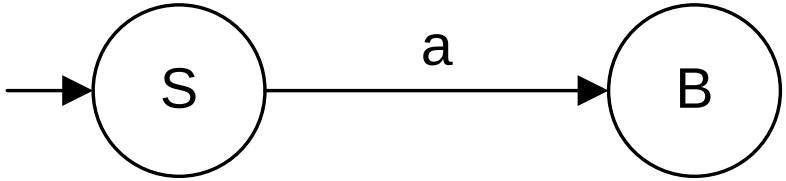
$S \to bS$	$S \to bS$
$S \to aA$	$S \to aA$
$S \to aB$	$S \to aB$
$B \to bC$	$B \to bC$
$C \to aA$	$C \to aA$
$A \to bR$	$A \to bR$
$Q \to aB$	$Q \to aB$
$A \to b$	**$A \to bD$,** $D \to \varepsilon$

D – Symbol końcowy (nie mylić z symbolem terminalnym)

Wykres gramatyki (bez produkcji końcowych) w postaci grafu zorientowanego

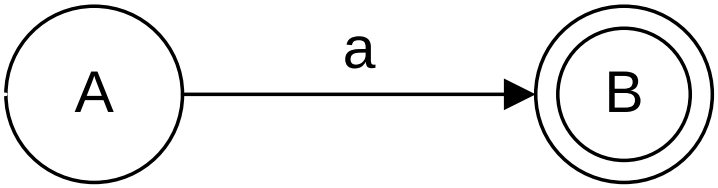


$A \rightarrow aB$
$A,B \in V \quad a \in \Sigma$
$A \neq S \quad (B \rightarrow \varepsilon) \notin P$



symbol początkowy gramatyki

$S \rightarrow aB$
$a \in \Sigma,\ B \in V$
$(B \rightarrow \varepsilon) \notin P$



symbol końcowy

$A \rightarrow aB$
$B \rightarrow \varepsilon$
$a \in \Sigma;\ A,B \in V$

Przykład (1)

$S \rightarrow bS$

$S \rightarrow aA$

$S \rightarrow aB$

$B \rightarrow bC$

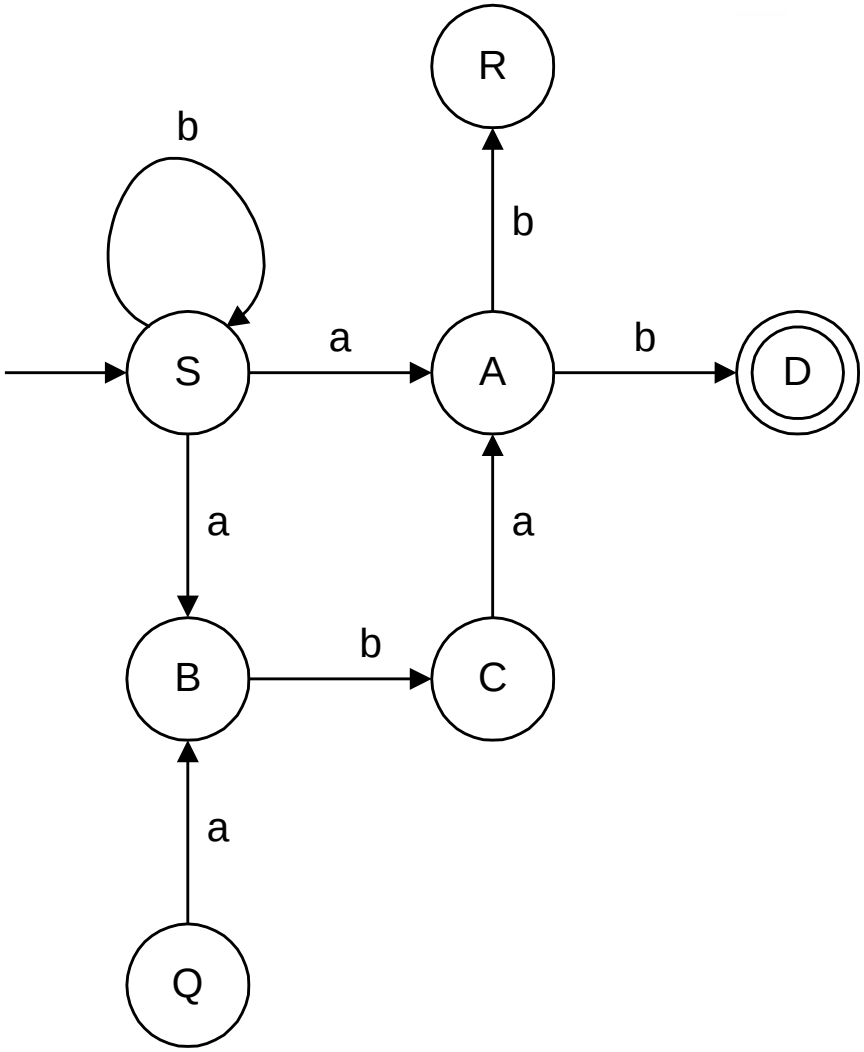
$C \rightarrow aA$

$A \rightarrow bR$

$Q \rightarrow aB$

$A \rightarrow bD$

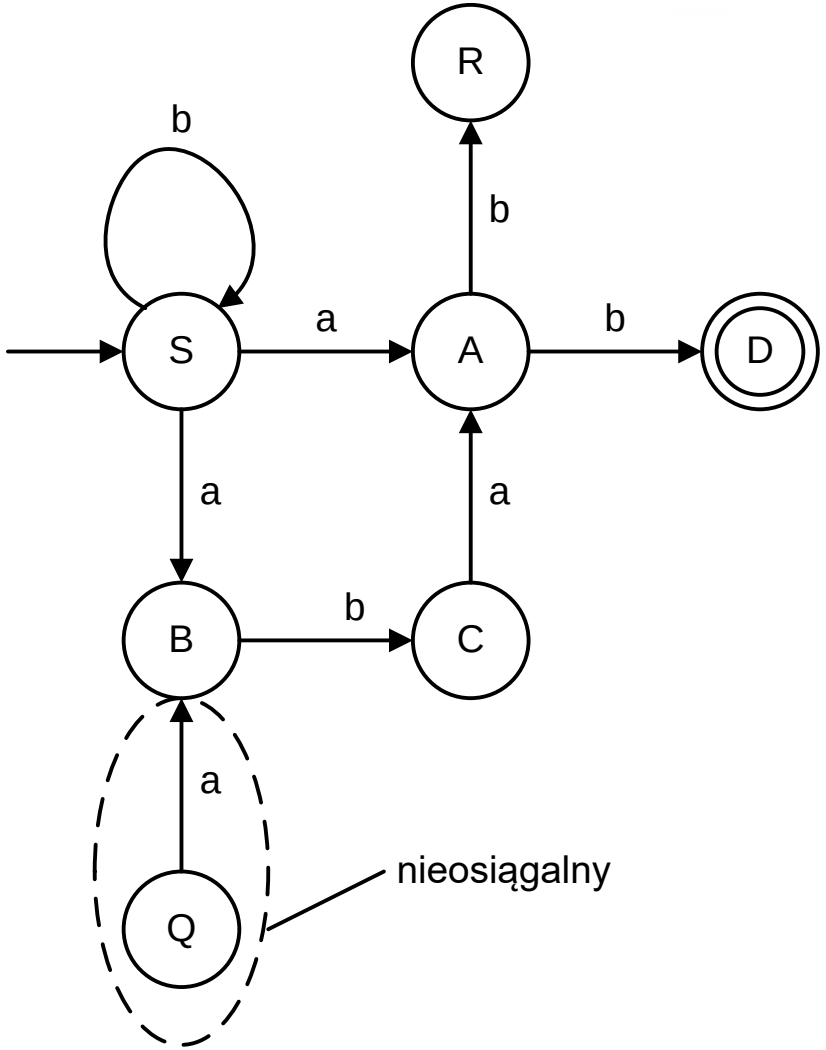
$D \rightarrow \varepsilon$



Przykład (2)

Usuwanie symboli nieosiągalnych

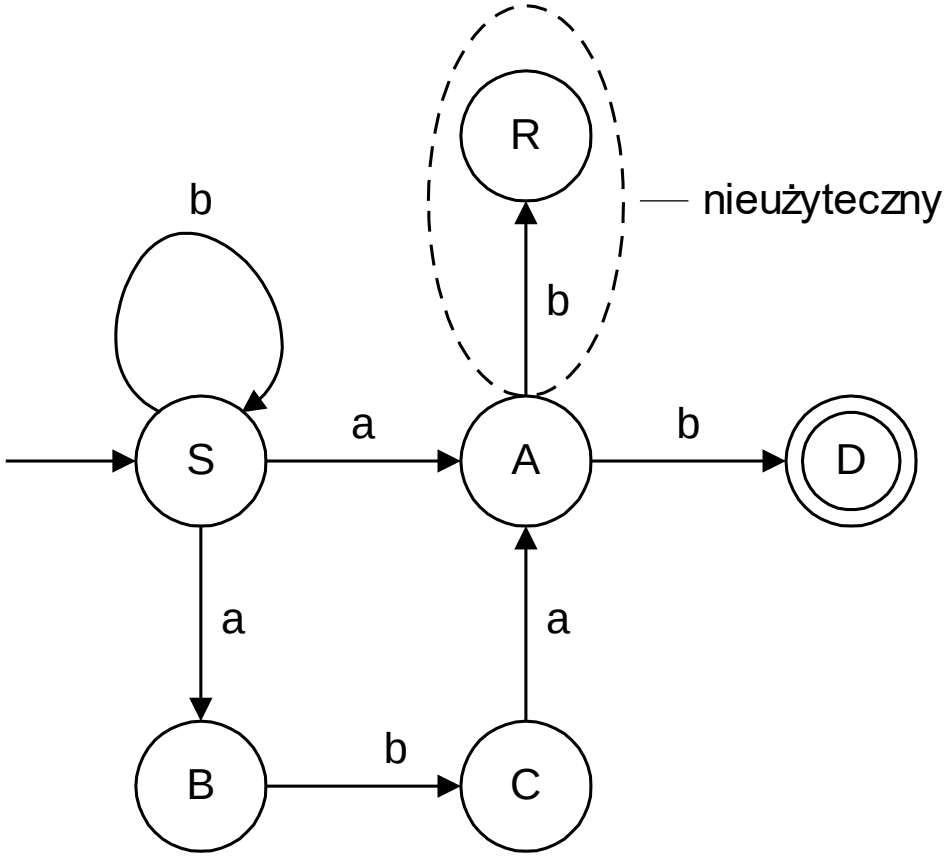
Można usunąć każdą produkcję U $\rightarrow$ aW, taką że U$\neq$S oraz symbol U nie występuje po prawej stronie żadnej produkcji.



Przykład (3)

Usuwanie symboli nieużytecznych

Można usunąć wszystkie produkcje U $\rightarrow$ aW, gdzie W nie jest symbolem końcowym oraz W nie występuje po lewej stronie żadnej produkcji, z wyjątkiem być może produkcji typu W $\rightarrow$ aW.



Powyższe stwierdzenia nie są precyzyjne. Dokładne algorytmy podano dla gramatyk bezkontekstowych. ($G_{RG} \subseteq G_{BK}$)

Przypomnienie o ścieżkach w grafie skierowanym

Ścieżka – ciąg wierzchołków grafu zgodny z istniejącymi krawędziami i ich zorientowaniem.

<u>Definicje:</u> Ścieżka końcowa $\Leftrightarrow$ ścieżka $K_0K_1\ldots K_n$ taka, że

$K_0 = S$

$K_n \in$ zbiór symboli końcowych

Ścieżka wyznaczona przez słowo $x=x_1x_2\ldots x_n \Leftrightarrow$ ścieżka końcowa $K_0K_1\ldots K_n$ taka, że

$(K_0 \rightarrow x_1K_1) \in P$

$(K_1 \rightarrow x_2K_2) \in P$

...

$(K_{n-1} \rightarrow x_nK_n) \in P$

$(K_n \rightarrow \varepsilon) \in P$

$x \in L(G) \Leftrightarrow \exists$ ścieżka końcowa wyznaczona przez słowo x.

Graf automatu skończonego $\equiv$ graf gramatyki prawostronnie regularnej bez produkcji końcowych

Automat skończony (lub gramatyka regularna bez produkcji końcowych) ⇒ wyrażenie regularne

Twierdzenie (tzw. pierwsze twierdzenie Kleene'a): Język generowany przez gramatykę prawostronnie regularną bez produkcji końcowych (czyli język akceptowany przez automat skończony) jest językiem regularnym (tzn. określonym przez wyrażenie regularne).

Sporządzamy graf gramatyki spełniający poniższe założenie.

Założenie: graf gramatyki jest spójny i każdy wierzchołek grafu jest albo końcowy albo leży przynajmniej na jednej ścieżce końcowej.

Automat skończony (lub gramatyka regularna bez produkcji końcowych) ⇒ wyrażenie regularne

[Dowód przez indukcję względem liczby krawędzi w grafie].

Podstawa indukcji: Jeśli graf ma $k=0$ krawędzi, to spełniając założenie ma albo jeden wierzchołek końcowy (a zarazem początkowy), więc język $L=\{\varepsilon\}$ (wyrażenie **ε**), albo nie ma żadnego wierzchołka, wtedy język $L=\emptyset$ (wyrażenie ∅)

Automat skończony (lub gramatyka regularna bez produkcji końcowych) $\Rightarrow$ wyrażenie regularne

Krok indukcyjny:

Założenie indukcyjne: graf mający $k<n$ krawędzi reprezentuje język regularny; graf ten jest spójny i każdy wierzchołek grafu jest albo końcowy albo leży przynajmniej na jednej ścieżce końcowej.

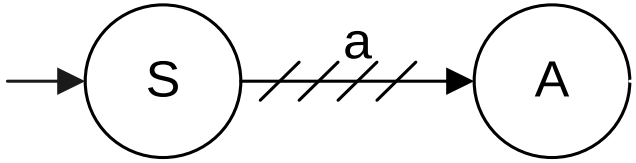
Teza indukcyjna: graf mający $k+1$ krawędzi reprezentuje język regularny; graf ten jest spójny i każdy wierzchołek grafu jest albo końcowy albo leży przynajmniej na jednej ścieżce końcowej.

Z grafu G dla $k+1 \leq n$ krawędzi usuwamy jedną krawędź odpowiadającą produkcji typu $S \rightarrow aA$ (S – symbol początkowy)

Automat skończony ⇒ wyrażenie regularne

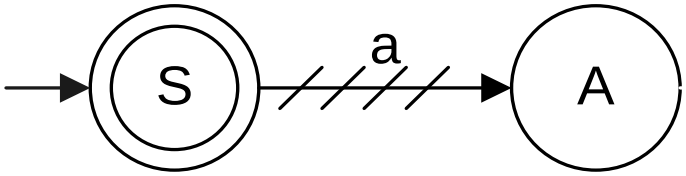
Rozważamy cztery grafy:

G_1:



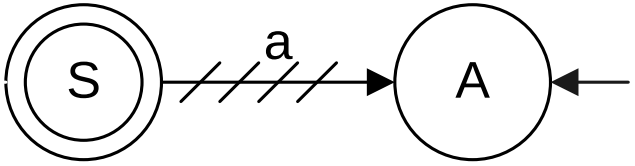
W G_1 reszta bez zmian.

G_2:



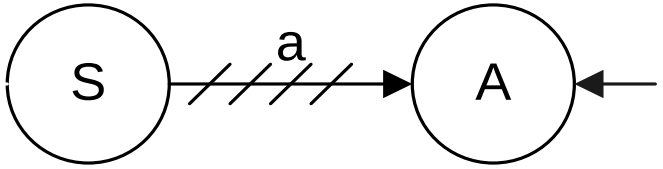
W G_2 wierzchołkiem początkowym i jedynym końcowym jest S.

G_3:



W G_3 wierzchołkiem początkowym jest A zaś jedynym końcowym jest S

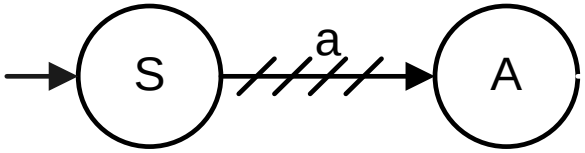
G_4:



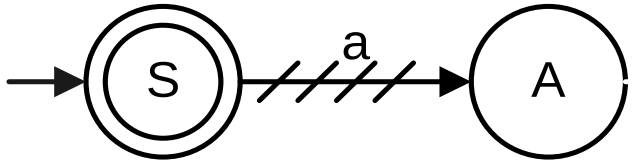
W G_4 wierzchołkiem początkowym jest A, końcowe bez zmian

Automat skończony ⇒ wyrażenie regularne

G_1:

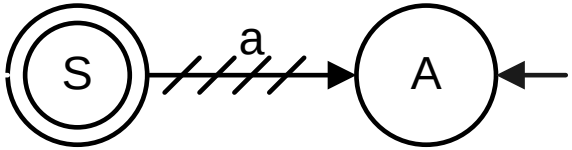


G_2:



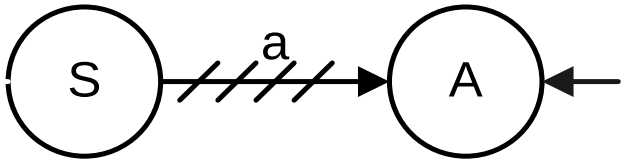
W G_2 wierzchołkiem początkowym i jedynym końcowym jest S.

G_3:



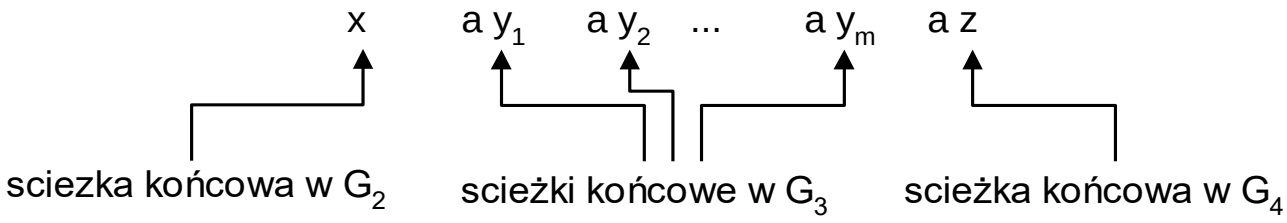
W G_3 wierzchołkiem początkowym jest A zaś jedynym końcowym jest S

G_4:



$$L(G) = L(G_1) \cup L(G_2)\,[\,aL(G_3)]^*aL(G_4)$$

Każda ścieżka końcowa w *G* jest albo końcowa w G_1, albo może być przedstawiona w postaci:



Automat skończony $\Rightarrow$ wyrażenie regularne

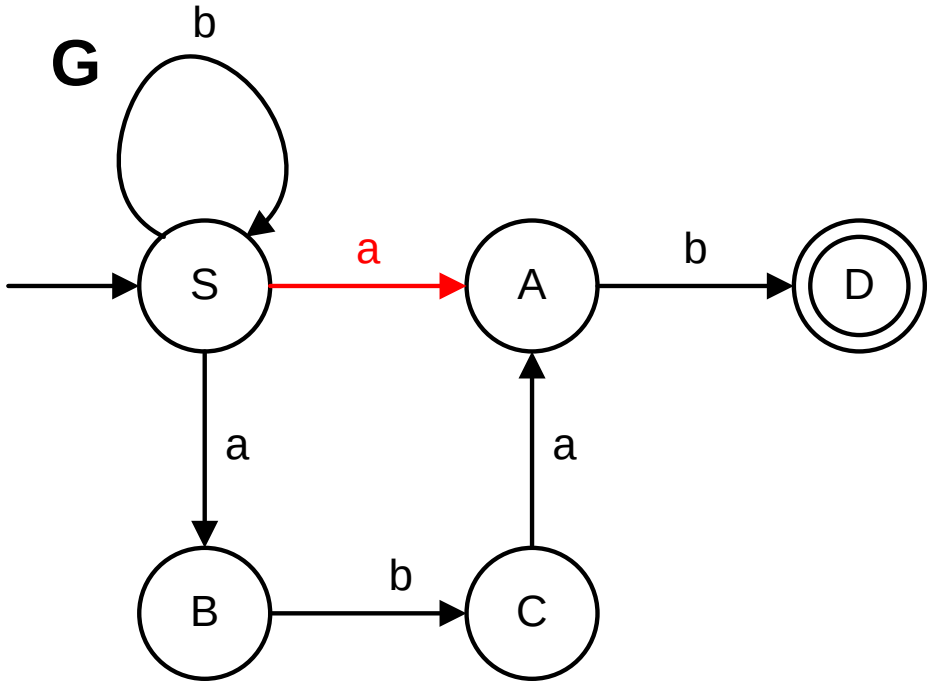
$$L(G) = L(G_1) \cup L(G_2) [\, aL(G_3)]^* aL(G_4)$$

Każda ścieżka końcowa w *G* jest powyższej postaci. Również na odwrót: każda ścieżka końcowa powyższej postaci jest ścieżką końcową w *G*.

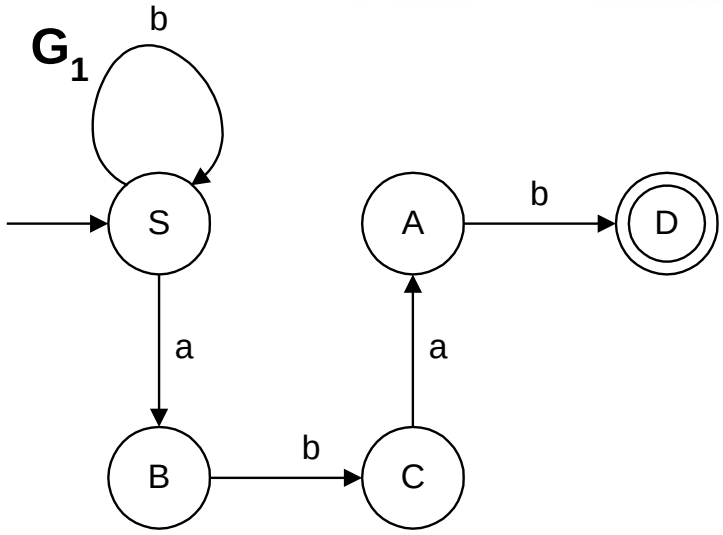
Z założenia indukcyjnego wszystkie języki występujące w prawej stronie powyższej równości (dla *k* krawędzi) są regularne oraz użyte do konstrukcji tej równości operacje są także operacjami regularnymi, a więc i język dla *G* (*k+1* krawędzi) jest także regularny, co kończy dowód twierdzenia.

Automat skończony ⇒ wyrażenie regularne Przykład (1)

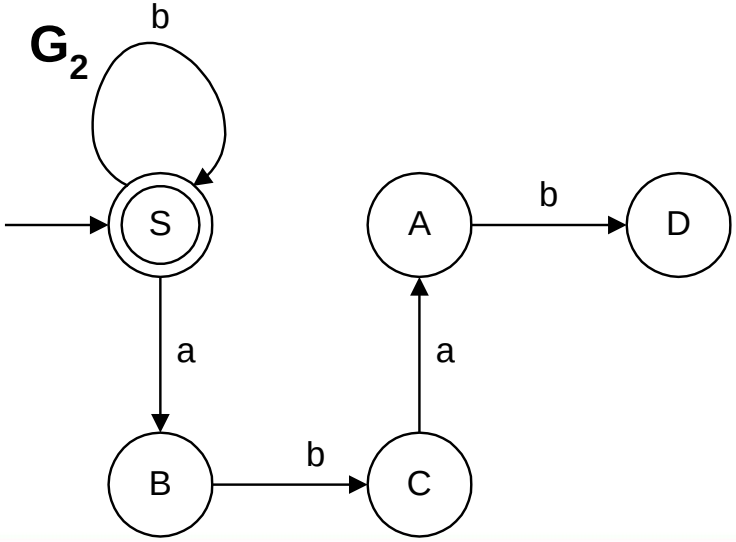
- Zbudować wyrażenie regularne opisujące język akceptowany przez automat skończony dany grafem (generowany przez gramatykę daną grafem):



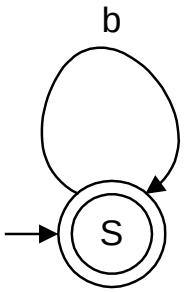
Automat skończony ⇒ wyrażenie regularne
Przykład (2)



$L(G_1) = \{b^n abab \mid n \geq 0\}$ = **b*abab**

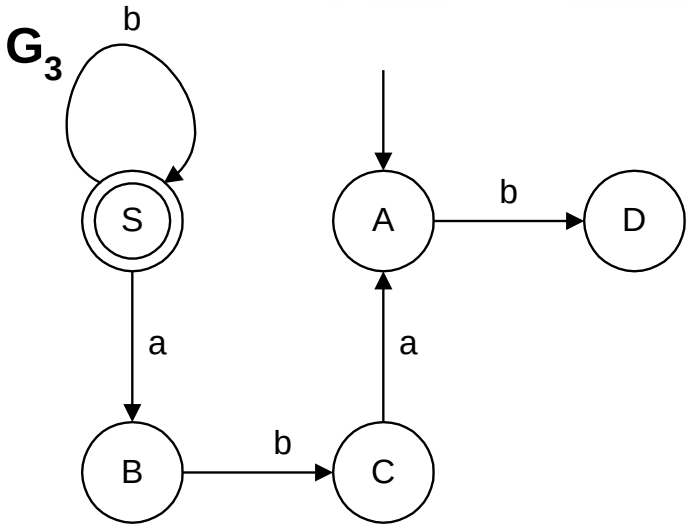


≡



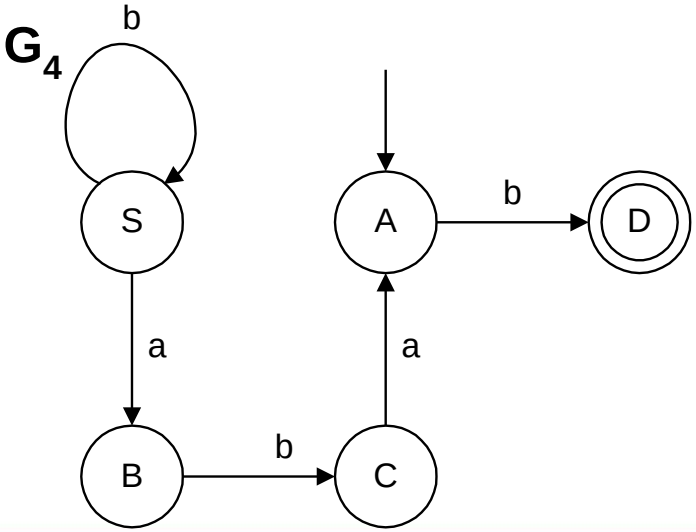
$L(G_2) = \{b^n \mid n \geq 0\}$ = **b***

Automat skończony ⇒ wyrażenie regularne
Przykład (3)



≡ A

$L(G_3) = \emptyset$



≡ A —b→ D

$L(G_4) = \{b\} = \mathbf{b}$

Automat skończony ⇒ wyrażenie regularne
Przykład (4)

$L(G_1)$= **b*abab**
$L(G_2)$ = **b***
$L(G_3)$ = ∅
$L(G_4)$ = **b**

$L(G) = L(G_1) \cup L(G_2)\,[aL(G_3)]^*aL(G_4) =$
$= \mathbf{b^*abab \mid b^*(a\emptyset)^*ab}$

b*abab | b*(a∅)*ab =
= b*abab | b* ∅*ab =
= b*abab | b*εab =
= b*abab | b*ab =
= b*ab(ab|ε)

$L(G)$ = **b*ab(ab|ε)**

Usuwanie symbolu początkowego z prawych stron produkcji

We : G = <V,Σ,P,S> - gramatyka regularna bez produkcji końcowych

Wy : G'=<V',Σ,P',S> - gramatyka regularna bez S w prawych stronach produkcji, taka że L(G') = L(G)

Metoda:

$P_1 := P;$

$V' := V;$

for $(A \rightarrow aB) \in P$ do

    if B=S then

    begin

        $V' := V' \cup \{K\};$

        $P_1 := P_1 - \{A \rightarrow aS\} \cup \{A \rightarrow aK\};$

    end;

$P' := P_1;$

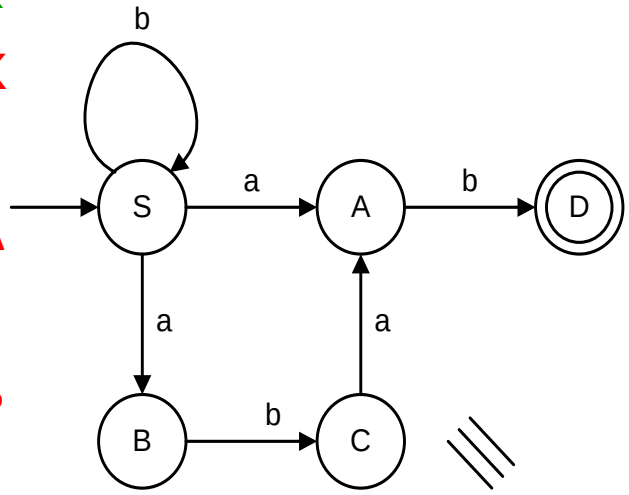
for $(A \rightarrow X) \in P_1$ and $(X = aB$ or $X = \varepsilon)$ do

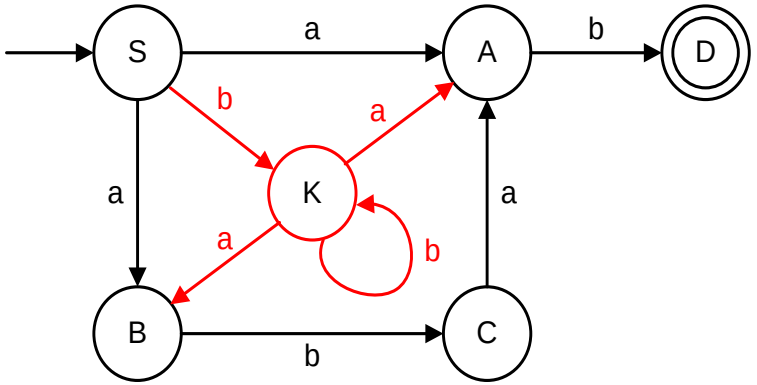
    if A=S then

        $P' := P' \cup \{K \rightarrow X\};$

Usuwanie symbolu początkowego z prawych stron produkcji

$S \rightarrow bS$	$\mathbf{S \rightarrow bK}$	$\mathbf{S \rightarrow bK}$
		$\mathbf{K \rightarrow bK}$
$S \rightarrow aA$	$S \rightarrow aA$	$S \rightarrow aA$
		$\mathbf{K \rightarrow aA}$
$S \rightarrow aB$	$S \rightarrow aB$	$S \rightarrow aB$
		$\mathbf{K \rightarrow aB}$
$B \rightarrow bC$	$B \rightarrow bC$	$B \rightarrow bC$
$C \rightarrow aA$	$C \rightarrow aA$	$C \rightarrow aA$
$A \rightarrow bD$	$A \rightarrow bD$	$A \rightarrow bD$
$D \rightarrow \varepsilon$	$D \rightarrow \varepsilon$	$D \rightarrow \varepsilon$





Konstrukcja sumy teoriomnogościowej, złożenia i domknięcia Kleene'go języków regularnych

Twierdzenie: L_1 i L_2 – języki regularne generowane przez gramatyki

$G_1 = <V_1,\Sigma_1,P_1,S_1>$

$G_2 = <V_2,\Sigma_2,P_2,S_2>$

Wówczas języki :

- $L_1 \cup L_2$
- L_1L_2
- L_1^*

są regularne.

Konstrukcję gramatyki $G = <V,\Sigma,P,S>$, takiej, że:

a) $L(G) = L_1(G_1) \cup L_2(G_2)$

b) $L(G) = L_1(G_1)\, L_2(G_2)$

c) $L(G) = [L_1(G_1)]^*$

dokonujemy przy założeniach i oznaczeniach :

- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $V_1 \cap V_2 = \varnothing$ (jeśli nie, to można nieterminale pomalować na różne kolory)
- G_1 i G_2 - gramatyki regularne bez produkcji końcowych
- F_1 i F_2 - zbiór nieterminalnych symboli końcowych gramatyk G1 i G2
- F - zbiór symboli końcowych gramatyki G

Konstrukcja sumy teoriomnogościowej, złożenia i domknięcia Kleene'go języków regularnych (a)

(a) Konstrukcja $G = \langle V_1 \cup V_2 \cup \{S\}, \Sigma, P, S \rangle$ takiej, że $L(G) = L_1(G_1) \cup L_2(G_2)$

if $\varepsilon \notin L_1 \cup L_2$ then $F := F_1 \cup F_2$

else $F := F_1 \cup F_2 \cup \{S\}$;

$P := \emptyset$;

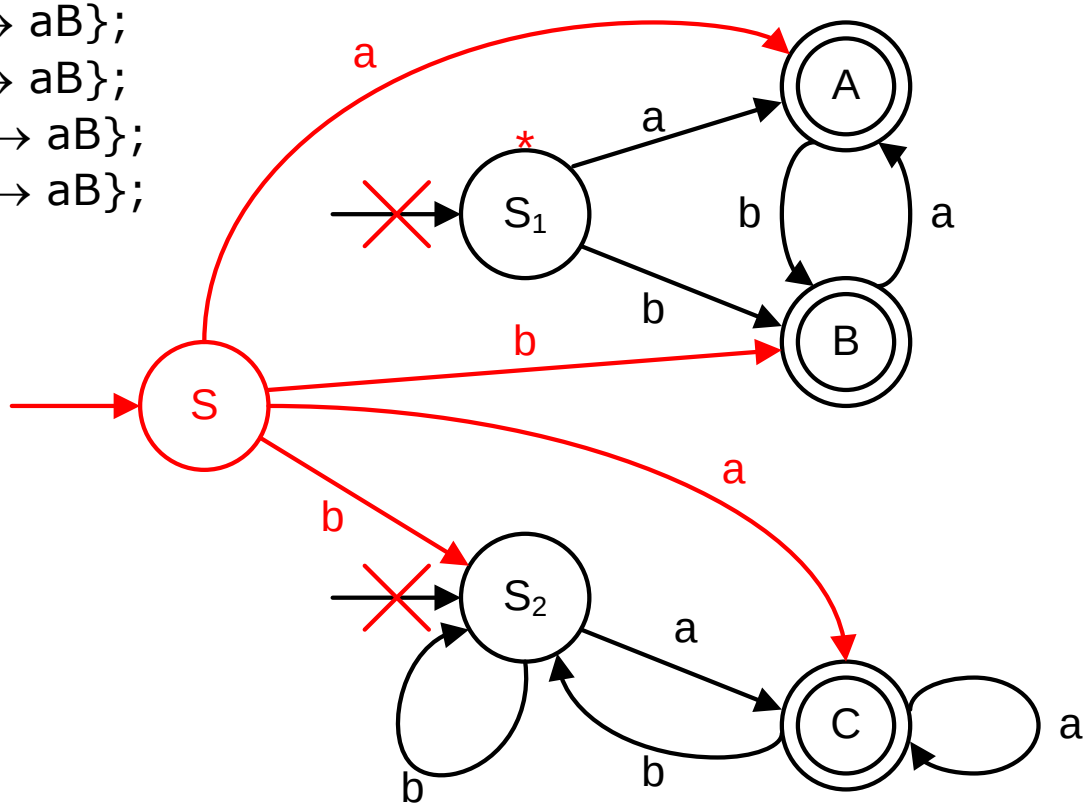
for $(A \to aB) \in P_1$ do $P := P \cup \{A \to aB\}$;

for $(A \to aB) \in P_2$ do $P := P \cup \{A \to aB\}$;

for $(S_1 \to aB) \in P_1$ do $P := P \cup \{S \to aB\}$;

for $(S_2 \to aB) \in P_2$ do $P := P \cup \{S \to aB\}$;

for $A \in F$ do $P := P \cup \{A \to \varepsilon\}$;



*S_1 stał się nieosiągalny

Konstrukcja sumy teoriomnogościowej, złożenia i domknięcia Kleene'go języków regularnych (b)

(b) Konstrukcja $G = < V_1 \cup V_2, \Sigma, P, S_1 >$ takiej, że $L(G) = L_1(G_1)L_2(G_2)$

if $S_2 \notin F_2$ then $F := F_2$ else $F := F_1 \cup F_2$;

$P := \emptyset$;

for $(A \to aB) \in P_1$ and $A \in V_1 - F_1$ do

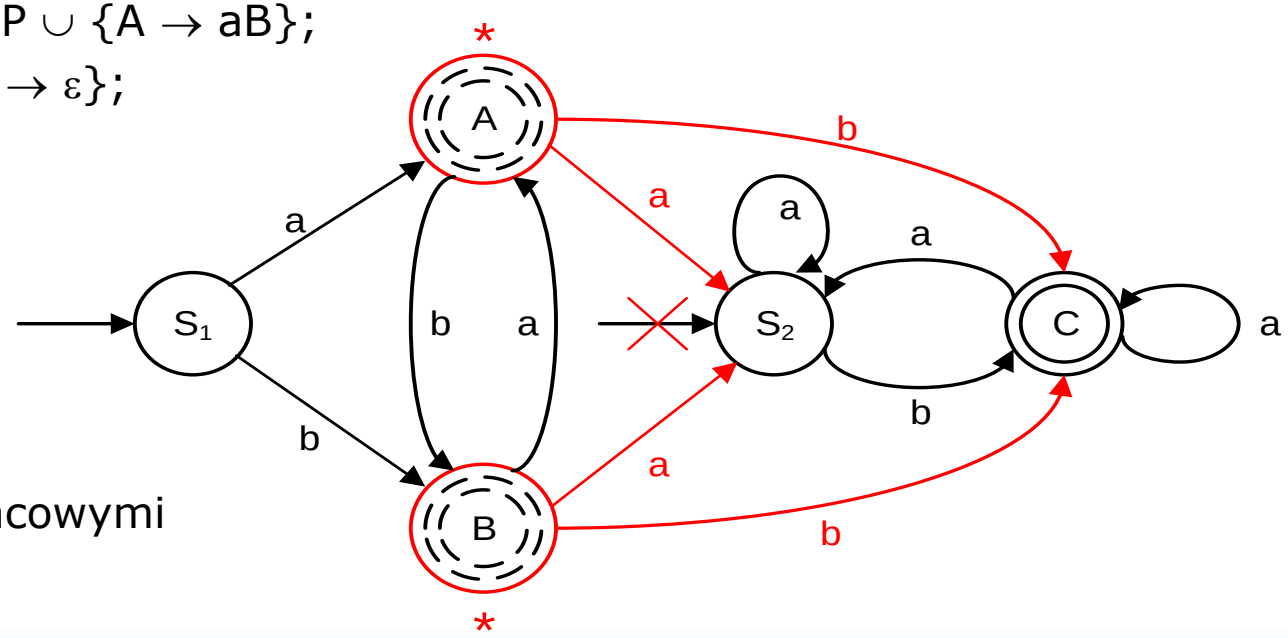
$P := P \cup \{A \to aB\}$;

for $(A \to aB) \in P_1$ and $A \in F_1$ do

$P := P \cup \{A \to aB\} \cup \{A \to bC \mid (S_2 \to bC) \in P_2\}$;

for $(A \to aB) \in P_2$ do $P := P \cup \{A \to aB\}$;

for $A \in F$ do $P := P \cup \{A \to \varepsilon\}$;



*A oraz B przestają być końcowymi

Konstrukcja sumy teoriomnogościowej, złożenia i domknięcia Kleene'go języków regularnych (c)

(c) Konstrukcja $G = \langle V_1 \cup \{S\}, \Sigma, P, S \rangle$ takiej , że : $L(G) = [L_1(G_1)]^*$

$F := F_1 \cup \{S\};$

$P := \varnothing;$

for $a \in \Sigma$ and $A \in V_1 - F_1$ do

$P := P \cup \{A \to aB \mid (A \to aB) \in P_1\};$

for $a \in \Sigma$ and $A \in F_1$ do

begin

$P := P \cup \{A \to aB \mid (A \to aB) \in P_1\};$

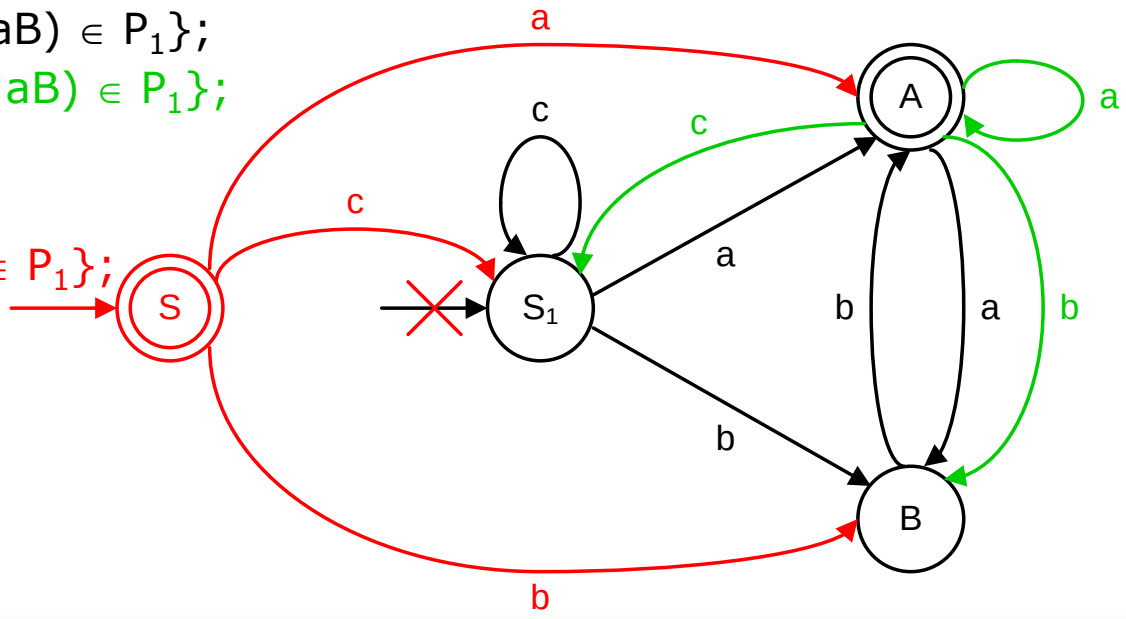
$P := P \cup \{A \to aB \mid (S_1 \to aB) \in P_1\};$

end;

for $a \in \Sigma$ do

$P := P \cup \{S_1 \to aB \mid (S \to aB) \in P_1\};$

for $A \in F$ do $P := P \cup \{A \to \varepsilon\};$



Przekształcanie wyrażenia regularnego na automat skończony

Rozważa się przekształcenie wyrażenia regularnego w automat skończony.

- Przekształcenie wyrażenia regularnego o długości *n* na automat skończony (niedeterministyczny z ε-przejściami) wymaga efektywnego zbudowania drzewa rozbioru syntaktycznego tego wyrażenia, co wymaga czasu *O(n)*.
- Mając już to drzewo rozbioru trzeba, analizując jego wierzchołki w technologii bottom-up, budować cząstkowe automaty niedeterministyczne dla każdego z nich, scalając je na bieżąco, co przy starannym zorganizowaniu procesu budowy wymaga łącznego czasu *O(n)*.
- Sumaryczny czas konstrukcji niedeterministycznego automatu skończonego z wyrażenia regularnego jest liniową funkcją długości wyrażenia regularnego.
- Dalsze ewentualne przekształcenie automatu niedeterministycznego w automat deterministyczny może wymagać czasu wykładniczego rzędu $O(s^3 2^s)$, gdzie *s* jest liczbą stanów automatu niedeterministycznego.

Przekształcanie niedeterministycznego automatu skończonego w automat deterministyczny

Rozważa się przekształcanie niedeterministycznego automatu skończonego (być może z ε-przejściami) i z liczbą stanów równą n w automat deterministyczny.

- Przy przekształcaniu niedeterministycznego automatu skończonego z ε-przejściami i z liczbą stanów równą n w automat deterministyczny, wymagane jest obliczenie ε-domknięć, co zabiera $O(n^3)$ czasu.
- W dalszej kolejności (wykonywana metodą podzbiorów stanów automatu niedeterministycznego) konstrukcja przejść z każdego pojedynczego stanu automatu deterministycznego (na podstawie posiadanych już ε-domknięć oraz tablicy przejść automatu niedeterministycznego) może być wykonana w czasie $O(n^3)$.
- Dominującym elementem w całym postępowaniu jest w zasadzie liczba stanów automatu deterministycznego, która może być rzędu 2^n (gdzie n – liczba stanów automatu niedeterministycznego).
- Wobec tego **czas wykonania przekształcenia automatu niedeterministycznego na automat deterministyczny w niekorzystnym przypadku jest rzędu $O(n^3 2^n)$.**
- W praktyce zdarza się często, że **gdy liczba utworzonych stanów automatu deterministycznego jest znacznie mniejsza od 2^n, (np. w zadaniach rozpoznawania słów kluczowych w tekście jest ona rzędu $O(n)$),** wówczas można uznać, że **czas wykonania przekształcenia automatu niedeterministycznego na deterministyczny jest rzędu $O(n^3 s)$, gdzie s jest liczbą stanów, jakie faktycznie ma automat deterministyczny**.

Przekształcanie automatu skończonego na wyrażenie regularne

Rozważa się przekształcenie automatu skończonego o n stanach na wyrażenie regularne.

- Niech n będzie liczbą stanów automatu skończonego (deterministycznego, niedeterministycznego lub niedeterministycznego z ε-przejściami). Przekształcenie tego automatu na wyrażenie regularne może zająć $O(n^3 4^n)$ czasu.
- Przekształcenie niedeterministycznego automatu skończonego (być może z ε-przejściami) z liczbą stanów równą n w automat deterministyczny zabiera w najgorszym wypadku $O(n^3 2^n)$ czasu.
- Jeśli byśmy najpierw przekształcili automat niedeterministyczny (być może z ε-przejściami) na automat deterministyczny, a później przekształcilibyśmy ten automat deterministyczny na wyrażenie regularne, to mogłoby to w niekorzystnym przypadku zabrać $O(n^3 4^{n^3 2^n})$ czasu, co jest wielkością podwójnie wykładniczą.

Testowanie przynależności słowa do języka regularnego

Rozważamy testowanie przynależności słowa o długości n do języka regularnego L.

- Jeżeli język regularny L jest reprezentowany przez deterministyczny automat skończony, a słowo testowane w ma długość $|w|=n$, zaś automat jest reprezentowany przez dwuwymiarową macierz będącą tablicą przejść, to czas potrzebny na odpowiedź: czy w należy do L – jest rzędu $O(n)$.
- Jeśli język L jest reprezentowany przez niedeterministyczny automat skończony, to konwersja automatu niedeterministycznego na deterministyczny mogłaby wymagać czasu wykładniczego względem liczby stanów automatu niedeterministycznego, choć czas samego testu byłby liniowy ze względu na długość słowa w.
- **Jeśli język L jest reprezentowany przez niedeterministyczny automat skończony o s stanach, a słowo testowane w ma długość $|w|=n$, to nie wykonując konwersji na automat deterministyczny, można przeprowadzić proces testowania w czasie $O(ns^2)$.**
- Jeśli reprezentacją L jest wyrażenie regularne o wielkości s, to można przeprowadzić konwersję do niedeterministycznego automatu skończonego w czasie $O(s)$ i następnie przeprowadzić proces testowania, co zabiera $O(ns^2)$ czasu na wejściu w o długości n.

Testowanie pustości języka regularnego

Rozważamy testowanie pustości języka regularnego.

- Testowanie, czy język regularny reprezentowany przez automat skończony (deterministyczny lub niedeterministyczny) jest pusty, polegające na zbadaniu, czy ze stanu początkowego osiągalny jest jakikolwiek stan akceptujący, wymaga czasu rzędu $O(n^2)$, gdzie n jest liczbą stanów automatu.
- Testowanie, czy język regularny reprezentowany przez wyrażenie regularne o wielkości n jest pusty, może polegać na przekształceniu tego wyrażenia w automat niedeterministyczny w czasie $O(n)$ (automat ten ma co najwyżej $O(n)$ stanów) i dalszym zbadaniu, czy ze stanu początkowego utworzonego automatu osiągalny jest jakikolwiek stan akceptujący, co wymaga czasu rzędu $O(n^2)$. Całe postępowanie zajmuje więc $O(n^2)$ czasu.
- Testowanie, czy język regularny reprezentowany przez wyrażenie regularne o wielkości n jest pusty bez konwersji na automat skończony, wymaga efektywnego zbudowania drzewa rozbioru syntaktycznego tego wyrażenia i badania podwyrażeń odpowiadających poszczególnym wierzchołkom tego drzewa, co wymaga czasu $O(n)$.