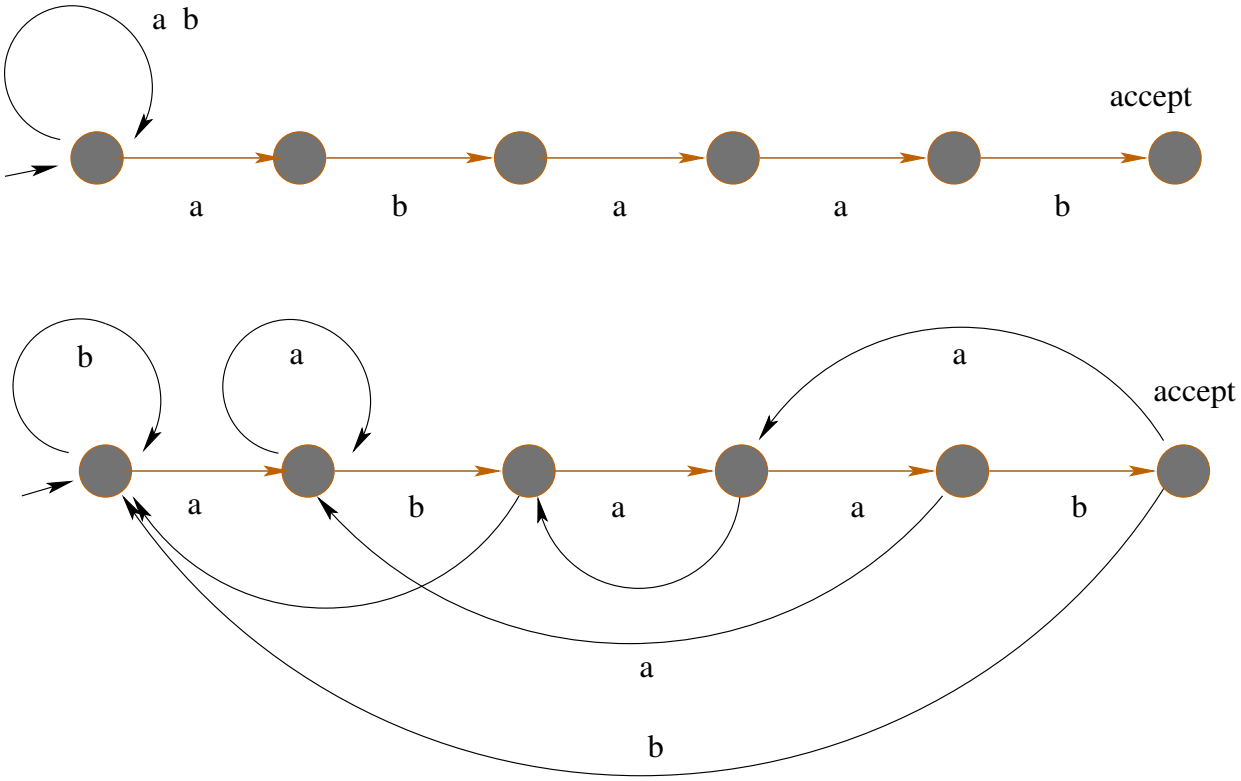
JAO – Wyklad 5
automaty dla problemow typu "string-matching"

Automat dla problemu string-matching'u - jeden wzorzec

Dane jest słowo x, zwane *wzorcem*, konstruujemy automat A akceptujacy język $L = L(A) = \Sigma^* \cdot x$. Potęgowa determinizacja *naiwnego* automatu niedeterministycznego daje minimalny automat deterministyczny dla L (wynika to z tw. Brzozowskiego).



W automacie determnistycznym dla jednego wzorca jeśli usuniemy *nieistotne* tranzycje prowadzące do stanu 0 to mamy co najwyżej 2n-1 tranzycji. Rozmiar opisu automatu jest niezależny od wielkości alfabetu.

Załóżmy, że dla każdego stanu automatu trzymamy istotne tranzycje w liście posortowanej malejąco względem "długości" stanu (jako prefiks) do którego tranzycja prowadzi. Mając taki automat możemy wykonywać string-matching brutalnie, w danym momencie szukamy tranzycji dla danej litery przeszukując listę liniową. Daje to liniowy, niezależny od alfabetu, algorytm dla string-matchingu.

Korzystamy z tablicy P prefikso-sufiksów dla słowa x. $P[i]$ jest maksymalną długością właściwego sufiksu słowa $x[1..i]$ będącego prefiksem x. Tablicę P liczymy w czasie liniowym.

Zbiorem stanów jest $Q = \{0, 1, \ldots n\}$, stany odpowiadaja prefiskom.

Algorytm

$\forall\ a \in \Sigma$ **do** $\delta(0, a) := 0$

for $i := 0$ **to** n **do**

if $i < n$ **then** $\delta(i, x_{i+1}) := i + 1$;

$\forall\ a \in \Sigma$ **do** $\delta(i + 1, a) := \delta(P[i + 1], a)$

W metodzie 2 nie korzystamy z tablicy prefikso-sufiksów.

Algorytm Konstrukcja *przez pączkowanie*

$\forall\ a \in \Sigma$ **do** $\delta(0, a) := 0$

for $i := 0$ **to** n **do**

$t := \delta(i, x_{i+1})$;

if $i < n$ **then** $\delta(i, x_{i+1}) := i + 1$;

$\forall\ a \in \Sigma$ **do** $\delta(i + 1, a) := \delta(t, a)$

W przypadku wielu wzorców konstruujemy automat dla języka $L = \Sigma^* Y$, gdzie Y jest (skończonym) zbiorem wzorców o łącznej sumie długości równej n (rozmiar wejscia do problemu).
Automat dla wielu wzorcw można skonstruować uogólniając metodę 1 lub 2 dla jednego wzorca. Zastosujemy jedynie metode drugą.

Stany odpowiadają prefiksom wzorców, "szkielet" automatu jest drzewem prefiksów.

Przejście (tranzycja) jest w szkielecie od prefiksu u do ua, jeśli oba są prefiksami pewnego wzorca, zapisujemy to jako $\delta'(i, a) = j$ gdy i, j odpowiadaja pewnym prefiksom u, ua. Szkielet (drzewo prefisków) jest podstawą do konstrukcji automatu.

Konstrukcja automatu jest realizowane poprzez przechodzenie drzewa prefiksów w kolejności BFS i aktualizowanie funkcji przejścia δ. Funkcja δ' reprezentuje drzewo prefiksów, określa przejścia od węzła do jego nastepnika w drzewie. Początkowo inicjalizujemy automat dla jednego pustego wzorca (zapętlamy korzeń drzewa).

Algorytm

$\forall\ a \in \Sigma$ **do** $\delta(root, a) := root$

for all nodes $u \neq root$ **w kolejności BFS** u **do**

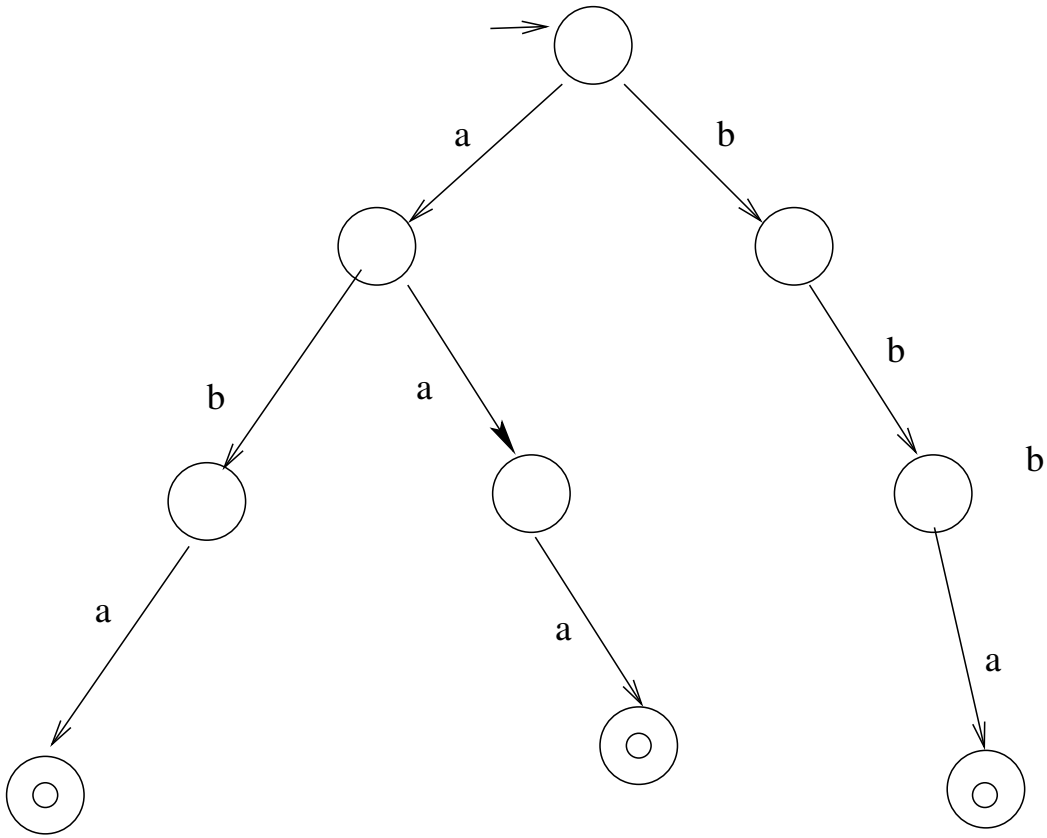
for all edges $u \xrightarrow{a} v$ **należących do szkieletu do**

$t := \delta(u, a)$; $\delta(u, a) := v$;

if $\delta'(t, s) \neq nil$ **then** $\delta(v, s) := \delta'(t, s)$

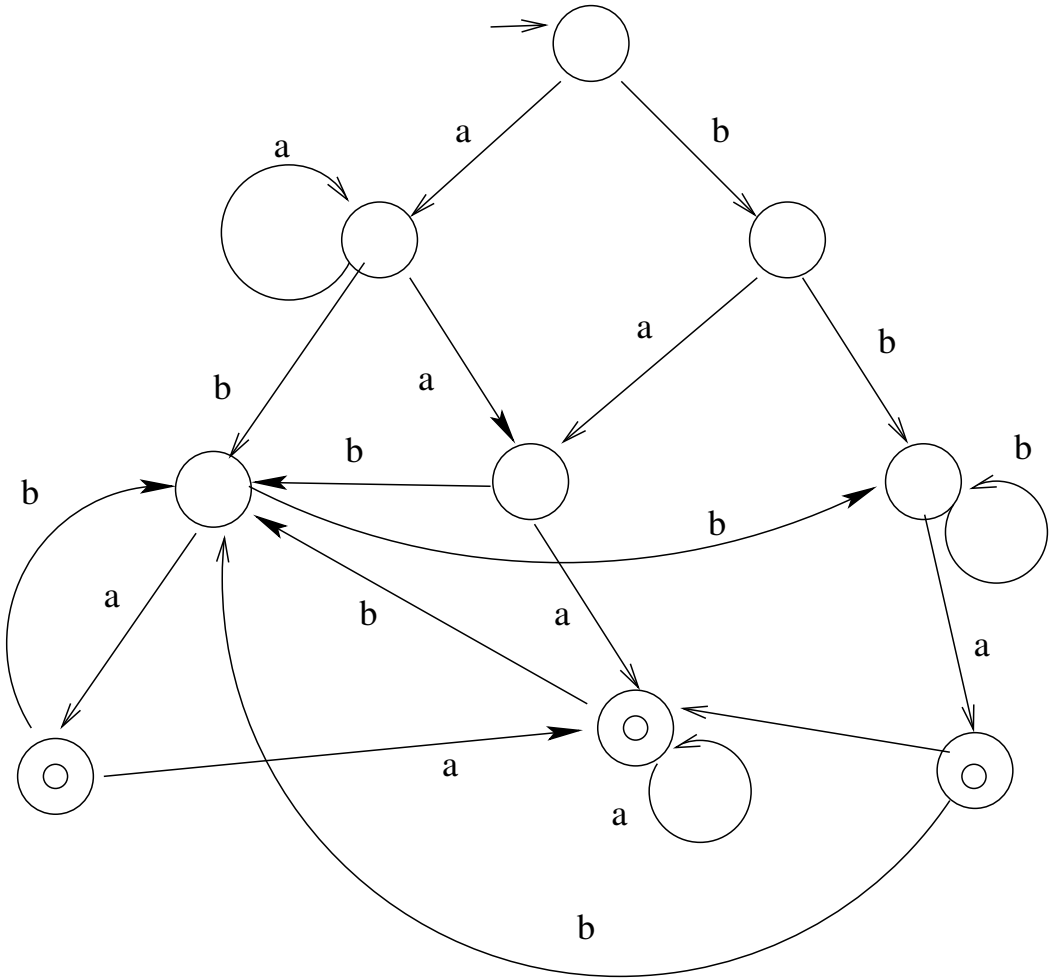
else $\delta(v, s) := \delta(t, s)$

Szkielet automatu dla zbioru wzorców $\{aba, aaa, bba\}$



Szkielet automatu, drzewo prefksów wzorców opisane funkcją δ' (przejście od węzła do jego następników).

Automat dla zbioru wzorców $\{aba, aaa, bba\}$



Dla automatu dla wielu wzorców zbiór tranzycji prowadzących do stanów różnych od korzenia istotnie zależy od rozmiaru alfabetu

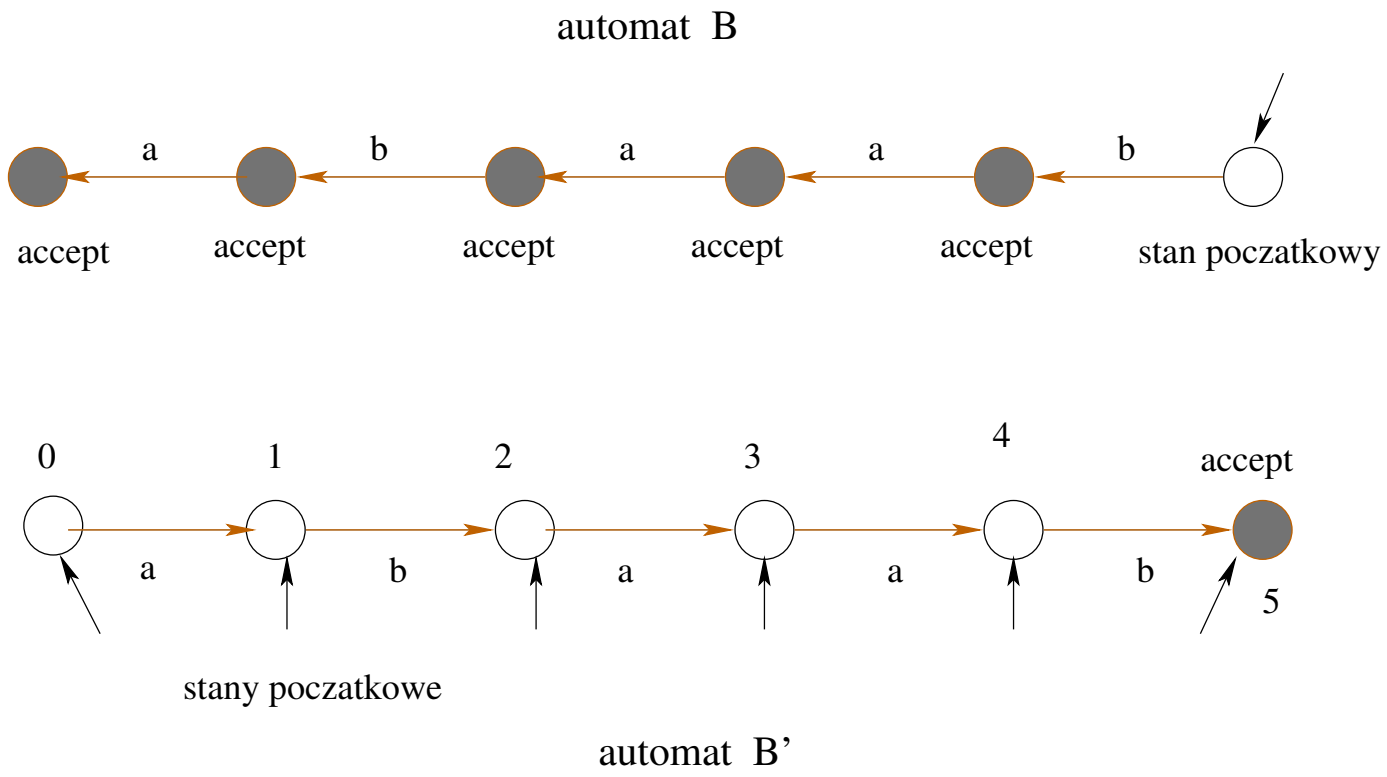
Automat sufiksowy akceptuje zbiór $SUF(x)$ wszystkich sufiksów zadanego słowa x długości n.

Minimalny automat ma $O(n)$ stanów i $O(n)$ istotnych tranzycji (nie prowadzących do stanu typu *śmietnik*).

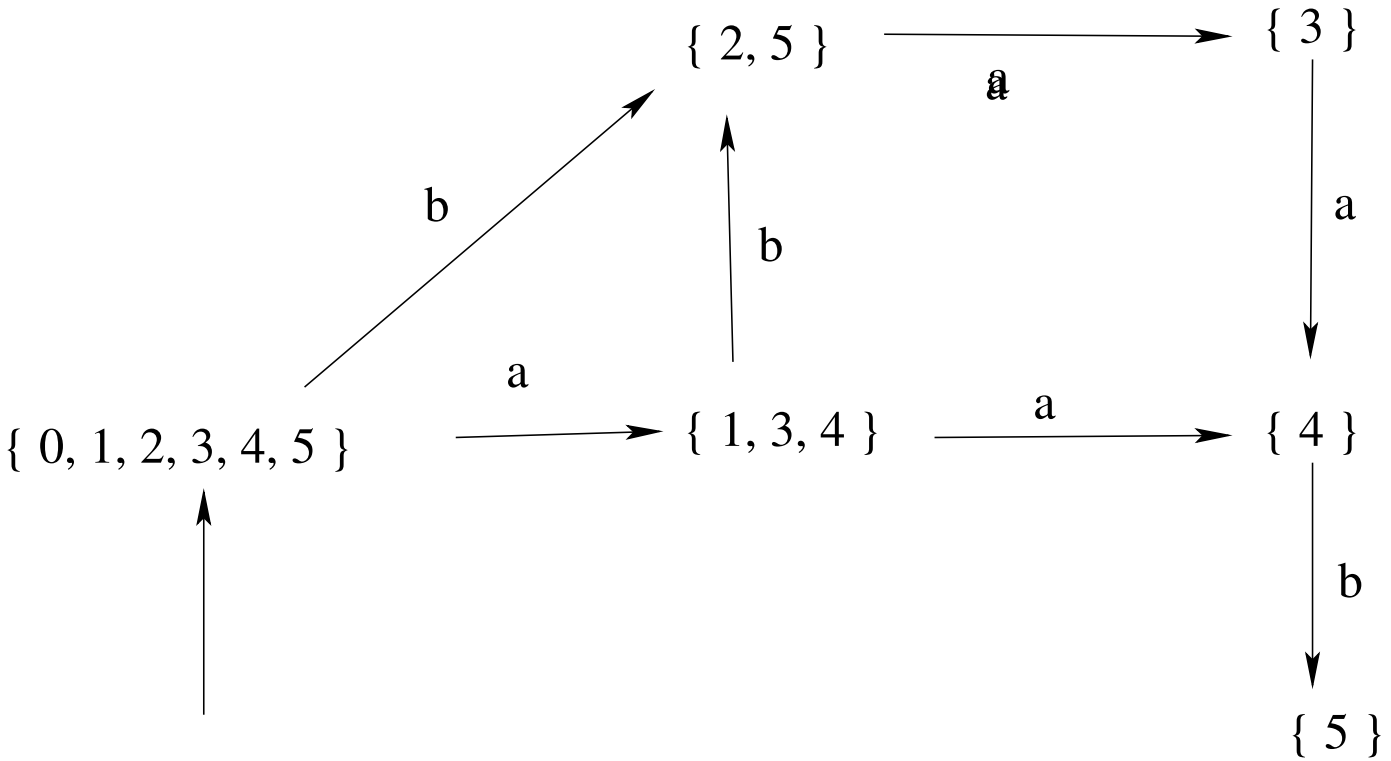
Minimalny automat sufiksowy powstaje przez determinizacje automatu B', gdzie B' jest odwróceniem deterministycznego automatu B akceptującego $PREF(x^R)$ (zbiór prefiksów x^R). Zauważmy, że

$$SUF(x) = (PREF(x)^R.$$

Automat B dla $PREF(x)$ oraz niedeterministyczny $B' = B^R$ dla $SUF(x)$



Automat sufiksowy dla $x = abaab$, determinizacja automatu B'



Stany akceptujące zawierają 5.

Konstrukcja automatu sufiksowego A dla słowa x.

Dla $w \in SUB(x)$ definiujemy:

$$EndPos(w) = \text{zbiór końcowych pozycji wystąpień } w.$$

Konstrukcja automatu A:

- Stanami są zbiory $EndPos(w)$, gdzie $w \in SUB(x)$
- $\delta(EndPos(w), a) = EndPos(wa)$.
- Stan początkowy to $EndPos(\epsilon) = \{1..n\}$.
- Stany akceptujące są postaci $EndPos(w)$, gdzie w jest dowolnym sufiksem tekstu.

Rodzina $\mathcal{F}$ podzbiorów postaci $EndPos(w)$ dla danego słowa x ma następującą własność dla każdych $X, Y \in \mathcal{F}$:

$$X \cap Y = \emptyset \text{ or } X \subseteq Y \text{ or } Y \subseteq X.$$

Lemat

Jeśli rodzina składająca się z pewnych podzbiorów zbioru n-elementowego ma powyższą własność to składa się z co najwyżej 2n podzbiorów.

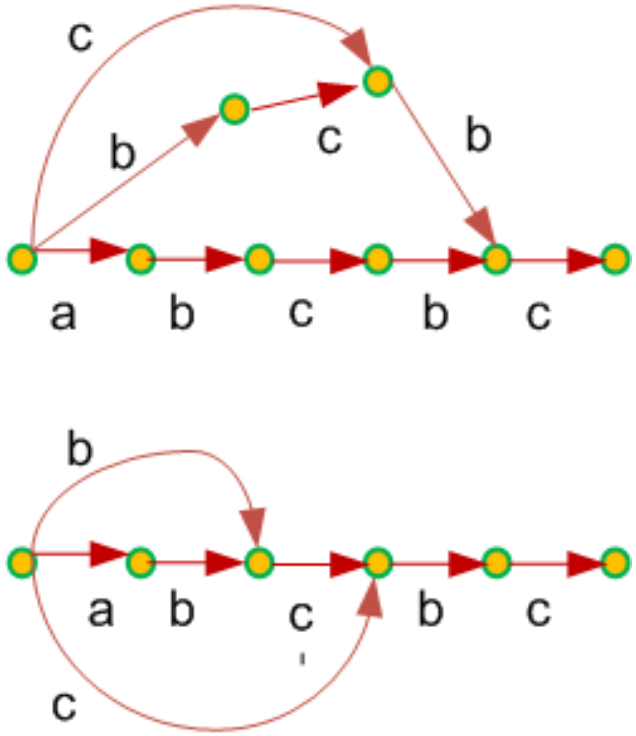
Twierdzenie

Automat sufiksowy ma co najwyżej 2n stanów oraz co najwyżej 3n tranzycji, które nie prowadzą do stanu śmietnik.

Zadanie. Ile jest tranzycji dla słowa postaci $ab^{n-2}c$.

Automat którego stanami są zbiory $EndPos(w)$ ma minimalną liczbę stanów dla języka wszystkich sufiksów. Przykładem słowa gdy minimalny automat sufiksowy nie jest minimalnym automatem akceptującym wszystkie podsłowa (stanami akceptującymi są teraz wszystkie stany automatu sufiksowego) jest tekst $abcbc$.
Automat minimalny dla wszystkich podsłów można otrzymać z A poprzez minimalizację (po zakwalifikowaniu wszystkich stanów jako akceptujące). Algorytm minimalizacji dla automatu acyklicznego (pomijając stan śmietnik) działa w czasie liniowym.

Minimalny automat sufiksowy i podsłów dla $x = abcbc$.



Automat sufiksowy ma 2 stany akceptujące, w automacie podsłów wszystkie stany są akceptujące.