JAO - Wykład 4
minimalizacja, równoważność i inne problemy decyzyjne dla deterministycznych automatów skończonych

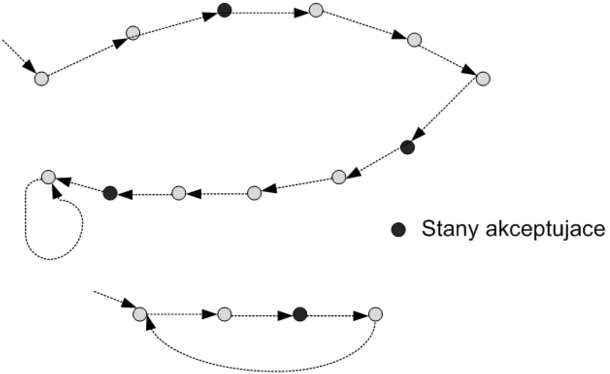
Dla języka L oznaczmy $L_k = L \cap \Sigma^{\leq k}$. Niech $L(A, q)$ będzie językiem akceptowanym przez automat A w którym stan początkowy zamieniliśmy na q.

Dwa stany p, q są z definicji k-równoważne, gdy są równoważne z dokładnością do słów długości co najwyżej k, czyli jeśli $L_k(A, p) = L_k(A, q)$. Oznaczamy przez R_k relację k-równoważności.

Fakt. $R_i = R_{i+1}$ implikuje $R_i = R_{i+2}$ oraz R_i jest końcową relacją równoważności stanów.

Wniosek. Minimalne słowo rozróżniajace dwa automaty o liczbie stanów n, m ma długość co najwyżej $n + m - 2$.

Minimalne słowo rozróżniajace może być długie



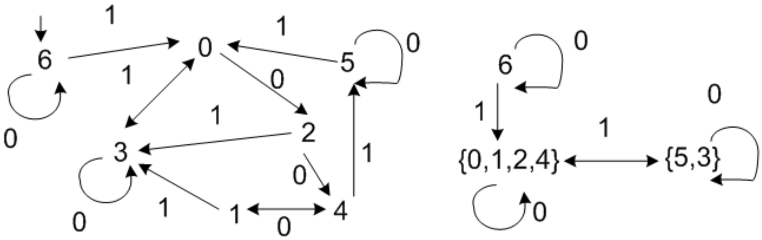
Najkrótsze słowo które rozróżnia te dwa automaty ma długość dokładnie $n + m - 2$

Podejście teoriografowe:
tworzymy graf G:

- węzły to pary stanów,
- tworzymy krawędzie $(s, s') \to (\delta(s.a), \delta(s', a))$ dla każdej pary stanów s, s' i symbolu $a \in \Sigma$.

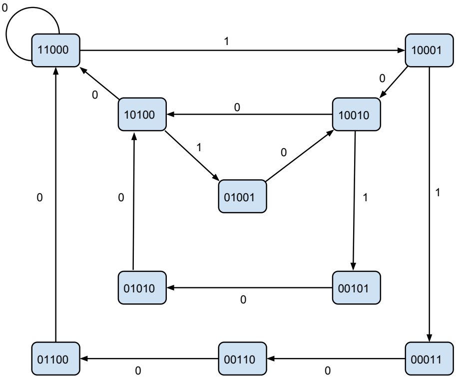
Stany s, s' nie są równoważne, wtedy i tylko wtedy gdy istnieje w G ścieżka od (s, s') do $F \times (Q - F)$. Daje to algorytm $O(n^2)$ na minimalizację automatu.

Sklejamy klasy równoważnych stanów, w rezultacie otrzymujemy automat minimalny (złożony ze stanów osiągalnych ze stanu początkowego).



Niech stanami akceptującymi bedą 3, 5. Po minimalizacja stany 0, 1, 2, 4 skleją sie w jeden stan, podobnie stany 3, 5. Nowy stan akcptujący to zbiór {3, 5} sklejony do jednego stanu.

Niech $L(m,k)$ bedzie zbiorem słów zerojedynkowych, takich że w każdym podsłowie długości m jest co najwyżej k jedynek. Minimalny automat dla $L(5,2)$ jest podany poniżej, bez stanu "śmietnik", $q_0 = 11000$. Ogólnie minimalny automat deterministyczny ma $\binom{m}{k} + 1$ wszystkich stanów.



.

Zamiast sprawdzać równoważność dwóch automatów łatwiej sprawdzać równoważność dokładnie dwóch stanów q_1, q_2 tego samego automatu (można dwa automaty *złączyć* w jeden).

Algorytm

kolejka K początkowo zawiera tylko parę (q_1, q_2);
tworzymy podział Q na bloki będące *singletonami*.
while $K \neq \emptyset$ **do**
$(p, q) := delete(K)$; $A := Find(p)$; $B := Find(q)$;
jeśli $A \neq B$ to
$\forall\, a \in \Sigma$: $Union(A, B)$; $insert((\delta(p, a),\ \delta(q, a)), K)$

Stany q_1, q_2 nie są równoważne wtw gdy jakiś blok zawiera jednocześnie stan akceptujący i nieakceptujący. Algorytm działa w czasie $O(n \cdot |\Sigma| \cdot \log^* n)$

- Problem "membership" dla standardowych wyrażeń można rozwiązać w czasie $O(n \cdot r)$ gdzie r długość wyrażenia, n długość tekstu. Algorytm poprzez symulację automatu.
- Problem "membership" dla rozszerzonych wyrażeń regularnych można rozwiązać w czasie wielomianowym metodą programowania dynamicznego.
- Inne problemy w czasie wielomianowym: czy język jest pusty, czy język jest nieskończony.
- Sprawdzanie czy $L(A) = \Sigma^*$. Wielomianowe dla automatów deterministycznych, natomiast *PSPACE*-zupełne dla niedeterministycznych.

Dla deterministycznego lub niedeterministycznego automatu B przez B^R oznaczmy automat w którym odwrócono wszystkie *strzałki* i zamieniono stany akceptujące z początkowymi. Automat B^R nazywamy *odwróceniem* automatu B.
Stan automatu jest *bezużyteczny* gdy nie jest siągalny ze stanu początkowego.

Twierdzenie

(Janusza Brzozowskiego)
Jeśli automat niedeterministyczny A jest odwróceniem pewnego deterministycznego automatu B bez bezużytecznych stanów, to deterministyczny automat potęgowy $P(A)$ jest automatem minimalnym.

Niech $\Sigma = \{a, b\}$ oraz niech:

$$L = \Sigma^n \cdot a \cdot \Sigma^*$$

Istnieje prosty automat deterministyczny A mający $n+2$ stany. Odwróceniem A^R jest automat niedeterministyczny mający też $n+2$ stany akceptujący język

$$L^R = \Sigma^* \cdot a \cdot \Sigma^n$$

Konstrukcja potęgowa daje automat warstwowy $P(A^R)$, stanami są wszystkie zbiory stanów zawierające stan początkowy A^R, wiemy że takich zbiorów jest 2^{n+1}.
Z twierdzenia Brzozowskiego wiemy że jest to konstrukcja optymalna, nie musimy dowodzić minimalności.

Dla automatu A niech M_a będzie $n \times n$ macierzą logiczną odpowiadającą przejściom za pomoca litery a. Wtedy akceptacja słowa $w = a_1 a_2 \ldots a_m$ może byc sprowadzona do mnożenia

$$M_{a_1} M_{a_2} M_{a_3} \times ..M_{a_m}.$$

Jeśli mamy algorytm mnożenia macierzy w czasie $O(n^\omega)$ to

- dla słowa w i liczby r możemy sprawdzić czy deterministyczny automat A akceptuje w^r w czasie $O(n^\omega \cdot (|w| + \log r))$.
- możemy policzyc liczbę słów długości r które akceptuje deterministyczny automat A w czasie $O(n^\omega \cdot \log r)$.
- jeśli A jest niedeterministyczny, akceptuje skończony język i $|\Sigma| = 1$ to $|L(A)|$ liczymy w czasie $O(n^\omega \cdot \log n)$.