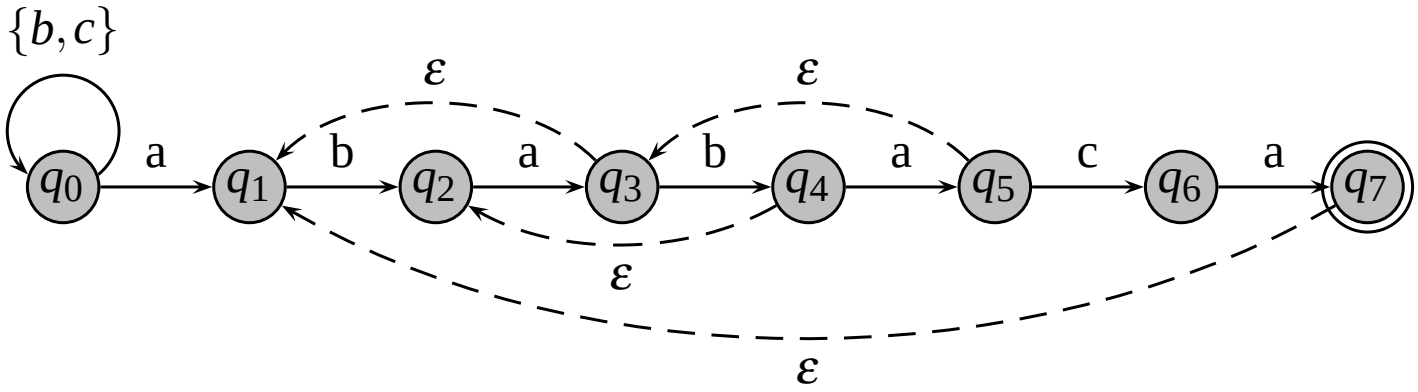
Morrisov-Prattov algoritmus 1970 $O(m+n)$



Epsilonové prechody iba ak nie je iná možnosť.

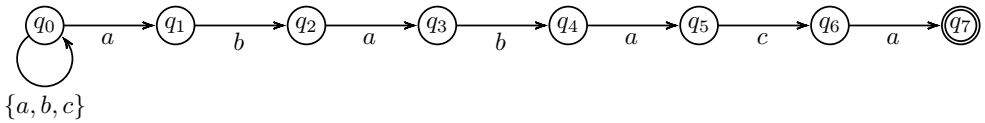
Epsilonový prechod z q_i do $q_{sp[i]}$ ($sp[i] < i$).

$sp[i] =$ dĺžka najdlhšieho vlastného sufixu $P[0..i-1]$, ktorý je prefix P.

Automat po prečítaní T skončí v stave q_i,

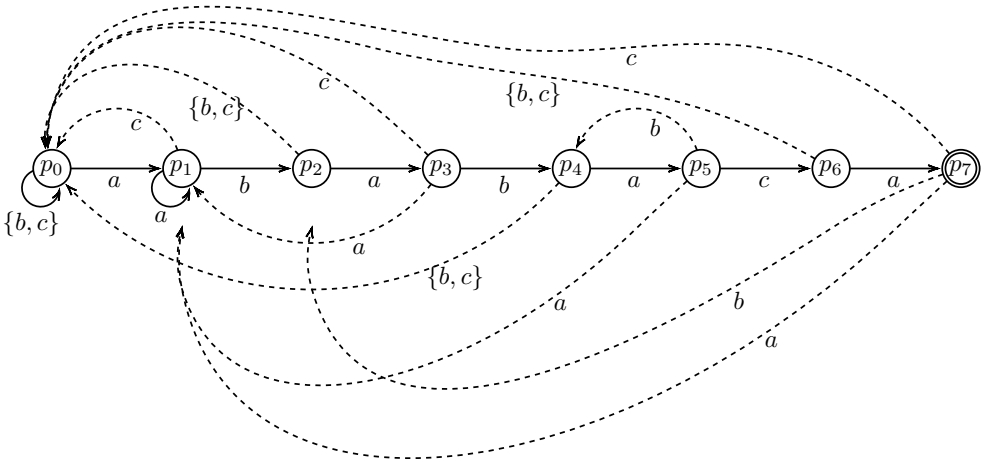
kde i je dĺžka najdlhšieho sufixu T, ktorý je prefixom P.

Nedeterministický konečný automat pre $\{xP|x \in \Sigma^*\}$



$(q_0, T) \vdash^* (q_i, \varepsilon) \iff$ sufix T dĺžky i je prefixom P.

Deterministický konečný automat pre $\{xP|x \in \Sigma^*\}$



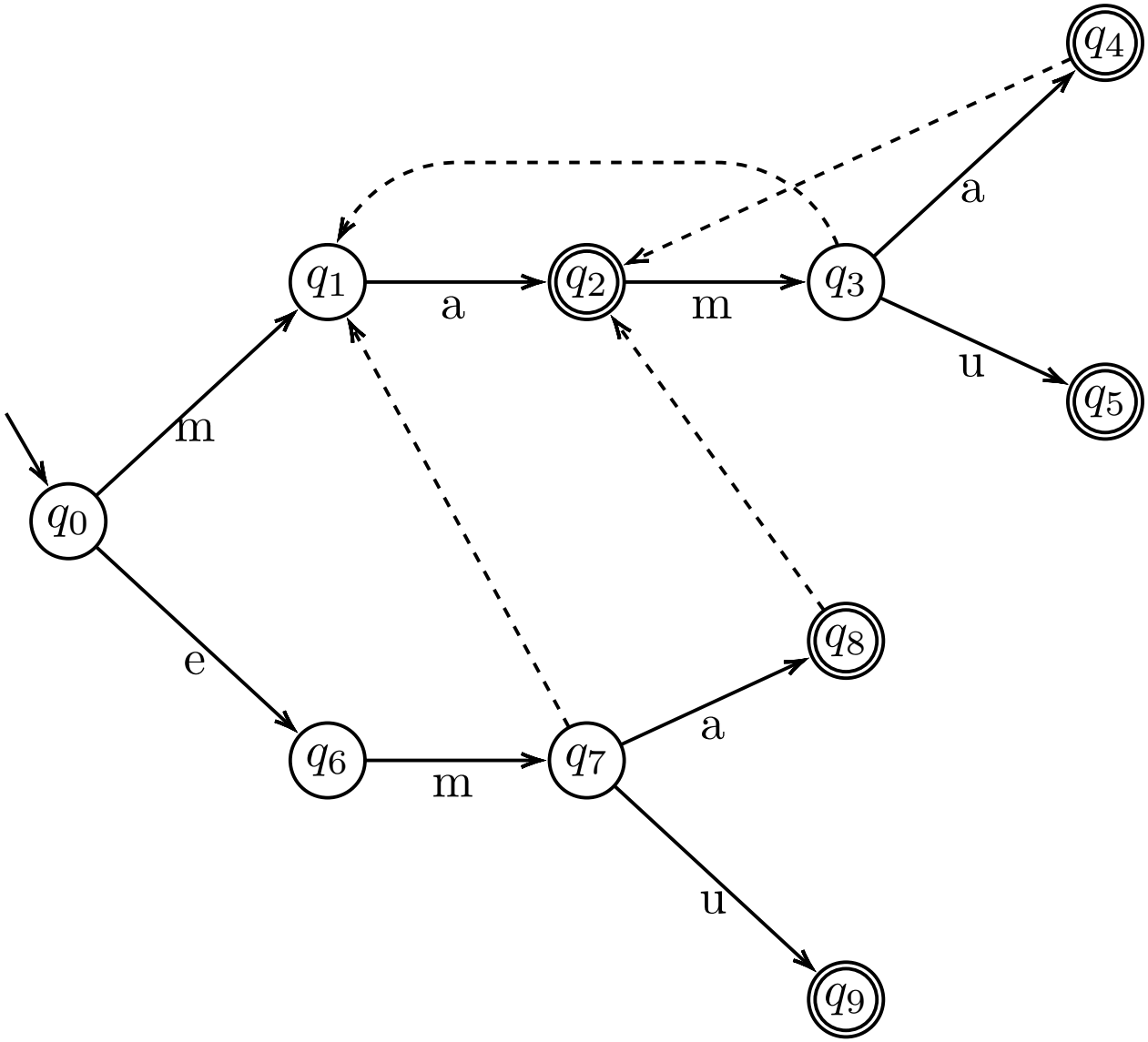
$(p_0, T) \vdash^* (p_i, \varepsilon) \iff$

i je dĺžka najdlhšieho sufixu T, ktorý je prefixom P

Hľadanie viacerých vzoriek

- Množina vzoriek $\mathcal{P} = \{P_1, P_2, \ldots, P_z\}$ a text T.
- Nech $m = \sum_{i=1}^{z} |P_i|$, $n = |T|$
- Triviálny algoritmus: spustíme KMP pre každú vzorku

Ahov-Corasickovej algoritmus 1975



Ahov-Corasickovej algoritmus

```
1  v=root;
2  for (i=0; i<n; i++) {
3    while (v!=root && v.child(T[i])==null) {
4      v=v.epsilon; // prechod po epsilonovych hranach
5    }
6    // ak sa da, ideme dopredu
7    if (v.child(T[i])!=null) {
8      v=v.child(T[i]);
9    }
10   print_occurrences(v);
11 }
```

Ahov-Corasickovej algoritmus

```
1  void print_occurrences(node v) {
2    u=v;
3    while (u!=null) {
4      if (u.id!=null) {
5        print u.id, i—|P[u.id]|+1;
6      }
7      u=u.output;
8    }
9  }
```

Výpočet epsilonových prechodov

```
1  root.epsilon=root;
2  call find_output in breadth—first order
3
4  findepsilon(v) {
5    // u.epsilon uz je zname pre vsetky slova u kratsie ako v
6    a = v.last;
7    u = v.parent.epsilon;
8    while (u.child(a)==null && u!=root) { u=u.epsilon; }
9    if (u.child(a)!=null) { u=u.child(a); }
10   v.epsilon=u;
11 }
```

Výpočet výstupných hrán

```
1  root.output=root;
2  call find_output in breadth—first order
3
4  void find_output(node v) {
5    // u.epsilon uz je zname pre vsetky slova u kratsie ako v
6    u = v.epsilon;
7    if (u.id!=null) { // ak je vzorka, ukazujeme nanho
8      v.output=u;
9    } else {          // inak jeho najblizsi vystup
10     v.output=u.output;
11   }
12 }
```

Boyerov-Moorov algoritmus 1977

```
1  i=0;
2  while(i<=n-m){
3     j=m-1;
4     while(j>=0 && P[j]==T[i+j]){
5        j--;
6     }
7     if(j<0){ print i; }
8     i += skip;
9  }
```

Ak skip = 1, triviálny algoritmus. Väčšie (ale bezpečné) skoky:

– pravidlo zlého znaku

– pravidlo dobrého sufixu

Pravidlo zlého znaku

Nech $R[x]$ je index posledného výskytu x v P

alebo -1 ak x sa nevyskytuje v P

Ak $P[j+1..m-1] = T[i+j+1..i+m-1]$ a $P[j] \neq T[i+j] = x$:

Ak $R[x] < j$, skip $= j - R[x]$

inak skip $= 1$.

Horspoolov algoritmus

Použi pravidlo zlého znaku na $x = T[i + m - 1]$

$R[x]$ je index posledného výskytu x v $P[0..m - 2]$

skip $= m - 1 - R[x]$

$O(mn + \sigma)$ najhorší prípad, $O(n/\sigma + m + \sigma)$ priemerný prípad