



Constrained Longest Common Subsequence Problem for Degenerate Strings

Michal Voráček, Ladislav Vagner
Czech Technical University in Prague



Informal problem definition

- LCS problem for Strings

$x = abcdaaa$

$y = abacaad$



Informal problem definition

- LCS problem for Strings

$x = abcd\color{red}aaa$

$y = ab\color{red}acaad$

$LCS(x,y) = ab\color{red}caa$



Informal problem definition

- CLCS problem for Strings

$x = abcdaaa$

$y = abacaad$

$z = bcd$



Informal problem definition

- CLCS problem for Strings

$x =$ *abcd**aaa*

$z =$ *bcd*

$y =$ *abacaad*

$CLCS(x,y,z) =$ *abcd*



Related work

- Tsai (2003): *The constrained longest common subsequence problem* – $O(n^2m^2r)$
- Chin et al. (2004) : *A simple algorithm for constrained sequence problems* - $O(nmr)$
- Arslan et al. (2004) : *Algorithms for the constrained longest subsequence problem* - $O(nmr)$ + approx. ver. in $O(dnmr)$



Preliminaries

Degenerate string x over an alphabet A

is a sequence of nonempty subsets of A , $x \in P(A)^+$

•

$$x = (a) \begin{pmatrix} a \\ c \\ g \end{pmatrix} (a) \begin{pmatrix} a \\ c \\ g \end{pmatrix} (c) (a) (a) (c) (a) (a) \begin{pmatrix} a \\ c \\ g \end{pmatrix} (a) \begin{pmatrix} a \\ c \\ g \end{pmatrix}$$



Preliminaries

Set of all subsequences of string x :

$$\text{Sub}(x) = \{u = u_1 u_2 \dots u_{n-1} \mid u_j = z_j, x = y_1 z_1 y_2 z_2 \dots y_{n-1} z_n y_n, \\ u \in A^*, z_j \in A, y_j \in A^*, 1 \leq j \leq n-1, 1 \leq i \leq n\}$$

$$x = [a][g][a][c][c][g][a][c][g][a][g][a][c]$$

$$u = [g][g][g][g]$$



Preliminaries

Set of all subsequences of degenerate string x :

$$Sub(x) = \{u = u_1 u_2 \dots u_{n-1} \mid u_j \in z_j, x = y_1 z_1 y_2 z_2 \dots y_{n-1} z_n y_n, \\ u \in A^*, z_j \in P(A), y_j \in P(A)^*, 1 \leq j \leq n-1, 1 \leq i \leq n \}$$

$$x = \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix} \begin{pmatrix} a \\ c \\ g \end{pmatrix}$$

$$u = \begin{pmatrix} g \end{pmatrix} \begin{pmatrix} g \end{pmatrix} \begin{pmatrix} g \end{pmatrix} \begin{pmatrix} g \end{pmatrix} \begin{pmatrix} g \end{pmatrix}$$



Preliminaries

Set of all constrained common subsequences for degenerate strings x, y and z :

$$CCSub(x, y, z) = \{ u \mid u \in CommonSub(x, y) \wedge \exists v \in z: v \in Sub(u) \}$$

Set of all constrained longest common subsequences for degenerate strings x, y and z :

$$CLCSub(x, y, z) = \{ u \mid u \in CCSub(x, y, z) \wedge \forall v \in CCSub(x, y, z) : |v| \leq |u| \}$$



Preliminaries

*The **CLCS** problem for degenerate strings x, y, z is to compute the set $CLCSub(x, y, z)$.*



Preliminaries

Maximum Length function for finite language L :

$$F_M(L) = \{ u \mid u \in L \wedge \forall v \in L : |v| \leq |u| \}$$

$$F_M(\{abc, bbc, bc, a\}) = \{abc, bbc\}$$



Preliminaries

Constrained Subsequence function

for finite language L for degenerate string z:

$$F_{CS}(L, z) = \{ u \mid u \in L \wedge \exists v \in z: v \in \text{Sub}(u) \}$$

$$F_{CS}(\{abc, aba, abd\}, a[c,a]) = \{ abc, aba \}$$



Preliminaries

Set of all common subsequences of degenerate strings x, y :

$$\text{CommonSub}(x,y) = \{ u \mid u \in \text{Sub}(x) \wedge u \in \text{Sub}(y) \}$$



Informal problem definition

- CLCS problem for Degenerate Strings

$x = [a,e]bcdaaa$

$z = b[c,d]d$

$y = [a,c]bacaa[d,e]$



Informal problem definition

- CLCS problem for Degenerate Strings

$x = [a,e]bddd\text{aaa}$

$z = b[c,d]d$

$y = [a,c]ba[c,d]aa[d,e]$

$CLCS(x,y,z) = abdd$



Algorithm outline

$$CLCSub(x, y, z) =$$

$$F_M(CCSub(x, y, z)) =$$

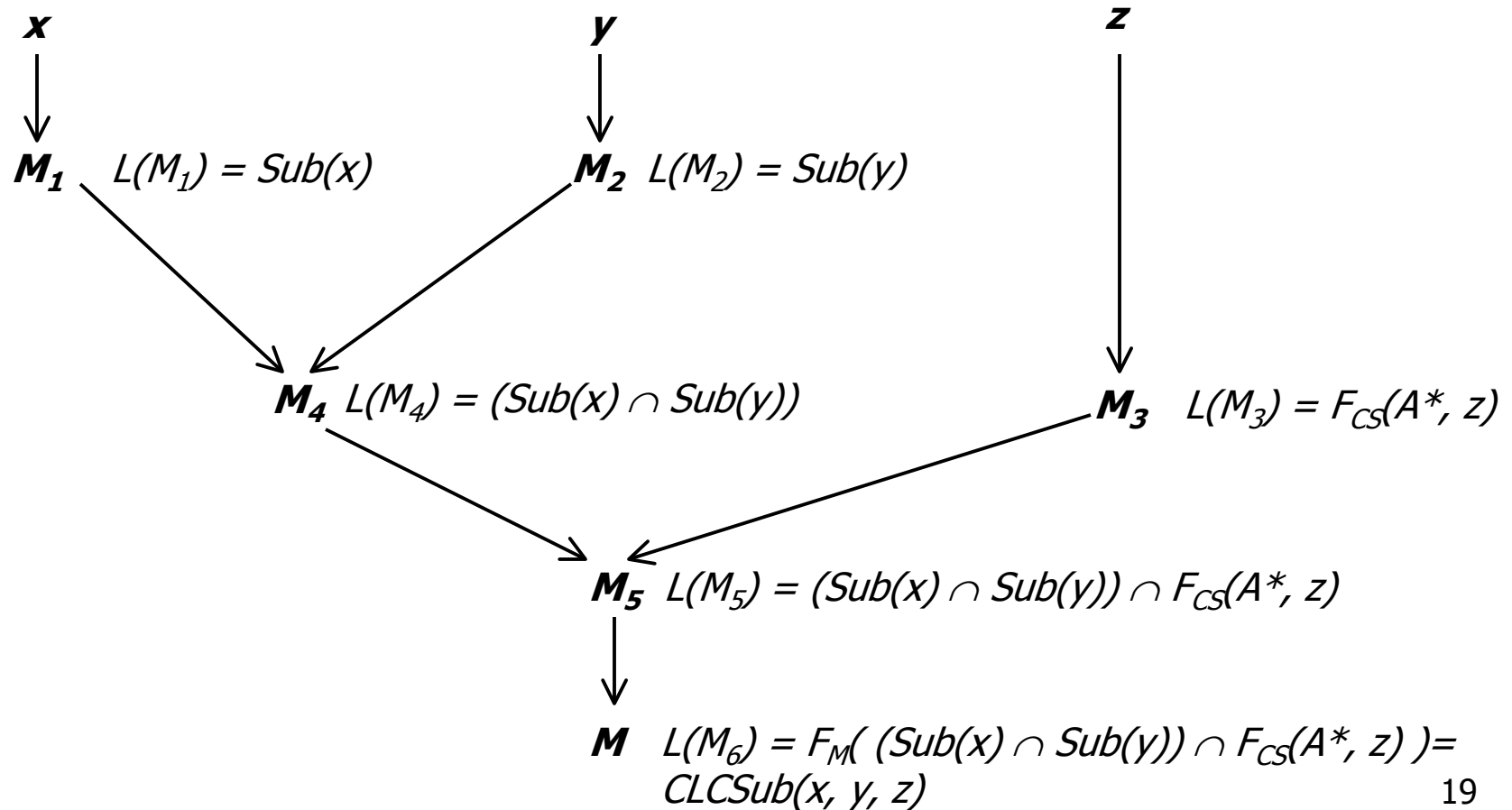
$$F_M(CommonSub(x, y) \cap F_{CS}(A^*, z)) =$$

$$F_M((Sub(x) \cap Sub(y)) \cap F_{CS}(A^*, z))$$



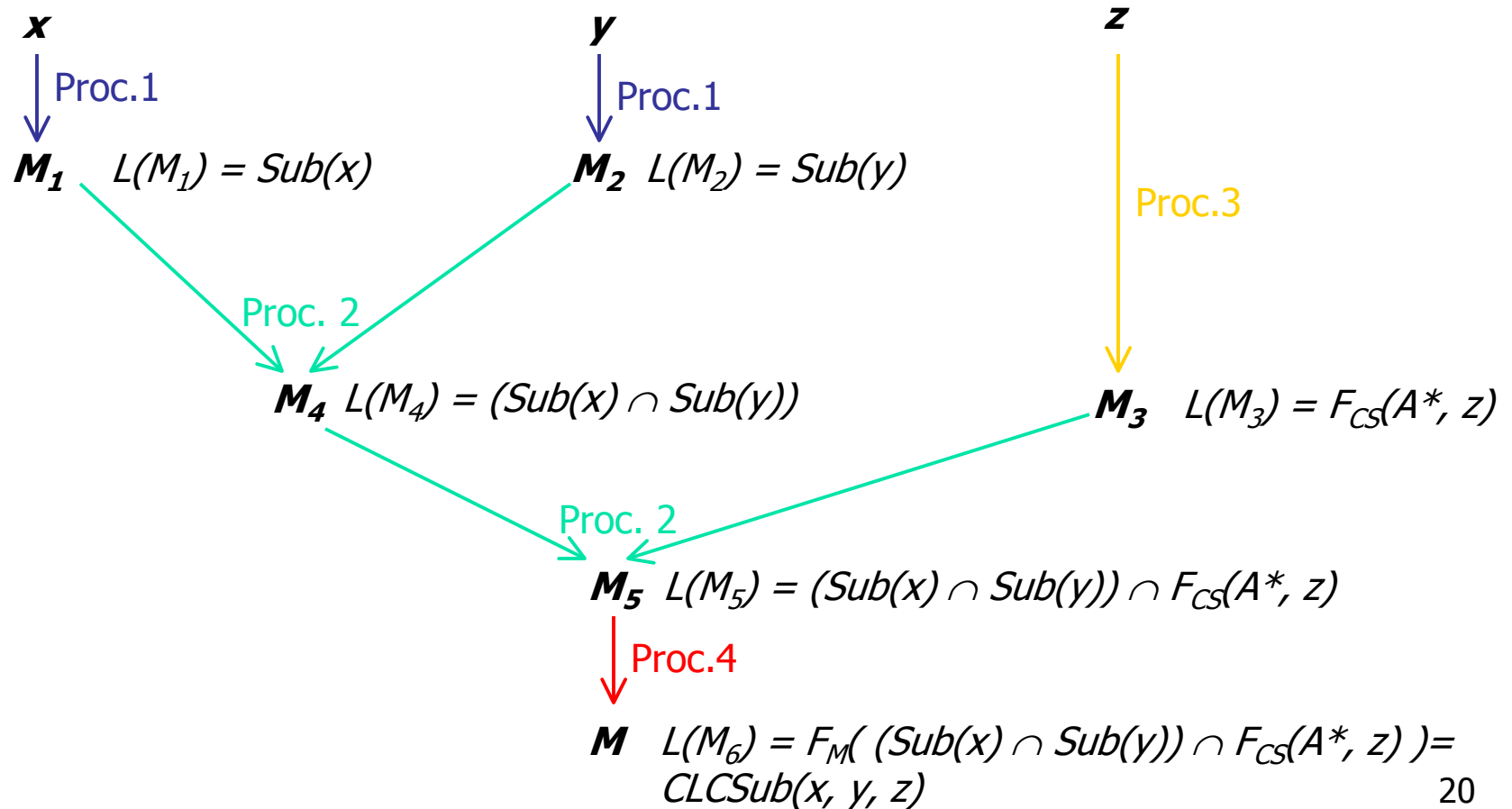
Algorithm outline

$$CLCSub(x, y, z) = F_M((Sub(x) \cap Sub(y)) \cap F_{CS}(A^*, z)) = L(M)$$



Algorithm outline

$$CLCSub(x, y, z) = F_M((Sub(x) \cap Sub(y)) \cap F_{CS}(A^*, z)) = L(M)$$





Procedure 1.

Construction of subsequence automaton

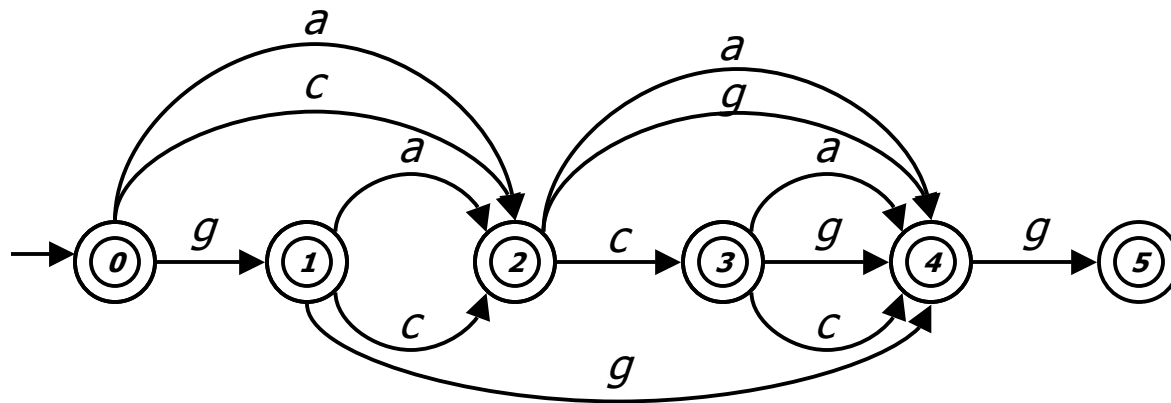
In: degenerate string x , $|x|=n$

Out: M , $L(M)=Sub(x)$

- extension of DASG(x) algorithm – [Troníček et al:1998]
- time complexity: $O(n/A)$
- automaton M
 - acyclic
 - $n+1$ states
 - n/A transitions

Procedure 1.

$$x = [g] \left[\begin{array}{c} a \\ c \end{array} \right] [c] \left[\begin{array}{c} a \\ c \\ g \end{array} \right] [g]$$





Procedure 2.

Construction of automaton for intersection of languages

In: determ. finite automata $M_1, M_2, |M_1|_S = n, |M_2|_S = m$

Out: $M, L(M) = L(M_1) \cap L(M_2)$

- standard algorithm – only accessible
- time complexity: $O(nm/|A|)$
- automaton M properties
 - deterministic
 - at most nm states
 - at most $nm/|A|$ transitions
 - acyclic if M_1 or M_2 is acyclic



Procedure 3.

Construction of constrained subsequence automaton

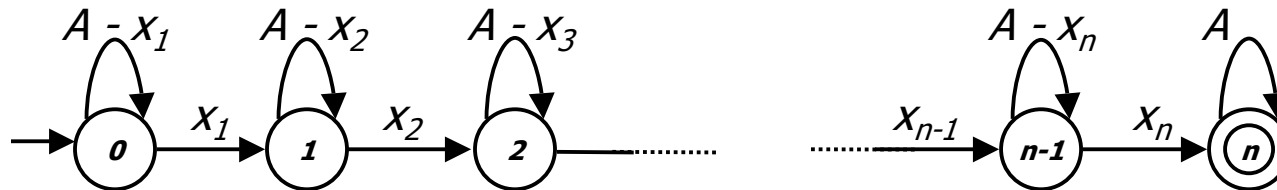
In: finite language L , degenerate string x , $|x|=n$

Out: DFA M , $L(M) = F_{CS}(A^*, x)$

- time complexity: $O(n/|A|)$
- automaton M properties
 - deterministic
 - $n+1$ states
 - $(n+1)/|A|$ transitions

Procedure 3.

Construction of constrained subsequence automaton



$$L(M) = F_{CS}(A^*, x_1 x_2 \dots x_{n-1} x_n)$$



Procedure 4.

Construction of automaton accepting subset of maximal length string for finite language

In: acyclic deterministic finite automaton M_1

Out: deterministic finite automaton M_2 , $L(M_2) = F_M(L(M_1))$

- modif. longest path alg. based on topological ordering of DAGs
- only transitions on longest paths to final states
- time complexity: $O(|M_1|_T |A|)$
- automaton M_2 properties
 - deterministic
 - at most $|M_1|_T$ transitions
 - at most $|M_1|_S$ states



Example

$$x = aba[c,b]$$

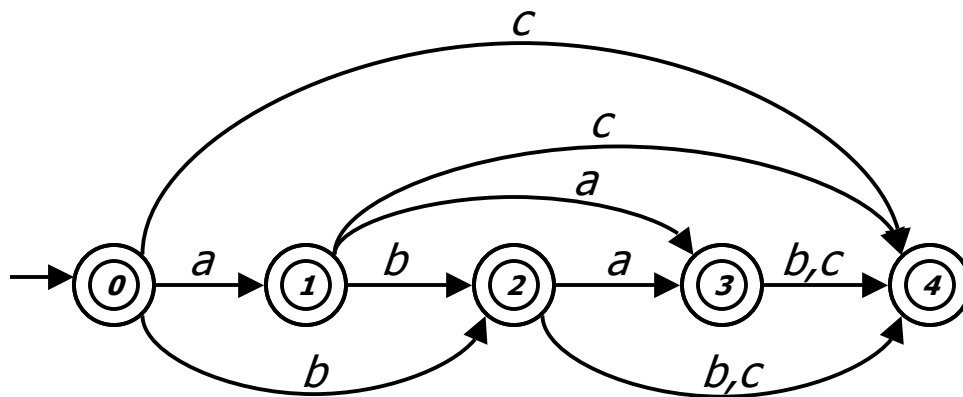
$$z = [a,c]a$$

$$y = abb[a,c]$$

$$CLCSub(x, y, z) = ?$$

Example

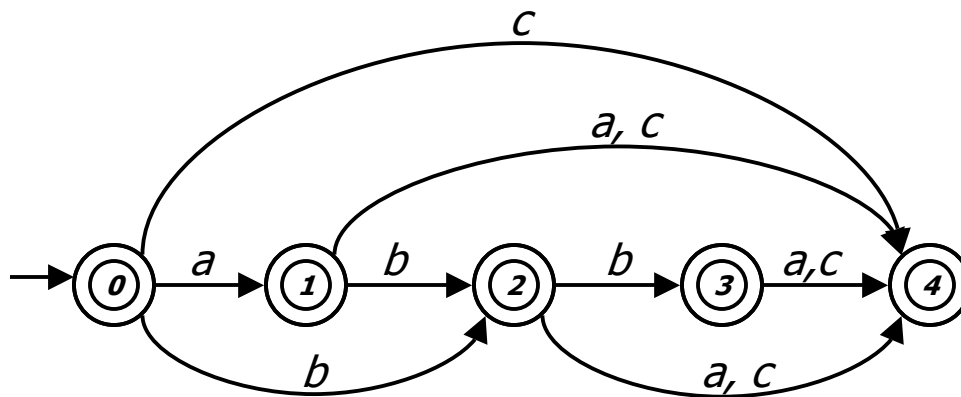
$$L(M_1) = \text{Sub}(aba[c,b])$$



M_1

Example

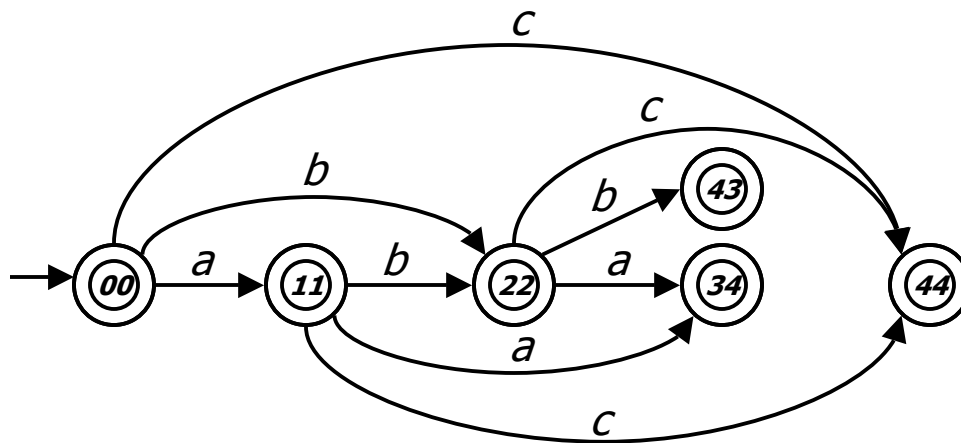
$$L(M_2) = \text{Sub}(abb[a,c])$$



M_2

Example

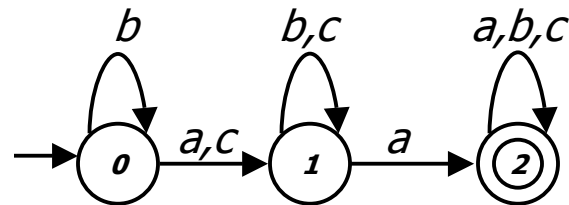
$$L(M_3) = L(M_1) \cap L(M_2) = \text{CommonSub}(x,y)$$



M_3

Example

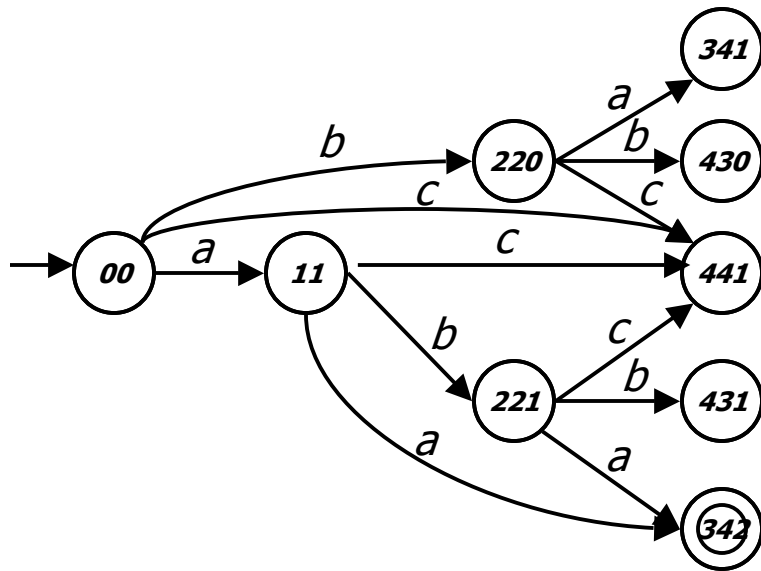
$$L(M_4) = F_{CS}(A^*, [a,c]a)$$



M_4

Example

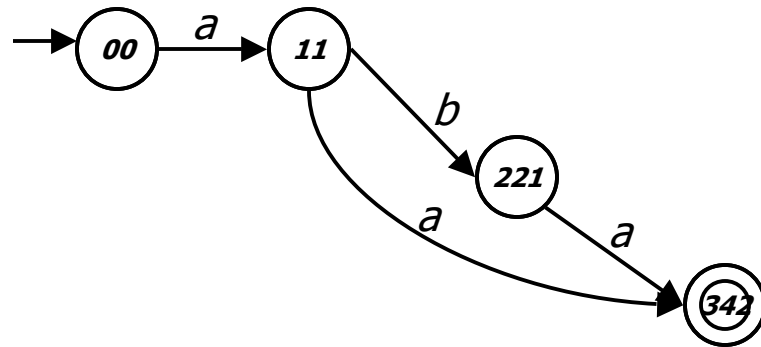
$$L(M_5) = L(M_3) \cap L(M_4) = \text{CCSub}(x,y,z)$$



M_5

Example

$M'_5 = M_5$ without useless states

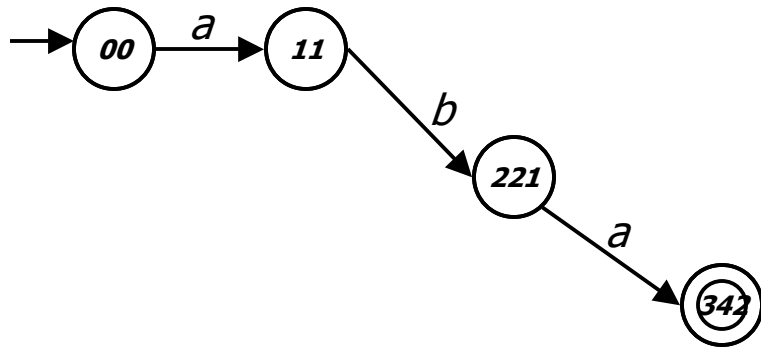


M'_5



Example

$$L(M_6) = F_M(M'_5) = CLCSub(x,y,z)$$

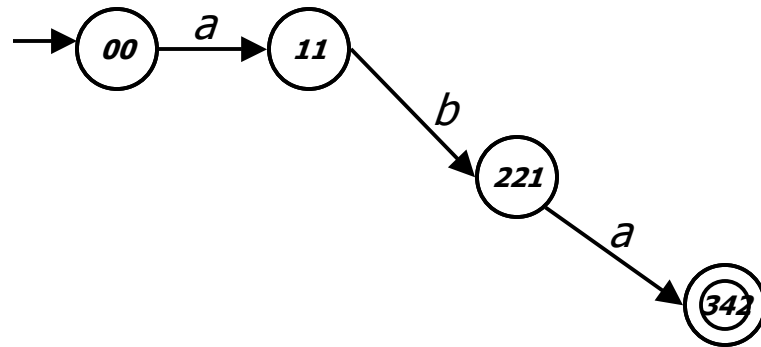


M_6

Example

$$L(M_6) = F_M(M'_5) = CLCSub(x, y, z)$$

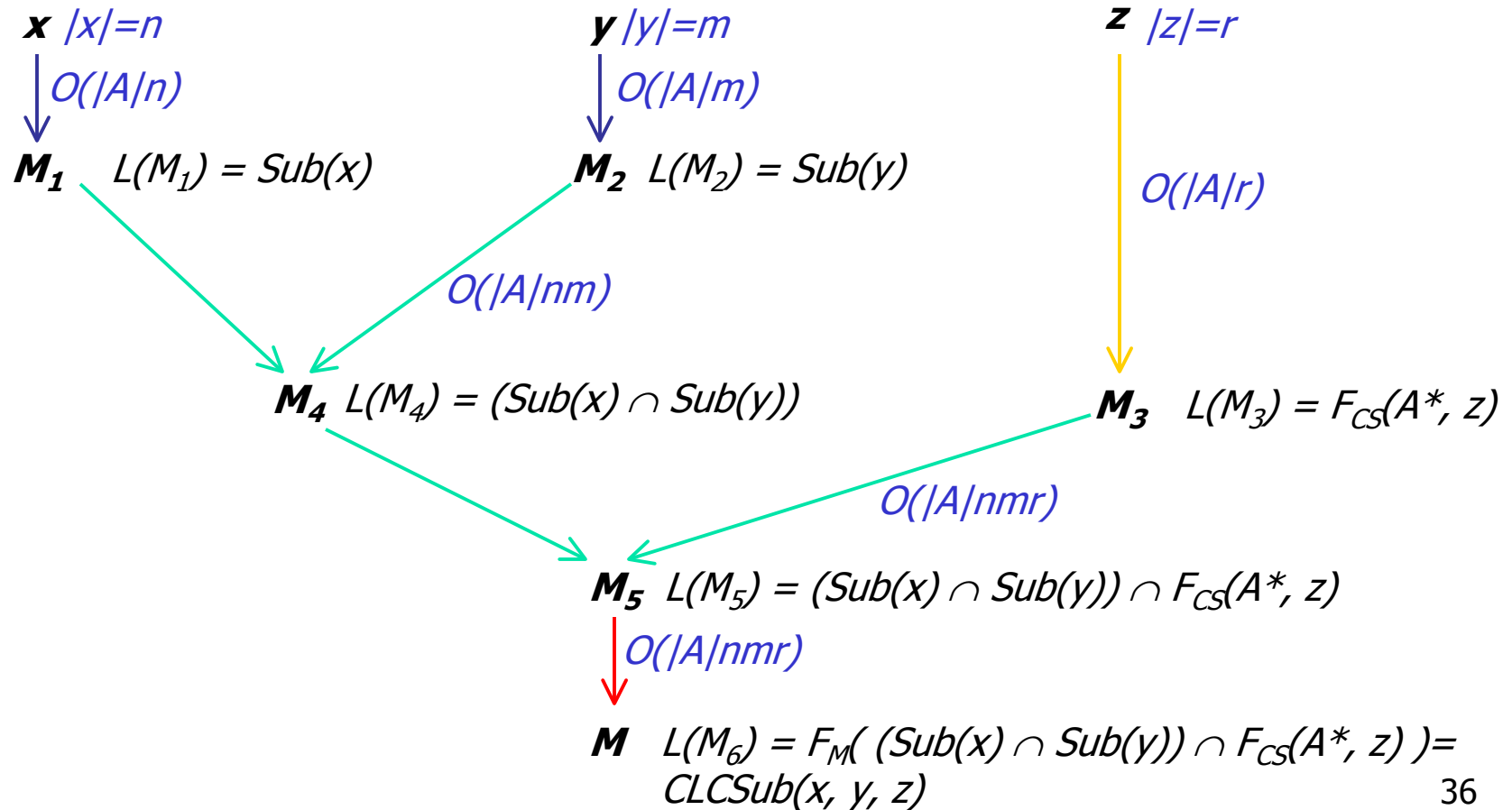
$$CLCSub(aba[c, b], abb[a, c], [a, c]a) = \{aba\}$$



M_6

Algorithm complexity

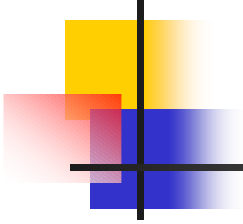
$$CLCSub(x, y, z) = F_M((Sub(x) \cap Sub(y)) \cap F_{CS}(A^*, z)) = L(M)$$





Conclusion

- introduced CLCS problem for DS – new problem
- proposed first algorithm
 - automata based
 - $O(|A|/nmr)$ time
 - useable also on CLCS for strings
 - extensible on multiple DS - $O(|A| (\prod n_i) r)$ time



Q & A