Efficient Algorithms for Regular Expression Constrained Sequence Alignment

Yun-Sheng Chung[†], Chin Lung Lu[¶], and Chuan Yi Tang[†]

[†]Department of Computer Science

National Tsing Hua University, Taiwan

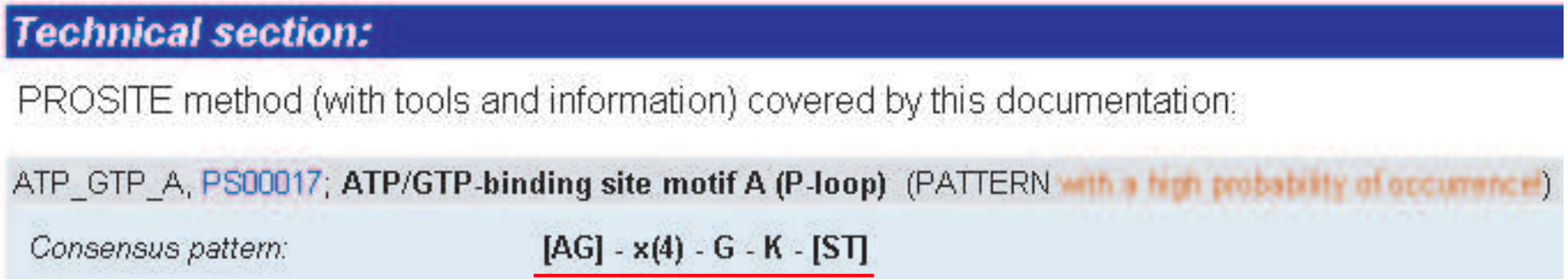
[¶]Department of Biological Science and Technology

National Chiao Tung University, Taiwan

17th Annual Symposium on Combinatorial Pattern Matching

Regular Expression Constraints

- PROSITE contains biologically important sites and motifs.
- PROSITE motifs can be represented by regular expressions.



P-loop motif: [AG]-x(4)-G-K-[ST]
Regular expression: $(\mathtt{A}+\mathtt{G})\Sigma\Sigma\Sigma\Sigma\mathtt{GK}(\mathtt{S}+\mathtt{T})$

Regular Expression Constraints

- In an alignment, it is reasonable to expect functional sites to be aligned together.
- Introduced by Arslan (CPM 2005).

```
T  G  F  P  S  V  G  K  T  K  D  D  -  -  -  -  A
|     |     |  |           |  |  |              |
T  -  F  -  S  V  A  -  -  K  D  D  D  G  K  S  A
```

```
T  -  -  -  G  F  P  S  V  G  K  T  K  D  D  A
|                          |  |              |
T  F  S  V  A  K  D  D  D  G  K  S  -  -  -  A
                *  *  *  *  *  *  *  *
```

$(\mathtt{G}+\mathtt{A})\Sigma\Sigma\Sigma\Sigma\mathtt{GK}(\mathtt{S}+\mathtt{T})$: the P-loop motif.

Outline

- Related works
- Weighted automata
- Arslan's algorithm
- Our algorithms
- Conclusion and future works

Related Works

- Tang et al. (CSB 2003, JBCB 2003) introduced the Constrained Multiple Sequence Alignment (CMSA) problem
  - Constraint: a sequence of characters
  - Example: Sequences from RNase family share the conserved sequence of residues $\mathtt{H}, \mathtt{K}, \mathtt{H}$. In this case the constraint is $\mathtt{H}, \mathtt{K}, \mathtt{H}$.

$$\mathrm{W\,A\,Q\,H} - - \mathrm{K\,P} - \mathrm{H\,C}$$
$$\mathrm{W} - - \mathrm{H\,A\,Q\,K\,P\,Y\,H\,C}$$

- Chin et al. (CSB 2004, JBCB 2005) then proposed a more effieicnt algorithm for pairwise CSA, along with a 2-approximation algorithm for CMSA.

Related Works

- Tsai et al. (Bioinformatics 2004) generalized the definition of constraints.
  - Constraint: Sequence of strings allowing mismatches
  - $P_1 = \texttt{AGCC}, P_2 = \texttt{CG}$
    $\epsilon = 0.25$

$$\cdots \mathrm{A}\,\mathrm{G}\,\mathrm{C}\,\mathrm{C} \cdots \mathrm{C}\,\mathrm{G} \cdots$$
$$\cdots \mathrm{A}\,\mathrm{G}\,\mathrm{U}\,\mathrm{C} \cdots \mathrm{C}\,\mathrm{G} \cdots$$

  - MuSiC: a web tool
- Lu and Huang (Bioinformatics 2005) gave a more space efficient algorithm
  - MuSiC-ME

RECSA

- Arslan (CPM 2005) introduced the regular expression constrained sequence alignment (RECSA) problem.
  - Constraint: Regular expression R.
  - $R = (\mathrm{G} + \mathrm{A})\Sigma\Sigma\Sigma\Sigma\mathrm{GK}(\mathrm{S} + \mathrm{T})$

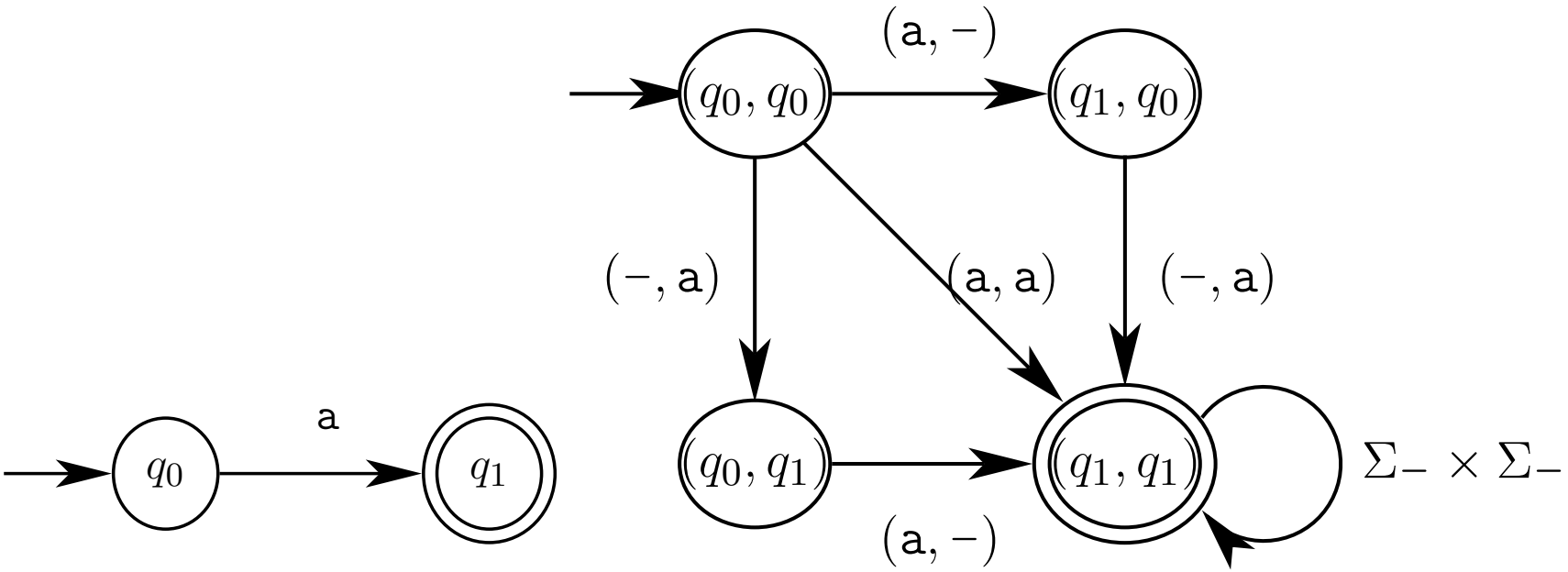
$$\mathrm{T} - - - \mathrm{G}\ \mathrm{F}\ \mathrm{P}\ \mathrm{S}\ \mathrm{V}\ \mathrm{G}\ \mathrm{K}\ \mathrm{T}\ \mathrm{K}\ \mathrm{D}\ \mathrm{D}\ \mathrm{A}$$
$$\mathrm{T}\ \mathrm{F}\ \mathrm{S}\ \mathrm{V}\ \mathrm{A}\ \mathrm{K}\ \mathrm{D}\ \mathrm{D}\ \mathrm{D}\ \mathrm{G}\ \mathrm{K}\ \mathrm{S} - - - \mathrm{A}$$

- In this current work, we give more time and space efficient algorithms.

Weighted Automata: Topology

- Regular expression constraint R is converted into an ϵ-free NFA A.
- The topology of a weighted automaton M is essentially the same as the topology of the product machine of A.

$R = \mathtt{a}$; $\Sigma_- = \Sigma \cup \{-\}$:

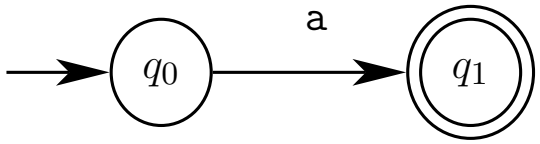


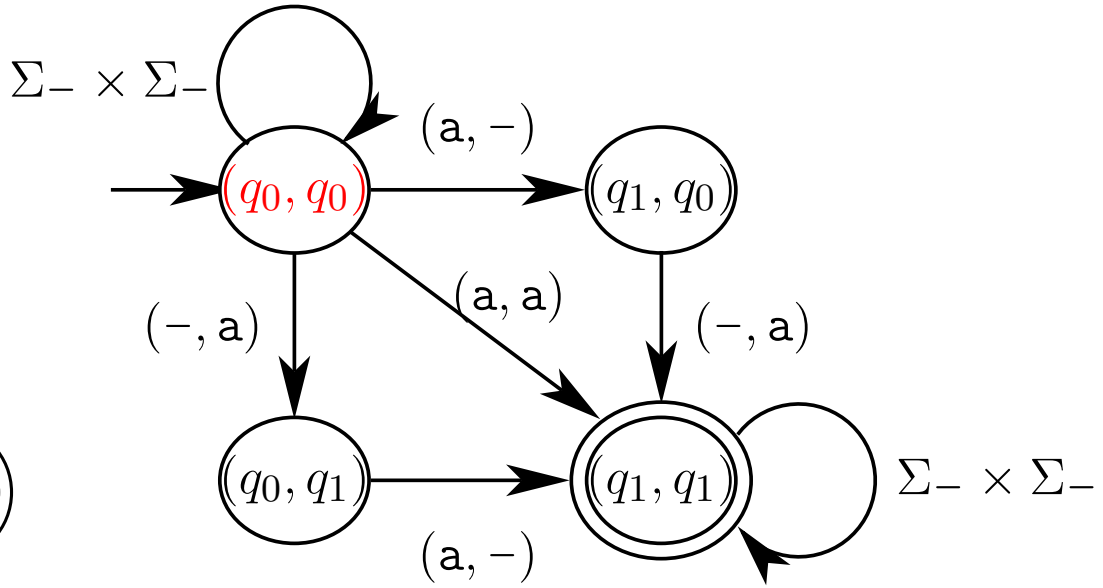
Weighted Automata

Let δ be the transition function of A. Then δ^M, the transition function of M, is defined as

$$\delta^M((p,q),(a,b)) =$$

$$\begin{cases} \delta(p,a) \times \delta(q,b) & \text{if } (p,q) \notin (F \times F) \cup \{(q_0,q_0)\} \\ (\delta(p,a) \times \delta(q,b)) \cup \{(p,q)\} & \text{otherwise} \end{cases}$$

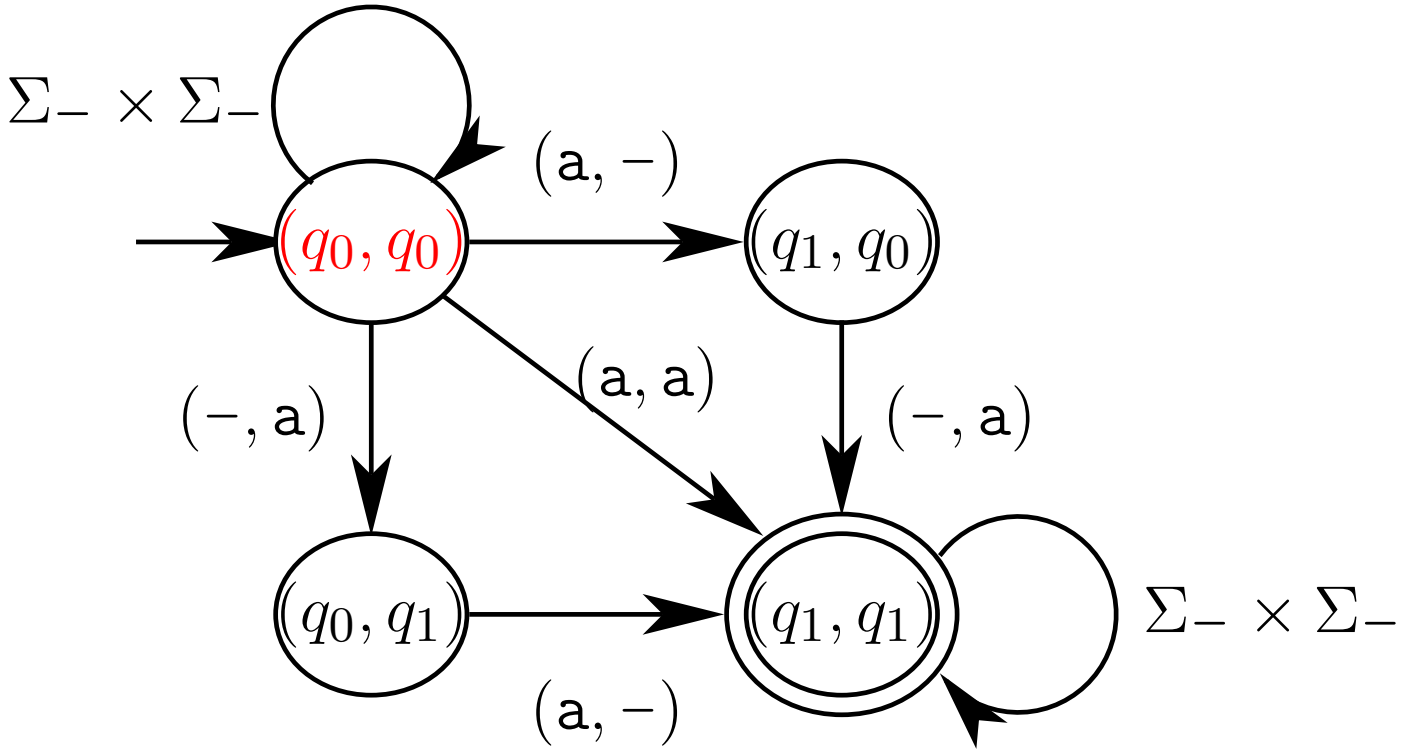




Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

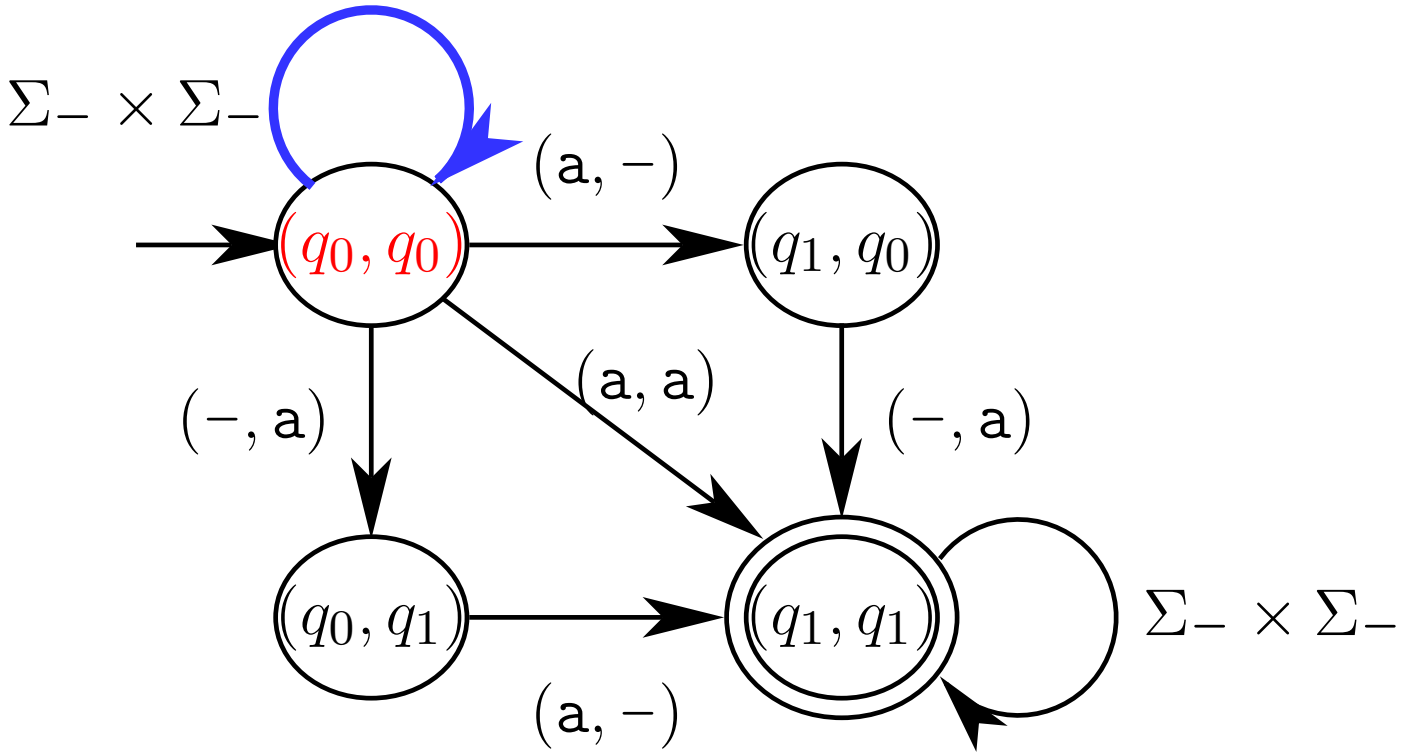
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$



Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

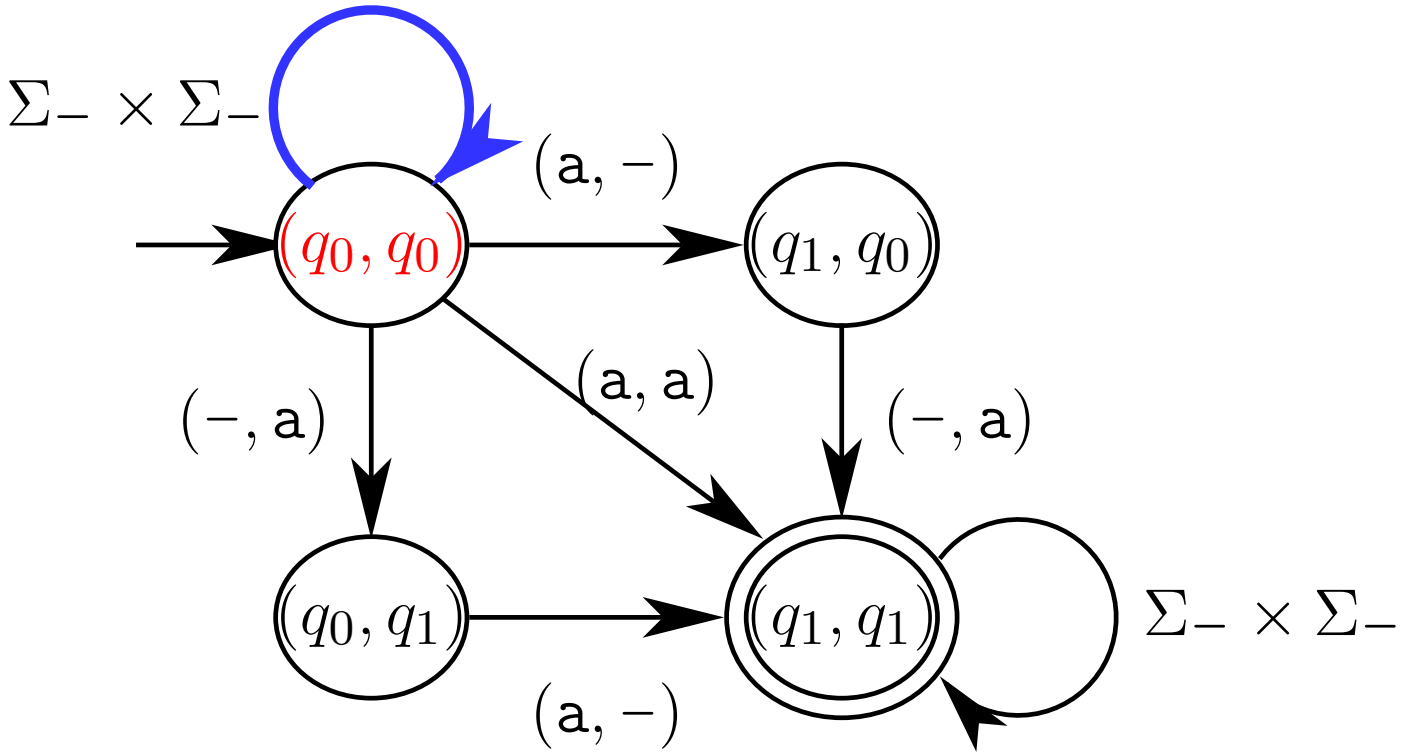
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$



Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

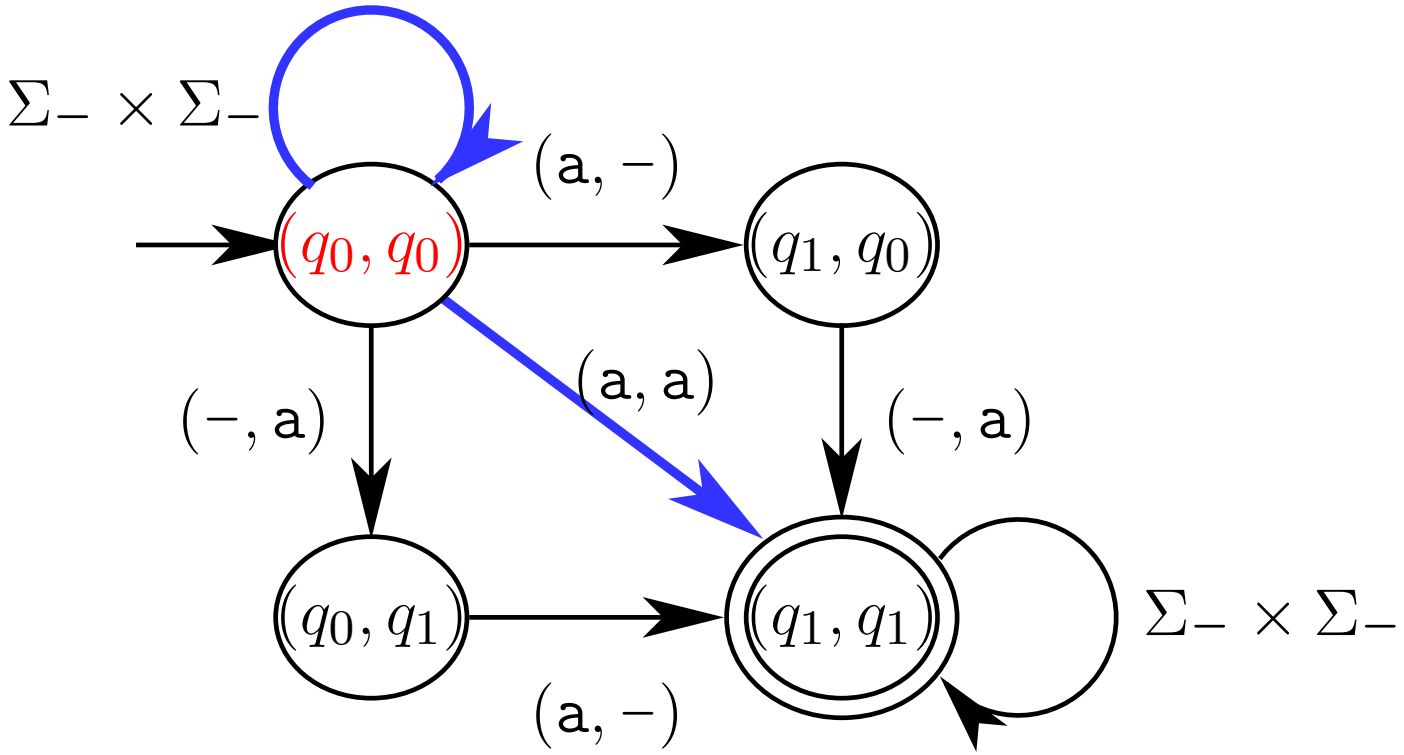
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$



Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

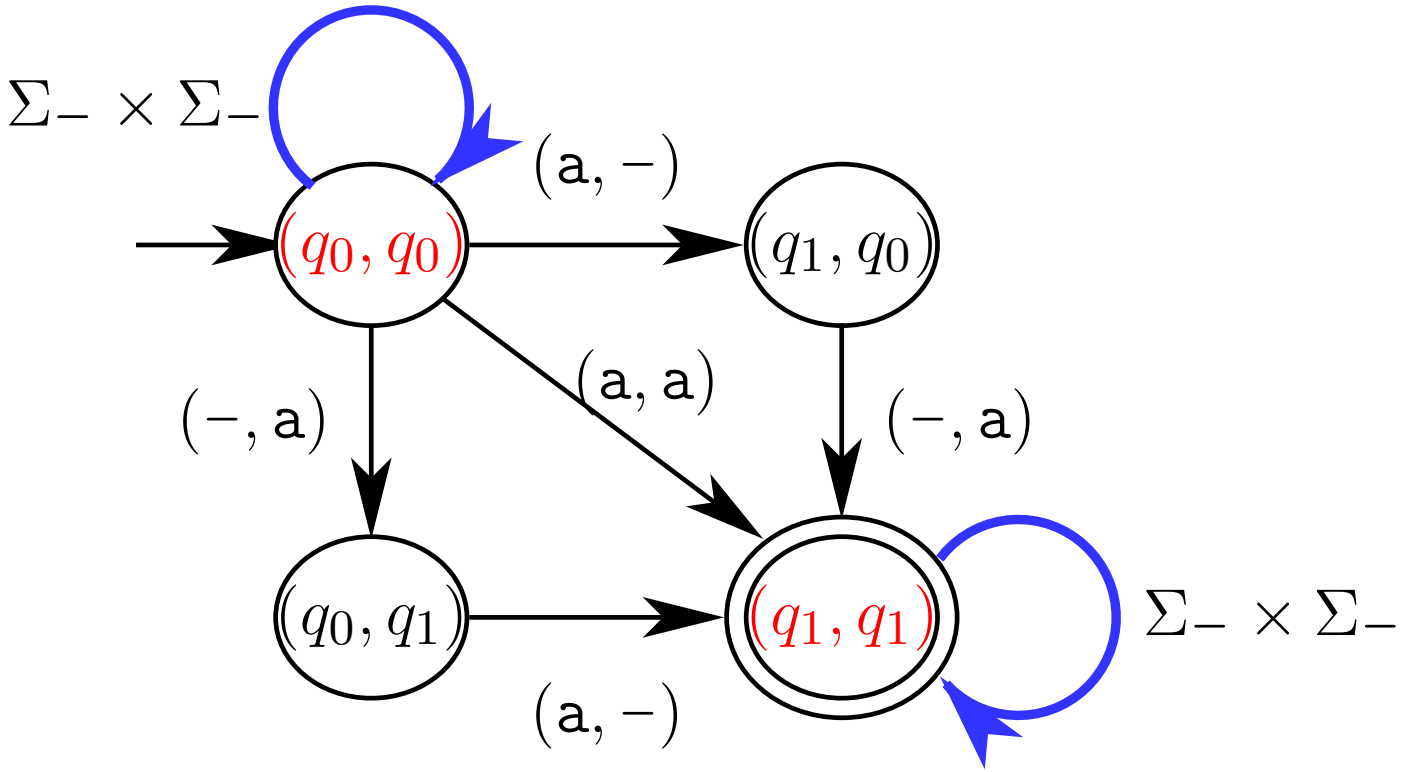
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$



Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

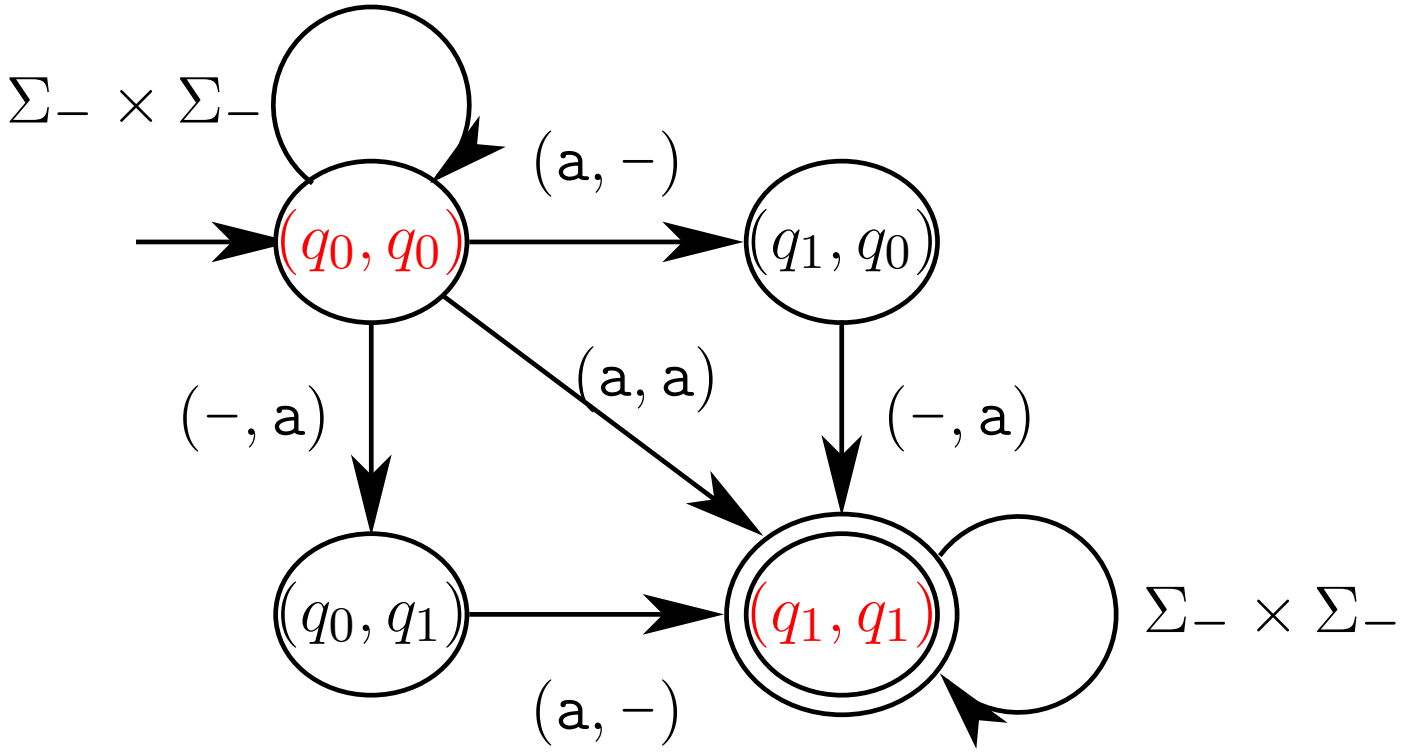
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$



Weighted Automata and Alignments

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

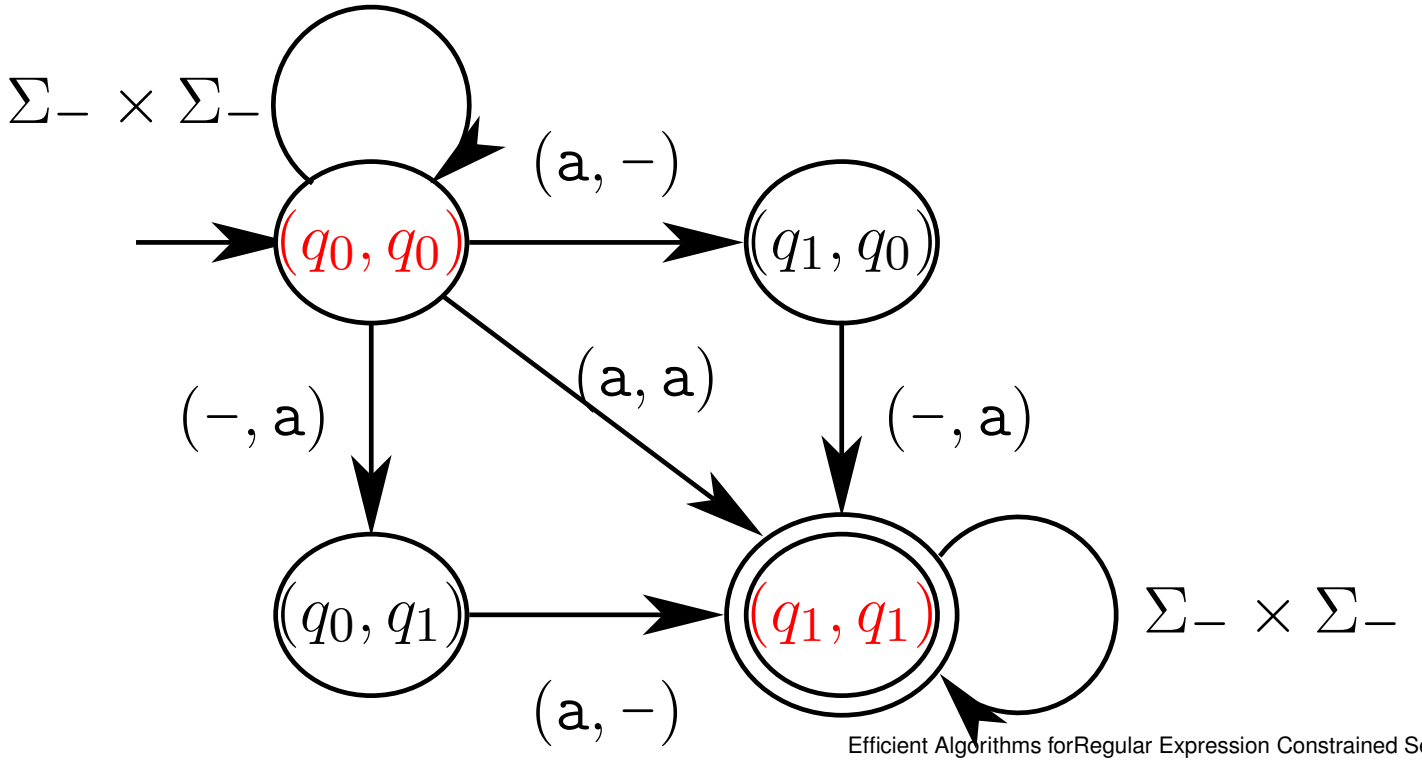
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$ score: 1



Weighted Automata and Alignments

$\mathcal{A}$ reaches the final state if and only if
$\mathcal{A}$ is a feasible constrained alignment

$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - & - \\ - & - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$ score: 0

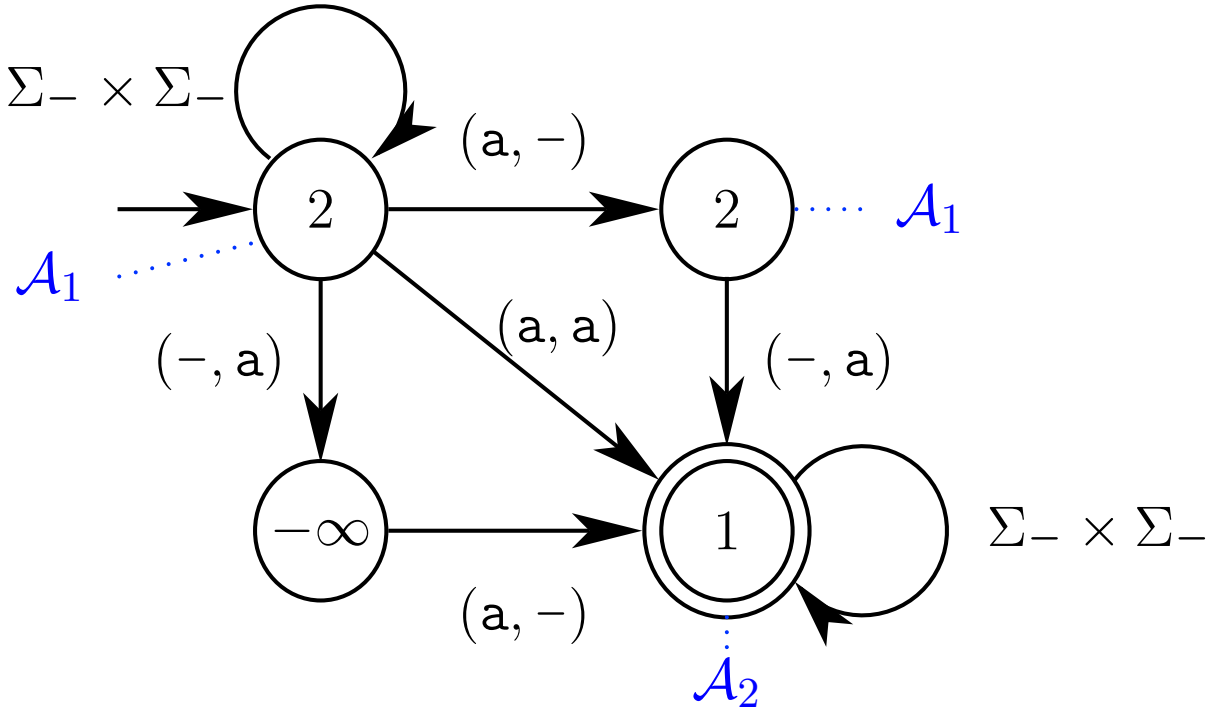


Weighted Automata: Scores

$S_1[1..3] = \texttt{tca}$, $S_2[1..3] = \texttt{atc}$, $R = \texttt{a}$

$$\mathcal{A}_1 = \begin{bmatrix} - & \mathrm{t} & \mathrm{c} & \mathrm{a} \\ \mathrm{a} & \mathrm{t} & \mathrm{c} & - \end{bmatrix} \quad \mathcal{A}_2 = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} & - & - \\ - & - & \mathrm{a} & \mathrm{t} & \mathrm{c} \end{bmatrix}$$

$M_{3,3}$: (match: scored 1; other: scored 0)



Weighted Automata: Computation

	ϵ	a	t	$\cdots$
ϵ	$M_{0,0}$	$M_{0,1}$	$M_{0,2}$	$\cdots$
t	$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	$\cdots$
c	$M_{2,0}$	$M_{2,1}$	$M_{2,2}$	$\cdots$

Let $a = S_1[i_1]$ and $b = S_2[i_2]$:

		$S_2[i_2]$
	M_{i_1-1,i_2-1}	M_{i_1-1,i_2}
$S_1[i_1]$	M_{i_1,i_2-1}	$M_{i_1,i_2} = \max\{M^{(a,b)}_{i_1-1,i_2-1}, M^{(a,-)}_{i_1-1,i_2}, M^{(-,b)}_{i_1,i_2-1}\}$

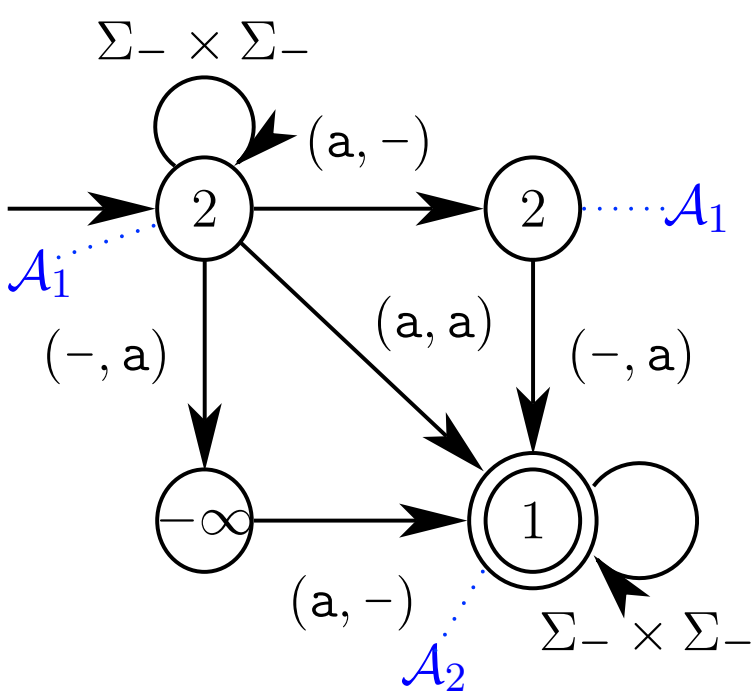
$$W_{i_1,i_2}(p,q) = \max\{W^{(a,b)}_{i_1-1,i_2-1}(p,q), W^{(a,-)}_{i_1-1,i_2}(p,q), W^{(-,b)}_{i_1,i_2-1}(p,q)\}$$

Weighted Automata: Insertion

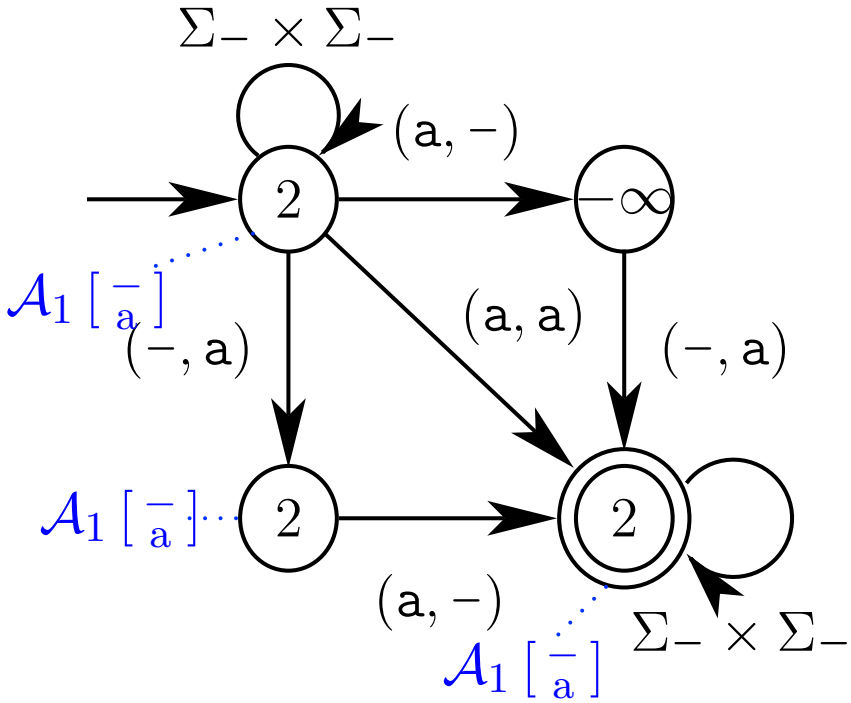
$S_1[1..3] = \texttt{tca}$, $S_2[1..4] = \texttt{atca}$

Additional edit operation: $(-, \texttt{a})$, with score 0

$M_{3,3}$



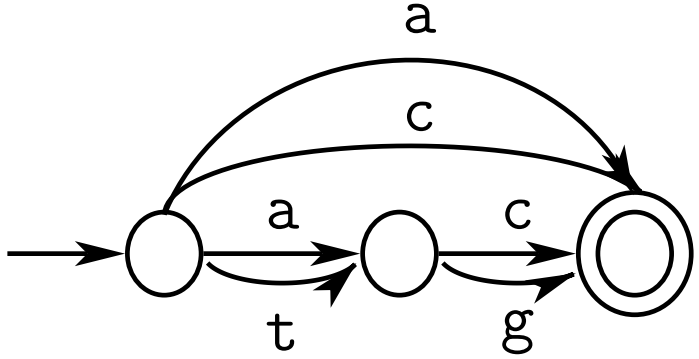
$M_{3,3}^{(-,\texttt{a})}$



The computation can be done by finding in $M_{3,3}$ all the arcs that are labeled with $(-, \texttt{a})$ (or $\Sigma_- \times \Sigma_-$).

Arslan's Algorithm

- Let r be the size of the transition function of M.
- In the algorithm by Arslan, the whole automaton is constructed and searched for each (i_1, i_2), hence taking $O(r)$ time and space.
- Total time and space: $O(rn^2)$ and $O(rn)$, repectively.
- Let V be the set of states in A.



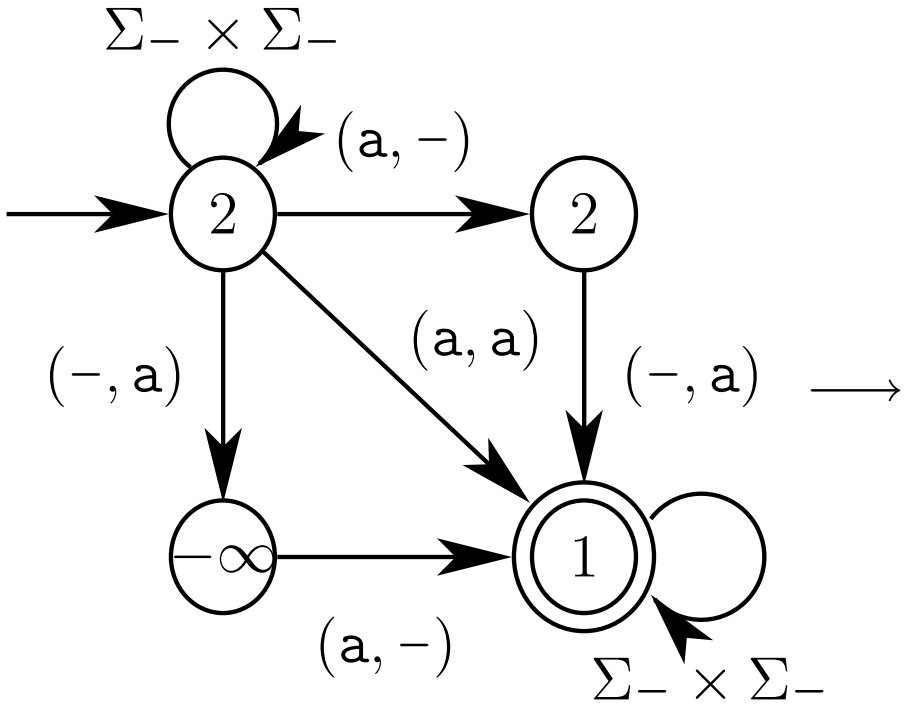
- #arcs in $A = \Theta(|\Sigma|\,|V|^2)$ in the worst case.
- $r =$ #arcs in M, which is $\Theta(|\Sigma|^2|V|^4)$ in the worst case.

Time and Space Complexity

- Worst case complexity for Arslan's algorithm: $O(|\Sigma|^2|V|^4n^2)$ time and $O(|\Sigma|^2|V|^4n)$ space.
- The space complexity is for the computation of the optimal score only. It is not mentioned how to reconstruct the optimal alignment without affecting the space complexity.
- In this work we propose two algorithms.
  - Worst case time: $O(|V|^3n^2)$ and $O(|V|^2 \log |V|n^2)$, respectively.
  - Space: $O(|V|^2n)$, with the optimal alignment reconstructed.

The First Algorithm

- It is more efficient to separate the transitions from the scores.



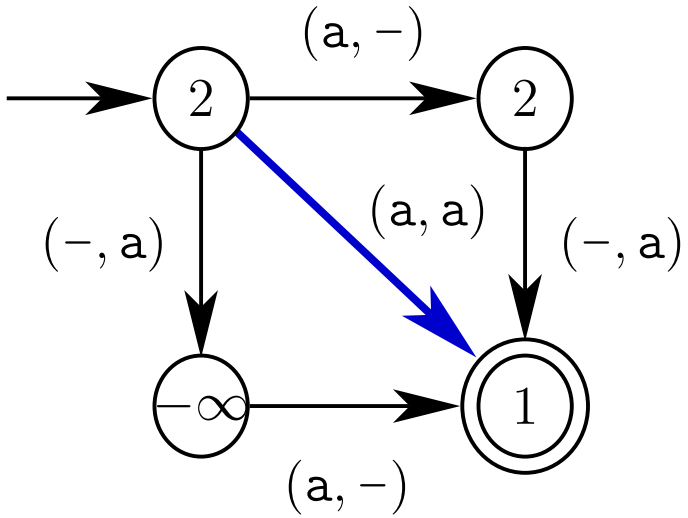
$\longrightarrow$

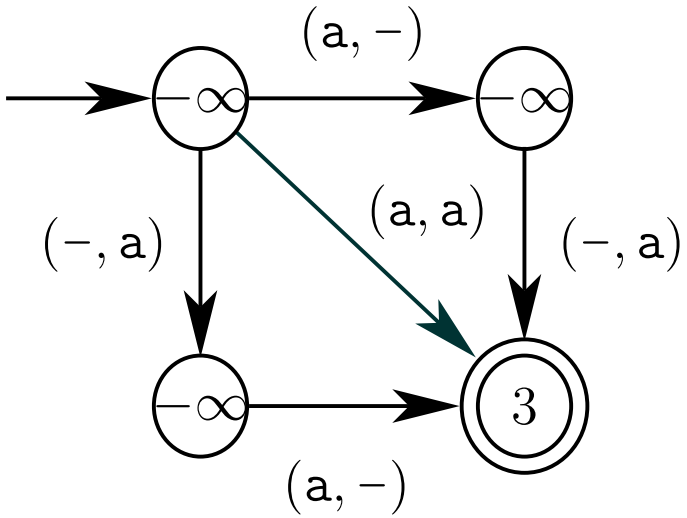
$L_{3,3}$	q_0	q_1
q_0	2	2
q_1	$-\infty$	1

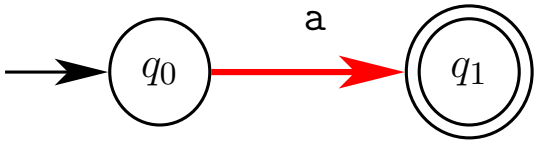
- Worst case space complexity:
  $O(|\Sigma|^2|V|^4n) \longrightarrow O(|V|^2n)$

Search in A, not M—$|V|^4 \longrightarrow |V|^3$

- Consider the computation of $M_{3,3}^{(\mathtt{a},\mathtt{a})}$ from $M_{3,3}$.







$L_{3,3}$	q_0	q_1
$q_0 \downarrow$	2	2
q_1	$-\infty$	1

temp	$q_0 \rightarrow$	q_1
q_0	$-\infty$	$-\infty$
q_1	2	2

$L_{3,3}^{(\mathtt{a},\mathtt{a})}$	q_0	q_1
q_0	$-\infty$	$-\infty$
q_1	$-\infty$	$2+1$

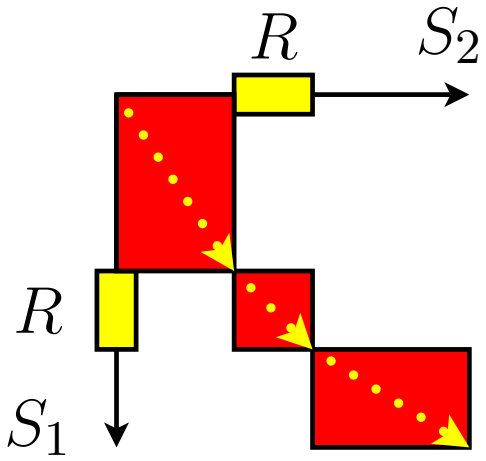
The First Algorithm

Let L be a temporary $|V| \times |V|$ table initialized to be all $-\infty$.
for $p \in V$ **do**
  for $p' \in V$ with $T[p', S_1[i_1]; p] = 1$ **do**
    for $q' \in V$ **do**
      $L[p, q'] \leftarrow$
        $\max\{L[p, q'], L_{i_1-1,i_2-1}[p', q'] + \gamma(S_1[i_1], S_2[i_2])\}$;
for $q \in V$ **do**
  for $q' \in V$ with $T[q', S_2[i_2]; q] = 1$ **do**
    for $p \in V$ **do**
      $L^{(S_1[i_1],S_2[i_2])}_{i_1-1,i_2-1}[p, q] \leftarrow \max\{L^{(S_1[i_1],S_2[i_2])}_{i_1-1,i_2-1}[p, q], L[p, q']\}$;

The insertion and deletion cases are similar.

Reconstruct the Alignment

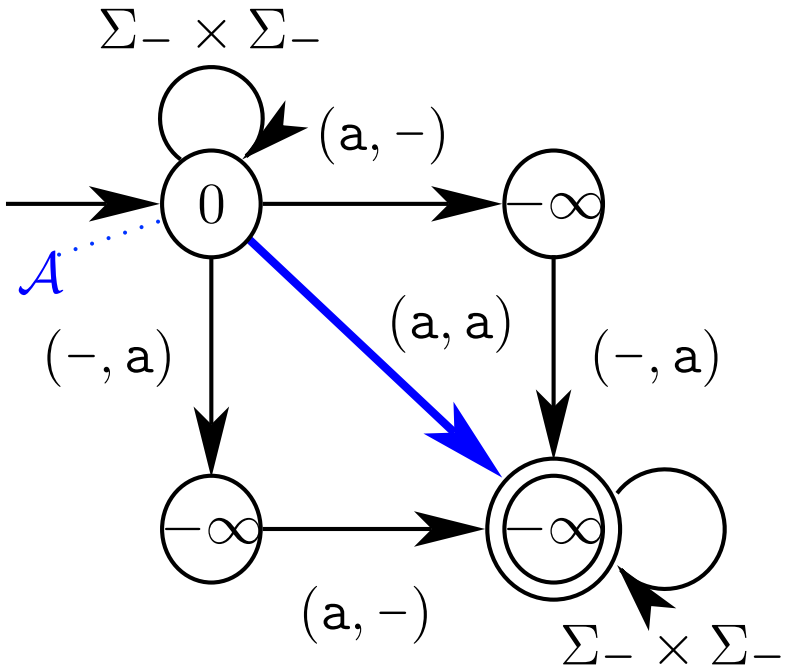
If we know which substrings of S_1 and S_2 are aligned to satisfy the constraint in an optimal constrained alignment, then the optimal alignment can be reconstructed easily in $O(n)$ space.



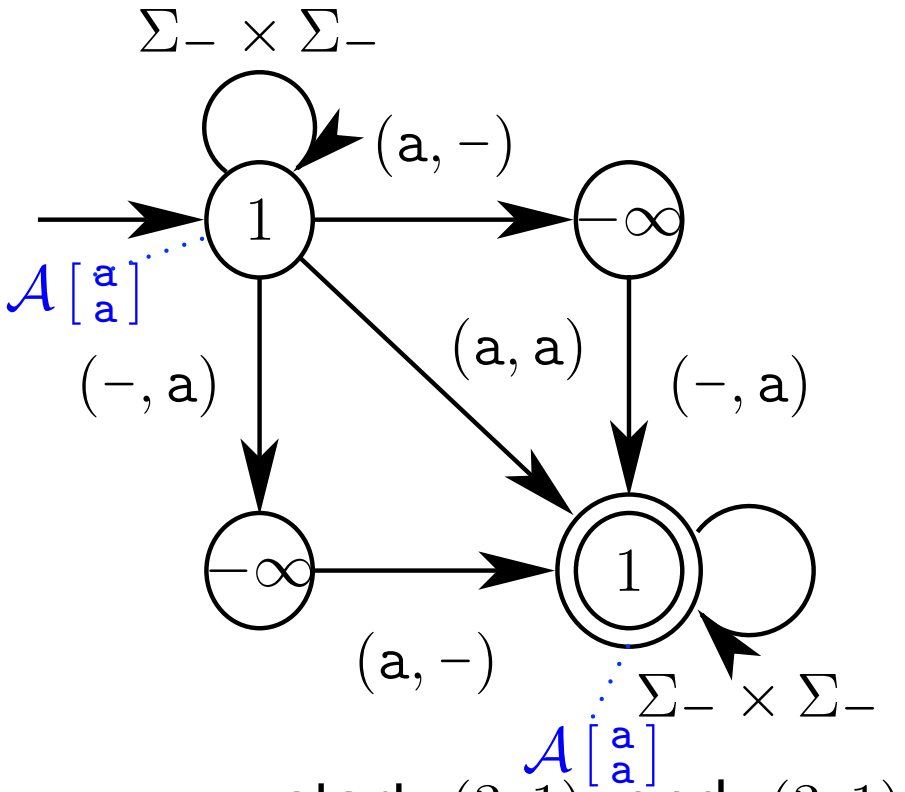
The indices of such substrings can be found by doing some additional bookkeeping during the computation without affecting the space complexity mentioned previously.

Reconstruct the Alignment

$M_{2,0}$



$M_{2,0}^{(\mathsf{a},\mathsf{a})}$



start: $(3, 1)$; end: $(3, 1)$

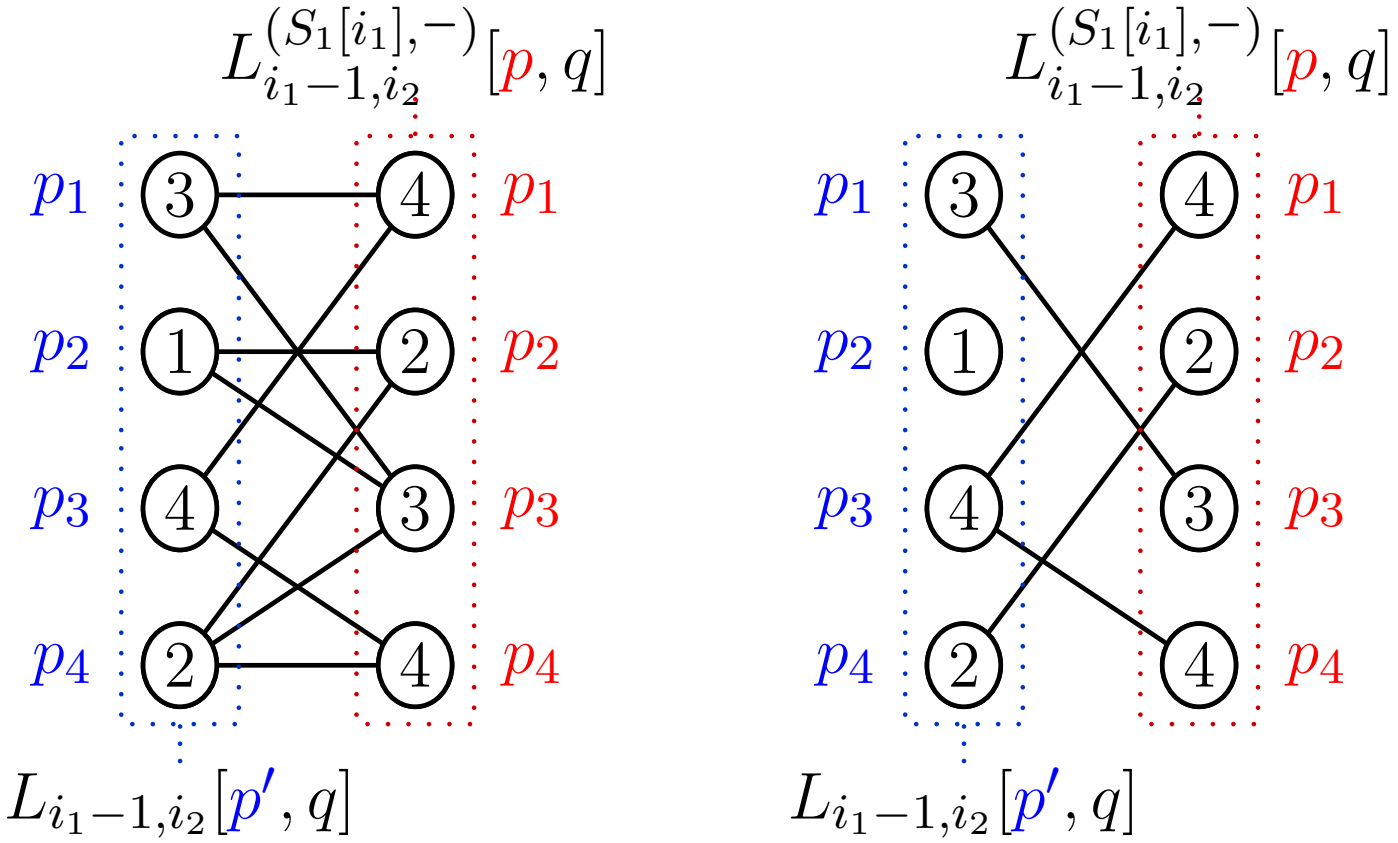
$$\mathcal{A} = \begin{bmatrix} \mathrm{t} & \mathrm{c} \\ - & - \end{bmatrix}, \mathcal{A}\left[\begin{smallmatrix}\mathsf{a}\\\mathsf{a}\end{smallmatrix}\right] = \begin{bmatrix} \mathrm{t} & \mathrm{c} & \mathrm{a} \\ - & - & \mathrm{a} \end{bmatrix}$$

The Second Algorithm

- The second algorithm requires $|V| = O(\log n)$.
- Let $T[p', a]$ be the bit vector $(T[p', a; p_1], \ldots, T[p', a; p_{|V|}])$.
- Under the assumption, $T[p', a]$ can be stored in a single word.
- We take $L_{i_1-1,i_2}^{(S_1[i_1],-)}[p, q]$ as an example. The other two cases are similar.

The Second Algorithm

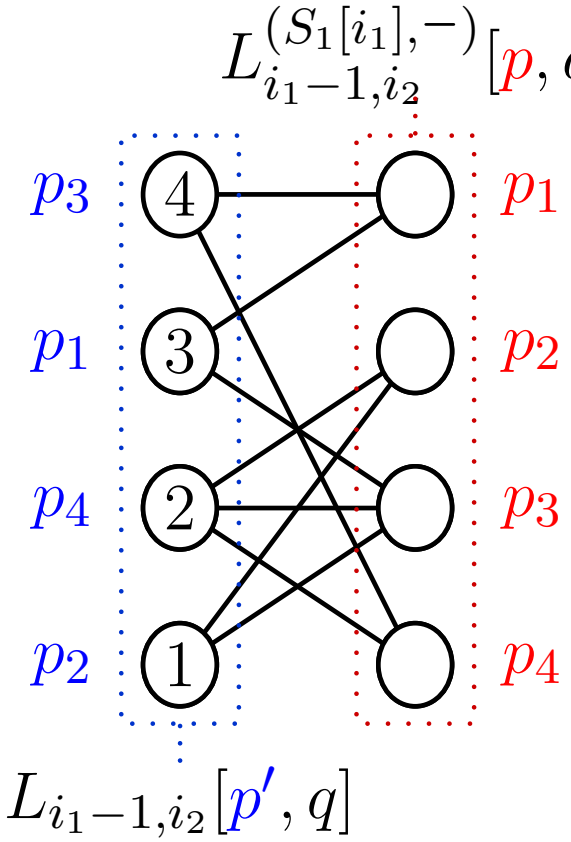
The arcs shown are all transitions in δ labeled with $S_1[i_1]$.
$V = \{p_1, p_2, p_3, p_4\}$. Fix $q \in V$.
$T[p_1, S_1[i_1]] = (1, 0, 1, 0), T[p_2, S_1[i_1]] = (0, 1, 1, 0),$
$T[p_3, S_1[i_1]] = (1, 0, 0, 1), T[p_4, S_1[i_1]] = (0, 1, 1, 1)$

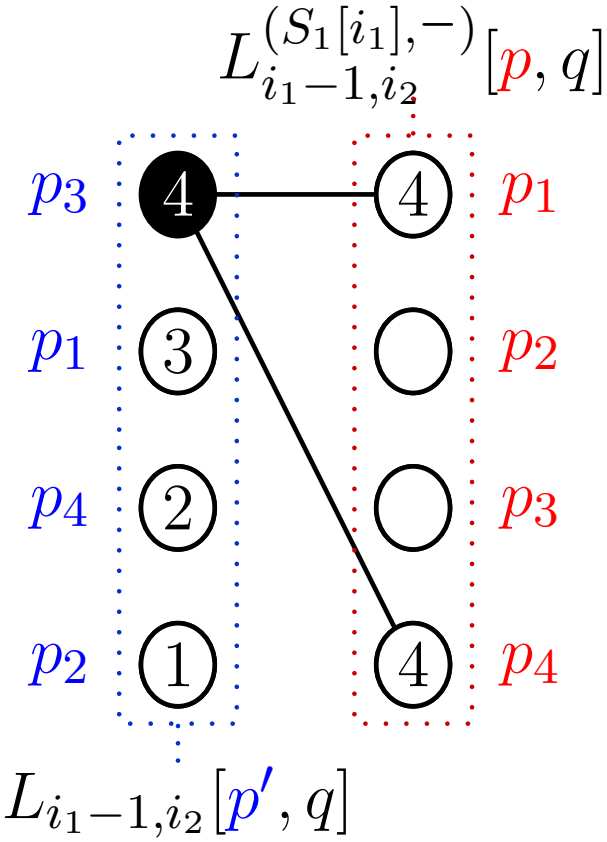


The Second Algorithm

Y: Bit vector representing the states p in V such that $L_{i_1-1,i_2}^{(S_1[i_1],-)}[p,q]$ has not yet been set.

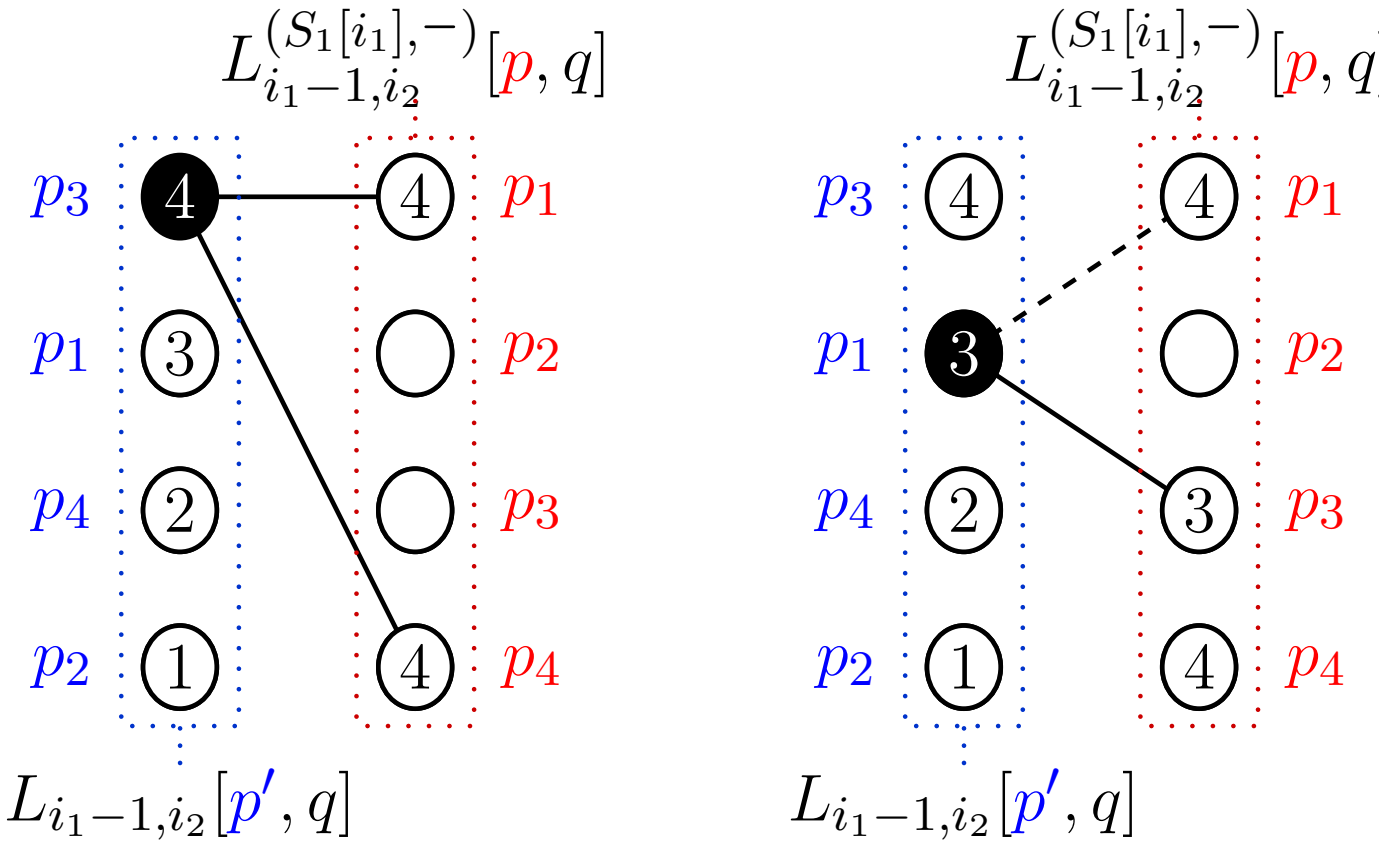
Initially: $Y = (1,1,1,1)$. $T[p_3, S_1[i_1]] \otimes Y = (1,0,0,1)$





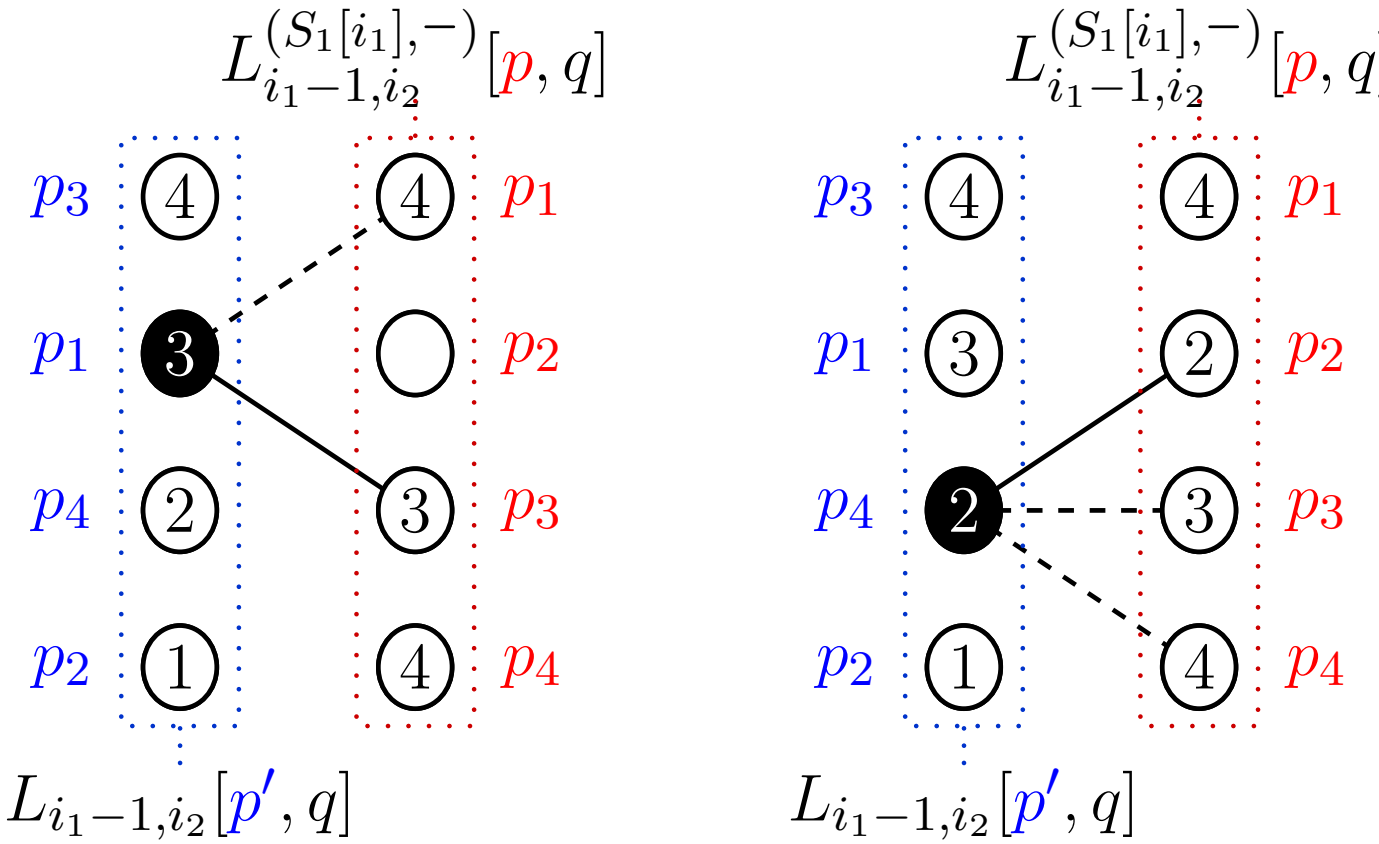
The Second Algorithm

$$Y \leftarrow Y - T[p_3, S_1[i_1]] \otimes Y = (1,1,1,1) - (1,0,0,1) = (0,1,1,0)$$
$$T[p_1, S_1[i_1]] \otimes Y = (0,0,1,0)$$



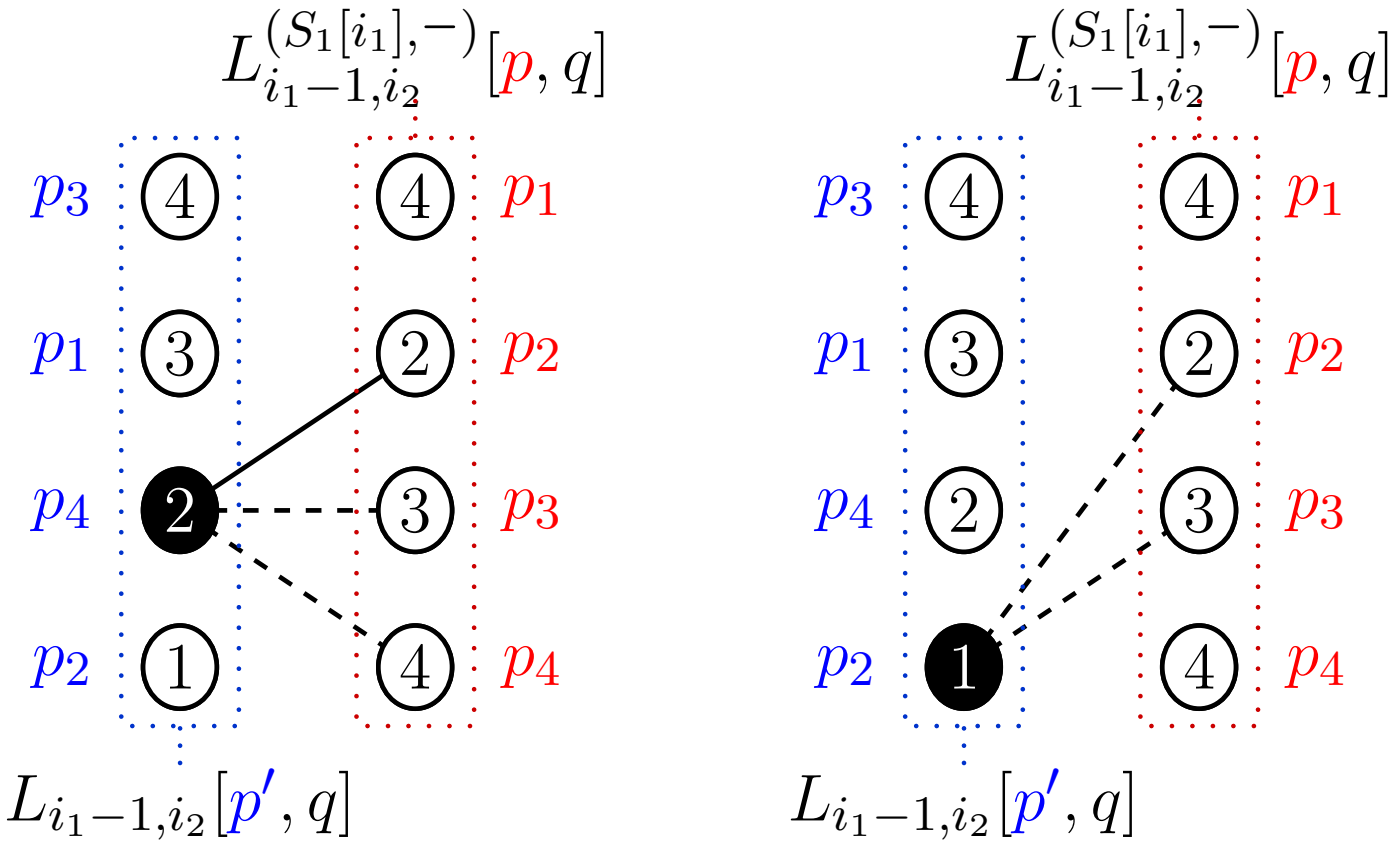
The Second Algorithm

$$Y \leftarrow Y - T[p_1, S_1[i_1]] \otimes Y = (0,1,1,0) - (0,0,1,0) = (0,1,0,0)$$
$$T[p_4, S_1[i_1]] \otimes Y = (0,1,0,0)$$

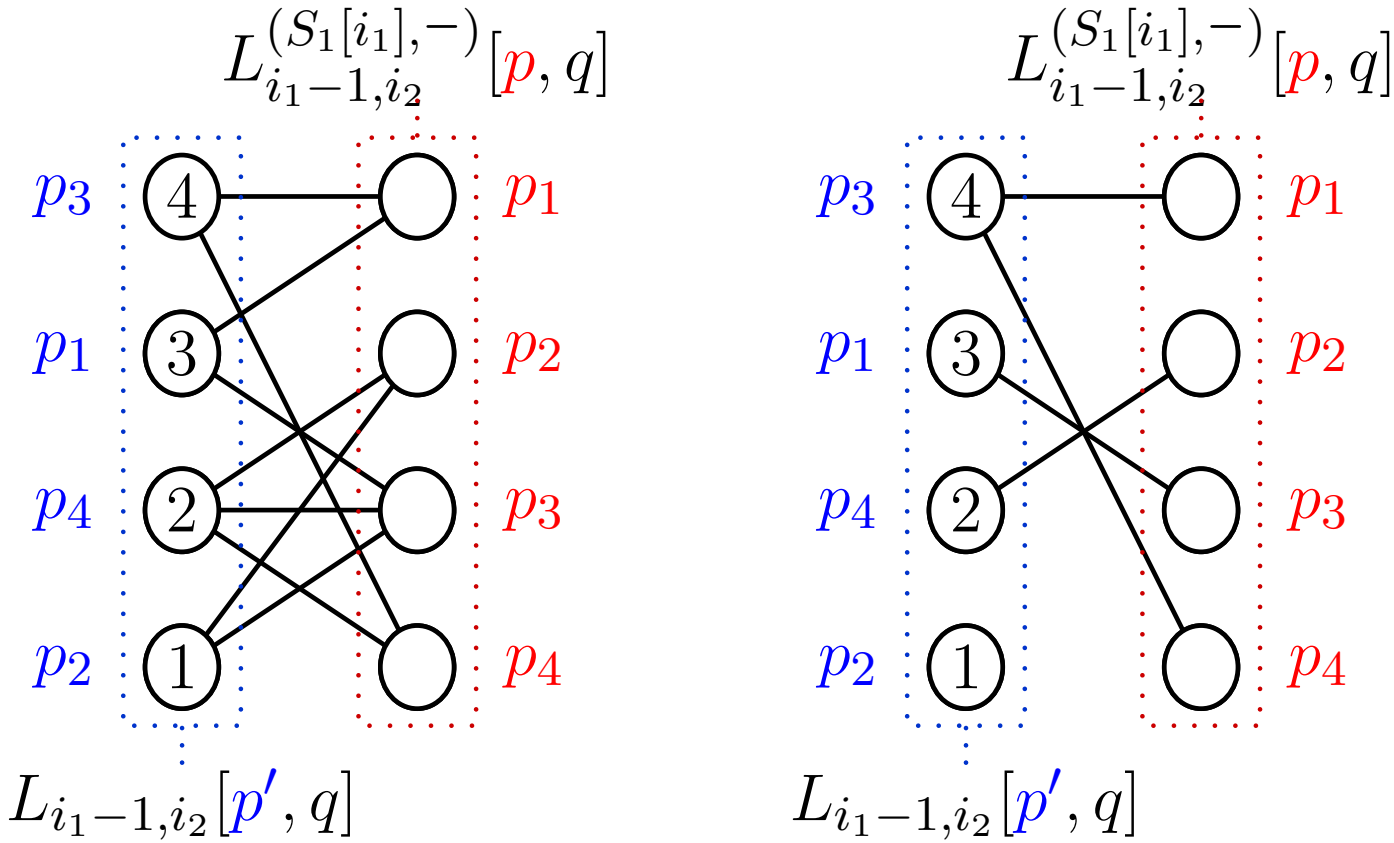


The Second Algorithm

$$Y \leftarrow Y - T[p_4, S_1[i_1]] \otimes Y = (0,1,0,0) - (0,1,0,0) = (0,0,0,0)$$
$$T[p_2, S_1[i_1]] \otimes Y = (0,0,0,0)$$



The Second Algorithm



Time: $O(|V|\log|V|)$ for each fixed q; $O(|V|^2\log|V|)$ for the entire $L^{(S_1[i_1],-)}_{i_1-1,i_2}$

Conclusion and Future Works

- More time and space efficient algorithms for regular expression constrained sequence alignment are proposed.
- A multiple alignment version is important.
  - Generalize the weighted automata structure to obtain optimal solutions: suggested by Arslan (CPM 2005).
  - Progressive heuristics: e.g., Lu and Huang's work (Bioinformatics 2005).
  - Approximation algorithms: in preparation.
- Local alignment version would also be useful.
  - Has a similar motivation as PHI-BLAST; may work complementarily by taking a different approach.

Thank you for your attention.