Minimization of FSA

Data Structures and Algorithms for Computational Linguistics III (ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2024/25

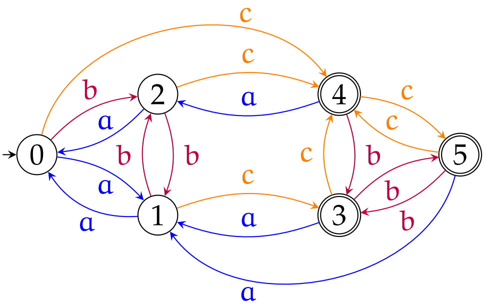
version: c8d9876 @2025-01-08

DFA minimization

- For any regular language, there is a unique *minimal* DFA
- By finding the minimal DFA, we can also prove equivalence (or not) of different FSA and the languages they recognize
- In general the idea is:
  - Throw away unreachable states (easy)
  - Merge equivalent states
- There are two well-known algorithms for minimization:
  - Hopcroft's algorithm: find and eliminate equivalent states by partitioning the set of states
  - Brzozowski's algorithm: 'double reversal'

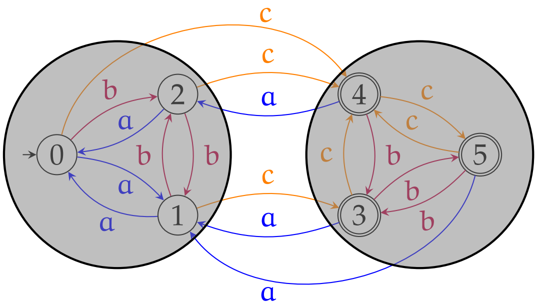
Finding equivalent states

Intuition



Finding equivalent states

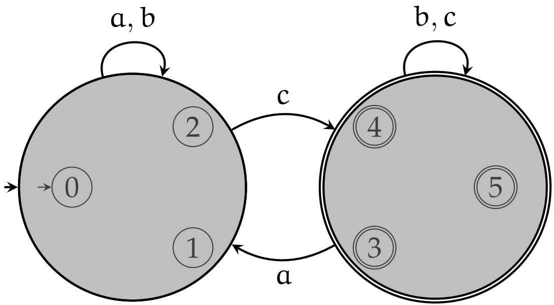
Intuition



The edges leaving the group of nodes are identical.
Their *right languages* are the same.

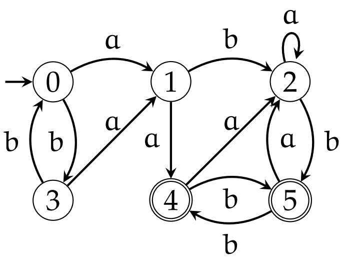
Finding equivalent states

Intuition

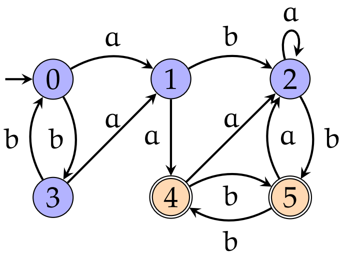


The edges leaving the group of nodes are identical.
Their *right languages* are the same.

Minimization by partitioning

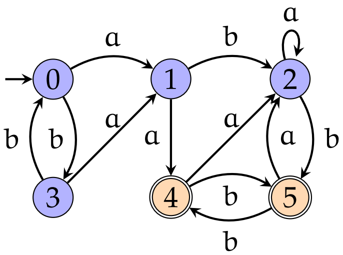


Minimization by partitioning



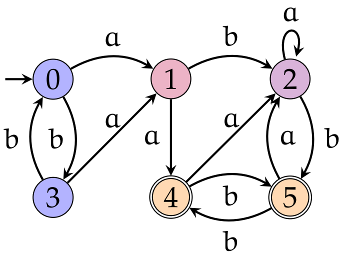
- Accepting & non-accepting states form a partition
  $Q_1 = \{0, 1, 2, 3\}$, $Q_2 = \{4, 5\}$

Minimization by partitioning



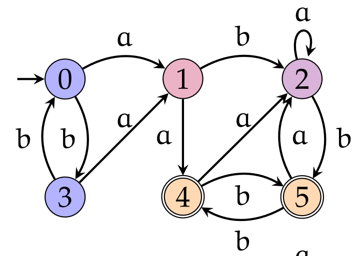
- Accepting & non-accepting states form a partition
  $Q_1 = \{0, 1, 2, 3\}$, $Q_2 = \{4, 5\}$
- For any of the nodes in a set, if any symbol leads to different sets of nodes, split

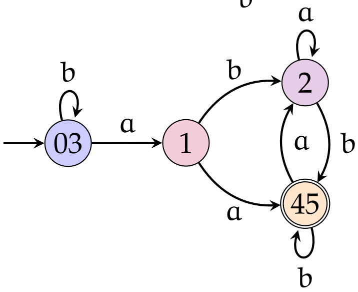
Minimization by partitioning



- Accepting & non-accepting states form a partition
  $Q_1 = \{0, 1, 2, 3\}$, $Q_2 = \{4, 5\}$
- For any of the nodes in a set, if any symbol leads to different sets of nodes, split
- $Q_1 = \{0, 3\}$, $Q_3 = \{1\}$, $Q_4 = \{2\}$, $Q_2 = \{4, 5\}$

Minimization by partitioning

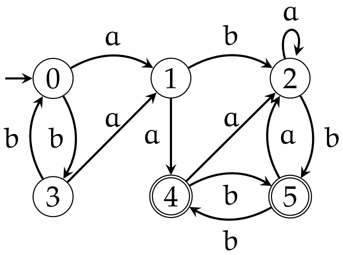




- Accepting & non-accepting states form a partition
  $Q_1 = \{0, 1, 2, 3\}$, $Q_2 = \{4, 5\}$
- For any of the nodes in a set, if any symbol leads to different sets of nodes, split
- $Q_1 = \{0, 3\}$, $Q_3 = \{1\}$, $Q_4 = \{2\}$, $Q_2 = \{4, 5\}$
- Stop when we cannot split any of the sets, merge the indistinguishable states

Minimization by partitioning

tabular version

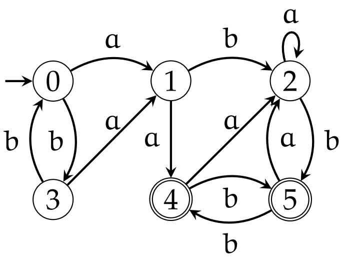


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

1					
2					
3					
4					
5					
	0	1	2	3	4

Minimization by partitioning

tabular version

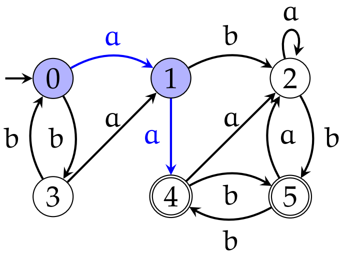


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

	0	1	2	3	4
1					
2					
3					
4	●	●	●	●	
5	●	●	●	●	

Minimization by partitioning

tabular version

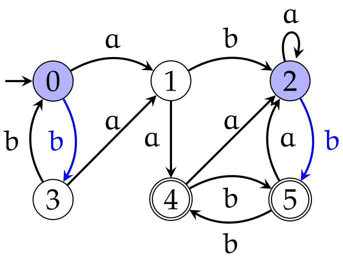


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

	0	1	2	3	4
1					
2					
3					
4	●	●	●	●	
5	●	●	●	●	

Minimization by partitioning

tabular version

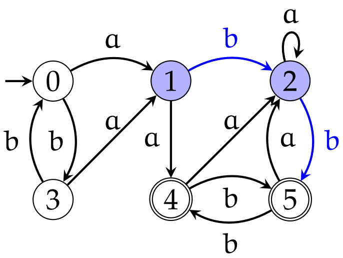


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

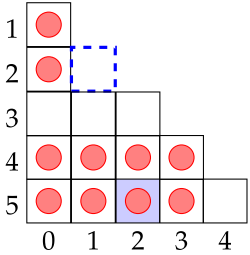
	0	1	2	3	4
1	●				
2					
3					
4	●	●	●	●	
5	●	●	●	●	

Minimization by partitioning

tabular version

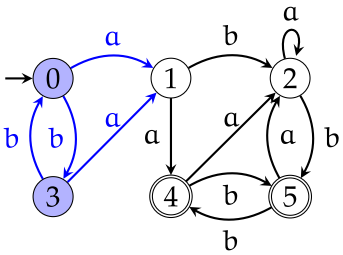


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

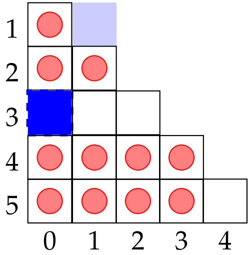


Minimization by partitioning

tabular version

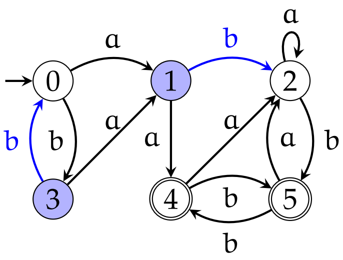


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

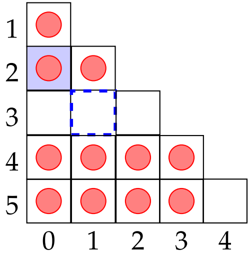


Minimization by partitioning

tabular version

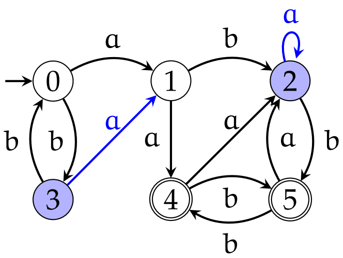


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$



Minimization by partitioning

tabular version

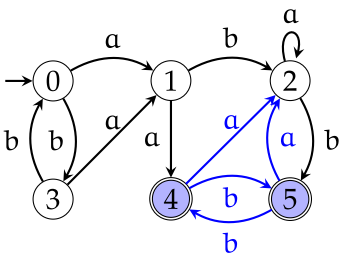


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

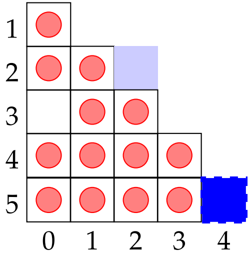
	0	1	2	3	4
1	●				
2	●	●			
3		●			
4	●	●	●	●	
5	●	●	●	●	

Minimization by partitioning

tabular version

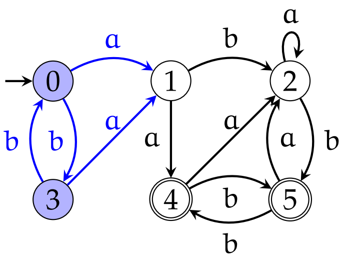


- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

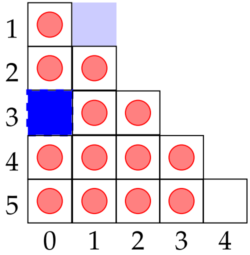


Minimization by partitioning

tabular version

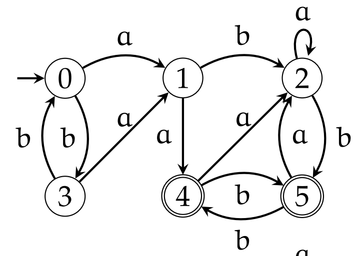


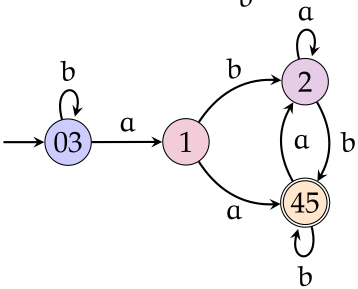
- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$



Minimization by partitioning

tabular version





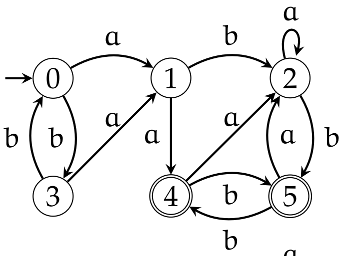
- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

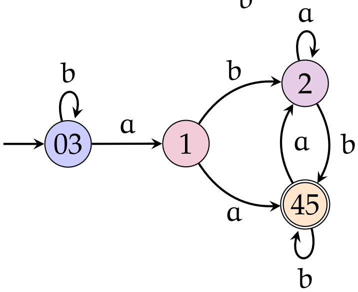
	0	1	2	3	4
1	●				
2	●	●			
3		●	●		
4	●	●	●	●	
5	●	●	●	●	

- Merge indistinguishable states

Minimization by partitioning

tabular version





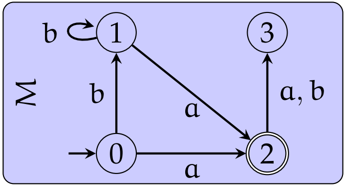
- Create a state-by-state table, mark *distinguishable* pairs: (q_1, q_2) such that $(\Delta(q_1, x), \Delta(q_2, x))$ is a distinguishable pair for any $x \in \Sigma$

	0	1	2	3	4
1	●				
2	●	●			
3		●	●		
4	●	●	●	●	
5	●	●	●	●	

- Merge indistinguishable states
- The algorithm can be improved by choosing which cell to visit carefully

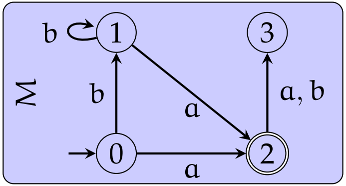
Brzozowski's algorithm

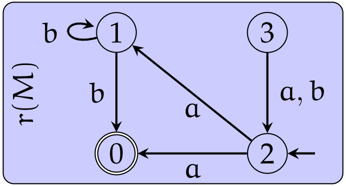
double reverse (r), determinize (d)



Brzozowski's algorithm

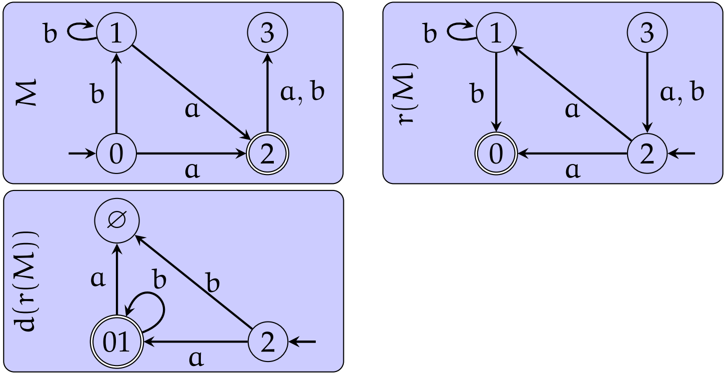
double reverse (r), determinize (d)





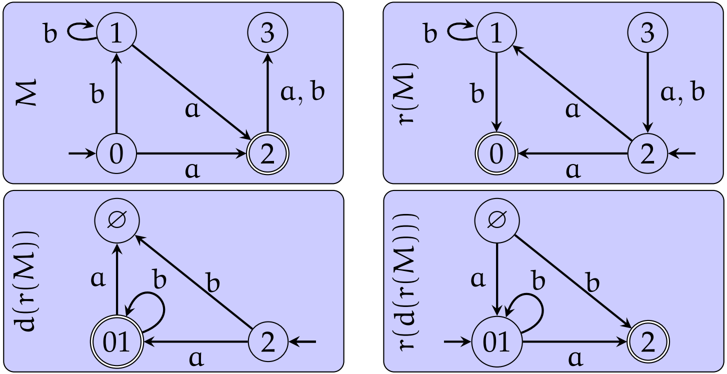
Brzozowski's algorithm

double reverse (r), determinize (d)



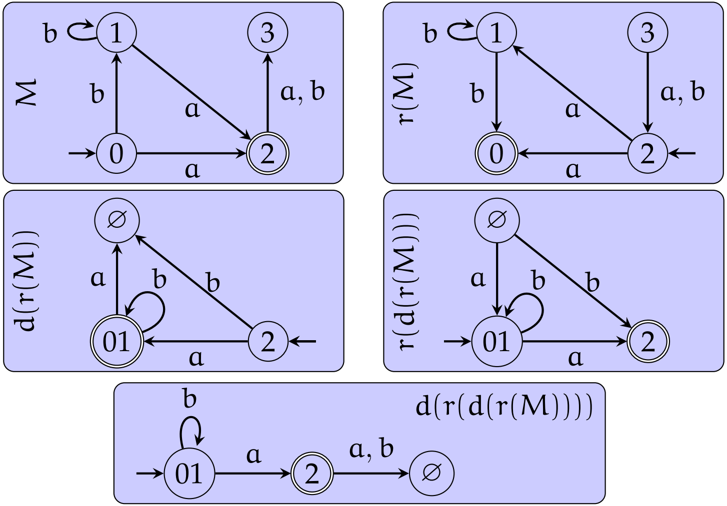
Brzozowski's algorithm

double reverse (r), determinize (d)



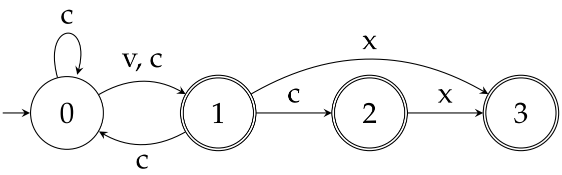
Brzozowski's algorithm

double reverse (r), determinize (d)



An exercise

find the minimum DFA for the automaton below



Minimization algorithms

final remarks

- There are many versions of the 'partitioning' algorithm. General idea is to form equivalence classes based on *right-language* of each state.
- Partitioning algorithm has $O(n \log n)$ complexity
- 'Double reversal' algorithm has exponential worst-time complexity
- Double reversal algorithm can also be used with NFAs (resulting in the minimal equivalent DFA – NFA minimization is intractable)
- In practice, there is no clear winner, different algorithms run faster on different input
- Reading suggestion: Hopcroft and Ullman (1979, Ch. 2&3), Jurafsky and Martin (2009, Ch. 2)

Minimization algorithms

final remarks

- There are many versions of the 'partitioning' algorithm. General idea is to form equivalence classes based on *right-language* of each state.
- Partitioning algorithm has $O(n \log n)$ complexity
- 'Double reversal' algorithm has exponential worst-time complexity
- Double reversal algorithm can also be used with NFAs (resulting in the minimal equivalent DFA – NFA minimization is intractable)
- In practice, there is no clear winner, different algorithms run faster on different input
- Reading suggestion: Hopcroft and Ullman (1979, Ch. 2&3), Jurafsky and Martin (2009, Ch. 2)

Next:

- FST
- FSA and regular languages

Acknowledgments, credits, references

Hopcroft, John E. and Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley. ISBN: 9780201029888.

Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. second edition. Pearson Prentice Hall. ISBN: 978-0-13-504196-3.

blank

blank

blank

blank