Linear-time minimization of Wheeler DFAs

Jarno Alanko [1, 2]   Nicola Cotumaccio [3]   Nicola Prezza [4]

[1] University of Helsinki, Finland

[2] Dalhousie University, Halifax, Canada

[3] GSSI, L'Aquila, Italy

[4] Ca' Foscari University, Venice, Italy

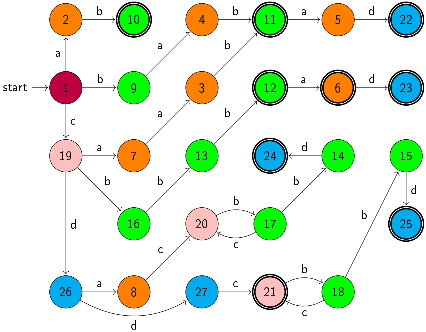
Wheeler automata

Wheeler automata generalize the nice properties of De Bruijn graphs.

- A Wheeler automaton on the alphabet Σ with n states and e edges can be stored using only $2(e+n)+n+e\log|\Sigma|+|\Sigma|\log e+o(n+e\log|\Sigma|)$ bits.
- This representation allows to decide whether a string α matches the automaton in only $O(|\alpha|\log|\Sigma|)$ time.

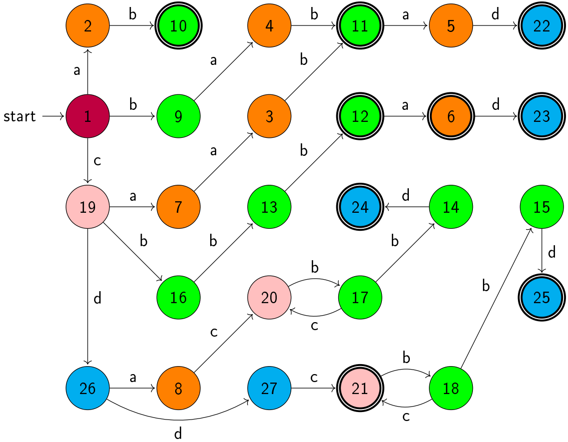
Wheeler automata

- Wheeler automata are automata endowed with a total order $\leq$ on the set of all states.
- We assume that all edges entering the same state have the same label.
- Here are the properties that the total order $\leq$ must satisfy.



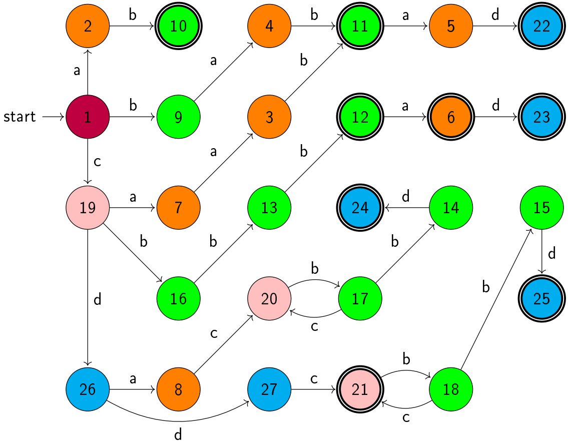
Wheeler automata

The initial state is the first state.



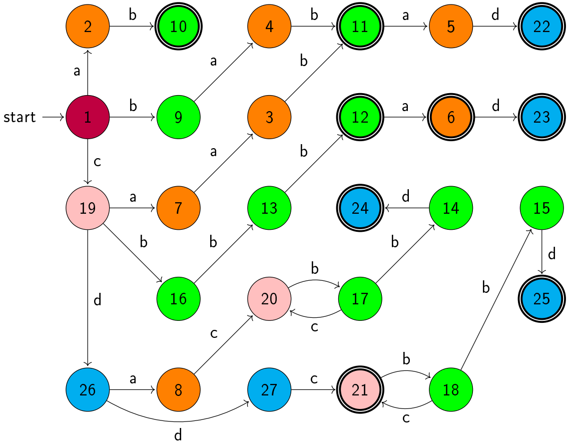
Wheeler automata

All states reached by a come before all states reached by b, which come before all states reached by c...



Wheeler automata

Equally-labeled edges must respect the total order (think of $(7, 3, a)$, $(9, 4, a)$, $(6, 23, d)$, $(15, 25, d)$).



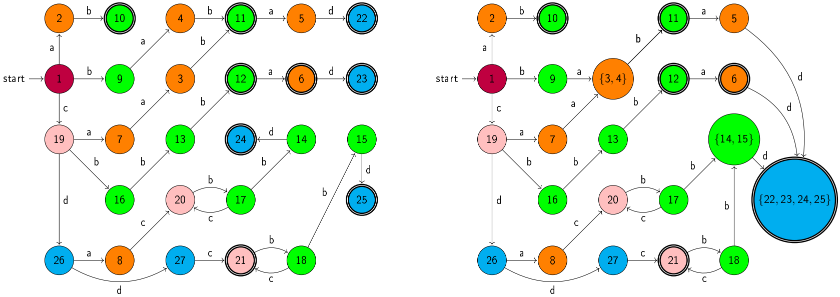
Minimization

- We know that if a language is recognized by some deterministic automaton, then there exists exactly one deterministic automaton recognizing the language and having the minimum number of states (the *minimum automaton*).
- It can be proved that the same holds true in the Wheeler case: if a language is recognized by some deterministic Wheeler automaton, then there exists exactly one deterministic Wheeler automaton recognizing the language and having the minimum number of states (the *minimum Wheeler automaton*).

Minimization

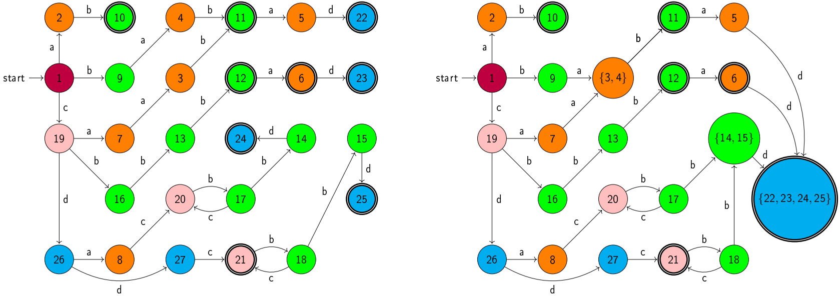
- Minimizing a deterministic Wheeler automaton (that is, building the minimum Wheeler automaton starting from a given Wheeler automaton) allows to retain the same information using less space and without affecting the running time of pattern matching queries.
- It was known how to minimize a deterministic Wheeler automaton in $O(n \log n)$ time.
- In our paper, we prove that minimization can be performed in linear time.

Minimization: an example



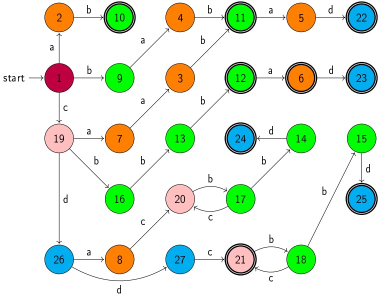
Minimization: an example

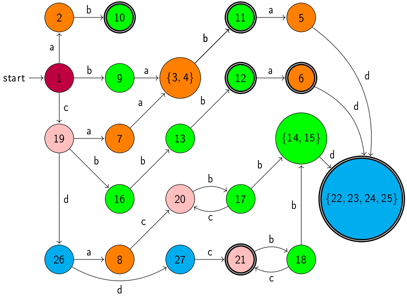
Only consecutive states are collapsed.



Minimization: an example

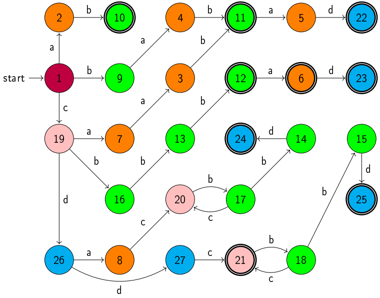
Only states reached by the same label are collapsed.

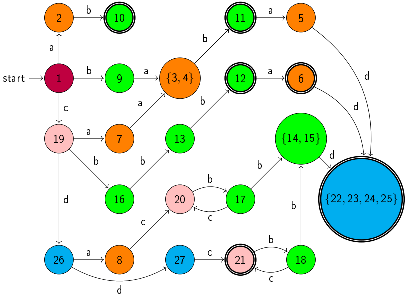




Minimization: an example

Only Nerode-equivalent states are collapsed.



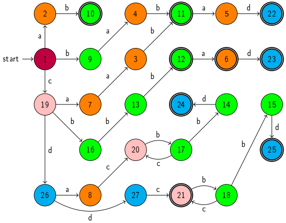


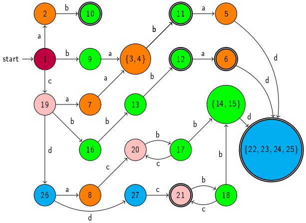
Minimization

- It can be proved that this all we have to do to obtain the minimum Wheeler automaton: collapsing the maximal runs of consecutive states reached by the same label and being Nerode equivalent.
- Since Nerode-equivalent states can be computed in $O(n \log n)$ time using Hopcroft's algorithm, we conclude that we can minimize a deterministic Wheeler automaton in $O(n \log n)$ time.

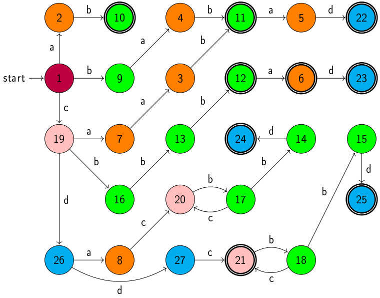
Minimization

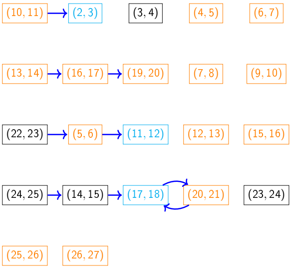
- We want to develop a linear time algorithm.
- Notice that the bottleneck of the $O(n \log n)$ time algorithm is Hopcroft's algorithm.
- However, since we know that equivalent states must be consecutive in the linear order, it will suffice to determine all pairs of consecutive elements which are NOT equivalent (for example $(15, 16)$ is a pair of non-equivalent states, while $(23, 24)$ is a pair of equivalent states).





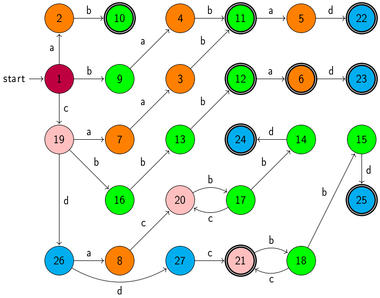
Pairs of consecutive elements reached by distinct labels (for example $(8,9)$) are non-equivalent.

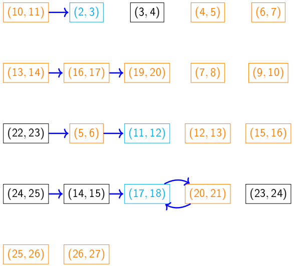




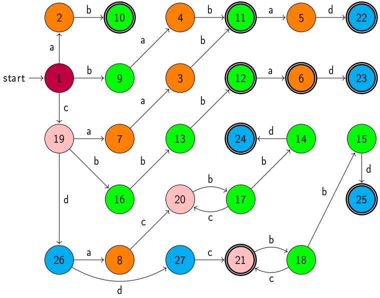
Minimization

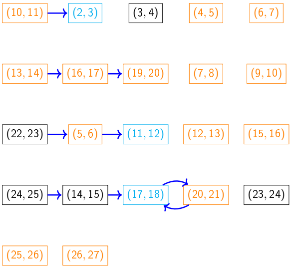
We create a graph with all pairs of consecutive elements reached by the same label.





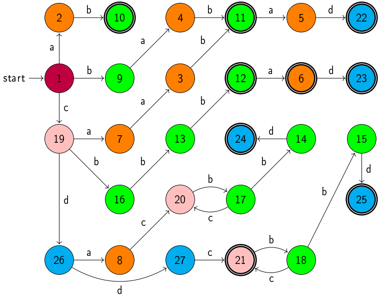
Now we only have to check which pairs are Nerode equivalent.

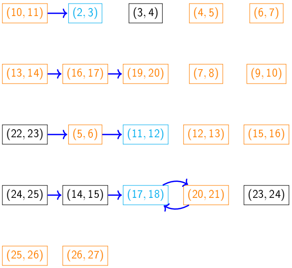




Orange pairs

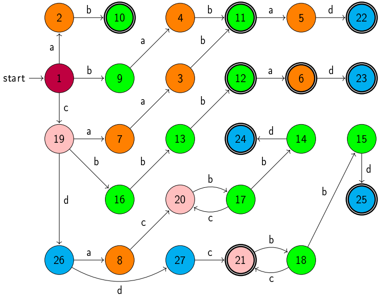
Pairs such that one state if final and the other is non-final (for example $(5, 6)$) are non-equivalent and marked orange.

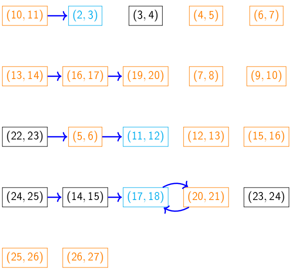




Orange pairs

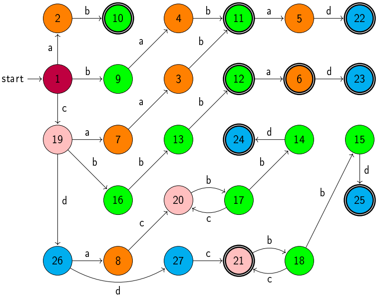
Pairs such that some label leaves one state but not the other (for example $(16, 17)$) are non-equivalent and marked orange.

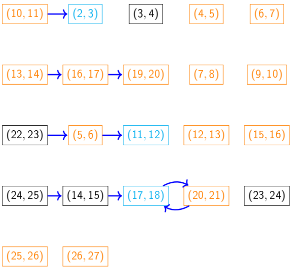




Blue pairs

- Pairs such that one can reach a pair of non-equivalent states by a common letter are also non-equivalent.
- To this end, create edges between pairs in a backward fashion (for example, $((10, 11), (2, 3))$).
- All pairs reachable by an already marked pair are marked blue.





Minimization

Non-marked pairs yield the equivalent states.

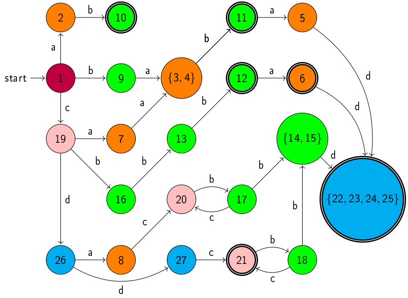
(10, 11) → (2, 3)   (3, 4)   (4, 5)   (6, 7)

(13, 14) → (16, 17) → (19, 20)   (7, 8)   (9, 10)

(22, 23) → (5, 6) → (11, 12)   (12, 13)   (15, 16)

(24, 25) → (14, 15) → (17, 18) ⇄ (20, 21)   (23, 24)

(25, 26)   (26, 27)



Correctness and running time

- The minimization algorithm is correct.
- The graph of all pairs has linear size because each pair has at most one outgoing edge.
- The graph of all pairs can be built in linear time.
- Reachability can be determined in linear time.

Experimental results

github.com/nicolaprezza/dBg-min

dataset	in ($\times 10^6$)	out ($\times 10^6$)	reduction	time (s)	nodes/s ($\times 10^6$)
cere.fasta	19.004	15.756	17.1%	17	1.118
influenza.fasta	6.469	4.792	25.9%	5	1.294
para.fasta	28.178	22.556	19.9%	26	1.084
ecoli.fastq	449.92	220.47	51%	398	1.130
human.fastq	650.51	438.68	32.6%	600	1.084
ecoli_pruned	317.173	201.940	36%	291	1.089
human_pruned	449.991	387.627	13.8%	431	1.044

Linear-time minimization of Wheeler DFAs

Jarno Alanko [1] [2]   Nicola Cotumaccio [3]   Nicola Prezza [4]

[1]University of Helsinki, Finland

[2]Dalhousie University, Halifax, Canada

[3]GSSI, L'Aquila, Italy

[4]Ca' Foscari University, Venice, Italy