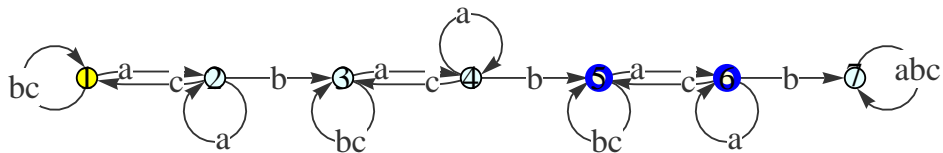


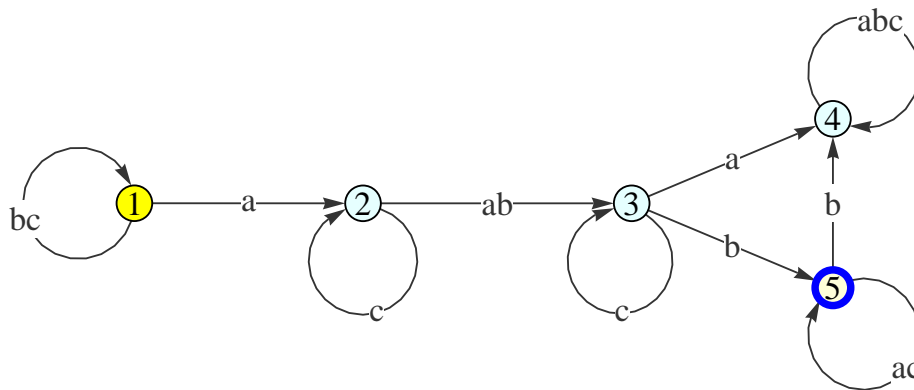
Solution: Representations of Regular Languages

Part A: Minimal Automata

To construct the minimal automaton for K note that it is easy to build the minimal DFAs for “at least two factors ab ” and “at least three factors ab ,” which can be combined to produce a machine for K . Alternatively, one can build the machine by brute thinking, building a DFA with backbone $ababab$ and figuring out the right back-transitions. Here is the result:



For L note that letter c is irrelevant (produces self-loops). To get two subwords ab we need a factor aab or abb , perhaps preceded by b 's and followed by a 's.



Here 1 is initial and 5 is final.

Part B: Regular Expressions

Since the minimal DFA for K has 7 states it is best to avoid Kleene’s algorithm or the equation method. Instead, one should read off the regular expression from the diagram of the automaton – though things are messy enough to warrant some verification (convert the expression and check the resulting automaton of equivalence with the one from above).

$$(b + a^*c)^*a^+b(b + a^*c)^*a^+b(b + a^*c)^*a^*$$

L is much easier:

$$(b + c)^*ac^*(a + b)c^*b(a + c)^*$$

Part C: MSO

To keep notation manageable, recall that successors are definable in MSO and write

$$\varphi(i, j) \equiv j = i + 1 \wedge Q_a(i) \wedge Q_b(j)$$

Then the formula for K is

$$\Phi \equiv \exists i, j, k, l (\varphi(i, j) \wedge \varphi(k, l) \wedge \forall s, t (\varphi(s, t) \rightarrow (s = i \wedge t = j) \vee (s = k \wedge t = l)))$$

For L simply change φ to

$$i < j \wedge Q_a(i) \wedge Q_b(j)$$

and add the following conditions to Φ :

$$\dots \wedge \neg(i = k \wedge j = l)$$

Solution: Primitivity and Minimality

Part A: Commutation

Induction on the length of uv . Assume without loss of generality that $|u| < |v|$. Then since $uv = vu$ we must have $uv_0 = v = v_0u$ for some non-empty word v_0 . By IH u and v_0 have the same root, which must also be the root of v .

Part B: Minimality

The diagram of $M(u)$ is just a cycle $\{0, 1, \dots, n\}$. Think of $u \in \mathbf{2}^m$ as a circular word and write ρ for the operation of rotating this word by one place counterclockwise. In other words, $\rho(u) = u_1 \dots u_{n-1}u_0$. But then the behavior of state p , $0 \leq p < n$, is just $\mathcal{L}(M(\rho^p(u)))$. Since primitivity is invariant under rotation there are two distinct states with the same behavior if, and only if, there is some $0 \leq p < n$ such that $L = \mathcal{L}(M(u)) = \mathcal{L}(M(\rho^p(u)))$.

Assume such a p exists. Then $a^i \in \mathcal{L}(M(\rho^p(u)))$ if, and only if, $(\rho^p(u))_{i \bmod n} = 1$, for all $i \geq 0$. It follows that $u = \rho^p(u)$. But then $u = xy = yx$ where x is the prefix of length p and u is not primitive.

The opposite direction is entirely similar.

Part C: Algorithmic Argument

The standard forward state-merging algorithm originally distinguishes only p such that $u_p = 0$ and p such that $u_p = 1$. After one round two states p and q are equivalent if $u_p u_{p+1} = u_q u_{q+1}$ (where indices are supposed to be taken mod n), and so forth. If p and q are still equivalent after k rounds then at least $u_p u_{p+1} \dots u_{p+k-1} = u_q u_{q+1} \dots u_{q+k-1}$.

Now, if $u = v^k$ the algorithm cannot distinguish between states p and $p + k$ and will produce a machine of at most $|v| = n/k$ states.

On the other hand, if u is primitive, then for every p and q there exists some i such that $u_{p+k} \neq u_{q+k}$ and the algorithm will place u_p and u_q into separate classes after at most k rounds.

Solution: Acceptance for Büchi Automata

Part A: Constant

For input $U = a^\omega$ erase all transitions from \mathcal{A} not labeled a . The remaining automaton accepts a^ω if there is a path from some initial state to a non-trivial (containing at least one edge) strongly connected component that contains a final state.

Part B: Periodic

Let $m = |u| > 0$. \mathcal{A} accepts U iff there is a run $\pi : p_0, p_1, \dots, p_n, \dots$ and some final state p such that for an infinite set $I \subseteq \mathbb{N}$ we have $p_i = p$. We may safely assume that $\forall i \in I (r = i \bmod m)$ for some modulus $0 \leq r < m$. Now write $u = u_0 u_1$ where $u_0, u_1 \in \Sigma^*$ and $|u_0| = r$. Then

$$\mathcal{L}(\mathcal{A}(I, p)) \cap u^* u_0 \neq \emptyset \text{ and } \mathcal{L}(\mathcal{A}(p, p)) \cap u_1 u^* u_0 \neq \emptyset$$

A moments thought shows that \mathcal{A} accepts U if, and only if, this condition holds for some $0 \leq r < m$. But the condition can easily be checked using ordinary FSM methods.

Part C: Ultimately Periodic

This is essentially the same as part (B), we only need to replace the initial segment $u^k u_0$ by $vu^k u_0$.

Part D: Running Time

Let n be the size of \mathcal{A} .

Part (A) is linear in n : use Tarjan's algorithm to determine the strongly connected components of the diagram of the residual automaton. Find all non-trivial SCCs that contain a final state and check that at least one of them is reachable from an initial state.

Part (B) requires up to $2m$ emptiness tests on ordinary NFAs of size $O(nm)$ where $m = |u|$. To see this note that $\mathcal{A}(I, p)$ and $\mathcal{A}(p, p)$ have size n and the canonical automaton for $u_1 u^* u_0$ has size $m + 1$. The standard product construction then produces a machine of size $O(nm)$ for the intersections. Hence the total running time is $O(nm^2)$.

Part (C) requires an additional $O(n|v|)$ steps to deal with the prefix v .