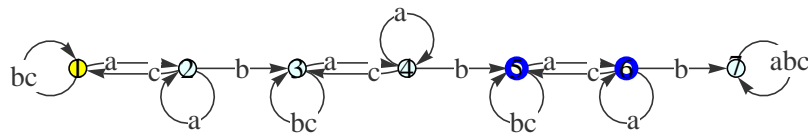


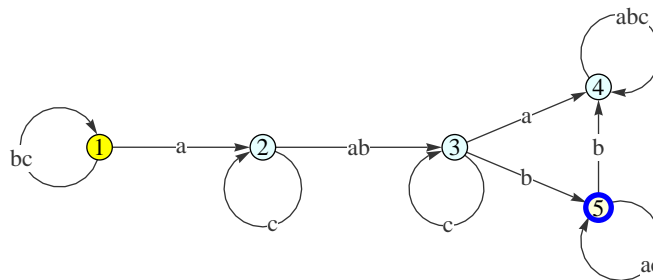
Solution: Representations of Regular Languages

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: 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 (meaning: containing at least one edge) strongly connected component that contains a final state.

Part B: Periodic

Construct a digraph G whose vertices are the states of \mathcal{A} and with edges $p \rightarrow q$ whenever there is a path from p to q labeled u in \mathcal{A} . G can be constructed by multiplying matrices over the language semiring or by running a form of BFS on all the states of \mathcal{A} .

But then \mathcal{A} accepts $U = u^\omega$ iff there is a path in G that starts in I and touches F infinitely often. Equivalently, there has to be a path in G starting at I that leads to a non-trivial strongly connected component containing a final state.

Part C: Ultimately Periodic

This is essentially the same as part (B), we only need to replace the set I of initial states by $\tau(I, v)$.

Part D: Running Time

Let n be the number of states in \mathcal{A} , t the number of transitions and m the length of u .

Part (A) is linear in $n + t$: 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), the computation of G can be handled in time $O((n + t)m)$. Thus we can use Tarjan's algorithm to find the SCCs in time $O((n + t)m)$ and decide the appropriate path-existence question in the same time.

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

Solution: Floyd goes Algebraic

Part A: Idempotents

Consider the set of all powers $A = \{a^i \mid i \geq 0\}$. Here we tacitly assume that the semigroup is actually a monoid and that $a^0 = 1$ is the identity element. If the semigroup has no identity we simply adjoin one.

We are looking for a power a^r , $r > 0$, of a with the property $a^{2r} = a^r$. But this is the same situation as in Floyd's cycle detection algorithm: one particle moves down the lasso at speed 2, the other at speed 1. When they meet we have the desired idempotent.

Part B: Plot

The periods p for which the Floyd time is equal to t clearly must be divisors of t . For $p \geq t$ we obtain the rightmost line with slope 1. The next line to the left has slope 2 and starts at $p = 35$, and so forth.

Part C: Exponents

We need to determine $r \geq 1$ minimal such that $a^{2r} = a^r$. Let $r = t + e$ where $t = t(a)$ and $0 \leq e < p = p(a)$. Floyd's algorithm finds the minimal e such that

$$(2e + t) \bmod p = e.$$

It is easy to see that the solution has the form

$$e = -t \bmod p$$

so that $r = t + (-t \bmod p)$. Note that we assume $0 \leq x \bmod p < p$; some implementations of the mod function may return negative values.

Part D: Group

Let $G = \{a^{t+i} \mid 0 \leq i < p\}$ be the collection of powers of a that lie on the loop. Consider the map $f : \mathbb{Z}_p \rightarrow G$, $f(i) = a^{t+i}$. f is well-defined and bijective since p is the period of a .

We claim that f is a semigroup homomorphism. To see this compute

$$f(i+j) = a^{t+i+j} = a^t a^{i+j} = a^{2t} a^{i+j} = a^{2r+i+j} = a^{r+i} a^{r+j} = f(i)f(j).$$

So f is a semigroup isomorphism. Since \mathbb{Z}_p is actually a group we're done.