MINIMUM MEMOIZATION SETS VERSUS MFN MEMOIZATION: TWO CONJECTURES REFUTED, WITH AN UNBOUNDED GAP

KOREA SUPERINTELLIGENCE LABS

Abstract. A backtracking regular expression matcher runs in linear time exactly when the underlying memoized automaton has no infinite degree of ambiguity (IDA), and a set of states whose memoization removes IDA is a *memoization set*. Berglund, van der Merwe and le Roux (NCMA 2026) propose memoizing a minimum feedback vertex set of the automaton, computed in one pass over the parse tree; they call the resulting set MFN, and they conjecture (Conjecture 1) that some minimum memoization set always lies inside MFN, equivalently (Conjecture 2) that the greedy refinement IAR^+ of MFN reaches a minimum memoization set for some order of the states. We show that both conjectures are false for the Glushkov construction. The regular expression $R_0 = ((a \mid a^+)\, b)^*$, with seven parse-tree nodes, has minimum memoization number 1, attained by the single state b_3, which is not a feedback vertex set, while $\mathrm{MFN}(\mathrm{Gl}(R_0))$ is a genuine minimum feedback vertex set of size 2 and no state of it can be removed without re-introducing IDA, so IAR^+ returns it for every order. The failure is not sporadic: for $R_k = ((a \mid a^+ \mid \cdots \mid a^+)\, b)^*$ with k copies of a^+ the smallest memoization set inside MFN has size $k+1$ while the minimum memoization number is 1, so restricting the search to MFN costs a factor linear in the size of the expression. An exhaustive enumeration over the two-letter alphabet shows that seven is the least size at which the conjecture fails: it holds on all 10 560 expressions of size at most six and fails on exactly eight of the 56 457 of size seven, all equivalent to R_0 under obvious symmetries. We also correct the source's Example 7 and reproduce every Glushkov value it prints. All of this is machine-checked in Lean 4; a separate C program, reported as such, extends the enumeration to all 136 524 153 expressions of size at most eleven, finds the first Thompson counterexamples at size eight, and over the same range finds no violation of the source's Theorems 4 and 5 and 1 248 832 instances of the ε-loop exception to its Theorem 3.

1. Introduction

Backtracking, memoization, IDA. A backtracking matcher for a regular expression R explores the run tree of a non-deterministic finite automaton for R on the input, and on unlucky inputs that tree is exponential ("ReDoS"). Davis, Servant and Lee [3] showed that *selective memoization* — recording, for a chosen set M of states, which pairs (state, input position) have already been explored — removes the blow-up while keeping the matcher's semantics; they proposed three nested choices of M (all states, the states of in-degree greater than one, and the "cycle ancestors", the targets of back edges in a topological sort from the initial state) without asking for the smallest such set. Van der Merwe, Mouton, van Litsenborgh and Berglund [2] gave, as the source recounts it, the precise criterion: the memoized matcher runs in linear time on every input if and only if the memoized automaton has no *infinite degree of ambiguity* (IDA), a notion going back to Weber and Seidl [4]. The same paper, again as restated in the source, introduced the memoization schemes IN (states of in-degree at least two), CN (targets of back edges in depth-first search) and IAR (greedy deletion from CN), and showed that finding a minimum memoization set is NP-complete for general NFAs (its p. 5).

The source and its two conjectures. Berglund, van der Merwe and le Roux [1] propose a new scheme, *Minimum Feedback Node* (MFN): memoize a minimum feedback vertex set of the automaton, which their parse-tree algorithms compute in linear time for both the Thompson and the Glushkov construction (their Algorithms 1 and 2, with Theorems 3 and 4 asserting that the returned set is a minimum feedback vertex set — for Thompson, "when ignoring epsilon cycles"). Every cycle meets a feedback vertex set, so MFN always removes IDA. But, as the source says on its p. 10, "the true minimum memoization may be smaller if some cycles do not contribute to IDA" — a^* has a cycle and needs no memoization at all — and it leaves two conjectures, which we quote from p. 10 of the arXiv PDF:

Conjecture 1 ([1], p. 10)**.** *For every regex R and every minimum memoization set H for* $\mathrm{Th}(R)$ *(respectively* $\mathrm{Gl}(R)$*), there exists a minimum memoization set H' with $|H'| = |H|$ and $H' \subseteq \mathrm{MFN}(\mathrm{Th}(R))$ (respectively $H' \subseteq \mathrm{MFN}(\mathrm{Gl}(R))$).*

Conjecture 2 ([1], p. 10)**.** *For some order on the states,* IAR^+ *produces a minimum memoization set.*

Here IAR^+ (Definition 5 of the source) starts from MFN and deletes states greedily whenever deletion does not re-introduce IDA; the source's Observation 3 shows the two conjectures equivalent, and its Section 7 lists "proving or disproving Conjecture 1 or 2" as the first item of future work. The source supports Conjecture 1 by an exchange argument ("one can attempt to swap x for the MFN state that covers the same cycles as x") and prints no verification range: its experiments compare memory and step counts of the schemes over a corpus of practical expressions, not counterexamples.

Results. We settle both conjectures negatively for the Glushkov construction and quantify the failure.

(i) (Theorem 3.1) $R_0 = ((a \mid a^+)\, b)^*$, seven parse-tree nodes, has minimum memoization number 1, attained by the single state b_3, which is not a feedback vertex set; $\mathrm{MFN}(\mathrm{Gl}(R_0)) = \{a_1, a_2\}$ is a minimum feedback vertex set, and neither $\{a_1\}$ nor $\{a_2\}$ removes IDA. Conjecture 1 fails at R_0.

(ii) (Theorem 3.3) No state of $\mathrm{MFN}(\mathrm{Gl}(R_0))$ can be deleted without re-introducing IDA, so IAR^+ returns $\{a_1, a_2\}$ for *every* order. Conjecture 2 fails directly, not only through the equivalence.

(iii) (Theorem 4.1) For $R_k = ((a \mid a^+ \mid \cdots \mid a^+)\, b)^*$ with k copies of a^+, of size $3k + 4$, the smallest memoization set inside $\mathrm{MFN}(\mathrm{Gl}(R_k))$ is MFN itself, of size $k+1$, against a minimum memoization number of 1. The best set IAR^+ can reach is thus $\Omega(|R|)$ times the optimum.

(iv) (Theorem 5.1) Over $\{a, b\}$, Conjecture 1 for Gl holds on every one of the 10 560 expressions of size at most 6 and fails on exactly 8 of the 56 457 of size 7 — R_0 up to swapping the branches of the alternation, exchanging the letters, and $*$ versus $+$. This is the first verification range for the conjecture that we know of: the source prints none.

(v) (Proposition 6.1) The source's Example 7 misreports $\mathrm{MFN}(\mathrm{Gl}((a \mid a)^*))$; the correct value has two states, and one state could not be a feedback vertex set at all.

(vi) (Propositions 6.2 and 6.3) Every Glushkov MFN value the source prints is reproduced from the definitions, and three negative controls show the finite objects are the intended ones.

Items (i)–(vi) are machine-checked in Lean 4 from a self-contained implementation of the Glushkov construction, the IDA criterion and Algorithm 2 (Section 7); the general-k statement in (iii) is proved on paper, its cases $k \leq 4$ by machine. Two further computations, in C and outside the formal development, are reported as such in Remarks 5.3 and 6.4: the enumeration extended to all 136 524 153 expressions of size at most 11, the first Thompson counterexamples (size 8), and zero violations of the source's Theorems 4 and 5.

What is not claimed. The Glushkov automaton has no ε-transitions, so every definition below is fixed by the source without any convention of ours; the Thompson half of Conjecture 1 requires an ε-removal convention the source does not pin down, and it appears here only in Remark 5.3, under a stated convention and without formal verification. We say nothing about the complexity of computing minimum memoization sets for Thompson automata (the source's second open item), and we make no claim about the 2021 conjecture of [2], of which the source's Conclusion (p. 14) says that the non-minimality of IAR in its examples is "a finding that challenges the conjecture of [9] (at least for the Glushkov construction)": no open copy of that paper could be found (checked 3 September 2026 on arXiv, OpenAlex, OpenAIRE, Semantic Scholar, dblp, the Umeå DiVA record `diva2:1641659` and the Stellenbosch repository), so its own wording is unverified, and everything attributed to it above is the source's restatement.

2. Preliminaries

2.1. **Regular expressions.** Regular expressions over an alphabet Σ are built from ε and the letters by alternation $E \mid E'$, concatenation $E \cdot E'$ (written EE'), and the three postfix operators E^*, E^+, $E^?$, which the source treats as primitive (its Section 2), as do we. The *size* $|R|$ is the number of nodes of the parse tree; so $|((a \mid a^+)\, b)^*| = 7$. Every letter occurrence of R is numbered left to right (*linearization*), giving positions $1, \ldots, m$.

2.2. **The Glushkov automaton.** Write $\mathrm{null}(E)$ for $\varepsilon \in L(E)$, and define $\mathrm{first}(E), \mathrm{last}(E) \subseteq \{1, \dots, m\}$ and a relation $\mathrm{follow}(E)$ on positions by the usual recursion: for a position p, first = last = $\{p\}$; ε has empty first and last; alternation takes unions; for EE', first = $\mathrm{first}(E) \cup [\mathrm{null}(E)]\,\mathrm{first}(E')$, last = $\mathrm{last}(E') \cup [\mathrm{null}(E')]\,\mathrm{last}(E)$ and $\mathrm{last}(E) \times \mathrm{first}(E')$ is added to follow; E^* and E^+ keep first and last and add $\mathrm{last}(E) \times \mathrm{first}(E)$ to follow; $E^?$ changes nothing but nullability; null is computed as usual (E^* and $E^?$ are always nullable, E^+ iff E is). The *Glushkov automaton* $\mathrm{Gl}(R)$ has state set $Q = \{q_0, 1, \dots, m\}$, transitions $\delta(q_0, c) = \{p \in \mathrm{first}(R) : \mathrm{sym}(p) = c\}$ and $\delta(p, c) = \{p' \in \mathrm{follow}(p) : \mathrm{sym}(p') = c\}$, and final states $\mathrm{last}(R) \cup \{q_0 \text{ if } \mathrm{null}(R)\}$. It is ε-free, and q_0 has no incoming transition. We follow the source and name a state by its letter and position, as in a_1, a_2, b_3. The prototype implementation was validated against the source's own worked automaton for $(abb)^* \mid (bc)$ (its p. 3); the verified implementation reproduces every Glushkov value the source prints (Proposition 6.2) and agrees with two independent programs on the enumeration of Section 5.

2.3. **IDA, memoization sets, feedback vertex sets.** A *memoized NFA* is an NFA together with a set $M \subseteq Q$ of memoized states. The source's Theorem 1 (p. 4, attributed to [2]; with nothing memoized it reduces to the classical criterion of Weber and Seidl [4], as the source notes on its p. 5) is the criterion everything below rests on; we quote it verbatim, omitting only the citation marker in its heading:

> **Theorem 1** (IDA characterization). An mNFA A has IDA if and only if there exist two distinct states p and q, and a string $v \in \Sigma^+$, such that there are paths on v from p to p, from p to q, and from q to q, with the cycle on q containing no memoized state.

For an ε-free automaton "a path on v" is a path of the transition graph labelled v, and "the cycle on q contains no memoized state" means that every state on that q-to-q path, q included, lies outside M. We take the right-hand side of Theorem 1 as the *definition* of "(A, M) has IDA". By the source's Theorem 2 (p. 5, again from [2]), (A, M) has no IDA if and only if the memoized backtracking matcher has linear matching time on all inputs.

Definition 2.1. $M \subseteq Q$ is a *memoization set* for A if (A, M) has no IDA. The *minimum memoization number* $m^*(A)$ is the least cardinality of a memoization set; a memoization set of that size is *minimum* (the source's Section 4). M is a *feedback vertex set* if deleting M leaves the transition graph acyclic, and $\mathrm{mfvs}(A)$ is the least size of one.

Q itself is a memoization set (no q-cycle can avoid Q), so m^* is well defined. Every feedback vertex set is a memoization set, since a q-cycle avoiding M would be a cycle avoiding M; this is the source's remark after its Definition 4. The converse fails: $\mathrm{Gl}(a^*)$ has a cycle but $m^* = 0$.

2.4. **Algorithm 2 and** MFN**.** The source's Algorithm 2 ("Glushkov PTA", p. 8) returns, for a regex R, a triple $(R_{op}, R_{req}, \mathrm{null}(R))$ by the recursion

$$G(\varepsilon) = (\emptyset, \emptyset, \mathrm{T}), \qquad G(a_p) = (\{p\}, \emptyset, \mathrm{F}),$$
$$G(A^?) = (o, r, \mathrm{T}), \qquad G(A^*) = (\emptyset,\ o \cup r,\ \mathrm{T}), \qquad G(A^+) = (\emptyset,\ o \cup r,\ n),$$
$$G(A \mid A') = (o \cup o',\ r \cup r',\ n \vee n'),$$

where $(o, r, n) = G(A)$ and $(o', r', n') = G(A')$, and for concatenation

$$G(A \cdot A') = \begin{cases} (o \cup o',\ r \cup r',\ \mathrm{T}) & \text{if } n \wedge n', \\ (o,\ r \cup r',\ \mathrm{F}) & \text{if } n' \text{ and not } n, \\ (o',\ r \cup r',\ \mathrm{F}) & \text{if } n \text{ and not } n', \\ (o,\ r \cup r',\ \mathrm{F}) & \text{if neither and } |o| \le |o'|, \\ (o',\ r \cup r',\ \mathrm{F}) & \text{if neither and } |o| > |o'|. \end{cases}$$

This is a line-by-line transcription of the algorithm as printed (lines 1–20 on p. 8 of the arXiv PDF). Note that the smaller-of-two choice occurs only in the concatenation case; an alternation takes the union. By the source's Definition 4 (p. 6), $\mathrm{MFN}(\mathrm{Gl}(R))$ is the set R_{req} returned, and its Theorem 4 (p. 7) states that R_{req} is a minimum feedback vertex set of $\mathrm{Gl}(R)$. In particular $\mathrm{MFN}(\mathrm{Gl}(R))$ is a memoization set.

2.5. IAR^+ **and the conjectures at a fixed regex.** The source's Definition 5 (p. 10), verbatim: "The set of states to memoize in IAR^+ is computed as follows. (i) Compute $\mathrm{MFN}(A)$; and (ii) place an order on the states from (i), and in order, remove the state under consideration if, by removing them, we will not re-introduce IDA." Whatever the order, the output is a subset of $\mathrm{MFN}(A)$ that is a memoization set. Its Observation 3 (p. 10) notes that Conjecture 1 implies Conjecture 2 (order the states of MFN so that a minimum memoization set contained in it comes last) and that Conjecture 2 implies Conjecture 1 by Definition 5.

For a fixed regex R and the Glushkov construction, put

$$c(R) \;=\; \min\{\,|M| : M \subseteq \mathrm{MFN}(\mathrm{Gl}(R)),\ M \text{ a memoization set}\,\},$$

with the convention $c(R) = |\mathrm{MFN}(\mathrm{Gl}(R))|+1$ if no subset of MFN is a memoization set (this convention is what the machine computes; it never arises when MFN is a memoization set). Always $c(R) \geq m^*(\mathrm{Gl}(R))$.

Lemma 2.2. *Suppose* $\mathrm{MFN}(\mathrm{Gl}(R))$ *is a memoization set (as it is whenever it is a feedback vertex set, hence for every* R *by the source's Theorem 4). Then Conjecture 1 holds at* R *for the Glushkov construction if and only if* $c(R) = m^*(\mathrm{Gl}(R))$. *Moreover, if no single state of* $\mathrm{MFN}(\mathrm{Gl}(R))$ *can be removed without re-introducing IDA, then* IAR^+ *returns* $\mathrm{MFN}(\mathrm{Gl}(R))$ *for every order on its states.*

Proof. All minimum memoization sets have cardinality m^*, and one exists. Conjecture 1 at R asks for a memoization set of size m^* inside MFN, i.e. $c(R) \leq m^*$; with $c(R) \geq m^*$ this is $c(R) = m^*$. For the second claim, IAR^+ examines the states of MFN in the given order, and the current set is MFN at the first step; since removing any state re-introduces IDA, nothing is removed at the first step, and the same argument applies at every later step. □

Deciding whether (A, M) has IDA is reachability in the triple product: (A, M) has IDA iff for some $p \neq q$ the triple (p, q, q) is reachable from (p, p, q) when the first two components move under δ and the third under δ restricted to states outside M. Since p and q lie on cycles, and q_0 lies on none, triples may be taken over the non-initial states. With that, every quantity in this paper — MFN, $c(R)$, m^*, mfvs, and the two conjectures at a fixed R — is a finite computation.

3. The counterexample

Theorem 3.1. *Let* $R_0 = ((a \mid a^+)\, b)^*$, *linearised* $((a_1 \mid a_2^+)\, b_3)^*$. *Then*

(a) $|R_0| = 7$, *and* $\mathrm{Gl}(R_0)$ *has the three non-initial states* a_1, a_2, b_3;
(b) $\mathrm{MFN}(\mathrm{Gl}(R_0)) = \{a_1, a_2\}$; *it is a feedback vertex set, and* $\mathrm{mfvs}(\mathrm{Gl}(R_0)) = 2$, *so it is a minimum feedback vertex set exactly as the source's Theorem 4 asserts;*
(c) $\{b_3\}$ *is a memoization set, and it is not a feedback vertex set;*
(d) $\emptyset$, $\{a_1\}$ *and* $\{a_2\}$ *are not memoization sets, while* $\{a_1, a_2\}$ *is;*
(e) *consequently* $m^*(\mathrm{Gl}(R_0)) = 1$ *and* $c(R_0) = 2$.

In particular Conjecture 1 fails at R_0 *for the Glushkov construction:* $H = \{b_3\}$ *is a minimum memoization set, and no memoization set of size* 1 *is contained in* $\mathrm{MFN}(\mathrm{Gl}(R_0))$.

Proof. The automaton is small enough to write down. first $= \{a_1, a_2\}$, last $= \{b_3\}$, and follow $= \{(a_2, a_2), (a_1, b_3), (a_2, b_3), (b_3, a_1), (b_3, a_2)\}$, so

$$\delta(q_0, a) = \{a_1, a_2\}, \quad \delta(a_1, b) = \{b_3\}, \quad \delta(a_2, a) = \{a_2\}, \quad \delta(a_2, b) = \{b_3\}, \quad \delta(b_3, a) = \{a_1, a_2\},$$

all other entries empty, and $F = \{q_0, b_3\}$.

(b) Algorithm 2: $G(a_1) = (\{a_1\}, \emptyset, \mathrm{F})$, $G(a_2^+) = (\emptyset, \{a_2\}, \mathrm{F})$, so the alternation gives $(\{a_1\}, \{a_2\}, \mathrm{F})$; the concatenation with b_3 has neither side nullable and $|o| = |o'| = 1$, so it keeps $o = \{a_1\}$; the outer star returns $r = \{a_1\} \cup \{a_2\}$. The self-loop at a_2 forces a_2 into every feedback vertex set and the cycle $a_1 \to b_3 \to a_1$ then forces a_1 or b_3, so mfvs $= 2$, and $\{a_1, a_2\}$ breaks both cycles.

(c) With $M = \{b_3\}$, a q-cycle avoiding M reads only a's, and the only a-transition between non-initial states is the self-loop at a_2; so $q = a_2$ and $v = a^k$. But no state $p \neq a_2$ has a cycle on a^k: $\delta(a_1, a) = \emptyset$, from b_3 one never returns without reading b, and q_0 has no incoming transition. So the criterion of Theorem 1 cannot fire. The self-loop $a_2 \to a_2$ survives the deletion of b_3, so $\{b_3\}$ is not a feedback vertex set.

(d) With $M = \{a_1\}$ take $p = a_1$, $q = a_2$, $v = ba$: $a_1 \to b_3 \to a_1$, $a_1 \to b_3 \to a_2$, and the q-cycle $a_2 \to b_3 \to a_2$ contains no memoized state. With $M = \{a_2\}$ take $p = a_2$, $q = a_1$, $v = ba$; with $M = \emptyset$ either witness applies. $\{a_1, a_2\}$ is a feedback vertex set, hence a memoization set.

(e) follows: a memoization set of size 1 exists and $\emptyset$ is not one, so $m^* = 1$; inside $\{a_1, a_2\}$ neither singleton nor $\emptyset$ works, so $c(R_0) = 2$.

Every item is also checked by kernel evaluation of the finite objects (Section 7): `R0_size`, `R0_mfn`, `R0_mfn_is_mfvs`, `R0_b3_memo`, `R0_empty_not_memo`, `R0_mfn_singletons` and `conj1_false_glushkov`. □

Remark 3.2 (Where the exchange argument breaks). The source's exchange argument would swap $x = b_3 \notin \mathrm{MFN}$ for "the MFN state that covers the same cycles as x". There is none: b_3 lies on the cycle through a_1 and on the cycle through a_2, and no single state of MFN lies on both. What makes $\{b_3\}$ sufficient is that the one cycle it misses, the self-loop at a_2, carries no ambiguity by itself.

Theorem 3.3. *For R_0 as above, memoizing $\mathrm{MFN}(\mathrm{Gl}(R_0))\setminus\{x\}$ leaves IDA for each $x \in \mathrm{MFN}(\mathrm{Gl}(R_0)) = \{a_1, a_2\}$. Hence IAR^+ applied to $\mathrm{Gl}(R_0)$ returns $\{a_1, a_2\}$, of size 2, for every order on the states, while $m^*(\mathrm{Gl}(R_0)) = 1$. Conjecture 2 fails for $\mathrm{Gl}(R_0)$.*

Proof. The two sets $\{a_2\}$ and $\{a_1\}$ have IDA by Theorem 3.1(d); the conclusion is the second half of Lemma 2.2. The statement is checked by kernel evaluation (`conj2_false_glushkov`), which evaluates the stuck condition for both states, $|\mathrm{MFN}| = 2$ and $m^* = 1$. This refutes Conjecture 2 directly, and in the stronger form "stuck for every order"; by the source's Observation 3 it also follows from Theorem 3.1. □

Remark 3.4 (Matching time, read off directly). The following is a computation outside the formal development, made with a small Python script and reported as such. By the source's Theorem 2 the refutation can be seen without the IDA criterion: build the memoized run tree of $\mathrm{Gl}(R_0)$ on the input $(ab)^n$ (a memoized configuration is expanded at most once) and count its nodes. With nothing memoized the counts are $13, 29, 61, 253, 1021, 16381$ at $n = 2, 3, 4, 6, 8, 12$ (exponential); with $\{a_1\}$ or with $\{a_2\}$ memoized they are $12, 22, 35, 70, 117, 247, 425, 925, 1617$ at $n = 2, 3, 4, 6, 8, 12, 16, 24, 32$ (quadratic); with $\{b_3\}$ they are $4n + 1$ and with $\{a_1, a_2\} = \mathrm{MFN}$ they are $6n - 1$ (both linear). The single state b_3 buys linear matching time on half the memoization table and, on this input, a smaller run tree than MFN.

4. The gap is unbounded

For $k \geq 1$ let A_k denote the alternation $a \mid a^+ \mid \cdots \mid a^+$ with one plain a and k copies of a^+, associated to the left, and put $R_k = (A_k\, b)^*$; so $R_1 = R_0$ and $|R_k| = 3k + 4$. Linearised, the positions are a_1 (the plain a), $a_2, \ldots, a_{k+1}$ (one per a^+) and b_{k+2}.

Theorem 4.1. *For every $k \geq 1$: $|R_k| = 3k + 4$, $\mathrm{MFN}(\mathrm{Gl}(R_k)) = \{a_1, \ldots, a_{k+1}\}$ has size $k + 1$, $m^*(\mathrm{Gl}(R_k)) = 1$, and the only memoization set contained in $\mathrm{MFN}(\mathrm{Gl}(R_k))$ is $\mathrm{MFN}(\mathrm{Gl}(R_k))$ itself, so $c(R_k) = k + 1$. Hence Conjecture 1 fails at every R_k, and the smallest memoization set that IAR^+ can reach exceeds the optimum by the factor $k + 1 = (|R_k| - 1)/3$, which is $\Omega(|R_k|)$. The cases $k = 1, 2, 3, 4$ (sizes $7, 10, 13, 16$) are machine-checked; the general case is the argument below.*

Proof. Write $A = \{a_1, \ldots, a_{k+1}\}$. The construction gives $\mathrm{first}(R_k) = A$, $\mathrm{last} = \{b\}$, and $\mathrm{follow} = \{(a_i, a_i) : i \geq 2\} \cup (A \times \{b\}) \cup (\{b\} \times A)$, so $\delta(q_0, a) = A$, $\delta(a_i, a) = \{a_i\}$ for $i \geq 2$, $\delta(a_1, a) = \emptyset$, $\delta(a_i, b) = \{b\}$ for all i, and $\delta(b, a) = A$.

MFN. Algorithm 2 gives $G(a_1) = (\{a_1\}, \emptyset, \mathrm{F})$ and $G(a_i^+) = (\emptyset, \{a_i\}, \mathrm{F})$, so the alternation returns $(\{a_1\}, \{a_2, \ldots, a_{k+1}\}, \mathrm{F})$; the concatenation with b has neither side nullable and $|o| = |o'| = 1$, so it keeps $o = \{a_1\}$; the star returns $o \cup r = A$.

$\{b\}$ is a memoization set. A q-cycle avoiding b uses only a-transitions between non-initial states, which are the self-loops at a_i, $i \geq 2$; so $q = a_i$ and $v = a^m$. A second state $p \neq q$ with a path on a^m from p to p must be some a_j, $j \geq 2$, $j \neq i$; but from a_j every path on a^m stays at a_j, so there is no path on a^m from p to q. Hence no IDA. As $\emptyset$ is not a memoization set ($p = a_1$, $q = a_2$, $v = ba$), $m^* = 1$.

Nothing smaller inside MFN. Let $M \subseteq A$ omit some a_i. Choose $j \neq i$ (possible since $|A| \geq 2$) and take $p = a_j$, $q = a_i$, $v = ba$: the paths $a_j \to b \to a_j$, $a_j \to b \to a_i$ and $a_i \to b \to a_i$ exist, and the q-cycle $a_i \to b \to a_i$ avoids M because $a_i \notin M$ and $b \notin A \supseteq M$. So M is not a memoization set. A itself is a feedback vertex set (deleting it leaves q_0 and b, with no cycle), hence a memoization set, so $c(R_k) = k+1$.

The instances $k = 1, \dots, 4$ are checked by compiled evaluation (`Rfam_one`, `gap_two`, `gap_unbounded`), the last of which evaluates $(|R_k|, |\mathrm{MFN}|, m^*, c(R_k)) = (3k+4, k+1, 1, k+1)$ and the failure of Conjecture 1 for each $k \le 4$. □

The source bounds MFN only against the other schemes (CN and IN, and Thompson against Glushkov in its Theorem 5); it states no bound against the true minimum, and we know of no approximation statement for IAR^+ elsewhere. Theorem 4.1 shows there is none to be had: the MFN-restricted optimum is not within any constant factor of m^*.

5. Where the counterexamples begin

Fix $\Sigma = \{a, b\}$ and enumerate all regular expressions built from ε, a, b with $|$, concatenation, $*$, $+$, $?$, by parse-tree size.

Theorem 5.1. (a) *The numbers of expressions of sizes* $1, \dots, 7$ *are* $3, 9, 45, 243, 1431, 8829, 56457$ *(so* $10\,560$ *of size* ≤ 6 *and* $67\,017$ *of size* ≤ 7*), and each expression is filed under its own size.*

(b) $c(R) = m^*(\mathrm{Gl}(R))$ *for every one of the* $10\,560$ *expressions of size at most* 6*.*

(c) *Exactly* 8 *of the* $56\,457$ *expressions of size* 7 *have* $c(R) \ne m^*(\mathrm{Gl}(R))$*, namely* $((a \mid a^+)\, b)^*$*,* $((a^+ \mid a)\, b)^*$*, their images under* $a \leftrightarrow b$*, and the four expressions obtained by replacing the outer* $*$ *by* $+$*.*

(d) *Each of the eight has* $m^* = 1$*,* $|\mathrm{MFN}| = 2$*,* $c(R) = 2$ *and* $\mathrm{mfvs} = 2$*.*

(e) *The inequality* $0 < c(R) < |\mathrm{MFN}(\mathrm{Gl}(R))|$ *holds for* 104 *of the expressions of size* ≤ 6 *and for* 1544 *of those of size* ≤ 7*, hence for* 1440 *of size* 7*.*

Consequently, by Lemma 2.2, Conjecture 1 for the Glushkov construction holds at every expression over $\{a, b\}$ *of size at most* 6 *and fails at exactly eight of size* 7*, and seven is the least size of a counterexample over a two-letter alphabet; up to swapping the branches of the alternation, exchanging the two letters, and* $*$ *versus* $+$*, the smallest counterexample is unique, and it is* R_0*.*

Proof. This is an exhaustive computation. The enumeration is built once as a table indexed by size (an expression of size $s \ge 2$ is a postfix operator applied to one of size $s-1$, or a binary operator applied to sizes i and $s-1-i$), and for each expression the program computes $\mathrm{Gl}(R)$, $\mathrm{MFN}(\mathrm{Gl}(R))$ by Algorithm 2, m^* by testing subsets of the non-initial states in order of increasing size, $c(R)$ by testing subsets of MFN likewise, and mfvs; IDA is decided by the triple-product reachability of Section 2. All of (a)–(e) are conjuncts of one compiled evaluation (`census`), with (b) and (c) restated as `no_counterexample_below_seven` and `eight_at_size_seven`. The size counts (a) and the failure counts to size 7 agree with an independently written C program over the same enumeration, and at sizes ≤ 5 with a third implementation in Python, on all four quantities $(|\mathrm{MFN}|, \mathrm{mfvs}, c, m^*)$ per expression. For the eight expressions in (c) the equality $c = |\mathrm{MFN}|$ means MFN is itself a memoization set, so Lemma 2.2 applies to them without appeal to the source's Theorem 4; for (b) it is applied with that theorem. □

Remark 5.2 (On item (e)). Item (e) is a control on the subset search of the proof, not a necessary condition for failure: it records how often the least memoization subset of MFN is a proper non-empty subset of MFN, that is, how often IAR^+ has room to delete some, but not every, state of MFN. Conjecture 1 holds automatically when $m^* = 0$ (take $H' = \emptyset$) and when $m^* = |\mathrm{MFN}|$ (take $H' = \mathrm{MFN}$, a memoization set), so a failure requires $0 < m^*(\mathrm{Gl}(R)) < |\mathrm{MFN}(\mathrm{Gl}(R))|$ instead; the eight counterexamples of (c) all have $c(R) = |\mathrm{MFN}| = 2$ and so lie outside the count of (e).

Remark 5.3 (The enumeration to size 11, in C). The following is a computation outside the formal development, made with a C program and reported as such. Over all $136\,524\,153$ expressions of size at most 11 over $\{a, b\}$ (there are $370\,575$, $2\,482\,731$, $16\,907\,697$ and $116\,696\,133$ of sizes $8, 9, 10, 11$), the numbers of expressions at which Conjecture 1 fails are

size	6	7	8	9	10	11
Glushkov	0	8	48	520	4 200	34 928
Thompson	0	0	8	56	672	6 072
of which with an ε-loop	0	0	0	24	224	3 272

The Thompson line needs a convention. The source fixes of $\mathrm{Th}(R)$ only that every subexpression has a distinguished entry and exit state and that the back edge of a closure goes to the entry state, and on the graph with its ε-transitions the literal triple product reports IDA where there is none (in $\mathrm{Th}(a^+)$, the states $s \xrightarrow{a} t \xrightarrow{\varepsilon} s$ would be flagged). The program therefore removes ε by closing before each symbol, and requires the q-cycle to avoid M through its ε-states as well; this convention is ours. Under it the smallest Thompson counterexamples have size 8 and no ε-loop: $(a\,(b^+ \mid b^+))^*$ and $((a^+ \mid a^+)\,b)^*$ with the letter swap and $*$ versus $+$, each with $|\mathrm{MFN}(\mathrm{Th}(R))| = 2$ against $m^* = 1$. The C program agrees with the machine-checked table to size 7 and with the Python implementation to size 5; the size-11 pass took 45 minutes 42 seconds on one core.

6. The source's printed examples

6.1. **An erratum.** Example 7 of the source (p. 11 of the arXiv PDF) reads, verbatim: "For $(a \mid a)^*$: $\mathrm{Th}((a \mid a)^*)$ has one closure state (the $*$-node), giving $|\mathrm{MFN}| = |\mathrm{CN}| = 1$. For $\mathrm{Gl}((a \mid a)^*)$, both states a_1 and a_2 are reachable via the back edges, giving $|\mathrm{CN}| = 2$. By Algorithm 2, the Glushkov PTA would select $\{a_1\}$, giving $|\mathrm{MFN}| = 1$ in this case, matching Thompson."

Proposition 6.1. *The last sentence is incorrect. Algorithm 2 returns* $\mathrm{MFN}(\mathrm{Gl}((a \mid a)^*)) = \{a_1, a_2\}$*, of size* 2*; moreover* $\{a_1\}$ *is not a feedback vertex set of* $\mathrm{Gl}((a \mid a)^*)$ *at all, and* $\mathrm{mfvs}(\mathrm{Gl}((a \mid a)^*)) = 2$*, so no correct minimum-feedback-vertex-set algorithm could return a set of size* 1*.*

Proof. Algorithm 2 takes the union $o \cup o'$ at an alternation — the smaller-of-two choice is made only at a concatenation — so $G(a_1 \mid a_2) = (\{a_1, a_2\}, \emptyset, \mathrm{F})$ and the star returns $r = \{a_1, a_2\}$. In $\mathrm{Gl}((a \mid a)^*)$ the star adds last $\times$ first $= \{a_1, a_2\} \times \{a_1, a_2\}$ to follow, so a_2 carries a self-loop and deleting a_1 leaves a cycle; both self-loops must be hit, so mfvs $= 2$. Kernel evaluation: `example7_correction`. The error is internal to the source: it contradicts Algorithm 2 as printed and the source's own Theorem 4. Its Theorem 5 ($|\mathrm{MFN}(\mathrm{Th}(R))| \le |\mathrm{MFN}(\mathrm{Gl}(R))|$) and the Corollary on p. 11 are unaffected, since $1 \le 2$. (A hand check, not part of the formal development, gives $m^*(\mathrm{Gl}((a \mid a)^*)) = 2$ as well.) □

6.2. **Reproductions.** The following values are printed by the source; we re-derive them from the definitions as a check that the implementation is the source's. Positions are numbered left to right.

Proposition 6.2. (a) *(Example 2, p. 9)* $|\mathrm{MFN}(\mathrm{Gl}((a^+)^+))| = 1$.

(b) *(Examples 3 and 8, pp. 10–11) For* $r_1 = ((a\,a^? \mid a\,a^?)\,a\,a^?)^*$, $\mathrm{MFN}(\mathrm{Gl}(r_1)) = \{a_5\}$*, and* $m^*(\mathrm{Gl}(r_1)) = 1$.

(c) *(Example 4, p. 10) For* $((a\,b^? \mid c\,d^?)\,e\,f^?)^*$, $\mathrm{MFN}(\mathrm{Gl}(\cdot)) = \{e\}$ *(position* 5*), and* $m^* = 0$*: "no memoization is required at all", as the source says.*

(d) *(Example 5, p. 10) For* $((a \mid b \mid c \mid d)\,e)^*$, $\mathrm{MFN}(\mathrm{Gl}(\cdot)) = \{e\}$ *(position* 5*).*

Proof. Finite evaluation: `example2_repro` (kernel), `example3_repro`, `example4_repro`, `example5_repro` (compiled). The CN values the source prints in Examples 3 and 5, and the order-sensitivity of IAR in Example 3, were reproduced in the Python prototype only and are not part of the formal development. □

6.3. **Negative controls.**

Proposition 6.3. (a) MFN *is not always a minimum memoization set:* $\mathrm{MFN}(\mathrm{Gl}(a^*)) = \{a_1\}$ *while* $m^*(\mathrm{Gl}(a^*)) = 0$ *(the source's own example on its p. 10).*

(b) *Conjecture 1 is not false everywhere:* $c(R) = m^*(\mathrm{Gl}(R))$ *for* $R = (a \mid a)^*$ *and for* $R = ((a \mid a)\,b)^*$*, the latter with* $m^* = 1$*.*

(c) *The IDA predicate is neither constantly true nor constantly false:* $\mathrm{Gl}(a^*)$ *with nothing memoized has no IDA,* $\mathrm{Gl}((a \mid a)^*)$ *with nothing memoized has IDA.*

Proof. Kernel evaluation of `control_mfn_not_minimum` for (a), of `control_conj1_holds_somewhere` for (b), and of `control_ida_nonvacuous` for (c). Item (a) refutes the too-large claim "MFN is always minimum", item (b) the too-small claim "the conjecture fails everywhere", so the test $c(R) = m^*$ of Theorem 5.1 is not a constant. □

Remark 6.4 (The source's Theorems 3–5 over the enumeration). The following is again a computation of the C program of Remark 5.3, outside the formal development. Over the same 136 524 153 expressions: no violation of the source's Theorem 5, $|\mathrm{MFN}(\mathrm{Th}(R))| \leq |\mathrm{MFN}(\mathrm{Gl}(R))|$ (which the source proves by a sketch); no case in which the Glushkov MFN fails to be a minimum feedback vertex set (Theorem 4), the minimum feedback vertex number being recomputed independently by branch and bound on witness cycles rather than read off the algorithm; and 1 248 832 cases in which the Thompson MFN is not a minimum feedback vertex set, every one of them with an ε-loop — exactly the exception the source states on its p. 6 ("Algorithm 1 does not compute an MFVS in the presence of ε-loops"), which is thereby confirmed and shown to be non-vacuous. We know of no earlier count of how often that exception occurs.

7. Verification

Everything stated as a theorem or proposition above, except the general-k case of Theorem 4.1, is machine-checked in Lean 4 [10] in three modules that import no mathematical library. The first defines regular expressions (letters are natural numbers, so the six-letter Examples 4 and 5 can be typed as printed), the Glushkov construction with first, last, follow and nullability as bit masks, the IDA criterion of Theorem 1 by Warshall closure over the m^3 triples of non-initial states, memoization sets, m^*, $c(R)$ (`minMemoIn` over the states of MFN), feedback vertex sets and mfvs, Algorithm 2 transcribed line by line, and the two predicates `conj1G` ($c(R) = m^*$) and `iarPlusStuck`. The second module carries Theorems 3.1, 3.3, the instances $k \leq 4$ of Theorem 4.1, and Propositions 6.1–6.3: fourteen statements are proved by kernel evaluation (`decide`), namely everything with at most three Glushkov positions — the eight statements of the refutation, `Rfam_one`, the erratum, Example 2 and the three controls — and five by compiled evaluation (`native_decide`), namely `gap_two`, `gap_unbounded` and the reproductions of Examples 3, 4 and 5. The third module carries Theorem 5.1 as one compiled evaluation over the 67 017 expressions of size at most 7, plus two corollaries proved from it by rewriting. Appendix A lists every statement verbatim with the axioms Lean reports for it: the refutation and the erratum depend on `propext` only; the compiled evaluations add one evaluation axiom each, and the census also `Quot.sound` (through the monadic enumeration, not through the mathematics). There is no `sorry` and no axiom of our own. Reported build times on one core were 5 seconds, 1 minute 45 seconds and 3 minutes 11 seconds for the three modules. The C and Python programs of Remarks 3.4, 5.3 and 6.4 are independent re-implementations of the same definitions and are not part of the verified development. The repository is private; repository reference to be added at submission.

Appendix A. The verified statements

The declarations below are quoted verbatim from the development, with the defining constants they use. `mfnG r` is $\mathrm{MFN}(\mathrm{Gl}(r))$ as a bit mask over positions (position p is bit p, so $6 = \{1, 2\}$ and $8 = \{3\}$), `(glushkov r).minMemo` is m^*, `(glushkov r).minMemoIn (bitsOf (mfnG r))` is $c(r)$, `isMemo M` is "M is a memoization set", `isFVS` and `mfvs` are as named, `popCnt` is cardinality, and `Rx.size` is parse-tree size; `la`, `lb`, …, `lf` are the letters $a, \ldots, f$ as `.sym 0`, …, `.sym 5`.

Definitions.

```
def conj1G (r : Rx) : Bool :=
  let A := glushkov r
  A.minMemoIn (bitsOf (mfnG r)) == A.minMemo

def iarPlusStuck (r : Rx) : Bool :=
  let A := glushkov r
  let mfn := mfnG r
  (bitsOf mfn).all fun x => A.hasIDA (mfn ^^^ (1 <<< x))

def R0 : Rx := .star (.cat (.alt la (.plus la)) lb)

def famAlt : Nat → Rx
  | 0     => la
  | k + 1 => .alt (famAlt k) (.plus la)

def Rfam (k : Nat) : Rx := .star (.cat (famAlt k) lb)

def Raa : Rx := .star (.alt la la)
```

Theorem 3.1.

```
theorem R0_size : R0.size = 7 := by decide

theorem R0_mfn : mfnG R0 = 6 ∧ (glushkov R0).m = 3 := by decide

theorem R0_mfn_is_mfvs :
    (glushkov R0).isFVS (mfnG R0) = true ∧ (glushkov R0).mfvs = 2 := by decide

theorem R0_b3_memo :
    (glushkov R0).isMemo 8 = true ∧ (glushkov R0).isFVS 8 = false := by decide

theorem R0_empty_not_memo : (glushkov R0).isMemo 0 = false := by decide

theorem R0_mfn_singletons :
    (glushkov R0).isMemo 2 = false ∧ (glushkov R0).isMemo 4 = false ∧
    (glushkov R0).isMemo 6 = true := by decide

theorem conj1_false_glushkov :
    (glushkov R0).minMemo = 1 ∧
    (glushkov R0).minMemoIn (bitsOf (mfnG R0)) = 2 ∧
    conj1G R0 = false := by decide
```

Axioms: `R0_size` none; each of the others `propext`.

Theorem 3.3.

```
theorem conj2_false_glushkov :
    iarPlusStuck R0 = true ∧ popCnt (mfnG R0) = 2 ∧ (glushkov R0).minMemo = 1 := by decide
```

Axioms: `propext`.

Theorem 4.1, cases $k \leq 4$.

```
theorem Rfam_one : Rfam 1 = R0 := by decide

theorem gap_two :
    (Rfam 2).size = 10 ∧ popCnt (mfnG (Rfam 2)) = 3 ∧
    (glushkov (Rfam 2)).minMemo = 1 ∧
    (glushkov (Rfam 2)).minMemoIn (bitsOf (mfnG (Rfam 2))) = 3 := by native_decide

theorem gap_unbounded :
    ((List.range 4).map fun j =>
        let k := j + 1
        ((Rfam k).size, popCnt (mfnG (Rfam k)), (glushkov (Rfam k)).minMemo,
         (glushkov (Rfam k)).minMemoIn (bitsOf (mfnG (Rfam k))), conj1G (Rfam k)))
      = [(7, 2, 1, 2, false), (10, 3, 1, 3, false),
         (13, 4, 1, 4, false), (16, 5, 1, 5, false)] := by native_decide
```

Axioms: `Rfam_one` none; `gap_two` and `gap_unbounded` `propext` and one `native_decide` evaluation axiom each.

Proposition 6.1.

```
theorem example7_correction :
    mfnG Raa = 6 ∧ popCnt (mfnG Raa) = 2 ∧
    (glushkov Raa).isFVS 2 = false ∧ (glushkov Raa).mfvs = 2 := by decide
```

Axioms: `propext`.

Proposition 6.2.

```
theorem example2_repro : popCnt (mfnG (.plus (.plus la))) = 1 := by decide

theorem example3_repro :
    bitsOf (mfnG (.star (.cat (.alt (.cat la (.opt la)) (.cat la (.opt la))) (.cat la (.opt la)))))
      = [5] ∧
    (glushkov (.star (.cat (.alt (.cat la (.opt la)) (.cat la (.opt la)))
      (.cat la (.opt la))))).minMemo = 1 := by native_decide

theorem example4_repro :
    bitsOf (mfnG (.star (.cat (.alt (.cat la (.opt lb)) (.cat lc (.opt ld))) (.cat le (.opt lf)))))
      = [5] ∧
    (glushkov (.star (.cat (.alt (.cat la (.opt lb)) (.cat lc (.opt ld)))
      (.cat le (.opt lf))))).minMemo = 0 := by native_decide

theorem example5_repro :
    bitsOf (mfnG (.star (.cat (.alt (.alt (.alt la lb) lc) ld) le))) = [5] := by native_decide
```

Axioms: `example2_repro` `propext`; the other three `propext` and one evaluation axiom each.

Proposition 6.3.

```
theorem control_mfn_not_minimum :
    mfnG (.star la) = 2 ∧ (glushkov (.star la)).minMemo = 0 := by decide

theorem control_conj1_holds_somewhere :
    conj1G Raa = true ∧ conj1G (.star (.cat (.alt la la) lb)) = true ∧
    (glushkov (.star (.cat (.alt la la) lb))).minMemo = 1 := by decide

theorem control_ida_nonvacuous :
    (glushkov (.star la)).hasIDA 0 = false ∧ (glushkov Raa).hasIDA 0 = true := by decide
```

Axioms: `propext` each.

Theorem 5.1. Here `tab7` is the table of all expressions of size at most 7 over letters `0,1`; `sizeCounts` and `failCounts` list, by size, the number of expressions and the number with `conj1G r = false`; `smallest` is the list of size-7 failures; and `contentful N` counts the expressions of size $\leq N$ with $0 < c(r) < |\mathrm{MFN}|$.

```
theorem census :
    sizeCounts = [3, 9, 45, 243, 1431, 8829, 56457] ∧
    ((List.range' 1 7).all fun s => (tab7[s]!).all fun r => r.size == s) = true ∧
    failCounts = [0, 0, 0, 0, 0, 0, 8] ∧
    contentful 6 = 104 ∧
    contentful 7 = 1544 ∧
    (smallest.all fun r =>
      (glushkov r).minMemo == 1 && popCnt (mfnG r) == 2 &&
      (glushkov r).minMemoIn (bitsOf (mfnG r)) == 2 && (glushkov r).mfvs == 2) = true ∧
    smallest =
      [ .star (.cat (.alt (.sym 0) (.plus (.sym 0))) (.sym 1)),
        .plus (.cat (.alt (.sym 0) (.plus (.sym 0))) (.sym 1)),
        .star (.cat (.alt (.sym 1) (.plus (.sym 1))) (.sym 0)),
        .plus (.cat (.alt (.sym 1) (.plus (.sym 1))) (.sym 0)),
        .star (.cat (.alt (.plus (.sym 0)) (.sym 0)) (.sym 1)),
        .plus (.cat (.alt (.plus (.sym 0)) (.sym 0)) (.sym 1)),
        .star (.cat (.alt (.plus (.sym 1)) (.sym 1)) (.sym 0)),
        .plus (.cat (.alt (.plus (.sym 1)) (.sym 1)) (.sym 0)) ] := by native_decide

theorem no_counterexample_below_seven : failCounts.take 6 = [0, 0, 0, 0, 0, 0] := by
  rw [census.2.2.1]; rfl

theorem eight_at_size_seven : failCounts.getLast! = 8 := by rw [census.2.2.1]; rfl
```

Axioms: `propext`, `Quot.sound`, and the one evaluation axiom of `census`, for all three.

References

[1] M. Berglund, B. van der Merwe and I. le Roux, *Selective memoization for efficient backtracking regular expression matching*, in: C. Câmpeanu, M. Kutrib and S. Lombardy (eds.), Proceedings of the 16th International Workshop on Non-Classical Models of Automata and Applications (NCMA 2026), Rouen, 29–30 June 2026, Electronic Proceedings in Theoretical Computer Science **446** (2026), 1–17; doi:10.4204/EPTCS.446.1; arXiv:2606.26678v1 (cs.FL), submitted 25 June 2026, the only version at the time of writing. The authors are listed alphabetically and the paper names I. le Roux as lead author. All page and statement numbers cited above are those of the arXiv PDF of v1; the numbering was cross-checked against a compiled copy of the e-print, which paginates differently.

[2] B. van der Merwe, J. Mouton, S. van Litsenborgh and M. Berglund, *Memoized regular expressions*, in: S. Maneth (ed.), Implementation and Application of Automata — 25th International Conference, CIAA 2021, Virtual Event, July 19–22, 2021, Proceedings, Lecture Notes in Computer Science **12803**, Springer, 2021, pp. 39–52; doi:10.1007/978-3-030-79121-6_4.

[3] J. C. Davis, F. Servant and D. Lee, *Using selective memoization to defeat regular expression denial of service (ReDoS)*, in: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, 2021, pp. 1–17; doi:10.1109/SP40001.2021.00032.

[4] A. Weber and H. Seidl, *On the degree of ambiguity of finite automata*, Theoret. Comput. Sci. **88** (1991), no. 2, 325–349; doi:10.1016/0304-3975(91)90381-B.

[5] D. Giammarresi, J.-L. Ponty, D. Wood and D. Ziadi, *A characterization of Thompson digraphs*, Discrete Appl. Math. **134** (2004), no. 1–3, 317–337; doi:10.1016/S0166-218X(03)00299-3. Cited by the source for the reducibility of Thompson automata.

[6] A. Shamir, *A linear time algorithm for finding minimum cutsets in reducible graphs*, SIAM J. Comput. **8** (1979), no. 4, 645–655; doi:10.1137/0208051. Cited by the source for linear-time minimum feedback vertex sets in reducible graphs.

[7] K. Thompson, *Regular expression search algorithm*, Comm. ACM **11** (1968), no. 6, 419–422; doi:10.1145/363347.363387.

[8] V. M. Glushkov, *The abstract theory of automata*, Russian Math. Surveys **16** (1961), no. 5, 1–53; doi:10.1070/RM1961v016n05ABEH004112 (translation of Uspekhi Mat. Nauk **16** (1961), no. 5(101), 3–62).

[9] B. K. M. Watling, *BRU (Brendan's Regex Utility)*, the matcher on which the source's experiments run, https://github.com/bkmwatling/srvm (redirecting to bkmwatling/BRU), GPL-3.0, inspected 3 September 2026: four branches and one tag (ciaa2024), 286 commits, last push 24 September 2025, one fork (michaeldwalsh/BRU, six further commits on lookaheads, to 11 May 2026). The public code carries Thompson and Glushkov constructions (src/fa/constructions), a memoisation

transformer, and the command-line option `--memo-scheme none | cn | in | iar`; in the entire history of both repositories the strings `feedback`, `MFN` and `PTA` occur only inside benchmark data files, so the PTA algorithms that the source's p. 11 says "have been implemented in BRU" are not in the public repository, which also holds no counterexample search.

[10] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in: A. Platzer and G. Sutcliffe (eds.), Automated Deduction — CADE 28, Lecture Notes in Computer Science **12699**, Springer, Cham, 2021, pp. 625–635; doi:10.1007/978-3-030-79876-5_37.