COMMIT NOW, PAY LATER: WHEN CAN A STREAMING PROGRAM KEEP TRACK OF WHERE ITS OUTPUT CAME FROM?

HANYU YANG

Abstract. Perfect offline, undecidable online. A streaming transformation can produce the right output and still lose track of where each letter came from. We ask whether a nondeterministic copyless streaming string transducer can be steered as its input arrives, with finite memory and bounded lookahead, so that every output letter stays close to the input position a deterministic reference assigns it. Bounded ambiguity is the dividing line. If each input has at most K accepting runs, the problem is decidable and we construct the controller. With linearly many runs it is RE-complete, although every input has a perfectly aligned run and the reference is fixed. The only difficulty is timing: a choice must be made before the input that decides it is visible. Without registers, any ambiguity stays decidable.

1. Introduction

We prove a decidability frontier for online control of copyless streaming string transducers. Given a nondeterministic transducer B and a deterministic reference A, can we resolve B with finite memory and bounded lookahead while keeping corresponding output origins uniformly close? To our knowledge, this ambiguity frontier is new. A constant bound on accepting runs makes the question decidable. Linear ambiguity already makes it RE-complete.

An origin records the input position where an output letter enters the computation. For two runs producing the same word, od is the largest difference between origins at corresponding output ranks. The controller chooses an existing edge of B at each input position. It sees a bounded window of input, including $\dashv$ when the window extends past the word. It cannot inspect register contents. We write $\mathrm{GOOD}(d, H)$ when lookahead d suffices to keep od $\leq H$ uniformly over the required inputs. Then Q_d asks whether some H suffices, and J allows both bounds to vary.

The question arises when several executions implement a transformation but their associations between input and output differ. Registers can hold letters long after their creation. Emission time loses that information. The reference fixes the intended origins, which the word function alone leaves

Date: 4 November 2026.

undetermined. Requiring the controller to use B's edges preserves the given implementation choices.

Bojańczyk develops origin semantics as a machine-independent account of these associations [Boj14]. Winter uses pointwise origin bounds in uniformisation problems for tree transducers [Win21]. Several nearby problems separate along three choices: offline versus online selection, synthesis versus selection of given edges, and relational versus functional specifications. Filiot, Jecker, Löding, and Winter study offline equivalence of streaming string transducers through bounded delay [FJLW23]. Filiot and coauthors decompose finite-valued streaming string transducers into deterministic components [FJL$^+$25]. Their decomposition preserves the computed relation; it gives no requirement to retain the original edge sequences or origins.

Bose and coauthors study offline comparison through resynchronisers [BKM$^+$19]. Kuperberg and Martens prove undecidability for resynchronisation classes containing bounded shifts [KM20]. Their hardness constructions allow several output words on one input. Our hard instances have one output word, and every input admits a run with exactly the reference origins. The controller must choose earlier. Two continuations can agree throughout its visible window and still require different choices.

Löding and Repke study delegators that choose automaton transitions with bounded input lookahead [LR13]. Their objective requires acceptance. Löding's erratum identifies a false branching lemma and invalidates the proofs of the published PSPACE upper bounds and lookahead bound [Lö23]. We establish the acceptance argument needed here independently, keeping compatible input continuations in the same witness. Exibard, Filiot, and Jecker supply the counter-tree idea for online selection among runs of multi-sequential relations [EFJ18].

For the positive result, the obstacle is that unsuccessful branches can disappear. A bound of K accepting runs permits arbitrarily many syntactic forks along an execution. We instead count live alternatives together with accepting continuations that justify charging them to K. Acceptance and compatibility stay coupled. An environment strategy defeating every controller yields a finite kill-witness tree, with at most 2^K leaves. Its limbs describe the responses to different controller choices. Coherence prevents incompatible input futures from being assembled into one obstruction.

Witness automata then track output ranks and birth positions jointly across this tree. Finite descriptions of their weight sets make the resource search effective. Failure is downward closed in the lookahead and distance bounds. The cutoff argument extracts an integer ray from an unbounded obstruction family. This gives a computable search threshold, after which a finite game yields the controller.

For hardness, we store counter information in register updates. Appending to one register and prepending to another arrange stored letters in opposite orders. A later probe exposes the imbalance through corresponding output ranks. The controller still chooses only edges. Padding places decisive

information beyond every fixed lookahead window. The halting direction uses an invariant controlling transfers and surviving junctions, so the probe analysis also gives a uniform origin bound.

Our main positive result, Theorem B$^\flat$, allows partial transitions and non-final states. Under $\mathbf{G}^\flat$, the reference is defined on $\mathrm{dom}(B)$, and accepting runs of B agree with its output word. The controller must accept every input in $\mathrm{dom}(B)$ and meet the uniform origin bound there. If B has finitely many accepting runs per word with a uniform bound, we decide $Q_d^\flat$ and $J^\flat$. The bound on ambiguity can be computed. On positive instances we construct finite memory, together with suitable resource bounds. Allowing arbitrary controller memory gives the same answers. The stronger promise $\mathbf{G}$ makes B input-enabled and all its states final; Theorem B follows as a special case.

Theorem C proves RE-completeness of Q_d for every fixed d, and of J, already under $\mathbf{G}$. Its instances have at most $|w| + 1$ runs on each input w. They have at most two origin-annotated outputs, including the reference's exact output with origins. The reference is fixed throughout the reduction. Thus perfect offline alignment coexists with undecidable online selection. At each fixed lookahead d, no computable function of instance size bounds the memory or the distance needed on positive instances in this class, and none bounds lookahead jointly with either.

The ambiguity gap is sharp. Measure ambiguity by the maximum number of accepting runs on words of length at most n. Weber and Seidl's ambiguity criterion supplies a pumping pattern whenever this number is unbounded [WS91, KMRW26]. It yields a linear lower bound for all sufficiently large n. Hence an ambiguity bound g with $\liminf_{n\to\infty} g(n)/n = 0$ forces finite ambiguity. We also slow the hardness construction to every positive linear rate. For every $g \geq 1$ with $\liminf_{n\to\infty} g(n)/n > 0$, both Q_d for fixed d and J remain RE-complete on the class bounded by g.

For fixed ambiguity bound K, the three decision problems are in PSPACE under $\mathbf{G}^\flat$. Supplied values of d and H are encoded in binary. Under $\mathbf{G}$, fixing also the total register count $k \geq 2$ gives NL-completeness. With unrestricted register count, $\mathrm{GOOD}(d, H)$ is PSPACE-complete already for a fixed ambiguity bound when H is supplied. Under $\mathbf{G}^\flat$, a unary input bound K gives EXPSPACE; finite ambiguity without a supplied bound gives 2-EXPSPACE. Under $\mathbf{G}$, EXPSPACE suffices without supplying K. Kiefer, Mazowiecki, Ryzhikov, and Wiśniewski provide the sharper ambiguity estimate used in the size analysis [KMRW26]. Their work is an unrefereed preprint. Weber and Seidl's earlier estimate also suffices for the 2-EXPSPACE class bound [WS91].

Finite memory can still be large. We give instances of size s with $K = 2$ requiring $2^{\Omega(s)}$ memory, regardless of the chosen lookahead and distance. For the one-way transducer model considered here, arbitrary ambiguity remains decidable, and J is EXPTIME-complete. The upper bound passes through

bounded lag and the delay games of Klein and Zimmermann [KZ16]; this reduction may be folklore.

2. Model and example

Throughout, Σ is the input alphabet and Γ the output alphabet, both finite. A machine reads one input letter per step and rewrites all its registers at every step. Registers hold words over Γ.

2.1. **Transducers with origins.** Fix a finite register set $\mathcal{X} = \{X_1, \dots, X_k\}$. A *substitution* is a map $\sigma : \mathcal{X} \to (\Gamma \cup \mathcal{X})^*$. It is *copyless* if every register occurs at most once in the concatenation $\sigma(X_1) \cdots \sigma(X_k)$. A copyless update may drop a register. It never duplicates one.

Definition 2.1 (SST with final states)**.** A *streaming string transducer* (SST) is a tuple $\mathcal{T} = (Q, q_0, \mathcal{X}, E, F, \Phi)$. Here Q is a finite set of states with initial state q_0, $\mathcal{X}$ is a finite set of registers, E is a finite set of edges, $F \subseteq Q$ is the set of final states, and $\Phi : F \to \mathcal{X}^*$ assigns a final expression to every final state, with each register occurring at most once. An edge e has a source $\mathrm{src}(e) \in Q$, a label $\mathrm{lab}(e) \in \Sigma$, a copyless update $\mathrm{upd}(e)$ and a target $\mathrm{dst}(e) \in Q$. Two distinct edges may agree in all four components. We write $\Delta(q, a)$ for the set of edges with source q and label a; this set may be empty.

Parallel edges stay distinct because we count runs as edge sequences. A final expression contains no output letter, so evaluating it creates nothing new. Remark 2.9 lifts this restriction.

A *run* of $\mathcal{T}$ on $w = a_1 \cdots a_n$ is a sequence of edges $\rho = e_1 \cdots e_n$ with $\mathrm{src}(e_1) = q_0$, $\mathrm{lab}(e_i) = a_i$ and $\mathrm{dst}(e_i) = \mathrm{src}(e_{i+1})$. We write $q_i = \mathrm{dst}(e_i)$ for the state after step i. The run is *accepting* if $q_n \in F$, and $\mathrm{dom}(\mathcal{T})$ is the set of words with an accepting run. When $F = Q$ every run is accepting.

Origins. An output letter enters the computation at the step whose update writes it. A register may then hold that letter for a long time, so the step at which it is finally emitted tells us little. We record the entering step. Formally, every occurrence of a letter $\gamma \in \Gamma$ in $\mathrm{upd}(e_i)$ is *born at* i. The annotated valuations $v_i : \mathcal{X} \to (\Gamma \times \mathbb{N}_{\geq 1})^*$ are $v_0(X) = \varepsilon$ and

$$v_i(X) = \mathrm{upd}(e_i)(X)\big[\, Y \mapsto v_{i-1}(Y),\ \gamma \mapsto (\gamma, i)\,\big].$$

The *annotated output* of an accepting run is $[\![\rho]\!] = \Phi(q_n)[v_n] \in (\Gamma \times [n])^*$, and $\lfloor [\![\rho]\!] \rfloor \in \Gamma^*$ is its underlying word. The birth of an output position is its *origin*. Every output letter has an origin in $[n]$, since final expressions create no letters. This is the origin semantics that Bojańczyk gave for streaming string transducers [Boj14].

Size. The *size* s of a pair of machines is the length of an explicit description: alphabets, states, edges, registers, and every letter and register occurrence in updates and final expressions, one unit each. A binary encoding has bit length b with $s \leq b = O(s \log s)$. Polynomial complexity classes agree in both measures, and all bounds below are in s.

2.2. **Instances and promises.** An *instance* is a pair (A, B) of SSTs over the same alphabets with disjoint register sets. A is the reference and B the machine we control.

Definition 2.2 (the promise $\mathbf{G}^\flat$)**.** An instance (A, B) satisfies $\mathbf{G}^\flat$ if:

- (R1) A is deterministic: $|\Delta_A(q, a)| \leq 1$ for all q, a. It may be partial. We write $\rho_A(w)$ for its run on w, when that run exists.
- (R2) B is arbitrary, possibly partial, with final states F_B.
- (R3) Both machines have copyless updates, a unique initial state, empty initial registers, and final expressions without letters.
- (R4) $\mathrm{dom}(B) \subseteq \mathrm{dom}(A)$, and every accepting run ρ of B on a word $w \in \mathrm{dom}(B)$ has $\lfloor [\![\rho]\!] \rfloor = \lfloor [\![\rho_A(w)]\!] \rfloor$.

The $K^\flat$*-promise* says that B has at most K accepting runs on every word. B is *finitely ambiguous* if this holds for some K; the least such K is the degree of ambiguity of the NFA (Q_B, E_B, F_B). B has *linear ambiguity* if it has at most $|v| + 1$ accepting runs on every v.

(R4) is word agreement: whatever branch B takes, the output word is the one A produces. Only the origins may differ. The word function alone fixes no timing, and A supplies one.

Remark 2.3 (the promise $\mathbf{G}$). The *strong promise* $\mathbf{G}$ is the case where all states of both machines are final, A is complete, and B is input-enabled on reachable states: $\Delta_B(q, a) \neq \emptyset$ for every reachable q and every a. Then every run is accepting, $\mathrm{dom}(B) = \Sigma^*$, the $K^\flat$-promise is the plain bound of K runs per word, and (R4) says that every run of B on every w outputs $\lfloor [\![\rho_A(w)]\!] \rfloor$. So $\mathbf{G} \subseteq \mathbf{G}^\flat$. Theorem 3.4 and the lower bounds hold already under $\mathbf{G}$; Theorem 3.1 is proved under $\mathbf{G}^\flat$.

Remark 2.4 (deciding the promise). Every part of $\mathbf{G}^\flat$ is decidable, and (R4) is in PSPACE. (R1) and (R3) are syntactic. The inclusion $\mathrm{dom}(B) \subseteq \mathrm{dom}(A)$ takes polynomial time: complete A with a non-final sink, exchange final and non-final states, and test the product with B for emptiness. Weber and Seidl decide finite ambiguity of an NFA and compute its degree [WS91]; Kiefer, Mazowiecki, Ryzhikov and Wiśniewski bound that degree [KMRW26]. Both count paths of an NFA without parallel edges, so we apply them to the NFA whose states are the edges of B. This gives $K \leq 2^{O(s)}$, computable in space $2^{O(s)}$ (Section 8). So K need not be part of the input.

For the rest of (R4), let $C = A \uplus B$ be the disjoint union: component states and registers side by side, every update extended by the identity on the other component's registers, which keeps it copyless, and a fresh initial state that copies the outgoing edges of both old initial states. The fresh state is final, with empty final expression, exactly when one of the old initial states was. Write $\mathcal{O}_T(w)$ for the set of output words of the accepting runs of T on w. Then $\mathcal{O}_C(w) = \mathcal{O}_A(w) \cup \mathcal{O}_B(w)$, and under (R1)–(R3)

$$\text{(R4)} \iff \mathrm{dom}(B) \subseteq \mathrm{dom}(A) \ \wedge \ \forall w \ |\mathcal{O}_C(w)| \leq 1.$$

For the left-to-right direction, a word of $\mathrm{dom}(B)$ has one output from A and the same output from every accepting run of B; a word outside $\mathrm{dom}(B)$ has at most A's output. For the converse, a word of $\mathrm{dom}(B)$ lies in $\mathrm{dom}(A)$, so A's output is in $\mathcal{O}_C(w)$, and every accepting run of B must produce it. The inclusion remains a separate test, since a word accepted by B alone may well have a unique output. Alur and Deshmukh decide functionality of nondeterministic SSTs in PSPACE [AD11, Thm. 4.1]. Their model allows partial transitions, discarded registers and a partial output function, so C is an instance of it. C has polynomial size, which gives the bound.

2.3. **Controllers.** Fix $d \in \mathbb{N}$. For $w = a_1 \cdots a_n$ and $1 \le i \le n$, the *d-window* at i is

$$\mathrm{win}_d(w, i) = a_i a_{i+1} \cdots a_{\min(n, i+d)} \cdot \big(\dashv \text{ iff } n - i < d \big).$$

The end marker $\dashv$ appears exactly when fewer than d letters follow position i. A full window shows no marker. For $d = 0$ the window is a_i alone and never shows $\dashv$; we call this *strict END*. On $w = ab$ with $d = 1$ the windows are ab and $b \dashv$; with $d = 0$ they are a and b. Win_d is the finite set of windows.

Partial transitions could leave a controller stuck. We complete B first. Let B_z be B with a fresh non-final absorbing state z and one identity edge (q, a, id, z) for every $q \in Q_B \cup \{z\}$ and a with $\Delta_B(q, a) = \emptyset$. The accepting runs of B_z are exactly those of B. Under **G** no reachable state needs a z-edge, so z is unreachable.

Definition 2.5 (controller)**.** A *controller for B with lookahead d* is $\mathcal{C} = (M, m_0, \chi)$ with a set M of memory states, an initial $m_0 \in M$, and

$$\chi : M \times Q^{\mathrm{r}}_{B_z} \times \mathrm{Win}_d \to E_{B_z} \times M,$$
$$\chi(m, q, \omega) = (e, m') \text{ only if } e \in \Delta_{B_z}(q, \omega_1),$$

where $Q^{\mathrm{r}}_{B_z}$ is the set of reachable states and ω_1 the first letter of ω. The controller is *finite-memory* if M is finite, and has *arbitrary memory* otherwise. On input w it produces the run $\rho_{\mathcal{C}}(w) = e_1 \cdots e_n$ given by $(e_i, m_i) = \chi(m_{i-1}, q_{i-1}, \mathrm{win}_d(w, i))$.

The controller chooses an existing edge of B at each position and executes it there. Births are never moved. It never replaces B by A; that would answer a different implementation question. It sees the input, including the end marker, and nothing of the registers. Since B_z is complete, a legal edge always exists; a controller that would get stuck in B takes the forced z-edge and rejects. Unreachable states carry no requirement, because they occur in no run. A controller may act differently on a word and on its extensions, since the windows near the end differ.

2.4. **Origin distance and the problems.** An *annotated word* is $\alpha \in (\Gamma \times \mathbb{N}_{\ge 1})^*$; $\lfloor \alpha \rfloor$ is its underlying word and $b_\alpha(x)$ the birth of its x-th letter,

its *rank* x. For annotated words α_A, α_B with the same underlying word of length N, the *origin distance* is

$$\mathrm{od}(\alpha_A, \alpha_B) = \max_{x \in [1,N]} |b_{\alpha_A}(x) - b_{\alpha_B}(x)|,$$

with $\max \emptyset = 0$, so $\mathrm{od}(\varepsilon, \varepsilon) = 0$. For runs ρ, ρ' on the same input with the same output word we write $\mathrm{od}(\rho, \rho')$ for $\mathrm{od}([\![\rho]\!], [\![\rho']\!])$. By (R4), $\mathrm{od}(\rho_A(w), \rho)$ is defined for every $w \in \mathrm{dom}(B)$ and every accepting run ρ of B on w. Winter uses this pointwise distance for k-origin-closeness [Win21]; it also underlies the k-shift resynchronizers of [KM20].

Fixing both resources gives one problem, and releasing them one at a time gives two more. For $d, H \in \mathbb{N}$:

- $\mathrm{GOOD}^{\flat}(d, H)$: some finite-memory controller with lookahead d produces, on every $w \in \mathrm{dom}(B)$, an accepting run with $\mathrm{od}(\rho_A(w), \rho_{\mathcal{C}}(w)) \leq H$;
- $\mathrm{GOOD}^{\flat,\infty}(d, H)$: the same with arbitrary memory;
- $Q_d^{\flat}$: $\exists H\ \mathrm{GOOD}^{\flat}(d, H)$, and $J^{\flat}$: $\exists d \exists H\ \mathrm{GOOD}^{\flat}(d, H)$; likewise $Q_d^{\flat,\infty}$ and $J^{\flat,\infty}$.

Words outside $\mathrm{dom}(B)$ impose nothing. Inside $\mathrm{dom}(B)$ the controller owes two things: acceptance, and a bounded distance. Under **G** every run is accepting and $\mathrm{dom}(B) = \Sigma^*$, and we drop the flat: $\mathrm{GOOD}(d, H)$, Q_d, J.

Lemma 2.6 (monotonicity)**.** *If $d \leq d'$ and $H \leq H'$, then* $\mathrm{GOOD}^{\flat}(d, H)$ *implies* $\mathrm{GOOD}^{\flat}(d', H')$*, and likewise with arbitrary memory.*

Proof. The d-window is a function of the d'-window at the same position: keep the first $d+1$ letters, and add $\dashv$ iff the d'-window has fewer than d letters after its first. If $n - i < d \leq d'$ the d'-window ends at a_n and shows $n - i < d$ letters after a_i; if $n - i \geq d$ it shows at least d. So a lookahead-d' controller with the same memory simulates any lookahead-d controller and produces the same run on every word. The objective plays no role. □

2.5. **A running example.**

Example 2.7. Let $\Sigma = \{a, \#, 0, 1\}$, $\Gamma = \{a, b\}$ and $p = ab$. Both machines have registers F and R and store p^k in both, where k counts the a's before the first $\#$. F grows at the right and R at the left.

- A has states pre (initial), wait, dF, dR. At pre, the letter a loops with $F \mapsto Fp$, $R \mapsto pR$; the letters $0, 1$ loop with the identity; $\#$ leads to wait with the identity. From wait, the letter 1 leads to dR and every other letter to dF, with the identity. dF and dR loop on every letter with the identity. Φ is F at pre, wait and dF, and R at dR.
- B has states pre, mF, mR. At pre it behaves as A on $a, 0, 1$, and on $\#$ it has two identity edges, to mF and to mR. Both loop on every letter with the identity. Φ is F at pre and mF, and R at mR.

All states are final, both machines are input-enabled, and every output is p^k. So (A, B) satisfies $\mathbf{G}$, and B has at most two runs on any word. Block j of F is born at the j-th a, and block j of R at the $(k+1-j)$-th. A decides at the letter after the first # which block order it emits. B must decide at the # itself.

Proposition 2.8 (one letter of lookahead)**.** *For the instance of Example 2.7,* $\mathrm{GOOD}(1, 0)$ *holds, and* Q_0^∞ *fails: every controller with lookahead* 0*, of any memory, has unbounded origin distance.*

Proof. With lookahead 1, choose mR at the first # iff the second symbol of the window is 1; if the window shows ⊣, choose mF. Then B emits the register that A emits, and both machines performed the same updates on the a's. The annotated outputs coincide and od $= 0$. One memory state suffices.

With lookahead 0, the words $a^k\#$ and $a^k\#1$ show the same windows up to position $k+1$. So every controller makes the same choice at the # on both. If it chooses mF, then on $a^k\#1$ the reference emits R with block births $k, \dots, 1$ while B emits F with births $1, \dots, k$. If it chooses mR, the same happens on $a^k\#$ with the roles exchanged. Block j then has births j and $k+1-j$, so od $= \max_j |2j-k-1| = k-1$ for $k \geq 1$. This is unbounded in k. □

A perfect choice exists offline for every input. The difficulty is choosing before the future arrives.

2.6. **Two remarks on the model.**

Remark 2.9 (general final expressions). Allowing $\Phi(q) \in (\Gamma \cup \mathcal{X})^*$, with the constants of the final expression born at step $n+1$, leaves the main results in place. Appending a terminal marker # to the input, with one #-edge per final state that moves $\Phi(q)$ into a single register, turns a general instance into one of Definition 2.1. Strict END is exactly the condition that the marker falls inside the window, so the lookahead d does not grow, even for $d = 0$. The map preserves both promises, the number of accepting runs on each word, the annotated outputs, and $\mathrm{GOOD}^\flat(d, H)$ for every (d, H) in both memory regimes, with controllers converting in both directions on the same memory set. The size grows linearly. Two parameters move: the register count grows by at most two, and the largest number of letters created in one step must now include the constants of the final expressions. With those two readings the theorems below hold for general final expressions, except the case $k \leq 1$ of Theorem 8.5, the fragment $\mathcal{F}_1$ of Appendix F and the promise $\mathbf{G}^\natural$ of Appendix H. Proposition G.6 gives the exact scope. Appendix G gives the construction and the proof.

Proposition 2.10 (change of reference)**.** *Let* A, A' *be deterministic references and* B *a machine such that* (A, B) *and* (A', B) *both satisfy* $\mathbf{G}$*, or both*

satisfy $\mathbf{G}^\flat$. *Put*

$$\Delta(A, A') = \sup_w \ \mathrm{od}(\rho_A(w), \rho_{A'}(w)),$$

with w *over* Σ^* *under* $\mathbf{G}$ *and over* $\mathrm{dom}(B)$ *under* $\mathbf{G}^\flat$, *and* $\sup \emptyset = 0$. *If* $\Delta = \Delta(A, A') < \infty$, *then:*

(1) $\mathrm{GOOD}_A(d, H)$ *implies* $\mathrm{GOOD}_{A'}(d, H + \Delta)$ *with the same controller, and symmetrically, in both memory regimes;*
(2) $Q_d(A, B) \iff Q_d(A', B)$ *for every* d, *and* $J(A, B) \iff J(A', B)$;
(3) when Q_d *holds, the least* H *for* A *and for* A' *differ by at most* Δ.

Conversely, if A, A' *are deterministic, complete, have all states final and have the same word function, and* $J(A, B) \iff J(A', B)$ *for every* B *making both pairs* $\mathbf{G}$*-instances, then* $\Delta(A, A') < \infty$.

Appendix G proves Proposition 2.10.

So the problems depend on the reference only through its class modulo bounded od, and two classes are always separated by some B. Under $\mathbf{G}$, the relation $\Delta(A, A') < \infty$ is itself J for the $\mathbf{G}$-instance (A, A') with $K = 1$, which Theorem B decides (Remark 3.2).

3. Results

Finite ambiguity permits controller synthesis under $\mathbf{G}^\flat$. Linear ambiguity makes the problem RE-complete, even under $\mathbf{G}$. Acceptance matters here: a controller must accept every word in $\mathrm{dom}(B)$, although other branches may die.

Theorem 3.1 (B♭: finite ambiguity)**.** *Let* (A, B) *satisfy* $\mathbf{G}^\flat$, *and suppose that* B *is finitely ambiguous. We can compute its maximum number* K *of accepting runs on one word. We can decide* $J^\flat$ *and, for each supplied* d, $Q_d^\flat$.

There is a computable threshold $T_*^\flat$ *such that*

$$J^\flat \quad \iff \quad \mathrm{GOOD}^\flat(T_*^\flat, T_*^\flat).$$

In the positive case, we can extract a finite-memory controller. On every $w \in \mathrm{dom}(B)$, *it chooses an accepting run whose origin distance from* A*'s run is at most* $T_*^\flat$. *Its memory can be chosen as*

$$(Q_A \cup \{p_\perp\}) \times \mathrm{States}(R^{\mathrm{od}}_{T_*^\flat}),$$

where R^{od}_H *is the deterministic monitor of Section 4. Allowing arbitrary memory leaves both decision problems unchanged.*

If $\mathrm{dom}(B) = \emptyset$, *then* $K = 0$ *and every controller wins. The algorithm uses* $K = 1$ *in this case.*

The theorem holds without the memory bound when acceptance and $\mathrm{od} \leq H$ *are owed only on a regular set* $O \subseteq \mathrm{dom}(B)$. *Then* K *is the degree of ambiguity of the product of* B *with a DFA for* O.

Exibard, Filiot, and Jecker supply the counter-tree idea [EFJ18]. We keep accepting continuations attached to the alternatives counted in such a tree. Its limbs must also share compatible input futures. Counting forks is insufficient. The coherent tree gives a finite obstruction, which the witness automata and cutoff argument make effective in Sections 5–6.

Remark 3.2 (B: the strong promise). Under **G**, Theorem 3.1 gives decidability of J and Q_d, with controller extraction. The threshold can again be chosen with $d = H = T_*$. In this case we can use the syntactic fork budget

$$\kappa_B = \sum_{\substack{(q,a) \text{ reachable} \\ |\Delta_B(q,a)| \geq 2}} (|\Delta_B(q,a)| - 1) \leq s.$$

Thus K need not be supplied to obtain the EXPSPACE upper bound.

Corollary 3.3 (B′: offline choice)**.** *Under* **G** *and the K-promise, we can decide whether*

$$\exists H\ \forall w\ \exists \beta \text{ a run of } B \text{ on } w: \quad \mathrm{od}(\rho_A(w), \beta) \leq H.$$

Proof sketch. We use the witness construction with no controller-choice component and omit the coherence guard. Each input can then use its own run. The full argument is in Appendix C. □

Theorem 3.4 (C: linear ambiguity)**.** *There is a computable reduction from deterministic two-counter machines M to pairs (A_*, B_M) satisfying* **G**. *The input alphabet is $\{0,1\}$, and the output alphabet is $\{a\}$. The reference A_* is fixed: it has 16 states and 8 registers. The machine B_M has $O(|Q_M|) \cdot 16$ states and 16 registers.*

For every input v, there are at most $|v|+1$ runs of B_M. Their annotated outputs satisfy

$$|\{[\![\beta]\!] : \beta \text{ is a run of } B_M \text{ on } v\}| \leq 2,$$
$$[\![\rho_{A_*}(v)]\!] \in \{[\![\beta]\!] : \beta \text{ is a run of } B_M \text{ on } v\}.$$

For every $d \in \mathbb{N}$,

$$M \text{ halts} \iff Q_d \iff Q_d^\infty \iff J.$$

If the halting run has length L, a controller succeeds with $d = 0$ and $H = 2L + 1$.

Consequently, Q_d is RE-complete for every fixed d, and J is RE-complete on this class.

Offline distance is zero. The online difficulty comes from continuations that agree throughout the visible window but require different earlier choices. In Section 7, register pairs append and prepend letters to store an imbalance. A later probe exposes it through the matching of output ranks. The halting direction uses the transfer and junction conditions of invariant (I4); padding supplies the challenges in the other direction.

Corollary 3.5 (No computable resource bounds)**.** *Fix d. On the positive instances of Theorem 3.4, no computable function of instance size bounds the memory required by a successful controller. Nor does a computable function bound a sufficient distance H.*

A fixed controller's bounded-distance property is decidable. A computable memory bound would therefore leave finitely many controllers to check. A computable distance bound would reduce the question to $\mathrm{GOOD}(d, H)$. Either bound would decide the halting problem. Section 7 gives the fixed-controller argument.

Proposition 3.6 (Ambiguity growth)**.** *Write*

$$a_B(n) = \max_{|w| \le n} |\{\beta : \beta \text{ is an accepting run of } B \text{ on } w\}|.$$

If B has unbounded ambiguity, there are integers $c_0 \ge 0$ and $\ell \ge 1$ such that

$$a_B(n) \ge \left\lfloor \frac{n - c_0}{\ell} \right\rfloor \qquad (n \ge c_0).$$

Hence a bound $a_B(n) \le g(n)$ with $\liminf_{n\to\infty} g(n)/n = 0$ forces finite ambiguity. On every such promised class, $J^\flat$ and each $Q_d^\flat$ are decidable.

For every bound g satisfying $g(n) \ge 1$ and $\liminf_{n\to\infty} g(n)/n > 0$, these problems are RE-complete. The lower bound holds under $\mathbf{G}$, for each fixed d, with a fixed reference and at most two annotated outputs per input. One of those outputs agrees exactly with the reference.

Unbounded ambiguity pumps linearly many accepting runs. Slowing the construction of Theorem 3.4 places its additional runs below any prescribed positive linear rate. The slowdown appears in Section 7 and the pumping argument in Appendix D.

Theorem 3.7 (Complexity)**.** *Let s be the instance size and $k = k_A + k_B$ the total register count. The decision bounds in Table 1 hold for fresh-free final expressions. For every supplied pair (d, H), $\mathrm{GOOD}(d, H)$ and $\mathrm{GOOD}^\flat(d, H)$ are decidable without an ambiguity restriction. Supplied d and H use binary encoding unless indicated otherwise. A supplied K uses unary encoding; a fixed K is a constant promise.*

At fixed K, $\mathrm{GOOD}(d, H)$ is PSPACE-complete under either promise when H is supplied. Hardness already holds with $K = 1$, every fixed d, and unary H.

Under $\mathbf{G}$, fixing both $K \ge 1$ and $k \ge 2$ makes $\mathrm{GOOD}(d, H)$, Q_d, and J NL-complete. Hardness holds with deterministic B and one register in each machine. For $k \le 1$, these problems are trivial under $\mathbf{G}$. Under $\mathbf{G}^\flat$, the proved upper bound at fixed K, k is PSPACE.

With K and the registers fixed, the rank-counter graph has polynomial size. We scan its rows in logarithmic space. Growing register storage therefore accounts for the remaining fixed-K gap under $\mathbf{G}$. For fixed K and

Problem	Fixed parameters; supplied parameters	Under $\mathbf{G}$	Under $\mathbf{G}^\flat$
GOOD(d, H)	Fixed K; supplied d, H	PSPACE-complete	PSPACE-complete
GOOD(d, H)	Fixed K, H; supplied d; unbounded k	NL-hard; in PSPACE	NL-hard; in PSPACE
Q_d, J	Fixed K; supplied d for Q_d	NL-hard; in PSPACE	NL-hard; in PSPACE
GOOD$, Q_d, J$	Fixed K, k, $k \geq 2$; supplied d, H where applicable	NL-complete	NL-hard; in PSPACE
GOOD$, Q_d, J$	Finite ambiguity; supplied unary K and applicable d, H	PSPACE-hard; in EXPSPACE	PSPACE-hard; in EXPSPACE
GOOD$, Q_d, J$	Finite ambiguity; K omitted; supplied applicable d, H	PSPACE-hard; in EXPSPACE	PSPACE-hard; in 2-EXPSPACE
Q_d, J	At most $\lvert w\rvert + 1$ runs; fixed d for Q_d	RE-complete	RE-complete

One-way model		Complexity
J	Any ambiguity; no supplied bounds	EXPTIME-complete
Q_d	Any ambiguity; supplied unary d	EXPTIME-complete
Q_d	Any ambiguity; supplied binary d	EXPTIME-hard; doubly exponential time in general; EXPTIME for $d \geq d_0$
GOOD$, Q_d, J$	Fixed K; supplied d, H where applicable	NL-complete (Proposition 9.2)

TABLE 1. Decision bounds. The flat-promise entries use GOOD$^\flat, Q_d^\flat, J^\flat$. Only unconditional bounds are included.

unbounded registers, the complexity of Q_d, J, and fixed-H GOOD remains open between NL-hardness and PSPACE membership. The fixed-H classification is open as far as we know.

Appendix E.1 collects the explicit lookahead, memory and space bounds.

Proposition 3.8 (Lookahead and memory lower bounds)**.** *Under* $\mathbf{G}$*, a family with* $K = 2$ *and five registers requires lookahead* $\Omega(s)$*. With ambiguity supplied, there are instances of size* $O(n^2)$ *and ambiguity* $n + 1$ *requiring lookahead* 2^n*.*

There is also a family of size s with $K = 2$ requiring $2^{\Omega(s)}$ controller states. This memory lower bound holds for every successful choice of d and H. All three lower bounds also hold under $\mathbf{G}^\flat$.

Proof sketch. We store n past bits in register placement and query them at the single fork. The resulting 2^n distinguishable histories force 2^n controller states. Storing those bits also suffices on this family. Appendix E gives the construction, the argument for arbitrary d, H, and the lookahead lower bounds. □

Corollary 3.9 (Delegating accepting runs)**.** *Let B be a finitely ambiguous NFA. It is decidable whether finite memory and some fixed lookahead suffice to choose an accepting run on every word in* $\mathrm{dom}(B)$*. At fixed K, the problem is in PSPACE. With unary K supplied it is in EXPSPACE, and with K omitted it is in 2-EXPSPACE. A successful controller can be extracted.*

We obtain this case by giving every run empty output. Acceptance remains the controller's obligation.

Proposition 3.10 (One-way contrast)**.** *In the one-way model of Section 9, J is EXPTIME-complete with unrestricted ambiguity. So is Q_d when d is supplied in unary. For each fixed d, finite memory suffices whenever arbitrary memory does.*

Write $n_A = |Q_A|$ and $n_B = |Q_B|$. A sufficient lookahead is

$$d_0 = 2^{\,2(2n_An_B+1)^2+1} - 1.$$

There are instances with $|Q_B| = O(n)$ requiring lookahead 2^n.

Proof sketch. Bounded origin distance reduces to bounded lag in this model. We then apply the delay-game machinery of Klein and Zimmermann [KZ16]. The decidability reduction may be folklore. Appendix F gives the proof and the EXPTIME lower bound. □

Registers change the comparison. They retain strings whose contribution to final output ranks can be rearranged. Appendix F also treats the one-register append-only fragment with final expressions in $\{X, \varepsilon\}$.

4. Monitoring a fixed bound

For fixed $d, H \in \mathbb{N}$, the problem $\mathrm{GOOD}^\flat(d, H)$ reduces to a finite game. This holds at any ambiguity. We first build a finite monitor for the origin bound. Its counter compares output ranks while the input reveals birth positions.

Let $k = |\mathcal{X}_A| + |\mathcal{X}_B|$, and let μ_X bound the number of letters introduced by one update of $X \in \{A, B\}$. Copylessness bounds the number of surviving letters with any one birth by μ_X.

Lemma 4.1 (Bounded rank counter)**.** *Let α_A, α_B be annotated words with births in $[1, n]$. Suppose each birth occurs at most μ_X times in α_X. Fix $c \geq 1$*

such that both words contain ranks $1, \dots, c-1$ *and* $|b_A(y) - b_B(y)| \le H$ *for every* $y < c$*. Then, for every* $r \in [0, n]$*,*

$$D(r) := |\{y < c : b_A(y) \le r\}| - |\{y < c : b_B(y) \le r\}| \in [-\mu_B H, \mu_A H].$$

Proof sketch. Cancel ranks whose two births lie on the same side of r. A remaining positive contribution has its A-birth in $(r-H, r]$, which allows at most $\mu_A H$ contributions. The negative contributions satisfy the symmetric bound. The full argument is in Appendix A. □

We choose c as the least rank with unequal letters or birth gap exceeding H. If the lengths differ and every common rank is good, we take $c = \min(|\alpha_A|, |\alpha_B|) + 1$. Every earlier rank satisfies the lemma.

Let S_A, S_B be the finite substitution sets and $\mathcal{F}_A, \mathcal{F}_B$ the permitted final expressions. Write

$$\Theta = (S_A \times S_B) \uplus (\mathcal{F}_A \times \mathcal{F}_B).$$

A well-formed joint word consists of substitution pairs followed by one final pair. Let L_H^{od} contain those joint words whose outputs agree as words and have origin distance at most H.

Theorem 4.2 (Explicit origin monitor)**.** *For every* $H \in \mathbb{N}$*, an NFA* $\mathcal{N}_H$ *accepts precisely the well-formed joint words outside* L_H^{od}*, with*

$$|\mathcal{N}_H| \le 16 \cdot 18^k \cdot (|\Gamma| + 1)\big((\mu_A + \mu_B)H + 1\big)(H + 2) + 1.$$

The language L_H^{od} *has a DFA with at most* $3 \cdot 2^{|\mathcal{N}_H|}$ *states. Both constructions take time polynomial in their state counts and the explicit description lengths of* $S_A, S_B, \mathcal{F}_A, \mathcal{F}_B$*. The same bounds apply over edge pairs* $E_A \times E_B$*, followed by a final pair.*

We guess one of four failures: a birth gap, unequal letters, or either direction of a length mismatch. For the first two, each output receives colours $\mathsf{P}^*\mathsf{M}\mathsf{S}^*$. The mark selects the tested rank. For a length mismatch, we mark the longer output and colour the shorter output entirely P.

For each register, we guess whether its contents survive to the final output. These use bits propagate backwards through the updates. We also store the entry and exit states of a colour automaton for each register. Concatenation composes these endpoints along each update. Empty initial registers have equal endpoints; the final expression must give an accepting colour path. The marked colour automaton has three states, including a rejecting sink. The all-P automaton has two. Together, use bits and endpoint pairs require at most 18^k choices.

Two flags record whether the marks have been born. Starting at zero, a counter R adds the number of surviving P-letters born on the A side and subtracts the corresponding number on the B side. We restrict it to $[-\mu_B H, \mu_A H]$. For a gap failure, a counter G starts at zero and increases after each row whose updated mark flags differ, saturating at $H + 1$. For a letter failure, we store the first marked letter and require the other to

differ; simultaneous marks are compared directly. Each marked side permits exactly one surviving mark.

The final pair checks the use bits, colour endpoints, and $R = 0$. A gap failure also requires $G = H + 1$; a letter failure requires unequal marks. A length failure needs no additional test. Successful checks enter a single accepting state with no outgoing transitions.

Proof sketch. The endpoint checks make the guessed colours consistent with the final outputs. The equation $R = 0$ aligns the two marked ranks. In a length failure, it places the mark immediately after the shorter output. The mark flags locate births, so G records their separation up to saturation.

For completeness, we mark the least bad rank. Lemma 4.1 keeps R within its allowed interval at every row. The final colours then satisfy the selected failure test. Determinisation, complementation, and a three-state well-formedness check give the DFA bound. Appendix A supplies the full transition rules and proof. □

Remark 4.3 (MSO construction). An MSO construction also gives a finite monitor; see Appendix A. The explicit construction above gives the stated size bound. On accepting runs covered by word agreement, the gap test alone suffices.

Under $\mathbf{G}^\flat$, we must also check acceptance on $\mathrm{dom}(B)$. Complete A and B by rejecting sinks $p_\perp$ and z, obtaining $A_\perp$ and B_z. Let R_H^{od} be the DFA from Theorem 4.2 for these completed machines. We use the finite arena $G_{d,H}^\flat$ with positions

$$(q_A, q_B, r, S, \mathrm{buf}, \mathrm{eof}),$$

where r is a monitor state, $S \subseteq Q_B$, $\mathrm{buf} \in \Sigma^{\leq d+1}$, and $\mathrm{eof} \in \{0, 1\}$. Initially the machine and monitor states are initial, $S = \{q_0^B\}$, the buffer is empty, and $\mathrm{eof} = 0$.

While $\mathrm{eof} = 0$ and the buffer has fewer than $d+1$ letters, the environment reveals a letter or declares EOF. The controller moves when the buffer is nonempty and either full or at EOF. It chooses an edge of B_z on the head letter a and consumes a. We update q_A, q_B, r using the chosen edge and $A_\perp$'s unique edge, and set $S := \delta_B(S, a)$.

The EOF rule is strict. At position i, EOF is visible exactly when $n - i < d$. Thus the buffer and EOF flag encode $\mathrm{win}_d(w, i)$. A terminal has an empty buffer and EOF declared. It is bad precisely when

$$S \cap F_B \neq \varnothing \quad \text{and} \quad \Big(q_B \notin F_B \text{ or } R_H^{\mathrm{od}} \text{ rejects the final pair}\Big).$$

We evaluate the monitor test only when $q_B \in F_B$. On an obligation word, $\mathbf{G}^\flat$ ensures that A's final expression is defined. The environment seeks a bad terminal; infinite plays favour the controller.

Theorem 4.4 (Fixed resources under $\mathbf{G}^\flat$)**.** *Under $\mathbf{G}^\flat$, with arbitrary ambiguity, for every $d, H \in \mathbb{N}$,*

$$\mathrm{GOOD}^\flat(d,H) \iff \mathrm{GOOD}^{\flat,\infty}(d,H) \iff \textit{the controller wins } G^\flat_{d,H}.$$

The superscript ∞ permits arbitrary memory. The game is finite and computable. A positional winning strategy gives a controller with memory set

$$(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R_H^{\mathrm{od}}) \times 2^{Q_B}.$$

Consequently, $Q_d^\flat$ and $J^\flat$ are recursively enumerable.

Proof sketch. We map each history to its displayed arena position. The set S records all states reachable in B on the consumed prefix. Hence a terminal has $S \cap F_B \neq \varnothing$ exactly on $\mathrm{dom}(B)$. The quotient preserves moves and losing terminals.

Finite reachability games admit positional winning strategies. The controller observes q_B and the window; its memory tracks q_A, r, S. An environment winning strategy defeats controllers of arbitrary memory. Enumerating H, or both d, H, gives the two semidecision procedures. Appendix A gives the full correspondence. □

The acceptance test permits a smaller controller memory. Write $\mathrm{SafeQ}_d(q,a,x,\mathrm{eof})$ for the acceptance-safe edge set of Section 5 (Lemma 5.2). It tests preservation of accepting continuations compatible with the current window. Its inputs are the current state and window.

Define $G^{\mathrm{q}}_{d,H}$ by omitting S from the positions. At a controller move with buffer ax, permit only edges in $\mathrm{SafeQ}_d(q_B,a,x,\mathrm{eof})$. An empty permitted set loses immediately. Other terminals are bad exactly when $q_B \in F_B$ and the monitor rejects the final pair.

Proposition 4.5 (Controller memory)**.** (a) *Under $\mathbf{G}^\flat$, for every $d, H \in \mathbb{N}$,*

$$\mathrm{GOOD}^\flat(d,H) \iff \mathrm{GOOD}^{\flat,\infty}(d,H) \iff \textit{the controller wins } G^{\mathrm{q}}_{d,H}.$$

A positional winning strategy gives a controller with memory set

$$(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R_H^{\mathrm{od}}).$$

(b) *For acceptance alone, an arbitrary-memory lookahead-d controller exists if and only if a memoryless one exists. Here memoryless means that the chosen edge depends only on the current state and window.*

Proof sketch. The acceptance lemmas in Section 5 show that every successful controller uses safe edges. Along such choices, the safe-edge test depends only on the current state and window, by induction on the position with Lemma 5.2. Conversely, safe choices ensure acceptance on every word in $\mathrm{dom}(B)$. The monitor checks the remaining origin obligation. A positional strategy therefore needs to retain only q_A and r.

For acceptance alone, give every edge an empty update and take a one-state reference. The monitor then accepts every joint word arising from a

run. The remaining arena positions contain only the current state, buffer, and EOF flag. Appendix A gives the full proof. □

5. Kill witnesses

Under the $K^\flat$-promise, failure at fixed d, H has a finite coherent witness with at most 2^K leaves. Each leaf records either rejection or an origin error exceeding H. Acceptance causes the difficulty: a branch may remain available for many steps before its accepting continuations disappear.

Exibard, Filiot, and Jecker supply the counter-tree idea [EFJ18]. We couple the tree to acceptance on its leaf words. An omitted controller choice must be killed by a compatible word.

Example 5.1. Let $\Sigma = \{a, b\}$ and $\Gamma = \{a\}$. Both machines use one initially empty register X; the states of B are s, t, f, with s initial and s, f final. Its edges are

$$s \xrightarrow{a \,/\, X := Xa} s, \qquad s \xrightarrow{a \,/\, X := Xa} t, \qquad t \xrightarrow{b \,/\, \mathrm{id}} f.$$

Both final states return X. The deterministic reference A has two final states p_0, p_1, starts in p_0, and has edges

$$p_0 \xrightarrow{a \,/\, X := Xa} p_0, \qquad p_0 \xrightarrow{b \,/\, \mathrm{id}} p_1.$$

Its final expressions also return X.

Here $\mathrm{dom}(B) = a^* \cup a^+ b$ and $\mathrm{dom}(A) = a^* \cup a^* b$. Every domain word has exactly one accepting run of B, whose output and origins agree with A. All three states of B are reachable and have accepting continuations. Nevertheless, the accepting run on a^n encounters two outgoing a-edges at every position.

One symbol of lookahead suffices. On a, we enter t precisely when the next symbol is b. This gives $\mathrm{GOOD}^\flat(1, 0)$. No lookahead-0 controller accepts the whole domain. The words a^n and $a^n b$ present identical windows through position n, but acceptance requires different edges there.

Fix $w = a_1 \cdots a_n$. We use the completion B_z, whose absorbing non-final state z receives a transition whenever an original transition is unavailable. For $0 \le i \le n$, put

$$C_i(w) = \{q \in Q_B : B \text{ accepts } a_{i+1} \cdots a_n \text{ from } q\}.$$

These sets satisfy

$$C_n(w) = F_B, \qquad C_{i-1}(w) = \{q : \exists e \in \Delta_B(q, a_i),\ \mathrm{dst}(e) \in C_i(w)\}.$$

For a run $\beta = e_1 \cdots e_n$ with states $q_0, \ldots, q_n$, define

$$\mathrm{Live}_w(\beta, i) = \{e \in \Delta_{B_z}(q_{i-1}, a_i) : \mathrm{dst}(e) \in C_i(w)\}.$$

All other available edges are *dead* at that position for w. After taking a dead edge, a run has no live edge at any later position.

The $K^\flat$-promise bounds the live alternatives along every run:

$$\sum_{i=1}^{n} \left|\mathrm{Live}_w(\beta, i) \setminus \{e_i\}\right| \le K - \mathbf{1}_{\{\beta \text{ accepts}\}}, \qquad w \in \mathrm{dom}(B). \tag{1}$$

Each alternative extends the same run prefix to an accepting run on w. Distinct first deviations give distinct accepting runs. In Example 5.1, the edges into t are dead on a^n, so the sum is zero.

Acceptance also constrains choices before the whole input is known. Let $\mathrm{Cons}_d(w, i)$ contain the words in $\mathrm{dom}(B)$ that present the same windows as w through position i. Without $\dashv$ in the current window, these words agree with w through position $i + d$. Once $\dashv$ appears, the only possible member is w itself. For a current state q, set

$$\mathrm{Safe}_d(w, q, i) = \{e \in \Delta_{B_z}(q, a_i) : \forall v \in \mathrm{Cons}_d(w, i),\ \mathrm{dst}(e) \in C_i(v)\}.$$

An empty consistent set makes every edge safe; an unsafe edge has a consistent domain word on which it is dead.

A lookahead-d controller accepts every word of $\mathrm{dom}(B)$ exactly when every choice it makes, on every input, is safe. Indeed, an unsafe choice is repeated on its killing word, because all windows seen so far agree. Conversely, on a nonempty domain word, safety of the last edge places the run in F_B. The empty word is accepted exactly when $q_0^B \in F_B$. These facts hold for arbitrary controller memory.

The safe set refers to the whole history through Cons_d. Along safe runs it is local. Write $L(q)$ for the set of words that B accepts from q, with $L(z) = \emptyset$, and $u^{-1}L = \{v : uv \in L\}$. For a letter a, a word x and a flag $\eta \in \{0, 1\}$, put

$$\mathrm{SafeQ}_d(q, a, x, \eta) = \begin{cases} \{e \in \Delta_{B_z}(q, a) : (ax)^{-1}L(q) \subseteq x^{-1}L(\mathrm{dst}e)\}, & \eta = 0, \\ \{e \in \Delta_{B_z}(q, a) : ax \in L(q) \Rightarrow x \in L(\mathrm{dst}e)\}, & \eta = 1. \end{cases}$$

At position i of w let $x_i = a_{i+1} \cdots a_{\min(n, i+d)}$, and let $\mathrm{eof}_i(w) = 1$ iff $n - i < d$, that is, iff the window shows $\dashv$.

Lemma 5.2 (safe sets are local)**.** *Let $\beta = e_1 \cdots e_n$ be a run of B_z on w with states $q_0, \ldots, q_n$, and let $1 \le i \le n$. If $e_j \in \mathrm{Safe}_d(w, q_{j-1}, j)$ for all $j < i$, then*

$$\mathrm{Safe}_d(w, q_{i-1}, i) = \mathrm{SafeQ}_d(q_{i-1}, a_i, x_i, \mathrm{eof}_i(w)).$$

Proof sketch. Let S_{i-1} be the set of states that B reaches on $a_1 \cdots a_{i-1}$. The words of $\mathrm{Cons}_d(w, i)$ are the words $a_1 \cdots a_i x_i v$ with $a_i x_i v \in L(S_{i-1})$, where $v = \varepsilon$ under $\dashv$. So $\mathrm{Safe}_d(w, q_{i-1}, i)$ is the test of SafeQ_d with $L(S_{i-1})$ in place of $L(q_{i-1})$. It suffices that $a_i x_i v \in L(S_{i-1})$ iff $a_i x_i v \in L(q_{i-1})$. If the right side holds, then $q_{i-1} \ne z$, so $e_1 \cdots e_{i-1}$ is a run of B and $q_{i-1} \in S_{i-1}$. Conversely, let $i \ge 2$; for $i = 1$ both sets are $\{q_0\}$. The word $w' = a_1 \cdots a_{i-1} a_i x_i v$ lies in $\mathrm{Cons}_d(w, i)$, hence in $\mathrm{Cons}_d(w, i - 1)$, since equal windows through i are equal through $i - 1$. The edge e_{i-1} is safe at $i - 1$, so $q_{i-1} \in C_{i-1}(w')$, which says $a_i x_i v \in L(q_{i-1})$. Appendix B. □

Definition 5.3 (Kill witness)**.** A shape $\mathcal{T}$ is a finite rooted tree with at least two children at each internal node. A kill witness at distance bound H assigns:

- a word $u_\lambda \in \mathrm{dom}(B)$, a run β_λ of B_z, and a type $\tau(\lambda) \in \{\mathrm{OD}, \mathrm{DEAD}\}$ to each leaf;
- an input position $i_\nu \geq 1$ to each internal node.

Positions strictly increase along paths. The assignments satisfy four conditions.

(K1) Below an internal node ν, all words have length at least i_ν. They agree through that position, and their runs agree through position $i_\nu - 1$. Write q_ν for their common state before the next edge and a_ν for the common input letter.

(K2) Each child κ of ν has an edge $e_\kappa \in \Delta_{B_z}(q_\nu, a_\nu)$. All leaves below κ take e_κ at position i_ν, and different children have different edges.

(K3) For a leaf λ and $p \leq |u_\lambda|$, put

$$\Lambda(\lambda, p) = \left\{ \mu : \begin{array}{l} \beta_\mu[1..p-1] = \beta_\lambda[1..p-1], \\ u_\mu[1..p] = u_\lambda[1..p] \end{array} \right\}.$$

Consider any edge from q^λ_{p-1} on the letter $u_\lambda[p]$. If no leaf in $\Lambda(\lambda, p)$ takes that edge at p, then it is dead there for at least one word u_μ with $\mu \in \Lambda(\lambda, p)$.

(K4) An OD leaf has an accepting run and satisfies

$$\mathrm{od}\big(\rho_A(u_\lambda), \beta_\lambda\big) > H.$$

A DEAD leaf has a rejecting run.

For distinct leaves λ, μ, let ν be their lowest common ancestor. Pad each word by a fresh symbol $\perp$: $\widetilde{u}_\lambda = u_\lambda \perp^\omega$. If the padded words first differ at position j, define

$$E(\lambda, \mu) = j - i_\nu - 1;$$

if they are equal, put $E(\lambda, \mu) = \infty$. The witness is *d-coherent* when $E(\lambda, \mu) \geq d$ for every pair of distinct leaves.

Coherence keeps the controller's information identical through each fork. The padding respects strict $\dashv$: at position p, the window shows $\dashv$ exactly when its padded symbol at $p + d$ is $\perp$. Conditions (K1)–(K2) also give

$$\Lambda(\lambda, p) = \{\lambda\} \cup \{\mu \neq \lambda : i_{\mathrm{lca}(\lambda,\mu)} \geq p\}.$$

Thus (K3) applies both at forks and between them. Its deadness tests use each leaf's own input suffix.

Let $\mathfrak{S}^\flat_K$ be the shapes satisfying

$$\sum_{\nu \text{ on the path}} (\deg \nu - 1) \leq K \tag{2}$$

on every root-to-leaf path. Every such shape has at most 2^K leaves.

Theorem 5.4 (Kill witnesses)**.** *Fix $d, H \in \mathbb{N}$.*

(a) *A d-coherent kill witness of any shape rules out every lookahead-d controller that accepts* $\mathrm{dom}(B)$ *with origin distance at most H, including controllers with arbitrary memory.*
(b) *Under the $K^\flat$-promise, if no such controller exists, there is a d-coherent kill witness with shape in $\mathfrak{S}_K^\flat$.*

Proof sketch. The full proof is in Appendix B.

For the first assertion, suppose a controller achieves the objective. Choose a leaf on which its run agrees with the recorded run for the longest prefix. Full agreement contradicts (K4). Otherwise, let p be the first disagreement and e the controller's edge. Coherence makes the controller choose e on every word in $\Lambda(\lambda, p)$. If a recorded run takes e there, its agreement prefix is longer. If none does, (K3) supplies a domain word on which e is dead. Both cases contradict the choice of leaf or the controller's acceptance.

For the converse, restrict the history arena to safe controller choices. A position without a safe edge is an immediate environment win. A winning controller in this arena would also win the original game. Open-game determinacy therefore gives a winning environment strategy. Unfold it, retaining every safe controller choice. Every play ends in an environment win. By König's lemma, this finitely branching tree is finite.

Contract positions with a unique safe edge. A bad terminal reached through safe choices is accepting, so it becomes an OD leaf. At a position i with no safe edge, choose an edge e_1 and a consistent word w_1 killing it. Let E_1 contain all other edges live for w_1 at i. For every $e \in E_1$, choose a consistent word w_e killing e. Attach leaves taking e_1 on w_1 and e on w_e. Complete each run arbitrarily in B_z. These leaves are DEAD, and w_1 kills every edge omitted from this fan.

Call the resulting leaves original. A missing edge can still be live for every original word sharing a given history. We insert a killing leaf for each such edge and history. The edge is unsafe: every safe choice already has a continuation among the original leaves. Choose its killing word consistent with the revealed input, follow the common run prefix, take that edge, and complete the run. We determine every insertion from the original leaves, independently of the leaves added elsewhere.

Coherence is preserved. Each new word shares the revealed prefix through $p+d$ at its insertion position p, or the whole word under $\dashv$. For earlier forks, it inherits the same revealed prefix as its block. After the inserted edge, every further edge of its run is dead. Before that edge, an omitted choice either has an original killing word or receives its own inserted leaf. This verifies (K3).

It remains to bound the shape. Take an original OD leaf, or the distinguished w_1 leaf of a fan. Every sibling edge along its path is live for its word. Safe siblings are live for all consistent words, and inserted siblings were live for every original word in their block. Fan siblings belong to E_1.

Charge each sibling to its position and edge in (1). Distinct positions and edges give distinct charges, whose total is at most K.

Every other fan leaf has the same internal nodes as its w_1 leaf. An inserted leaf follows an initial segment of a path to an original OD or distinguished fan leaf. Its path therefore satisfies the same budget (2). □

Under the $K^\flat$-promise, the theorem gives

(3) $\mathrm{GOOD}^{\flat,\infty}(d,H) \iff$ no d-coherent kill witness has shape in $\mathfrak{S}_K^\flat$.

Here ∞ permits arbitrary controller memory.

Remark 5.5 (Complete forks under $\mathbf{G}$). Under $\mathbf{G}$, kill witnesses reduce to complete-fork witnesses with path budget $K-1$. Every reachable state accepts every remaining input word, so all available edges are live and DEAD leaves are impossible. Condition (K3) consequently requires every available edge to occur among the compatible leaves. At a fork, every outgoing edge is represented. Between forks, the run has only one available edge.

Every run is accepting, so (1) becomes

$$\sum_i \big(|\Delta_B(q_{i-1},a_i)|-1\big) \le K-1.$$

6. Witness automata and cutoff

Theorem 3.1 comes down to one finite game. There is a computable $T_*^\flat$ with $J^\flat$ iff $\mathrm{GOOD}^\flat(T_*^\flat,T_*^\flat)$, and for each d a computable $H_*^\flat(d)$ with $Q_d^\flat$ iff $\mathrm{GOOD}^\flat(d,H_*^\flat(d))$ (Proposition 6.7). The kill witnesses of Section 5 are finite trees, so a finite automaton can read them, one row per input position and all leaves together. Its weights record output ranks, birth positions and gaps. The weight sets are semilinear, and the guards that make a witness an obstruction are lower bounds in d and H. An obstruction that survives every bound therefore contains an integer ray. One that does not has a size we can compute.

6.1. **The automata.** Fix a $\mathbf{G}^\flat$-instance (A,B) with the $K^\flat$-promise, and its completions $A_\perp$ and B_z from Section 2. For $u\in\Sigma^*$ let $C_p(u)\subseteq Q_B$ be the states from which the suffix of u after position p has an accepting run. Then $C_{|u|}(u)=F_B$ and $C_{p-1}(u)=\mathrm{pre}_{u[p]}(C_p(u))$, where $\mathrm{pre}_a(C)=\{q:\mathrm{dst}(e)\in C \text{ for some } e\in\Delta_B(q,a)\}$ (Lemma B.1). An edge e read at position p is dead for u iff $\mathrm{dst}(e)\notin C_p(u)$, and $u\in\mathrm{dom}(B)$ iff $q_0^B\in C_0(u)$.

A *case* is a triple $(\mathcal{T},\varphi,\tau)$: a shape $\mathcal{T}\in\mathfrak{S}_K^\flat$, a set φ of leaf pairs, and a typing τ of the leaves by $\{\mathrm{OD},\mathrm{DEAD}\}$. The pairs in φ are those whose words will differ. The automaton $\mathcal{W}_{\mathcal{T},\varphi,\tau}^\flat$ guesses a kill witness of this shape and typing, checks (K1)–(K3) on the way, and weighs it. Leaves that share a prefix share their letters and B-edges inside one automaton, so their annotations stay correlated.

Clones and colours. At an OD leaf λ the automaton runs two *clones*: the run of $A_\perp$ on the leaf's word and the guessed run of B_z. Every letter occurrence in their register updates receives a colour from $\{\mathsf{P}, \mathsf{M}, \mathsf{S}\}$. An occurrence is *live* if its register content reaches the final expression. By Lemma A.1, each output position is exactly one live occurrence, born in the row of its update. The colour DFA $\mathcal{P}_{\mathrm{mk}}$ has states pre (initial), post (accepting) and a sink z, with transitions pre $\xrightarrow{(\gamma,\mathsf{P})}$ pre, pre $\xrightarrow{(\gamma,\mathsf{M})}$ post, post $\xrightarrow{(\gamma,\mathsf{S})}$ post, and all others to z. On the colour track it accepts $\mathsf{P}^*\mathsf{M}\mathsf{S}^*$. Exactly one position is marked. The automaton checks that the colours of each clone's live occurrences, in output order, lie in $L(\mathcal{P}_{\mathrm{mk}})$. For this it uses the register tracking of Appendix A: use bits $l_\lambda^X \in \{0,1\}^{\mathcal{X}_X}$ say which registers currently reach the output, and endpoint maps $f_\lambda^X : \mathcal{X}_X \to P^2$ store the states of $\mathcal{P}_{\mathrm{mk}}$ at both ends of each register's coloured content. By Lemma A.2, the coloured output lies in $L(\mathcal{P}_{\mathrm{mk}})$ iff endpoint maps exist that satisfy (E0) initially, (Ei) at each step and (EF) at the final expression. The use bits obey a backward law with a unique solution once their final values are fixed, so the automaton guesses them and checks the boundary at termination.

States. Every leaf λ carries a status in $\{\mathrm{RUN}, \mathrm{DONE}\}$, a state $q_\lambda^B \in Q_B \cup \{z\}$ and a set $C_\lambda \subseteq Q_B$. An OD leaf also carries q_λ^A, the use bits and endpoint maps of both clones, and two birth flags $\eta_\lambda^A, \eta_\lambda^B \in \{0,1\}$. Every internal node ν carries a phase in $\{\mathrm{PRE}, \mathrm{POST}\}$. Every pair of leaves carries a phase, in $\{\mathrm{WAIT}, \mathrm{COUNT}, \mathrm{CLOSED}\}$ if the pair lies in φ and in $\{\mathrm{WAIT}, \mathrm{SAME}\}$ otherwise. Initially every leaf is RUN with $q_\lambda^B = q_0^B$ and a guessed C_λ containing q_0^B. OD leaves start with $q_\lambda^A = q_0^A$, diagonal endpoint maps (p_X, p_X) with p_X arbitrary, arbitrary use bits, and flags 0. Nodes are PRE and pairs WAIT.

Rows. A row gives each leaf a letter $a_\lambda \in \Sigma \cup \{\perp\}$ and names a set Mark of internal nodes. A RUN leaf with $a_\lambda \neq \perp$ also gets an edge $e_\lambda^B \in \Delta_{B_z}(q_\lambda^B, a_\lambda)$ and a set $C'_\lambda \subseteq Q_B$ with $C_\lambda = \mathrm{pre}_{a_\lambda}(C'_\lambda)$. An OD leaf gets in addition an edge $e_\lambda^A \in \Delta_{A_\perp}(q_\lambda^A, a_\lambda)$ and a colouring of every letter occurrence in $\mathrm{upd}(e_\lambda^A)$ and $\mathrm{upd}(e_\lambda^B)$. For a clone X let $\#_{\mathsf{M}}^X$ and $\#_{\mathsf{P}}^X$ count the M- and P-coloured occurrences in $\mathrm{upd}(e_\lambda^X)(Y)$, summed over the registers Y whose new use bit is 1. The row is allowed iff:

- **(T1) Step.:** A DONE leaf reads $\perp$ and changes nothing. A RUN leaf reading $a_\lambda \neq \perp$ moves q_λ^B along e_λ^B and replaces C_λ by C'_λ. At an OD leaf, q_λ^A moves along e_λ^A, the old use bits follow from the new ones by the backward law, the new endpoint maps satisfy (Ei) for the chosen colours, and the flags satisfy $\eta_\lambda^X + \#_{\mathsf{M}}^X \leq 1$ and become $\eta_\lambda'^X = \eta_\lambda^X + \#_{\mathsf{M}}^X$.
- **(T2) Termination.:** A RUN leaf reading $\perp$ requires $C_\lambda = F_B$ and becomes DONE. A DEAD leaf requires $q_\lambda^B \notin F_B$. An OD leaf requires $q_\lambda^A \in F_A$, $q_\lambda^B \in F_B$, use bits equal to "occurs in the final expression" for both clones, and (EF).

(T3) Shared past.: For every node ν that is PRE and not in Mark, all leaves below ν read the same letter, which is not $\perp$, and take the same B-edge.

(T4) Fork.: For every $\nu \in$ Mark: ν is PRE and its parent, if any, is POST; all leaves below ν read the same letter $a \neq \perp$ and are in one state q; all leaves below a child κ take one edge e_κ, and $\kappa \mapsto e_\kappa$ is injective into $\Delta_{B_z}(q, a)$. Then ν becomes POST.

(T5) Kill.: For a RUN leaf λ reading $a \neq \perp$, let $\mathrm{Bl}(\lambda)$ consist of λ and the leaves μ whose lowest common ancestor with λ is PRE before the row. Every $e \in \Delta_{B_z}(q^B_\lambda, a)$ taken by no leaf of $\mathrm{Bl}(\lambda)$ must satisfy $\mathrm{dst}(e) \notin C'_\mu$ for some $\mu \in \mathrm{Bl}(\lambda)$. The z-edge is always killed, because $z \notin C'_\mu$.

(T6) Gaps.: For a pair $\pi = \{\lambda, \lambda'\}$ with lowest common ancestor ν, compare padded letters, where a DONE or terminating leaf contributes $\perp$. In WAIT, when $\nu \in$ Mark the pair moves to COUNT if $\pi \in \varphi$ and to SAME otherwise. In COUNT, equal letters add 1 to E_π and different letters move the pair to CLOSED. In SAME the letters must be equal.

(T7) Weights.: At an OD leaf in RUN reading $a_\lambda \neq \perp$, add $\#^A_{\mathsf{P}} - \#^B_{\mathsf{P}}$ to R_λ and $[\eta'^A_\lambda \neq \eta'^B_\lambda]$ to G_λ. All other weights are 0.

A state is accepting if all leaves are DONE, all nodes POST and all pairs in φ CLOSED. A walk stops at its first accepting state. A row in which every leaf is already DONE is not allowed.

The weight of a walk is the sum of its rows, a vector in $\mathbb{Z}^m$ with coordinates (R_λ, G_λ) for OD leaves and $(E_\pi)_{\pi\in\varphi}$. Here $m \leq 2N + \binom{N}{2}$ with $N = 2^K$. Every row has weight norm at most $W = \max(\mu_A, \mu_B, 1)$, where μ_X bounds the letter occurrences in one update of X (Section 2). Neither d nor H enters the automaton.

Lemma 6.1 (origin-distance gadget)**.** *Let λ be an OD leaf of an accepting walk, with word u of length n, clone runs $\rho_A(u)$ and β, and annotated outputs $\alpha_A = [\![\rho_A(u)]\!]$ and $\alpha_B = [\![\beta]\!]$ with birth maps b_A and b_B. Then each clone's coloured output has exactly one* M*-position, of some rank $c^X \geq 1$, and the total weight of the leaf is*

$$R_\lambda = c^A - c^B, \qquad G_\lambda = \big|b_A(c^A) - b_B(c^B)\big|.$$

Conversely, for any ranks $c^A \in [1, |\alpha_A|]$ and $c^B \in [1, |\alpha_B|]$ there is a colouring of the two runs that satisfies (T1), (T2) and the flag rule and has these weights.

Proof. By Lemma A.2, (T1) and (T2) hold for clone X iff its coloured output lies in $\mathsf{P}^*\mathsf{M}\mathsf{S}^*$. So exactly one output position is marked; let c^X be its rank. By Lemma A.1 every output position is one live occurrence in the update of its birth row. Hence the live P-occurrences number $c^X - 1$, and $R_\lambda = (c^A - 1) - (c^B - 1)$. The single live M-occurrence is born in row $b_X(c^X)$. So the cap $\eta + \#_{\mathsf{M}} \leq 1$ is never violated, and after row r the flag is

$\eta^X = [b_X(c^X) \le r]$. The G-weight counts the rows $r \le n$ at which the two flags differ. Both births are at most n, so these rows form the interval from $\min(b_A(c^A), b_B(c^B))$ up to but excluding $\max(b_A(c^A), b_B(c^B))$, and there are $|b_A(c^A) - b_B(c^B)|$ of them. Dead occurrences are never counted, whatever their colour. The exclusive-or of the flags yields the absolute value directly, so the automaton guesses no sign.

For the converse, colour each live occurrence P, M or S according to whether its rank is below, at or above c^X, and colour dead occurrences P. Lemma A.2 supplies the endpoint maps. The flags are then determined. □

Lemma 6.2 (witness automata under $\mathbf{G}^\flat$)**.** *From an accepting walk of $\mathcal{W}^\flat_{\mathcal{T},\varphi,\tau}$ read off, for each leaf λ, the word u_λ of its letters before the first $\perp$ and the sequence β_λ of its B-edges, and for each internal node ν the row i_ν at which it is marked. Then $(u_\lambda, \beta_\lambda, i_\nu)$ satisfies (K1)–(K3) and the DEAD part of (K4) for $\mathcal{T}$ and τ; every u_λ lies in* $\mathrm{dom}(B)$*; the runs β_λ at OD leaves are accepting; exactly the pairs in φ have different words; and $E_\pi = E(\pi)$ for $\pi \in \varphi$. The weight vectors of accepting walks are exactly the vectors*

$$\Big(c^A_\lambda - c^B_\lambda,\ \big|b_{A,\lambda}(c^A_\lambda) - b_{B,\lambda}(c^B_\lambda)\big|\Big)_{\lambda \in \tau^{-1}(\mathrm{OD})}, \qquad \big(E(\pi)\big)_{\pi\in\varphi},$$

over all such data and all ranks $c^A_\lambda, c^B_\lambda \in [1, N_\lambda]$, where N_λ is the common length of $[\![\rho_A(u_\lambda)]\!]$ and $[\![\beta_\lambda]\!]$ (equal by (R4)) and $b_{A,\lambda}, b_{B,\lambda}$ are their birth maps. Conversely, every such datum with such ranks extends to an accepting walk.

Proof sketch. The proof is an invariant along the walk, one clause per field; Appendix C writes it out. Three clauses carry the argument. First, a leaf is RUN in row r iff $r \le |u_\lambda|$, and q^B_λ is then the state of β_λ after $\min(r, |u_\lambda|)$ steps, by (T1). Second, C_λ in row r equals $C_r(u_\lambda)$: (T2) pins it to F_B at the end, (T1) imposes the backward law between rows, and Lemma B.1 says these determine the sequence. So the initial guard $q^B_0 \in C_\lambda$ says $u_\lambda \in \mathrm{dom}(B)$, and $\mathrm{dst}(e) \notin C'_\mu$ in row p says that e is dead for u_μ at p. Third, a node is PRE in row r iff $r < i_\nu$. So $\mathrm{Bl}(\lambda)$ in row p is the set $\Lambda(\lambda, p)$ of Definition 5.3, and all its leaves are in state q^B_λ by (T3) and (T4). Hence (T4) and (T5) together are (K3). (T3) and (T4) give (K1) and (K2), and they mark each node once, in ancestor order, while all leaves below it still run. (T2) gives the DEAD part of (K4) and acceptance at OD leaves. For a pair in φ, (T6) counts the equal rows strictly between i_ν and the first difference, which is $E(\pi)$, and acceptance forces that difference to exist. For a pair outside φ, (T3), (T4) and SAME force equal padded words. At an OD leaf, $u_\lambda \in \mathrm{dom}(B)$ and β_λ is accepting, so by (R4) both outputs have the same underlying word, and Lemma 6.1 gives the weight.

For the converse, set every C_λ to the true $C_r(u_\lambda)$; this satisfies the guard, (T1) and (T2). (K3) gives (T5). DEAD runs end outside F_B. The marks and phases are determined by the i_ν and the words, and Lemma 6.1 supplies colours, endpoint maps and flags. □

6.2. **Deciding a fixed bound through witnesses.** Let $V^{\flat}_{\mathcal{T},\varphi,\tau} \subseteq \mathbb{Z}^m$ be the set of weight vectors of accepting walks of $\mathcal{W}^{\flat}_{\mathcal{T},\varphi,\tau}$, and define

$$\begin{aligned}\mathrm{BAD}^{\flat}(d,H) \;:\Longleftrightarrow\;\; & \exists(\mathcal{T},\varphi,\tau)\;\exists v \in V^{\flat}_{\mathcal{T},\varphi,\tau}: \\ & \forall\lambda \in \tau^{-1}(\mathrm{OD})\,(R_\lambda = 0 \wedge G_\lambda \geq H+1) \;\wedge\; \forall\pi \in \varphi\,(E_\pi \geq d).\end{aligned}$$

Every guard is an equality or a lower bound in (d,H). So $\mathrm{BAD}^{\flat}$ is downward closed: $\mathrm{BAD}^{\flat}(d,H)$ with $d' \leq d$ and $H' \leq H$ gives $\mathrm{BAD}^{\flat}(d',H')$.

Theorem 6.3. *Under the $K^{\flat}$-promise,* $\mathrm{BAD}^{\flat}(d,H) \iff \neg\mathrm{GOOD}^{\flat,\infty}(d,H) \iff \neg\mathrm{GOOD}^{\flat}(d,H)$.

Proof. ($\Rightarrow$) Take a guarded vector and its data from Lemma 6.2. At an OD leaf, $R_\lambda = 0$ gives $c^A_\lambda = c^B_\lambda =: c$, one rank of one word, and $G_\lambda \geq H+1$ gives $|b_A(c) - b_B(c)| > H$. So $\mathrm{od}(\rho_A(u_\lambda), \beta_\lambda) > H$, and (K4) holds. Pairs outside φ have equal words and pairs in φ have $E(\pi) = E_\pi \geq d$, so the witness is d-coherent. Theorem 5.4(a) rules out every controller with lookahead d, of any memory. ($\Leftarrow$) Theorem 5.4(b) gives a d-coherent kill witness with shape in $\mathfrak{S}^{\flat}_K$. Let φ be its pairs with different words and τ its typing. At each OD leaf the distance exceeds H and the output is nonempty, so some rank c has $|b_A(c) - b_B(c)| > H$; take $c^A = c^B = c$. Lemma 6.2 gives a vector satisfying the guards. The last equivalence is Theorem 4.4. □

Lemma 6.4 (no all-DEAD witness on one word)**.** *A case with no OD leaf and $\varphi = \emptyset$ has no accepting walk.*

Proof. Otherwise Lemma 6.2 gives a kill witness with every leaf DEAD and one common word $u \in \mathrm{dom}(B)$. Let γ be an accepting run of B on u. We follow γ down the tree. Start at any leaf. Given a leaf λ whose edges agree with γ before position $p \leq |u|$, the edge $\gamma(p)$ is live for u at p, because γ continues from it to acceptance. All leaves read u, so if no leaf of $\Lambda(\lambda,p)$ took $\gamma(p)$, (K3) would make it dead. Some leaf of $\Lambda(\lambda,p)$ therefore takes it and agrees with γ through p. After position $|u|$ we hold a leaf with $\beta_\lambda = \gamma$, an accepting run. This contradicts DEAD. □

So under $\mathbf{G}^{\flat}$ every obstruction to acceptance at all lookaheads has a pair in φ whose gap can be pumped.

6.3. **Two bounds from integer programming.** Lemma 6.5 makes each $V^{\flat}_{\mathcal{T},\varphi,\tau}$ a finite union of sets $a + \mathbb{N}\cdot V$ with small a and V. Lemma 6.6 bounds one coordinate over the integer points of such a set or produces a ray.

Lemma 6.5 (cycle decomposition)**.** *Let $\mathcal{G}$ be a finite directed multigraph with $Z \geq 1$ vertices, an initial vertex ι, a set F of final vertices and edge weights ψ in $\mathbb{Z}^q$. Write $\psi(\sigma)$ for the weight of a walk σ and $V(\sigma)$ for the set of vertices it visits. A simple cycle is a closed walk of positive length whose*

vertices are pairwise distinct except for its two endpoints. Then the weights of walks from ι to F form the set

$$\bigcup_{\pi} \big(\psi(\pi) + \mathbb{N} \cdot \mathrm{Cyc}(\pi)\big),$$

where π ranges over walks from ι to F of length at most $Z(Z-1)$, $\mathrm{Cyc}(\pi)$ is the set of weights of simple cycles with all vertices in $V(\pi)$, and $\mathbb{N} \cdot V$ is the additive monoid generated by V. If every edge weight has norm at most W, then $|\psi(\pi)|_\infty \le Z^2 W$, every element of $\mathrm{Cyc}(\pi)$ has norm at most ZW, and $\mathrm{Cyc}(\pi)$ has at most $(2ZW+1)^q$ elements.

Proof sketch. Mark position 0 of a walk and the last visit to each vertex, at most $Z+1$ marks. Inside each of the at most Z segments, repeatedly cut out the closed subwalk from the first repeated vertex to its next occurrence. It is a simple cycle on vertices of the walk, and no last-visit position lies inside it, so every visited vertex survives. What remains is a walk π of length at most $Z(Z-1)$ with $V(\pi) = V(\sigma)$, and each removed cycle can be spliced back at a vertex it shares with π. Several initial vertices give a finite union. Appendix C has the details. □

Lemma 6.6 (vertex bound)**.** *Let $M \in \mathbb{Z}^{m \times N}$ and $b \in \mathbb{Z}^m$ have entries of absolute value at most $\beta \ge 1$, let $P = \{x \in \mathbb{R}^N_{\ge 0} : Mx = b\}$ be nonempty, and fix a coordinate c. If x_c is bounded above on P, then $\max_P x_c \le m!\,\beta^m$. Otherwise there is $r \in \mathbb{Z}^N_{\ge 0}$ with $Mr = 0$ and $r_c \ge 1$.*

Proof sketch. In the bounded case the maximum is attained at a basic feasible solution. Its support indexes at most m independent columns, and Cramer's rule on a nonsingular square submatrix gives $x_c = \det N_c / \det N$ with $|\det N| \ge 1$ and $|\det N_c| \le m!\,\beta^m$. In the unbounded case P is a polytope plus the cone $\{r \ge 0 : Mr = 0\}$, which finitely many integer vectors generate, and one of them has $r_c > 0$. Appendix C. □

6.4. **Cutoff and the proof of Theorem 3.1.**

Proposition 6.7 (cutoff)**.** *Let Z, W and m bound the state counts, the edge-weight norms and the numbers of coordinates of all cases $\mathcal{W}^\flat_{\mathcal{T},\varphi,\tau}$ that have an accepting walk; $m \le 2N + \binom{N}{2}$ with $N = 2^K$ always works. Put*

$$T^\flat_* = m!\,(Z^2W+1)^m + 1, \qquad H^\flat_*(d) = m!\,(Z^2W + d + 1)^m + 1.$$

Then $J^\flat \iff \mathrm{GOOD}^\flat(T^\flat_*, T^\flat_*)$, and for each d, $Q^\flat_d \iff \mathrm{GOOD}^\flat(d, H^\flat_*(d))$. The same holds with arbitrary memory.

Proof. $\mathrm{GOOD}^\flat(T^\flat_*, T^\flat_*)$ gives $J^\flat$ by definition. Suppose it fails. By Theorem 6.3 some case and some $v \in V^\flat_{\mathcal{T},\varphi,\tau}$ satisfy the guards of $\mathrm{BAD}^\flat(T^\flat_*, T^\flat_*)$. Accepting states have no outgoing rows, so Lemma 6.5 applies to the automaton, and v lies in a piece $a + \mathbb{N} \cdot V$ with $|a|_\infty \le Z^2W$ and $|v'|_\infty \le ZW$ for every $v' \in V$. Every point of this piece is the weight of an accepting walk.

Write the guards on the piece as a linear system in unknowns $x \in \mathbb{N}^V$, $T \in \mathbb{N}$ and slacks $\sigma \in \mathbb{N}$. For each OD leaf take the rows $(a+\sum_{v'} x_{v'v'})_{R_\lambda} = 0$ and $(a + \sum_{v'} x_{v'v'})_{G_\lambda} - T - \sigma_\lambda = 1$, and for each $\pi \in \varphi$ the row $(a + \sum_{v'} x_{v'v'})_{E_\pi} - T - \sigma_\pi = 0$. These are at most m rows. Once a moves to the right-hand side, all entries have absolute value at most Z^2W+1. The vector v gives an integer solution with $T = T^\flat_* > m!\,(Z^2W+1)^m$. By Lemma 6.6, T is unbounded on the real solutions, so there is an integer ray $r \geq 0$ with $Mr = 0$ and $r_T \geq 1$. Adding multiples of r keeps every unknown a natural number and sends T to infinity. Each such solution is a point of the piece that satisfies the guards of $\mathrm{BAD}^\flat(T,T)$, so $\mathrm{BAD}^\flat(T,T)$ holds for arbitrarily large T and, by downward closure, for every T. Theorem 6.3 then gives $\neg\mathrm{GOOD}^\flat(T,T)$ for every T, and monotonicity (Lemma 2.6) gives $\neg J^\flat$.

A case with no OD leaf and $\varphi = \emptyset$ has no guard. By Lemma 6.4 it has no accepting walk, so it contributes no piece. For $Q^\flat_d$ fix d. The E-rows become $(\cdot)_{E_\pi} - \sigma_\pi = d$, only the G-rows involve T, and the entries are bounded by Z^2W+d+1. The same argument gives $H^\flat_*(d)$. Theorem 6.3 covers arbitrary memory as well. □

The ray is the whole content of a negative answer. Appendix C makes it explicit: a finite multicycle of one witness automaton with an integer ray is a pumping certificate for $\neg J^\flat$, and the least d with $Q^\flat_d$ and, for each d, the least H with $\mathrm{GOOD}^\flat(d,H)$ are computable.

Proof of Theorem 3.1. Compute K first. The degree of ambiguity of B as a finite automaton is computable, and Weber and Seidl [WS91] and Kiefer, Mazowiecki, Ryzhikov and Wiśniewski [KMRW26] bound it when it is finite. Compute Z and W from the construction of the automata $\mathcal{W}^\flat$, and from them $T^\flat_*$, or $H^\flat_*(d)$ for a given d. Build the finite reachability game $G^\flat_{T^\flat_*,T^\flat_*}$, or $G^\flat_{d,H^\flat_*(d)}$, of Theorem 4.4 and solve it by attractor computation. Proposition 6.7 with Theorem 4.4 makes the answer that of $J^\flat$, or of $Q^\flat_d$. On a positive instance a positional winning strategy is a controller with lookahead $T^\flat_*$ and distance $T^\flat_*$ (Theorem 4.4). Proposition 4.5(a) gives one with memory $(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R^{\mathrm{od}}_{T^\flat_*})$; it comes from the quotient game of Proposition 4.5, whose safe sets are inclusions of regular languages and hence computable. The arbitrary-memory answers agree by Theorem 4.4. The clause on a regular obligation domain O, without the memory bound, reduces to $O = \mathrm{dom}(B)$ by a product of B with a DFA for O (Lemma H.3, Appendix H). Theorem B is the case of a **G**-instance (Section 3). □

Remark 6.8 (dropping word agreement). Theorem B$^\natural$ (Appendix H) removes the word part of (R4). Accepting runs of B may output any word, and the controller must produce A's word with $\mathrm{od} \leq H$ on a regular obligation domain $O \subseteq \mathrm{dom}(A) \cap \mathrm{dom}(B)$. Appendix H proves it by editing Section 5 and this section item by item.

7. Linear ambiguity is undecidable

Theorem C is the negative half of the frontier. We prove it here, and then show that nothing computable bounds the memory or the distance that a positive instance needs.

Theorem 7.1 (Theorem 3.4, restated)**.** *There are a deterministic copyless SST A_* over the input alphabet $\{0,1\}$, with 16 states and 8 registers, and a computable map $M \mapsto B_M$ from two-counter machines to copyless SSTs with 16 registers, such that for every M:*

(1) (A_*, B_M) satisfies **G***;*
(2) B_M has at most $|v|+1$ runs on every input v;
(3) on every input, the runs of B_M produce at most two annotated outputs, and one of them is the output of A_*;
(4) if M halts from zero counters in L steps, then $\mathrm{GOOD}(0, 2L{+}1)$ *holds; if for some d a controller with lookahead d and arbitrary memory keeps* od *bounded on all inputs, then M halts.*

Hence Q_d, for each fixed d, and J are RE*-complete, already on instances with these four properties.*

The counter arithmetic lives in the register updates. The controller still only picks edges of B_M. Both machines keep one pair of registers X_κ, Y_κ per coordinate κ. X_κ appends fresh letters on the right and Y_κ prepends them on the left. The reference adds two letters to each. B_M adds $2+w$ and $2-w$ for an offset $w \in \{-1,0,1\}$ that the edge chooses. The pair keeps its total length, so the two outputs agree as words, while the junction between X_κ and Y_κ moves by the accumulated offset. That accumulated offset is a counter value. A probe later pushes letters born at the probe into both registers, and a shifted junction then pairs an old letter with a probe letter. The origin distance reads off the counter.

Choosing online is what hurts. Offline, a run that mirrors the reference always exists. Online, the controller has to guess every zero test before the environment confirms it, and padding keeps the confirmation beyond any lookahead window. A halting machine yields a controller with zero lookahead and finite memory. A non-halting machine lets the environment pile up errors past every bound. The game follows the counter-tree pattern of Exibard, Filiot and Jecker [EFJ18].

7.1. **The source machine.** A two-counter machine M has states Q_M, an initial state q_I and a halting state q_F. Every $q \neq q_F$ carries one instruction with $i \in \{1,2\}$: $\mathrm{inc}(i, q')$ adds one to c_i and moves to q'; $\mathrm{jz}(i, q_z, q_d)$ moves to q_z if $c_i = 0$, and otherwise subtracts one from c_i and moves to q_d. We ask whether M halts from $(q_I, 0, 0)$. When it does, $\gamma_0, \ldots, \gamma_L$ is its run, $\gamma_k = (q_k, c_1^k, c_2^k)$, and $c^* \leq L$ is the largest counter value along it.

Lemma 7.2. *Halting from zero counters, for machines of this form, is* RE*-complete under many-one reductions.*

letter x	rows other than the identity
$\mathtt{W}, \mathtt{st}, \mathtt{pass}, \mathtt{prb}, \mathtt{ec}_0, \mathtt{ec}_1$	none
$\mathtt{zr}(i,s)$	$\mathrm{row}(E) = [(E,+),(i,s)]$, $\mathrm{row}(i) = [\,]$
$\mathtt{ab}(i)$	$\mathrm{row}(E) = [(E,+),(i,+)]$, $\mathrm{row}(1) = \mathrm{row}(2) = \mathrm{row}(T) = [\,]$
$\mathtt{rst}$	$\mathrm{row}(1) = \mathrm{row}(2) = \mathrm{row}(T) = [\,]$

TABLE 2. Row lists of the letters of Σ_0.

Proof sketch. Membership is by simulation. For hardness we embed the machines of Dudenhefner [Dud23, Def. 2.1], whose termination from a configuration $(0,(a,b))$ is RE-hard [Dud23, Lemma 2.4]. A chain of $a+b$ increments loads the initial counters; an instruction inc_c at index k becomes $\mathrm{inc}(c+1, q_{k+1})$, and $\mathrm{dec}_c\, j$ becomes $\mathrm{jz}(c+1, q_{k+1}, q_j)$; every index $\geq \ell$ is q_F. Details are in Appendix D. □

7.2. **Register pairs.** The coordinates are $\mathcal{K} = \{1, 2, E, T\}$: the two counters, an error account E and a clock T. The input letters are the 13 letters of

$$\Sigma_0 = \{\mathtt{W},\ \mathtt{st},\ \mathtt{pass},\ \mathtt{rst},\ \mathtt{prb},\ \mathtt{zr}(i,\pm),\ \mathtt{ab}(i)\ (i=1,2),\ \mathtt{ec}_0,\ \mathtt{ec}_1\}.$$

Each letter x has a *row list* $\mathrm{row}_x(\kappa)$ for every coordinate κ, a list of pairs (source coordinate, sign). Rows absent from Table 2 are the identity $[(\kappa,+)]$. Every source occurs in at most one row. This is what keeps the update copyless.

A row list acts linearly on $z \in \mathbb{Z}^{\mathcal{K}}$ by $(\mu_x z)_\kappa = \sum_{(\kappa',s)\in \mathrm{row}_x(\kappa)} s\, z_{\kappa'}$. So $\mathtt{zr}(i,s)$ adds $s z_i$ to z_E and clears z_i; $\mathtt{ab}(i)$ adds z_i to z_E and clears z_1, z_2, z_T; $\mathtt{rst}$ clears z_1, z_2, z_T.

The update $U(x,w)$. Both machines have registers X_κ, Y_κ for $\kappa \in \mathcal{K}$. Fix an offset $w \in \{-1,0,1\}^{\mathcal{K}}$ and write $\mathrm{row}_x(\kappa) = [(\kappa_1,s_1),\dots,(\kappa_m,s_m)]$. Let S_j^X be X_{κ_j} if $s_j = +$ and Y_{κ_j} if $s_j = -$, and let S_j^Y be the other register of that pair. The update $U(x,w)$ sets

$$X_\kappa := S_1^X \cdots S_m^X\, a^{2+w_\kappa}, \qquad Y_\kappa := a^{2-w_\kappa}\, S_m^Y \cdots S_1^Y.$$

X_κ appends on the right. Y_κ prepends on the left, with the sources in reverse order. The crossing letter $\mathtt{zr}(i,-)$ therefore sends Y_i into X_E and X_i into Y_E, and a cleared register receives only its fresh letters. Each source pair is used once, one register on each side, so $U(x,w)$ is copyless. Since $|w_\kappa| \leq 1$, every update puts at least one fresh letter into every register. Both machines end with $\Phi = X_1Y_1\, X_2Y_2\, X_EY_E\, X_TY_T$.

The reference A_M has one state and applies $U(x,0)$ on every letter x. Figure 1 shows one pair after four steps.

Remark 7.3 (blocks). Replace every fresh a by a fixed block $p \in \Gamma^+$, all letters of one copy sharing a birth. Nothing below changes: corresponding blocks have equal length in both machines, and od is computed per block. $p = ab$ gives a binary output alphabet. Lengths are then counted in blocks.

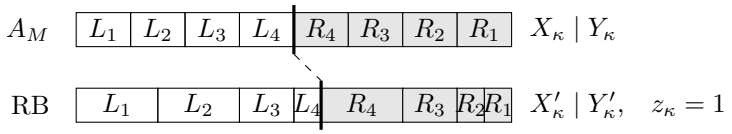


FIGURE 1. One pair κ after four steps with offsets $+1, +1, 0, -1$. White pieces L_t were appended to X_κ, grey pieces R_t were prepended to Y_κ; piece t is born at step t. Both rows hold the same pieces in the same order and have the same length. The junction between X_κ and Y_κ sits z_κ letters further right in RB. A probe inserts fresh pieces at this junction, so a shift pairs an old letter with a probe letter.

state	letter	edges
every state $\sigma \neq$ CAN	`W`	$(0, \sigma)$
FIRST	`st`	$(0, \mathrm{P}(\mathrm{FIRST}, 0))$, $(0, \mathrm{P}(\mathrm{FIRST}, 1))$
$\mathrm{C}(q)$ with $\mathrm{inc}(i, q')$	`st`	$(e_i + e_T,\ \mathrm{C}(q'))$
$\mathrm{C}(q)$ with $\mathrm{jz}(i, q_z, q_d)$	`st`	$(0, \mathrm{P}(\mathrm{C}(q), 0))$, $(0, \mathrm{P}(\mathrm{C}(q), 1))$
ACC	`rst`	$(0,\ \mathrm{F}(q_I))$
$\mathrm{Z}(i, q)$	`zr`$(i, +)$, `zr`$(i, -)$	$(e_T,\ \mathrm{F}(q))$
$\mathrm{D}(i, q)$	`pass`	$(e_T,\ \mathrm{C}(q))$
$\mathrm{D}(i, q)$	`ab`(i)	$(0,\ \mathrm{F}(q_I))$
$\mathrm{F}(q)$	`st`	$(0, \mathrm{P}(\mathrm{F}(q), 0))$, $(0, \mathrm{P}(\mathrm{F}(q), 1))$
$\mathrm{C}(q)$, Z, D, F	`prb`	$(0,\ \mathrm{PRB})$
PRB	`prb`	$(0,\ \mathrm{PRB})$
$\mathrm{P}(\vartheta, b)$	`ec`$_b$	$\mathrm{res}(\vartheta, b)$
CAN	any	CAN

TABLE 3. Edges of B_M; e_κ is the unit offset at κ.

7.3. **The game machine B_M.**

Registers. Two disjoint groups: the game registers $\mathrm{RB} = \{X'_\kappa, Y'_\kappa\}$ and the mirror registers $\mathrm{PA} = \{X''_\kappa, Y''_\kappa\}$.

States. FIRST is initial. The game phases are $\mathrm{C}(q)$ for $q \in Q_M$, with $\mathrm{ACC} := \mathrm{C}(q_F)$, and $\mathrm{Z}(i, q)$, $\mathrm{D}(i, q)$, $\mathrm{F}(q)$. A *choice phase* ϑ is FIRST, a state $\mathrm{C}(q)$ whose instruction is a jz, or a state $\mathrm{F}(q)$; each has two pending states $\mathrm{P}(\vartheta, 0)$ and $\mathrm{P}(\vartheta, 1)$. Finally PRB and the sink CAN. All states are final.

Edges. Table 3 lists the edges as (offset, target). Every pair (state other than CAN, letter) that it omits has one edge, to CAN. A choice letter `st` in a choice phase has two edges, both with offset 0, and the choice takes effect at the echo letter `ec`$_b$ through Table 4.

Register actions and final expressions. On every edge on letter x, PA receives $U(x, 0)$, the update of A_M. On an edge with offset w into a state other than CAN, RB receives $U(x, w)$. On an edge into or out of CAN, RB is

ϑ	$b=0$	$b=1$
FIRST	$(0,\ \mathrm{C}(q_I))$	$(0,\ \mathrm{C}(q_I))$
$\mathrm{C}(q)$ with $\mathrm{jz}(i,q_z,q_d)$	ZERO: $(e_T,\ \mathrm{Z}(i,q_z))$	DEC: $(-e_i+e_T,\ \mathrm{D}(i,q_d))$
$\mathrm{F}(q)$	FINISH: $(e_T,\ \mathrm{C}(q))$	SPEND: $(-e_E+e_T,\ \mathrm{F}(q))$

TABLE 4. Choice resolution $\mathrm{res}(\vartheta,b)$, applied at the echo letter.

unchanged. The final expression is Φ on PA in CAN and Φ on RB everywhere else.

Three features carry the proof. The choice letter is neutral and the choice-dependent offset is applied at the echo, so two sibling runs have identical RB contents while they are pending. Every edge outside CAN applies the row lists of its letter, so RB undergoes the same register-level moves as A_M. And `prb` is legal in the game phases and in PRB only; after a probe, only `prb` and `W` avoid CAN.

7.4. **Imbalance, promise, ambiguity.** For a run ρ of B_M that has not entered CAN, put $z(0)=0$ and $z(t)=\mu_{x_t}z(t-1)+w_t$, where w_t is the offset of its t-th edge. This is the *imbalance* of ρ.

Lemma 7.4 (imbalance)**.** *Let ρ be such a run on v and $t\le|v|$. In A_M, $|X_\kappa|=|Y_\kappa|=:l_\kappa(t)$ for every κ. In ρ's* RB*, $|X'_\kappa|=l_\kappa(t)+z_\kappa(t)$ and $|Y'_\kappa|=l_\kappa(t)-z_\kappa(t)$.*

Proof. Induction on t. In A_M both registers of κ receive $\sum_{\mathrm{row}}l_{\kappa'}+2$ letters. In RB, a source with sign $+$ contributes $l_{\kappa'}+z_{\kappa'}$ to X'_κ, a source with sign $-$ contributes $l_{\kappa'}-z_{\kappa'}$, and $2+w_\kappa$ fresh letters follow. The total is $l_\kappa(t)+(\mu_x z+w)_\kappa$. Y'_κ is symmetric. □

Lemma 7.5. (A_M,B_M) *satisfies* **G**.

Proof. A_M has one state and one edge per letter. Every state of B_M has an edge on every letter, by the default to CAN. $U(x,w)$ is copyless, and the edges of B_M act on the disjoint groups RB and PA separately; the fresh constant may be repeated in both groups. Each final expression uses every register of one group once and contains no letters, the initial state is unique, and all states are final. For word agreement the output alphabet is unary, so lengths suffice. In CAN, PA has received $U(x_t,0)$ at every step and equals A_M's registers as annotated words. Otherwise $\sum_\kappa(|X'_\kappa|+|Y'_\kappa|)=\sum_\kappa 2l_\kappa$ by Lemma 7.4. □

Lemma 7.6 (runs and offline zero)**.** *For every input v: (i) B_M has at most $|v|+1$ runs on v; (ii) all runs outside* CAN *have the same annotated output; (iii) some run has the annotated output of A_M.*

Proof. We claim that after every prefix the runs outside CAN are at most one run, or exactly two *siblings*. Siblings share their prefix up to their last fork, the last `st` read in a choice phase ϑ. After it they sit in $\mathrm{P}(\vartheta,0)$ and $\mathrm{P}(\vartheta,1)$ and perform identical updates, so their RB contents agree, until an

echo or an illegal letter resolves them. Induction on the prefix: a single run outside CAN and outside the pending states has one edge on every letter, except a choice letter in a choice phase, where both edges carry $U(\mathtt{st}, 0)$ and lead to the two pending states. From siblings, $\mathtt{W}$ keeps both with the same update, $\mathtt{ec}_b$ keeps sibling b and sends sibling $1-b$ to CAN, and every other letter sends both to CAN. CAN runs never branch.

(i) At most one run branches at each position, into two. (ii) Siblings have identical RB, and every state outside CAN outputs Φ on RB. (iii) If some run is in CAN its output is A_M's, by Lemma 7.5. If no run enters CAN, then $v \in \mathtt{W}^* \cup \mathtt{W}^*\mathtt{st}\mathtt{W}^*$: FIRST survives only $\mathtt{W}$ and $\mathtt{st}$, and siblings survive together only on $\mathtt{W}$. On such words RB has received exactly the updates $U(\mathtt{W}, 0)$ and $U(\mathtt{st}, 0)$ that A_M received. □

Call the fresh letters that one update puts into one register a *piece*: L_t^κ into X_κ and R_t^κ into Y_κ, born at t. A piece has length 2 in A_M and length $2 \pm w_\kappa(t) \in \{1, 2, 3\}$ in RB. Registers are only concatenated, so pieces are never split.

Lemma 7.7 (common piece order)**.** *Let ρ be a run of B_M that has not entered* CAN*. At every time, each register of A_M and the corresponding register of ρ's* RB *contain the same sequence of pieces. So the two outputs are concatenations of the same pieces in the same order.*

Proof. At step t both machines apply the row lists of x_t, through $U(x_t, 0)$ and $U(x_t, w_t)$. The row lists alone determine how the sequences of pieces evolve. □

In particular both pieces of one step lie in the same pair, so every pair has the same total length and the same multiset of births in both machines.

7.5. **Probes expose the imbalance.**

Lemma 7.8 (lower bridge)**.** *Let ρ be a run of B_M on h that ends in a state* C(q)*,* Z*,* D *or* F*, with imbalance $z = z(|h|)$, and let $2m \geq |z|_\infty$. Then ρ has a unique extension on $h \cdot \mathtt{prb}^m$, and*

$$\mathrm{od}\big(\rho_{A_M}(h\,\mathtt{prb}^m),\ \rho \cdot \mathtt{prb}^m\big) \ \geq\ \max_\kappa \lceil |z_\kappa|/2 \rceil.$$

Proof. $\mathtt{prb}$ leads to PRB and loops there, with the neutral update $U(\mathtt{prb}, 0)$ in both machines. So the extension is unique. Let $t = |h|$, fix κ with $z := z_\kappa \neq 0$, and let $x_A = l_\kappa(t)$. By Lemma 7.7 the same number P of output letters precedes pair κ in both outputs. In A_M the pair reads $X_\kappa(h)$, then $2m$ probe letters born $t+1, t+1, \ldots, t+m, t+m$, then $2m$ probe letters born $t+m, \ldots, t+1$, then $Y_\kappa(h)$. In ρ it reads the same, with $X'_\kappa(h)$ of length $x_A + z$ and $Y'_\kappa(h)$ of length $x_A - z$. If $z > 0$, take rank $P + x_A + z$. In ρ it is the last letter of $X'_\kappa(h)$, born at most t. In A_M it is the z-th probe letter, born $t + \lceil z/2 \rceil$, because $z \leq 2m$. If $z < 0$, take the $(x_A + |z|)$-th letter from the right end of the pair: in ρ it is the first letter of $Y'_\kappa(h)$, born at most t, and in A_M it is the $|z|$-th probe letter from the right, born $t + \lceil |z|/2 \rceil$. □

7.6. **A halting machine gives a controller.** Assume M halts, with run $\gamma_0, \dots, \gamma_L$. The controller τ has memory $(k, \pi) \in \{0, \dots, L\} \times \{0, \dots, c^*\}$, initially $(0, 0)$, and sees the state of B_M and the current letter. It has no lookahead.

- *Choices.* At FIRST on `st`: P(FIRST, 0). At a jz-state C(q) on `st`: $b = 0$ (ZERO) iff $c_i^k = 0$. At F(q) on `st`: $b = 1$ (SPEND) iff $\pi > 0$.
- *Memory*, updated on edges into states other than CAN: res(FIRST, $\cdot$) sets $(k, \pi) := (0, 0)$; an inc edge or res(C(q), $\cdot$) sets $k := k + 1$; `ab`(i) from D(i, q) sets $\pi := c_i^k$ and $k := 0$; SPEND sets $\pi := \max(\pi - 1, 0)$; `rst` sets $(k, \pi) := (0, 0)$. Nothing else changes the memory.

τ is total: the choices are legal edges for every (k, π), every other pair (state, letter) has one edge, and the only updates that could leave the memory range are $k := k + 1$ at $k = L$, where k is kept, and SPEND at $\pi = 0$, where π is kept. On plays of τ neither case occurs. In the first, (I1) and (I2) below make the state $q_L = q_F$, which has no inc edge and no resolution. The second needs SPEND at $\pi = 0$, and τ spends only at $\pi > 0$.

Lemma 7.9 (good-play invariants)**.** *Along every input, while the run of τ avoids* CAN*:*

(I1) in C(q) *and* P(C(q), $\cdot$)*:* $(q, z_1, z_2) = \gamma_k$*,* $z_E = 0$ *and* $\pi = 0$*; in* Z(i, q) *and* D(i, q) *the same holds, with* $z_i \geq 0$*, and* $z_i = 0$ *in* Z*;*

(I2) in F(q) *and* P(F(q), $\cdot$)*:* $(q, z_1, z_2) = \gamma_k$ *and* $z_E = \pi \in [0, c^*]$*;*

(I3) $z_1, z_2, z_E \in [0, c^*]$ *and* $0 \leq z_T \leq 4L + 1$ *at all times, so* $\bar{Z} := \max_t |z(t)|_\infty \leq 4L + 1$*;*

(I4) every import happens with $z_E = 0$*, and every* `zr` *import with* $z_i = 0$*.*

Proof. Induction over the edges of the run. In FIRST and P(FIRST, $\cdot$), $z = 0$ and $(k, \pi) = (0, 0)$. A state C(q) is entered in four ways: from res(FIRST) with $z = 0$, $k = 0$ and $\gamma_0 = (q_I, 0, 0)$; by an inc edge, whose offset $e_i + e_T$ and update $k := k+1$ match γ_{k+1}; by `pass` with offset e_T and k unchanged, carrying (I1) over from D; or by FINISH, which τ takes only at $\pi = 0$, hence $z_E = 0$, and which keeps $(q, z_1, z_2) = \gamma_k$ by (I2). The memory π changes only at `ab`, SPEND, `rst` and res(FIRST), so $\pi = 0$ throughout C, Z and D. A choice letter applies $U(\texttt{st}, 0)$ and changes neither z nor k.

The choices follow γ. ZERO is taken iff $c_i^k = 0 = z_i$; it enters Z(i, q_z) with offset e_T and $k := k + 1$, and $\gamma_{k+1} = (q_z, c_1^k, c_2^k)$. DEC is taken iff $c_i^k > 0$; it enters D(i, q_d) with offset $-e_i + e_T$ and $k := k+1$, so $z_i = c_i^k - 1 = c_i^{k+1} \geq 0$. The letter `zr` is legal only in Z, where $z_i = 0$ and $z_E = 0$; both stay 0, and F(q) receives γ_k. The letter `ab`(i) is legal only in D, where $z_E = 0$; it sets $z_E := z_i = c_i^k = \pi$, clears z_1, z_2, z_T and sets $k := 0$, which is γ_0 in F(q_I). SPEND, taken iff $\pi > 0$, lowers z_E and π together. `rst` is legal only at ACC, where $z_E = 0$; it clears z_1, z_2, z_T, resets the memory and enters F(q_I) with γ_0. `W`, `prb` and PRB change nothing. This proves (I1), (I2), (I4), and (I3) for z_1, z_2, z_E.

The clock remains. z_T starts at 0 and is cleared at `ab` and `rst`. Since the last clear it gained at most c^*+1 through the SPENDs and the FINISH right after the clear, plus at most 3 per step of M taken from γ_0: an inc gives 1, ZERO gives 3 (resolution, `zr`, FINISH), DEC gives 2 (resolution, `pass`). At most L steps occur, because $k \le L$ and ACC admits no increment. So $z_T \le c^* + 1 + 3L \le 4L + 1$. □

The upper bridge rests on two facts from Appendix D. The *transport lemma* (Lemma D.1): if two outputs concatenate the same pieces $\pi_1, \dots, \pi_N$, of length 2 in the first and λ_k in the second, let $\mathrm{dis}_k = \sum_{k'<k}(\lambda_{k'} - 2)$ be the displacement of the start of piece k, call a junction $(k-1, k)$ *live* if $\mathrm{dis}_k \neq 0$, let $\mathrm{dis}^* = \max_k |\mathrm{dis}_k|$ and let g be the largest birth jump across a live junction; then $\mathrm{od} \le g\lceil \mathrm{dis}^*/2 \rceil$. The *lifetime* of a counter runs from its last clear to the current step. An import step reads `zr` or `ab`(i) and moves the pair (X_i, Y_i) into the accumulator as one block, an *atom*. Its sign is $+$ if X_i lands in X_E and $-$ if Y_i does. The atom's *entry* is the junction where it starts in X_E, together with the mirror junction where its partner Y_i ends in Y_E. Its *exit* is the pair of junctions at the other ends. The *junction table* (Lemma D.2): a birth jump above 1 occurs only at a pair boundary, which has displacement 0; at the entry of an atom imported with sign $+$, where the displacement is $z_E(s-1)$ for the import step s; and at the exit of an atom imported with sign $-$, where it is $z_E(s-1) - z_i(s-1)$.

Lemma 7.10 (upper bridge)**.** *For every input v,* $\mathrm{od}(\rho_{A_M}(v), \rho_\tau(v)) \le \lceil \bar{Z}/2 \rceil \le 2L + 1$.

Proof. If the run of τ is in CAN the outputs coincide. Otherwise Lemma 7.7 puts the two outputs in the setting of the transport lemma. By (I4), every import has $z_E(s-1) = 0$, and every `zr` import has $z_i(s-1) = 0$ as well. So the two junction types with a large birth jump have displacement 0 and are never live. Every live junction has birth jump at most 1, and $g \le 1$. For dis^*, Lemma D.2 writes every displacement as a value of z along the run, or as a difference of two values of the same coordinate within one counter lifetime, once the z_E terms removed by (I4) are gone. All these values lie in $[0, \bar{Z}]$ by (I3), so $\mathrm{dis}^* \le \bar{Z}$. The transport lemma gives $\mathrm{od} \le \lceil \bar{Z}/2 \rceil$, and $\lceil (4L+1)/2 \rceil = 2L + 1$. □

Corollary 7.11. *If M halts in L steps, then* $\mathrm{GOOD}(0, 2L+1)$ *holds, and by Lemma 2.6 so does Q_d for every d.*

Remark 7.12. Bounded imbalance alone does not bound the origin distance. Invariant (I4) is what the upper bridge uses.

7.7. **Bounded distance forces halting.** Write Q_d^∞ for the variant of Q_d in which the controller may use arbitrary memory.

Lemma 7.13 (adaptive environment)**.** *Fix d and a controller σ with lookahead d and arbitrary memory. Build an infinite word v_∞ block by block, from the state ϕ and imbalance z of σ's run after the blocks so far:*

- *if ϕ is a choice phase, the block is* $\mathtt{st}\,\mathtt{W}^d\,\mathtt{ec}_b$*, where b is the choice of σ at that* st*;*
- *otherwise the block is one letter:* st *in an* inc*-state* $\mathrm{C}(q)$*;* $\mathtt{zr}(i,-)$ *in* $\mathrm{Z}(i,q)$ *if $z_i > 0$, else* $\mathtt{zr}(i,+)$*;* pass *in* $\mathrm{D}(i,q)$ *if $z_i \geq 0$, else* $\mathtt{ab}(i)$*;* rst *in* ACC*.*

Then: (a) v_∞ is well defined and σ's run on it never enters CAN*; (b) for every block boundary h after the first block and every word u, σ's run on hu agrees on its first $|h|$ edges with its run along v_∞; (c) if σ achieves* $\mathrm{od} \leq H$ *on all inputs, then $|z(h)|_\infty \leq 2H$ at every such block boundary.*

Proof. (a) The choice of σ at the st in position i depends on its memory before i and on the window $\mathtt{st}\,\mathtt{W}^d$, which shows no end marker since more letters follow. The memory before i depends on windows at positions below i, which reach at most position $i - 1 + d$, inside the fixed part $\mathtt{st}\,\mathtt{W}^{d-1}$. So b is determined before $\mathtt{ec}_b$ is written. Every letter is legal in its phase: W is legal in pending states and $\mathtt{ec}_b$ matches the surviving sibling.

(b) Every choice position i in h is followed inside h by $\mathtt{W}^d\mathtt{ec}_b$, so $i+d < |h|$. The choices in h depend only on windows at positions $\leq i$, which lie inside h, show no end marker in hu either, and agree in hu and v_∞. All other edges are forced, since B_M has a unique edge there. The memory near the end of h may differ, but no choice in h depends on it.

(c) h ends in a game phase outside the pending states, so take $u = \mathtt{prb}^m$ with $2m \geq |z(h)|_\infty$ and apply Lemma 7.8 through (b). □

Lemma 7.14. *If $|z|_\infty \leq Z_0$ at every block boundary of v_∞, then M halts.*

Proof. Counters stay nonnegative at C phases. An inc adds 1; a correct DEC leaves $z_i \geq 0$ and is followed by pass; a DEC at 0 gives $z_i = -1$, which triggers ab and clears the counters; ZERO is followed by zr, which clears z_i; rst clears.

z_E never increases. SPEND subtracts 1; $\mathtt{zr}(i,s)$ adds $sz_i = -|z_i|$; $\mathtt{ab}(i)$ is sent only when $z_i = -1$ and adds -1; nothing else touches z_E.

Wrong choices are charged by the forced block that follows them. A wrong ZERO, with $z_i > 0$, enters $\mathrm{Z}(i,\cdot)$ with z_E unchanged; the next block is $\mathtt{zr}(i,-)$, which lowers z_E by $z_i \geq 1$. A wrong DEC, with $z_i = 0$, leaves $z_i = -1$ in $\mathrm{D}(i,\cdot)$; the next block is $\mathtt{ab}(i)$, which lowers z_E by 1. A SPEND lowers z_E by 1 in its own block. Since $z_E(0) = 0$, z_E never increases and $|z_E| \leq Z_0$ at block boundaries, there are at most Z_0 wrong choices and SPENDs in total.

After the penalty block of the last wrong choice and after the last SPEND, the simulation is faithful: every choice is correct, every F phase is FINISHed at once, ab is never sent, so z_T is cleared only by rst; the values (q, z_1, z_2) at C phases follow the transition function of M; and each step of M adds at least 1 to z_T, through the inc edge or the resolution edge. Since $z_T \leq Z_0$, ACC is reached within $Z_0 + 1$ further steps. Then rst and FINISH lead to $\mathrm{C}(q_I)$ with $z_1 = z_2 = 0$, the initial configuration of M. From there the

faithful simulation must reach ACC again within $Z_0 + 1$ steps, or z_T would exceed Z_0. So M halts from $(q_I, 0, 0)$. □

Corollary 7.15. *For every d, if Q_d^∞ holds then M halts.*

Proof. Lemma 7.13(c), then Lemma 7.14 with $Z_0 = 2H$. □

7.8. **Binary framing and the fixed reference.** Theorem 7.1 is stated over $\{0, 1\}$. Encode `W` as the bit 0 and every other letter as 1 followed by a fixed 4-bit code. A deterministic decoder with 16 states reads a bit as `W` unless it completes a frame, in which case it reads the frame's letter; an unused code is an illegal letter for B_M and sends it to CAN. The reference A_{bin} has the decoder as its state set and applies $U(e, 0)$ for the event e of each bit. The machine B_{bin} is the product of B_M with the decoder. Every bit is one input position, so births are bit positions.

Lemma 7.16. *Everything in Sections 7.4–7.7 holds for $(A_{\text{bin}}, B_{\text{bin}})$: **G**; at most $|v|+1$ runs, at most two annotated outputs, one of them the reference's; if M halts, τ applied at completing bits has zero lookahead and achieves* $\text{od} \le \lceil \bar{Z}/2 \rceil$*; and Q_d^∞ implies that M halts.*

Proof sketch. The runs of B_{bin} on v are, edge for edge, the runs of B_M on the event word of v, and the run of A_{bin} is that of A_M, births included. Internal bits act as `W`, which is legal and neutral in every state outside CAN, and the decoder state together with the completing bit determines the letter, so τ needs no lookahead. In Lemma 7.13 a choice block becomes the frame of `st`, then 0^d, then the frame of $\texttt{ec}_b$, and the probe becomes m frames of `prb`. Appendix D has the details. □

Proof of Theorem 7.1. Set $(A_*, B_M) := (A_{\text{bin}}, B_{\text{bin}})$. The map is computable: B_M has $O(|Q_M|)$ phases times 16 decoder states and 16 registers. Properties 1 to 3 are Lemmas 7.5, 7.6 and 7.16. For property 4, Corollary 7.11 with Lemma 7.16 gives $\text{GOOD}(0, 2L+1)$, and Corollary 7.15 with Lemma 7.16 gives the converse.

The reference is fixed. $U(x, 0)$ depends only on the row lists of Table 2, which are fixed for the 13 letters and never mention M. The decoder and the code are fixed once. So A_{bin} has 16 states, 8 registers and the final expression Φ, independently of M. Only the control of B_M depends on M. The same holds for every block p of Remark 7.3.

J is RE on **G**-instances by Theorem 4.4, and $\text{GOOD}(d, H)$ is decidable for given d, H, so Q_d is RE as well. Halting from zero counters many-one reduces to Q_d and to J through $M \mapsto (A_*, B_M)$, and that source problem is RE-hard by Lemma 7.2. □

7.9. **Slowing down to any linear rate.** B_M has at most $|v| + 1$ runs. Any slower linear rate is reached by a clock.

Proposition 7.17 (slow-down)**.** *For every $c \ge 1$ there is a computable map $M \mapsto (A_*^{(c)}, B_M^{(c)})$ such that:*

(1) $(A_*^{(c)}, B_M^{(c)})$ *satisfies* $\mathbf{G}$*, and* $A_*^{(c)}$ *depends on* c *only;*
(2) $B_M^{(c)}$ *has at most* $\lfloor |w|/c \rfloor + 1$ *runs on every input* w*, and they produce at most two annotated outputs, one of them that of* $A_*^{(c)}$*;*
(3) for every d*:* M *halts* $\iff Q_d \iff Q_d^\infty \iff J$*.*

Proof sketch. Give each machine $X \in \{A_*, B_M\}$ a clock modulo c. The states are $Q_X \times \{0, \dots, c-1\}$, the initial state is $(q_0, 0)$ and $\Phi(q, r) = \Phi_X(q)$. At residue $r < c-1$ every letter takes one identity edge to $(q, r+1)$ and writes nothing. At $r = c-1$ every edge e of X from q on a becomes an edge to $(\operatorname{dst} e, 0)$ with update $\operatorname{upd}(e)$. For $w = a_1 \cdots a_n$ put $w^{\downarrow} = a_c a_{2c} \cdots a_{c\lfloor n/c \rfloor}$. The runs of $X^{(c)}$ on w are in bijection with the runs of X on $w^{\downarrow}$, and every birth is multiplied by c. This holds on every input, well formed or not, so Theorem 7.1(2) and (3) give the run bound and the two annotated outputs. The fillers are identity steps in both machines, so od is multiplied by c as well.

If M halts, τ played at the positions cj gives $\mathrm{GOOD}(0, c(2L+1))$. Conversely, let a lookahead-d' controller keep $\operatorname{od} \le H'$ for $B_M^{(c)}$. A controller for B_M with lookahead $\lceil d'/c \rceil$ simulates it on the words $(0^{c-1}v_1) \cdots (0^{c-1}v_{|v|})$; its window determines the simulated window, end marker included. It keeps $\operatorname{od} \le H'/c$, and M halts by Theorem 7.1(4). Appendix D. □

Proposition 3.6 says that an unboundedly ambiguous automaton already pumps linearly many runs. With Proposition 7.17, no regime lies strictly between bounded and linear ambiguity, and Theorem 7.1 covers every unbounded one.

Proof of the lower bound in Proposition 3.6. Let $g \ge 1$ with $\liminf_{n\to\infty} g(n)/n > 0$. Choose a finite α with $0 < \alpha < \liminf g(n)/n$; the limit inferior may be infinite. Choose n_2 with $g(n) \ge \alpha n/2$ for all $n \ge n_2$, and an integer $c > \max(n_2, 4/\alpha)$. The instances of Proposition 7.17 have at most $\lfloor n/c \rfloor + 1$ runs on words of length at most n. For $n < c$ this is $1 \le g(n)$, which is where $g \ge 1$ is used. For $n \ge c$ it is at most $\alpha n/4 + 1 \le \alpha n/2 \le g(n)$, because $n \ge c > 4/\alpha$ and $n \ge n_2$. So these instances lie in the class bounded by g, and they are $\mathbf{G}^\flat$-instances as well (Remark 2.3). Proposition 7.17(3) gives RE-hardness for every fixed d, and Theorem 4.4 gives membership. □

7.10. **No computable memory or distance bound.** On the positive instances of Theorem 7.1 a controller exists. We show that no computable function of the instance size bounds its memory, or the distance it achieves. The tool is that a single fixed controller can be checked.

Lemma 7.18 (a fixed controller's distance is decidable)**.** *Given a* $\mathbf{G}$*-instance* (A, B)*, a lookahead* d *and a finite-memory controller* $\mathcal{C} = (M, m_0, \chi)$ *for* B *with lookahead* d*, it is decidable whether* $\sup_w \operatorname{od}(\rho_A(w), \rho_{\mathcal{C}}(w)) < \infty$*.*

Proof. Case $d = 0$*.* Let D have states $Q_B^{\mathrm{r}} \times M$, initial state (q_0^B, m_0) and final expressions $\Phi_D(q, m) = \Phi_B(q)$. For $\chi(m, q, a) = (e, m')$ it has the

single edge $(q, m) \xrightarrow{a} (\operatorname{dst} e, m')$ with update $\operatorname{upd}(e)$. D is deterministic and complete, since χ is total on $M \times Q_B^{\mathrm{r}} \times \mathrm{Win}_0$ and $\operatorname{dst} e$ is reachable. It is copyless, and its run on w applies exactly the updates of $\rho_{\mathcal{C}}(w)$, so the two annotated outputs agree and word agreement with A holds. Thus (A, D) is a $\mathbf{G}$-instance with one run per input, and $J(A, D)$ holds iff the controller's distance is bounded. Remark 3.2 decides J at $K = 1$, in PSPACE by Theorem 8.3.

Case $d \geq 1$. Here $\mathbf{G}^\flat$ is needed. Build B'' from B, M and a third component that guesses the current window $\omega \in \mathrm{Win}_d$, plays $\chi(m, q, \omega)$, and then verifies the guess letter by letter; the only states without edges are those reached after a window $a \dashv$, and only those and the initial state are final. Every input w has exactly one accepting run of B'', whose B-edges form $\rho_{\mathcal{C}}(w)$: the final state forces the last window, and a downward induction forces every earlier one. So (A, B'') is a $\mathbf{G}^\flat$-instance with one accepting run, and $J^\flat(A, B'')$ holds iff the controller's distance is bounded. A memoryless controller with lookahead $d + 1$ realises the accepting run on every input and plays a fixed legal edge elsewhere, which makes it total. Theorem 3.1 decides $J^\flat$. Appendix D has the construction. □

Write s_M for the size of (A_*, B_M).

Corollary 7.19 (no computable memory bound)**.** *Fix d. There is no computable $f : \mathbb{N} \to \mathbb{N}$ such that every halting M admits a finite-memory controller for (A_*, B_M) with lookahead d, memory of size at most $f(s_M)$, and bounded* od*. For every computable f, infinitely many halting M violate the bound. The same holds for a joint bound $f(s_M)$ on lookahead and memory.*

Proof. Suppose f exists. Given M, compute (A_*, B_M), s_M and $f(s_M)$. The controllers with lookahead d and memory $[m]$ for $m \leq f(s_M)$ form a finite computable set: the maps $[m] \times Q_B^{\mathrm{r}} \times \mathrm{Win}_d \to E_B \times [m]$ that choose legal edges. Decide each by Lemma 7.18. If one keeps od bounded, Q_d holds and M halts by Theorem 7.1(4). If none does, M does not halt, by the assumption on f. This decides halting. For the joint form, enumerate $d \leq f(s_M)$ as well. If only finitely many halting M violated f, raise f at those finitely many sizes to the least memories that work; the result is computable and nothing violates it. □

Corollary 7.20 (no computable distance bound)**.** *There is no computable g with* $\mathrm{GOOD}(g(s_M), g(s_M))$ *for every halting M. In particular, for each fixed d there is no computable g with* $\mathrm{GOOD}(d, g(s_M))$ *for every halting M, and every computable g is violated by infinitely many halting M.*

Proof. $\mathrm{GOOD}(d, H)$ is decidable for given d and H (Theorem 4.4). If g existed, decide $\mathrm{GOOD}(g(s_M), g(s_M))$. A yes gives J, so M halts. A no gives that M does not halt, by the assumption on g. The fixed-d form follows from Lemma 2.6, since $\mathrm{GOOD}(d, g)$ implies $\mathrm{GOOD}(\max(d, g), \max(d, g))$. Infinitely many violations: as in Corollary 7.19. □

Under finite ambiguity the lookahead threshold and the memory are computable from the instance (Section 8). On the family of Theorem 7.1 the least lookahead is 0 on every positive instance, so the incomputability sits entirely in the memory and in H. The controller τ matches this with memory $(L+1)(c^*+1)$ and $H = 2L+1$, where L is the halting time. By Proposition 7.17 both corollaries hold at every linear slope. The $d \geq 1$ case of Lemma 7.18 needs Theorem 3.1; at $d = 0$ the deterministic product and Remark 3.2 suffice.

8. Complexity

At fixed ambiguity, GOOD, Q_d and J are in PSPACE. Under **G**, with the register count fixed as well at two or more registers, all three are NL-complete. The threshold problem GOOD is PSPACE-complete at fixed K as soon as H is part of the input, and every problem is PSPACE-hard as soon as the ambiguity is. Table 1 collects the bounds. Appendix E has the proofs.

Throughout, s is the instance size, $k = k_A + k_B$ the register count, and $\mu = \max(\mu_A, \mu_B)$ the largest number of output letters in one update. A pair (q, a) with $|\Delta_B(q, a)| \geq 2$ is a *branching pair*. The *fork budget* is

$$\kappa_B = \sum_{(q,a)\ \text{reachable, branching}} \big(|\Delta_B(q,a)| - 1\big) \leq s,$$
$$\kappa = \min(K-1, \kappa_B), \qquad N = 2^\kappa.$$

"K fixed" means the K-promise for a constant K that the algorithm knows. Otherwise the algorithm uses κ_B, which is computable in polynomial time, and needs only the promise of finite ambiguity. Instances satisfy **G** unless a $\flat$ is written. The parameters d and H are in binary unless stated. All problems are the od versions.

8.1. The rank counters as a graph.

Under **G** the fork budget bounds every single run.

Lemma 8.1 (fork pairs)**.** *Assume* **G** *and finite ambiguity. A run of B passes each branching pair (q, a) at most once. So the sum of* $(\deg - 1)$ *over its fork positions is at most κ_B, and at most $K-1$ under the K-promise.*

Proof sketch. Two visits to (q, a) enclose a loop at q that starts with an a-edge. Pumping that loop r times still gives a run, because every state is final, and the count behind Lemma A.4 yields at least $1 + r$ runs on the pumped word. That contradicts finite ambiguity. See Appendix E. □

So the shapes of Section 6.4 lie in the finite class $\mathfrak{S}'_\kappa$ of trees whose root-to-leaf paths have fork sum at most κ. Such a tree has at most N leaves. The cutoff of Section 6.4 uses only the finiteness of the shape class, so nothing below needs the value of K.

Proposition 8.2 (sizes and cutoff)**.** *Assume* **G** *and finite ambiguity. Every witness automaton* $\mathcal{W}^{\mathrm{od}}_{\mathcal{T},\varphi}$ *with* $\mathcal{T} \in \mathfrak{S}'_\kappa$ *has at most* Z *states, edge weights of norm at most* W*, and at most* m *coordinates, where*

$$\log_2 Z \le 6Ns + 2N^2, \qquad W \le \max(\mu, 1) \le s, \qquad m \le 2N + \tbinom{N}{2}.$$

Hence $J \iff \mathrm{GOOD}(T_*, T_*)$ *with* $\log_2 T_* = O(N^3 s + N^4)$*, and* $Q_d \iff \mathrm{GOOD}(d, H_*(d))$ *with* $\log_2 H_*(d) = O(N^3 s + N^4 + N^2 \log(d+1))$*. For fixed* K*,* $T_* = 2^{O(s)}$*. In general* $T_* \le 2^{2^{O(s)}}$*.*

Proof sketch. Per leaf, a state holds a status, one state of each machine, a use bit and an endpoint map in $\{\mathrm{pre}, \mathrm{post}, z\}^2$ per register, and two flags. That is at most $6s$ bits for $s \ge 2$. Node and pair phases add $N + \binom{N}{2}\log_2 3$ bits. In one row R moves by at most μ, and G, E by at most 1. Proposition 6.7 turns Z, W, m into T_* and $H_*(d)$. □

Fix (d, H). The *bounded witness graph* $\mathcal{G}_{d,H}$ has as vertices the tuples $(\mathcal{T}, \varphi, \mathfrak{s}, (R_\lambda), (G_\lambda), (E_\pi))$ with $\mathcal{T} \in \mathfrak{S}'_\kappa$, $\mathfrak{s}$ a state of $\mathcal{W}^{\mathrm{od}}_{\mathcal{T},\varphi}$, $R_\lambda \in [-\mu_B H, \mu_A H]$, $G_\lambda \in [0, H+1]$ and $E_\pi \in [0, d]$. Its edges are the rows of $\mathcal{W}^{\mathrm{od}}$, with G and E saturating at $H+1$ and d, and with no edge when R leaves its range. Sources are initial states with zero counters. Targets are accepting states with every $R_\lambda = 0$, every $G_\lambda = H+1$ and every $E_\pi = d$.

Theorem 8.3 (space bounds)**.** *Assume* **G** *and finite ambiguity.*

- *(a)* $\mathrm{BAD}(d, H)$ *holds iff some target of* $\mathcal{G}_{d,H}$ *is reachable from a source. Hence* $\mathrm{GOOD}(d, H)$ *is in* $\mathrm{NSPACE}(S_{d,H})$ *and in* $\mathrm{DSPACE}(S^2_{d,H})$*, where* $S_{d,H} = \mathrm{poly}(N, s) + O(N \log(\mu H + 2) + N^2 \log(d+2))$*.*
- *(b)* Q_d *and* J *are in* $\mathrm{NSPACE}(2^{O(\kappa)}\mathrm{poly}(s, \log(d+2)))$*.*
- *(c) For fixed* K*, all three problems are in PSPACE, and* $J \in \mathrm{NSPACE}(O(s)) \subseteq \mathrm{DTIME}(2^{O(s)})$*. For* $K = O(\log s)$ *they are in PSPACE. For finitely ambiguous* B *with no bound on* K *they are in EXPSPACE.*

Proof sketch. A source-to-target path projects to an accepting walk whose exact weight meets the guards, since saturation only underestimates G and E and R is exact. Conversely, a bad witness gives at each leaf a least bad rank c_λ; the words agree by (R4), so the rank is the same in both machines. The walk of the od gadget with P-occurrences exactly the ranks below c_λ keeps R_λ within range by Lemma 4.1 and ends at 0. A vertex takes $\log_2 Z$ bits plus the counters, and an edge is checked in polynomial time from the explicit A and B. Savitch's theorem bounds reachability; the Immerman–Szelepcsényi theorem gives the nondeterministic form of the complement, and deterministic classes are closed under complement anyway. For (b), substitute T_* and $H_*(d)$, which have $O(N^3 s + N^4 + N^2 \log(d+1))$ bits. For (c), every quantity is polynomial in s once N is, and $\kappa \le s$ in general. □

Corollary 8.4 (elementary controllers)**.** *On a positive instance of J there is a controller with lookahead T_* that keeps* od $\le T_*$ *with at most*

$$3|Q_A| \cdot 2^{|\mathcal{N}_{T_*}|} \;\le\; 3|Q_A| \cdot 2^{2^{O(k)}(\mu T_*+1)(T_*+2)} \;=\; 2^{2^{O(N^3 s+N^4)}}$$

memory states, where $\mathcal{N}_H$ is the GAP*-only checker of Theorem 4.2. It is computable in time polynomial in $|Q_A||Q_B|\,2^{|\mathcal{N}_{T_*}|}\,|\Sigma|^{T_*+1}$. The memory is doubly exponential in s for fixed K and triply exponential in general. The same holds for Q_d with $(d, H_*(d))$.*

Proof sketch. Use the complement of the subset construction of $\mathcal{N}_{T_*}$ in the extraction of Proposition 4.5. That is sound because the extraction evaluates its DFA only on joint words of runs over the same input, whose words agree by (R4), and there the GAP-only checker is exact. □

8.2. **Fixed register count.** Let $\mathcal{I}_{K,k}$ be the **G**-instances with the K-promise and $k_A + k_B \le k$.

Theorem 8.5 (fixed ambiguity and fixed register count)**.** *Fix $K \ge 1$ and $k \ge 0$. On $\mathcal{I}_{K,k}$:*

- *(a) J is in NL.*
- *(b) Q_d is in NL, with d in binary, in unary, or fixed.*
- *(c)* GOOD(d, H) *is in NL, with d and H in binary.*
- *(d) There is a polynomial $p_{K,k}$ with $T_* \le p_{K,k}(s)$. On a positive instance of J a controller with lookahead T_* keeps* od $\le T_*$ *with memory at most $3|Q_A| \cdot 2^{q_{K,k}(s)}$, for a polynomial $q_{K,k}$.*
- *(e) For $k \ge 2$, each of J, Q_d (every fixed d) and* GOOD(d, H) *(every fixed d, H) is NL-hard already on $\mathcal{I}_{1,2}$, with $k_A = k_B = 1$, $|Q_A| = 1$ and deterministic B. So all three are NL-complete on $\mathcal{I}_{K,k}$ for every $K \ge 1$ and $k \ge 2$.*
- *(f) For $k \le 1$ every instance of $\mathcal{I}_{K,k}$ is positive for all three problems and all d, H.*

The cutoff $H_*(d)$ bounds H at fixed d. A dual cutoff bounds d at fixed H.

Lemma 8.6 (dual cutoff)**.** *Assume* **G** *and the K-promise. Let Z_0 bound the number of states of every witness automaton $\mathcal{W}^{\mathrm{od}}_{\mathcal{T},\varphi}$ with $\mathcal{T} \in \mathfrak{S}'_\kappa$, let W and m be as in Proposition 8.2, and put $\beta(x) = Z_0^2 W + x + 1$ and $D_*(H) = m!\,\beta(H)^m + 1$. For every H and every $d \ge D_*(H)$,* GOOD$(d, H) \iff$ GOOD$(D_*(H), H)$.

Proof. Appendix E. □

With K and k both constant, a leaf state has $8 \cdot 18^k |Q_A||Q_B|$ possibilities, so a witness automaton has $Z_0 \le c_{K,k} s^{2N}$ states. Then T_* is polynomial in s, and so are $H_*(x)$ and $D_*(x)$ for $x \le T_*$. These numbers have $O(\log s)$ bits and are computable in logspace. This is where the quantifiers over d and H disappear. J becomes GOOD(T_*, T_*), and Q_d becomes GOOD$(d', H_*(d'))$

with $d' = \min(d, T_*)$. A binary pair (d, H) becomes $(d, \min(H, H_*(d)))$ if $d < T_*$, $(\min(d, D_*(H)), H)$ if $d \geq T_* > H$, and (T_*, T_*) if $d, H \geq T_*$. The first case is Proposition 8.2 with Lemma 2.6, the second is Lemma 8.6, and the third is $J \iff \mathrm{GOOD}(T_*, T_*)$ with Lemma 2.6. In every case both arguments are at most T_*, $H_*(x)$ or $D_*(x)$ with $x \leq T_*$. The vertices of $\mathcal{G}_{d',H'}$ then have $O(\log s)$ bits.

Proof sketch. The obstacle is the edge test. A row colours every letter occurrence of the updates at every leaf, which is $\Theta(N\mu)$ symbols, and μ can be $\Theta(s)$. We scan instead. For each register expression, the colour conditions of the gadget are a left-to-right run of the three-state marking DFA, a one-bit count of M-occurrences, and a running difference of P-occurrences in $[-\mu, \mu]$. So we guess one colour per occurrence and keep an $O(\log s)$-bit pointer. The refined graph has logspace-decidable adjacency, and the three problems reduce to non-reachability in it, which is in NL by the Immerman–Szelepcsényi theorem. For (d), apply Corollary 8.4 with T_* polynomial; $|\mathcal{N}_{T_*}|$ is then polynomial because k is fixed. For (f), $k \leq 1$ forces $k_A = 0$ or $k_B = 0$; the output of that machine is empty, so by (R4) every output is empty and $\mathrm{od} = 0$. Part (e) is Proposition 8.7. Appendix E. □

Proposition 8.7 (one register each)**.** *From a directed graph (V, E) and vertices $s \neq t$ one computes in logspace a* $\mathbf{G}$*-instance with $k_A = k_B = 1$, $|Q_A| = 1$ and B deterministic and complete, such that* $\mathrm{od} = 0$ *on every word if t is unreachable from s, and* $\sup_w \mathrm{od} = \infty$ *otherwise. Hence J, every Q_d and every* $\mathrm{GOOD}(d, H)$ *are NL-hard on $\mathcal{I}_{1,2}$.*

Proof sketch. A appends one letter per step. B walks the graph on the input letters and appends as well, until the walk from s first reaches t; from then on it prepends. Prepending reverses the stored births, so the first rank has A-birth 1 and B-birth $|w|$. The reversed stretch can be padded at will. Non-reachability is NL-complete. □

Corollary 8.8 (register count is the resource)**.** *Fix $K \geq 1$. Let $f \geq \log$ be space-constructible and nondecreasing, and let $\mathcal{I}_{K,\leq f}$ be the* $\mathbf{G}$*-instances with the K-promise and $k \leq f(s)$.*

- *(a) On $\mathcal{I}_{K,\leq f}$, J, Q_d and* $\mathrm{GOOD}(d, H)$ *are in* $\mathrm{NSPACE}(O_K(f(n)))$*, n the input length.*
- *(b) For every $c \geq 1$, none of the three problems on $\mathcal{I}_{K,\leq \log^c s}$ is PSPACE-hard under logspace reductions.*
- *(c) So at fixed K the question NL versus PSPACE for J and Q_d concerns only families whose register count is not $O(\log^c s)$ for any c. Hardness above NL at bounded k would collapse that class into NL.*

Proof sketch. The same scan with $k \leq f(s)$ gives vertices of $O_K(f(s) + \log s)$ bits. A PSPACE-hardness proof on $\mathcal{I}_{K,\leq \log^c s}$ would put $\mathrm{NSPACE}(\log^{c+1} n)$ into $\mathrm{NSPACE}(\log^c n)$, against the space hierarchy theorem. □

Remark 8.9 ($\mathbf{G}^\flat$ at fixed k). The upper bounds of Theorem 8.5 are proved under $\mathbf{G}$ only. Under $\mathbf{G}^\flat$ the witness automata of Section 6.4 store at every leaf a co-accessibility set $C_\lambda \subseteq Q_B$, that is $|Q_B|$ bits per leaf even at $k = 0$, and we know no replacement with $O(\log s)$ bits. The obstacle is present without registers: at $k = 0$ with empty outputs, $J^\flat$ is the bounded-delegator problem for an NFA with at most K accepting runs per word, which Corollary 3.9 places in PSPACE. So under $\mathbf{G}^\flat$ at fixed K and fixed $k \geq 2$ the three problems are NL-hard (Proposition 8.17) and in PSPACE (Proposition 8.18), with nothing known in between. For $k \leq 1$ acceptance is still an obligation, so (f) does not apply either.

8.3. **PSPACE lower bounds.**

Theorem 8.10 (the threshold problem at $K = 1$)**.** *Let $c \geq 2$ be fixed. From complete DFAs $D_1, \ldots, D_m$ over Λ one computes in logspace an instance (A, B) such that*

(1) $\mathbf{G}$ holds, B is deterministic and complete, so the K-promise holds for every $K \geq 1$, and the output alphabet is unary;
(2) for $|w| \geq P := cm + 1$, written $w = w'u$ with $|w'| = P$: $\mathrm{od}(\rho_A(w), \rho_B(w)) = cm$ if $u \in \bigcap_j L(D_j)$ and $\leq c(m-1)+1$ otherwise; for $|w| < P$, $\mathrm{od} \leq c(m-1)$;
(3) for every d and $H := c(m-1)+1$: $\mathrm{GOOD}(d, H)$ holds iff $\bigcap_j L(D_j) = \emptyset$, for finite and for arbitrary memory.

With $c = 2$ the threshold is $H = 2m - 1$.

The preamble of P steps makes A store m blocks of c letters, born in order, then one letter Z. B stores block j in the register of D_j's initial state and then lets the input drive each DFA by moving the block between registers. Its final expression lists the rejecting blocks, then Z', then the accepting blocks. So Z' is displaced by c times the number of accepting blocks, while every other displacement stays at most $c(m-1)+1$. The threshold turns a count into a conjunction. Kozen proved DFA intersection PSPACE-complete, and PSPACE is closed under complement.

Corollary 8.11. *(a) For every fixed $K \geq 1$ and fixed d, $\mathrm{GOOD}(d, H)$ on $\mathbf{G}$-instances with the K-promise is PSPACE-complete, with H in unary or in binary.*
(b) Computing $\sup_w \mathrm{od}(\rho_A(w), \rho_B(w))$ for a $\mathbf{G}$-instance with deterministic B is PSPACE-hard, as is deciding whether it is at least $2m$.
(c) Q_d and J hold on every instance of Theorem 8.10, since $\mathrm{od} \leq cm$.

So Theorem 8.10 says nothing about Q_d or J. At fixed K and unbounded register count these lie between NL and PSPACE. The hardness needs H to grow with m. Whether $\mathrm{GOOD}(d, H)$ is PSPACE-hard at fixed K and fixed H, for instance $K = 1$ and $H = 0$, which is origin equivalence of two deterministic copyless SSTs restricted to $\mathbf{G}$-pairs, is open as far as we know; Theorem 8.5 settles it only for fixed k.

Theorem 8.12 (ambiguity in the input)**.** *From DFAs $D_1, \dots, D_m$ over Λ one computes in logspace an instance (A, B) such that*

(1) **G** *holds, the output alphabet is unary, B has at most m runs on every word, with exactly one fork, of degree m, and $\kappa_B = m - 1$;*
(2) if $\bigcap_j L(D_j) = \emptyset$, then $\mathrm{GOOD}(0, 0)$ *holds;*
(3) if $\bigcap_j L(D_j) \neq \emptyset$, then $\mathrm{GOOD}(d, H)$ *fails for every d and H, even with arbitrary memory.*

Hence J, Q_d for every fixed d, and $\mathrm{GOOD}(d, H)$ *for every fixed d, H are PSPACE-hard on* **G***-instances with ambiguity at most m, in particular with K given in unary.*

Here B carries one token per DFA and moves it with the DFA's state. The single fork, on a letter `f`, chooses which DFA is tested at the end: in branch j the accepting registers of D_j are listed after the appended suffix. A controller that tracks all DFAs picks a rejecting j and keeps $\mathrm{od} = 0$. If every DFA accepts, every choice loses to the padding $\mathtt{W}^{d+H+1}$. For fixed m, DFA intersection is in NL, so this reduction gives nothing at fixed K.

8.4. **Lookahead.**

Theorem 8.13 (exponential lookahead at ambiguity $n+1$)**.** *For every $n > 1$ there is a* **G***-instance (A_n, B_n) such that B_n has at most $n+1$ runs per word, a single fork position of degree $n+1$ at the first letter, and $O(n)$ states; the $(n+1)$-promise is given; $|\Sigma| = n + 2$, $\Gamma = \{x\}$, four registers in total, and $s = O(n^2)$; J and* $\mathrm{GOOD}(2^n, 2^n)$ *hold; and Q_d fails for every $d < 2^n$. The same holds in the one-way model of Section 9. So the needed lookahead is $2^{\Omega(\sqrt{s})}$, even under finite ambiguity.*

Proof sketch. Klein and Zimmermann [KZ16] build safety automata $\mathcal{A}'_n$ of size $O(n)$ whose delay game Player O wins with constant delay $2^n + 1$ and loses with every delay at most 2^n. O's only relevant move is the first, which selects one of $n + 1$ copies. We encode the game as in Section 9: B forks once, a rejecting step moves a token into a register that is emitted last, and a padding letter lets the environment stretch the distance. $\mathrm{GOOD}(d, H)$ holds for some H iff O wins with delay $d + 1$. Appendix E. □

With $K = n+1$ given, Proposition 8.2 gives $\log T_* = 2^{O(n)}\mathrm{poly}(n)$ against the lower bound $\log d \geq n$.

Proposition 8.14 (linear lookahead at $K = 2$)**.** *For every $n \geq 1$ there is a* **G***-instance (A_n, B_n) over $\Sigma = \{a, b\}$, $\Gamma = \{x\}$, with $n+3$ and $2n+5$ states, five registers in total and size $O(n)$, such that B_n has at most two runs per word with a single fork position, the first letter, of degree 2;* $\mathrm{GOOD}(n, 0)$ *holds; and Q_d fails for every $d < n$, even with arbitrary memory. So the least lookahead for Q_d is exactly $n = \Omega(s)$.*

The fork at position 1 puts a token into P_1 or P_2, and the letter at position $n + 1$ decides which placement is emitted first. Two words that agree on

their first n letters force the same choice at lookahead $d < n$, and one of them is wrong by $|w| - 1$. The successful controller is memoryless. At fixed K the least lookahead therefore lies between $\Omega(s)$ and $2^{O(s)}$, and between $\Omega(s)$ and $s^{O(1)}$ when k is also fixed.

8.5. **Memory.**

Theorem 8.15 (exponential memory at $K = 2$)**.** *For every $n \geq 1$ there is a $\mathbf{G}$-instance (A_n, B_n) with $\Sigma = \{0, 1, \mathtt{t}, \mathtt{f}\}$, $\Gamma = \{x\}$, two states and $n + 1$ registers in A_n, three states and $2n + 1$ registers in B_n, and $s = 45n + 60$, such that*

(1) B_n has at most two runs on every word, with exactly one branching pair, of degree 2, so $\kappa_B = 1$;
(2) a controller with lookahead 0 and memory $\{0, 1\}^n$ keeps $\mathrm{od} = 0$ on every word, so $\mathrm{GOOD}(d, H)$ holds for all d, H;
(3) for every d and every controller with lookahead d and fewer than 2^n reachable memory values, and for every H, some word has $\mathrm{od} > H$.

Hence for every (d, H) the least memory of a controller witnessing $\mathrm{GOOD}(d, H)$ is exactly 2^n, which is $2^{\Omega(s)}$.

Both machines keep the n most recent bits in a shift register of slots. A stores the bit letters by position; B stores each in one of two registers per slot, by value. The fork at $\mathtt{f}$ tests the oldest slot: the wrong branch emits that slot after the suffix. The environment pads with $\mathtt{t}^d$ before it reveals, by feeding further bits, which old bit reaches slot n, so no lookahead helps. A controller with fewer than 2^n memory values confuses two histories, and one of them defeats its choice. The construction is in Appendix E.

Corollary 8.16. *The same holds for $\mathrm{GOOD}^\flat$, $Q_d^\flat$ and $J^\flat$ with the $2^\flat$-promise.*

The gap between $2^{\Omega(s)}$ and the doubly exponential bound of Corollary 8.4 at fixed K is open. The family has $3n + 2$ registers, so Theorem 8.5(d) does not apply to it. At linear ambiguity no computable memory bound exists at all (Corollary 3.5).

8.6. **Under $\mathbf{G}^\flat$, and without a supplied K.**

Proposition 8.17 (lower bounds transfer)**.** *On a $\mathbf{G}$-instance, $\mathrm{GOOD}^\flat(d, H) \iff \mathrm{GOOD}(d, H)$ for all (d, H), with the same controllers, and the $K^\flat$-promise is the K-promise. So every lower bound of this section holds for the corresponding $\mathbf{G}^\flat$ problem with the same K.*

Proof sketch. All states are final and B is input-enabled, so the sink of B_z is unreachable and the controllers of the two models coincide, with the same runs. □

No upper bound transfers. The proofs above use Lemma 8.1 and the automata $\mathcal{W}^{\mathrm{od}}$, and under $\mathbf{G}^\flat$ a run may pass a branching pair arbitrarily

often (Example 5.1). The automata $\mathcal{W}^\flat$ of Section 6.4 give the same bounds with $N = 2^K$.

Proposition 8.18 (fixed-K upper bounds under $\mathbf{G}^\flat$)**.** *Assume the $K^\flat$-promise and put $N = 2^K$.*

1. *Every $\mathcal{W}^\flat_{\mathcal{T},\varphi,\tau}$ obeys the bounds of Proposition 8.2, so $J^\flat \iff \mathrm{GOOD}^\flat(T^\flat_*, T^\flat_*)$ with $\log_2 T^\flat_* = O(N^3 s + N^4)$, and $Q^\flat_d \iff \mathrm{GOOD}^\flat(d, H^\flat_*(d))$ with $\log_2 H^\flat_*(d) = O(N^3 s + N^4 + N^2 \log(d+1))$.*
2. *(a) $\mathrm{GOOD}^\flat(d, H)$ is in $\mathrm{NSPACE}(O(Ns + N^2 + N\log(H+2) + N^2\log(d+2)))$. (b) $Q^\flat_d$ is in $\mathrm{NSPACE}(O(N^4 s + N^5 + N^3 \log(d+2)))$ and $J^\flat$ in $\mathrm{NSPACE}(O(N^5 s + N^6))$. For fixed K, for $K \le c \log_2 s$ with c known, and for K in unary with $K = O(\log s)$, all three are in PSPACE, and for fixed K, $J^\flat \in \mathrm{NSPACE}(O(s)) \subseteq \mathrm{DTIME}(2^{O(s)})$. With K in unary and no further bound, all three are in EXPSPACE.*
3. *For every fixed $K \ge 1$, $\mathrm{GOOD}^\flat(d, H)$ is PSPACE-complete, for fixed d or d in the input, H in unary or in binary, already with deterministic B.*
4. *On a positive instance of $J^\flat$ a controller with lookahead $T^\flat_*$ meets the objective of $\mathrm{GOOD}^\flat(T^\flat_*, T^\flat_*)$ with memory at most $(|Q_A|+1)\cdot 2^{|\mathcal{N}_{T^\flat_*}|} = 2^{2^{O(N^3 s + N^4)}}$. The same holds for $Q^\flat_d$ with $(d, H^\flat_*(d))$.*

Proof sketch. A leaf of $\mathcal{W}^\flat$ additionally stores $C_\lambda \subseteq Q_B$, which fits into the $6s$ bits per leaf. The graph $\mathcal{G}^\flat_{d,H}$ is built as in Section 8.1; a row test now also checks the co-accessibility update, the final-state conditions and the kill condition, each with $O(\log s)$-bit pointers. Theorem 5.4 identifies $\mathrm{GOOD}^\flat$ with the absence of a reachable target. Hardness in (3) is Theorem 8.10 transferred. Appendix E. □

Without a supplied K the budget under $\mathbf{G}^\flat$ is K itself, and we need a bound on it. Weber and Seidl [WS91] bounded the degree of ambiguity of an n-state NFA by $2^{O(n \log n)}$, and Kiefer, Mazowiecki, Ryzhikov and Wiśniewski [KMRW26] improved this to $2^{O(n)}$ in a preprint. Runs of B are edge sequences and parallel edges are distinct, so we pass through the edges.

Lemma 8.19 (edge-state NFA)**.** *Let $\mathcal{E}_B$ have states $E_B \cup \{\iota\}$, a transition $\iota \xrightarrow{a} e$ iff $\mathrm{src}(e) = q_0^B$ and $\mathrm{lab}(e) = a$, a transition $e \xrightarrow{a} e'$ iff $\mathrm{dst}(e) = \mathrm{src}(e')$ and $\mathrm{lab}(e') = a$, and final states the edges into F_B, plus ι iff $q_0^B \in F_B$. For every w the accepting runs of B on w are in bijection with those of $\mathcal{E}_B$. It has $|E_B| + 1 \le s + 1$ states.*

Proposition 8.20 (size-only ambiguity bound)**.** *If (A, B) satisfies $\mathbf{G}^\flat$ and B is finitely ambiguous, then $K \le 2^{c(|E_B|+1)} \le 2^{O(s)}$ for an absolute constant c [KMRW26], and $K \le 2^{O(s \log s)}$ by [WS91]. The same holds under $\mathbf{G}$ for the K-promise.*

The class results below need only some bound $2^{\mathrm{poly}(s)}$, which the older bound also gives. By the lower-bound family of [KMRW26], K can be $2^{\Omega(\sqrt{s})}$

on $\mathbf{G}^\flat$-instances, so any analysis with 2^K leaves is at least doubly exponential in $\sqrt{s}$.

Lemma 8.21. *"B has more than K' accepting runs on some word" is in* $\mathrm{NSPACE}(O((K'+1)\log s + (K'+1)^2))$*, with K' in binary.*

Proof sketch. Guess the word letter by letter, follow $K'+1$ runs as state pointers, and keep a bit matrix recording which pairs have already diverged. Accept when all runs are in final states and all pairs have diverged. □

Iterating $K' = 1, 2, \ldots$ with Savitch's theorem computes K, or an upper bound on it, in space $2^{O(s)}$; an upper bound suffices because the $K'^\flat$-promise stays true for $K' \geq K$.

Proposition 8.22 (without a supplied K)**.** *On $\mathbf{G}^\flat$-instances with B finitely ambiguous and K not given, with d and H in binary:*

(1) $\mathrm{GOOD}^\flat(d, H)$*, $Q_d^\flat$ and $J^\flat$ are in* $\mathrm{NSPACE}(2^{2^{O(s)}}\mathrm{poly}(\log(d+2), \log(H+2)))$*, hence in 2-EXPSPACE; with the bound of* [WS91] *the exponent becomes $2^{O(s\log s)}$, and the class is the same.*

(2) On a positive instance of $J^\flat$ the cutoff satisfies $\log_2 T_*^\flat = 2^{2^{O(s)}}$, so $T_*^\flat$ is triply exponential, and the memory of Proposition 8.18(4) is at most quadruply exponential in s.

(3) The problems are PSPACE-hard (Theorem 8.12 with Proposition 8.17).

Proof sketch. Compute K, set $N = 2^K \leq 2^{2^{O(s)}}$, and substitute into Proposition 8.18(1)–(2), whose proofs never assume K fixed. Savitch's theorem squares the space, which keeps the form. □

Under $\mathbf{G}$ the same question is easier. There the algorithm uses $\kappa_B \leq s$ and never needs K, so Theorem 8.3(c) gives EXPSPACE. Above PSPACE nothing is known in either case.

9. The one-way contrast

For one-way transducers, the controller problem remains decidable with arbitrarily many runs. Let A be deterministic and real-time, and let B be nondeterministic and real-time. Both machines are total, all states accept, and each state has a final output word. We assume that every run of B on every input u produces $A(u)$. Transition outputs have the origin of the input letter being read. Final-output letters have origin $|u|+1$. Controllers use the strict $\dashv$ convention.

Proposition 9.1 (One-way controllers)**.** *For these machines, with no ambiguity bound, J is EXPTIME-complete. The problem Q_d is also EXPTIME-complete when d is supplied in unary. Put $N = |Q_A||Q_B|$ and*

$$M = \max_{p \in Q_A,\, q \in Q_B} \big|\, |f_B(q)| - |f_A(p)| \,\big|.$$

Fix d. If a lookahead-d controller achieves a uniform finite origin bound, then one with memory $Q_A \times Q_B$ witnesses

$$\mathrm{GOOD}(d, H_d), \qquad H_d = MN|\Sigma|^{d+1} + d.$$

This implication also holds when the initial controller has arbitrary memory. Moreover,

$$J \iff Q_{d_0}, \qquad d_0 = 2^{2(2N+1)^2+1} - 1.$$

The decidability assertion may be folklore for this precise one-way model.

Proof idea. Word agreement bounds the lag between the two outputs by M, so at most M ranks are pending at any time. For fixed d this is a Büchi game, and over all d a delay game in the sense of Klein and Zimmermann [KZ16]. Appendix F gives the proof and the hardness reduction. □

Bounding the number of runs changes the picture.

Proposition 9.2 (One-way controllers at fixed ambiguity)**.** *Fix $K \geq 1$, and let B have at most K runs on every input word. Then J, Q_d and $\mathrm{GOOD}(d, H)$ are NL-complete for the machines of this section, with d and H supplied in binary. Hardness holds already at $K = 1$, for every fixed d and H, with one output letter, a deterministic B and a one-state A.*

Proof sketch. A pair of this section is a pair of one-register SSTs with updates $X := X\alpha$ and final expressions $X \cdot f(q)$, with the same origins, so it satisfies **G** with general final expressions. Remark 2.9 turns it in logarithmic space into a **G**-instance with two registers and the same answers, and Theorem 8.5 gives the upper bound. For hardness, B reads a word of vertices as a walk from s and writes its letter on the step that first reaches t, or at the end if no such step occurs, while A always writes it at the end. The origin distance is then zero on every word if t is unreachable and unbounded otherwise, which reduces the complement of reachability to all three problems; Appendix F gives the full proof. □

Registers can retain letters and rearrange their positions in the final output. The counter-machine construction uses this freedom to expose stored arithmetic through later origin comparisons. Appendix F also treats $\mathcal{F}_1$ under **G**: each machine has one register, updates $X := Xu$, and final expressions in $\{X, \varepsilon\}$. There, J and Q_d with unary d remain EXPTIME-complete without an ambiguity bound. Allowing both append and prepend in one register remains open.

10. Related work and open problems

Related work. Here we record whose tools we used. Exibard, Filiot and Jecker [EFJ18] prune the runs of a union of sequential transducers inside a game and bound the delay with a tree of counters. Our kill-witness trees in Section 5 descend from that construction. Klein and Zimmermann [KZ16] supply the delay games behind the one-way proposition, and

Holtmann, Kaiser and Thomas [HKT10] already cover its decidability part. We found no statement of the proposition itself. It may be folklore. Ravikumar and Santean [RS07] introduced lookahead delegators, and Löding and Repke [LR13] studied them for NFAs. Löding's erratum [Lö23] reports that Lemma 3 of that paper is false and that its PSPACE upper bounds and its lookahead bound rest on it, while the polynomial bound at fixed lookahead is re-proved there and the hardness parts are reported intact. Section 5 replaces that lemma by a subset argument, and no result here uses either paper. Bose, Muscholl, Penelle and Puppis [BMPP18] compare two transducers up to a given bounded resynchroniser. Bose, Krishna, Muscholl, Penelle and Puppis [BKM$^+$19] and Kuperberg and Martens [KM20] ask whether some resynchroniser exists. That is decidable for functional one-way transducers and undecidable once the target is relational and the class contains bounded shifts. Filiot, Jecker, Löding and Winter [FJLW23] compare two SSTs offline through delays, and with Muscholl and Puppis [FJL$^+$25] they split a k-valued SST into k deterministic ones. The split keeps the relation and forgets the edges, so it picks no run of B. Winter [Win21] uses our pointwise origin metric for tree transducers at a fixed bound. Bojańczyk [Boj14] gives the origin semantics, and Alur and Deshmukh [AD11] the functionality test behind the promise check. Weber and Seidl [WS91] and Kiefer, Mazowiecki, Ryzhikov and Wiśniewski [KMRW26] bound the ambiguity of a finitely ambiguous automaton by an exponential in its size, which Section 8 uses when K is not supplied; the second is a preprint.

Fixed K, unbounded registers. This is the question we would most like answered. Under $\mathbf{G}$ with K fixed and the register count fixed at $k \geq 2$, J and Q_d are NL-complete (Theorem 8.5). With K fixed and the register count free, we know PSPACE from above (Theorem 3.7) and NL-hardness from below, already with one register in each machine ($k = 2$). Branching is controlled in both cases. So the gap is the cost of register interaction alone. A resolution should say whether the extra space carries information the controller needs, or only an expensive encoding of information it already has. Two refinements are open as well. The version with H fixed too, down to $H = 0$, is open as far as we know. The pumping certificates of Theorem C.6 have exponential size, and polynomial ones at fixed K would put J under $\mathbf{G}^\flat$ in coNP.

K in the input. With K supplied, the problems are PSPACE-hard and in EXPSPACE, the upper bound under $\mathbf{G}^\flat$ taking K in unary. Which end is right? The hardness proof lets the register count grow with the instance. At fixed register count with K supplied, PSPACE-hardness is open.

The promise $\mathbf{G}^\flat$. Three gaps remain. At fixed K and fixed register count, only PSPACE is proved under $\mathbf{G}^\flat$. The log-space row scan behind Theorem 8.5 relies on every branch surviving; under $\mathbf{G}^\flat$ the controller must also know which branches can still accept, and that information may cost more. When K is not supplied, the bound is 2-EXPSPACE, against EXPSPACE under $\mathbf{G}$. With no registers at all the problem is bounded delegation for

NFAs. With at most K accepting runs per word it is in PSPACE, and hardness is open. For all NFAs it is decidable in doubly exponential time through delay games, and the exact complexity is open now that the PSPACE upper bound of Löding and Repke rests on a false lemma.
One register with prepend. The fragment with one append-only register and final output X or ε is decidable under $\mathbf{G}$ at every ambiguity and EXPTIME-complete (Section 9 and Appendix F). Let the same register take prepends as well and we know nothing. Two append-only registers are open too, and so are resets. Theorem C uses sixteen registers in B and eight in the reference, with both update directions. Our gadget needs both directions to expose the stored imbalance. Whether undecidability needs them is open, and so is the least register budget at which it sets in.
Memory and lookahead. At $K = 2$ a controller needs $2^{\Omega(s)}$ states (Theorem 8.15), and Theorem $\mathrm{B}^\flat$ provides doubly exponentially many. The same gap holds under $\mathbf{G}^\flat$. Lookahead at fixed K sits between a linear lower bound and $2^{O(s)}$. The least lookahead, the least distance and the frontier between them are computable (Proposition C.12). At fixed K the frontier's points can be listed with polynomial working space, but there may be exponentially many, and its shape is open.
Beyond $\mathbf{G}^\flat$. Appendix H drops word agreement. Drop the origin bound as well and ask only for a correct output: whether finite memory suffices is open.
Other references. The reference here is a deterministic copyless SST. Bojańczyk [Boj14] states that deterministic two-way transducers and MSO transductions translate into SSTs with the same origins, without a written proof. If it holds, such a transducer can serve as the reference under $\mathbf{G}^\flat$, and under $\mathbf{G}$ once it is completed. The translation changes the instance size, so our complexity bounds do not carry over as stated.

Appendix A. Monitors and the fixed-bound game

We prove Lemma 4.1, Theorem 4.2, the MSO construction of Remark 4.3, Theorem 4.4 and Proposition 4.5, after four lemmas on single runs; notation is that of Section 4, with $k = k_A + k_B$.

A.1. **Four lemmas on one run.** Fix a run $\rho = e_1 \cdots e_n$ of a copyless SST through states $q_0, \ldots, q_n$, and write $v_i(X)$ for the content of X after step i. Its *use bits* $l_i : \mathcal{X} \to \{0, 1\}$ are defined backwards. Put $l_n(X) = 1$ iff X occurs in $\Phi(q_n)$. For $i \geq 1$, put $l_{i-1}(X) = l_i(Y)$ if X occurs in $\mathrm{upd}(e_i)(Y)$, and $l_{i-1}(X) = 0$ if X occurs in no $\mathrm{upd}(e_i)(Y)$. The register Y is unique by copylessness, so l_n fixes every l_i. We call X *live* at time i if $l_i(X) = 1$, and *dead* otherwise.

Lemma A.1 (liveness)**.** *For every $i \leq n$ and every register Y:*

(a) *if Y is live at time i, then $v_i(Y)$ occurs in the annotated output of ρ as one contiguous factor, its placement, and the placements of distinct live registers are disjoint;*
(b) *if Y is dead at time i, no letter of $v_i(Y)$ reaches the output;*
(c) *a letter occurrence in $\mathrm{upd}(e_i)(Y)$ reaches the output iff Y is live at time i, and then exactly once.*

Proof. Backward induction on i. At $i = n$ the output is $v_n(Z_1)\cdots v_n(Z_m)$ for $\Phi(q_n) = Z_1 \cdots Z_m$, each register at most once, which gives (a) and (b). For the step from i to $i-1$ write $\mathrm{upd}(e_i)(Y) = u_0X_1u_1\cdots X_ru_r$, so that $v_i(Y) = u_0\, v_{i-1}(X_1)\, u_1 \cdots v_{i-1}(X_r)\, u_r$. Each register occurs in at most one such expression, and there at most once. If Y is live, its placement splits into the contiguous placements of the X_j and of the letters of the u_j, which are disjoint; this gives (a) for the X_j and (c) for the letters of the u_j. If Y is dead, so are the X_j, and nothing from $v_i(Y)$ is placed. A register occurring in no expression is discarded, and its content never reaches the output. □

So the output positions born at step i are exactly the letter occurrences of $\mathrm{upd}(e_i)$ inside the expressions of registers live at time i.

Let $\mathcal{P} = (P, I, F, \to)$ be an NFA over an alphabet Λ that is complete and has an absorbing non-accepting sink z: $z \notin F$ and $z \xrightarrow{\lambda} z$ for every λ, hence $z \xrightarrow{x} z$ for every word x. Suppose the update letters of ρ carry colours, so that its output is a word over $\Lambda = \Gamma \times \mathrm{Colours}$. An *endpoint labelling* is a family $f_i : \mathcal{X} \to P \times P$, $0 \le i \le n$, with components $f_i(X) = (f_i(X)^-, f_i(X)^+)$, such that

(E0) $f_0(X) = (p_X, p_X)$ for every X, with $p_X \in P$ arbitrary;
(Ei) for $i \ge 1$ and $\mathrm{upd}(e_i)(Y) = u_0X_1u_1\cdots X_ru_r$, the automaton has paths $f_i(Y)^- \xrightarrow{u_0} f_{i-1}(X_1)^-$, $f_{i-1}(X_j)^+ \xrightarrow{u_j} f_{i-1}(X_{j+1})^-$ for $1 \le j < r$, and $f_{i-1}(X_r)^+ \xrightarrow{u_r} f_i(Y)^+$; for $r = 0$, a path $f_i(Y)^- \xrightarrow{u_0} f_i(Y)^+$;
(EF) for $\Phi(q_n) = Z_1 \cdots Z_m$: $f_n(Z_1)^- \in I$, $f_n(Z_j)^+ = f_n(Z_{j+1})^-$ for $j < m$, and $f_n(Z_m)^+ \in F$; for $m = 0$, $I \cap F \neq \emptyset$.

Lemma A.2 (endpoint labelling)**.** *An endpoint labelling exists iff the coloured output of ρ lies in $L(\mathcal{P})$.*

Proof. ($\Rightarrow$) Induction on i gives a path $f_i(Y)^- \xrightarrow{v_i(Y)} f_i(Y)^+$ for every Y: (E0) covers the empty initial contents, and (Ei) glues the paths of the X_j to the paths over the u_j. Then (EF) concatenates these paths into an accepting path on the output.

($\Leftarrow$) Fix an accepting path on the output. For Y live at time i let $f_i(Y)$ be the states of the path just before and just after the placement of $v_i(Y)$ (Lemma A.1(a)); an empty placement gives a pair (p, p). For dead Y put $f_i(Y) = (z, z)$. For live Y, (Ei) holds because the placement splits as in the proof of Lemma A.1 and the X_j are live. For dead Y every X_j is dead, and $z \xrightarrow{u} z$ for every u. (E0) holds because registers start empty, and (EF) is the accepting path. □

Filiot, Jecker, Löding and Winter introduced endpoint pairs [FJLW23]. The sink is needed: a dead register places nothing, so without z it might have no admissible pair.

Let α_A, α_B be annotated words with the same underlying word of length N, and let $b_A(x)$, $b_B(x)$ be the births at rank x. Put $O_A(t) = \{x : b_A(x) \le t\}$, and $O_B(t)$ likewise.

Lemma A.3 (old sets)**.** *(a)* $\mathrm{od}(\alpha_A, \alpha_B) \le H$ *iff* $O_A(t) \subseteq O_B(t+H)$ *and* $O_B(t) \subseteq O_A(t+H)$ *for every* t*.*

(b) If α_A *is the output of a run of a copyless SST with* k_A *registers, then* $O_A(t)$ *is the disjoint union of the placements of the registers live at time* t*, hence a union of at most* k_A *intervals of* $[1, N]$*.*

Proof. (a) If $\mathrm{od} \le H$ and $b_A(x) \le t$, then $b_B(x) \le b_A(x) + H \le t + H$, and symmetrically. Conversely $t = b_A(x)$ gives $b_B(x) \le b_A(x) + H$, and $t = b_B(x)$ the other inequality.

(b) A letter born at or before t that reaches the output sits at time t in some register, which is live by Lemma A.1(b). Every letter in a live register at time t reaches the output by Lemma A.1(a) and was born at or before t. Placements are contiguous and disjoint. □

Lemma A.4 (fork depth)**.** *Let* B *be input-enabled on its reachable states, as under* $\mathbf{G}$*, with at most* K *runs on every word. Let* β *be a run of* B *on* $w = a_1 \cdots a_n$ *through states* $q_0, \ldots, q_n$*, and put* $\phi_i = |\Delta_B(q_{i-1}, a_i)|$*. Then* $\sum_{i \le n}(\phi_i - 1) \le K - 1$*. So at most* $K - 1$ *positions have* $\phi_i \ge 2$*, and* $\phi_i \le K$ *everywhere.*

Proof. For each i and each $e \in \Delta_B(q_{i-1}, a_i)$ other than the edge of β, let $\beta_{i,e}$ follow β up to step $i-1$, take e at step i and continue on $a_{i+1} \cdots a_n$ by any edges; these exist because every state reached is reachable and B is input-enabled there. These runs and β are pairwise distinct. $\beta_{i,e}$ differs from β at step i; $\beta_{i,e}$ and $\beta_{i,e'}$ differ at step i when $e \ne e'$; for $i < j$, $\beta_{i,e}$ leaves β at step i while $\beta_{j,e'}$ still follows it there. All are runs on w, so $1 + \sum_i (\phi_i - 1) \le K$. □

Input-enabledness is essential. Example 5.1 gives a trim unambiguous B with non-final states where the sum is unbounded; Section 5 replaces this lemma by its live-fork bound.

A.2. **The least bad rank.**

Proof of Lemma 4.1. Ranks $y < c$ with both births at most r cancel, so

$$D(r) = \#\{y < c : b_A(y) \le r < b_B(y)\} - \#\{y < c : b_B(y) \le r < b_A(y)\}.$$

In the first set $b_B(y) \le b_A(y) + H$ and $b_B(y) > r$, so $b_A(y) \in (r - H, r]$. That interval has at most H steps, and each step is the birth of at most μ_A positions of α_A. So the first set has at most $\mu_A H$ elements, and by symmetry the second has at most $\mu_B H$. □

For a well-formed joint word with outputs α_A, α_B of lengths N_A, N_B, a rank c is *bad* if $c \le \min(N_A, N_B)$ and the two words differ at c in letter or by more than H in birth, or if $c = \min(N_A, N_B) + 1$ and $N_A \neq N_B$. The *least bad rank* is the least such c. It exists iff the word lies outside L_H^{od}, and every rank below it meets the hypothesis of Lemma 4.1.

A.3. **Proof of Theorem 4.2.**

Proof. Colour automata. We use two complete DFAs, each with an absorbing non-accepting sink z. $\mathcal{P}_{\mathrm{mk}}$ over $\Gamma \times \{\mathsf{P}, \mathsf{M}, \mathsf{S}\}$ recognises $\mathsf{P}^*\mathsf{M}\mathsf{S}^*$ on the colour track, with states pre (initial), post (accepting) and z. $\mathcal{P}_{\mathrm{all}}$ over $\Gamma \times \{\mathsf{P}\}$ has the single state pre, initial and accepting, besides z.

States. A state of $\mathcal{N}_H$ is a tuple $(\tau, l^A, l^B, f^A, f^B, \eta^A, \eta^B, R, G, \hat{\gamma})$.

- $\tau \in \{\mathrm{GAP}, \mathrm{LET}, \mathrm{LEN}_A, \mathrm{LEN}_B\}$ is the *type*, guessed once and kept. Clone $X \in \{A, B\}$ is *unmarked* and uses $\mathcal{P}_\tau^X = \mathcal{P}_{\mathrm{all}}$ if $(\tau, X) \in \{(\mathrm{LEN}_A, B), (\mathrm{LEN}_B, A)\}$; otherwise it is *marked* and uses $\mathcal{P}_\tau^X = \mathcal{P}_{\mathrm{mk}}$.
- $l^X \in \{0,1\}^{\mathcal{X}_X}$ are use bits and $f^X : \mathcal{X}_X \to (P_\tau^X)^2$ are endpoint maps, together at most $2^k \cdot 9^k = 18^k$ choices.
- $\eta^X \in \{0,1\}$ are mark flags, $R \in [-\mu_B H, \mu_A H]$ is the rank counter, $G \in [0, H+1]$ the gap counter, and $\hat{\gamma} \in \Gamma \cup \{\bot\}$ a stored letter.

Initial states have any τ, any l^X, diagonal $f^X(Y) = (p_Y, p_Y)$ with any p_Y as in (E0), $\eta^X = 0$, $R = G = 0$ and $\hat{\gamma} = \bot$. One further state is accepting and has no outgoing transition.

Transitions on $(\sigma_A, \sigma_B) \in S_A \times S_B$. Guess new use bits l'^X, new endpoint maps f'^X, and a colour of $\mathcal{P}_\tau^X$ for every letter occurrence of σ_X. Require:

(T1) l^X arises from l'^X by the backward law of Section A.1, with σ_X as the update;
(T2) f^X and f'^X satisfy (Ei) for $\mathcal{P}_\tau^X$ and the chosen colours;
(T3) for a marked clone, $\eta^X + \#_{\mathsf{M}}^X \le 1$, where $\#_{\mathsf{M}}^X$ counts the M-coloured occurrences in $\sigma_X(Y)$ over the registers Y with $l'^X(Y) = 1$; then $\eta'^X = \eta^X + \#_{\mathsf{M}}^X$;
(T4) $R' = R + \#_{\mathsf{P}}^A - \#_{\mathsf{P}}^B$, with $\#_{\mathsf{P}}^X$ counted the same way, and $R' \in [-\mu_B H, \mu_A H]$;
(T5) if $\tau = \mathrm{GAP}$, then $G' = \min(H+1, G + [\eta'^A \neq \eta'^B])$;
(T6) if $\tau = \mathrm{LET}$, consider the M-occurrences counted in (T3). If exactly one clone counts one and $\hat{\gamma} = \bot$, store its letter in $\hat{\gamma}'$. If exactly one clone counts one and $\hat{\gamma} \neq \bot$, its letter must differ from $\hat{\gamma}$. If both clones count one, the two letters must differ.

Transition on a final pair (Φ_A, Φ_B). Require for both clones that $l^X(Y) = 1$ iff Y occurs in Φ_X, and that f^X satisfies (EF) for $\mathcal{P}_\tau^X$. Move to the accepting state iff $R = 0$ and the type condition holds: $G = H+1$ for GAP; nothing further for LET, since (EF) forces $\eta^A = \eta^B = 1$ and (T6) compared the two letters; nothing further for LEN_X. Only well-formed words are

accepted, because the accepting state is entered on a final pair and never left.

Size. The factors multiply to $4 \cdot 18^k \cdot 4 \cdot ((\mu_A + \mu_B)H + 1)(H + 2)(|\Gamma| + 1)$, plus the accepting state. An unmarked clone has only 4^{k_X} endpoint maps, within the bound.

Soundness. Take an accepting run of $\mathcal{N}_H$ on w. The final transition fixes l^X at the end and (T1) enforces the backward law, so the use bits are the true ones and (T3), (T4) count live occurrences only. By Lemma A.2 the coloured output of each clone lies in $L(\mathcal{P}_\tau^X)$, and by Lemma A.1(c) each live occurrence is one output letter. So a marked clone has exactly one M-letter, at some rank c^X, and exactly $c^X - 1$ live P-occurrences; an unmarked clone has all N_X live occurrences coloured P. The condition $R = 0$ at the end equates the two P-counts.

- GAP: $c^A = c^B =: c$. The flag η^X turns 1 in the row that bears the live M-occurrence, so after row r we have $\eta^X = 1$ iff $b_X(c) \le r$. The rows with $\eta^A \neq \eta^B$ lie between the two births, and $G = H + 1$ says there are at least $H + 1$ of them, that is $|b_A(c) - b_B(c)| > H$. If the underlying words are equal this is od $> H$; if they differ, $w \notin L_H^{\mathrm{od}}$ anyway.
- LET: $c^A = c^B$ and the letters at that rank differ, so the words differ.
- LEN_A: $c^A - 1 = N_B$, so $N_A \ge N_B + 1$. LEN_B is symmetric.

Completeness. Let w be well formed and outside L_H^{od}, with least bad rank c. Choose τ by the reason c is bad: a birth gap above H at equal letters gives GAP; a letter mismatch gives LET; $c = \min(N_A, N_B) + 1$ with $N_A > N_B$ gives LEN_A, and symmetrically LEN_B. Colour the live occurrences of a marked clone by their rank relative to c (P below, M at, S above), every live occurrence of an unmarked clone P, and dead occurrences arbitrarily. Take the true use bits and the endpoint maps of Lemma A.2 ($\Leftarrow$), with the sink for dead registers. After row r the counter equals $D(r)$ of Lemma 4.1: for a marked clone the live P-occurrences born by r are the ranks below c born by r, and for an unmarked clone every rank is below $c = N_X + 1$. All ranks below c are good, so the lemma keeps R in range at every row, and $R = 0$ at the end because both clones have $c - 1$ ranks below c. The type condition holds by the choice of τ. Two empty outputs have no bad rank and are rejected, as they should be since od $= 0$. The case $N_A > 0 = N_B$ is caught by LEN_A with $c = 1$.

Determinisation and computation. The subset construction gives a DFA for $L(\mathcal{N}_H)$ with at most $2^{|\mathcal{N}_H|}$ states. Well-formedness is recognised by a DFA with three states: only substitution pairs seen, one final pair seen last, and a sink. Their product, accepting iff the second factor accepts and the first rejects, recognises L_H^{od} with at most $3 \cdot 2^{|\mathcal{N}_H|}$ states. A transition of $\mathcal{N}_H$ is checked by a dynamic programme over the letter occurrences of (σ_A, σ_B) that tracks the colour-automaton state and the two counts, in time polynomial in the lengths of the substitutions. So $\mathcal{N}_H$ is computable in

time polynomial in its state count and in the lengths of S_A, S_B, F_A, F_B, and the DFA in time polynomial in its own state count. Each edge of A or B determines its substitution, so the same bounds hold for the edge-level version R_H^{od} over $E_A \times E_B$. □

Lemma A.5 (the gap type suffices under word agreement)**.** *Let w be a well-formed joint word whose two outputs have the same underlying word. Then $\mathrm{od}(\alpha_A(w), \alpha_B(w)) > H$ iff $\mathcal{N}_H$ has an accepting run of type* GAP *on w. On such words the types* LET*,* LEN_A*,* LEN_B *and the component $\hat{\gamma}$ can be dropped, which removes the factor $4(|\Gamma| + 1)$.*

Proof. A GAP run forces a birth gap above H at a common rank, by soundness. Conversely, with equal underlying words the least bad rank is a birth gap, and the run built in the completeness part has type GAP. □

In the arena of Section A.5 the monitor is consulted only on words of $\mathrm{dom}(B)$ with an accepting run of B, where (R4) of Definition 2.2 gives word agreement; under **G** this is (G4). Without word agreement all four types are needed (Appendix H).

A.4. **The MSO route.** A joint word $w = (\sigma_1^A, \sigma_1^B) \cdots (\sigma_n^A, \sigma_n^B)(\Phi_A, \Phi_B)$ is a structure with positions $1, \ldots, n+1$ and the letter of each position. An *A-occurrence* is a pair $o = (i, k)$ with $1 \le i \le n$ and $1 \le k \le |\sigma_i^A|_\Gamma \le \mu_A$, the k-th letter occurrence of $\sigma_i^A(X_1) \cdots \sigma_i^A(X_{k_A})$; its letter $\gamma(o)$ and birth $b(o) = i$ are read off the letter at position i. It is *live* if it reaches $\alpha_A(w)$, and $<_A$ orders live occurrences by rank in $\alpha_A(w)$. B-occurrences, their liveness and $<_B$ are defined likewise. In MSO over joint words an occurrence variable is a position variable plus a slot fixed by a finite disjunction, and a set of occurrences is a tuple of position sets indexed by slots. The slot is only an address; the order among letters of one step comes from Bef.

Lemma A.6 (definability)**.** *For $X \in \{A, B\}$ and slots k, k' there are MSO formulas $\mathrm{Live}_k^X(i)$ and $\mathrm{Bef}_{k,k'}^X(i, i')$ such that, on well-formed words, $\mathrm{Live}_k^X(i)$ holds iff (i, k) is a live X-occurrence, and $\mathrm{Bef}_{k,k'}^X(i, i')$ holds iff $(i, k) \neq (i', k')$, both are live, and $(i, k) <_X (i', k')$.*

Proof. The X-projection of w is a sequence of copyless substitutions followed by a final expression, and Lemma A.2 uses nothing else.

For Bef take the complete four-state DFA with absorbing sink for $0^*1\,0^*2\,0^*$ on the colour track over $\Gamma \times \{0, 1, 2\}$. Colour (i, k) with 1, (i', k') with 2, and every other occurrence with 0. The coloured output lies in the language iff both occurrences are live and the first precedes the second, because a dead occurrence contributes no letter. By Lemma A.2 this is equivalent to the existence of an endpoint labelling $(f_j)_{0 \le j \le n}$ with $f_j : \mathcal{X}_X \to P^2$. Encode $f_1, \ldots, f_n$ by a partition of the positions $1, \ldots, n$ into the finitely many sets F_g, $g \in (P^2)^{\mathcal{X}_X}$, with position j in F_{f_j}. The diagonal values $(p_Y)_Y$ of f_0 are free, so the formula takes a finite disjunction over them and checks (E1) against the chosen f_0. Each of (Ej) and (EF)

is a condition on f_{j-1}, f_j, the letter at position j, and whether j is i or i' with the right slot; this is first-order in the successor relation. So

$$\mathrm{Bef}^X_{k,k'}(i,i') \;=\; \bigvee_{(p_Y)_Y \in P^{\mathcal{X}_X}} \exists (F_g)_g \,\big[\text{partition} \wedge \text{local conditions}\big].$$

Live is the same construction with the language $0^*1\,0^*$. □

Lemma A.7 (matching characterisation)**.** *A well-formed w lies in L^{od}_H iff there is a map f from the live A-occurrences to the live B-occurrences such that (M1) f is a bijection; (M2) $\gamma(f(o)) = \gamma(o)$; (M3) $|b(f(o)) - b(o)| \le H$; (M4) $o <_A o'$ implies $f(o) <_B f(o')$.*

Proof. ($\Rightarrow$) Equal words have a common length N. Let f send the occurrence of rank x in α_A to the occurrence of rank x in α_B. (M1), (M2) and (M4) are immediate, and (M3) is od $\le H$.

($\Leftarrow$) By (M1) both outputs have the same length N. Let o have A-rank r. By (M4) the $r-1$ occurrences below o go to distinct occurrences below $f(o)$, and the $N-r$ above o to distinct occurrences above $f(o)$. These exhaust the $N-1$ occurrences other than $f(o)$, so $f(o)$ has B-rank r. Hence f is the rank map; (M2) says the words agree at every rank, and (M3) says od $\le H$. □

Theorem A.8 (MSO monitor)**.** *For every H, L^{od}_H is effectively regular, and so is its inverse image R^{od}_H under the letter-to-letter map from $(E_A \times E_B)^*$ followed by a final pair to joint words.*

Proof. We encode f by finitely many monadic labels. For each A-slot $k \le \mu_A$, offset $\delta \in [-H, H]$ and B-slot $k' \le \mu_B$ take a set variable $M_{k,\delta,k'}$, read as "$i \in M_{k,\delta,k'}$ iff $f(i,k) = (i+\delta, k')$", and put $\mathrm{Map}(i,k;j,k') = \bigvee_\delta \big(i \in M_{k,\delta,k'} \wedge j = i+\delta\big)$; "$j = i+\delta$" is first-order. The sentence is

$$\exists (M_{k,\delta,k'})_{k,\delta,k'} \;\big[\mathrm{WF} \wedge \mathrm{Fun} \wedge \mathrm{Bij} \wedge \mathrm{Let} \wedge \mathrm{Ord}\big],$$

where WF says the word is well formed; Fun says every live A-occurrence lies in exactly one $M_{k,\delta,k'}$, its target $(i+\delta, k')$ exists and satisfies $\mathrm{Live}^B_{k'}$, and dead occurrences lie in none; Bij says every live B-occurrence has exactly one Map-preimage, a finite disjunction over (k,δ); Let says $\mathrm{Map}(o,o')$ implies $\gamma(o) = \gamma(o')$, which is local in the two letters; and Ord says that $\mathrm{Bef}^A(o_1,o_2)$, $\mathrm{Map}(o_1,o_1')$ and $\mathrm{Map}(o_2,o_2')$ imply $\mathrm{Bef}^B(o_1',o_2')$. A word of L^{od}_H satisfies the sentence under the labelling by the rank map, whose offsets lie in $[-H,H]$ by (M3). Conversely a satisfying labelling defines a map with (M1)–(M4), and Lemma A.7 applies. The label set has $\mu_A(2H+1)\mu_B$ elements, and an MSO sentence over finite words defines a regular language effectively. Two empty outputs are accepted, in agreement with od $= \max\emptyset = 0$. Each edge determines its substitution, so R^{od}_H is the inverse image of a regular language under a letter-to-letter map. □

This route gives no useful size bound; Theorem 4.2 does.

A.5. **The fixed-bound game.** Fix an instance (A, B) satisfying $\mathbf{G}^\flat$ and $d, H \in \mathbb{N}$.

The history arena $\mathfrak{G}_{d,H}$. A position is a history: the letters revealed so far, a flag eof $\in \{0, 1\}$, and the edges the controller has chosen. Write buf for the revealed letters not yet consumed.

- The environment moves when eof $= 0$ and $|\text{buf}| < d + 1$: it reveals one letter or sets eof $= 1$.
- The controller moves when buf $\neq \varepsilon$ and either $|\text{buf}| = d + 1$ or eof $= 1$. Let a be the first letter of buf and q_B the current state of its run. It picks $e \in \Delta_B(q_B, a)$, and a is consumed. If $\Delta_B(q_B, a) = \emptyset$, or the run is already blocked, a is consumed without an edge and the run is *blocked* from then on.
- A position with eof $= 1$ and buf $= \varepsilon$ is terminal. It determines the input w and the controller's run β. It is *bad* iff $w \in \text{dom}(B)$ and β is blocked, or ends in a non-final state, or has $\text{od}(\rho_A(w), \beta) > H$.

The environment wins a play iff it reaches a bad terminal; an infinite play, where eof never comes, goes to the controller. On $w \in \text{dom}(B) \subseteq \text{dom}(A)$ the run $\rho_A(w)$ exists, so badness is well defined.

Lemma A.9 (windows)**.** *At the controller's move for input position i of $w = a_1 \cdots a_n$,* buf $= a_i \cdots a_{\min(n,i+d)}$*, and* eof $= 1$ *iff $n - i < d$. So the controller observes exactly* $\text{win}_d(w, i)$*, and the controller strategies of $\mathfrak{G}_{d,H}$ are exactly the lookahead-d controllers with arbitrary memory.*

Proof. The flag is set only while $|\text{buf}| < d + 1$, and no letter is revealed afterwards. So eof $= 1$ at the move for i means that $a_i \cdots a_n$ is the whole buffer, of length at most d, that is $n - i < d$. If $n - i < d$ the buffer never holds $d + 1$ letters, so the move for i waits for eof $= 1$. If $n - i \geq d$ the move happens with $|\text{buf}| = d + 1$, which is $a_i \cdots a_{i+d}$. In both cases the buffer and the flag encode $\text{win}_d(w, i)$. The interleaving of moves is fixed by the rules, so a history is a function of the windows seen and the controller's own edges, and conversely. □

Hence

$$\text{GOOD}^{\flat,\infty}(d, H) \iff \text{the controller wins } \mathfrak{G}_{d,H}. \tag{4}$$

The finite arena $G^\flat_{d,H}$. Let R_H be a DFA over $E_A \times E_B$, followed by one final pair, that recognises R^{od}_H; Theorem 4.2 or Theorem A.8 gives one, and by Lemma A.5 the GAP part suffices. A position is $(q_A, q_B, S, r, \text{buf}, \text{eof})$ with $q_A \in Q_A \cup \{\bot\}$, $q_B \in Q_B \cup \{\bot\}$, $S \subseteq Q_B$, r a state of R_H or $\bot$, buf $\in \Sigma^{\leq d+1}$ and eof $\in \{0, 1\}$. The initial position is $(q^A_0, q^B_0, \{q^B_0\}, r_0, \varepsilon, 0)$, and the move rules are those of $\mathfrak{G}_{d,H}$. A controller move on a with edge e sets $q_B := \text{dst}(e)$, $q_A := \text{dst}(e_A)$ for the unique edge e_A of A from q_A on a, $r := R_H(r, (e_A, e))$, and $S := \{\text{dst}(e') : q' \in S,\ e' \in \Delta_B(q', a)\}$. A missing edge of A or of B, or a $\bot$ already present, sets the affected components among q_A, q_B, r to $\bot$. A terminal is bad iff $S \cap F_B \neq \emptyset$ and either $q_B \notin F_B$

or R_H rejects after reading $(\Phi_A(q_A), \Phi_B(q_B))$ from r. Under $\mathbf{G}$ the set S and the value $\bot$ never matter, and the positions are $(q_A, q_B, r, \mathrm{buf}, \mathrm{eof})$ with q_B reachable.

The set S is the set of states of B reachable on the consumed prefix. So at a terminal, $w \in \mathrm{dom}(B)$ iff $S \cap F_B \neq \emptyset$, and then $q_A \neq \bot$ because $\mathrm{dom}(B) \subseteq \mathrm{dom}(A)$. Mapping a history to its tuple is a quotient of $\mathfrak{G}_{d,H}$ onto $G^\flat_{d,H}$ that preserves the moves and the badness of terminals. The arena is finite and computable, because R_H is.

Lemma A.10. $\mathrm{GOOD}^\flat(d, H)$ *holds iff the controller wins* $G^\flat_{d,H}$*. A positional winning strategy yields a finite-memory controller with memory* $(Q_A \cup \{\bot\}) \times 2^{Q_B} \times$ (*states of* R_H)*; under* $\mathbf{G}$*, with memory* $Q_A \times$ (*states of* R_H)*.*

Proof. A finite-memory controller $\mathcal{C}$ induces a strategy in $G^\flat_{d,H}$, its choices being a function of the play so far by Lemma A.9. It wins iff every terminal it reaches is good, that is iff on every $w \in \mathrm{dom}(B)$ its run is accepting with $\mathrm{od} \leq H$, which is $\mathrm{GOOD}^\flat(d, H)$.

Conversely take a positional winning strategy; reachability games on finite arenas have them. It is defined at every controller position, so for every reachable q_B, which is the domain Definition 2.5 requires. Its choice depends on $(q_A, q_B, S, r, \mathrm{buf}, \mathrm{eof})$. The controller observes q_B and the window, which encodes buf and eof. The components q_A, S and r are functions of the consumed letters and the controller's own edges, so a memory $(Q_A \cup \{\bot\}) \times 2^{Q_B} \times (\text{states of } R_H)$ tracks them, updated at each move from the consumed letter and the chosen edge. Under $\mathbf{G}$, S and $\bot$ are dropped. □

Proof of Theorem 4.4. Lemma A.10 gives $\mathrm{GOOD}^\flat(d, H)$ iff the controller wins $G^\flat_{d,H}$, and $\mathrm{GOOD}^\flat(d, H)$ implies $\mathrm{GOOD}^{\flat,\infty}(d, H)$ trivially. If the controller loses $G^\flat_{d,H}$, the environment has a positional winning strategy there. A controller $\mathcal{C}$ of arbitrary memory induces a strategy in $G^\flat_{d,H}$ by (4) and the quotient; the environment's strategy drives their play to a bad terminal, whose word is one on which $\mathcal{C}$ fails. So $\neg\mathrm{GOOD}^\flat(d, H)$ implies $\neg\mathrm{GOOD}^{\flat,\infty}(d, H)$. The arena is finite and computable, and its winner is found by computing the environment's attractor of the bad terminals. The memory bound is the second half of Lemma A.10, with $|R_H| \leq 3 \cdot 2^{|\mathcal{N}_H|}$ by Theorem 4.2 and Lemma A.5. Appendix B removes the factor 2^{Q_B} and proves Proposition 4.5. □

Corollary A.11. *Under* $\mathbf{G}^\flat$ *with no ambiguity bound,* $Q^\flat_d$ *and* $J^\flat$ *are recursively enumerable and coincide with their arbitrary-memory versions.*

Proof. Each $\mathrm{GOOD}^\flat(d, H)$ is decidable by Theorem 4.4. Enumerate H, and for $J^\flat$ also d. □

Appendix B. Kill witnesses

We prove Theorem 5.4, Theorem 4.4 and Proposition 4.5, after the lemmas on live and safe edges that Section 5 relies on, and we give the two examples behind Definition 5.3.

B.1. **Co-accessibility, live edges, safe edges.** For $p \in Q_B \cup \{z\}$ let $L(p)$ be the set of words on which B has an accepting run from p; so $L(z) = \emptyset$. For $P \subseteq Q_B$ put $L(P) = \bigcup_{p\in P} L(p)$, and let $\delta_B(P, u) \subseteq Q_B$ be the set of states that runs of B from P on u reach. Thus $\delta_B(\{q_0^B\}, u)$ is the subset construction of B, with z left out.

Fix $w = a_1 \cdots a_n$ and a run $\beta = e_1 \cdots e_n$ of B_z on w with states $q_0, \ldots, q_n$. For $0 \le i \le n$ the *co-accessibility set* is

$$C_i(w) = \{p \in Q_B :\ a_{i+1} \cdots a_n \in L(p)\},$$

and for $C \subseteq Q_B$ and a letter a we put $\mathrm{pre}_a(C) = \{p \in Q_B :\ \mathrm{dst}(e) \in C \text{ for some } e \in \Delta_B(p, a)\}$. The *live edges* at i are

$$\mathrm{Live}_w(\beta, i) = \{e \in \Delta_{B_z}(q_{i-1}, a_i) :\ \mathrm{dst}(e) \in C_i(w)\};$$

the set depends on β only through q_{i-1}. The other edges of $\Delta_{B_z}(q_{i-1}, a_i)$ are *dead for w at i*. The *live alternatives* at i are $\mathrm{Live}_w(\beta, i) \setminus \{e_i\}$.

Lemma B.1 (co-accessibility)**.** *(a) $C_n(w) = F_B$ and $C_{i-1}(w) = \mathrm{pre}_{a_i}(C_i(w))$ for $1 \le i \le n$. These two equations determine the sequence: any $C'_0, \ldots, C'_n \subseteq Q_B$ with $C'_n = F_B$ and $C'_{i-1} = \mathrm{pre}_{a_i}(C'_i)$ for all i equals $(C_i(w))_i$. Moreover $w \in \mathrm{dom}(B)$ iff $q_0^B \in C_0(w)$.*

(b) If e_i is dead for w at i, then $q_j \notin C_j(w)$ for every $j \ge i$. So β is not accepting, and $\mathrm{Live}_w(\beta, j) = \emptyset$ for every $j > i$.

(c) $z \notin C_i(w)$ for every i. Every z-edge is dead, for every word at every position.

Proof. (a) A word $a_i \cdots a_n$ lies in $L(p)$ iff some $e \in \Delta_B(p, a_i)$ has $a_{i+1} \cdots a_n \in L(\mathrm{dst}(e))$. That is the backward law, and the law fixes the sequence by downward induction from $i = n$. The last claim is the definition of $\mathrm{dom}(B)$.

(b) $q_i = \mathrm{dst}(e_i) \notin C_i(w)$ by deadness. Suppose $q_j \notin C_j(w)$ for some $j \ge i$. If e_{j+1} enters z, then $q_{j+1} = z \notin C_{j+1}(w)$. Otherwise $e_{j+1} \in \Delta_B(q_j, a_{j+1})$, and $q_{j+1} \in C_{j+1}(w)$ would put q_j into $\mathrm{pre}_{a_{j+1}}(C_{j+1}(w)) = C_j(w)$. Hence $q_n \notin C_n(w) = F_B$. A live edge at some $j > i$ would have its source q_{j-1} in $\mathrm{pre}_{a_j}(C_j(w)) = C_{j-1}(w)$.

(c) $C_i(w) \subseteq Q_B$ by definition. □

Consistent words and safe edges. Fix d. For $1 \le i \le n$ let $\mathrm{eof}_i(w) = [\, n - i < d\,]$; this flag says whether $\mathrm{win}_d(w, i)$ shows $\dashv$. Put $x_i = a_{i+1} \cdots a_{\min(n,i+d)}$ and $S_{i-1} = \delta_B(\{q_0^B\}, a_1 \cdots a_{i-1})$. The words *consistent* with w at i are

$$\mathrm{Cons}_d(w, i) = \{w' \in \mathrm{dom}(B) :\ w'[1..\min(n, i+d)] = w[1..\min(n, i+d)], \text{ and } w' = w \text{ if } \mathrm{eof}_i(w)\}.$$

For $q \in Q_B \cup \{z\}$ the *safe edges* are

$$\mathrm{Safe}_d(w,q,i) = \{e \in \Delta_{B_z}(q,a_i) :\ \mathrm{dst}(e) \in C_i(w') \text{ for every } w' \in \mathrm{Cons}_d(w,i)\}.$$

If $\mathrm{Cons}_d(w,i) = \emptyset$, every edge is safe. For a state q, a letter a, a word x and a flag $\eta \in \{0,1\}$, the *state-based safe set* is

$$\mathrm{SafeQ}_d(q,a,x,\eta) = \begin{cases} \{e \in \Delta_{B_z}(q,a) :\ (ax)^{-1}L(q) \subseteq x^{-1}L(\mathrm{dst}e)\} & \eta = 0, \\ \{e \in \Delta_{B_z}(q,a) :\ ax \in L(q) \Rightarrow x \in L(\mathrm{dst}e)\} & \eta = 1. \end{cases}$$

Lemma B.2 (observability)**.** $\mathrm{Safe}_d(w,q,i)$ *is a function of* $(q, S_{i-1}, a_i, x_i, \mathrm{eof}_i(w))$. *Without* $\dashv$*, an edge* e *is safe iff* $(a_i x_i)^{-1}L(S_{i-1}) \subseteq x_i^{-1}L(\mathrm{dst}e)$*. With* $\dashv$*,* e *is safe iff* $a_i x_i \in L(S_{i-1})$ *implies* $x_i \in L(\mathrm{dst}e)$*. A controller with lookahead* d *observes all five data, since* S_{i-1} *is a function of its past windows.*

Proof. Without $\dashv$ we have $n \geq i + d$, and $\mathrm{Cons}_d(w,i)$ consists of the words $a_1 \cdots a_i x_i v$ in $\mathrm{dom}(B)$, which are those with $a_i x_i v \in L(S_{i-1})$. For such a word, $\mathrm{dst}(e) \in C_i(w')$ iff $x_i v \in L(\mathrm{dst}e)$. With $\dashv$, x_i is the rest of w, so $\mathrm{Cons}_d(w,i) = \{w\} \cap \mathrm{dom}(B)$, and $w \in \mathrm{dom}(B)$ iff $a_i x_i \in L(S_{i-1})$. □

B.2. **Two examples.** Call a kill witness *complete-fork* if at every internal node ν every edge of $\Delta_{B_z}(q_\nu, a_\nu)$ is taken by some child, and at every non-fork position of every leaf only one edge is available. Then (K3) holds for free, because every edge it speaks about is taken by a leaf of $\Lambda(\lambda, p)$. Under **G** these are the witnesses of Remark 5.5. Outside **G** they do not suffice, even with $K = 2$ and B trim.

Example B.3 (complete forks need unbounded shapes)**.** Extend the instance of Example 2.7 by a letter \$, so $\Sigma = \{a, \#, 0, 1, \$\}$. In A, the states pre and wait go to dF on \$ with the identity update, and dF, dR loop on \$ with the identity; A stays deterministic, complete and all-final. In B, the states mF and mR loop on \$ with the identity, and pre goes to mF on \$ with the identity. Add a non-final state τ with one edge $\mathrm{pre} \xrightarrow{c} \tau$ for each $c \in \{a, 0, 1\}$, carrying the update of the c-edge at pre, and the single edge $\tau \xrightarrow{\$/\mathrm{id}} \mathrm{mF}$.

The promise holds. The states pre, mF, mR are final and pre reads every letter, so $\mathrm{dom}(B) = \Sigma^* = \mathrm{dom}(A)$. Since $L(\tau) = \$\Sigma^*$, B is trim. Let k count the a's before the first letter in $\{\#, \$\}$, or in the whole word. A outputs the F-word or the R-word with k blocks. An accepting run of B either avoids τ, and then behaves as in Example 2.7, or enters τ at the letter before the first \$ with no # before it, and then outputs F with the same k blocks and the same births. So (R4) holds. The choice at the first # and the choice of τ exclude each other, so $K = 2$.

Lookahead 0 fails. The words $a^k\#$ and $a^k\#1$ with $k \geq H + 2$ contain no \$. A controller that enters τ on them fails acceptance, and one that avoids τ loses on one of them as in Proposition 2.8, with $\mathrm{od} = k - 1 > H$. So $\neg Q_0^{\flat,\infty}$.

Fix H and a 0-coherent complete-fork witness for the bound H. Some leaf is OD. Otherwise follow the controller "never τ; at # choose mF" down

the tree: at a fork its edge is taken by a child, elsewhere the edge is forced, so it ends at a leaf whose run is the controller's own run on that word, and that run is accepting. Let λ be an OD leaf. If \$ precedes every # in u_λ, every accepting run outputs F with A's births, and od $= 0$. So the first # of u_λ precedes every \$, and the run of λ stays in pre up to that #, since entering τ earlier would meet a letter other than \$ next. Every accepting run has od at most the distance between the first and the last a before that #, so the # is at a position $\geq H + 2$. Each of the positions $1, \ldots, H + 1$ reads a letter of $\{a, 0, 1\}$ in pre, where two edges are available. So each is the fork position of an ancestor of λ. The path to λ has more than H forks.

No finite set of shapes serves every H. A kill witness in the sense of Definition 5.3 needs one fork: take the two leaves $a^k\#$ and $a^k\#1$ of Proposition 2.8. Every τ-edge is dead on both words, because no \$ follows.

Example B.4 (the safe region depends on d)**.** Let $\Sigma = \{a, b, c\}$. Let B be the NFA with states $q_0, \ldots, q_5$, final state q_5, empty updates and edges

$$q_0 \xrightarrow{a} q_1,\ q_0 \xrightarrow{a} q_2; \quad q_1 \xrightarrow{a} q_5;$$

$$q_2 \xrightarrow{a} q_3,\ q_2 \xrightarrow{a} q_4,\ q_2 \xrightarrow{b} q_5; \quad q_3 \xrightarrow{a} q_3,\ q_3 \xrightarrow{b} q_5; \quad q_4 \xrightarrow{a} q_4,\ q_4 \xrightarrow{c} q_5,$$

and q_5 looping on every letter. Let A have one state, complete, with empty updates. Then $L(q_1) = a\Sigma^*$, $L(q_2) = b\Sigma^* \cup aa^*(b + c)\Sigma^*$, $L(\{q_1, q_2\}) = a\Sigma^* \cup b\Sigma^* = a^{-1}L(q_0)$ and $L(q_0) = a(a + b)\Sigma^*$. Look at position 1, where $a_1 = a$ and $S_0 = \{q_0\}$.

With $d = 0$ there is no $\dashv$ and $x = \varepsilon$. The set $L(\{q_1, q_2\})$ lies in neither $L(q_1)$, which misses b, nor $L(q_2)$, which misses aa. So $\mathrm{Safe}_0 = \emptyset$, and no lookahead-0 controller accepts all of $\mathrm{dom}(B)$.

With $d = 1$ and next letter a: $a^{-1}L(\{q_1, q_2\}) = \Sigma^* = a^{-1}L(q_1)$, while $a^{-1}L(q_2) = a^*(b + c)\Sigma^*$. So $\mathrm{Safe}_1 = \{q_0 \to q_1\}$. With next letter b: $b^{-1}L(q_2) = \Sigma^*$ and $b^{-1}L(q_1) = \emptyset$, so $\mathrm{Safe}_1 = \{q_0 \to q_2\}$. The controller that follows these choices accepts $\mathrm{dom}(B)$ with lookahead 1.

Löding [Lö23] reports this automaton, due to Hugenroth, as the counterexample to Lemma 3 of [LR13]: it has a lookahead-1 controller, and no set Q' of states of the kind that lemma asserts.

B.3. **Live forks and safe edges.**

Lemma B.5 (live forks)**.** *Assume the $K^\flat$-promise. Let $w \in \mathrm{dom}(B)$ and let $\beta = e_1 \cdots e_n$ be a run of B_z on w. Then*

$$\sum_{i=1}^{n} \left|\mathrm{Live}_w(\beta, i) \setminus \{e_i\}\right| \ \leq\ K - [\beta \textit{ accepting}].$$

Proof. Let e be a live alternative at i. Its target lies in $C_i(w) \subseteq Q_B$, so $q_{i-1} \neq z$: the only edge from z is its loop, which is never live by Lemma B.1(c). Since z absorbs, $e_1 \cdots e_{i-1}$ is a run of B. Pick an accepting run γ of B from $\mathrm{dst}(e)$ on $a_{i+1} \cdots a_n$, and set $\beta^{i,e} = e_1 \cdots e_{i-1}\, e\, \gamma$. This is an accepting run of B on w. These runs are pairwise distinct and distinct from β: two with

the same i differ at i; for $i < j$, $\beta^{i,e}$ leaves β at i while $\beta^{j,e'}$ still follows it there; and each $\beta^{i,e}$ leaves β at i. Together with β, if accepting, they are distinct accepting runs on w. There are at most K of them. □

Remark B.6 (bounded live branching is finite ambiguity). The converse holds. Suppose that along every accepting run β on every w the sum of Lemma B.5 is at most M. The accepting runs on w form a prefix tree. At a branch point at position i the branches take distinct edges, each live for w at i, because an accepting run continues from it. So a branch point of degree g on a root-to-leaf path β contributes $g-1$ live alternatives of β, and $\sum(\deg -1) \le M$ on every path. Such a tree has at most 2^M leaves, by induction on the tree: a root of degree g leaves a budget of $M-g+1$ to each subtree, and $g \cdot 2^{M-g+1} \le 2^M$. So B is finitely ambiguous with $K \le 2^M$. Measured by forks into co-accessible states instead, the condition is strictly stronger: Example 5.1 is trim, has $K = 1$, and has n such forks on a^n.

Lemma B.7 (safe edges)**.** *Fix d, a word w of length n, a position $i \le n$ and a state q.*

(i) For $w' \in \mathrm{Cons}_d(w,i)$ and all $p \le i$: $\mathrm{win}_d(w',p) = \mathrm{win}_d(w,p)$.

(ii) If $e \in \mathrm{Safe}_d(w,q,i)$ and $w' \in \mathrm{Cons}_d(w,i)$, then e is live for w' at i from q.

(iii) If $e \in \Delta_{B_z}(q,a_i) \setminus \mathrm{Safe}_d(w,q,i)$, then e is dead at i for some $w' \in \mathrm{Cons}_d(w,i)$.

(iv) If $w \in \mathrm{dom}(B)$ and β is a run of B_z on w with $e_n \in \mathrm{Safe}_d(w,q_{n-1},n)$, then β is accepting. For $n = 0$ the empty run is accepting iff $\varepsilon \in \mathrm{dom}(B)$.

(v) $\mathrm{Cons}_d(w,i) \subseteq \mathrm{Cons}_d(w,i-1)$ for $i \ge 2$, and $\mathrm{Cons}_{d+1}(w,i) \subseteq \mathrm{Cons}_d(w,i)$; hence $\mathrm{Safe}_d(w,q,i) \subseteq \mathrm{Safe}_{d+1}(w,q,i)$. If $w \in \mathrm{dom}(B)$, every edge of $\mathrm{Safe}_d(w,q,i)$ is live for w at i from q.

Proof. (i) If $\mathrm{eof}_i(w)$, then $w' = w$. Otherwise $n \ge i+d$, $|w'| \ge i+d$, and the two words agree on $[1, i+d]$. For $p \le i$ the window at p reads positions $p, \dots, p+d$, all in $[1, i+d]$, and neither word shows $\dashv$ there, since $n-p \ge d$ and $|w'|-p \ge d$.

(ii) and (iii) are the definition of Safe_d.

(iv) $w \in \mathrm{Cons}_d(w,n)$, so (ii) gives $q_n = \mathrm{dst}(e_n) \in C_n(w) = F_B$. For $n = 0$, the run is accepting iff $q_0^B \in F_B$ iff $\varepsilon \in \mathrm{dom}(B)$.

(v) Let $w' \in \mathrm{Cons}_d(w,i)$. If $\mathrm{eof}_{i-1}(w)$, then $\mathrm{eof}_i(w)$, so $w' = w \in \mathrm{Cons}_d(w,i-1)$. Otherwise w' agrees with w through $\min(n, i+d) \ge \min(n, i-1+d)$ and lies in $\mathrm{dom}(B)$. For the second inclusion, write $\mathrm{eof}^{(d)}$ for the flag at lookahead d. Agreement through $\min(n, i+d+1)$ implies agreement through $\min(n, i+d)$. If $\mathrm{eof}_i^{(d)}(w)$ holds, so does $\mathrm{eof}_i^{(d+1)}(w)$, and both sets equal $\{w\} \cap \mathrm{dom}(B)$. If only $\mathrm{eof}_i^{(d+1)}(w)$ holds, then $\mathrm{Cons}_{d+1}(w,i) = \{w\} \cap \mathrm{dom}(B) \subseteq \mathrm{Cons}_d(w,i)$. Fewer consistent words leave more safe edges. The last claim is (ii) with $w' = w$. □

Lemma B.8 (acceptance is a safety constraint)**.** *Let $\mathcal{C}$ be a controller with lookahead d and any memory. Then $\rho_{\mathcal{C}}(w)$ is accepting for every $w \in \mathrm{dom}(B)$ iff, for every input w and every $i \le |w|$, the edge that $\mathcal{C}$ takes at i lies in $\mathrm{Safe}_d(w, q_{i-1}, i)$, where q_{i-1} is the state of $\rho_{\mathcal{C}}(w)$ after $i-1$ steps.*

Proof. ($\Leftarrow$) is Lemma B.7(iv). ($\Rightarrow$) Suppose $\mathcal{C}$ takes some $e \notin \mathrm{Safe}_d(w, q_{i-1}, i)$ at (w, i). Lemma B.7(iii) gives $w' \in \mathrm{Cons}_d(w, i)$ with e dead at i. By Lemma B.7(i), $\mathcal{C}$ sees the same windows on w' as on w at positions $1, \dots, i$, so it takes the same edges, reaches q_{i-1} and takes e. By Lemma B.1(b) the run $\rho_{\mathcal{C}}(w')$ is not accepting, although $w' \in \mathrm{dom}(B)$. □

Lemma B.9 (safe sets along safe runs)**.** *Let β be a run of B_z on w and $1 \le i \le n$. If $e_j \in \mathrm{Safe}_d(w, q_{j-1}, j)$ for all $j < i$, then*

$$\mathrm{Safe}_d(w, q_{i-1}, i) = \mathrm{SafeQ}_d(q_{i-1}, a_i, x_i, \mathrm{eof}_i(w)).$$

Proof. By Lemma B.2 and the definition of SafeQ_d, it suffices to show, for every word v (only $v = \varepsilon$ under $\dashv$),

$$a_i x_i v \in L(S_{i-1}) \iff a_i x_i v \in L(q_{i-1}). \tag{5}$$

($\Leftarrow$) If $a_i x_i v \in L(q_{i-1})$, then $q_{i-1} \neq z$. Since z absorbs, $e_1 \cdots e_{i-1}$ is a run of B, so $q_{i-1} \in S_{i-1}$. ($\Rightarrow$) For $i = 1$, $S_0 = \{q_0^B\} = \{q_0\}$. For $i \ge 2$, let $a_i x_i v \in L(S_{i-1})$. Then $w' = a_1 \cdots a_{i-1} a_i x_i v$ lies in $\mathrm{dom}(B)$ and in $\mathrm{Cons}_d(w, i)$; under $\dashv$, $v = \varepsilon$ and $w' = w$. By Lemma B.7(v), $w' \in \mathrm{Cons}_d(w, i-1)$. Since e_{i-1} is safe at $i-1$, Lemma B.7(ii) gives $q_{i-1} = \mathrm{dst}(e_{i-1}) \in C_{i-1}(w')$, that is, $a_i x_i v \in L(q_{i-1})$. □

So along a run that has been safe so far, the safe set at the next position is a function of the current state and the window. The subset S_{i-1} plays no part in the choice.

B.4. **Proof of Theorem 5.4.** We use the notation of Definition 5.3. For a leaf λ the padded word is $\tilde{u}_\lambda = u_\lambda \perp^\omega$. For leaves $\lambda \neq \mu$ with lowest common ancestor ν, the gap is $E(\lambda, \mu) = j - 1 - i_\nu$, where j is the first position at which $\tilde{u}_\lambda$ and $\tilde{u}_\mu$ differ, and $E(\lambda, \mu) = \infty$ if they are equal. The witness is d-coherent if every gap is at least d. By (K1) and (K2),

$$\Lambda(\lambda, p) = \{\lambda\} \cup \{\mu : \ i_{\mathrm{lca}(\lambda,\mu)} \ge p\} : \tag{6}$$

leaves whose lowest common ancestor with λ sits at or after p share letters through p and edges through $p-1$, and leaves whose lowest common ancestor sits before p take a different edge there, by injectivity. So (K3) covers the missing edges at a fork, where $p = i_\nu$, and the other edges at non-fork positions. Dead edges are relative to each leaf's own word, and no window occurs.

Let $\mathfrak{S}_K^\flat$ be the set of shapes in which every root-to-leaf path has $\sum_\nu (\deg \nu - 1) \le K$. Its members have at most 2^K leaves, by the induction in Remark B.6.

Proof of Theorem 5.4(a). Let $\mathcal{C}$ be a controller with lookahead d that achieves the objective. For a leaf λ let $\mathrm{ag}(\lambda)$ be the largest r such that $\rho_{\mathcal{C}}(u_\lambda)$ and β_λ agree on positions $1, \dots, r$. Choose λ with $r = \mathrm{ag}(\lambda)$ maximal.

If $r = |u_\lambda|$, then $\rho_{\mathcal{C}}(u_\lambda) = \beta_\lambda$, and by (K4) $\mathcal{C}$ fails on $u_\lambda \in \mathrm{dom}(B)$.

Otherwise put $p = r + 1$. On u_λ, $\mathcal{C}$ takes at p some edge $e \neq \beta_\lambda(p)$ from the state q^λ_{p-1} of β_λ. Let $\mu \in \Lambda(\lambda, p)$, $\mu \neq \lambda$. By (6), $i_{\mathrm{lca}(\lambda,\mu)} \geq p$, so by coherence the padded words agree through $i_{\mathrm{lca}} + d \geq p + d$ or are equal. The window at $p' \leq p$ is a function of the padded letters $p', \dots, p' + d$, with $\dashv$ iff the letter at $p' + d$ is $\perp$. So u_μ and u_λ show the same windows at $1, \dots, p$. Hence $\rho_{\mathcal{C}}(u_\mu)$ agrees with $\rho_{\mathcal{C}}(u_\lambda)$ on $1, \dots, p$: it follows $\beta_\mu = \beta_\lambda$ on $1, \dots, p-1$ and takes e at p. If some $\mu \in \Lambda(\lambda, p)$ has $\beta_\mu(p) = e$, then $\mathrm{ag}(\mu) \geq p > r$, against maximality. Otherwise (K3) gives $\mu \in \Lambda(\lambda, p)$ with e dead for u_μ at p. Then $\rho_{\mathcal{C}}(u_\mu)$ is not accepting by Lemma B.1(b), while $u_\mu \in \mathrm{dom}(B)$. So $\mathcal{C}$ fails. The distance bound enters only in the first case. □

Proof of Theorem 5.4(b). Assume the $K^\flat$-promise and that no controller with lookahead d, of any memory, accepts $\mathrm{dom}(B)$ with $\mathrm{od} \leq H$.

Step 1: two arenas. Let $\mathfrak{G}^\flat$ be the history arena of Section 4, played on B_z: the environment reveals letters with lookahead d and eventually declares $\dashv$, and the controller picks edges of B_z. A terminal (w, β) is bad iff $w \in \mathrm{dom}(B)$ and either β is not accepting or $\mathrm{od}(\rho_A(w), \beta) > H$. Controller strategies in $\mathfrak{G}^\flat$ are exactly the arbitrary-memory controllers with lookahead d. So the controller has no winning strategy.

At a controller move for input position i, the revealed letters and the $\dashv$ flag determine the set Cons of domain words consistent with them. It equals $\mathrm{Cons}_d(w, i)$ for every word w with that prefix and flag, and so, by Lemma B.2, the set $\mathrm{Safe}_d(w, q_{i-1}, i)$ is also determined; write Safe for it. Let $\mathfrak{G}^\flat_{\mathrm{safe}}$ be $\mathfrak{G}^\flat$ with two changes: at a controller move only edges of Safe may be chosen, and if Safe is empty the position is a terminal won by the environment. Call such a position *safe-empty*.

The environment wins $\mathfrak{G}^\flat_{\mathrm{safe}}$. The environment's objective is open, so the game is determined by the Gale–Stewart theorem. A winning strategy σ of the controller never reaches a safe-empty position. So it is a legal strategy in $\mathfrak{G}^\flat$, reaches the same terminals, and wins there. This contradicts the hypothesis. A safe-empty position has Cons $\neq \emptyset$: an empty Cons makes every edge safe, and $\Delta_{B_z}(q, a)$ is never empty.

Step 2: the safe tree. Fix a winning strategy ε of the environment in $\mathfrak{G}^\flat_{\mathrm{safe}}$ and unfold it, branching at controller positions over the safe edges. Every branch is finite, because an infinite branch would be a play consistent with ε that the environment does not win. The tree is finitely branching, so it is finite by Kőnig's lemma. Since ε is deterministic, the controller's choices determine the play. Contract the controller positions with exactly one safe

edge. The remaining branch points, with two or more safe edges, form a tree whose end points are of two kinds.

(L1) A bad terminal (w, β). Every edge of β was safe and $w \in \mathrm{dom}(B)$, so β is accepting by Lemma B.7(iv), and hence od $> H$. These become OD leaves.
(L2) A safe-empty controller position at input position i, with revealed letters $a_1 \cdots a_i x$ and an $\dashv$ flag.

The plays below a branch point at i share all letters revealed by then: those through $i + d$, or the whole word once $\dashv$ is declared. They also share the edges before i.

Step 3: fans at (L2). At an (L2) position the state is some q, the set $\Delta_{B_z}(q, a_i)$ is nonempty, and every edge in it is unsafe. Pick $e_1 \in \Delta_{B_z}(q, a_i)$ and, by Lemma B.7(iii), a word $w_1 \in \mathrm{Cons}$ with e_1 dead at i. Let E_1 be the set of edges $e \neq e_1$ that are live for w_1 at i. For each $e \in E_1$ take $w_e \in \mathrm{Cons}$ with e dead at i, and put $w_{e_1} = w_1$. For each $e \in \{e_1\} \cup E_1$ create a DEAD leaf with word w_e and the run that follows the common prefix, takes e, and then takes any edges of B_z. The run is not accepting by Lemma B.1(b), and $w_e \in \mathrm{dom}(B)$. If $|\{e_1\} \cup E_1| \geq 2$ this is a fork at i; otherwise the single leaf replaces the end point. Every edge outside $\{e_1\} \cup E_1$ is dead for w_1 at i. All w_e agree with the revealed letters. Under $\dashv$, $\mathrm{Cons} = \{w\}$, every edge is dead for w, and $E_1 = \emptyset$.

Call the leaves obtained so far *original*. For an original leaf λ and $p \leq |u_\lambda|$ let $\Lambda_0(\lambda, p)$ be $\Lambda(\lambda, p)$ computed among the original leaves. If p is at most the end point of λ's play (for a fan leaf, $p \leq i$), this is the set of original leaves below the play prefix at λ's controller move for p. For a fan leaf and $p > i$, $\Lambda_0(\lambda, p) = \{\lambda\}$, since fan leaves take distinct edges at i.

Step 4: kill leaves. Call (λ, p, e) a *gap triple* if λ is original, $p \leq |u_\lambda|$, the edge $e \in \Delta_{B_z}(q^\lambda_{p-1}, a^\lambda_p)$ is taken at p by no leaf of $\Lambda_0(\lambda, p)$, and e is live at p for every leaf of $\Lambda_0(\lambda, p)$.

In a gap triple, e is unsafe at the play prefix of λ's controller move for p. If p is a branch point, every safe edge leads to a child with a leaf below it, so e is unsafe. If p is a contracted position, its unique safe edge is the one taken. If p is an (L2) position, every edge is unsafe; in fact no gap triple arises there, since an edge outside the fan is dead for $w_1 \in \Lambda_0$. If p lies beyond an (L2) position on a fan leaf's word, then $\Lambda_0(\lambda, p) = \{\lambda\}$ and $\mathrm{Live} = \emptyset$ by Lemma B.1(b), so there is no gap triple.

For each gap triple, one per $(\Lambda_0(\lambda, p), p, e)$, take $w \in \mathrm{Cons}$ at that play prefix with e dead at p, by Lemma B.7(iii), and add a DEAD leaf with word w and the run $\beta_\lambda[1..p-1]$, then e, then any edges. This creates a fork at p above the leaves of $\Lambda_0(\lambda, p)$, or widens one. Gap triples are defined through original leaves only, so the insertions do not interact.

Step 5: verification. (K1), (K2) and (K4) hold by construction: the children of a branch point take distinct safe edges, fan leaves take distinct

edges, and an inserted leaf takes an edge that no leaf of its block takes; OD leaves are accepting with od $> H$, and DEAD leaves take a dead edge.

(K3). Take a leaf λ, a position $p \le |u_\lambda|$ and an edge e taken by no leaf of $\Lambda(\lambda, p)$. If λ is an inserted or fan leaf and p lies after its branching position, then $\Lambda(\lambda, p) = \{\lambda\}$ and $\mathrm{Live}_{u_\lambda}(\beta_\lambda, p) = \emptyset$ by Lemma B.1(b); so λ kills e. Otherwise there is an original leaf $\hat{\lambda}$ with the same edges before p and the same letters through p: $\hat{\lambda} = \lambda$ if λ is original, and an original leaf of the block where λ was inserted otherwise. Then $\Lambda_0(\hat{\lambda}, p) \subseteq \Lambda(\lambda, p)$, so no leaf of $\Lambda_0(\hat{\lambda}, p)$ takes e. If e were live for all of them, $(\hat{\lambda}, p, e)$ would be a gap triple, and its inserted leaf, which lies in $\Lambda(\lambda, p)$, would take e. So some leaf of $\Lambda_0(\hat{\lambda}, p) \subseteq \Lambda(\lambda, p)$ kills e.

Coherence. Original leaves below different children of a branch point at i agree through $i + d$, or have equal words once $\dashv$ is declared, so their gaps are at least d; the same holds for fan leaves. Let ι be inserted at p for the block of $\hat{\lambda}$. Its word agrees with the revealed letters through $p + d$, or equals the word if $\dashv$ had been declared. A leaf μ of the block, original or inserted at a position $\ge p$, agrees with the same revealed letters, and $\mathrm{lca}(\iota, \mu)$ is the node at p; so $E(\iota, \mu) \ge d$. For μ outside the block, $\mathrm{lca}(\iota, \mu) = \mathrm{lca}(\hat{\lambda}, \mu)$ is a node at some $i_\nu < p$, and the first difference j of $\tilde{u}_{\hat{\lambda}}$ and $\tilde{u}_\mu$ is at least $i_\nu + d + 1$, or never occurs. If $j \le p + d$, then $\tilde{u}_\iota$ and $\tilde{u}_\mu$ first differ at j. Otherwise they agree through $p + d \ge i_\nu + d + 1$. Either way $E(\iota, \mu) \ge d$.

Shape bound. For an original leaf λ let $\mathrm{Dev}(\lambda) = \{(p, e) : e \in \mathrm{Live}_{u_\lambda}(\beta_\lambda, p) \setminus \{\beta_\lambda(p)\}\}$. Lemma B.5 gives $|\mathrm{Dev}(\lambda)| \le K$. Let λ be an (L1) leaf or the fan leaf w_{e_1} of a fan, and let ν be a node on its path, at position p. Every sibling edge e of λ's path at ν gives an element (p, e) of $\mathrm{Dev}(\lambda)$: a safe sibling is live for every word of Cons at that play prefix by Lemma B.7(ii), and u_λ is such a word; an inserted sibling is live for every leaf of $\Lambda_0(\lambda, p)$, by the definition of a gap triple; a fan sibling is an edge of E_1, live for $w_1 = u_\lambda$. Distinct nodes sit at distinct positions, so these elements are distinct, and $\sum_\nu (\deg \nu - 1) \le |\mathrm{Dev}(\lambda)| \le K$ along the path. The path of a fan leaf w_e with $e \in E_1$ has the same nodes as the path of w_{e_1}. The nodes on the path of a leaf inserted at p are the nodes at positions $< p$ on the path of an original leaf of its block, plus the node at p, which lies on that path too; every block contains an (L1) leaf or a fan leaf w_{e_1}, and we take that one. So every root-to-leaf path satisfies $\sum(\deg - 1) \le K$. Every internal node has at least two children: a branch point has at least two safe edges, a fan at least two leaves, and an inserted node has the block and at least one inserted leaf. The shape lies in $\mathfrak{S}^\flat_K$. □

Corollary B.10. *Under the $K^\flat$-promise,* $\mathrm{GOOD}^{\flat,\infty}(d, H)$ *holds iff there is no d-coherent kill witness with shape in $\mathfrak{S}^\flat_K$.*

Proof. Both directions are Theorem 5.4. □

B.5. **Fixed resources: the finite games.** Let R_H^{od} be a DFA for the well-formed joint words with $\mathrm{od} \le H$, obtained by determinising and complementing the monitor $\mathcal{N}_H$ of Theorem 4.2. It reads pairs of edges of $A_\perp$ and B_z, then one letter $(\Phi_A(q_A), \Phi_B(q_B))$ with $q_A \in F_A$ and $q_B \in F_B$, as in Section 4.

Recall the arena $G_{d,H}^\flat$. Positions are $(q_A, q_B, r, S, \mathrm{buf}, \mathrm{eof})$ with $q_A \in Q_A \cup \{p_\perp\}$, $q_B \in Q_B \cup \{z\}$, r a state of R_H^{od}, $S \subseteq Q_B$, $\mathrm{buf} \in \Sigma^{\le d+1}$ and $\mathrm{eof} \in \{0,1\}$. The moves are those of Section 4; on a controller move with edge $e \in \Delta_{B_z}(q_B, a)$ on the head letter a, also $S := \delta_B(S, a)$. Initially $S = \{q_0^B\}$. A terminal is bad iff

$$S \cap F_B \neq \emptyset \ \wedge\ \big(q_B \notin F_B \ \vee\ R_H^{\mathrm{od}} \text{ rejects after } (\Phi_A(q_A), \Phi_B(q_B))\big).$$

At a terminal for the word w we have $S = \delta_B(\{q_0^B\}, w)$, so $S \cap F_B \neq \emptyset$ iff $w \in \mathrm{dom}(B)$. Then $w \in \mathrm{dom}(A)$, so $q_A \in F_A$ and $\Phi_A(q_A)$ is defined. $\Phi_B(q_B)$ is needed only when $q_B \in F_B$.

Proof of Theorem 4.4. Map a history of $\mathfrak{G}^\flat$ (Step 1 of the proof of Theorem 5.4(b)) to $(q_A, q_B, r, S, \mathrm{buf}, \mathrm{eof})$, with q_A the state of $A_\perp$ on the consumed letters, r the state of R_H^{od} on the consumed edge pairs, and $S = \delta_B(\{q_0^B\}, \text{consumed letters})$. This quotient preserves moves. It preserves badness of terminals: $S \cap F_B \neq \emptyset$ iff $w \in \mathrm{dom}(B)$; $q_B \notin F_B$ iff β is not accepting; and for accepting β on $w \in \mathrm{dom}(B)$ the joint word of $(\rho_A(w), \beta)$ has equal underlying words by (R4), so R_H^{od} rejects iff $\mathrm{od} > H$.

A finite-memory controller induces a strategy in $G_{d,H}^\flat$, and that strategy wins iff the controller achieves the objective. If the controller wins $G_{d,H}^\flat$, it has a positional winning strategy, since finite reachability games are positionally determined. The strategy depends on $(q_A, q_B, r, S, \mathrm{buf}, \mathrm{eof})$. The window encodes buf and eof, the state q_B is observed, and q_A, r, S are functions of the past windows and the controller's own edges. So a finite memory $(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R_H^{\mathrm{od}}) \times 2^{Q_B}$ tracks them. If the environment wins, it has a positional winning strategy as well. Any arbitrary-memory controller induces a strategy in $G_{d,H}^\flat$, against which the environment reaches a bad terminal, that is, a word on which the controller fails. So the three statements are equivalent. Each $\mathrm{GOOD}^\flat(d, H)$ is decidable, since the arena is finite and computable, and enumerating H, and then d, shows that $Q_d^\flat$ and $J^\flat$ are recursively enumerable. □

Recall the arena $G_{d,H}^{\mathrm{q}}$ of Proposition 4.5. Its positions are $(q_A, q_B, r, \mathrm{buf}, \mathrm{eof})$, with the moves of Section 4, except that at a controller move with head letter a and remaining buffer x only edges of $\mathrm{SafeQ}_d(q_B, a, x, \mathrm{eof})$ may be chosen, and the position is a terminal won by the environment if that set is empty. Other terminals are bad iff $q_B \in F_B$ and R_H^{od} rejects after $(\Phi_A(q_A), \Phi_B(q_B))$. If $q_B \in F_B$, the run is an accepting run of B, so $w \in \mathrm{dom}(B) \subseteq \mathrm{dom}(A)$ and $\Phi_A(q_A)$ is defined.

Proof of Proposition 4.5. (a) Let $\mathcal{C}$ be an arbitrary-memory controller achieving the objective. By Lemma B.8 it takes a safe edge at every position of every input. By Lemma B.9 and induction on i, the safe set at i along $\rho_{\mathcal{C}}(w)$ equals $\mathrm{SafeQ}_d(q_{i-1}, a_i, x_i, \mathrm{eof}_i(w))$, and $\mathrm{buf} = a_i x_i$ at that move. So $\mathcal{C}$ induces a strategy in $G^{\mathrm{q}}_{d,H}$ that never reaches a safe-empty position. At a terminal $(w, \rho_{\mathcal{C}}(w))$ with $q_B \in F_B$ the run is accepting, so $w \in \mathrm{dom}(B) \subseteq \mathrm{dom}(A)$, $q_A \in F_A$, $\mathrm{od} \leq H$, and R^{od}_H accepts. So $\mathcal{C}$ wins, and the controller has a positional winning strategy σ.

Conversely, let σ be positional and winning. As a controller it tracks q_A and r, which are functions of the past windows and its own edges. On an input w, by induction on i and Lemma B.9, every edge it takes lies in $\mathrm{SafeQ} = \mathrm{Safe}_d$. So by Lemma B.7(iv) its run on $w \in \mathrm{dom}(B)$ is accepting, $q_B \in F_B$ at the end, and since the terminal is not bad, R^{od}_H accepts and $\mathrm{od} \leq H$. The memory is $(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R^{\mathrm{od}}_H)$. The remaining equivalence is Theorem 4.4.

(b) Take empty outputs, so that $\mathrm{od} \equiv 0$ and no terminal of $G^{\mathrm{q}}_{d,0}$ is bad, and drop q_A and r: A has one state, and R^{od}_0 accepts every joint word that arises. The arena then has positions $(q_B, \mathrm{buf}, \mathrm{eof})$, and the argument of (a) shows that an arbitrary-memory controller achieving acceptance exists iff the controller wins this safety game. A positional winning strategy is a function of q_B and the window, which encodes buf and eof. That is a memoryless controller. A delegator in the sense of Definition 1 of [LR13] sees the current state, the current letter and up to k further letters, fewer only at the end of the word; this is win_k with strict $\dashv$. It picks a successor state rather than an edge, which makes no difference for acceptance with empty updates. □

Appendix C. Witness automata, the cutoff and certificates

We prove Lemmas 6.5, 6.6, 6.1 and 6.2, Proposition 6.7, Theorem 3.1 and Corollary 3.3, and then state and prove the pumping certificates (Theorem C.6) and the least parameters (Proposition C.12).

C.1. **Cycles and integer points.**

Proof of Lemma 6.5. ($\subseteq$) Let σ be a walk from ι to F with vertex positions $0, \ldots, L$. Mark position 0 and, for every visited vertex, the position of its last visit. There are at most $Z + 1$ marks, so at most Z segments between consecutive marks. Inside a segment, find the first position whose vertex already occurred in the segment, and delete the closed subwalk between the two occurrences. No vertex repeats before that position, so the deleted subwalk is a simple cycle, with vertices in $V(\sigma)$. Repeat until no segment has a repeated vertex. Each deletion keeps the first occurrence of its repeated vertex. A deletion removes a marked position only when the second occurrence is the segment's end, and then the same vertex survives at the kept first occurrence. So every vertex of σ still occurs at the end. Each segment is now

a simple path of length at most $Z-1$. Their concatenation π is a walk from ι to F with $V(\pi)=V(\sigma)$ and $|\pi|\le Z(Z-1)$, and $\psi(\sigma)=\psi(\pi)+\sum_c \psi(c)$ over the deleted cycles. Each of them has $V(c)\subseteq V(\pi)$.

($\supseteq$) A simple cycle c with $V(c)\subseteq V(\pi)$ shares a vertex with π. Rotate c to start there and splice in the required number of copies. The result is again a walk from ι to F. $\square$

If every edge weight has norm at most W, then $|\psi(\pi)|_\infty \le Z^2 W$, every element of $\mathrm{Cyc}(\pi)$ has norm at most ZW, and $\mathrm{Cyc}(\pi)$ has at most $(2ZW+1)^q$ elements. Several initial vertices give a finite union.

Proof of Lemma 6.6. Bounded case. The program $\max\{x_c : x\in P\}$ is feasible and bounded, so its optimum is attained at a basic feasible solution x. The support S of x indexes linearly independent columns of M, so $|S|\le \operatorname{rank} M\le m$. Choose $|S|$ rows on which the columns M_S are independent, giving a nonsingular square matrix N with right-hand side b'. Cramer's rule gives $x_c=\det N_c/\det N$, where N_c replaces the column of c by b'. Here $\det N$ is a nonzero integer and $|\det N_c|\le |S|!\,\beta^{|S|}\le m!\,\beta^m$.

Unbounded case. The polyhedron P is the sum of the convex hull of its vertices and the cone $\{r\ge 0 : Mr=0\}$. This cone is rational, so finitely many integer vectors generate it. If every generator had $r_c=0$, then x_c would be bounded on P by its maximum over the vertices. So some generator has $r_c\ge 1$. $\square$

Three further lemmas serve the certificates of Section C.4.

Lemma C.1 (small solution)**.** *Let $A\in\mathbb{Z}^{m\times N}$ and $b\in\mathbb{Z}^m$ have entries of absolute value at most $\tau\ge 1$. If $Ax=b$ has a solution in $\mathbb{N}^N$, it has one with every coordinate at most*

$$J=\Delta+(N+1)\,m\,(\tau+1)\,\Delta^2,\qquad \Delta=(2m+2)!\,(\tau+1)^{2m+2}.$$

Proof. Two facts about a column set $S\subseteq[N]$ come first.

Fact 1. If $A_S\alpha=0$ has a nonzero solution $\alpha\ge 0$ in $\mathbb{R}^S$, it has one in $\mathbb{N}^S$ with entries at most Δ. The polytope $\{\alpha\ge 0 : A_S\alpha=0,\ \mathbf{1}^\top\alpha=1\}$ is nonempty and bounded, so it has a vertex α. The support columns of $\binom{A_S}{\mathbf{1}^\top}$ at α are linearly independent, so there are at most $m+1$ of them. Choose that many rows forming a nonsingular square matrix and apply Cramer's rule with a right-hand side of 0's and one 1. Multiplying α by the absolute determinant gives an integer vector with entries at most $(m+1)!\,\tau^{m+1}\le\Delta$.

Fact 2. Otherwise there is $h\in\mathbb{Z}^m$ with $|h_i|\le\Delta$ and $h^\top A_j\ge 1$ for all $j\in S$. By Gordan's alternative some real h has $h^\top A_j>0$ for all $j\in S$; rescale to ≥ 1. Consider the program $\min\sum_i u_i$ subject to $h^\top A_j\ge 1$ for $j\in S$, $u-h\ge 0$ and $u+h\ge 0$. It is feasible and bounded below by 0. Its feasible region is pointed: a line direction $(\delta u,\delta h)$ would need $\delta u\ge|\delta h|$ and $-\delta u\ge|\delta h|$. So an optimum is attained at a vertex, which $2m$ independent active constraints determine, with coefficients of absolute value at most τ and right-hand sides in $\{0,1\}$. Cramer's rule and scaling by the absolute

determinant give an integer h with entries at most $(2m)!\,(\tau+1)^{2m} \le \Delta$, and the scaling keeps $h^\top A_j \ge 1$.

Main argument. Let x be a solution with minimum coordinate sum and $S = \{j : x_j \ge \Delta\}$. If $S = \emptyset$ we are done. If Fact 1 applies to S, its vector a satisfies $a \le x$ on S, and $x - a$ (extended by 0) is a solution with smaller sum, a contradiction. Otherwise take h from Fact 2. Then

$$\sum_{j\in S} x_j \le \sum_{j\in S} (h^\top A_j)\, x_j = h^\top b - \sum_{j\notin S} (h^\top A_j)\, x_j \le m\Delta\tau + N\, m\Delta\tau\, \Delta,$$

since $x_j < \Delta$ off S. So every coordinate is at most J. □

Lemma C.2 (support reduction)**.** *Let $A \in \mathbb{Z}^{n\times p}$, $n \ge 1$, have entries of absolute value at most $\tau \ge 1$, and let $b \in \mathbb{Z}^n$. If $Ax = b$ has a solution in $\mathbb{N}^p$, it has one whose support has size k with $2^k \le (2k\tau+1)^n$. In particular $k < 2n\log_2(8n\tau)$.*

Proof. Let $x \in \mathbb{N}^p$ solve the system with support S, $|S| = k$, and suppose $2^k > (2k\tau+1)^n$. For $U \subseteq S$ the sum $\sum_{j\in U} A_j$ lies in $[-k\tau, k\tau]^n$, a box with $(2k\tau+1)^n < 2^k$ points. So two distinct subsets have the same sum. Removing their intersection gives disjoint $U, U' \subseteq S$, not both empty, with $\sum_U A_j = \sum_{U'} A_j$; swap so that $U \ne \emptyset$. Let $t = \min_{j\in U} x_j \ge 1$, attained at j_0, and put $x' = x - t\mathbf{1}_U + t\mathbf{1}_{U'}$. Then $Ax' = b$, $x' \in \mathbb{N}^p$ and $\operatorname{supp}(x') \subseteq S \setminus \{j_0\}$. Iterate. The support shrinks strictly, so the process stops at a solution with $2^k \le (2k\tau+1)^n$.

For the explicit bound let $L = \log_2(8n\tau) \ge 3$. The map $k \mapsto k - n\log_2(2k\tau+1)$ is increasing for $k > n/\ln 2$, and $2nL \ge 6n$. At $k = 2nL$ we have $2k\tau + 1 \le 8n\tau L$, so $n\log_2(2k\tau+1) \le n(L + \log_2 L) < 2nL = k$ and $2^k > (2k\tau+1)^n$. So a terminal solution has $k < 2nL$. □

Lemma C.3 (small ray)**.** *Let $M \in \mathbb{Z}^{n\times p}$ and $b \in \mathbb{Z}^n$ have entries of absolute value at most $\beta \ge 1$. Let $P = \{x \in \mathbb{R}^p_{\ge 0} : Mx = b\}$ be nonempty, fix a coordinate c, and suppose x_c is unbounded above on P. Then there is $r \in \mathbb{N}^p$ with $Mr = 0$, $r_c \ge 1$, $|\operatorname{supp}(r)| \le n+1$ and every entry at most $(n+1)!\,\beta^{n+1}$.*

Proof. By the unbounded case of Lemma 6.6, some $r^0 \in \mathbb{Z}^p_{\ge 0}$ has $Mr^0 = 0$ and $r^0_c \ge 1$. So $Q = \{r \ge 0 : Mr = 0,\ r_c = 1\}$ contains r^0/r^0_c. It lies in $\mathbb{R}^p_{\ge 0}$, so it is pointed and has a vertex. A vertex is a basic feasible solution of the system with matrix $M' = \binom{M}{e_c^\top}$ and right-hand side $(0,\dots,0,1)$. Its support S indexes independent columns of M', so $|S| \le n+1$. Choose $|S|$ rows on which M'_S is nonsingular, giving N. By Cramer's rule $r_j = \det N_j / \det N$, where N_j replaces a column by a 0/1 vector, so $|\det N_j| \le |S|!\,\beta^{|S|}$. Then $r' := |\det N|\, r$ lies in $\mathbb{N}^p$, has $Mr' = 0$, $r'_c = |\det N| \ge 1$, support S and entries at most $(n+1)!\,\beta^{n+1}$. □

C.2. **The witness automata.**

Proof of Lemma 6.1. Fix the OD leaf λ and a clone $X \in \{A, B\}$. The colour track of its output lies in $\mathsf{P}^*\mathsf{MS}^*$, the language of $\mathcal{P}_{\mathrm{mk}}$, so exactly one

output position carries M. Let c^X be its rank. By Lemma A.1, every output position is exactly one live letter occurrence, in the update of its birth row. So the live P-occurrences are the $c^X - 1$ positions before the marked one, and the R_λ-weights sum to $(c^A - 1) - (c^B - 1) = c^A - c^B$. The one live M-occurrence sits in row $\beta^X := b_X(c^X)$. Hence the birth flag of clone X after row r is $[\beta^X \le r]$, and the cap $\eta + \#_{\mathsf{M}} \le 1$ of (T1) is never violated. The G_λ-weight counts the rows r with $[\beta^A \le r] \ne [\beta^B \le r]$. These are the rows in $[\min(\beta^A, \beta^B), \max(\beta^A, \beta^B))$, and there are $|\beta^A - \beta^B|$ of them. Every such row is at most n, so the leaf is RUN there. After DONE both flags are 1 and nothing is added. Dead occurrences are never counted, whatever colour they carry.

Conversely, given ranks $c^A, c^B \in [1, N_\lambda]$, colour each live occurrence P, M or S according to whether its rank is below, equal to or above c^X, and colour dead occurrences P. The coloured outputs lie in $L(\mathcal{P}_{\mathrm{mk}})$, Lemma A.2 supplies endpoint maps with the sink for dead registers, and the flags are then determined. The count above gives the stated weights. □

Proof of Lemma 6.2. Fix an accepting walk $S_0 \xrightarrow{1} S_1 \xrightarrow{2} \cdots \xrightarrow{R} S_R$. For a leaf λ let n_λ be the number of rows before the first $\perp$ in its track, so $|u_\lambda| = n_\lambda$, and write $t_\lambda = \min(r, n_\lambda)$ at row r. For an internal node ν let i_ν be a row with $\nu \in \mathrm{Mark}$. For a pair $\pi = \{\lambda, \lambda'\}$ with $\nu = \mathrm{lca}(\pi)$ let j_π be the first row $r > i_\nu$ at which the padded letters of λ and λ' differ, and $j_\pi = \infty$ if there is none. We claim, for every $r \le R$:

- (I1) λ is RUN in S_r iff $r \le n_\lambda$. A leaf leaves RUN only through (T2), at row $n_\lambda + 1$, and DONE is absorbing by (T1). Acceptance makes every leaf DONE, so $n_\lambda + 1 \le R$.
- (I2) q^B_λ in S_r is the state of β_λ after t_λ steps, and at an OD leaf q^A_λ is the state of $\rho_A(u_\lambda)$ after t_λ steps. This is (T1): edges are taken from $\Delta_{B_z}(q^B_\lambda, a_\lambda)$ and $\Delta_{A_\perp}(q^A_\lambda, a_\lambda)$.
- (I3) C_λ in S_r equals $C_r(u_\lambda)$ for $r \le n_\lambda$. (T2) pins $C_\lambda = F_B = C_{n_\lambda}(u_\lambda)$ in S_{n_λ}, (T1) imposes $C_\lambda = \mathrm{pre}_a(C'_\lambda)$ between consecutive rows, and these determine the sequence (Lemma B.1).
- (I4) At an OD leaf, the use bits in S_r are the true use bits of each clone at time t_λ, relative to its final expression. (T2) pins them in S_{n_λ} to "occurs in Φ", and (T1) imposes the backward law between consecutive rows, which has a unique solution from that boundary (Lemma A.1). This also fixes the use bits in S_0.
- (I5) At an OD leaf, the endpoint maps in $S_0, \ldots, S_{t_\lambda}$ satisfy the conditions of Lemma A.2 relative to $\mathcal{P}_{\mathrm{mk}}$ and the chosen colours, by (T1); row $n_\lambda + 1$ checks the final condition by (T2).
- (I6) ν is PRE in S_r iff $r < i_\nu$. POST is absorbing, (T4) needs PRE before the row, and acceptance needs POST; so i_ν exists and is unique. (T4) needs the parent POST before the row, so $i_{\mathrm{parent}} < i_\nu$.
- (I7) π is WAIT in S_r iff $r < i_\nu$. For $\pi \in \varphi$: it is COUNT with accumulated $E_\pi = r - i_\nu$ iff $i_\nu \le r < j_\pi$, and CLOSED with $E_\pi = j_\pi - i_\nu - 1$

iff $r \geq j_\pi$. The move to COUNT has weight 0, each later row with equal letters adds 1, and the first unequal row closes with weight 0. For $\pi \notin \varphi$: it is SAME iff $r \geq i_\nu$, and every row after i_ν has equal padded letters.

Each clause follows by induction on r from the conditions it names.

Now the conclusions. (T3) at rows $r < i_\nu$ and (T4) at row i_ν give one letter other than $\perp$ for all leaves below ν, and one B-edge at rows $< i_\nu$. With (I2), the leaves below ν share a state q_ν before row i_ν. This is (K1), and the injective child map of (T4) is (K2). Marks happen in ancestor order by (I6), and at rows where every leaf below is RUN, because the letter is not $\perp$. For (K3): by (I3), the initial guard $q_0^B \in C_\lambda$ says exactly $u_\lambda \in \mathrm{dom}(B)$, and the guard $\mathrm{dst}(e) \notin C'_\mu$ at row p says exactly that e is dead for u_μ at p (Lemma B.1). By (I6), the lca of λ and μ is PRE before row p iff its fork position is $\geq p$, so $\mathrm{Bl}(\lambda)$ at row p is the block $\Lambda(\lambda, p)$ of Definition 5.3, and all its leaves are in state q_λ^B by (T3) and (T4). So (T5) with (T4) is (K3). (T2) gives the DEAD half of (K4) and accepting runs at OD leaves. For the pairs: acceptance puts every $\pi \in \varphi$ in CLOSED, so $j_\pi < \infty$ and $E_\pi = j_\pi - i_\nu - 1 = E(\pi)$. For $\pi \notin \varphi$, rows $\leq i_\nu$ have equal letters by (T3) and (T4), rows $> i_\nu$ by SAME, and both tracks read $\perp$ beyond row R. So $u_\lambda = u_{\lambda'}$. At an OD leaf, β_λ is accepting on $u_\lambda \in \mathrm{dom}(B)$, so by (R4) the outputs of $\rho_A(u_\lambda)$ and β_λ have one underlying word, and Lemma 6.1 gives the coordinates (R_λ, G_λ).

Conversely, take data satisfying (K1)–(K3) and the DEAD half of (K4), with all words in $\mathrm{dom}(B)$, accepting runs at OD leaves, exactly the pairs of φ with different words, and ranks at the OD leaves. Use rows $r = 1, \ldots, R := \max_\lambda n_\lambda + 1$. Track λ reads $u_\lambda[r]$ for $r \leq n_\lambda$ and $\perp$ afterwards, Mark $= \{\nu : i_\nu = r\}$ at row r, and the phases are those of (I6) and (I7). Set C_λ at row r to the true set $C_r(u_\lambda)$. It satisfies the initial guard, (T1) and (T2) by Lemma B.1, and (K3) gives (T5). A DEAD leaf's run is not accepting, so it ends outside F_B, as (T2) requires. At OD leaves, Lemma 6.1 supplies colours, flags, use bits and endpoint maps. (T3), (T4) and (T6) hold by (K1), (K2) and the choice of φ. The last row is the first in which every leaf is DONE, and every pair in φ is CLOSED by then, since its first difference lies at a row $\leq R$. □

C.3. **The cutoff and the proof of Theorem 3.1.**

Proof of Proposition 6.7. Three facts carry the argument. $\mathrm{BAD}^\flat(d, H)$ is downward closed in (d, H), because its guards are lower bounds. Each $V^\flat_{\mathcal{T},\varphi,\tau}$ is a finite union of *pieces* $a + \mathbb{N} \cdot V$ with $|a|_\infty \leq Z^2 W$ and $|v|_\infty \leq ZW$ for $v \in V$, by Lemma 6.5 and the bounds after it; the lemma applies since accepting states have no outgoing rows, so a spliced cycle never passes through acceptance. The third fact is Lemma 6.6.

Fix a case with an accepting walk and a piece $a + \mathbb{N} \cdot V$ of its $V^\flat$. Take real variables $\lambda \in \mathbb{R}^V_{\geq 0}$, $T \geq 0$ and slacks $\sigma \geq 0$, and the rows

$$\Big(a + \sum_{v \in V} \lambda_v v\Big)_{R_\lambda} = 0, \qquad (\,\cdot\,)_{G_\lambda} - T - \sigma_\lambda = 1, \qquad (\,\cdot\,)_{E_\pi} - T - \sigma_\pi = 0,$$

over OD leaves λ and pairs $\pi \in \varphi$. This is a system $\{x \geq 0 : Mx = b\}$ with at most m rows and entries of absolute value at most $\beta := Z^2W+1$. Its integer solutions with value T are the vectors of the piece that meet the guards of $\mathrm{BAD}^\flat(T, T)$. Suppose it has an integer solution and T is unbounded on its real solution set. Lemma 6.6 gives an integer ray r with $Mr = 0$ and $r_T \geq 1$. Adding multiples of r keeps λ integral, so the vector stays in the piece, and $\mathrm{BAD}^\flat(T, T)$ holds for arbitrarily large T, hence for every T by downward closure. Theorem 6.3 and Lemma 2.6 give $\neg J^\flat$. Otherwise every integer solution has $T \leq m!\,\beta^m$. A piece with no integer solution contributes nothing. A case with no OD leaf and $\varphi = \emptyset$ has no row with T, but by Lemma 6.4 it has no accepting walk and so no piece.

So on a positive instance $\mathrm{BAD}^\flat(T^\flat_*, T^\flat_*)$ fails, and Theorem 6.3 gives $\mathrm{GOOD}^\flat(T^\flat_*, T^\flat_*)$. The converse is trivial.

For $Q^\flat_d$ keep d as a constant. The rows become $(\cdot)_{G_\lambda} - H - \sigma_\lambda = 1$ and $(\cdot)_{E_\pi} - \sigma_\pi = d$, with entries at most $Z^2W + d + 1$, and H is the variable. If H is unbounded on the real solutions of a piece with an integer solution, the ray gives $\mathrm{BAD}^\flat(d, H)$ for all H, so $\neg Q^\flat_d$. Otherwise every integer solution has $H \leq m!\,(Z^2W + d + 1)^m$. The arbitrary-memory form is the same, through the first equivalence of Theorem 6.3. □

Proof of Theorem 3.1. Finite ambiguity gives $K \leq 2^{O(s)}$ [WS91, KMRW26], and Lemma 8.21 tests each candidate, so K is computable. From the construction of the automata $\mathcal{W}^\flat_{\mathcal{T},\varphi,\tau}$ over the cases with $\mathcal{T} \in \mathfrak{S}^\flat_K$ we compute Z, W and m, and from them $T^\flat_*$ (respectively $H^\flat_*(d)$). Build the finite game $G^\flat_{T^\flat_*, T^\flat_*}$ (respectively $G^\flat_{d, H^\flat_*(d)}$) of Theorem 4.4 and solve it by the attractor computation for finite safety games. Proposition 6.7 with Theorem 4.4 makes the answer $J^\flat$ (respectively $Q^\flat_d$). On a positive instance a positional winning strategy gives the controller (Theorem 4.4). Proposition 4.5 gives one with memory $(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R^{\mathrm{od}}_{T^\flat_*})$; it is computed by solving $G^{\mathrm{q}}_{T^\flat_*, T^\flat_*}$, whose safe sets are inclusions of regular languages. The answers agree for arbitrary memory by Theorem 4.4.

A second route avoids the game. By Lemma 6.5, $\mathrm{BAD}^\flat(d, H)$ is an existential Presburger formula in (d, H): a finite disjunction over cases and pieces of linear constraints on the cycle multiplicities. So $\exists d\, \exists H\, \neg\mathrm{BAD}^\flat(d, H)$ is a sentence of Presburger arithmetic, which is decidable. □

Proof of Corollary 3.3. Under **G** every reachable state of B accepts every remaining word, so $C_r(u)$ contains every reachable state and no edge is ever

dead. The kill condition (K3) then says that at every fork each edge is taken by some leaf of the block, and the cases with all leaves typed OD and $\varphi = \emptyset$ are the complete-fork automata of Remark 5.5. Let $\mathrm{OFF}(H)$ say that for every w some run β of B on w has $\mathrm{od}(\rho_A(w), \beta) \le H$, and define

$$\mathrm{BAD}_{\mathrm{off}}(H) : \Longleftrightarrow\ \exists \mathcal{T}\ \exists v \in V^{\flat}_{\mathcal{T},\emptyset,\mathrm{OD}} :\ \forall \lambda\ \big(R_\lambda = 0 \wedge G_\lambda \ge H + 1\big),$$

with $\mathcal{T}$ ranging over the admissible shapes. We show $\neg\mathrm{OFF}(H) \iff \mathrm{BAD}_{\mathrm{off}}(H)$.

($\Leftarrow$) By Lemma 6.2, a guarded vector comes from a witness in which all leaves carry one word w, and every leaf has a rank c with $|b_A(c) - b_{B,\lambda}(c)| \ge H + 1$. So every leaf run has od $> H$. The induction in the proof of Lemma 6.4 uses only (K3) and the common word, so it applies here: every run of B on w equals some β_λ. Hence no run on w has od $\le H$, and $\mathrm{OFF}(H)$ fails.

($\Rightarrow$) Let w be a word on which every run of B has od $> H$. Build the tree of all runs of B on w and contract the positions where the current state has one edge on the current letter. By the K-promise there are at most K runs. Each branch point (i, q) has one child per edge of $\Delta_B(q, w_i)$, and each child extends to a run since B is input-enabled. So (K2) holds, (K1) holds because all words are equal, and (K3) holds because every edge is taken by a child. Along every root-to-leaf path $\sum(\deg - 1) \le K - 1$ by Lemma A.4, so the shape is admissible. At each leaf choose a rank c with $|b_A(c) - b_{B,\lambda}(c)| \ge H + 1$; it exists because od $> H$ and the output is nonempty. Lemma 6.2 gives a vector meeting the guards.

Cutoff. $\mathrm{BAD}_{\mathrm{off}}(H)$ is downward closed in H. Run the argument of Proposition 6.7 with only the rows $(\cdot)_{R_\lambda} = 0$ and $(\cdot)_{G_\lambda} - H - \sigma_\lambda = 1$, with H as the variable and entries at most $Z^2W + 1$. If some piece has an integer solution and H is unbounded on its real solutions, Lemma 6.6 gives an integer ray, $\mathrm{BAD}_{\mathrm{off}}(H)$ holds for every H, and $\exists H\, \mathrm{OFF}(H)$ fails. Otherwise every integer solution has $H \le m!\,(Z^2W + 1)^m$, so $\mathrm{OFF}(H_{\mathrm{off}})$ holds with $H_{\mathrm{off}} = m!\,(Z^2W + 1)^m + 1$. Here Z, W, m are the bounds of Proposition 6.7 for the instance.

Decision. By Lemma 6.5, $\mathrm{BAD}_{\mathrm{off}}(H)$ is an existential Presburger formula in H, so $\exists H\, \neg\mathrm{BAD}_{\mathrm{off}}(H)$ is decidable. No coherence guard occurs: offline, the choice sees the whole word. On the instances of Theorem 3.4, which satisfy no K-promise, $\mathrm{OFF}(0)$ holds by offline zero. □

C.4. **Pumping certificates.** Throughout this part, (A, B) is a $\mathbf{G}^{\flat}$-instance, and A stands for $A_\perp$, B for B_z. A *case* is $(\mathcal{T}, \varphi, \tau)$ as in Section 6.1, and $\mathcal{W} = \mathcal{W}^{\flat}_{\mathcal{T},\varphi,\tau}$ is read as a finite multigraph whose edges are the allowed triples (state, row, state), with weights ψ in $\mathbb{Z}^{m_{\mathrm{case}}}$. The coordinates are R_λ, G_λ for OD leaves and E_π for $\pi \in \varphi$. Per row, G_λ and E_π change by 0 or 1, and R_λ by at most μ in absolute value. Accepting states have no outgoing rows. Z, W and m are as in Proposition 6.7; under the $K^{\flat}$-promise, Proposition 8.18(1) gives $\log_2 Z \le 6Ns + 2N^2$,

$W \le \max(\mu, 1) \le s$ and $m \le 2N + \binom{N}{2}$, with $N = 2^K$. The automaton and the proof of Lemma 6.2 make sense for any shape with every internal node of degree ≥ 2; neither uses $\mathcal{T} \in \mathfrak{S}_K^\flat$ or the $K^\flat$-promise. Below, a case may have any shape unless $\mathfrak{S}_K^\flat$ is stated. Write $\beta := Z^2W + 1$, $T_*^\flat = m!\,\beta^m + 1$ and $H_*^\flat(d) = m!\,(Z^2W + d + 1)^m + 1$.

Definition C.4 (pumping certificate)**.** A *pumping certificate* is a tuple

$$\mathfrak{c} = \big(\mathcal{T}, \varphi, \tau;\ \varsigma;\ (c_j, \ell_j, a_j, r_j)_{j=1}^{k}\big)$$

such that:

- (P0) $(\mathcal{T}, \varphi, \tau)$ is a case;
- (P1) the *stem* $\varsigma = (s_0, \theta_1, s_1, \dots, \theta_L, s_L)$ is a walk of $\mathcal{W}$, listed with its states s_i and rows θ_i, from an initial state to an accepting state;
- (P2) each c_j is a closed walk of $\mathcal{W}$ of length ≥ 1 from s_{ℓ_j} to s_{ℓ_j}, with $0 \le \ell_j \le L$, and $a_j, r_j \in \mathbb{N}$;
- (P3) *base balance:* the base vector $\mathbf{v}_0 := \psi(\varsigma) + \sum_j a_j \psi(c_j)$ has $(\mathbf{v}_0)_{R_\lambda} = 0$ for every OD leaf λ;
- (P4) *ray:* the growth vector $\mathbf{g} := \sum_j r_j \psi(c_j)$ has $\mathbf{g}_{R_\lambda} = 0$ and $\mathbf{g}_{G_\lambda} \ge 1$ for every OD leaf λ, and $\mathbf{g}_{E_\pi} \ge 1$ for every $\pi \in \varphi$.

States and rows are encoded as in Proposition 8.18(2a): a state in $\lceil \log_2 Z \rceil$ bits, a row in $O(N(\log s + \mu)) + |I(\mathcal{T})|$ bits. The integers are in binary, and each c_j is written next to position ℓ_j of ς. In this layout the certificate can be checked as a stream.

A *$\neg Q_d$-certificate* replaces (P4) by $\mathbf{g}_{R_\lambda} = 0$ and $\mathbf{g}_{G_\lambda} \ge 1$ for every OD leaf, and adds to (P3) that $(\mathbf{v}_0)_{E_\pi} \ge d$ for every $\pi \in \varphi$. A *$\neg D_H$-certificate*, for "no lookahead achieves bound H", replaces (P4) by $\mathbf{g}_{R_\lambda} = 0$ for every OD leaf and $\mathbf{g}_{E_\pi} \ge 1$ for every $\pi \in \varphi$, and adds to (P3) that $(\mathbf{v}_0)_{G_\lambda} \ge H + 1$ for every OD leaf.

Lemma C.5 (pumped walks)**.** *For every $t \in \mathbb{N}$, inserting $a_j + t r_j$ copies of c_j at position ℓ_j of ς, for every j, gives an accepting walk ς_t of $\mathcal{W}$ with $\psi(\varsigma_t) = \mathbf{v}_0 + t\mathbf{g}$ and $L + \sum_j (a_j + t r_j)|c_j|$ rows.*

Proof. A closed walk has an outgoing row at each of its states, and an accepting state has none, so no c_j passes through an accepting state. Inserting a closed walk at a state the stem visits keeps the endpoints; several insertions at one position are concatenated. The admissibility conditions (T1)–(T7) constrain single transitions, and the initial and accepting conditions concern only the first and last state. So ς_t runs from an initial to an accepting state and visits an accepting state only at its end, which makes it an accepting walk. Weights add per row. □

Theorem C.6 (pumping certificates for $\neg J^\flat$)**.** *Let (A, B) be a $\mathbf{G}^\flat$-instance.*

(1) (Soundness; any ambiguity.) If a pumping certificate exists, then $\neg J^{\flat,\infty}$, hence $\neg J^\flat$. Put $L_0 := L + \sum_j a_j|c_j|$ and $L_1 := \sum_j r_j|c_j|$. For all d, H there is a d-coherent kill witness of shape $\mathcal{T}$, with

at most $|\Lambda(\mathcal{T})|$ *leaf words in* $\mathrm{dom}(B)$*, each of length less than* $L_0 + L_1 \max(d, H+1)$*, such that every lookahead-*d *controller of any memory, on one of these words, either produces a non-accepting run or has* $\mathrm{od} > H$*.*

(2) (Completeness, with size bound; $K^\flat$*-promise.) Assume the* $K^\flat$*-promise and* $\neg J^\flat$*. Then a pumping certificate exists with* $\mathcal{T} \in \mathfrak{S}_K^\flat$*, every* c_j *a simple cycle (so* $|c_j| \le Z$*),* $L \le Z(Z-1)$*, and the indices split as* $\{1, \dots, k\} = I_0 \sqcup I_1$ *with* $r_j = 0$ *for* $j \in I_0$ *and* $a_j = 0$ *for* $j \in I_1$*, where*

- $|I_0| \le k_0 := \lfloor 2N \log_2(8NZW) \rfloor$ *and* $a_j \le J_0$ *for* $j \in I_0$*, with* J_0 *the bound* J *of Lemma C.1 for* N *rows,* k_0 *columns and* $\tau = Z^2 W$*;*
- $|I_1| \le m+1$ *and* $r_j \le \bar{r} := (m+1)!\,\beta^{m+1}$ *for* $j \in I_1$*.*

With $N = 2^K$ *and Proposition 8.18(1):* $\log_2 k_0 = O(\log(Ns + N^2))$*,* $\log_2 J_0 = O(N^2 s + N^3)$ *and* $\log_2 \bar{r} = O(N^3 s + N^4)$*, the order of* $\log_2 T_*^\flat$*. The encoded certificate has* $2^{O(Ns+N^2)}$ *bits, so* $2^{O(s)}$ *bits at fixed* K*. In part 1,* $\log_2 L_0 = O(N^2 s + N^3)$ *and* $\log_2 L_1 = O(N^3 s + N^4)$*.*

(3) (Characterisation.) Under the $K^\flat$*-promise:* $\neg J^\flat \iff \neg J^{\flat,\infty} \iff$ *a pumping certificate exists* $\iff$ *one exists within the bounds of part 2.*

(4) (Verification.) Given the instance and a string, deciding whether the string encodes a pumping certificate takes deterministic time polynomial in s *and the length of the string. A one-pass verifier of the streamed layout, on a string of length* ℓ *whose binary multiplicities have at most* B *bits, uses space*

$$O\big(\log Z + m\,(B + \log(\ell W + 2)) + N(\log s + \mu) + N^2\big),$$

the last two terms buffering one row and the case description. Within the bounds of part 2 this is $O(N^5 s + N^6)$*.*

Proof of 1. Fix d, H and let $t = \max(d, H+1)$. By Lemma C.5, ς_t is an accepting walk with weight $\mathbf{v}_t = \mathbf{v}_0 + t\mathbf{g}$. For every OD leaf, $(\mathbf{v}_t)_{R_\lambda} = 0$, and $(\mathbf{v}_t)_{G_\lambda} \ge t\,\mathbf{g}_{G_\lambda} \ge H+1$, because every per-row G-weight is ≥ 0 and so $(\mathbf{v}_0)_{G_\lambda} \ge 0$. Likewise $(\mathbf{v}_t)_{E_\pi} \ge t \ge d$ for every $\pi \in \varphi$. So $\mathbf{v}_t$ meets the guards of $\mathrm{BAD}^\flat(d, H)$ for this case.

Now the forward half of Theorem 6.3, with the shape arbitrary. Lemma 6.2 turns the walk into data satisfying (K1)–(K3) and the DEAD half of (K4), with all words in $\mathrm{dom}(B)$, accepting runs at OD leaves, $E(\pi) = E_\pi \ge d$ for $\pi \in \varphi$, and equal words otherwise. So the witness is d-coherent. At an OD leaf, $R_\lambda = 0$ gives $c_\lambda^A = c_\lambda^B$, one rank of one underlying word by (R4), and $G_\lambda \ge H+1$ gives a birth gap $> H$ at that rank, so $\mathrm{od} > H$. This is a d-coherent kill witness. Theorem 5.4(a) holds for any shape and uses no ambiguity bound; it shows that no lookahead-d controller of any memory achieves acceptance on $\mathrm{dom}(B)$ together with

od $\le H$, and its proof exhibits the failing word as a leaf word. By clause (I1) in the proof of Lemma 6.2, each leaf word is shorter than the number of rows, which is $L_0 + tL_1$. Since (d, H) was arbitrary, $\neg\mathrm{GOOD}^{\flat,\infty}(d, H)$ for all (d, H), so $\neg J^{\flat,\infty}$. Finite-memory controllers are controllers, so $\neg J^{\flat}$ follows. □

Proof of 2. Assume the $K^{\flat}$-promise and $\neg J^{\flat}$. Then $\neg\mathrm{GOOD}^{\flat}(T_*^{\flat}, T_*^{\flat})$, so $\mathrm{BAD}^{\flat}(T_*^{\flat}, T_*^{\flat})$ by Theorem 6.3. So there are a case $(\mathcal{T}, \varphi, \tau)$ with $\mathcal{T} \in \mathfrak{S}_K^{\flat}$ and $v \in V_{\mathcal{T},\varphi,\tau}^{\flat}$ with $v_{R_\lambda} = 0$ and $v_{G_\lambda} \ge T_*^{\flat} + 1$ for OD leaves, and $v_{E_\pi} \ge T_*^{\flat}$ for $\pi \in \varphi$. By Lemma 6.4 the case has an OD leaf or $\varphi \neq \emptyset$.

Stem. By Lemma 6.5, with several initial vertices handled by a finite union, $v = \psi(\varsigma) + \sum_{u \in V} \lambda_u u$, where ς is an accepting walk with $|\varsigma| \le Z(Z-1)$, $V = \mathrm{Cyc}(\varsigma)$ and $\lambda \in \mathbb{N}^V$. For every $u \in V$ fix one simple cycle c_u with $\psi(c_u) = u$ and $V(c_u) \subseteq V(\varsigma)$, rotated to start at a state of ς and attached there. Then $|c_u| \le Z$, $|u|_\infty \le ZW$ and $|\psi(\varsigma)|_\infty \le Z^2W$.

Ray. This is the system of Proposition 6.7, with variables $\lambda \in \mathbb{R}_{\ge 0}^V$, $T \ge 0$ and slacks $\sigma \ge 0$, and rows

$$\Big(\sum_u \lambda_u u\Big)_{R_\lambda} = -\psi(\varsigma)_{R_\lambda},$$

$$(\,\cdot\,)_{G_\lambda} - T - \sigma_\lambda = 1 - \psi(\varsigma)_{G_\lambda}, \quad (\,\cdot\,)_{E_\pi} - T - \sigma_\pi = -\psi(\varsigma)_{E_\pi}.$$

These are at most m rows with entries of absolute value at most β. The point $(\lambda,\ T = T_*^{\flat},\ \sigma_\lambda = v_{G_\lambda} - T_*^{\flat} - 1,\ \sigma_\pi = v_{E_\pi} - T_*^{\flat})$ is a nonnegative solution with $T = m!\,\beta^m + 1$. By the bounded case of Lemma 6.6, T is therefore unbounded above on the real solution set. Lemma C.3 gives an integer $r \ge 0$ with $Mr = 0$, $r_T \ge 1$, support at most $m+1$ and entries at most $\bar{r}$. Put $\mathbf{g} = \sum_u r_{\lambda_u} u$. Then $Mr = 0$ reads $\mathbf{g}_{R_\lambda} = 0$, $\mathbf{g}_{G_\lambda} = r_T + r_{\sigma_\lambda} \ge 1$ and $\mathbf{g}_{E_\pi} = r_T + r_{\sigma_\pi} \ge 1$. Let I_1 index the cycles c_u with $r_{\lambda_u} > 0$; there are at most $m+1$.

Base. If there is no OD leaf, (P3) is vacuous and $I_0 = \emptyset$. Otherwise let $n \le N$ be the number of OD leaves and consider $A\lambda = b$ over $\lambda \in \mathbb{N}^V$, with A the R-projections of the vectors u (entries at most ZW) and $b = -\psi(\varsigma)_R$ (entries at most Z^2W). The λ from the stem solves it. Lemma C.2 with $\tau = ZW$ gives a solution with support S, $|S| < 2n\log_2(8nZW) \le k_0$. Lemma C.1 on the columns S with $\tau = Z^2W$ gives one with entries at most J_0. Let I_0 index the cycles c_u for $u \in S$, with these multiplicities a_u. Then $(\mathbf{v}_0)_R = 0$.

A cycle used in both base and ray is listed twice, once in I_0 and once in I_1. So (P0)–(P4) hold with the stated split.

Sizes. Lemma C.1 with $n \le N$ rows gives $\log_2 \Delta = O(N\log N + N\log(Z^2W)) = O(N^2 s + N^3)$ and $\log_2 J_0 = O(\log\Delta + \log k_0) = O(N^2 s + N^3)$. Next, $\log_2 \bar{r} = O(m\log m + m\log\beta) = O(N^3 s + N^4)$, using $m = O(N^2)$ and $\log\beta = O(Ns + N^2)$. The certificate has at most $Z(Z-1) + (k_0 + m + 1)Z$ rows and states of $O(Ns + N^2)$ bits each, plus $k_0 + m + 1$ positions and integers of $O(N^3 s + N^4)$ bits. The

total is $2^{O(Ns+N^2)}$ bits, dominated by Z^2. Finally $L_0 \le Z^2 + k_0 J_0 Z$ and $L_1 \le (m+1)\bar{r}Z$. □

Proof of 3. Combine parts 1 and 2. □

Proof of 4. A row test is the check of Proposition 8.18(2a), polynomial in s and N with $O(\log s)$-bit pointers. The other checks are syntactic, plus additions and comparisons of vectors. For the streamed form, keep the current state, the last stem state (to check that each inserted c_j starts and ends there) and the running sums of $\psi(\varsigma)$, $\sum_j a_j \psi(c_j)$ and $\sum_j r_j \psi(c_j)$, with m coordinates each. A cycle's weight is summed row by row, each partial sum of absolute value at most $|c_j|W \le \ell W$, and is multiplied by a_j and r_j only once the cycle is complete. So every intermediate value has absolute value at most $\ell W \cdot 2^B \cdot \ell$, which needs $O(B + \log(\ell W + 2))$ bits. The term $\log(\ell W + 2)$ cannot be dropped: the final vectors may be small while the running sums along ς are large, because R-weights are signed and cancel. Within the bounds of part 2, $\ell \le 2^{O(Ns+N^2)}$ and $B = O(N^3 s + N^4)$, so each coordinate has $O(N^3 s + N^4)$ bits and the total is $O(m(N^3 s + N^4)) = O(N^5 s + N^6)$. □

Guessing a certificate in the streamed form and verifying it puts $\neg J^\flat$ in NSPACE($O(N^5 s + N^6)$), the bound Proposition 8.18(2b) gives for $J^\flat$; by the Immerman–Szelepcsényi theorem the two agree. The certificate describes the obstruction. It gives no faster decision procedure.

Proposition C.7 (dual cutoff: lookahead for a fixed bound)**.** *Under the $K^\flat$-promise, for every H,*

$$\exists d\ \mathrm{GOOD}^\flat(d, H) \iff \mathrm{GOOD}^\flat(D^\flat_*(H), H),$$
$$D^\flat_*(H) := m!\,(Z^2 W + H + 1)^m + 1,$$

and likewise with arbitrary memory.

Proof. ($\Leftarrow$) is trivial. ($\Rightarrow$) Suppose $\neg\mathrm{GOOD}^\flat(D^\flat_*(H), H)$. By Theorem 6.3 a case and a guarded v exist for $(D^\flat_*(H), H)$. Take its piece (Lemma 6.5) with H constant and d as the variable. The rows are $(\cdot)_{R_\lambda} = -\psi_R$, $(\cdot)_{G_\lambda} - \sigma_\lambda = H + 1 - \psi_G$ and $(\cdot)_{E_\pi} - d - \sigma_\pi = -\psi_E$: at most m rows with entries at most $Z^2W + H + 1$. If $\varphi = \emptyset$ the d-column is zero and d is trivially unbounded. The point has $d = D^\flat_*(H)$, above the bound $m!\,(Z^2W + H + 1)^m$ of Lemma 6.6, so d is unbounded above. Lemma 6.6 gives an integer ray r with $Mr = 0$ and $r_d \ge 1$. Adding t copies of it keeps $R = 0$, keeps $G \ge H + 1$ because the slacks balance the G-rows, and gives $E \ge D^\flat_*(H) + t r_d$. So $\mathrm{BAD}^\flat(d', H)$ holds for arbitrarily large d', hence for all d' by downward closure, and Theorem 6.3 gives $\neg\mathrm{GOOD}^\flat(d, H)$ for all d. The arbitrary-memory form goes through the same theorem. □

Corollary C.8 (certificates for $\neg Q^\flat_d$ and for a fixed bound)**.** *Let (A, B) be a $\mathbf{G}^\flat$-instance.*

(a) *A* $\neg Q_d$*-certificate implies* $\neg Q_d^{\flat,\infty}$. *A* $\neg D_H$*-certificate implies* $\neg\exists d\,\mathrm{GOOD}^{\flat,\infty}(d,H)$. *Neither needs an ambiguity bound.*

(b) *Under the* $K^\flat$*-promise the converses hold, with certificates within the bounds of Theorem C.6(2) up to two changes. The ray coefficients are bounded by* $(m+1)!\,(Z^2W+d+1)^{m+1}$ *(respectively* $(m+1)!\,(Z^2W+H+1)^{m+1}$*). The base is constrained in at most* m *rows, so* k_0 *becomes* $\lfloor 2m\log_2(8m\tau')\rfloor$ *and* J_0 *is computed with* m *rows and* $\tau' = Z^2W+d$ *(respectively* Z^2W+H+1*). With* $u=d$ *(respectively* $u=H$*) given in binary:* $\log_2\tau' = O(Ns+N^2+\log(u+2))$*, there are* $O(N^2(Ns+N^2+\log(u+2)))$ *cycles, every multiplicity has* $O(N^3s+N^4+N^2\log(u+2))$ *bits, and the encoded certificate has* $2^{O(Ns+N^2)}(1+\log(u+2))^2$ *bits, which is* $2^{O(s)}\mathrm{poly}(\log(u+2))$ *at fixed* K.

Proof. (a) As in Theorem C.6(1), with t arbitrary and d fixed (respectively $t=d$ and H fixed). The base supplies $E\geq d$ (respectively $G\geq H+1$) and the ray supplies growth in the other coordinate. All per-row E- and G-weights are ≥ 0, so the ray never lowers them.

(b) As in Theorem C.6(2), starting from $\mathrm{BAD}^\flat(d,H_*^\flat(d))$ (Proposition 6.7) or from $\mathrm{BAD}^\flat(D_*^\flat(H),H)$ (Proposition C.7), with H (respectively d) as the unbounded variable. The base system consists of the R-rows and the E-rows $(\cdot)_{E_\pi} - s_\pi = d - \psi_E$ (respectively the G-rows $(\cdot)_{G_\lambda} - s_\lambda = H+1-\psi_G$), with slack columns s. Apply Lemma C.2 and then Lemma C.1 to it. Slack columns are unit vectors and absorb surplus; they are no cycles. Counting the stem ($\leq Z^2$ rows), the cycles ($\leq Z$ rows each, rows of $O(Ns+N^2)$ bits) and the integers gives the size. □

Remark C.9 (the **G** case). On a **G**-instance the complete-fork automata $\mathcal{W}^{\mathrm{od}}_{\mathcal{T},\varphi}$ of Proposition 8.2 serve, with every leaf in the role of an OD leaf: $R_\lambda = 0$ in base and ray, $\mathbf{g}_{G_\lambda}\geq 1$ for every λ, and $\mathbf{g}_{E_\pi}\geq 1$ for $\pi\in\varphi$. Soundness holds for every **G**-instance of any ambiguity, since Lemma 6.1 and Theorem 5.4(a) use no ambiguity bound and, by Remark 5.5, kill witnesses on a **G**-instance are complete-fork witnesses. Completeness under finite ambiguity follows from Propositions 8.2 and 6.7 over the shapes of Theorem 8.3, with $N=2^\kappa$ and $\kappa=\min(K-1,\kappa_B)$ in the bounds of Theorem C.6(2). K need not be known, since enumerating those shapes suffices, and since $\kappa\leq\kappa_B\leq s$ the bounds are in s alone. Proposition C.7 and Proposition C.12 hold unchanged with the cutoffs of Proposition 6.7 on the **G**-instance.

Offline, $\neg\exists H\,\mathrm{OFF}(H)$ has the analogous certificate in a case with $\varphi=\emptyset$: base $R=0$, ray with $R=0$ and $G\geq 1$ at every leaf. It is sound for every **G**-instance and complete under the K-promise, through H_{off} of Corollary 3.3. On the instances of Theorem 3.4, $\mathrm{OFF}(0)$ holds, so none exists there.

Remark C.10 (completeness needs bounded ambiguity). Soundness holds with no ambiguity bound. Completeness cannot, for this or for any certificate notion whose validity is decidable. The set of instances with a certificate

is recursively enumerable: enumerate cases, walks, cycles and integers, and check each candidate by Theorem C.6(4). Take the instances (A_*, B_M) of Theorem 3.4. They satisfy $\mathbf{G}$, hence $\mathbf{G}^\flat$ (Remark 2.3), at linear ambiguity. Suppose every non-halting M had a certificate. By Theorem 3.4, M halts iff J holds, and certificates are sound, so non-halting would be recursively enumerable. Halting is as well, so halting would be decidable. So some instance of Theorem 3.4 has $\neg J$ and no pumping certificate. The proof of Theorem C.6(2) fails exactly at Theorem 5.4(b): without the K-promise the witness shape is not confined to a finite class, so there are infinitely many automata and no single cutoff.

Remark C.11 (why each condition is needed). Base without zero rank balance: if $(\mathbf{v}_0)_{R_\lambda} \neq 0$, the pumped vectors have $c^A \neq c^B$ at that leaf, and G_λ compares births at different ranks, which says nothing about od. A ray with $\mathbf{g}_R \neq 0$ fails the same way after pumping. A ray growing only G (or only E) certifies $\neg Q_d^\flat$ for $d \leq \min_\pi(\mathbf{v}_0)_{E_\pi}$ (respectively $\neg\exists d\,\mathrm{GOOD}^\flat(d, H)$ for $H < \min_\lambda(\mathbf{v}_0)_{G_\lambda}$), as in Corollary C.8; both growth conditions are needed for $\neg J^\flat$. A cycle that meets no stem state cannot be spliced in; (P2) is the attachment condition $V(c) \subseteq V(\varsigma)$ of Lemma 6.5 in the weaker form "visits some stem state". A real base solution of $R = 0$ gives no walk; the certificate is integral, and completeness takes an integer point from $\mathrm{BAD}^\flat(T_*^\flat, T_*^\flat)$ before Lemma 6.6. The stem alone may need $Z^2 = 2^{12Ns+4N^2}$ rows, because $\mathcal{W}^\flat$ has exponentially many states; this proof gives nothing smaller. Whether a polynomial-size certificate exists is open. It would put fixed-K $J^\flat$ in coNP.

C.5. **Least lookahead and least bound.** On a $\mathbf{G}^\flat$-instance define

$$d^* = \min\{d : Q_d^\flat\}, \qquad H_\infty = \min_d H_{\min}(d),$$

$$H_{\min}(d) = \min\{H : \mathrm{GOOD}^\flat(d, H)\}, \qquad d_{\min}(H) = \min\{d : \mathrm{GOOD}^\flat(d, H)\},$$

each ∞ if its set is empty. By Lemma 2.6 the set $U = \{(d, H) : \mathrm{GOOD}^\flat(d, H)\}$ is upward closed in $\mathbb{N}^2$. So $H_{\min}$ and $d_{\min}$ are non-increasing, and the set $\mathcal{F}$ of minimal elements of U, the lookahead–distance frontier, is a finite antichain. All of this is the same for arbitrary memory (Theorem 4.4).

Proposition C.12 (least lookahead, least bound, frontier)**.** *Assume $\mathbf{G}^\flat$ and the $K^\flat$-promise.*

1. *$J^\flat \iff d^* < \infty$, and then $d^* \leq T_*^\flat$. d^* is computable by binary search on $[0, T_*^\flat]$, using $\lceil \log_2(T_*^\flat + 1) \rceil = O(N^3 s + N^4)$ decisions of $Q_x^\flat \iff \mathrm{GOOD}^\flat(x, H_*^\flat(x))$.*
2. *For each d: $H_{\min}(d) < \infty \iff Q_d^\flat$, and then $H_{\min}(d) \leq H_*^\flat(d)$. It is computable by binary search with $\mathrm{GOOD}^\flat(d, \cdot)$.*
3. *For each H: $d_{\min}(H) < \infty \iff \mathrm{GOOD}^\flat(D_*^\flat(H), H)$, and then $d_{\min}(H) \leq D_*^\flat(H)$. It is computable by binary search.*

(4) *If $J^\flat$ holds: $H_\infty \le T_*^\flat$, and H_∞ is the least H with* $\mathrm{GOOD}^\flat(D_*^\flat(H), H)$, *found by binary search on* $[0, T_*^\flat]$. *Moreover*

$$\mathcal{F} = \{(d_{\min}(H), H) : \ H_\infty \le H \le H_{\min}(d^*),\ d_{\min}(H) < d_{\min}(H-1) \text{ or } H = H_\infty\}.$$

It is computable, every element lies in $[d^*, D_*^\flat(H_\infty)] \times [H_\infty, H_*^\flat(d^*)]$, *and* $|\mathcal{F}| \le H_*^\flat(d^*) - H_\infty + 1$.

(5) *(Complexity, fixed K.) d^* (in binary, $O(s)$ bits), $H_{\min}(d)$ and $d_{\min}(H)$ (inputs in binary) and H_∞ are computable in polynomial space. The frontier $\mathcal{F}$ is enumerated in polynomial working space, its points written in sequence; its output may be exponentially long. Computing $H_{\min}(d)$ is PSPACE-hard already at $K = 1$, for every fixed d. Deciding $d^* < \infty$ is $J^\flat$, which is NL-hard, and computing d^* is not known to be harder than $J^\flat$. d^* can be $\Omega(s)$ at $K = 2$ and $2^{\Omega(\sqrt{s})}$ with K in the input. In general, d^* and $H_{\min}(d)$ are computable in* $\mathrm{DSPACE}(2^{O(K)}\mathrm{poly}(s, \log(d+2)))$.

Proof. (1) $J^\flat \iff \mathrm{GOOD}^\flat(T_*^\flat, T_*^\flat)$ (Proposition 6.7) gives $Q^\flat_{T_*^\flat}$, so $d^* \le T_*^\flat$. $Q_x^\flat$ is monotone in x (Lemma 2.6), and each test is decided by Proposition 6.7 with Theorem 4.4.

(2) $Q_d^\flat \iff \mathrm{GOOD}^\flat(d, H_*^\flat(d))$ by Proposition 6.7, and $\mathrm{GOOD}^\flat(d, \cdot)$ is monotone.

(3) Proposition C.7 and monotonicity.

(4) $\mathrm{GOOD}^\flat(T_*^\flat, T_*^\flat)$ gives $H_\infty \le T_*^\flat$. The property $d_{\min}(H) < \infty$ is upward closed in H and decided by (3). For $H \ge H_{\min}(d^*)$ we have $d_{\min}(H) = d^*$, since $\mathrm{GOOD}^\flat(d^*, H)$ holds and $\mathrm{GOOD}^\flat(d, H)$ implies $Q_d^\flat$. For $H < H_\infty$, $d_{\min}(H) = \infty$. So the minimal elements of U are the points where the non-increasing step function $d_{\min}$ drops on $[H_\infty, H_{\min}(d^*)]$, together with its first point, and $H_{\min}(d^*) \le H_*^\flat(d^*)$ by (2).

(5) Binary search is a polynomial-space driver over polynomially many oracle calls, of the following sizes. $Q_x^\flat$ with $x \le T_*^\flat$ is in $\mathrm{NSPACE}(O(N^4 s + N^5 + N^3 \log(x+2))) = \mathrm{NSPACE}(O(N^6 s + N^7))$ by Proposition 8.18(2b). $\mathrm{GOOD}^\flat(d, H)$ with $H \le H_*^\flat(d)$ is in the nondeterministic space of Proposition 8.18(2a), where the term $N \log H$ is $O(N^4 s + N^5 + N^3 \log(d+2))$; with $d \le D_*^\flat(H)$ the term $N^2 \log d$ is $O(N^5 s + N^6 + N^4 \log(H+2))$. Savitch's theorem removes the nondeterminism, and the Immerman–Szelepcsényi theorem handles the complemented calls. At fixed K all of these are $O(s + \log(d+2) + \log(H+2))$. PSPACE-hardness: $\mathrm{GOOD}^\flat(d, H) \iff H_{\min}(d) \le H$, and $\mathrm{GOOD}^\flat$ is PSPACE-hard at $K = 1$ for every fixed d (Proposition 8.18(3), from Corollary 8.11 by Proposition 8.17). NL-hardness: Proposition 8.7 through Proposition 8.17. The magnitudes of d^*: Proposition 8.14 and Theorem 8.13 through Proposition 8.17. □

Under **G**, Theorem 3.1 outputs $d = H = T_*$. Proposition C.12 adds that the optimal pair data are computable, at fixed K in polynomial space for

d^* and $H_{\min}(d)$, and with polynomial working space for $\mathcal{F}$; the dual cutoff of Proposition C.7 is what the frontier needs.

Appendix D. The reduction for linear ambiguity

We prove the statements of Section 7 whose proofs the body defers, then the slow-down of Section 7.9 and the two corollaries of Section 7.10. Throughout, $\mathcal{K} = \{1, 2, E, T\}$; when M halts, L is the length of its run from zero counters, $\gamma_k = (q_k, c_1^k, c_2^k)$ is its k-th configuration, and $c^* = \max_k \max(c_1^k, c_2^k) \le L$. A run ρ of B_M that has not entered CAN has imbalance $z(0) = 0$ and $z(t) = \mu_{x_t} z(t-1) + w_t$, where w_t is the offset of its t-th edge.

D.1. **Imbalance and runs.**

Proof of Lemma 7.4. Induction on t. At $t = 0$ every register is empty and $z(0) = 0$. Let $\mathrm{row}_{x_t}(\kappa) = [(\kappa_1, s_1), \dots, (\kappa_m, s_m)]$. In A_M the update $U(x_t, 0)$ gives X_κ and Y_κ the same length $\sum_j l_{\kappa_j}(t-1) + 2$, which is $l_\kappa(t)$. In the group RB of ρ, the register X'_κ receives X'_{κ_j} when $s_j = +$ and Y'_{κ_j} when $s_j = -$, then $2 + w_\kappa$ fresh letters. By induction,

$$|X'_\kappa(t)| = \sum_{s_j=+} \big(l_{\kappa_j} + z_{\kappa_j}\big) + \sum_{s_j=-} \big(l_{\kappa_j} - z_{\kappa_j}\big) + 2 + w_\kappa = l_\kappa(t) + (\mu_{x_t} z(t-1))_\kappa + w_\kappa,$$

which is $l_\kappa(t) + z_\kappa(t)$. The register Y'_κ receives the other register of every source pair and $2 - w_\kappa$ fresh letters. So the signs flip and $|Y'_\kappa(t)| = l_\kappa(t) - z_\kappa(t)$. □

Proof of Lemma 7.6. Call two runs *siblings* if they agree up to their last fork, an `st` read in a choice phase ϑ, and since then sit in $\mathrm{P}(\vartheta, 0)$ and $\mathrm{P}(\vartheta, 1)$ with identical updates. Siblings have identical RB contents. After the fork their `W`-edges have different sources, so they are different runs although every later update agrees.

Claim. After every prefix, the runs outside CAN are at most one run or exactly one pair of siblings. We argue by induction on the prefix. A single run in a non-pending state outside CAN has one edge on every letter, except on `st` in a choice phase. There it has two edges, both with update $U(\mathtt{st}, 0)$, into $\mathrm{P}(\vartheta, 0)$ and $\mathrm{P}(\vartheta, 1)$; the two successors are siblings. From a pair of siblings, `W` keeps both with the common update $U(\mathtt{W}, 0)$. The echo $\mathtt{ec}_b$ keeps sibling b, moving it to $\mathrm{res}(\vartheta, b)$, and sends sibling $1 - b$ to CAN. Every other letter sends both to CAN. Runs in CAN never branch.

(i) By the claim at most one run branches at each position, and it branches into two. The count starts at one.

(ii) Siblings have identical RB, and every state outside CAN outputs Φ on RB.

(iii) If some run has entered CAN, its output is A_M's, since PA has mirrored A_M's updates at every step (proof of Lemma 7.5). Otherwise no letter has sent a run to CAN. FIRST survives only `W` and `st`, and after the

$\mathtt{st}$ both siblings survive only $\mathtt{W}$. So $v \in \mathtt{W}^* \cup \mathtt{W}^*\mathtt{st}\,\mathtt{W}^*$. On such a word every RB has received only $U(\mathtt{W}, 0)$ and $U(\mathtt{st}, 0)$, which are A_M's updates, births included.

So the annotated outputs on v are at most two: the common one of the runs outside CAN, and A_M's. □

D.2. **Piece calculus.** A *piece* is the block of fresh letters that one update puts into one register: L_t^κ into X_κ and R_t^κ into Y_κ at step t, both born at t. In A_M every piece has length 2. In the RB group of a run outside CAN, L_t^κ has length $2 + w_\kappa(t)$ and R_t^κ has length $2 - w_\kappa(t)$, both in $\{1, 2, 3\}$. Updates only concatenate registers. So a piece is never split, and every output letter lies in exactly one piece.

Proof of Lemma 7.7. At step t both machines execute the row lists of x_t: A_M through $U(x_t, 0)$, and the run through $U(x_t, w_t)$ on every edge outside CAN. The row lists alone fix which registers are concatenated, in which order, and where the fresh piece goes; the offset changes only its length. By induction on t, each register holds the same sequence of piece names in both machines. The final expression Φ concatenates the registers in one fixed order. □

Sources move as pairs, so L_t^κ and R_t^κ always lie in the same register pair. Hence each pair has the same total length and the same multiset of births in A_M and in the run.

Lemma D.1 (transport)**.** *Let α_A and α_B be concatenations of the same pieces $\pi_1 \cdots \pi_N$ in the same order, with births $\beta_1, \ldots, \beta_N$. Let piece k have length 2 in α_A and length $\lambda_k \geq 1$ in α_B, with $\sum_k \lambda_k = 2N$. Put $\mathrm{dis}_k = \sum_{k'<k}(\lambda_{k'} - 2)$, the start of piece k in α_B minus its start in α_A, and $\mathrm{dis}^* = \max_k |\mathrm{dis}_k|$. The junction $(k-1, k)$ is live if $\mathrm{dis}_k \neq 0$. Let g be the largest $|\beta_k - \beta_{k-1}|$ over live junctions, and $g = 0$ if none is live. Then*

$$\mathrm{od}(\alpha_A, \alpha_B) \;\leq\; g\,\lceil \mathrm{dis}^*/2 \rceil.$$

Proof. Fix a rank j. In α_A it lies in piece $k_A = \lceil j/2 \rceil$. In α_B it lies in the piece k_B with $2(k_B - 1) + \mathrm{dis}_{k_B} < j \leq 2k_B + \mathrm{dis}_{k_B+1}$. The births at rank j are β_{k_A} and β_{k_B}.

Suppose $k_A < k_B$. Then $2(k_B - 1) - \mathrm{dis}^* < j \leq 2k_A$, so $k_B - k_A < \mathrm{dis}^*/2 + 1$, hence $k_B - k_A \leq \lceil \mathrm{dis}^*/2 \rceil$. For $k_A < k \leq k_B$, the start of piece k in α_B is at most the start of k_B there, which is below $j \leq 2k_A \leq 2(k-1)$, the start of piece k in α_A. So $\mathrm{dis}_k < 0$: every junction between k_A and k_B is live.

Suppose $k_B < k_A$. Then $2(k_A - 1) < j \leq 2k_B + \mathrm{dis}^*$, and again $k_A - k_B \leq \lceil \mathrm{dis}^*/2 \rceil$. For $k_B < k \leq k_A$, the start of piece k in α_B is at least the end of k_B there, which is at least $j > 2(k_A - 1) \geq 2(k-1)$. So $\mathrm{dis}_k > 0$, and the junctions between k_B and k_A are live.

Across each live junction the birth moves by at most g. So $|\beta_{k_A} - \beta_{k_B}| \leq g\,|k_A - k_B| \leq g\lceil \mathrm{dis}^*/2 \rceil$. □

Lemma D.2 (register shapes, discrepancies, junctions)**.** *Let ρ be a run of B_M on v that has not entered* CAN*, at a time $n \geq 1$, with imbalance $z(\cdot)$. A prefix or suffix of a register content is one cut at a piece boundary; its discrepancy is its length in ρ minus its length in A_M.*

(a) Counters. Let $\kappa \in \{1, 2, T\}$, and let c be the step of the last clear of κ, or $c = 1$ if it was never cleared. Then $X_\kappa(n) = L_c^\kappa \cdots L_n^\kappa$ and $Y_\kappa(n) = R_n^\kappa \cdots R_c^\kappa$. Put $\zeta(s) = z_\kappa(s)$ for $s \geq c$ and $\zeta(c-1) = 0$. The prefix of $X_\kappa(n)$ through L_s^κ has discrepancy $\zeta(s)$; the suffix of $Y_\kappa(n)$ from R_s^κ has discrepancy $-\zeta(s)$; the prefix of $Y_\kappa(n)$ through R_s^κ has discrepancy $\zeta(s-1) - \zeta(n)$.

(b) The accumulator. Let $s_1 < \cdots < s_r \leq n$ be the import steps, those reading `zr` *or* `ab`*, with signs σ_k and sources i_k; put $s_0 = 1$ and $s_{r+1} = n + 1$. Then*

$$X_E(n) = \mathcal{E}_0\, \alpha_1\, \mathcal{E}_1 \cdots \alpha_r\, \mathcal{E}_r, \qquad Y_E(n) = \mathcal{E}_r^R\, \alpha_r^R \cdots \alpha_1^R\, \mathcal{E}_0^R,$$

where $\mathcal{E}_k = L_{s_k}^E \cdots L_{s_{k+1}-1}^E$ and $\mathcal{E}_k^R = R_{s_{k+1}-1}^E \cdots R_{s_k}^E$ are all nonempty, and the atoms are $(\alpha_k, \alpha_k^R) = (X_{i_k}(s_k - 1), Y_{i_k}(s_k - 1))$ if $\sigma_k = +$ and $(Y_{i_k}(s_k - 1), X_{i_k}(s_k - 1))$ if $\sigma_k = -$. An atom is the content of a counter over one lifetime, so (a) describes it. The prefix of $X_E(n)$ through L_s^E has discrepancy $z_E(s)$; a prefix ending inside α_k has $z_E(s_k - 1)$ plus the discrepancy of the corresponding prefix of α_k; the suffix of $Y_E(n)$ from R_s^E has $-z_E(s)$; a suffix starting inside α_k^R has the discrepancy of the corresponding suffix of α_k^R minus $z_E(s_k - 1)$.

(c) Output level. Put $\alpha_A = [\![\rho_{A_M}(v)]\!]$ and $\alpha_B = [\![\rho]\!]$, in the setting of Lemma D.1. A pair boundary has dis $= 0$*. At a piece boundary inside pair κ,* dis *is the prefix discrepancy if the boundary lies in X_κ, and minus the suffix discrepancy from the next piece on if it lies in Y_κ. Always $|\mathrm{dis}| \leq 3 \max_{t \leq n} |z(t)|_\infty$.*

(d) Junctions. Every junction of consecutive output pieces has one of the following types, where c is the first step of the atom's lifetime.

type	*where*	*birth jump*	dis
J1	*inside a counter lifetime, an atom, an $\mathcal{E}_k$ or an $\mathcal{E}_k^R$*	1	*any*
J2	*middle of a pair, L_n^κ | R_n^κ*	0	*any*
J3	*pair boundary*	*any*	0
J4	$\mathcal{E}_{k-1}$ | α_k *and* α_k^R | $\mathcal{E}_{k-1}^R$	*$\sigma_k = +$: at most $s_k - 1 - c$; $\sigma_k = -$:* 0	$z_E(s_k - 1)$
J5	α_k | $\mathcal{E}_k$ *and* $\mathcal{E}_k^R$ | α_k^R	*$\sigma_k = +$: 1; $\sigma_k = -$: at most $s_k - c$*	$z_E(s_k - 1) + \sigma_k z_{i_k}(s_k - 1)$

So a birth jump above 1 *occurs only at J3, which is never live, at J4 with* $\sigma_k = +$*, and at J5 with* $\sigma_k = -$*.*

Proof. (a) Until a clear, every step appends L_t^κ at the right of X_κ and prepends R_t^κ at the left of Y_κ; a clear at step c leaves $X_\kappa = L_c^\kappa$ and $Y_\kappa = R_c^\kappa$. The prefix through L_s^κ is $X_\kappa(s)$, with discrepancy $z_\kappa(s)$ by Lemma 7.4, and the suffix from R_s^κ is $Y_\kappa(s)$, with discrepancy $-z_\kappa(s)$. The last formula is the total, $-\zeta(n)$, minus the suffix from R_{s-1}^κ.

(b) E is never cleared. An import at s_k puts the atom at the right end of $X_E(s_k - 1)$ and at the left end of $Y_E(s_k - 1)$, before the fresh pieces of step s_k. Later steps add only to the right of X_E and to the left of Y_E. So everything left of $L_{s_k}^E$ in X_E is frozen from step s_k on, and symmetrically in Y_E. The blocks with $k \geq 1$ are nonempty because $s_{k+1} > s_k$, and $\mathcal{E}_0$ because $s_1 \geq 3$: imports are legal only in Z and D, which a run outside CAN reaches only after FIRST, an `st` and an echo. The discrepancy of a frozen prefix is that of $X_E(s)$, from Lemma 7.4, plus the atom's internal part where the cut lies in an atom; likewise for suffixes of Y_E.

(c) Pair lengths agree by Lemma 7.7, so the output prefix through a pair boundary has discrepancy 0. Inside pair κ the output prefix discrepancy is the register prefix discrepancy when the cut is in X_κ. When the cut is in Y_κ it is $z_\kappa(n)$ plus the prefix discrepancy in Y_κ, which is minus the suffix discrepancy, since the pair totals agree. The general bound is $|z_E| + 2|z_i|$ at worst, by (a) and (b).

(d) Births are read off (a) and (b). An ascending atom $X_i(s-1)$ runs from birth c up to $s-1$, and a descending atom $Y_i(s-1)$ from $s-1$ down to c. $\mathcal{E}_{k-1}$ ends at birth $s_k - 1$ and $\mathcal{E}_k$ starts at s_k; $\mathcal{E}_k^R$ ends with $R_{s_k}^E$ and $\mathcal{E}_{k-1}^R$ starts with $R_{s_k-1}^E$. The last piece of $X_\kappa(n)$ and the first of $Y_\kappa(n)$ are both born at n. The dis values follow from (b) and (c). At J5 on the X side, for instance, $\mathrm{dis} = z_E(s_k - 1) + |\alpha_k|_\rho - |\alpha_k|_{A_M} = z_E(s_k - 1) + \sigma_k z_{i_k}(s_k - 1)$, and the Y side gives the same number through the suffix formula. □

D.3. **The lower bridge.**

Proof of Lemma 7.8. From each of these states `prb` leads to PRB, which loops on `prb`, and both machines apply the neutral update $U(\texttt{prb}, 0)$. So the extension is unique. Let $t = |h|$, fix κ with $z := z_\kappa \neq 0$, and put $x_A = l_\kappa(t)$. By Lemma 7.7 the same number P of output letters precedes the pair κ in both outputs. After the probes the pair reads, in A_M,

$$X_\kappa(h) \cdot \big(2m \text{ letters born } t{+}1, t{+}1, \ldots, t{+}m, t{+}m\big) \cdot \big(2m \text{ letters born } t{+}m, t{+}m, \ldots, t{+}1, t{+}1\big) \cdot Y_\kappa(h),$$

and in ρ the same with $X'_\kappa(h)$ of length $x_A + z$ and $Y'_\kappa(h)$ of length $x_A - z$, by Lemma 7.4.

If $z > 0$, look at rank $P + x_A + z$. In ρ it is the last letter of $X'_\kappa(h)$, born at or before t. In A_M it is the z-th probe letter, which exists as $z \leq 2m$, born at $t + \lceil z/2 \rceil$. If $z < 0$, look at the first letter of $Y'_\kappa(h)$ in ρ. It is the

$(x_A + |z|)$-th rank from the right end of the pair, and the right ends agree. In ρ it is born at or before t; in A_M it is the $|z|$-th probe letter from the right, born at $t + \lceil |z|/2 \rceil$. Either way od $\geq \lceil |z_\kappa|/2 \rceil$, and we take the κ that maximises it. □

D.4. **The controller for a halting machine.**

Proof of Lemma 7.9. Totality. The map χ is defined on all of $\{0..L\} \times \{0..c^*\} \times Q_B^{\mathrm{r}} \times \mathrm{Win}_0$, including pairs of memory and state that no play of τ reaches. The listed choices are legal edges for every (k, π), and every other pair of state and letter has one edge. The only update that could leave the memory set is $k := k+1$ at $k = L$; there we keep k. On plays of τ this never arises: by (I1) and (I2) the state is q_k, and $q_L = q_F$ has neither an inc edge nor a resolution $\mathrm{res}(\mathrm{C}(q_F), \cdot)$.

Invariants. Induction over the edges of the play. In FIRST and $\mathrm{P}(\mathrm{FIRST}, \cdot)$ we have $z = 0$ and $(k, \pi) = (0, 0)$.

- $\mathrm{C}(q)$ is entered in four ways. From $\mathrm{res}(\mathrm{FIRST}, \cdot)$: $z = 0$, $k = 0$ and $\gamma_0 = (q_I, 0, 0)$. From an $\mathrm{inc}(i, q')$ edge: offset $e_i + e_T$ and $k := k + 1$, and γ_{k+1} is γ_k with c_i raised by one and state q'. From `pass`: offset e_T, k unchanged, and (I1) in D gives (I1) in C. From FINISH: it is chosen only with $\pi = 0$, hence $z_E = 0$ by (I2), and $(q, z_1, z_2) = \gamma_k$ carries over. The value π changes only at `ab`, SPEND, `rst` and $\mathrm{res}(\mathrm{FIRST}, \cdot)$, so $\pi = 0$ throughout C, Z and D.
- `st` from a jz-state $\mathrm{C}(q)$ into $\mathrm{P}(\mathrm{C}(q), b)$ applies $U(\mathtt{st}, 0)$ and leaves z and k.
- The resolutions follow γ. ZERO is chosen iff $c_i^k = 0 = z_i$; it enters $\mathrm{Z}(i, q_z)$ with offset e_T and $k := k+1$, and $\gamma_{k+1} = (q_z, c_1^k, c_2^k)$. DEC is chosen iff $c_i^k > 0$; it enters $\mathrm{D}(i, q_d)$ with offset $-e_i + e_T$ and $k := k+1$, so $z_i = c_i^k - 1 = c_i^{k+1} \geq 0$.
- `zr` is legal only in Z, where $z_i = z_E = 0$. So z_E stays $0 = \pi$, z_i stays 0, and $\mathrm{F}(q)$ is entered with $(q, z_1, z_2) = \gamma_k$.
- $\mathtt{ab}(i)$ is legal only in D, where $z_E = 0$. It makes $z_E = z_i = c_i^k = \pi$, clears z_1, z_2, z_T and sets $k := 0$, which matches $\gamma_0 = (q_I, 0, 0)$ in $\mathrm{F}(q_I)$.
- SPEND is chosen iff $\pi > 0$. It lowers z_E and π together and leaves (q, z_1, z_2).
- `rst` is legal only at ACC, where $z_E = 0$. It clears z_1, z_2, z_T, sets $(k, \pi) := (0, 0)$ and enters $\mathrm{F}(q_I)$ with γ_0.
- `W`, `prb` and the state PRB change nothing.

This gives (I1) and (I2), and (I4) is read off the `zr` and `ab` cases. For (I3), the counters and z_E lie in $[0, c^*]$ by (I1) and (I2). The clock z_T starts at 0 and is cleared at `ab` and `rst`. Since its last clear, or the start, it gained at most $c^* + 1$ from the SPENDs and the FINISH right after the clear, and at most 3 per step of M taken from γ_0: an inc edge adds 1; ZERO adds 3, through the resolution, `zr` and FINISH; DEC adds 2, through the resolution

and `pass`. At most L such steps occur, since $k \le L$ and ACC admits no increment. So $z_T \le c^* + 1 + 3L \le 4L + 1$. □

D.5. **The adaptive environment.**

Proof of Lemma 7.13. (a) Let i be the position of the `st` of a choice block. The choice of σ there depends on its memory before i and on its window at i. That window is $\mathtt{st}\,\mathtt{W}^d$ without $\dashv$, because more letters follow. The memory before i depends on the windows at positions below i, which reach at most position $i - 1 + d$, inside the fixed part $\mathtt{st}\,\mathtt{W}^{d-1}$. So b is determined before the echo $\mathtt{ec}_b$ is written. Every letter of a block is legal where it is read: `W` in pending states, $\mathtt{ec}_b$ in $\mathrm{P}(\vartheta, b)$, and each one-letter block by the edge table. Hence the run never enters CAN.

(b) Every choice position i in h is followed inside h by $\mathtt{W}^d\,\mathtt{ec}_b$, so $i+d < |h|$. By induction over the positions $j \le |h|$: the choice at i depends only on windows at positions at most i, which lie inside h, show no $\dashv$ in hu either, and agree in hu and v_∞. At every other position B_M has one edge, so the edge is forced. The memory near the end of h may differ between hu and v_∞, but no choice inside h reads it.

(c) A block boundary h after the first block ends in a non-pending game state: C, Z, D or F, with ACC included. Take $u = \mathtt{prb}^m$ with $2m \ge |z(h)|_\infty$. By (b) the run of σ on hu begins with its run on h, so Lemma 7.8 gives $H \ge \mathrm{od} \ge \max_\kappa \lceil |z_\kappa(h)|/2 \rceil$, that is $|z(h)|_\infty \le 2H$. □

D.6. **Binary framing.**

Proof of Lemma 7.16. By the framing of Section 7.8, the runs of B_{bin} on v are, edge for edge, the runs of B_M on the event word $\mathrm{ev}(v)$, and the run of A_{bin} is the run of A_M on $\mathrm{ev}(v)$. Births are bit positions in both.

The promise transfers. The only new letter is `INV`; its rows are the identity, and it sends B to CAN. Lemmas 7.6, 7.7, 7.8, 7.9 and 7.10 hold on $\mathrm{ev}(v)$: they concern arbitrary words over Σ_0, and an internal bit decodes as `W`, which is legal and neutral in every state outside CAN. The choices of τ occur only at completing bits, and the decoder state with the completing bit determines the letter, so τ needs no lookahead. The vector z changes only at completing bits, so $\bar{Z}$ is unchanged.

For Lemma 7.13, write each block in frames: a choice block is $1\,\mathrm{id}(\mathtt{st})\,0^d\,1\,\mathrm{id}(\mathtt{ec}_b)$ and a one-letter block is $1\,\mathrm{id}(\lambda)$. A 0 after a completed frame decodes as `W`, which is legal in pending states. The choice position is the last bit of the `st` frame. Inside its block, $d + 5$ bits follow it, and its window is that bit followed by 0^d. So the window shows neither $\dashv$ nor the echo frame. Windows at earlier bits reach at most position $i - 1 + d$, inside the padding. Internal bits are forced. The probe extension $(1\,\mathrm{id}(\mathtt{prb}))^m$ applies neutral updates in both machines, so Lemma 7.8 holds with $5m$ probe positions. Lemma 7.14 is untouched. An incomplete last frame contributes only neutral ticks. □

D.7. **Proof of Theorem 3.4.** We first complete the source lemma.

Proof of Lemma 7.2. Membership is by simulation. For hardness we embed the machines of Dudenhefner [Dud23]. Let M' be a two-counter machine in the sense of Definition 2.1 there, with ℓ instructions indexed $0, \dots, \ell - 1$, each either inc_c or $\mathrm{dec}_c\, j$ for a counter $c \in \{0, 1\}$, and let $a, b \in \mathbb{N}$. We build M in the model of Section 7.1.

- States: q_F, a chain $p_0 = q_I, p_1, \dots, p_{a+b}$, and $q_0, \dots, q_{\ell-1}$. Identify p_{a+b} with q_0, or with q_F if $\ell = 0$, and read q_k as q_F for every $k \geq \ell$.
- Initialisation: p_s executes $\mathrm{inc}(1, p_{s+1})$ for $s < a$ and $\mathrm{inc}(2, p_{s+1})$ for $a \leq s < a + b$.
- Instructions: index k holding inc_c becomes $\mathrm{inc}(c + 1, q_{k+1})$; index k holding $\mathrm{dec}_c\, j$ becomes $\mathrm{jz}(c + 1, q_{k+1}, q_j)$.

From $(q_I, 0, 0)$, M reaches (q_0, a, b) in $a + b$ steps. From there it simulates M' step for step from the configuration $(0, (a, b))$. Increments match. A dec on a zero counter moves to the next index, which is the zero branch of jz. A dec on a positive counter decrements and jumps to j, which is the other branch. A configuration of M' is terminal exactly when its index is at least ℓ, since every instruction has a step in both cases, and those indices map to q_F. So M' terminates from $(0, (a, b))$ iff M halts from zero counters, and $(M', a, b) \mapsto M$ is computable. Dudenhefner shows in Lemma 2.4 of [Dud23] that termination from $(0, (a, b))$ is RE-hard under many-one reductions, so halting from zero counters in our model is RE-hard, and RE-complete. □

Proof of Theorem 3.4. Take the binary pair $(A_{\mathrm{bin}}, B_{\mathrm{bin}})$ of Section 7.8 as (A_*, B_M). The construction is effective in M: B_M has $O(|Q_M|)$ phases times 16 decoder states, and 16 registers.

Promise, run count, annotations. Lemmas 7.5 and 7.6 at the letter level, transferred by Lemma 7.16: **G** holds, there are at most $|v| + 1$ runs on v, and at most two annotated outputs, one of them A_*'s. So an offline choice with $\mathrm{od} = 0$ exists on every word.

Halting and bounded distance. If M halts, Corollary 7.11 with Lemma 7.16 gives $\mathrm{GOOD}(0, 2L + 1)$, hence Q_d for every d by Lemma 2.6. Q_d implies Q_d^∞ by definition. If Q_d^∞ holds for some d, then M halts by Corollary 7.15 with Lemma 7.16. Finally J is $\exists d\, Q_d$.

The reference is fixed. The one-state machine A_M applies $U(x, 0)$ on letter x, and $U(x, 0)$ depends only on the row lists of Section 7.2, which are fixed for the 13 letters of Σ_0 and never mention M. The framing gives A_{bin} the decoder's states β and f_u for $u \in \{0, 1\}^{\leq 3}$, which are $1 + 1 + 2 + 4 + 8 = 16$ states, and on each bit it applies $U(e, 0)$ for the bit's event e, with `INV` read as `W`. The decoder and the code id are chosen once. Its registers are X_κ, Y_κ for $\kappa \in \mathcal{K}$, which are 8, and its final expression is Φ. So $A_{\mathrm{bin}} = A_*$ is one machine for every M. Only the control of B_M, the states $\mathrm{C}(q)$, $\mathrm{Z}(i, q)$, $\mathrm{D}(i, q)$, $\mathrm{F}(q)$ and their pending states, depends on M. With a block p as

in Remark 7.3, the reference depends on p alone. No step above uses a property of A_* beyond these.

RE-*completeness.* On **G**-instances, $\mathrm{GOOD}(d, H)$ is decidable for given (d, H) by Theorem 4.4 with Proposition 8.17. So $Q_d = \exists H\, \mathrm{GOOD}(d, H)$ and $J = \exists d\, \exists H\, \mathrm{GOOD}(d, H)$ are in RE. Hardness: halting from zero counters many-one reduces to Q_d, for each fixed d, and to J through $M \mapsto (A_*, B_M)$, and that source problem is RE-complete by Lemma 7.2. Undecidability follows. □

D.8. **The growth frontier.**

Proof of Proposition 3.6. Let $\mathcal{E}$ be the trim part of the edge-state NFA $\mathcal{E}_B$ of Lemma 8.19, the states reachable from ι that reach a final state. It has the same accepting runs as B up to the bijection of that lemma, so it is not finitely ambiguous. Weber and Seidl [WS91] characterise finite ambiguity of a trim NFA: it fails exactly when there are states $p \neq q$ and a word v with paths $p \xrightarrow{v} p$, $p \xrightarrow{v} q$ and $q \xrightarrow{v} q$. Kiefer, Mazowiecki, Ryzhikov and Wiśniewski [KMRW26] restate the criterion for the transition-set model that $\mathcal{E}$ uses. Fix such p, q, v and fixed paths π_{pp}, π_{pq}, π_{qq}. Here $v \neq \varepsilon$, since an empty path cannot join p to $q \neq p$. By trimness there are paths $\iota \xrightarrow{u} p$ and $q \xrightarrow{x} f$ with f final.

For $m \geq 1$ and $1 \leq i \leq m$, the path

$$R_i = (\iota \xrightarrow{u} p)\ \pi_{pp}^{i-1}\ \pi_{pq}\ \pi_{qq}^{m-i}\ (q \xrightarrow{x} f)$$

is an accepting run on $uv^m x$. After the prefix uv^j it sits at p when $j < i$ and at q when $j \geq i$. So R_i and $R_{i'}$ with $i < i'$ are at different states after uv^i, and the m runs are distinct. The word has length $|u| + |x| + m|v|$. Put $c_0 = |u| + |x|$ and $k = |v|$. For $n \geq c_0$ take $m = \lfloor (n - c_0)/k \rfloor$; the word has length at most n, and a_B is monotone in n. The case $m = 0$ is empty. □

Corollary D.3 (sublinear growth is bounded)**.** *Let $g : \mathbb{N} \to \mathbb{N}$ with $\liminf_n g(n)/n = 0$. If (A, B) satisfies $\mathbf{G}^\flat$ and $a_B(n) \leq g(n)$ for all n, then B is finitely ambiguous. So $J^\flat$, $Q_d^\flat$ and $\mathrm{GOOD}^\flat$ are decidable on this class by Theorem 3.1, in 2-EXPSPACE by Proposition 8.22, with no knowledge of g. Under* **G** *the same holds with Remark 3.2.*

Proof. If B were not finitely ambiguous, Proposition 3.6 would give $a_B(n) \geq (n - c_0)/k - 1$ for all $n \geq c_0$. Along a sequence n_j with $g(n_j)/n_j \to 0$ this exceeds $g(n_j)$ for large j. □

For the undecidable side we need Theorem 3.4 at every linear slope.

Construction. Let $(A, B) = (A_*, B_M)$ be the instance of Theorem 3.4 over $\Sigma = \{0, 1\}$. For $X \in \{A, B\}$, the machine $X^{(k)}$ has states $Q_X \times \{0, \ldots, k-1\}$, initial state $(q_0, 0)$ and $\Phi(q, r) = \Phi_X(q)$. For $r \neq k - 1$ and every letter a it has one edge $(q, r) \xrightarrow{a} (q, r+1)$, with the identity update and no fresh letters. For $r = k - 1$ and every $e \in \Delta_X(q, a)$ it has an edge $(q, k-1) \xrightarrow{a} (\mathrm{dst}\, e, 0)$ with update $\mathrm{upd}(e)$. For $w = a_1 \cdots a_n$ write $w^{\downarrow} = a_k a_{2k} \cdots a_{k\lfloor n/k \rfloor}$.

Proof of Proposition 7.17. (i) *Runs.* The runs of $X^{(k)}$ on w are in bijection with the runs of X on $w^{\downarrow}$: filler positions are forced, and position kj carries the j-th edge. A letter born at j in the run on $w^{\downarrow}$ is born at kj in $X^{(k)}$, and the identity steps move nothing. The final expressions agree. So the annotated output in $X^{(k)}$ is that of X with every birth multiplied by k, and od in $X^{(k)}$ is k times od in X on $w^{\downarrow}$. The promise transfers: $A^{(k)}$ is deterministic and complete and $B^{(k)}$ is input-enabled, because the filler edges exist everywhere and position kj copies the edges of X; copylessness is clear; word agreement holds through the bijection. The run count is at most $|w^{\downarrow}| + 1 = \lfloor |w|/k \rfloor + 1$, and the two annotations come from Theorem 3.4. $A_*^{(k)}$ depends only on A_* and k.

(ii) *Halting gives* $\mathrm{GOOD}(0, k(2L+1))$. Let τ' take the forced edge at filler positions and, at position kj, play the choice of τ from Section 7.6, updating τ's memory only there. The residue r is part of the state of $B^{(k)}$, so the controller sees it. Its run on w is the image of τ's run on $w^{\downarrow}$, so od $\le k(2L+1)$ by Lemmas 7.10 and 7.16. τ' has finite memory and lookahead 0, and Lemma 2.6 gives every d.

(iii) $Q_{d'}^{\infty}$ *gives halting.* Let σ' have lookahead d', arbitrary memory and od $\le H'$ on all words. Put $D = \lceil d'/k \rceil$. For $v \in \Sigma^*$ let $\mathrm{pad}(v) = (0^{k-1}v_1)(0^{k-1}v_2)\cdots(0^{k-1}v_{|v|})$, so $\mathrm{pad}(v)^{\downarrow} = v$ and $|\mathrm{pad}(v)| = k|v|$. We define a lookahead-D controller σ for B. Its memory stores $v_1 \cdots v_{j-1}$ and the memory of σ'. At position j of v it simulates σ' over the positions $k(j-1)+1, \ldots, kj$ of $\mathrm{pad}(v)$ and plays the B-edge under σ''s edge at kj.

The simulation needs the window of σ' at each such position p. That window is $\mathrm{pad}(v)[p..\min(kn, p+d')]$, followed by $\dashv$ iff $kn - p < d'$. Its letters sit at positions at most $p + d' \le kj + d'$, so they involve only $v_{j'}$ with $j' \le j + \lfloor d'/k \rfloor \le j + D$, capped at n; these are in σ's window or in its memory. For the end marker: if σ's window shows $\dashv$, then n is known exactly. If it does not, then $n \ge j + D \ge j + d'/k$, so $kn \ge kj + d' \ge p + d'$, and the window of σ' shows no $\dashv$ either. So σ is well defined. Its run on v is the B-image of σ''s run on $\mathrm{pad}(v)$, and by (i) its od is od on $\mathrm{pad}(v)$ divided by k, at most H'/k. So Q_D^{∞} holds for (A_*, B_M), and M halts by Theorem 3.4.

(iv) RE membership is as in the proof of Theorem 3.4. □

Corollary D.4 (the growth frontier is sharp)**.** *Let $g : \mathbb{N} \to \mathbb{N}$, and let $\mathcal{C}_g$ be the class of $\mathbf{G}$-instances, or of $\mathbf{G}^{\flat}$-instances, with $a_B(n) \le g(n)$ for all n.*

- *(i) If $\liminf g(n)/n = 0$, then J and Q_d are decidable on $\mathcal{C}_g$.*
- *(ii) If $\liminf g(n)/n > 0$ and $g \ge 1$, then for every fixed d, Q_d and J are* RE*-complete on $\mathcal{C}_g$, already with a fixed reference.*

Proof. (i) is Corollary D.3. For (ii), choose a finite c with $0 < c < \liminf g(n)/n$; the liminf may be $+\infty$, as for polynomial g. Choose n_2 with $g(n) \ge cn/2$ for all $n \ge n_2$, and an integer $k > \max(n_2, 4/c)$. For $n < k$ we have $\lfloor n/k \rfloor + 1 = 1 \le g(n)$. For $n \ge k$ we have $n \ge n_2$ and $n > 4/c$, so

$\lfloor n/k \rfloor + 1 \le cn/4 + 1 \le cn/2 \le g(n)$. So the instances of Proposition 7.17 at this k lie in $\mathcal{C}_g$, and they are RE-complete there with the fixed reference $A_*^{(k)}$. Under $\mathbf{G}^\flat$ use Proposition 8.17. The condition $g \ge 1$ is needed under $\mathbf{G}$ only because every word has a run there. □

D.9. **No computable memory or distance bound.**

Proof of Lemma 7.18. Lookahead 0. Let D have states $Q_B^{\mathrm{r}} \times M$, initial state (q_0^B, m_0) and $\Phi_D(q,m) = \Phi_B(q)$. For a state (q,m) and a letter a with $\chi(m,q,a) = (e,m')$, D has the single edge $(q,m) \xrightarrow{a} (\operatorname{dst} e, m')$ with update $\operatorname{upd}(e)$. It is deterministic and complete, since χ is total on $M \times Q_B^{\mathrm{r}} \times \mathrm{Win}_0$ and $\operatorname{dst} e$ is reachable. It is copyless, and on w it applies exactly the updates of $\rho_{\mathcal{C}}(w)$, so it has the same annotated output and word agreement holds. Hence (A,D) is a $\mathbf{G}$-instance with one run per word, and $J(A,D)$ holds iff $\sup_w \operatorname{od}(\rho_A(w), \rho_{\mathcal{C}}(w)) < \infty$. Remark 3.2 decides it, in PSPACE by Theorem 8.3.

Lookahead $d \ge 1$. The controller reads windows, so we let B'' guess them and verify the guesses. Its states are $Q_B^{\mathrm{r}} \times M \times (\mathrm{Win}_d \cup \{\star, \emptyset\})$, with initial state $(q_0^B, m_0, \star)$. Its final states are the $(q,m,\emptyset)$, with $\Phi_B(q)$, and the initial state, with $\Phi_B(q_0^B)$. From (q,m,γ) with $\gamma \ne \emptyset$ on letter a: the current window ω is γ if $\gamma \in \mathrm{Win}_d$, and any element of Win_d if $\gamma = \star$, with first letter a in both cases; with $\chi(m,q,\omega) = (e,m')$ and any $\omega' \in \operatorname{succ}(\omega)$ there is an edge to $(\operatorname{dst} e, m', \omega')$ with update $\operatorname{upd}(e)$. Here $\operatorname{succ}(b_0 \cdots b_d) = \{b_1 \cdots b_d c : c \in \Sigma\} \cup \{b_1 \cdots b_d \dashv\}$, $\operatorname{succ}(b_0 \cdots b_j \dashv) = \{b_1 \cdots b_j \dashv\}$ for $j \ge 1$, and $\operatorname{succ}(b_0 \dashv) = \{\emptyset\}$. A state with third component $\emptyset$ has no edges.

Claim: on every w, B'' has exactly one accepting run, and its B-edges form $\rho_{\mathcal{C}}(w)$. For $w = \varepsilon$ the empty run at the final initial state is the only one. Let $n \ge 1$ and let $\omega_1, \ldots, \omega_n$ be the windows an accepting run uses. It ends in an $\emptyset$-state, so $\omega_n = a_n \dashv = \mathrm{win}_d(w,n)$. Suppose $\omega_{i+1} = \mathrm{win}_d(w, i+1)$; we know $\omega_{i+1} \in \operatorname{succ}(\omega_i)$ and that ω_i starts with a_i. If ω_i is full, it is a_i followed by the first d symbols of ω_{i+1}. Either ω_{i+1} is full, so $n \ge i+1+d$, or $\omega_{i+1} = a_{i+1} \cdots a_{i+d} \dashv$, so $n = i+d$; in both cases $\omega_i = a_i \cdots a_{i+d} = \mathrm{win}_d(w,i)$. If $\omega_i = b_0 \cdots b_j \dashv$ with $j \ge 1$, then $\omega_{i+1} = b_1 \cdots b_j \dashv = a_{i+1} \cdots a_n \dashv$, so $n = i+j$ and $\omega_i = \mathrm{win}_d(w,i)$. The case $\omega_i = b_0 \dashv$ with $i < n$ is impossible, since its successor $\emptyset$ has no edges. So every guessed window is the true one, and the edges and memories follow χ, which gives $\rho_{\mathcal{C}}(w)$. The true windows satisfy all constraints, which gives existence.

Hence (A, B'') satisfies $\mathbf{G}^\flat$: A is deterministic and complete, $\operatorname{dom}(B'') = \Sigma^*$, word agreement holds because the unique accepting run outputs $[\![\rho_{\mathcal{C}}(w)]\!]$, and the $1^\flat$-promise holds.

Claim: $J^\flat(A, B'')$ holds iff $\sup_w \operatorname{od}(\rho_A(w), \rho_{\mathcal{C}}(w)) < \infty$. For the direction from bounded distance, take a memoryless controller with lookahead $d+1$ for the completion B''_z of B'' by a sink z (Section 2.3). It sees $\mathrm{win}_d(w,i)$ and $\mathrm{win}_d(w,i+1)$. At a state (q,m,γ) with $\gamma = \star$, or with $\gamma \in \mathrm{Win}_d$ equal to $\mathrm{win}_d(w,i)$, it plays the B''-edge with $\omega = \mathrm{win}_d(w,i)$ and $\omega' = \mathrm{win}_d(w,i+1)$,

or $\omega' = \emptyset$ when the window is $a_i \dashv$. At every other state, a wrong stored guess, an $\emptyset$-state or z, it plays a fixed legal edge of B''_z, which exists since B''_z is complete. So the controller is total on the reachable states. By induction on i, on every actual play the stored guess is the true window, so only the first case is used, and the controller produces the unique accepting run. This gives $\mathrm{GOOD}^\flat(d+1, H)$. Conversely, any controller meeting the objective produces the unique accepting run, whose annotated output is $[\![\rho_{\mathcal{C}}(w)]\!]$.

$J^\flat$ is decidable by Theorem 3.1. □

Proof of Corollary 7.19. Suppose f exists. Given M, compute (A_*, B_M), its size s_M and $f(s_M)$. The controllers with lookahead d and memory $[m]$ for $m \le f(s_M)$ form a finite computable set: the maps $[m] \times Q_B^{\mathrm{r}} \times \mathrm{Win}_d \to E_B \times [m]$ that choose legal edges, where Q_B^{r} is computable. Decide each by Lemma 7.18. If one has bounded od, then Q_d holds and M halts by Theorem 3.4. If none does, then M does not halt, by the assumption on f. This decides halting from zero counters, against Lemma 7.2. For the joint form, enumerate $d \le f(s_M)$ as well.

If only finitely many halting M violated f, raise f at those finitely many sizes to the least memory that works, which is finite. The result is a computable f' with no violations, which we have just excluded. □

Proof of Corollary 7.20. $\mathrm{GOOD}(d, H)$ is decidable for given (d, H), by Theorem 4.4 with Proposition 8.17. If g existed, decide $\mathrm{GOOD}(g(s_M), g(s_M))$. A positive answer gives J, so M halts. A negative answer means M does not halt, by the assumption on g and Lemma 2.6. This decides halting, against Lemma 7.2. The fixed-d form follows from Lemma 2.6, since $\mathrm{GOOD}(d, g)$ implies $\mathrm{GOOD}(\max(d, g), \max(d, g))$. Infinitely many violations: as in Corollary 7.19. □

Under finite ambiguity the threshold and the memory are computable from the instance (Theorem 3.1, Corollary 8.4, Proposition 8.22). On the family of Theorem 3.4 the least lookahead is 0 on every positive instance, so the incomputability sits entirely in the memory and in H. This matches τ: memory $(L+1)(c^*+1)$ and $H = 2L+1$, with L the halting time. By Proposition 7.17, both corollaries hold at every linear slope.

Appendix E. Complexity proofs

We prove the complexity, lookahead, memory, and ambiguity-growth results deferred to this appendix.

E.1. **The resource bounds.** The resource bounds use

$$N = \begin{cases} 2^{\min(K-1, \kappa_B)}, & \mathbf{G}, \\ 2^K, & \mathbf{G}^\flat. \end{cases}$$

On positive instances, the respective thresholds and extracted controllers satisfy

$$\log_2 T = O(N^3 s + N^4), \qquad |\,\mathrm{Mem}\,| \le \begin{cases} |Q_A|\, 2^{2^{O(N^3 s+N^4)}}, & \mathbf{G}, \\ (|Q_A|+1)\, 2^{2^{O(N^3 s+N^4)}}, & \mathbf{G}^\flat. \end{cases} \tag{7}$$

Here $T = T_*$ or $T_*^\flat$, and $d = H = T$ suffices. For fixed K, this gives exponential lookahead and doubly exponential memory. Under $\mathbf{G}$, fixing k as well gives polynomial lookahead and memory at most $|Q_A|2^{\mathrm{poly}(s)}$.

For fixed distance H, the complement of the origin monitor has an NFA with

$$2^{O(k)}|\Gamma|(\mu H+1)(H+2)$$

states, where μ bounds the letters inserted by one update. Determinization gives a DFA singly exponential in this quantity.

The finite-ambiguity decision procedures use nondeterministic space

$$\begin{cases} 2^{O(\kappa)}\,\mathrm{poly}(s, \log(d+2), \log(H+2)), & \mathbf{G}, \quad \kappa = \min(K-1, \kappa_B), \\ 2^{O(K)}\,\mathrm{poly}(s, \log(d+2), \log(H+2)), & \mathbf{G}^\flat. \end{cases}$$

Terms for unsupplied parameters are omitted. At fixed K, both J and $J^\flat$ belong to $\mathrm{NSPACE}(O(s))$.

Kiefer, Mazowiecki, Ryzhikov, and Wiśniewski give the ambiguity bound $K \le 2^{O(s)}$ used when K is omitted [KMRW26]. Their work is an unrefereed preprint. Weber and Seidl also supply an exponential-polynomial ambiguity bound sufficient for the 2-EXPSPACE conclusion [WS91]. Under $\mathbf{G}^\flat$, the displayed bounds then give triply exponential lookahead and quadruply exponential memory.

E.2. **Fork budgets and bounded witness graphs.**

Proof of Lemma 8.1. Suppose a run visits a branching pair (q, a) twice. The segment between those visits reads a nonempty word u and returns to q. Its first edge reads a. Repeat this segment r times after the original prefix.

At each repetition, take an alternative a-edge and then complete the remaining input. Such a completion exists by input-enabledness. All resulting runs accept. Their first departures from the repeated path occur at different positions, so they are distinct. Together with the repeated path, they give at least $r+1$ runs. This contradicts finite ambiguity.

Thus each branching pair contributes at most once to the fork sum. For a fixed word and run, the same first-departure argument gives

$$1 + \sum_i \big(|\Delta_B(q_{i-1}, a_i)| - 1\big)$$

distinct runs. The two bounds are therefore κ_B and $K-1$, respectively. □

A shape whose path budgets are at most κ has at most 2^κ leaves. Indeed, if its root has degree g, each subtree has budget at most $\kappa-g+1$. Induction gives

$$L(\kappa) \le \max_{2\le g\le \kappa+1} g\, 2^{\kappa-g+1} \le 2^\kappa, \qquad L(0)=1.$$

Here $g \le 2^{g-1}$. The complete-fork case of Theorem 5.4 consequently uses only $\mathfrak{S}'_\kappa$. The witness and cutoff arguments remain valid on this finite shape class.

Proof of Proposition 8.2. An OD leaf has

$$2\cdot|Q_A||Q_B|\cdot 2^k\cdot 9^k\cdot 4 = 8\cdot 18^k|Q_A||Q_B|$$

possible local states. These factors record its status, machine states, use bits, endpoint maps, and birth flags. Since $|Q_A|+|Q_B|+k\le s$, its logarithm is at most

$$3+\log_2|Q_A|+\log_2|Q_B|+k\log_2 18 \le 6s.$$

Node phases contribute at most 2^N possibilities. Pair phases contribute at most $3^{\binom{N}{2}}$. Hence

$$\log_2 Z \le 6Ns+N+\binom{N}{2}\log_2 3 \le 6Ns+2N^2.$$

A row changes a rank coordinate by at most μ. Each gap or coherence coordinate changes by at most one. There are at most $2N+\binom{N}{2}$ coordinates.

Take $m=2N+\binom{N}{2}$ and $W=\max(\mu,1)$. Proposition 6.7 applies with

$$T_* = m!(Z^2W+1)^m+1, \qquad H_*(d)=m!(Z^2W+d+1)^m+1.$$

Using $\log_2 m! \le m\log_2 m$, we obtain

$$\log_2 T_* \le m\log_2 m + m(12Ns+4N^2+\log_2 s+1)+1.$$

Adding d inside the power contributes $O(N^2\log(d+1))$. For fixed K, N is constant. In general $\kappa\le s$, which gives the stated exponential and double-exponential bounds. □

Proof of Theorem 8.3. A path to a target of $\mathcal{G}_{d,H}$ projects to an accepting witness walk. The rank counters remain exact. The unsaturated gap and coherence weights are at least their stored values. The target therefore satisfies

$$R_\lambda = 0, \qquad G_\lambda > H, \qquad E_\pi \ge d,$$

which are the guards for $\mathrm{BAD}(d,H)$.

Conversely, take a coherent bad witness with shape in $\mathfrak{S}'_\kappa$. At each leaf λ, choose the least output rank c_λ whose birth gap exceeds H. Word agreement makes this a common rank of the two outputs. Use Lemma 6.1 with that rank marked in both runs. The live P-occurrences are precisely the ranks below c_λ.

After input step r, the rank counter is the difference between the numbers of those occurrences already born in the two runs. Lemma 4.1 places it in

$[-\mu_B H, \mu_A H]$. At termination it is zero. Thus the range restriction deletes none of this walk. Saturating its other counters reaches a target of $\mathcal{G}_{d,H}$.

A vertex stores its case, a witness-automaton state, and the counters. It needs

$$\operatorname{poly}(N, s) + O\big(N \log(\mu H + 2) + N^2 \log(d + 2)\big)$$

bits. A guessed row contains, per leaf, a letter, two edge identifiers, and at most $O(\mu)$ colours. The row conditions can be checked by scanning the explicit updates. For constant N, this takes $O(s)$ workspace beyond the counters. A search need consider only simple paths. A counter bounded by the number of encoded vertices therefore makes the nondeterministic search halt within the same space bound.

Theorem 6.3, restricted to the complete-fork cases, identifies non-reachability with $\mathrm{GOOD}(d, H)$. The Immerman–Szelepcsényi theorem gives the nondeterministic complement bound. Savitch's theorem gives deterministic space $O(S_{d,H}^2)$.

For Q_d, substitute $(d, H_*(d))$. For J, substitute (T_*, T_*). The thresholds have

$$O(N^3 s + N^4 + N^2 \log(d + 1))$$

bits and can be computed within that space. When K is fixed, a vertex and its row test for J use $O(s)$ bits. The configuration graph then has $2^{O(s)}$ vertices, giving deterministic time $2^{O(s)}$. When $K = O(\log s)$, the bounds are polynomial. Without a supplied K, we use $\kappa_B \le s$ and obtain EXPSPACE. □

Proof of Corollary 8.4. Determinize the GAP-only automaton $\mathcal{N}_H$ and complement its accepting set. On joint words of the two runs, word agreement makes this an exact distance monitor. Its state is a subset of $\mathcal{N}_H$.

In the quotient arena of Proposition 4.5, the controller stores the reference state and this subset. The current B-state and input window are supplied to its transition function. Hence its memory has at most $|Q_A| 2^{|\mathcal{N}_H|}$ values. The finite safety game can be solved in time polynomial in

$$|Q_A||Q_B|\, 2^{|\mathcal{N}_H|}\, |\Sigma|^{d+1}.$$

Set $(d, H) = (T_*, T_*)$ and use

$$|\mathcal{N}_H| \le 2^{O(k)} (\mu H + 1)(H + 2).$$

Proposition 8.2 yields the stated memory bound. The substitution $(d, H_*(d))$ proves the assertion for Q_d. □

E.3. **Fixed ambiguity and register count.** Here K is fixed. We use $N = 2^{K-1}$, $m = 2N + \binom{N}{2}$, and all shapes in $\mathfrak{S}'_{K-1}$. Keep the factors in the preceding state count:

$$Z_0 = (8 \cdot 18^k |Q_A||Q_B|)^N\, 2^N\, 3^{\binom{N}{2}}, \qquad W_0 = \max(\mu, 1).$$

Then

$$\log_2 Z_0 = O_K(k + \log s).$$

Define

$$\beta(x) = Z_0^2 W_0 + x + 1, \qquad T_* = m!\beta(0)^m + 1,$$
$$H_*(x) = m!\beta(x)^m + 1, \qquad D_*(x) = m!\beta(x)^m + 1, \qquad P_* = m!\beta(T_*)^m + 1.$$

At fixed k, these quantities are polynomial in s when $x \le T_*$. They are computable in logarithmic space. Indeed, all input counts fit in $O(\log s)$ bits, and only constantly many products and fixed powers are required. For varying k, their bit lengths are $O_K(k + \log s)$.

Lemma E.1 (Cutoff in lookahead at fixed distance)**.** *For $d \ge D_*(H)$,*

$$\mathrm{GOOD}(d, H) \iff \mathrm{GOOD}(D_*(H), H).$$

Proof. It suffices to prove the corresponding equivalence for BAD. The implication from larger to smaller lookahead follows from the guards $E_\pi \ge d$.

Suppose a bad vector exists at $(D_*(H), H)$. By Lemma 6.5, it belongs to a piece $a + \mathbb{N}V$ with

$$\|a\|_\infty \le Z_0^2 W_0, \qquad \|v\|_\infty \le Z_0 W_0 \quad (v \in V).$$

Write $y = a + \sum_{v \in V} \lambda_v v$. With nonnegative variables $\lambda_v, \sigma_\lambda, \sigma_\pi, t$, impose

$$y_{R_\lambda} = 0, \qquad y_{G_\lambda} - \sigma_\lambda = H + 1, \qquad y_{E_\pi} - t - \sigma_\pi = 0.$$

There are at most m equations. Their coefficients and right-hand sides have absolute value at most $\beta(H)$. Our vector supplies an integer solution with $t = D_*(H) > m!\beta(H)^m$.

Lemma 6.6 therefore makes t unbounded on this polyhedron. It also supplies a nonnegative integer recession direction with positive t-coordinate. Add its integer multiples to the existing integer solution. The coefficients λ_v remain nonnegative integers, so the resulting vectors remain in the same piece. Their coherence coordinates exceed arbitrarily large thresholds. Downward closure gives $\mathrm{BAD}(d, H)$ for every d. □

Lemma E.2 (Bounded parameters)**.** *The following replacements preserve the indicated decision problem:*

$$J \iff \mathrm{GOOD}(T_*, T_*), \qquad Q_d \iff \mathrm{GOOD}(\bar{d}, H_*(\bar{d})), \quad \bar{d} = \min(d, T_*).$$

For $\mathrm{GOOD}(d, H)$*, use*

$$(d', H') = \begin{cases} (d, \min(H, H_*(d))), & d < T_*, \\ (\min(d, D_*(H)), H), & d \ge T_* > H, \\ (T_*, T_*), & d, H \ge T_*. \end{cases}$$

Every resulting parameter is at most P_*. For fixed K, k, the replacements are computable in logarithmic space, including when the supplied parameters are binary.

Proof. Proposition 6.7 gives the assertion for J and the distance cutoff at fixed d. If $d > T_*$, then

$$Q_d \Longrightarrow J \Longrightarrow \mathrm{GOOD}(T_*, T_*) \Longrightarrow \mathrm{GOOD}(T_*, H_*(T_*)).$$

Monotonicity gives the converse implication to Q_d. For GOOD, the first case uses the distance cutoff. The second uses Lemma E.1. In the third case, existence of the supplied controller implies J, and the diagonal cutoff supplies the reverse reduction.

The arguments passed to H_* and D_* never exceed T_*. Hence the output parameters are bounded by P_*. To compare a binary parameter with a threshold, first compare their significant bit lengths. Only a parameter with sufficiently few bits needs to be stored. □

Lemma E.3 (Scanning a witness row)**.** *For parameters at most P_*, the bounded witness graph can be refined by scan vertices so that reachability between its original vertices is preserved. A refined vertex uses $O_K(k+\log s)$ bits. For fixed K, k, its adjacency relation is decidable in logarithmic space.*

Proof. An original vertex stores a case, a witness state, and constantly many counters. It uses $O_K(k+\log s)$ bits. The colours of a whole row can be longer. We scan them.

First guess the next vertex v' and the row skeleton σ. The skeleton records each leaf's input letter, selected edges, and node marks. Check all conditions independent of colours: edge legality and destinations, the backward use-bit equations, termination, node and pair phases, fork coverage, and counter updates for G and E. In particular, check

$$G'_\lambda = \min\{H' + 1, G_\lambda + [\eta'^A_\lambda \neq \eta'^B_\lambda]\}$$

on each active input row. The accepting-state and DONE restrictions are checked here too.

It remains to test the endpoint conditions, birth flags, and rank updates. Fix a leaf, one of its two clones, and a destination register Y. Start the three-state marking automaton at $f'(Y)_1$ and scan $\mathrm{upd}(e)(Y)$. At a constant occurrence, guess one colour in $\{\mathsf{P}, \mathsf{M}, \mathsf{S}\}$ and apply the corresponding marking transition. At an old register occurrence X, require the current state to equal $f(X)_1$, then replace it by $f(X)_2$. At the end, require $f'(Y)_2$. These checks are exactly the endpoint equations for every intervening constant block.

Count a guessed colour only when the new use bit of Y is one. Keep a one-bit live M-count c_{M} for the current clone; reject a scan that exceeds one. At the clone's end require

$$\eta + c_{\mathsf{M}} = \eta'.$$

Also keep a signed live P-count Δ for the current leaf. Add occurrences from A and subtract those from B. After both clones, require

$$R'_\lambda = R_\lambda + \Delta.$$

Reset the counts before the next clone or leaf, as appropriate. Throughout the scan, $\Delta \in [-\mu, \mu]$.

A scan vertex stores v, v', σ, the leaf and clone indices, an expression pointer, the marking state, c_{M}, and Δ. After the final successful check,

connect it to v'. If no colours are needed, use a direct edge after checking the unchanged rank coordinates.

Each fresh occurrence appears in exactly one update expression. Thus different expression scans choose disjoint colour variables. A successful scan specifies a legal colouring of the entire row. Conversely, any legal row supplies one successful scan by revealing its colours in order.

The skeleton has $O_K(\log s)$ bits. Pointers and counts have $O(\log s)$ bits. Input scans recover sources, labels, destinations, update symbols, and final expressions. The backward use-bit law is checked by scanning all right-hand sides for each register. Fork coverage compares constantly many edge identifiers with the outgoing-edge count. These operations give the stated adjacency bound. □

Proof of Theorem 8.5(a)–(d),(f). Apply Lemma E.2. The resulting refined graph has polynomially many vertices when K, k are fixed. Its initial vertices have zero counters and the initial witness data. Its targets satisfy the usual accepting conditions, $R_\lambda = 0$, $G_\lambda = H' + 1$, and $E_\pi = d'$.

Lemma E.3 and Theorem 8.3(a) reduce each problem to non-reachability. The graph, with a new common source and target, is output by enumerating its vertex encodings and adjacent pairs. This is a logspace reduction. Non-reachability belongs to NL.

For (d), $Z_0 = O_{K,k}(s^{2N})$ gives

$$T_* \le p_{K,k}(s), \qquad \deg p_{K,k} = m(4N + 1).$$

The GAP monitor at this threshold has polynomial size. Corollary 8.4 gives memory $|Q_A|2^{q_{K,k}(s)}$.

For (f), one machine has no register. Its fresh-free final expressions produce ε. Word agreement forces every run of the other machine to do the same. Choose any legal edge at each state and letter. This controller has distance zero. □

Proof of Proposition 8.7 and Theorem 8.5(e). Use input alphabet V and output alphabet $\{a\}$. The reference has one state and one register X; every input letter applies $X := Xa$, and its final expression is X.

The deterministic machine B has states $V \cup \{\bot, T\}$, initially s, and one register Y, also used as every final expression. From $v \in V$ on $u \in V$, an edge of the graph leading to $u \neq t$ applies $Y := Ya$ and enters u. An edge leading to t applies $Y := aY$ and enters T. A non-edge applies $Y := Ya$ and enters $\bot$. The sink $\bot$ always appends; T always prepends.

Both machines are complete and copyless. Every step contributes one unary letter, so word agreement holds. The construction is output in logspace by scanning $V \times V$.

If t is unreachable, B always appends and the annotated outputs agree. If a word first reaches T at position i, its final births are

$$n, n-1, \dots, i, 1, 2, \dots, i-1.$$

At rank one, the reference birth is 1 and the other birth is n. Follow a shortest path from s to t, then append t^{H+1}. The resulting distance exceeds H. Since B is deterministic, this holds for every controller and every lookahead. Non-reachability is NL-complete, proving all the lower bounds. □

Proof of Corollary 8.8(a),(b), and the bounded-register consequence in (c). The same refined graph uses $O_K(k + \log s)$ bits per vertex. The threshold arithmetic uses that space as well. For input length n, monotonicity of f and $s \le n$ give

$$k + \log s = O(f(n)).$$

A bounded nondeterministic reachability search, followed by space complementation, proves (a).

Suppose one of the problems with $k = O(\log^c s)$ were PSPACE-hard. Every language in NSPACE($\log^{c+1} n$) would then reduce to it. A logspace reduction has polynomial output length. Recomputing its output bits as needed and applying (a) would use $O(\log^c n)$ space. This contradicts the nondeterministic space hierarchy theorem.

Theorem 8.5 settles bounded register count. Any PSPACE-hardness reduction at fixed K must therefore escape every bound $k = O(\log^c s)$. At constant k, hardness for a class $\mathcal{C}$ implies $\mathcal{C} \subseteq$ NL. □

E.4. **PSPACE lower bounds.**

Proof of Theorem 8.10. Write $D_j = (P_j, p_j^0, \delta_j, F_j)$, with disjoint state sets. We may assume a nonempty common input alphabet Λ. Put $P = cm + 1$. Both machines count their first P steps in states $0, \dots, P$, then remain in state P.

The reference has registers $Y_1, \dots, Y_m, Z$. At step $t \le cm$, it appends a to $Y_{\lceil t/c \rceil}$. At step P, it sets $Z := a$. All other entries are identities. After the preamble every update is the identity, and every final expression is $Y_1 \cdots Y_m Z$.

The registers of B are $X_{j,p}$ for $p \in P_j$, together with Z'. At step $t \le cm$, append a to $X_{\lceil t/c \rceil, p^0_{\lceil t/c \rceil}}$. At step P, set $Z' := a$. Afterward, on $\lambda \in \Lambda$, apply

$$X_{j,p} := \prod_{\substack{p' \in P_j \\ \delta_j(p',\lambda)=p}} X_{j,p'}, \qquad Z' := Z'.$$

Choose a fixed order for each product. Every final expression of B lists the rejecting registers in increasing j, then Z', then the accepting registers in increasing j. Within one DFA, use any fixed order.

Both machines have exactly one edge per state and input letter. Each old register occurs once among the right-hand sides of a simulation update, because each DFA transition function is total. Preamble updates are copyless, and all finals list every register once. Both outputs therefore equal $a^{\min(|w|,P)}$. This proves the promise.

The letters created for DFA j form one block. After a suffix u, that block occupies precisely $X_{j,\delta_j(p_j^0,u)}$. Induction on u proves this: the unique nonempty register moves to its transition target, and all other concatenated registers are empty. A full block has births $(j-1)c+1,\dots,jc$. The reference birth at rank r is r.

First suppose $|w| \geq P$. Let h be the number of accepting blocks after u. For a letter at offset o in block j, its displacement $\mathrm{rank}_B - \mathrm{birth}$ is

$$\begin{cases} -c\,\#\{i<j : i \text{ accepts}\}, & j \text{ rejects},\\ 1+c\,\#\{i>j : i \text{ rejects}\}, & j \text{ accepts}. \end{cases}$$

The letter in Z' has displacement $-ch$. Consequently,

$$\mathrm{od} = \max\left\{ch,\ \max_{j \text{ rejects}} c\,\#\{i<j : i \text{ accepts}\},\ \max_{j \text{ accepts}} \big(1+c\,\#\{i>j : i \text{ rejects}\}\big)\right\},$$

where empty maxima contribute zero. If $h=m$, this is cm. Otherwise each term is at most $c(m-1)+1$.

For $|w|<P$, the marker Z' is absent. Let $s_j \leq c$ be the current length of block j. Every block before a nonempty block is complete. A rejecting block's letters move left by the total length of earlier accepting blocks. An accepting block's letters move right by the total length of later rejecting blocks. Either total is at most $c(m-1)$. This also covers an unfinished preamble block.

All register indices, states, and updates can be enumerated with logarithmic-space counters. With explicit identity entries, the size is

$$O\left(cm|\Lambda|\left(\sum_j |P_j| + m\right)\right).$$

Set $H = c(m-1)+1$. Since $c \geq 2$, we have $H < cm$. An accepted intersection word u gives a violation on $\lambda_0^P u$, for any $\lambda_0 \in \Lambda$. If the intersection is empty, the computed bounds hold on every input. There is only one run. Lookahead and memory therefore leave the equivalence unchanged. □

Proof of Corollary 8.11. Take $c=2$. The threshold $H = 2m-1$ has a unary encoding of polynomial length. DFA intersection emptiness is PSPACE-complete, and Theorem 8.10 reduces it to $\mathrm{GOOD}(d,H)$ for every fixed d. Theorem 8.3 supplies membership.

On these deterministic instances,

$$\sup_w \mathrm{od}(\rho_A(w),\rho_B(w)) \geq 2m \iff \bigcap_j L(D_j) \neq \emptyset.$$

This proves the threshold and computation lower bounds in (b). The same calculation bounds the supremum by $2m$, so all these instances satisfy every Q_d and J. □

Proof of Theorem 8.12. Complete the input DFAs if necessary. Use alphabet $\Lambda \cup \{\mathtt{f}, \mathtt{W}\}$. The reference has registers T, S and states $\mathrm{init}, \mathrm{run}, \mathrm{aft}$. Its first input letter sets $T := a^m$ and enters run. There, letters of Λ do nothing, $\mathtt{W}$ appends a to S, and $\mathtt{f}$ enters aft without changing registers. Every letter in aft appends a to S. The final expression is always TS.

The machine B has registers $X_{j,p}$ and S'. Its first letter places one a in each X_{j,p_j^0} and enters run. In that state, a letter $\lambda \in \Lambda$ applies the DFA-routing update

$$X_{j,p} := \prod_{\delta_j(p',\lambda)=p} X_{j,p'}.$$

The letter $\mathtt{W}$ appends a to S'. On $\mathtt{f}$, there are m identity edges, leading to $\mathrm{aft}_1, \ldots, \mathrm{aft}_m$. In each such state, every letter appends a to S'. Entries not changed by these rules are identities.

Before the fork, the final expression lists every X-register, then S'. In aft_j, it lists all registers of the other DFAs, then the rejecting registers of D_j, then S', then the accepting registers of D_j. All lists use fixed orders.

Each DFA carries one token born at input step 1. After a prefix $x\,u$, with $u \in (\Lambda \cup \{\mathtt{W}\})^*$, token j is in

$$X_{j,\delta_j(p_j^0,\pi(u))},$$

where π erases $\mathtt{W}$. This follows from the routing update. The updates are copyless, all states are final, and every input letter is enabled. Both machines preserve their m initial tokens and create the same suffix letters. Their unary output words agree.

The only branching pair is $(\mathrm{run}, \mathtt{f})$, of degree m. After this choice no further fork is possible. Thus the ambiguity is at most m and $\kappa_B = m - 1$.

If the intersection is empty, a finite controller tracks all DFA states. At the fork it chooses a rejecting D_j. The final output then consists of the m step-1 tokens followed by the suffix, with exactly the reference births. Before any fork the same equality holds. Hence $\mathrm{GOOD}(0, 0)$ holds.

If u belongs to every DFA language, use

$$x\,u\,\mathtt{f}\,\mathtt{W}^{d+H+1}.$$

Every branch places its selected token last. At the final rank, its birth is 1, whereas the reference birth is the final input position. Their gap exceeds H. This defeats every controller, including controllers with arbitrary memory.

The explicit updates and final lists are computable in logspace. PSPACE-completeness of DFA intersection and closure under complement give the required reductions. The value $K = m$ can be supplied in unary. □

E.5. **Lookahead lower bounds.**

Proof of Proposition 8.14. The reference has registers T, S and states

$$s_0, c_1, \ldots, c_n, \mathrm{d}_a, \mathrm{d}_b.$$

The first letter sets $T := x$ and enters c_1. The next $n-1$ letters advance through the c-states without changing registers. At position $n+1$, letter e enters d_e and appends x to S. Every later letter also appends x to S. The final expression is TS in every state.

The machine B has registers P_1, P_2, S' and states

$$s_0, L_1, \dots, L_n, R_1, \dots, R_n, (L,a), (L,b), (R,a), (R,b).$$

On the first letter, one edge sets $P_1 := x$ and enters L_1; the other sets $P_2 := x$ and enters R_1. The two chains then advance without changing registers. At position $n+1$, letter e enters (L,e) or (R,e) and appends x to S'. Every later letter appends to S'. All unmentioned entries are identities. Before the decider, the final expression is P_1P_2S'. After it, use

$$\Phi_B(\cdot, a) = P_1S'P_2, \qquad \Phi_B(\cdot, b) = P_2S'P_1.$$

The machines are complete and copyless. Every nonempty input has output length $1+\max(0, |w|-n)$ in every run. There are at most two runs, determined at the first step. The displayed state counts are $n+3$ and $2n+5$. The alphabet and register count are constant, so the size is $O(n)$.

The token has birth 1. The suffix letters have births $n+1, \dots, |w|$. Branches (L,a) and (R,b) place the token first and give distance zero. The other two branches place it last, giving distance at least $|w|-1$.

At lookahead n, the first window either shows END or contains the decider. If END appears, there is no decider and either choice works. Otherwise choose L for a and R for b. This controller is memoryless and has distance zero.

Now fix $d < n$, H, and any controller. The words

$$w_a = a^n a\, a^{H+1}, \qquad w_b = a^n b\, a^{H+1}$$

have the same initial window a^{d+1}, with END hidden. The controller makes the same first choice on both. One choice is wrong, and its distance is at least $n+H+1 > H$. □

Proof of Theorem 8.13. Klein and Zimmermann construct the safety automata $\mathcal{A}'_n$ in their exponential-lookahead lower bound [KZ16]. Their input and output alphabets each have $n+1$ letters. Only the first output choice matters: it selects one of $n+1$ components. The automata have $O(n)$ states. Player O wins with initial input block 2^n+1 and loses with any initial block at most 2^n.

Adjoin a fresh input letter `w`. From each safe state it enters a safe sink $S_\top$. This preserves the winning delays. To extend a winning strategy, replace the first `w` and everything after it by a fixed ordinary input letter when consulting the old strategy. Before `w`, this gives a safe play; at `w`, the new automaton enters $S_\top$. For the converse, restrict inputs to the old alphabet.

The reference has one register S. It appends x on each ordinary input letter and does nothing on `w`. Its final expression is S.

For B, retain the safe states and add an absorbing state U for rejection. Use registers S_1, S_2, Z. A safe ordinary transition appends x to S_1. An ordinary transition causing rejection instead sets $Z := x$ and enters U. Inside U, ordinary letters append to S_2. In $S_\top$, they append to S_1. The letter $\mathtt{w}$ changes no register; it enters $S_\top$ from safe states and loops in U. All other register entries are identities. Set

$$\Phi_B(q) = S_1 \quad (q \neq U), \qquad \Phi_B(U) = S_1 S_2 Z.$$

For each safety-game output choice, use its induced transition. Identical transitions after the initial choice are identified.

Every ordinary input step creates one unary letter. If rejection occurs, its letter is retained in Z and emitted last. Thus every run has the reference output word, and $\mathbf{G}$ holds. The only possible fork occurs at the first letter and selects a component. There are at most $n+1$ runs. The construction has $O(n)$ states, $O(n^2)$ transitions, and four registers.

Fix d. Suppose O wins the extended safety game with initial input block $d+1$. On a finite input w of length r, use that strategy while END is hidden. These choices extend to an infinite safe play. After END becomes visible, take any legal edge. If U is entered at step t, then $r - t < d$. All changed rank pairings involve letters born in $[t, r]$: the letters following the entry token move one place left, and that token moves last. Hence $\mathrm{od} \leq r - t < d$. The fixed-resource theorem supplies finite memory.

Conversely, take any lookahead-d controller. Its choices on windows without END induce an O-strategy. If O loses at this delay, some consistent infinite play first rejects at a step t. That step reads an ordinary letter. Keep the first $t + d$ input letters of this play and append

$$\mathtt{w}^H c,$$

where c is ordinary. The controller's windows through step t are unchanged and contain no END. It therefore enters U at t. The last reference letter is born at $t + d + H + 1$, while the last letter of B is the token born at t. Their distance exceeds H.

Thus Q_d holds exactly at the winning delays of the safety game. The cited thresholds give failure for $d < 2^n$ and $\mathrm{GOOD}(2^n, 2^n)$.

For the one-way form, emit x on safe ordinary transitions, emit ε on entry into U, and compensate by a final x in U. Ordinary transitions inside U emit x; $\mathtt{w}$ emits ε. The lag-game reduction proved in Appendix F identifies its winning delays with the same safety game. State, ambiguity, and size bounds are unchanged. □

E.6. **Exponential controller memory.**

Proof of Theorem 8.15. The reference has states $\mathrm{pre}, \mathrm{aft}$ and registers $Y_1, \ldots, Y_n, S$. The machine B has states $\mathrm{pre}, \mathrm{aft}_0, \mathrm{aft}_1$ and registers $Z_{k,b}$, for $1 \leq k \leq n$ and $b \in \{0, 1\}$, together with S'. All registers start empty.

Each update below includes identity entries for registers whose contents are retained.

In pre, a bit b makes the reference apply

$$Y_1 := x, \qquad Y_{k+1} := Y_k \ (k < n), \qquad S := S.$$

On the same bit, B applies

$$Z_{1,b} := x, \quad Z_{1,1-b} := \varepsilon, \quad Z_{k+1,b'} := Z_{k,b'} \ (k < n,\ b' \in \{0,1\}), \quad S' := S'.$$

Both machines stay in pre. The oldest slot is discarded. The letter $\mathtt{t}$ is an identity loop there.

On $\mathtt{f}$, the reference takes an identity edge to aft. The machine B has two identity edges, leading to aft_0 and aft_1. Afterward, $\mathtt{t}$ appends x to S or S'. Every other letter gives an identity loop.

The reference final expression is always $Y_n \cdots Y_1 S$. Put

$$\mathcal{Z}_{<n} = Z_{n-1,0} Z_{n-1,1} \cdots Z_{1,0} Z_{1,1}.$$

The final expressions of B are

$$\Phi_B(\mathrm{pre}) = Z_{n,0} Z_{n,1} \mathcal{Z}_{<n} S', \qquad \Phi_B(\mathrm{aft}_c) = Z_{n,c} \mathcal{Z}_{<n} S' Z_{n,1-c}.$$

All shifts use each old register at most once. Every final expression lists each register once. The machines are complete, and only $(\mathrm{pre}, \mathtt{f})$ branches. After that fork, neither branch can return to pre.

Suppose the bit letters read before the fork occur at steps $t_1 < \cdots < t_L$, with values $b_1, \ldots, b_L$. Slot k is occupied precisely when $k \le \min(L, n)$. It then holds the letter

$$\ell_k = (x, t_{L+1-k}).$$

In the reference, that letter occupies Y_k. In B, it occupies $Z_{k,b_{L+1-k}}$, and the other register of the pair is empty. Induction on bit steps proves these facts. The padding letter leaves them unchanged. After the fork they remain fixed, while S, S' receive the same annotated suffix $\mathbf{s}$.

Both output words have length $\min(L, n) + |\mathbf{s}|$. Thus word agreement holds on every input. Before the fork, the annotated outputs agree. After choosing branch c, they still agree if slot n is empty or carries bit c. Otherwise all n slots are occupied, and the outputs are

$$\ell_n \ell_{n-1} \cdots \ell_1 \mathbf{s} \quad \text{and} \quad \ell_{n-1} \cdots \ell_1 \mathbf{s} \ell_n.$$

If $\mathbf{s}$ is nonempty, the last-rank birth gap is the final suffix birth minus the birth of ℓ_n.

For the upper bound, store $m = (m_1, \ldots, m_n) \in \{0,1\}^n$, initially 0^n. On a bit b in pre, replace it by $(b, m_1, \ldots, m_{n-1})$. On $\mathtt{t}$, leave it unchanged. At $\mathtt{f}$, choose aft_{m_n}. Every occupied slot has its bit in the corresponding memory coordinate. The selected branch therefore has distance zero. All 2^n memory values occur after suitable words of length n.

For the lower bound, fix d, H and a controller with fewer than 2^n reachable memory values. For $x \in \{0,1\}^n$, $1 \le j \le n$, and $r \ge 1$, consider

$$w(x, j, r) = x\, \mathtt{t}^d\, 0^{j-1} \mathtt{f}\, \mathtt{t}^r.$$

During the first n steps, END is hidden. Every window lies inside $x\mathtt{t}^d$. The controller's memory after these steps is consequently a function $F(x)$, independent of j, r. The machine state is always pre.

Choose distinct x, y with $F(x) = F(y)$, and choose j with $x_j \neq y_j$. Fix $r > H$. The two words now have identical suffixes after position n, identical lengths, and identical windows from that point onward. Starting from the same memory and machine state, the controller chooses the same branch c at their fork.

After $j - 1$ further bits, slot n contains the original bit at position j. Its birth is j. One of x_j, y_j differs from c. On that word, the final rank therefore has birth gap

$$(n + d + j + r) - j = n + d + r > H.$$

This proves the lower bound for every fixed lookahead.

Finally, the occurrence-size contributions are

alphabets and states	10
registers	$3n + 2$
reference edges and updates	$8n + 17$
edges and updates of B	$26n + 26$
final expressions	$8n + 5$.

The two empty update expressions contribute no occurrences. The sum is $45n + 60$. □

E.7. **Partial machines and accepting runs.**

Proof of Proposition 8.17 and Corollary 8.16. Under **G**, every word has a run and every run accepts. The completion B_z adds no outgoing edge at a reachable state of B, because all input letters are already enabled there. Hence its added sink is unreachable. Controllers, defined on reachable states, choose exactly the same edges in both models. Their objectives and run counts coincide. Every stated lower bound therefore transfers, with its original parameters and controllers. Applying this to Theorem 8.15 proves Corollary 8.16. □

Proof of Proposition 8.18. Put $N = 2^K$. An OD leaf stores its status, states in $Q_A \cup \{p_\perp\}$ and $Q_B \cup \{z\}$, a set $C_\lambda \subseteq Q_B$, use bits, endpoint maps, and two birth flags. Its logarithmic state count is at most

$$3 + \log_2(|Q_A| + 1) + \log_2(|Q_B| + 1) + |Q_B| + k \log_2 18 \leq 6s.$$

A DEAD leaf needs only its status, B-state, and set C_λ. It satisfies the same bound. Node and pair phases give

$$\log_2 Z \leq 6Ns + 2N^2.$$

The case $(\mathcal{T}, \varphi, \tau)$ is fixed when its automaton is built. Completion edges have identity updates, so the maximum number of fresh letters remains μ.

Thus $W \le \max(\mu, 1) \le s$ and $m \le 2N + \binom{N}{2}$. Proposition 6.7 gives

$$\log_2 T_*^\flat = O(N^3 s + N^4), \qquad \log_2 H_*^\flat(d) = O(N^3 s + N^4 + N^2 \log(d+1)).$$

For fixed d, H, form $\mathcal{G}_{d,H}^\flat$ from the flat witness automata. An OD leaf carries

$$R_\lambda \in [-\mu_B H, \mu_A H], \qquad G_\lambda \in [0, H+1].$$

DEAD leaves carry neither counter. Each $\pi \in \varphi$ carries $E_\pi \in [0, d]$. Edges follow witness rows, with saturation for G, E and exact updates for R. Sources have zero counters. Targets are accepting witness states with

$$R_\lambda = 0, \qquad G_\lambda = H + 1, \qquad E_\pi = d.$$

A path to a target gives an accepting walk satisfying every guard. For the reverse implication, use Lemma 6.2 to recover the leaf words, runs, and coherent branching data of a guarded walk. Each OD leaf lies in $\mathrm{dom}(B)$ and carries accepting runs. Their output words agree by $\mathbf{G}^\flat$. Replace the marked rank at that leaf by its least rank with gap exceeding H. The converse direction of the same lemma realizes the unchanged tree data with these new marks. Lemma 4.1 keeps every exact R_λ in its allowed interval. The DEAD data and coherence coordinates remain valid. Saturation therefore yields a path to a target.

The case and phase data use $O(N^2)$ bits. The local states use $O(Ns+N^2)$ bits, and the counters use

$$O\big(N \log s + N \log(H+2) + N^2 \log(d+2)\big).$$

A whole row fits in $O(Ns)$ bits. Besides the ordinary row conditions, check

$$C_\lambda = \mathrm{pre}_{a_\lambda}(C'_\lambda)$$

by scanning outgoing edges for each state of B. At termination, check $C_\lambda = F_B$ and the required final-state tests.

For the kill condition, compute the block of leaves still sharing the current history. For every enabled edge absent from that block's selected edges, find a leaf in the block whose successor co-accessibility set excludes its destination. The block takes $O(N)$ bits. These checks require only input pointers beyond the stored row and vertices. Consequently reachability uses

$$S_{d,H}^\flat = O\big(Ns + N^2 + N \log(H+2) + N^2 \log(d+2)\big)$$

nondeterministic space. Theorem 6.3 and space complementation give the same bound for $\mathrm{GOOD}^\flat$. Savitch's theorem gives its deterministic square.

Substituting the thresholds yields

$$Q_d^\flat \in \mathrm{NSPACE}\big(O(N^4 s + N^5 + N^3 \log(d+2))\big),$$

$$J^\flat \in \mathrm{NSPACE}\big(O(N^5 s + N^6)\big).$$

The factors N and N^2 count the gap and coherence counters. For fixed K, the latter space is $O(s)$, and its configuration graph has $2^{O(s)}$ vertices. For a known bound $K \le c \log_2 s$, all displayed quantities are polynomial. A

supplied unary K also gives these polynomial bounds when $K = O(\log s)$. Without that restriction, unary K gives EXPSPACE.

For (3), Theorem 8.10 supplies deterministic instances with $K = 1$ and unary $H = 2m - 1$. Proposition 8.17 transfers them. The upper bound just proved permits binary d, H.

For (4), apply Proposition 4.5 at the diagonal cutoff. The quotient strategy stores the reference state and a state of the distance monitor. At relevant terminals, the chosen B-run accepts and its word agrees with the reference. The complemented GAP subset automaton is therefore sufficient. Its memory bound is

$$(|Q_A| + 1)2^{|\mathcal{N}_{T_*^\flat}|} = 2^{2^{O(N^3 s + N^4)}}.$$

The same extraction at $(d, H_*^\flat(d))$ proves the assertion for $Q_d^\flat$. □

E.8. **Computing the ambiguity bound.**

Proof of Lemma 8.19. An edge sequence $e_1 \cdots e_r$ of B determines the state sequence $\iota, e_1, \ldots, e_r$ in $\mathcal{E}_B$. The first transition checks the initial source and first input letter. Each later transition checks consecutiveness and the next letter. The converse map reads the successive edge-state names. These maps are inverse.

For nonempty inputs, the last edge-state is final exactly when its destination belongs to F_B. For the empty input, ι is final exactly when q_0^B is final. Thus acceptance is preserved in both directions. Distinct parallel edges of B have distinct state names in $\mathcal{E}_B$. The state count is $|E_B| + 1$. □

Proof of Proposition 8.20. Trim $\mathcal{E}_B$ without changing its accepting runs. It is an ordinary NFA with at most $|E_B| + 1$ states.

Kiefer, Mazowiecki, Ryzhikov, and Wiśniewski decompose the accepting-run count into at most $5^{n/2}$ chain-automaton counts, each on at most n states, and bound each count by $2^{O(n)}$ [KMRW26]. Their constants are uniform over the input alphabet. Thus the degree is at most

$$5^{n/2} 2^{O(n)} = 2^{O(n)}, \qquad n \le |E_B| + 1.$$

Lemma 8.19 transfers this bound to B. Their result is an unrefereed preprint. Weber and Seidl's bound $2^{O(n \log n)}$ gives the stated alternative [WS91]. Since $|E_B| \le s$, both estimates have the required dependence on s. □

Proof of Lemma 8.21. Put $h = K' + 1$. Guess a common input word one letter at a time, while following h runs. Store their current states and an $h \times h$ Boolean matrix. The entry for a pair becomes one when that pair chooses different edge identifiers, and remains one thereafter.

On each letter, guess one enabled edge for each run. Reject if any required edge is absent. Accept at a stopping point exactly when every current state is final and every off-diagonal pair has been distinguished. Two runs on the same word differ exactly when some chosen edge differs. The procedure

therefore accepts precisely the required instances. Its initial configuration also handles the empty word.

The current states and guessed edges use $O(h \log s)$ bits. The matrix uses $O(h^2)$ bits. A path-length counter for the finite configuration graph uses the same order of space and makes the procedure halt. □

Proof of Proposition 8.22. First test whether $\mathrm{dom}(B)$ is empty. If it is, the objective is vacuous, and the exact ambiguity is zero. Otherwise test successive values $K' = 1, 2, \dots$ using Lemma 8.21 and deterministic reachability. Stop when no word has more than K' accepting runs. Finite ambiguity guarantees termination.

By Proposition 8.20, the loop stops at $K \le 2^{O(s)}$. Savitch's theorem bounds the space of each test by

$$O\big(((K'+1)\log s + (K'+1)^2)^2\big).$$

The loop therefore computes K in space $2^{O(s)}$. For the empty-domain case, use $K = 1$ in the subsequent bounds.

Set $N = 2^K \le 2^{2^{O(s)}}$. The algorithms in Proposition 8.18 apply for any valid supplied bound. Substituting this N into their space bounds gives

$$2^{2^{O(s)}} \operatorname{poly}(\log(d+2), \log(H+2)).$$

Squaring it preserves the deterministic 2-EXPSPACE bound. The preliminary computation of K uses less space.

The cutoff satisfies

$$\log_2 T_*^\flat = O(N^3 s + N^4) \le 2^{2^{O(s)}}.$$

Thus $T_*^\flat$ is bounded by a triple exponential. Substitution into the GAP-monitor memory bound gives

$$|\,\mathrm{Mem}\,| \le 2^{2^{2^{2^{O(s)}}}}.$$

Using Weber and Seidl's ambiguity bound instead replaces the innermost $O(s)$ by $O(s \log s)$ and leaves the space class unchanged [WS91].

Finally, the instances of Theorem 8.12 are finitely ambiguous. Omitting their supplied bound and applying Proposition 8.17 proves the PSPACE lower bounds. □

E.9. **Ambiguity growth.**

Proof of the slow-down construction used in Proposition 7.17. Fix an integer $k \ge 1$ and start from (A_*, B_M). For either machine X, replace its states by $Q_X \times \{0, \dots, k-1\}$. At residues below $k-1$, every letter takes one identity edge to the next residue. At residue $k-1$, simulate an original edge on the current letter and return to residue zero. Use $\Phi_X(q)$ at every state (q, r).

For $w = a_1 \cdots a_n$, put

$$w^{\downarrow} = a_k a_{2k} \cdots a_{k\lfloor n/k \rfloor}.$$

Deleting the forced steps gives a bijection between the new runs on w and the old runs on $w^{\downarrow}$. An old birth j becomes birth kj. Hence

$$\mathrm{od}_{\mathrm{new}}(w) = k\,\mathrm{od}_{\mathrm{old}}(w^{\downarrow})$$

for corresponding runs. Copylessness, completeness, and word agreement follow directly. The number of runs is at most $\lfloor |w|/k \rfloor + 1$. There remain at most two annotations, including exact agreement with the reference. The new reference depends only on k.

If the machine halts in L steps, simulate its lookahead-zero controller at the active positions. The residue is visible in the state, and only active positions update its memory. This gives distance at most $k(2L+1)$.

Conversely, suppose the slowed instance has a controller with lookahead d', arbitrary memory, and distance bound H'. Set $D = \lceil d'/k \rceil$ and encode an old input by

$$\mathrm{pad}(v) = (0^{k-1}v_1)\cdots(0^{k-1}v_{|v|}).$$

At position j of v, simulate the new controller throughout the jth block. Return its edge at position kj.

Every simulated window ends at a position at most $kj + d'$. Its non-padding letters therefore come from indices at most $j + \lfloor d'/k \rfloor \le j + D$. They are visible to the old controller. Store the previous input and the simulated memory to supply any earlier letters. If the old window shows END, the full remaining padded input is known. If it does not, $|v| \ge j + D$, so

$$k|v| \ge kj + d' \ge p + d'$$

for every simulated position p in the block. The simulated window then also hides END. Thus the simulation respects strict END.

The resulting old controller has lookahead D and distance at most H'/k. Theorem 3.4 forces the counter machine to halt. □

Proof of Proposition 3.6. Trim the edge-state NFA from Lemma 8.19. Weber and Seidl's finite-ambiguity criterion, also stated by Kiefer, Mazowiecki, Ryzhikov, and Wiśniewski, supplies states $p \neq q$ and a word v with paths

$$p \xrightarrow{v} p, \qquad p \xrightarrow{v} q, \qquad q \xrightarrow{v} q$$

whenever this NFA has unbounded ambiguity [WS91, KMRW26]. The word v is nonempty. Trimness supplies fixed paths from the initial state to p on u and from q to a final state on x.

For every $r \ge 1$, the word $uv^r x$ has at least r accepting runs. For each $i \in \{1, \ldots, r\}$, take $i-1$ loops at p, then the path to q, then $r-i$ loops at q. For $i < i'$, these runs occupy different states after uv^i. They are distinct.

Set $c_0 = |u| + |x|$ and $\ell = |v|$. Choosing $r = \lfloor (n - c_0)/\ell \rfloor$ proves the asserted lower bound; the case $r = 0$ is immediate. If $a_B(n) \le g(n)$ and $\liminf g(n)/n = 0$, this linear lower bound fails along a subsequence. Hence B is finitely ambiguous. Theorem 3.1 gives decidability without requiring g as input.

For the other direction, assume $g \geq 1$ and $\liminf g(n)/n > 0$. Choose a finite $c > 0$ below that liminf. There is n_2 such that $g(n) \geq cn/2$ for $n \geq n_2$. Choose an integer $k > \max(n_2, 4/c)$. For $n < k$,

$$\lfloor n/k \rfloor + 1 = 1 \leq g(n).$$

For $n \geq k$,

$$\lfloor n/k \rfloor + 1 \leq cn/4 + 1 \leq cn/2 \leq g(n).$$

The slow-down construction therefore places every reduced instance in the promised class. Its reference is fixed once g and k are fixed. Its two-annotation and exact offline-choice properties are preserved. It reduces halting to Q_d for every fixed d, and to J. Proposition 8.17 gives the flat-promise reductions.

Enumerating H decides the positive instances of Q_d by the fixed-resource theorem. Enumerating (d, H) does the same for J. This proves RE membership and completes the two classifications. □

E.10. **Delegators.**

Proof of Corollary 3.9. Give B no registers and empty final expressions. Take a one-state complete reference with empty output. This satisfies $\mathbf{G}^\flat$, and every pair of accepting outputs has distance zero. The controller's only obligation is to accept $\mathrm{dom}(B)$.

Theorem 3.1 gives decidability and extraction. Propositions 8.18 and 8.22 give PSPACE at fixed K, EXPSPACE with unary K, and 2-EXPSPACE when K is omitted. For a fixed lookahead, the quotient construction of Proposition 4.5 uses a one-state output monitor. Thus the controller can be memoryless in the current B-state and window, as in the delegator definition of [LR13], with the correction in [Lö23].

A sharper size count gives the finite-ambiguity lookahead estimate. Write $n = |Q_B|$ and $N = 2^K$. An OD leaf would require a marked output occurrence, so every case with an accepting walk has DEAD leaves only. There are at most $\binom{N}{2}$ coordinates, all coherence coordinates, and $W = 1$. Each leaf stores a status, a state of B_z, and a subset of Q_B. Hence

$$\log_2 Z \leq N(1 + \log_2(n+1) + n) + N + 2\binom{N}{2} \leq N(2n+3) + N^2.$$

The cutoff formula, with $m \leq \binom{N}{2}$, gives

$$\log_2 T \leq m \log_2 m + m(2 \log_2 Z + 1) + 1 = O(8^K n + 16^K).$$

If no accepting case has a coordinate, the objective is constant in the resource bounds and we may use $T = 1$. □

Lemma E.4 (Delegators as safety delay games)**.** *Let B be an NFA, with arbitrary ambiguity, and let B_z be its completion. There is a deterministic safety automaton $\mathcal{A}$ with*

$$M = (|Q_B| + 1)2^{|Q_B|} + 2$$

states such that a lookahead-d controller accepts $\mathrm{dom}(B)$ *if and only if O wins its constant-delay game with initial input block* $d+1$*. Consequently bounded-lookahead delegation is decidable in time*

$$2^{\mathrm{poly}(s)\,2^{O(|Q_B|)}},$$

and a successful lookahead can be chosen at most $2^{2M^2+1}-1$*.*

Proof. Use input alphabet $\Sigma \cup \{\dashv\}$ and output alphabet $E_{B_z} \cup \{-\}$. The states are

$$((Q_B \cup \{z\}) \times 2^{Q_B}) \cup \{\top, \bot\},$$

initially $(q_0^B, \{q_0^B\})$. Both added states are absorbing, and only $\bot$ rejects. For $a \in \Sigma$, an output edge e leads from (q, S) to

$$(\mathrm{dst}(e), \delta_B(S, a))$$

if $e \in \Delta_{B_z}(q, a)$, and to $\bot$ otherwise. On $\dashv$, independently of the output symbol, enter $\bot$ exactly when

$$S \cap F_B \neq \emptyset \quad \text{and} \quad q \notin F_B;$$

otherwise enter $\top$.

Let a lookahead-d controller accept $\mathrm{dom}(B)$. Before the first $\dashv$, O simulates it using the visible window. If the marker is absent from that window, consult the controller on the finite prefix ending at the window's last letter. At all positions used so far, strict END is hidden in this prefix exactly as in the infinite play. If the marker is visible, consult it on the word ending immediately before that marker. These simulations agree on all earlier choices because their earlier windows agree. At and after the marker, output $-$.

Every chosen edge is legal. If no marker occurs, no rejecting state is reached. If the marker follows w, then S records all states reachable by B on w. Whenever S meets F_B, the controller's chosen state also belongs to F_B. The marker transition is therefore safe. The empty word satisfies the same test.

Conversely, let O have a winning strategy. On finite input w, simulate it on $w \dashv^\omega$. At position i, the input history and current window determine its first $i+d$ letters: pad by $\dashv$ once the finite input ends. When END is hidden, $|w| \geq i+d$, so no padding occurs inside that prefix. The strategy chooses a legal edge, since an illegal edge would immediately lose. At the first marker, safety forces the chosen run to accept whenever $w \in \mathrm{dom}(B)$. Store the input history to perform this simulation. Define arbitrary legal choices on inconsistent memory configurations to make the controller total. The fixed-resource theorem replaces this controller by finite memory.

Klein and Zimmermann prove that winning with some delay for a deterministic parity automaton implies winning with bounded constant delay [KZ16]. Applied to this safety automaton with two priorities, their

bound is $f(0) \le 2^{2M^2+1}$. Their algorithm takes time exponential in a polynomial of the automaton size and alphabet size. Here the latter is

$$(|\Sigma| + 1)(|E_{B_z}| + 1) = \mathrm{poly}(s).$$

Substitution gives the stated time and lookahead bounds. □

Proof of the bounds collected in Theorem 3.7 and Proposition 3.8. For supplied d, H, Theorem 4.4 gives decidability at arbitrary ambiguity. For finite ambiguity under $\mathbf{G}$, use Theorems 8.3 and 8.5. For $\mathbf{G}^\flat$, use Propositions 8.18 and 8.22. Theorem 8.10 supplies fixed-K PSPACE-hardness when H is supplied, even in unary and with d fixed. Theorem 8.12 supplies PSPACE-hardness with unary K. Proposition 8.7 supplies the NL lower bounds, including fixed d, H. Proposition 8.17 preserves all these reductions.

At linear ambiguity, Theorem 3.4 and the fixed-resource enumeration give RE-completeness. The one-way entries follow from Proposition 9.1, proved in Appendix F. The controller bounds follow from Corollary 8.4 and Proposition 8.18.

Finally, Proposition 8.14, Theorem 8.13, and Theorem 8.15 give the three resource lower bounds. Their constructions satisfy $\mathbf{G}$, so Proposition 8.17 also proves every flat-promise assertion. □

Appendix F. The one-way case

We prove Proposition 9.1, the statements on the one-register fragment summarised in Proposition 3.10, and Proposition 9.2.

F.1. **Lag and pending letters.** Throughout, (A, B) is a one-way instance of Section 9: A is deterministic and complete, B is total, every state of either machine accepts, and for every input u and every run ρ of B on u the output of ρ followed by the final word of its last state is $A(u)$. We write $f_A(p)$ and $f_B(q)$ for the final words and let M bound their lengths. The fixed-d argument below may be folklore. The fragment $\mathcal{F}_1$ and both hardness proofs reuse its lemmas.

Fix $u = u_1 \cdots u_n$ and a run ρ of B on u. Let $p_0, \dots, p_n$ be the states of A and $q_0, \dots, q_n$ those of ρ, and let α_t and β_t be the words output at step t. Put $a(t) = |\alpha_1 \cdots \alpha_t|$ and $b(t) = |\beta_1 \cdots \beta_t|$ for $t \le n$, and $a(n+1) = b(n+1) = |A(u)|$, since step $n+1$ appends the final words. For an output position $x \in [1, |A(u)|]$ let $\mathrm{orig}_A(x) = \min\{t : a(t) \ge x\}$ and $\mathrm{orig}_\rho(x) = \min\{t : b(t) \ge x\}$. Then $\mathrm{od}(u, \rho) = \max_x |\mathrm{orig}_A(x) - \mathrm{orig}_\rho(x)|$, with $\max \emptyset = 0$.

Lemma F.1 (bounded lag)**.** *Put $\lambda(p, q) = |f_B(q)| - |f_A(p)|$. For every run ρ and every $t \le n$, $a(t) - b(t) = \lambda(p_t, q_t)$. So $|a(t) - b(t)| \le M$, and $\lambda(p_0, q_0) = 0$.*

Proof. The prefix $u_1 \cdots u_t$ is an input. A reaches p_t on it, and the prefix of ρ is a run on it ending in q_t, because every state accepts. The promise gives

$\alpha_1 \cdots \alpha_t \, f_A(p_t) = A(u_1 \cdots u_t) = \beta_1 \cdots \beta_t \, f_B(q_t)$. Compare lengths. At $t = 0$ both sides are $A(\varepsilon)$. □

The same equation shows that (p, q) determines the pending word: if $\lambda > 0$ then $w \, f_A(p) = f_B(q)$ for a word w of length λ. We only use lengths.

Lemma F.2 (origin distance is time pending)**.** *Call $x \in [1, |A(u)|]$ pending after step t if $\min(a(t), b(t)) < x \le \max(a(t), b(t))$. Then*

$$|\mathrm{orig}_A(x) - \mathrm{orig}_\rho(x)| = \#\{t \in [0, n] : x \text{ is pending after step } t\}.$$

Proof. a is nondecreasing with $a(0) = 0$ and $a(n+1) = |A(u)|$, so for $t \in [0, n+1]$ we have $x \le a(t)$ iff $t \ge \mathrm{orig}_A(x)$. The same holds for b and orig_ρ. So x is pending after t exactly when one of the two inequalities holds and the other fails, that is, for t in $[\min, \max - 1] \subseteq [1, n]$, where min and max are the two origins. □

The pending positions after a step form an interval of at most M positions, and they are served from the low end. The *leader* is A when $\lambda > 0$ and B when $\lambda < 0$; the other machine is the *follower*.

Round $t+1$ reads u_{t+1} from (p_t, q_t) and outputs α_{t+1} and β_{t+1}. It is an *E-round* if $\lambda(p_t, q_t) = 0$, or $\lambda > 0$ and $\beta_{t+1} \neq \varepsilon$, or $\lambda < 0$ and $\alpha_{t+1} \neq \varepsilon$: the follower speaks. This depends only on p_t, q_t, u_{t+1} and the chosen transition τ of B, and we write $E(p_t, q_t, u_{t+1}, \tau)$ for the predicate. Round 1 is an E-round.

Lemma F.3 (service)**.** *Let x be pending after step s and put $k = x - \min(a(s), b(s)) \in [1, M]$. If rounds $s+1, \ldots, s'$ with $s' \le n$ contain at least k E-rounds, then x is not pending after s'.*

Proof. Say A leads at s. Then $x \le a(s) \le a(r)$ for all $r \ge s$, so x is pending after r exactly when $b(r) < x$. While x is pending, $\lambda > 0$, so each E-round at such a time has $\beta \neq \varepsilon$ and raises b by at least 1. From $b(s) = x - k$, after k such rounds $b \ge x$. The case where B leads is symmetric. □

Lemma F.4 (stuck)**.** *Let $0 \le t \le t' \le n$ with $\lambda(p_t, q_t) \neq 0$, and let rounds $t+1, \ldots, t'$ all be non-E. Then $x = \min(a(t), b(t)) + 1$ is pending after each of the steps $t, \ldots, t'$, so $|\mathrm{orig}_A(x) - \mathrm{orig}_\rho(x)| \ge t' - t + 1$.*

Proof. Say $\lambda(p_t, q_t) > 0$. Then $b(t) < x \le a(t)$, so x is pending after t. In a non-E round with $\lambda > 0$ the follower B outputs ε, so b stays constant while a does not decrease. Hence λ stays positive and x stays pending. The case $\lambda < 0$ is symmetric. Lemma F.2 counts the steps. □

F.2. **The Büchi arena.** Positions of G_d are $V_d = Q_A \times Q_B \times \Sigma^{d+1}$. In (p, q, w), w is the current letter followed by d letters of lookahead. A round from (p, q, w) has three parts.

(1) Controller chooses $\tau = (q, w_1, \beta, q') \in \Delta_B$. One exists by totality.
(2) The round is an E-round iff $E(p, q, w_1, \tau)$ holds.
(3) Environment chooses $c \in \Sigma$. The next position is $(\delta_A(p, w_1), q', w_2 \cdots w_{d+1} c)$.

Environment also chooses the initial $w \in \Sigma^{d+1}$; the play starts at (p_0, q_0, w). Controller wins a play iff it has infinitely many E-rounds. A play is an infinite input with a B-run chosen with lookahead d. Controller's past choices are functions of the input read so far, so a history is determined by that input. Hence Controller strategies in G_d are exactly the $\dashv$-free lookahead-d controllers on infinite inputs.

Fixpoint. For $X \subseteq V_d$ let $Y_0(X)$ be the set of $v \in X$ with some τ such that $E(v, \tau)$ holds and every successor of (v, τ) lies in X. Let $Y_{j+1}(X) = Y_j(X) \cup \{v \in X : \exists \tau$ with every successor in $Y_j(X)\}$, $\mathrm{Attr}(X) = \bigcup_j Y_j(X)$, and let W_d be the greatest X with $\mathrm{Attr}(X) = X$.

Lemma F.5 (soundness of the fixpoint)**.** *Define σ on W_d: at rank 0, in $Y_0(W_d)$, play an E-move with all successors in W_d; at rank $j > 0$, in $Y_j(W_d) \setminus Y_{j-1}(W_d)$, play a move with all successors in $Y_{j-1}(W_d)$. Under σ the play stays in W_d, and from any position of W_d an E-round occurs within $|W_d| \leq |V_d|$ rounds.*

Proof. $W_d = \mathrm{Attr}(W_d)$, so every position of W_d has a rank and the prescribed moves exist. Their successors lie in W_d. From rank $j > 0$ the rank strictly drops, and at rank 0 an E-round occurs. The chain $Y_0 \subseteq Y_1 \subseteq \cdots$ grows strictly until it stabilises, so ranks are below $|W_d|$. □

By the standard solution of Büchi games, Controller wins from v iff $v \in W_d$. We use only this completeness direction. Controller wins G_d iff every initial position (p_0, q_0, w) lies in W_d.

F.3. **Fixed lookahead.**

Proof of Proposition 9.1, the fixed-d equivalence. Fix d and put $H_d = M|V_d| + d$. We show that these are equivalent: (a) some controller with lookahead d and arbitrary memory keeps od bounded, that is, Q_d^∞; (b) Controller wins G_d; (c) $\mathrm{GOOD}(d, H_d)$ holds with a controller of memory $Q_A \times Q_B$.

(c)⇒(a) is trivial.

(a)⇒(b). Let $\mathcal{C}$ and H witness (a), and let $\sigma_{\mathcal{C}}$ be the $\dashv$-free behaviour of $\mathcal{C}$, viewed as a G_d strategy. Suppose some play α consistent with $\sigma_{\mathcal{C}}$ has finitely many E-rounds, with last E-round $t_0 \geq 1$. Put $n = t_0 + H + d + 1$ and let u be the first n input letters of α. For $i \leq n - d$ the window at i lies inside u and $\dashv$ is hidden, so the choices of $\mathcal{C}$ on u agree with $\sigma_{\mathcal{C}}$ on α through step $n - d = t_0 + H + 1$. In particular the states after step t_0 agree. Round $t_0 + 1$ of α is non-E, so $\lambda(p_{t_0}, q_{t_0}) \neq 0$. Rounds $t_0 + 1, \ldots, t_0 + H$ are non-E in $\rho_{\mathcal{C}}(u)$ as well, and $t_0 + H \leq n$. Lemma F.4 with $t = t_0$ and $t' = t_0 + H$ gives a position with origin distance at least $H + 1$. The steps counted lie before any $\dashv$-phase choice. This contradicts the choice of H. With $H = 0$ the lemma is applied with $t' = t$, and the choice of n keeps round $t_0 + 1$ inside u.

(b)⇒(c). All initial positions lie in W_d. Let σ be the positional strategy of Lemma F.5 and define $\mathcal{C}_\sigma$ with memory the current pair (p, q). At position i, if $\dashv$ is hidden and $(p, q, u_i \cdots u_{i+d}) \in W_d$, play $\sigma(p, q, u_i \cdots u_{i+d})$. Otherwise play a fixed transition of $\Delta_B(q, u_i)$. In both cases update (p, q). So χ is total. The memory Q_A would suffice, since χ receives q; we keep $Q_A \times Q_B$ as stated.

Take u with $|u| = n$. Positions $1, \ldots, n - d$ produce a σ-consistent play prefix inside W_d, so every $|V_d|$ consecutive rounds inside $[1, n-d]$ contain an E-round. Let x be an output position. If its two origins agree, its distance is 0; this covers every x when $M = 0$. Otherwise let $s \geq 1$ be the smaller origin, the step after which x first becomes pending (Lemma F.2), and $k \in [1, M]$ as in Lemma F.3. Three cases.

- $s + k|V_d| \leq n - d$. By Lemma F.3, x is served by step $s + k|V_d|$, so its distance is at most $M|V_d|$.
- $s \leq n - d < s + k|V_d|$. Then x is served at step $n + 1$, where $a(n+1) = b(n+1)$, so its distance is at most $n+1-s < M|V_d|+d+1$.
- $s > n - d$, which includes every x when $n < d$. Its distance is at most $n + 1 - s \leq d$.

So od $\leq H_d$. For fixed d, the nested fixpoint runs in time $O(|V_d|^2\,|\Delta_B|\,|\Sigma|)$. □

F.4. **All lookaheads at once.** *Monotonicity.* If Controller wins G_d and $d' \geq d$, she wins $G_{d'}$: at (p, q, w') play $\sigma_d(p, q, w'_1 \cdots w'_{d+1})$. The projection of a $G_{d'}$-play is a σ_d-consistent G_d-play with the same E-rounds.

Delay games. Klein and Zimmermann [KZ16] study games $\Gamma_f(L)$ in which Player I writes an ω-word over Σ_I and Player O an ω-word over Σ_O, O's i-th letter being chosen once $f(0) + i - 1$ letters of I are known, for a constant delay function f. O wins iff the pair lies in L. With $f(0) = d + 1$, O's i-th letter sees $i + d$ input letters: this is lookahead d.

Let $\mathcal{A}$ be the deterministic parity automaton over $\Sigma \times \Delta_B$ with states $Q_A \times Q_B \times \{0, 1\} \cup \{\bot\}$ and initial state $(p_0, q_0, 0)$. On (c, τ) from $(p, q, \cdot)$: if $\tau = (q, c, \beta, q') \in \Delta_B$, go to $(\delta_A(p, c), q', [E(p, q, c, \tau)])$; otherwise go to the sink $\bot$. The colour of $(\cdot, \cdot, 1)$ is 2, that of $(\cdot, \cdot, 0)$ and of $\bot$ is 1, with max-parity. So $|\mathcal{A}| = 2n_A n_B + 1$ with two colours, where $n_A = |Q_A|$ and $n_B = |Q_B|$, and $L_p(\mathcal{A})$ is the set of pairs (α, β) in which β is a B-run on α with infinitely many E-rounds. O-strategies are maps $\Sigma^* \to \Delta_B$, the same objects as G_d strategies, and an illegal τ leads to the losing sink. Hence Controller wins G_d iff O wins $\Gamma_f(L_p(\mathcal{A}))$ for the constant f with $f(0) = d+1$.

Proof of Proposition 9.1, the bound d_0 and EXPTIME membership. Put $K_{\mathrm{KZ}} = 2^{2(2n_A n_B+1)^2+1}$ and $d_0 = K_{\mathrm{KZ}} - 1$. Suppose J holds. By the fixed-d equivalence, Controller wins some G_d, so O wins $\Gamma_f(L_p(\mathcal{A}))$ for some constant f. Klein and Zimmermann [KZ16, Thm 4.8] then give a constant f with $f(0) \leq 2^{|\mathcal{A}|^2 k+1} = K_{\mathrm{KZ}}$, where $k = 2$ is the number of colours.

So Controller wins $G_{f(0)-1}$, hence G_{d_0} by monotonicity, and the fixed-d equivalence gives $\mathrm{GOOD}(d_0, H_{d_0})$. The converse implications are trivial.

For membership, J holds iff O wins $\Gamma_f(L_p(\mathcal{A}))$ for some f, which [KZ16, Thm 4.4] decides in exponential time; $\mathcal{A}$ is computed in polynomial time from the instance. Solving G_{d_0} directly also decides J, in doubly exponential time. For Q_d with d in unary, $|V_d| = n_A n_B |\Sigma|^{d+1}$ is exponential in the input, and the fixed-d procedure runs in time polynomial in $|V_d|$. □

Decidability needs only some computable bound on $f(0)$, so the earlier bound of Holtmann, Kaiser and Thomas [HKT10] also suffices. The exponential-time bound uses [KZ16, Thm 4.4].

F.5. **Infinite plays.** Fix an infinite input u and a run ρ of B on it, and define $a(t)$, $b(t)$ and pending positions for all $t \in \mathbb{N}$ as in Section F.1. The pending period of a position is an interval by Lemma F.2, so the *age* of a pending x at step t is $t - \min(\mathrm{orig}_A(x), \mathrm{orig}_\rho(x)) + 1$. Let $w(t)$ be the largest age of a pending position at t, and 0 if none is pending.

Proposition F.6 (infinite plays)**.** *For lookahead-d controllers on infinite inputs, these are equivalent:*

(i) some controller serves every pending position eventually, on every play;
(ii) some controller and some b satisfy $w(t) \le b$ for all t, on every play;
(iii) some controller keeps w bounded on every play, with a bound depending on the play;
(iv) Controller wins G_d.

With Section F.3, each is equivalent to Q_d.

Each equivalence is between existence statements. The step from (iv) to (ii) replaces the given strategy by the positional one of Lemma F.5; for one fixed strategy, (i) does not give (ii) (Example F.9).

Proof. (i)⇒(iv). If a play has a last E-round t_0, then $\lambda(p_{t_0}, q_{t_0}) \neq 0$, and Lemma F.4 for every $t' \ge t_0$ keeps one position pending forever, against (i).

(iv)⇒(ii). Use the strategy σ of Lemma F.5. Every $|V_d|$ consecutive rounds contain an E-round, so by Lemma F.3 every position is served within $k|V_d| \le M|V_d|$ rounds of becoming pending. So $w(t) \le M|V_d|$ on every play.

(ii)⇒(iii) is trivial. (iii)⇒(i): a position pending after every step $t \ge s$ has age at least $t - s + 1$ at t, so w is unbounded on that play. □

Final words create no waiting on infinite plays. On finite inputs they flush every pending position at step $n + 1$, at a cost covered by the d $\dashv$-phase steps in H_d. The only place where $\dashv$ and finite length matter is (a)⇒(b) in Section F.3, where strict $\dashv$ makes the first $n - d$ choices $\dashv$-free. With non-strict $\dashv$ there are $n - d - 1$ such steps and H_d shifts by 1.

F.6. **Three examples.** All three use $\Gamma = \{x\}$ and a one-state A with $f_A = \varepsilon$.

Example F.7 (no lookahead works, though every input has a run of distance at most 1)**.** $\Sigma = \{a, b, c, e\}$; A maps $a, e \mapsto x$ and $b, c \mapsto \varepsilon$. B has states N, D_b, D_c with $f_B(N) = \varepsilon$ and $f_B(D_b) = f_B(D_c) = x$, so $M = 1$. From N: a/ε and e/ε go to D_b or D_c, and b/ε, c/ε go to N. From D_b: $b/x \to N$, $c/\varepsilon \to D_b$, $e/x \to D_b$, and a/x to D_b or D_c; D_c is symmetric with b, c swapped. Fix d and a controller. On inputs $a\, e^k \cdots$ with $k \geq d$ the controller commits at the initial a to D_b or D_c after seeing $a\, e^d$, and it stays there through the e's. Environment chooses the tail afterwards: c^ω against D_b and b^ω against D_c. The committed state never pays on the tail letter, so one position stays pending forever and G_d is lost. Offline, the run committing to D_c on $a\, e^k c^\omega$ pays at every e and at the first c, with distance 1. The padding e^k hides which tail will come.

Example F.8 (why w is defined per step)**.** $\Sigma = \{a, b, c\}$; A maps $a \mapsto x$ and $b, c \mapsto \varepsilon$. B has states N, S with $f = \varepsilon$ and D_b, D_c with $f = x$. From N: $a/\varepsilon \to D_b$ or D_c; $b, c/\varepsilon \to N$. From D_b: $b/x \to S$; $c/\varepsilon \to D_b$; $a/xx \to S$; D_c symmetric. From S all letters loop, with a/x and $b, c/\varepsilon$. With $d = 0$, G_0 is lost: ab^k against ac^k gives distance $k + 1$ for one of them. Yet on every infinite play at most one position waits long, so a per-position condition such as "all but finitely many positions wait at most b steps" holds for every strategy. Under the per-step $w(t)$ the stranded position makes w unbounded, and Proposition F.6 holds.

Example F.9 (for one strategy, (i) does not give (ii))**.** $\Sigma = \{a, b\}$; A maps $a \mapsto x$ and $b \mapsto \varepsilon$. B has states N with $f = \varepsilon$ and D with $f = x$, and transitions $N \xrightarrow{a/\varepsilon} D$, $N \xrightarrow{b/\varepsilon} N$, $D \xrightarrow{a/xx} N$, $D \xrightarrow{b/x} N$ and $D \xrightarrow{b/\varepsilon} D$. Here $\lambda(N) = 0$ and $\lambda(D) = 1$, and each transition keeps the output counts consistent with these lags, so the promise holds. Consider the strategy that, on its r-th visit to D, pays on the r-th b, or at once when an a arrives. Every pending position is served eventually on every input, so (i) holds for it. On $(ab)(ab^2)(ab^3)\cdots$ the r-th visit waits r steps, so w is unbounded and (ii), (iii) fail for this strategy. The positional strategy that pays at the first b ensures (ii).

F.7. **EXPTIME-hardness.**

Proof of Proposition 9.1, EXPTIME-hardness. Let $\mathcal{A}$ be a complete deterministic safety automaton over $\Sigma_I \times \Sigma_O$ with initial state q_I. We may assume that its rejecting states form one sink $\bot_{\mathcal{A}}$. If q_I is rejecting, map $\mathcal{A}$ to the fixed no-instance of Example F.7. Otherwise let $\Sigma = \Sigma_I \cup \{\mathtt{w}\}$ with $\mathtt{w}$ fresh and $\Gamma = \{x\}$.

- A has one state p, outputs x on every $c \in \Sigma_I$ and ε on $\mathtt{w}$, with $f_A = \varepsilon$.
- B has states $Q_{\mathcal{A}}^{\mathrm{acc}} \cup \{U, S_\top\}$, initial state q_I, $f_B(U) = x$ and $f_B = \varepsilon$ elsewhere. For accepting q, $c \in \Sigma_I$ and $b \in \Sigma_O$, with $q' =$

$\delta_{\mathcal{A}}(q,(c,b))$, it has (q,c,x,q') if q' is accepting and (q,c,ε,U) otherwise. It also has $(q,\mathtt{w},\varepsilon,S_\top)$ for accepting q. In U and in $S_\top$ each $c\in\Sigma_I$ outputs x and $\mathtt{w}$ outputs ε, both looping.

Δ_B is a set, so choices of b with the same target give one transition.

Promise. B is total and all its states accept. On every run the output is one x per Σ_I-letter: the entry into U outputs ε, and $f_B(U)=x$ repays it. So $B(u)=A(u)$ and $M=1$.

Rounds. $\lambda(p,q)=|f_B(q)|$. It is 0 in accepting states and in $S_\top$, so every round from those states is an E-round. In U the follower is B, and a round is an E-round iff its letter is in Σ_I.

Claim. Controller wins G_d iff O wins $\Gamma_{f_d}(L_\forall(\mathcal{A}'))$, where f_d is constant with $f_d(0)=d+1$ and $\mathcal{A}'$ extends $\mathcal{A}$ by sending $\mathtt{w}$ from any accepting state to an accepting sink. Controller strategies and O-strategies correspond as in Section F.4: an O-letter b at (q,c) maps to the transition with target $\delta_{\mathcal{A}}(q,(c,b))$, or to U, and a transition maps to any b inducing it. In U, in $S_\top$ and on $\mathtt{w}$ there is no choice. Along corresponding plays, B's state equals the state of $\mathcal{A}'$ while the latter accepts, and B is in U exactly when $\mathcal{A}'$ has rejected. If the strategy never reaches U, every round is an E-round and Controller wins. If some consistent play prefix reaches U, Environment continues with $\mathtt{w}^\omega$, which gives no further E-round, and Controller loses.

Adding $\mathtt{w}$ *is harmless.* For every f, O wins $\Gamma_f(L_\forall(\mathcal{A}'))$ iff O wins $\Gamma_f(L_\forall(\mathcal{A}))$. Given τ winning for $\mathcal{A}$, let $\tau'(y)=\tau(y')$, where y' replaces everything from the first $\mathtt{w}$ of y on by a fixed letter $c_0\in\Sigma_I$. In a τ'-play whose input has its first $\mathtt{w}$ at position j, the O-letters coincide with those of the τ-play on the prefix before j followed by c_0^ω, which is safe. So $\mathcal{A}'$ accepts before position j and moves to the safe sink there. Plays without $\mathtt{w}$ are τ-plays. The converse restricts to Σ_I^ω.

J. By the two previous subsections, J holds iff Controller wins some G_d, iff O wins $\Gamma_f(L_\forall(\mathcal{A}))$ for some constant f, iff O wins for some f at all, by [KZ16, Thm 4.8] applied to the safety automaton read as a parity automaton. The map $\mathcal{A}\mapsto(A,B)$ is polynomial: $|Q_B|=|Q_{\mathcal{A}}|+1$ and $|\Delta_B|\le|Q_{\mathcal{A}}||\Sigma_I||\Sigma_O|+O(|Q_{\mathcal{A}}|+|\Sigma_I|)$. Klein and Zimmermann [KZ16, Thm 4.3] proved the last problem EXPTIME-hard.

Q_d with unary d. Take the reduction in the proof of [KZ16, Thm 4.3], which maps a machine M and input x to a safety automaton with a polynomial p such that O wins with constant delay $f(0)=p(|x|)+3$ iff M rejects x. Put $d=p(|x|)+2$. By the claim and the fixed-d equivalence, Q_d holds iff M rejects x. Here d is polynomial in the instance and written in unary, and EXPTIME is closed under complement. □

This reduction uses unbounded ambiguity: B forks at every Σ_I-step where O has a real choice. No reduction with a constant ambiguity bound can work: at fixed K the problem is in NL (Section F.9), and NL $\subsetneq$ EXPTIME.

F.8. **The one-register fragment.** We now work under **G** with SST notation.

Definition F.10 (the fragment $\mathcal{F}_1$)**.** An instance (A, B) is in $\mathcal{F}_1$ if it satisfies **G** and

- each machine has exactly one register, X_A and X_B, written X when the machine is clear;
- every edge update is $X := Xu$ for some $u \in \Gamma^*$;
- $\Phi(q) \in \{X, \varepsilon\}$ for every state q of either machine.

Write $F_A^X = \{p : \Phi_A(p) = X\}$ and $F_B^X = \{q : \Phi_B(q) = X\}$. There is no ambiguity restriction on B.

The case $\Phi \equiv X$ is the one-way model with $f_A = f_B = \varepsilon$. Let $n_A = |Q_A|$, $n_B = |Q_B|$, $N = n_A n_B$, and let m be the largest $|u|$ over all edges. For a run on $w = w_1 \cdots w_n$ let α_t, β_t be the words appended at step t, with $a(t)$ and $b(t)$ their cumulative lengths as before. Letters appended at step t are born at t. The output on the prefix $w_{1..t}$ is $\alpha_1 \cdots \alpha_t$ if $p_t \in F_A^X$ and ε otherwise, and likewise for B. There is no final step.

Product graph. Its vertices are the pairs $(p, q) \in Q_A \times Q_B$, with an edge $(p, q) \to (\delta_A(p, c), \mathrm{dst}(e))$ for every $c \in \Sigma$ and $e \in \Delta_B(q, c)$. Let R be the set of pairs reachable from (p_0, q_0) and $E_{XX} = F_A^X \times F_B^X$. A pair in R is *live* if E_{XX} is reachable from it and *dead* otherwise. Successors of dead pairs are dead.

Lemma F.11 (lag on live pairs)**.** *There is a function λ from live pairs to $\mathbb{Z}$, computable in polynomial time, such that for every input w, every run ρ of B on w and every $t \le |w|$ with (p_t, q_t) live:*

(i) $a(t) - b(t) = \lambda(p_t, q_t)$*;*
(ii) $|\lambda(p_t, q_t)| \le M'$*, where* $M' = \max\{|\lambda(p, q)| : (p, q)$ *live*$\} \le (N-1)m$*;*
(iii) $\lambda(p_0, q_0) = 0$ *when* (p_0, q_0) *is live.*

Proof. Let (p, q) be live and fix a shortest product path from it to E_{XX}: an input v with $|v| \le N - 1$ and a B-path π from q on v. Let $\alpha_p(v)$ be the word A appends along v from p and β_π the word appended along π. Put $\lambda(p, q) = |\beta_\pi| - |\alpha_p(v)|$.

(i) Let w and ρ reach (p, q) at time t. Then $\rho\pi$ is a run of B on wv, and both machines end in E_{XX}, so both outputs are full registers. By (R4), $\alpha_1 \cdots \alpha_t\, \alpha_p(v) = \beta_1 \cdots \beta_t\, \beta_\pi$, and comparing lengths gives $a(t) - b(t) = \lambda(p, q)$. The right-hand side does not depend on (w, ρ), so λ is well defined. It does not depend on the choice of (v, π) either: two choices give the same left-hand side for one fixed reaching pair (w, ρ), which exists because live pairs lie in R. (ii) Both lengths lie in $[0, |v|m]$. (iii) is (i) at $t = 0$. For computability, find R by a forward search, liveness by a backward search from E_{XX}, and a shortest path from each live pair by one more search, summing output lengths along it. □

Lemma F.12 (where od is nonzero; forced exposure)**.** *(i) Let ρ be a run on w with $|w| = n$. If $(p_n, q_n) \notin E_{XX}$, then* $\mathrm{od}(\rho_A(w), \rho) = 0$*. If $(p_n, q_n) \in E_{XX}$, then $a(n) = b(n)$, every pair (p_t, q_t) with $t \le n$ is*

live, and for $x \in [1, a(n)]$,

$$|\mathrm{orig}_A(x) - \mathrm{orig}_\rho(x)| = \#\{t \in [0,n] : x \text{ is pending after step } t\},$$

with origins in $[1,n]$.

(ii) Let (p,q) *be live, reached at time* T *by some input* w *and run* ρ *with* $\max(a(T), b(T)) \geq 1$. *Then there is* v *with* $|v| \leq N-1$ *such that every run of* B *on* wv, *whatever it does on* w, *ends in* E_{XX} *with* $a(|wv|) = b(|wv|) \geq 1$.

Proof. (i) If $p_n \in F_A^X$ and $q_n \notin F_B^X$, then (R4) gives $\alpha_1 \cdots \alpha_n = \varepsilon$, so both outputs are empty and $\mathrm{od} = \max \emptyset = 0$. The symmetric case and the case with neither state in F^X are the same. If $(p_n, q_n) \in E_{XX}$, then (R4) gives $\alpha_1 \cdots \alpha_n = \beta_1 \cdots \beta_n$, so $a(n) = b(n)$, and every (p_t, q_t) reaches (p_n, q_n) along ρ, so it is live. For the formula, a is nondecreasing with $a(0) = 0$, and for $x \in [1, a(n)]$ and $t \in [0,n]$, $x \leq a(t)$ iff $t \geq \mathrm{orig}_A(x)$; likewise for b. The proof of Lemma F.2 applies. Every pending x satisfies $x \leq \max(a(t), b(t)) \leq a(n)$, so it is counted.

(ii) Take (v, π) as in the proof of Lemma F.11. Along $\rho\pi$ the final pair is in E_{XX}, and by (i) its common output length is at least $\max(a(T), b(T)) \geq 1$. A is deterministic, so A on wv ends in some $p' \in F_A^X$ with a nonempty register, whatever B does. Let ρ' be any run of B on wv. By (R4) its output is A's output, which is nonempty, and a B-output is nonempty only in a state of F_B^X. So ρ' ends in E_{XX}. □

Lemmas F.3 and F.4 hold unchanged on any stretch of live pairs, with M' for M: their proofs use only the values $a(t)$ and $b(t)$, which satisfy Lemma F.11 there, and the E-predicate.

The game G'_d. It is the arena G_d on $V_d = Q_A \times Q_B \times \Sigma^{d+1}$ with this predicate: a round from (p,q,w) with edge τ is an *E′-round* if (p,q) is dead or not in R, or (p,q) is live and $E(p,q,w_1,\tau)$ holds with λ from Lemma F.11: $\lambda = 0$, or $\lambda > 0$ and τ appends a nonempty word, or $\lambda < 0$ and A appends a nonempty word on w_1 at p. Controller wins iff the play has infinitely many E′-rounds. The fixpoint W'_d and Lemma F.5 are as before with E′ in place of E.

Theorem F.13 (the fragment $\mathcal{F}_1$)**.** *Let* $(A,B) \in \mathcal{F}_1$. *Put* $H'_d = M'|V_d| + d$, $K_{\mathrm{KZ}} = 2^{2(2n_An_B+1)^2+1}$ *and* $d_0 = K_{\mathrm{KZ}} - 1$.

(1) For each d *these are equivalent: (a)* Q_d^∞; *(b) Controller wins* G'_d; *(c)* $\mathrm{GOOD}(d, H'_d)$ *holds with a controller of memory* $Q_A \times Q_B$. *For fixed* d *this is decidable in time polynomial in* $|V_d|$ *and the instance size.*

(2) $J \iff Q_{d_0} \iff$ *Controller wins* G'_{d_0}.

(3) J *on* $\mathcal{F}_1$ *is in EXPTIME, and so is* Q_d *with* d *in unary.*

Proof. (c)⇒(a) is trivial.

(a)⇒(b). Let $\mathcal{C}$ and H witness (a), and let $\sigma_{\mathcal{C}}$ be its ⊣-free behaviour as a G'_d strategy. Suppose some $\sigma_{\mathcal{C}}$-play α has finitely many E′-rounds. Round

1 is E′: either (p_0, q_0) is dead, or it is live with $\lambda = 0$. Let $t_0 \geq 1$ be the last E′-round. Every later round is non-E′, so every pair (p_t, q_t) with $t \geq t_0$ is live with $\lambda \neq 0$, since a round from a dead pair or from a live pair with $\lambda = 0$ is E′. By Lemma F.4 on this live stretch, $x = \min(a(t_0), b(t_0)) + 1$ is pending after each of the steps $t_0, \ldots, t_0 + H$. Put $T = t_0 + H + 1 + d$. The pair (p_T, q_T) is live with $\lambda \neq 0$, so $\max(a(T), b(T)) \geq 1$, and Lemma F.12(ii) gives v such that every B-run on $u' = \alpha_1 \cdots \alpha_T v$ ends in E_{XX} with nonempty output. Positions $i \leq T - d = t_0 + H + 1$ of u' see windows inside $\alpha_1 \cdots \alpha_T$ with $\dashv$ hidden, so $\rho_{\mathcal{C}}(u')$ coincides with the play through step $t_0 + H + 1$, and x is pending after $t_0, \ldots, t_0 + H$ in $\rho_{\mathcal{C}}(u')$ as well. That run ends in E_{XX}, so Lemma F.12(i) gives od $\geq H + 1$. This contradicts the choice of H.

The hard step is the cut at T. In Section F.3 the obligation at the end of u was automatic, because every state exposes. Here the play is cut at T and extended by the exposing word v. The controller's run on u' need not pass through (p_T, q_T), since its choices after position $T - d$ may see $\dashv$. It does not have to: the pending witness lies in steps $t_0, \ldots, t_0 + H \leq T - d - 1$, where the run coincides with the play, and the final E_{XX} is forced for every B-run by the input alone.

(b)⇒(c). Let σ be the positional strategy of Lemma F.5 for G'_d and define $\mathcal{C}_\sigma$ with memory (p, q) as in Section F.3: play σ while $\dashv$ is hidden and the position is in W'_d, and a fixed legal edge otherwise. Take u with $|u| = n$ and let $\rho = \rho_{\mathcal{C}_\sigma}(u)$. If $(p_n, q_n) \notin E_{XX}$ then od $= 0$ by Lemma F.12(i). Otherwise every pair on ρ is live, so on ρ each E′-round is an E-round. Positions $1, \ldots, n - d$ form a σ-play prefix inside W'_d, so every $|V_d|$ consecutive rounds in $[1, n - d]$ contain an E-round. Take x with distinct origins, let s be the smaller one, so x is first pending after s, and let $k = x - \min(a(s), b(s)) \in [1, M']$. If $s + k|V_d| \leq n - d$, Lemma F.3 bounds the distance by $k|V_d| \leq M'|V_d|$. Otherwise x is no longer pending after n, because $a(n) = b(n)$, so the distance is at most $n - s < k|V_d| + d$. So od $\leq H'_d$. For fixed d, computing λ, R and liveness takes polynomial time by Lemma F.11, and the nested fixpoint takes $O(|V_d|^2 |\Delta_B||\Sigma|)$. The E′-predicate is true on dead pairs; this is safe because (b)⇒(c) uses E′ only on runs ending in E_{XX}, where every pair is live.

Parts 2 and 3 follow Section F.4. Monotonicity in d holds because the projection argument does not depend on the predicate. The parity automaton $\mathcal{A}$ is unchanged except that its colour bit is $[E'(p, q, c, \tau)]$, computable in polynomial time by Lemma F.11; so $|\mathcal{A}| = 2n_A n_B + 1$ with two colours, and [KZ16, Thm 4.8] gives the same d_0. [KZ16, Thm 4.4] gives EXPTIME for J. For unary d, G'_d is solved in time polynomial in $|V_d| = N|\Sigma|^{d+1}$. □

Resets $X := u$ are outside the fragment: output appended before a reset is never exposed. Two append-only registers are outside it as well, since Φ can permute registers and equal lengths then no longer align births. Under $\mathbf{G}^\flat$ nothing is proved: Lemma F.12(ii) uses input-enabledness and the all-final (R4) on every prefix. Proposition F.6 is not transferred.

Proposition F.14 (hardness inside $\mathcal{F}_1$)**.** *On $\mathcal{F}_1$, J is EXPTIME-complete, and so is Q_d with d in unary. The hard instances have $|\Gamma| = 1$, a two-state A and unbounded ambiguity.*

Proof. Membership is Theorem F.13(3). For hardness we adapt the reduction of Section F.7. That reduction has $f_B(U) = x$, a constant in a final expression, which $\mathcal{F}_1$ forbids. Here the missing letter is held back in the register and exposed by a new input letter $\mathtt{e}$.

Let $\mathcal{A}$ be a complete deterministic safety automaton over $\Sigma_I \times \Sigma_O$ with rejecting sink $\perp$ and accepting initial state q_I. Let $\Sigma = \Sigma_I \cup \{\mathtt{w}, \mathtt{e}\}$ and $\Gamma = \{x\}$.

- A has states $\{0, 1\}$; the bit records whether the last letter was $\mathtt{e}$, with $\Phi_A(1) = X$ and $\Phi_A(0) = \varepsilon$. It appends x on every $c \in \Sigma_I$ and ε on $\mathtt{w}, \mathtt{e}$.
- B has states (σ, bit) with $\sigma \in Q_{\mathcal{A}}^{\mathrm{acc}} \cup \{U, T\}$, initial state $(q_I, 0)$, $\Phi_B(\cdot, 1) = X$ and $\Phi_B(\cdot, 0) = \varepsilon$. Every edge sets the bit to [letter $= \mathtt{e}$]. On the σ-component: for accepting q, $c \in \Sigma_I$ and $b \in \Sigma_O$ with $q' = \delta_{\mathcal{A}}(q, (c, b))$, append x and go to q' if q' is accepting, otherwise append ε and go to U; on $\mathtt{w}$ or $\mathtt{e}$ append ε and go to T. In U: on $c \in \Sigma_I$ append x and stay, on $\mathtt{w}$ append ε and stay, on $\mathtt{e}$ append x and go to T. In T: append x on $c \in \Sigma_I$ and ε on $\mathtt{w}, \mathtt{e}$, always staying.

Promise. A is deterministic and complete, B is input-enabled, and both are one-register and append-only. Every B-run has appended one x per Σ_I-letter, minus exactly one while in U: the missing x is skipped on entry to U and repaid on the $\mathtt{e}$ that leaves it. When the last letter is $\mathtt{e}$, the run is outside U, both machines expose, and both registers hold $x^{\#\Sigma_I}$. Otherwise both outputs are ε. So (R4) holds and the instance lies in $\mathcal{F}_1$.

Arena. The letter $\mathtt{e}$ leads every pair into E_{XX}, so every reachable pair is live. By Lemma F.11, $\lambda = 1$ in U-states, since along $\mathtt{e}$ the machine B appends x and A appends ε, and $\lambda = 0$ elsewhere. So every round from an accepting state or from T is E′. In U the follower is B, and a round is E′ iff its letter is in $\Sigma_I \cup \{\mathtt{e}\}$. This is the arena of the claim in Section F.7 with one extra letter $\mathtt{e}$, which behaves like $\mathtt{w}$ from accepting states. From U, $\mathtt{e}$ is an E′-round into T; that only helps Controller.

Claim. Controller wins G'_d iff O wins $\Gamma_{f_d}(L_\forall(\mathcal{A}))$.

If O has a winning strategy τ with constant delay f_d, let Controller at a position with input y so far, window included, play the edge induced by $\tau(y')$, where y' replaces everything from the first $\mathtt{w}$ or $\mathtt{e}$ of y on by c_0, truncated to the same length. Before the first $\mathtt{w}$ or $\mathtt{e}$ the choices are those of a τ-play on a word over Σ_I, which is safe, so B stays in accepting states; at the first $\mathtt{w}$ or $\mathtt{e}$ it moves to T. U is never entered, so every round is E′.

Conversely let σ win G'_d. With lookahead, a play that has just entered U may already hold $\mathtt{e}$ in its window, which takes it to T by an E′-round. So we restrict first. The restriction of σ to inputs over Σ_I induces an O-strategy τ_σ with the same delay: an edge into an accepting q' maps to any b inducing

it, and the edge into U to any b with a rejecting successor. Suppose some τ_σ-play over Σ_I^ω rejects first at step j, so that σ's run enters U at step j. Keep the first $j+d$ input letters of that play and continue with $\mathtt{w}^\omega$. Choices at positions $\le j$ depend only on letters $\le j+d$, so the run still enters U at step j. The window then holds only Σ_I-letters and $\mathtt{w}$'s, which keep B in U, and from step $j+d$ on every round is a $\mathtt{w}$-round in U, which is non-E$'$. So the play has finitely many E$'$-rounds, against the choice of σ. Hence τ_σ wins.

Theorem F.13(1,2) now replaces the fixed-d and d_0 steps in the two hardness arguments of Section F.7, and [KZ16, Thm 4.3] gives EXPTIME-hardness of J and of Q_d with unary d. If q_I is rejecting, output the image of the fixed automaton $\mathcal{A}_0$ with $\Sigma_I = \{c\}$, $\Sigma_O = \{0\}$ and $\delta(q_I,(c,0)) = \bot$. That image is a no-instance: after c the run is in U, and $\mathtt{w}^\omega$ wins for Environment. □

Proposition F.15 (one register with append and prepend)**.** *Let $\mathcal{F}_1^{\pm}$ be the class of* **G***-instances in which each machine has one register, updates $X := uXv$, and $\Phi \equiv X$. Write $X := u_t X v_t$ for the update at step t, so the register after step t is $P(t)S(t)$ with $P(t) = u_t \cdots u_1$ and $S(t) = v_1 \cdots v_t$; births weakly decrease along P and weakly increase along S. Put $s(t) = |P_A(t)| - |P_B(t)|$ and let $m_\pm$ bound $|u_e|$ and $|v_e|$ over all edges.*

- *(a) The construction of Appendix D does not embed in $\mathcal{F}_1^{\pm}$.*
- *(b) Lemma F.1 fails: $s(t)$ is not a function of (p_t, q_t), and it can be unbounded on one pair.*
- *(c) Lemma F.2 fails: one step can raise* od *from* 0 *to any value.*
- *(d) If* od $\le H$ *on the prefix $w_{1..t}$, then $|s(t)| \le m_\pm(2H+1)$.*

Proof. (a) Theorem 7.1 uses 8 and 16 registers, four independent imbalance coordinates, one per register pair of Section 7.2, and edges that move registers across pairs. In $\mathcal{F}_1^{\pm}$ the two outputs are one word under (R4), so $|P_A|+|S_A| = |P_B|+|S_B|$: the two block-length differences are $s(t)$ and $-s(t)$, a single length coordinate, and no register can move. Equal block lengths do not determine the origin history, so other reductions are not excluded: with unary output, A prepending on c and appending on c', B appending on c and prepending on c', both one-state and idle on $\mathtt{w}$, the word $c\,\mathtt{w}^{K-2}c'$ has $s(K) = 0$ but origin sequences $(1,K)$ and $(K,1)$, so od $= K-1$.

(b) Take $\Gamma = \{x\}$, A one-state with $X := xX$ on every letter, and B one-state with two edges per letter, $X := xX$ and $X := Xx$. (R4) holds because the outputs are unary of equal length. The run that always appends has $s(t) = t$ on the single pair, while the run that always prepends has od $= 0$.

(c) Same machines, with the identity $X := X$ on a letter c' in both, and input $c\,c'^{K-2}c$. Let the B-run prepend at step 1 and append at step K. On the prefix of length $K-1$, od $= 0$. At step K the births of A are $(K,1)$ and those of B are $(1,K)$, so od $= K-1$, though no letter was ever pending in the sense of Section F.1.

(d) Suppose $s(t) > 0$ and let $R = (|P_B(t)|, |P_A(t)|]$, a set of output positions. In A, R lies inside P_A, where births weakly decrease; in B it lies inside S_B, where births weakly increase. For $x < y$ in R, $b_B(y) - b_B(x) \le (b_A(y) + H) - (b_A(x) - H) \le 2H$, since $b_A(y) \le b_A(x)$. So the births of B on R lie in at most $2H + 1$ consecutive steps, each appending at most $m_\pm$ letters, and $|R| \le m_\pm(2H + 1)$. The case $s(t) < 0$ is symmetric. □

By (d) a bound H keeps the split bounded. By (c) a bounded change of s costs roughly the gap between consecutive births near the split, so a finite summary would have to record, for each bounded shift, whether the history near the split is dense enough to tolerate it. Whether such a summary exists with a computable bound is open.

F.9. **Fixed ambiguity.** Let $\mathcal{O}_K$ be the class of one-way instances in which B has at most K runs on every word. Runs are sequences of transitions, and Δ_B is a set. Inputs are (A, B, d) with d in binary and (A, B, d, H) with d, H in binary. REACH is directed s–t reachability with $s = t$ counted as reachable, and reductions are logspace many-one.

Lemma F.16 (one-way at fixed K is in NL)**.** *Fix $K \ge 1$. On $\mathcal{O}_K$, J, Q_d with d in binary, and* GOOD(d, H) *with d, H in binary are in NL, and so are their arbitrary-memory versions, which have the same answers.*

Proof. Three logspace maps, each preserving the answer.

Embedding. Map (A, B) to the SSTs $A^{\mathrm{g}}, B^{\mathrm{g}}$ with the general final expressions of Remark 2.9: A^{g} has states Q_A, initial p_0, one register X_A, one edge $(p, c, X_A := X_A \cdot \mathrm{out}_A(p, c), \delta_A(p, c))$ per (p, c), and $\Phi(p) = X_A \cdot f_A(p)$; B^{g} has states Q_B, initial q_0, one register X_B, one edge $(q, c, X_B := X_B\beta, q')$ per transition (q, c, β, q'), and $\Phi(q) = X_B \cdot f_B(q)$. (R1) holds since δ_A is total and single-valued; (R2) since B is total; (R3) since $X := X\alpha$ and $\Phi(q) = X \cdot f(q)$ are copyless, registers start empty, all states are final and $X_A \ne X_B$. For (R4), let $u = u_1 \cdots u_n$. In A^{g} the letters of $\mathrm{out}_A(p_{t-1}, u_t)$ are born at t and those of $f_A(p_n)$ at $n + 1$; the one-way definition $\mathrm{orig}_A(x) = \min\{t \in [1, n + 1] : a(t) \ge x\}$ assigns the same steps, and for $n = 0$ both give birth 1. The same holds for every run of B through the bijection between transitions and edges. So the annotated outputs and od coincide, and the one-way promise is exactly (R4). The runs of B and of B^{g} are in bijection, so the K-promise is the same, and controllers are the same objects on the same reachable states. Hence GOOD(d, H) agrees on (A, B) and $(A^{\mathrm{g}}, B^{\mathrm{g}})$ for all d, H and both memory regimes. The map is a transcription with $O(\log)$-bit pointers.

Terminal marker. The marker construction of Appendix G turns $(A^{\mathrm{g}}, B^{\mathrm{g}})$ into a **G**-instance $(A^{\mathrm{g}\#}, B^{\mathrm{g}\#})$ with the K-promise and the same GOOD(d, H) for every (d, H) and both memory regimes, hence the same Q_d and J. It adds no register, so the image has $k^{\#} = 2$ registers in total and lies in $\mathcal{I}_{K,2}$; it is logspace with $s^{\#} \le 6s + O(1)$, and d, H pass unchanged.

Fixed registers. Theorem 8.5(a)–(c) places J, Q_d and $\mathrm{GOOD}(d,H)$ on $\mathcal{I}_{K,2}$ in NL, and NL is closed under logspace reductions. Since $k^{\#}=2$, the exclusion of $k\le 1$ there is irrelevant. The image lies in $\mathcal{F}_1$, but nothing about $\mathcal{F}_1$ is used.

Memory regimes. Theorem 4.4 gives $\mathrm{GOOD}=\mathrm{GOOD}^{\infty}$ on the **G**-instance $(A^{\mathrm{g}\#},B^{\mathrm{g}\#})$, hence on (A,B). □

Proposition F.17 (one-way NL-hardness at $K=1$)**.** *From a directed graph (V,E) and $s,t\in V$ one computes in logspace a one-way instance (A,B) with B deterministic and total, so $(A,B)\in\mathcal{O}_1$, with $|Q_A|=1$, $|\Gamma|=1$, $M=1$ and every output word nonempty, such that J, every Q_d and every $\mathrm{GOOD}(d,H)$ hold iff t is not reachable from s. This holds for every fixed d and (d,H), with d or d,H in the input in any encoding, and for the arbitrary-memory versions. Each of these problems is NL-hard on $\mathcal{O}_1$.*

Proof. For $s=t$ the reduction outputs the fixed instance I_0: the construction below on $V_0=\{0,1\}$, $E_0=\{(0,1)\}$, $s_0=0$, $t_0=1$. Assume $s\neq t$.

Skeleton. Let $\Sigma=V$, $\Gamma=\{x\}$ and $Q=V\cup\{\bot,T\}$ with $\bot,T\notin V$. Define a total deterministic $\delta:Q\times\Sigma\to Q$ by, for $v\in V$ and $u\in\Sigma$: $\delta(v,u)=u$ if $(v,u)\in E$ and $u\neq t$; $\delta(v,u)=T$ if $(v,u)\in E$ and $u=t$; $\delta(v,u)=\bot$ if $(v,u)\notin E$; and $\delta(\bot,u)=\bot$, $\delta(T,u)=T$. A step from a state in V to T is *entering*. For $w=w_1\cdots w_n$ let $q_j=\hat{\delta}(s,w_1\cdots w_j)$.

Claim. (i) Along every word at most one step is entering. (ii) $q_n=T$ iff some step $i\in[1,n]$ is entering, iff $w_1\cdots w_i$ is a walk $s\to w_1\to\cdots\to w_i=t$ in (V,E) with $w_j\neq t$ for $j<i$. (iii) Some word has $q_n=T$ iff t is reachable from s.

T is absorbing and entered only by an entering step. No rule leads to the state t, and $q_0=s\neq t$, so t is never a V-state; once the run leaves V it stays in $\{\bot,T\}$. This gives (i) and the first equivalence of (ii). While in V, $q_j=w_j$ and each step uses an edge, which gives the second. For (iii), a shortest path $s=v_0\to\cdots\to v_\ell=t$ has $\ell\ge 1$ and $v_j\neq t$ for $j<\ell$, so $v_1\cdots v_\ell$ enters at step ℓ; the converse is (ii).

Every word over Σ is a legal input: a non-edge sends the run to $\bot$, letters after T are absorbed, and $w=\varepsilon$ gives $q_0=s\in V$. The graph sits in the machines, and the input is free.

Construction. A has one state p with $\delta_A(p,u)=p$, $\mathrm{out}_A\equiv\varepsilon$ and $f_A(p)=x$. B has states Q, initial state s, transitions $(q,u,\beta_{q,u},\delta(q,u))$ with $\beta_{q,u}=x$ on entering steps and ε otherwise, $f_B(T)=\varepsilon$ and $f_B(q)=x$ for $q\neq T$.

Promise. A is deterministic and total. Δ_B has exactly one transition per (q,u), so B is total and deterministic with one run per word, and $K=1$. All states accept. $A(u)=x$ for every u. By (i), the output of B is $x\cdot f_B(T)=x$ if $q_n=T$ and $\varepsilon\cdot f_B(q_n)=x$ otherwise. $M=1$.

Origin distance. Fix w of length n. If no step is entering, both outputs are x, born at $n+1$ in both machines, at 1 when $n=0$, so $\mathrm{od}(w)=0$. If step i is entering, the x of A is born at $n+1$ and that of B at i, so $\mathrm{od}(w)=n+1-i\ge 1$.

Reduction. If t is unreachable from s, no word enters by (iii), so od $\equiv 0$. The one-state controller taking the unique transition exists for every d and achieves every H, so GOOD(d, H), every Q_d, J and all arbitrary-memory versions hold. If t is reachable, fix d, H, take the shortest-path word $v_1 \cdots v_\ell$ of (iii) and $w = v_1 \cdots v_\ell\, t^H$. Then $i = \ell$, $n = \ell + H$ and od$(w) = H + 1$. Since B is deterministic, every controller of every lookahead and memory produces the unique run on w, so GOOD(d, H) fails for all d, H in both memory regimes, and with it every Q_d and J. For $s = t$, I_0 has t_0 reachable from $s_0 \neq t_0$, so it is a no-instance. In all cases the image is positive iff t is unreachable.

Logspace. Compare s with t, and on $s = t$ write I_0. Otherwise write $\Sigma = V$, Q, and for each $(q, u) \in Q \times V$ one entry of δ with its output, deciding $(q, u) \in E$ by one scan of the edge list and $u = t$ by comparison; then f_A, f_B and a constant d or (d, H) in the required encoding. Loop indices take $O(\log |V| + \log |E|)$ bits, and the output has size $O(|V|^2)$. Malformed encodings are recognised in logspace and mapped to I_0 or to a fixed yes-instance, as the convention of REACH requires.

Hardness. The map reduces REACH to the complement of each problem, hence $\overline{\text{REACH}}$ to each problem, and $\overline{\text{REACH}}$ is NL-hard by the Immerman–Szelepcsényi theorem. Every image satisfies the promise, so the problems are NL-hard as promise problems on $\mathcal{O}_1$, and hardness transfers to every language that agrees with them on promised instances. □

A second construction on the same skeleton gives A the states Q with $\text{out}_A(q, u) = x$ on entering steps and $f_A \equiv \varepsilon$, and B the outputs ε with $f_B(T) = x$. The promise holds by (i) and (ii), and od $= n + 1 - i$ when step i enters; otherwise both outputs are empty and od $= 0$ only by the convention $\max \emptyset = 0$. We use the construction above because it needs no such convention and has a one-state reference, as in Proposition 8.7.

Proof of Proposition 9.2. Lemma F.16 gives membership and Proposition F.17 gives hardness, already at $K = 1$ with $|\Gamma| = 1$, $M = 1$, a deterministic B and a one-state reference. □

The NL bound does not come from the game G_d, which is exponential in d, but from the SST image, on which Theorem 8.5 applies. So no EXPTIME-hardness proof can keep the ambiguity below a constant. With K in the input, one-way J lies between NL and EXPTIME, and nothing finer is proved. Lemma F.16 needs the two birth conventions to agree, with final letters born at $n+1$ and at 1 when $n = 0$. If either moved, od would change by at most 1: J and Q_d would be unaffected, and GOOD(d, H) would be sandwiched between $H \pm 1$.

Remark F.18 (final words make the one-way model nontrivial). If $f_A \equiv f_B \equiv \varepsilon$, then od $= 0$ for every one-way instance, word and run, at any ambiguity.

Proof. Let ρ be a run of B on u. Its prefix of length j is a run on $u_1 \cdots u_j$, by totality and since all states accept, so by the promise at $u_1 \cdots u_j$ it has

output length $a(j)$. So $b(j) = a(j)$ for $j \le n$ and at $n+1$, where nothing is added, and the two origin maps coincide. □

So every one-way hardness proof needs nonempty final words: Section F.7 uses $f_B(U) = x$ and Proposition F.17 uses $f_A(p) = x$. The remark uses word agreement on every prefix of a run. Under $\mathbf{G}^\flat$ it holds only on accepted words, and partial machines lie outside the one-way model.

Appendix G. General final expressions

We prove Remark 2.9: a terminal marker turns a general final expression into an ordinary update, with the same annotated outputs, and the results of the body transfer through this map.

G.1 The general model.

Definition G.1 (general SST)**.** A *general SST* is an SST of Definition 2.1 whose final function is $\Phi : Q \to (\Gamma \cup \mathcal{X})^*$, each register occurring at most once in $\Phi(q)$. A general SST with final states has $\Phi : F \to (\Gamma \cup \mathcal{X})^*$ under the same restriction. Runs and valuations v_i are unchanged. For a run ρ on $w = a_1 \cdots a_n$ ending in q_n (with $q_n \in F$ when final states are present) we put

$$[\![\rho]\!] = \Phi(q_n)\big[Y \mapsto v_n(Y),\ \gamma \mapsto (\gamma, n+1)\big] \in (\Gamma \times [n+1])^*.$$

A constant of the final expression is born at step $n+1$. For $w = \varepsilon$ the constants of $\Phi(q_0)$ are born at 1. The promises $\mathbf{G}_{\text{gen}}$ and $\mathbf{G}^\flat_{\text{gen}}$ are those of Remark 2.3 and Definition 2.2, with the fresh-free condition deleted and with this semantics in the word condition. Controllers, od and the problems GOOD, GOOD$^\infty$, Q_d, J and their $\flat$ versions keep their definitions. Only the annotated outputs change, and the thresholds in od range over $\{0, \dots, n{+}1\}$. A fresh-free instance is a general instance with the same annotated outputs. So the body's problems are the restrictions of the general ones to fresh-free instances.

G.2 The terminal marker. Let $\# \notin \Sigma$ and $\Sigma^\# = \Sigma \cup \{\#\}$. For a general SST $\mathcal{T} = (Q, q_0, \mathcal{X}, E, \Phi)$ fix a register $Z \in \mathcal{X}$, adding one if $\mathcal{X} = \emptyset$. The machine $\mathcal{T}^\# = (Q \cup \{f\}, q_0, \mathcal{X}, E^\#, \Phi^\#)$, with f fresh, has:

- every edge of E; if Z was added, each update is extended by $Z := \varepsilon$;
- for each $q \in Q$ an edge $e_q^\# = (q, \#, \sigma_q, f)$ with $\sigma_q(Z) = \Phi(q)$ and $\sigma_q(Y) = \varepsilon$ for $Y \neq Z$;
- for each $c \in \Sigma^\#$ a loop (f, c, ι_Z, f) with $\iota_Z(Z) = Z$ and $\iota_Z(Y) = \varepsilon$ for $Y \neq Z$;
- $\Phi^\#(q) = \varepsilon$ for $q \in Q$ and $\Phi^\#(f) = Z$.

Each σ_q is copyless, because $\Phi(q)$ is its only nonempty right-hand side and uses each register at most once. $\Phi^\#$ is fresh-free. So $\mathcal{T}^\#$ is an SST of Definition 2.1 over $\Sigma^\#$. With final states, $\mathcal{T}^{\#\flat}$ is built in the same way, except that $e_q^\#$ exists only for $q \in F$, f has no outgoing edge, and $F^\# = \{f\}$.

For an instance we write $(A,B)^{\#} = (A^{\#}, B^{\#})$ and $(A,B)^{\#\flat} = (A^{\#\flat}, B^{\#\flat})$, with $Z_A \in \mathcal{X}_A$ and $Z_B \in \mathcal{X}_B$, so the register sets stay disjoint.

Throughout, $w = a_1 \cdots a_n \in \Sigma^*$ and windows are those of Section 2.3.

Lemma G.2 (window correspondence)**.** *Fix d. There are maps π_d, from the windows over $\Sigma^{\#}$ whose first letter lies in Σ, to Win_d, and $\iota_d : \mathrm{Win}_d \to \mathrm{Win}_d^{\#}$, both preserving the first letter, such that for every w, every $v \in (\Sigma^{\#})^*$ and every $1 \le i \le n$,*

$$\pi_d(\mathrm{win}_d(w\#v, i)) = \mathrm{win}_d(w, i), \qquad \iota_d(\mathrm{win}_d(w, i)) = \mathrm{win}_d(w\#, i),$$

and π_d is the identity on the windows of words in Σ^*.

Proof. If $\#$ occurs in ω, write $\omega = x\#y$ with $x \in \Sigma^+$ and put $\pi_d(\omega) = x \dashv$; otherwise $\pi_d(\omega) = \omega$. If ω contains no $\dashv$, put $\iota_d(\omega) = \omega$; if $\omega = x \dashv$ with $|x| = \ell$, put $\iota_d(\omega) = x\#$, followed by $\dashv$ iff $\ell < d$.

Let $u = w\#v$. The window $\mathrm{win}_d(u, i)$ covers positions $i, \dots, \min(|u|, i+d)$, so it contains the marker at position $n+1$ iff $n - i < d$. This is exactly when $\mathrm{win}_d(w, i)$ shows $\dashv$, by strict $\dashv$. If $n - i < d$, then $\mathrm{win}_d(w, i) = a_i \cdots a_n \dashv$ and $\mathrm{win}_d(u, i) = a_i \cdots a_n \# y$ for some y, so π_d returns $a_i \cdots a_n \dashv$. If $n - i \ge d$, then $i + d \le n < |u|$, both windows equal the full word $a_i \cdots a_{i+d}$, and π_d changes nothing. A word in Σ^* has no marker in any window.

For ι_d: if $n - i \ge d$, both windows are $a_i \cdots a_{i+d}$. If $n - i < d$, then $\min(n+1, i+d) = n+1$, so $\mathrm{win}_d(w\#, i)$ is $a_i \cdots a_n \#$ followed by $\dashv$ iff $n + 1 - i < d$. With $\ell = n - i + 1 = |a_i \cdots a_n|$ this reads $\ell < d$, and ℓ is visible in $\mathrm{win}_d(w, i) = a_i \cdots a_n \dashv$. For $d = 0$ no window at a position $i \le n$ contains $\#$ or $\dashv$, and both maps are the identity. □

The lookahead is unchanged, even at $d = 0$. Strict $\dashv$ means exactly that the marker lies inside the window.

Lemma G.3 (origins are preserved)**.** *Let $\rho = e_1 \cdots e_n$ be a run of $\mathcal{T}$ on w ending in q_n, and let $v \in (\Sigma^{\#})^*$. Put $\rho^{\#} = \rho\, e_{q_n}^{\#}\, g_1 \cdots g_{|v|}$, where the g_j are the loops at f. Then $\rho^{\#}$ is the unique extension of ρ to a run of $\mathcal{T}^{\#}$ on $w\#v$, and $[\![\rho^{\#}]\!]_{\mathcal{T}^{\#}} = [\![\rho]\!]_{\mathcal{T}}$. A run of $\mathcal{T}^{\#}$ on a word of Σ^* outputs ε. The same holds for $\mathcal{T}^{\#\flat}$ with $q_n \in F$ and $v = \varepsilon$.*

Proof. $e_q^{\#}$ is the only $\#$-edge at q, and f has one edge per letter, so the extension is unique. For $i \le n$ the edges are those of $\mathcal{T}$, so the valuations agree on $\mathcal{X}$; an added Z holds ε throughout. At step $n+1$ register Z holds $\Phi(q_n)[Y \mapsto v_n(Y), \gamma \mapsto (\gamma, n+1)]$ and every other register is empty. Each loop keeps Z and creates nothing, and $\Phi^{\#}(f) = Z$. So the two annotated outputs agree, births included. A run on a word of Σ^* ends in Q, where $\Phi^{\#} = \varepsilon$. □

Proposition G.4 (exact reduction under $\mathbf{G}$)**.** *Let (A, B) satisfy $\mathbf{G}_{\mathrm{gen}}$.*

(1) $(A^{\#}, B^{\#})$ satisfies $\mathbf{G}$. Conversely, if $(A^{\#}, B^{\#})$ satisfies $\mathbf{G}$, then (A, B) satisfies $\mathbf{G}_{\mathrm{gen}}$.

(2) *For $u \in (\Sigma^{\#})^*$, $B^{\#}$ has exactly as many runs on u as B has on w, where $w = u$ if $u \in \Sigma^*$, and otherwise w is the prefix of u before its first $\#$. So the K-promise, the bound of at most $|v|+1$ runs on v, and the fork budget κ_B are the same for B and $B^{\#}$.*

(3) *For every $d, H \in \mathbb{N}$ and either memory regime, $\mathrm{GOOD}(d, H)$ holds for (A, B) iff it holds for $(A^{\#}, B^{\#})$. Controllers convert in both directions with the same memory set M. Hence Q_d, J, the least lookahead, and the least H for each d coincide.*

(4) *$|Q_{X^{\#}}| = |Q_X| + 1$ and $k_{X^{\#}} = \max(k_X, 1)$, so $k^{\#} \le k + 2$. The per-step output bound becomes $\widehat{\mu} = \max\left(\mu_A, \mu_B, \max_{X,q} |\Phi_X(q)|_\Gamma\right) \le s$. The size is $s^{\#} \le 6s + O(1)$, and the map is computable in logarithmic space.*

Proof. (1) $A^{\#}$ keeps the Σ-edges of A, has one $\#$-edge at each state of Q_A and one edge per letter at f. So it is deterministic and complete. The reachable states of $B^{\#}$ are the reachable states of B together with f, since only Σ-edges stay inside Q_B. A reachable $q \in Q_B$ has its Σ-edges by the promise for B, and it has $e_q^{\#}$; f has every letter. Copylessness and freshness were checked with the construction. Word agreement: on $u \in \Sigma^*$ every run of both machines outputs ε by Lemma G.3. On $u = w\#v$ every run of $B^{\#}$ is $\beta^{\#}$ for a run β of B on w, by uniqueness in Lemma G.3, and

$$\lfloor [\![\beta^{\#}]\!] \rfloor = \lfloor [\![\beta]\!] \rfloor = \lfloor [\![\rho_A(w)]\!] \rfloor = \lfloor [\![\rho_{A^{\#}}(u)]\!] \rfloor$$

by word agreement for (A, B). Conversely, the conditions on (A, B) are read off the Σ-edges of $(A^{\#}, B^{\#})$, and word agreement on w is word agreement for the marker instance on $w\#$, again by Lemma G.3.

(2) Runs of $B^{\#}$ on $u \in \Sigma^*$ are the runs of B on u. Runs on $w\#v$ are exactly the $\beta^{\#}$ with β a run of B on w, and $|w| + 1 \le |w\#v| + 1$. The new pairs $(q, \#)$ and (f, c) have one edge each and the old pairs are unchanged, so the sum defining κ_B does not move.

(3) *Original to marker.* Let $\mathcal{C} = (M, m_0, \chi)$ be a controller for B with lookahead d and $\mathrm{od}(\rho_A(w), \rho_{\mathcal{C}}(w)) \le H$ for all $w \in \Sigma^*$. Define $\mathcal{C}^{\#} = (M, m_0, \chi^{\#})$ on the reachable states of $B^{\#}$: for $q \in Q_B$ and a window ω starting in Σ, $\chi^{\#}(m, q, \omega) = \chi(m, q, \pi_d(\omega))$; for $q \in Q_B$ and ω starting with $\#$, $\chi^{\#}(m, q, \omega) = (e_q^{\#}, m)$; at f, the unique edge and m. This is legal, because π_d preserves the first letter and $B^{\#}$ has the Σ-edges of B at q. On $u \in \Sigma^*$ the map π_d is the identity on the windows that occur, so $\rho_{\mathcal{C}^{\#}}(u) = \rho_{\mathcal{C}}(u)$, both outputs are ε, and the distance is 0. On $u = w\#v$, induction on $i \le n$ with Lemma G.2 shows that the memory states and chosen edges of $\mathcal{C}^{\#}$ agree with those of $\mathcal{C}$ on w. So $\rho_{\mathcal{C}^{\#}}(u) = \rho_{\mathcal{C}}(w)^{\#}$, and by Lemma G.3 both annotated outputs are those of the original instance on w. The distance is unchanged.

Marker to original. Let $\mathcal{C}^{\#} = (M, m_0, \chi^{\#})$ be good for $(A^{\#}, B^{\#})$. Define $\chi(m, q, \omega) = \chi^{\#}(m, q, \iota_d(\omega))$ for reachable $q \in Q_B$. It is legal, since ι_d preserves the first letter and the Σ-edges of $B^{\#}$ at q are those of B. By

Lemma G.2 and induction, $\rho_{\mathcal{C}}(w)$ is the length-n prefix of $\rho_{\mathcal{C}^\#}(w\#)$, whose next edge is the forced $e^{\#}_{q_n}$. By Lemma G.3 the annotated outputs on w and on $w\#$ coincide for A and for B, so od $\leq H$ on every w.

Both maps keep M. So finite memory is preserved, and the arbitrary-memory problems correspond as well. Since (d, H) is unchanged, Q_d, J, the least d and the least H for each d coincide.

(4) Per machine, $\mathcal{T}^\#$ adds one letter, one state, at most one register and $|Q| + |\Sigma| + 1$ edges. Its occurrences are those of $\mathcal{T}$, with $\sum_q |\Phi(q)|$ moved from the final expressions to the #-edges, plus $|\Sigma| + 2$ for the loops at f and $\Phi^\#(f)$; an entry $\sigma(Y) = \varepsilon$ carries no occurrence. Summing over A and B, with $|Q_A|+|Q_B| \leq s$ and $|\Sigma| \leq s$, gives $s^\# \leq s+(|Q_A|+|Q_B|)+4|\Sigma|+O(1) \leq 6s + O(1)$. A convention that charges one unit per register entry of every update adds at most $|Q| \cdot k$ for the #-edges, which the edges of reachable states already pay once unreachable states are deleted, so the size stays $O(s)$. The #-edge creates the $|\Phi(q)|_\Gamma$ constants of $\Phi(q)$ in one step, so $\mu^\# = \widehat{\mu}$. This loss is inherent: the general semantics itself gives every final constant the birth $n + 1$, and Lemma 4.1 counts letters per birth. The construction scans the input once with $O(\log s)$-bit pointers. □

Proposition G.5 (exact reduction under $\mathbf{G}^\flat$)**.** *Let (A, B) satisfy $\mathbf{G}^\flat_{\mathrm{gen}}$. Then $(A^{\#\flat}, B^{\#\flat})$ satisfies $\mathbf{G}^\flat$; $\mathrm{dom}(X^{\#\flat}) = \mathrm{dom}(X) \cdot \#$ for $X \in \{A, B\}$; the accepting runs of $B^{\#\flat}$ on $w\#$ are in bijection with the accepting runs of B on w, so the $K^\flat$-promise and the degree K are the same; for every (d, H) and both memory regimes, $\mathrm{GOOD}^\flat(d, H)$ holds for (A, B) iff it holds for $(A^{\#\flat}, B^{\#\flat})$, with controllers converting with the same memory; and size, registers and $\widehat{\mu}$ are as in Proposition G.4(4). For a regular obligation domain O, use $O \cdot \#$: the product of $B^{\#\flat}$ with a DFA for $O\#$ has the same degree of ambiguity as the product of B with a DFA for O.*

Proof. $A^{\#\flat}$ is deterministic and partial, with one #-edge per final state and none at f, so (R1) holds; (R2) and (R3) are as in Proposition G.4. A run of $X^{\#\flat}$ is accepting iff it ends in f, iff it is $\rho\, e^{\#}_{q_n}$ with ρ an accepting run of X on w, because only final states have a #-edge and f has no exit. So $\mathrm{dom}(X^{\#\flat}) = \mathrm{dom}(X)\#$, accepting runs are in bijection, and (R4) follows from the general (R4) and Lemma G.3.

Controllers act on the sink-completed machine B_z of Section 2.3. In $B_z^{\#\flat}$ the Σ-edges at $q \in Q_B$ are those of B_z, since z-edges are added exactly where $\Delta_B(q, a) = \emptyset$; at $q \in Q_B$ the letter # has the edge $e^{\#}_q$ if $q \in F_B$ and the z-edge otherwise; f has only z-edges. The two controller maps are those of Proposition G.4(3), with π_d, ι_d unchanged and the forced choice at a #-window. The obligations of the marker instance are exactly the words $w\#$ with $w \in \mathrm{dom}(B)$; Lemma G.2 is used with $v = \varepsilon$. The run $\rho_{\mathcal{C}^\#}(w\#)$ is accepting iff its prefix on w ends in F_B, because otherwise # leads to the absorbing non-final z; that is, iff $\rho_{\mathcal{C}}(w)$ is accepting. Then the annotated outputs coincide by Lemma G.3. For obligation domains, $O\#$ is regular,

and the runs of the product with a DFA for $O\#$ correspond to the runs of the product of B with a DFA for O in the same way. □

G.3 Transfer. The next statement makes Remark 2.9 precise.

Proposition G.6 (transfer to general final expressions)**.** *With final outputs in* $(\Gamma \cup \mathcal{X})^*$ *and constants born at* $n+1$*:*

- *(B♭)* *Theorem 3.1, Propositions 4.5, 6.7 and 8.18(1),(2),(4) hold for* $\mathbf{G}^{\flat}_{\mathrm{gen}}$*-instances, with the same* K *and* $N = 2^K$ *and with* s *replaced by* $O(s)$*. The memory of Proposition 4.5 has* $|Q_A|+2$ *in place of* $|Q_A|+1$ *in its first factor.*
- *(B)* *Remark 3.2 holds for* $\mathbf{G}_{\mathrm{gen}}$*-instances with the* K*-promise, with* T_* *the cutoff of* $(A^{\#}, B^{\#})$ *and the same* $N = 2^{\kappa}$*,* $\kappa = \min(K-1, \kappa_B)$*. So* $\log_2 T_* = O(N^3 s + N^4)$*;* J *and* Q_d *are in PSPACE for fixed* K *and for* $K = O(\log s)$*;* $J \in \mathrm{NSPACE}(O(s)) \subseteq \mathrm{DTIME}(2^{O(s)})$ *for fixed* K*; EXPSPACE for finite ambiguity with* K *not supplied; the least lookahead on a positive instance is* $2^{O(s)}$ *for fixed* K*; and the controller memory is* $(|Q_A|+1) \cdot 2^{2^{O(N^3 s + N^4)}}$*, by Proposition 8.2, Theorem 8.3 and Corollary 8.4 applied to* $(A^{\#}, B^{\#})$*. Corollary 3.3 transfers by the same exactness.*
- *(C)* *Theorem 3.4 holds for general instances as stated, with the same fixed reference* A_**, because its instances are fresh-free. Membership in RE transfers by Propositions G.4 and G.5, so* Q_d *and* J *stay RE-complete at linear ambiguity.*

(Lower bounds) *Every lower bound of Sections 7 and 8 holds for the general model, by inclusion of the fresh-free model. PSPACE-completeness of* GOOD *at fixed* K *(Corollary 8.11, Proposition 8.18(3)) and RE-completeness at linear ambiguity hold for the general model: the upper bound by the reduction, the lower bound by inclusion.*

(Promise) $\mathbf{G}_{\mathrm{gen}}$ *and* $\mathbf{G}^{\flat}_{\mathrm{gen}}$ *are decidable, by Remark 2.4 applied to the marker instance.*

(Parameters) *Every bound stated in* s *survives with* $s^{\#} = O(s)$*. A bound stated in* μ *must be read with* $\widehat{\mu}$*, so a result that assumes fixed* μ *transfers only to general instances with fixed* $\widehat{\mu}$*. A result for a fixed register count* k *transfers with* $k^{\#} \le k+2$*: this covers the fixed-*k *parts of Theorem 8.5 and Corollary 8.8, since the reduction is logspace. The case* $k \le 1$ *of Theorem 8.5 does not transfer, because a general instance with* $k \le 1$ *can map to* $k^{\#} = 2$*. Theorem 8.15 and every other lower bound transfer by inclusion.*

Proof. Upper bounds. Map the instance to $(A^{\#}, B^{\#})$, or to $(A^{\#\flat}, B^{\#\flat})$, which satisfies $\mathbf{G}$, or $\mathbf{G}^{\flat}$, with the same K, κ_B, d and H by Propositions G.4 and G.5. Apply the cited body results to the image. For the offline statement, the quantity $\min_{\beta}$ od is preserved by the run bijection and Lemma G.3. The output controller is converted back by ι_d with the same memory. Every bound is in terms of $s^{\#} = O(s)$, N, $|Q_A|+1$, $k^{\#} \le k+2$ and $\mu^{\#} = \widehat{\mu} \le s$.

The parameter μ enters through the weight bound $W \le \max(\mu, 1)$ of Proposition 6.7, the counter ranges $[-\mu_B H, \mu_A H]$ of Section 8.1 and Proposition 8.18(2), and the state count of $\mathcal{N}_H$ in Theorem 4.2, hence the memory bounds of Corollary 8.4 and Proposition 8.18(4). A bound in s survives because $\widehat{\mu} \le s^{\#}$. The reduction is logspace with linear output, so NL, NSPACE($O(s)$), PSPACE and EXPSPACE are closed under it: recompute the output bits on demand.

Lower bounds and Theorem 3.4. A fresh-free instance is a general instance with identical annotated outputs. So every hardness reduction of Sections 7 and 8 is already a reduction to the general problem. □

Origin convention. Exactness depends on births at $n+1$. Under the alternative convention, where final constants are born at n (and at 1 for $n = 0$), only final constants move, each by exactly 1 and in the same way in A and B. So od changes by at most 1, and GOOD(d, H) moves to $H \pm 1$. Under either convention Q_d, J and the least lookahead are the same. We use $n+1$.
The one-way model. A one-way pair of Section 9 with final words f_A, f_B is a pair of one-register general SSTs with updates $X := X\alpha$ and $\Phi(q) = X \cdot f(q)$, with the same origins and the same od. Conversely, a one-register general SST with updates $X := X\alpha$ and $\Phi(q) = X \cdot c_q$ at every state, $c_q \in \Gamma^*$, and with B total, is a one-way pair. So the one-way model is a submodel of the general model. The marker reduction does not map one-way instances to one-way instances, since its image has $\Phi^{\#} = \varepsilon$ on Q. Proposition 3.10 handles final words directly, and Appendix F is unchanged.
Limits. A final relation, with several final expressions per state, is outside this construction: each expression would need its own #-edge, K would change, and the run bijection would have to be redone. Under $\mathbf{G}$ the $\#\flat$ construction must not be used, because it leaves $\mathbf{G}$ through partial and non-final states, and only the bounds of Theorem 3.1 would apply. We do not extend the results of Appendix F on the fragment $\mathcal{F}_1$ or of Appendix H on $\mathbf{G}^{\natural}$ to general final expressions.

G.4 Change of reference.

Lemma G.7 (od is a metric)**.** *On annotated words with a common underlying word,* od *is symmetric, vanishes exactly on equal words, and satisfies* $\mathrm{od}(\alpha, \gamma) \le \mathrm{od}(\alpha, \beta) + \mathrm{od}(\beta, \gamma)$.

Proof. Rank by rank, $|b_\alpha(x) - b_\gamma(x)| \le |b_\alpha(x) - b_\beta(x)| + |b_\beta(x) - b_\gamma(x)|$, and each term is bounded by the corresponding distance. Distance 0 means equal births at every rank, on the same word. □

Proof of Proposition 2.10. Fix a controller $\mathcal{C}$ and an input w in the relevant set. Both annotations of w carry B's word, by word agreement, so Lemma G.7 applies and

$$\mathrm{od}(\rho_{A'}(w), \rho_{\mathcal{C}}(w)) \le \mathrm{od}(\rho_{A'}(w), \rho_A(w)) + \mathrm{od}(\rho_A(w), \rho_{\mathcal{C}}(w)),$$

which is at most $\Delta + \mathrm{od}(\rho_A(w), \rho_{\mathcal{C}}(w))$. Under $\mathbf{G}^\flat$ the acceptance obligation does not mention the reference. This gives (1), and (2) and (3) follow from it.

For the converse take $B = A'$, with its registers renamed apart from those of A. The pair (A, A') is a $\mathbf{G}$-instance: A' is complete, and the word functions agree. The pair (A', A') has distance 0, so $J(A', A')$ holds, and by hypothesis so does $J(A, A')$. Since A' is deterministic, every controller produces $\rho_{A'}(w)$. So $J(A, A')$ says exactly that $\Delta(A, A') < \infty$. □

Appendix H. Dropping word agreement

We prove the extension in Remark 6.8, including its cutoff, controller extraction, and complexity bounds.

H.1. **Overview.** A leaf of a kill witness is now bad if its run is accepting and the two annotated outputs either differ as words or lie at distance above H. The witness automata get three leaf types beside OD: a letter mismatch at one rank, and a length mismatch in either direction, each with a counter R_λ that must end at 0; the length types colour the shorter clone P^*. The kill-witness theorem, the finite game and Theorem 6.3 carry over once R_H^{od} is the four-type DFA of Theorem 4.2; the GAP-only checker is exact only when the words agree. Lemma 6.4 keeps its all-DEAD form but loses its role. A case with $\varphi = \emptyset$ and only mismatch leaves can have an accepting walk, and such a walk is a permanent obstruction with neither d nor H in its guards. These cases are exactly the words $u \in O$ on which no accepting run of B outputs A's word. The cutoff argument absorbs them: a piece with no row in T has the ray e_T, and $\neg J^\natural$ is then correct. The cutoff $T_*^\natural$ has the same form, and the fixed-K complexity bounds survive with s replaced by the size of $(A, B \times D_O)$.

Definition H.1 (The promise $\mathbf{G}_O^\natural$)**.** An instance consists of A, B and a complete DFA $D_O = (Q_O, o_0, \delta_O, F_O)$ over Σ, with $O = L(D_O)$. The machines satisfy (R1)–(R3) of Definition 2.2, and

$$O \subseteq \mathrm{dom}(A) \cap \mathrm{dom}(B).$$

Thus both machines are copyless and have fresh-free final expressions. The reference A is deterministic and may be partial. Accepting runs of B may produce different output words. The $K^\natural$-promise requires at most K accepting runs on each word of O; write K_O for the least such bound.

Two annotated words are *H-good* when their underlying words agree and their origin distance is at most H. We write $\mathrm{GOOD}^\natural(d, H)$ when a finite-memory controller for B_z, with lookahead d, chooses an accepting run on every $w \in O$ whose annotated output is H-good with $[\![\rho_A(w)]\!]$. The superscript $\natural, \infty$ permits arbitrary memory. Finally, put

$$Q_d^\natural = \exists H\, \mathrm{GOOD}^\natural(d, H), \qquad J^\natural = \exists d, H\, \mathrm{GOOD}^\natural(d, H),$$

and use the same convention for arbitrary memory.

Theorem H.2 ($\mathrm{B}^\natural$)**.** *Given a $\mathbf{G}^\natural_O$-instance with $K_O < \infty$, we can compute K_O, decide $J^\natural$, and decide $Q^\natural_d$ for each supplied d. On a positive instance of $J^\natural$, we can construct a controller with $d = H = T^\natural_*$, where $T^\natural_*$ is defined in Proposition H.10. Its memory can be taken to be*

$$(Q_A \cup \{p_\perp\}) \times Q_O \times \mathrm{states}(R^{\mathrm{od}}_{T^\natural_*}),$$

where R^{od}_H checks the full H-goodness condition. When $O = \mathrm{dom}(B)$, the factor Q_O can be omitted.

The answers agree for arbitrary-memory controllers. For every fixed pair (d, H), with any ambiguity, $\mathrm{GOOD}^\natural(d, H)$ and $\mathrm{GOOD}^{\natural,\infty}(d, H)$ are equivalent and decidable by a finite game. Consequently $Q^\natural_d$ and $J^\natural$ are recursively enumerable.

H.2. **The obligation domain.**

Lemma H.3 (Product with the obligation automaton)**.** *Let $B' = B \times D_O$ have states $Q_B \times Q_O$, initial state (q^B_0, o_0), and final states $F_B \times F_O$. For each $e \in \Delta_B(q, a)$ and $o \in Q_O$, give it the edge*

$$(q, o) \xrightarrow{a/\mathrm{upd}(e)} (\mathrm{dst}(e), \delta_O(o, a)).$$

Keep the registers of B, and set $\Phi'(q, o) = \Phi_B(q)$ at final states. Then $\mathrm{dom}(B') = O$. On each $w \in O$, projection gives a bijection between accepting runs of B' and B, preserving their annotated outputs and their ambiguity.

For all d, H, the original instance and the instance with machine B' and obligation domain $\mathrm{dom}(B')$ have the same answer to $\mathrm{GOOD}^\natural(d, H)$, also with arbitrary memory. A controller for B' with memory M yields one for B with memory $M \times Q_O$; the reverse construction keeps M.

Proof. The second component of a product run is determined by its input. Hence every run of B has exactly one lift, with identical updates. The lift accepts precisely when the original run accepts and D_O accepts. Since $O \subseteq \mathrm{dom}(B)$, its domain is O. Final expressions also agree.

An input letter has no outgoing edge from (q, o) precisely when it has none from q. Thus the forced edges into z correspond. Given a controller for B, lift its chosen edge uniquely. The product state already supplies o, so no memory is added.

Conversely, store o in the memory of a controller for B. Update it using the head letter of each window. At a reachable pair (q, o), project the edge chosen by the product controller. At an unreachable pair choose any fixed legal edge of B_z. This completes the controller's definition. At z, take its forced loop and continue updating the simulated memory. Induction on the input position shows that actual histories always follow reachable product states until z is entered. The runs therefore enter z together. On O, their acceptance and annotated outputs coincide. □

We henceforth work with $O = \mathrm{dom}(B) \subseteq \mathrm{dom}(A)$ and write $\mathbf{G}^{\natural}$. The product lemma restores general O at the end. If O is empty, every obligation is vacuous and $K_O = 0$. Otherwise $K_O \geq 1$. The acceptance constructions in Sections 4 and 5 apply to this domain.

H.3. **Kill witnesses and fixed resources.**

Definition H.4 (Output-failure leaves)**.** A kill witness has the data and conditions (K1)–(K3) of Definition 5.3. Its leaf types are OUT and DEAD. At an OUT leaf, β_λ accepts and

$$\big([\![\rho_A(u_\lambda)]\!], [\![\beta_\lambda]\!]\big)$$

fails to be H-good. At a DEAD leaf, β_λ rejects. The shapes $\mathfrak{S}^{\flat}_K$, gap coordinates, and d-coherence retain their previous meanings.

Lemma H.5 (Kill witnesses under $\mathbf{G}^{\natural}$)**.** *A d-coherent witness from Definition H.4 defeats every lookahead-d controller, with arbitrary memory. Under the $K^{\natural}$-promise, failure of* $\mathrm{GOOD}^{\natural,\infty}(d, H)$ *yields such a witness with shape in $\mathfrak{S}^{\flat}_K$.*

Proof. We use the argument of Theorem 5.4. For soundness, follow the controller through the witness. If its run follows a leaf to the end, the leaf type gives failure: either rejection or failure of H-goodness. If it leaves all represented continuations, (K3) supplies a compatible input on which the chosen edge has no accepting continuation. Coherence preserves the controller's observed windows up to that choice. Lemma B.1 then gives rejection on an obligation word.

For completeness, use the same history arena, declaring (w, β) bad when

$$w \in \mathrm{dom}(B) \quad \text{and} \quad \big(\beta \text{ rejects or the output pair fails to be } H\text{-good}\big).$$

The environment seeks a bad terminal. The reachability argument in Theorem 5.4 gives a winning environment strategy when the controller cannot succeed. Its unfolding over controller choices is finitely branching. Every branch reaches a bad terminal. König's lemma therefore bounds its depth.

The pruning argument distinguishes unsafe choices from terminal failures along safe choices. Unsafe choices retain their rejecting continuations and give DEAD leaves. Along a safe run, an obligation word must be accepted. Consequently a bad terminal reached through safe choices has an accepting run whose output pair fails H-goodness. Give it type OUT.

All remaining parts of the construction concern input words and edges: the fans, their branching positions, the gap triples, and (K1)–(K3). They remain valid. The live-alternative count uses acceptance of the surviving leaves and their compatible continuations. Both properties still hold. DEAD leaves need not be accepting, so Lemma B.5 gives the path budget K, as in Appendix B. Thus the contracted shape belongs to $\mathfrak{S}^{\flat}_K$. □

Lemma H.6 (Fixed resources)**.** *For every d, H and any ambiguity,*

$$\mathrm{GOOD}^{\natural}(d,H) \iff \mathrm{GOOD}^{\natural,\infty}(d,H) \iff \textit{the controller wins } G^{\natural}_{d,H},$$

where $G^{\natural}_{d,H}$ is a finite game. For $O = \mathrm{dom}(B)$, a winning controller needs memory only

$$(Q_A \cup \{p_\perp\}) \times \mathrm{states}(R_H^{\mathrm{od}}).$$

For general O, an additional factor Q_O suffices.

Proof. Use all four types of the NFA in Theorem 4.2: GAP, LET, LEN_A, and LEN_B. Complement its determinization to obtain R_H^{od}. On a well-formed joint word, this DFA accepts precisely when the outputs are H-good.

Take the arena of Theorem 4.4, retaining its strict $\dashv$ convention. At a terminal, let S be the subset of B-states reachable on the input, and let q_B be the selected state. The terminal is bad exactly when

$$S \cap F_B \neq \emptyset \quad \text{and} \quad \Big(q_B \notin F_B \text{ or } R_H^{\mathrm{od}} \text{ rejects the completed joint word}\Big).$$

The final-expression pair is read only in the accepting case. There $\mathrm{dom}(B) \subseteq \mathrm{dom}(A)$ ensures that A's final expression is defined.

This quotient preserves bad terminals. The subset test detects precisely the obligation words. On such a word, the DFA tests both output requirements. The history strategies and arena strategies therefore correspond as in Theorem 4.4. Finite reachability games admit positional winning strategies for the environment and positional safety strategies for the controller. This proves decidability and memory sufficiency with the subset component included.

We can remove that component by Proposition 4.5. Its safe-edge conditions depend only on accepting continuations. The regular-language inclusions defining those conditions remain computable. Its terminal test needs $\mathrm{dom}(B) \subseteq \mathrm{dom}(A)$ and a DFA for the required output relation; both hold here. The resulting controller stores only the reference state and DFA state, since it observes its current B-state and window. Lemma H.3 supplies the factor Q_O. □

Enumerating H decides positive instances of $Q_d^{\natural}$. Enumerating pairs (d, H) does the same for $J^{\natural}$.

H.4. **The output gadgets.** A case is $(\mathcal{T}, \varphi, \tau)$, where $\mathcal{T} \in \mathfrak{S}_K^{\flat}$, φ records the leaf pairs with different input words, and

$$\tau : \Lambda(\mathcal{T}) \longrightarrow \{\mathrm{GAP}, \mathrm{LET}, \mathrm{LEN}_A, \mathrm{LEN}_B, \mathrm{DEAD}\}.$$

The first four are output types. The automaton $\mathcal{W}^{\natural}_{\mathcal{T},\varphi,\tau}$ keeps the skeleton, co-accessibility fields C_λ, and killing rules of Lemma 6.2. A DEAD leaf is unchanged.

At each output leaf, run the two coloured clones. Let $\mathcal{P}_{\mathrm{mk}}$ recognize $\mathsf{P}^*\mathsf{M}\mathsf{S}^*$, and let $\mathcal{P}_{\mathrm{all}}$ recognize P^*. Use complete DFAs with absorbing rejecting sinks. The endpoint maps and use bits satisfy the colour-tracking

conditions of Lemma A.2. All occurrence counts below include only live fresh occurrences.

For GAP, both clones use $\mathcal{P}_{\mathrm{mk}}$. Each clone has a flag η^X, initially zero, with

$$\eta^X + \#^X_{\mathsf{M}} \le 1, \qquad \eta'^X = \eta^X + \#^X_{\mathsf{M}}.$$

On each running row, add

$$\Delta R_\lambda = \#^A_{\mathsf{P}} - \#^B_{\mathsf{P}}, \qquad \Delta G_\lambda = [\eta'^A \neq \eta'^B].$$

For LET, use the same marked clones and flags. Also store $\hat{\gamma} \in \Gamma \cup \{\bot\}$, initially $\bot$. If both marked occurrences are born in one row, require distinct letters. If exactly one is born and $\hat{\gamma} = \bot$, store its letter. If exactly one is born and $\hat{\gamma} \neq \bot$, require its letter to differ from $\hat{\gamma}$. The only weight is

$$\Delta R_\lambda = \#^A_{\mathsf{P}} - \#^B_{\mathsf{P}}.$$

For LEN_A, use $\mathcal{P}_{\mathrm{mk}}$ and its flag on A, and $\mathcal{P}_{\mathrm{all}}$ on B. For LEN_B, interchange these roles. Keep the same signed R_λ weight.

At an output terminal, require both clone states to be final and $C_\lambda = F_B$. Require the use bits to match register occurrences in the final expressions, and require acceptance by each clone's colour DFA. For $N = 2^K$, the coordinates are one R_λ per output leaf, one G_λ per GAP leaf, and one E_π per $\pi \in \varphi$. Hence

$$m \le 2N + \binom{N}{2}.$$

Lemma H.7 (Weights of output leaves)**.** *An accepting walk yields words u_λ, runs β_λ, and branching positions satisfying (K1)–(K3). Every u_λ belongs to* $\mathrm{dom}(B)$*. Output leaves accept, and* DEAD *leaves reject. Exactly the pairs in φ have different input words, with $E_\pi = E(\pi)$.*

Write N^X_λ for the output length of clone X, and write $\gamma_X(c), b_X(c)$ for its letter and birth at rank c. The possible output-leaf weights are exactly

type	*weight*	*rank conditions*
GAP	$(c^A - c^B,\ \lvert b_A(c^A) - b_B(c^B)\rvert)$	$1 \le c^X \le N^X_\lambda$
LET	$c^A - c^B$	$1 \le c^X \le N^X_\lambda, \quad \gamma_A(c^A) \neq \gamma_B(c^B)$
LEN_A	$(c^A - 1) - N^B_\lambda$	$1 \le c^A \le N^A_\lambda$
LEN_B	$N^A_\lambda - (c^B - 1)$	$1 \le c^B \le N^B_\lambda$.

Conversely, any such skeleton data and rank choices extend to an accepting walk with these weights.

Proof. The skeleton part follows from Lemma 6.2. Its co-accessibility invariant supplies the obligation words and the omitted-edge conditions. The input phases determine the unequal pairs and their gaps. These conditions concern only states, edges, and input letters.

By Lemmas A.1 and A.2, each output position corresponds to one live fresh occurrence. A marked clone has exactly one M-position. If its rank is

c^X, there are $c^X - 1$ live P-occurrences. Its flag after row r is

$$\eta^X(r) = [b_X(c^X) \le r].$$

An all-P clone contributes its entire output length. Summing the signed counts gives the four displayed R-weights.

For GAP, the flags differ precisely between the two marked births. Both births occur before the input ends, because finals are fresh-free. Their unequal-flag count is therefore the absolute birth difference, as in Lemma 6.1.

For LET, there are exactly two marked occurrences. If their births coincide, the row compares their letters directly. Otherwise the first birth stores its letter and the second compares against it. No other occurrence changes this test. Thus acceptance requires exactly the displayed letter inequality.

Conversely, colour a marked clone's live occurrences by their ranks: P below the chosen rank, M at it, and S above it. Colour every live occurrence of an unmarked clone P. Dead occurrences can receive arbitrary colours. The converse of Lemma A.2 supplies the endpoint maps and use bits. Set the flags by the displayed birth formula. At a LET leaf, the prescribed distinct letters pass the comparison. The converse skeleton construction supplies the remaining fields. □

H.5. **Bad vectors and the offline obstruction.** Let $V^{\natural}_{\mathcal{T},\varphi,\tau}$ be the weights of accepting walks in the corresponding automaton. Define $\mathrm{BAD}^{\natural}(d, H)$ by the existence of a case and $v \in V^{\natural}_{\mathcal{T},\varphi,\tau}$ satisfying

$$\begin{aligned} R_\lambda &= 0 && \text{at every output leaf,} \\ G_\lambda &\ge H + 1 && \text{at every GAP leaf,} \\ E_\pi &\ge d && \text{for every } \pi \in \varphi. \end{aligned}$$

This predicate is downward closed in both parameters.

Lemma H.8 (Bad vectors)**.** *Under the $K^{\natural}$-promise,*

$$\mathrm{BAD}^{\natural}(d, H) \iff \neg\mathrm{GOOD}^{\natural,\infty}(d, H) \iff \neg\mathrm{GOOD}^{\natural}(d, H).$$

Proof. Take a guarded accepting vector and apply Lemma H.7. At a GAP leaf, $R = 0$ identifies a common rank. Its birth difference exceeds H. If the underlying words agree, their origin distance exceeds H; if they differ, H-goodness already fails. At a LET leaf, $R = 0$ identifies distinct letters at the same rank. At a LEN_A leaf,

$$R = 0 \quad \Longrightarrow \quad N^B = c^A - 1 < N^A.$$

The other length type is symmetric. Give all these leaves type OUT. The E-guards give coherence for unequal input words; equal words need no finite-gap guard. Lemma H.5 now defeats every controller.

Conversely, take the witness supplied by Lemma H.5. Keep its DEAD leaves. At an OUT leaf with equal words, choose a rank with birth difference greater than H and use GAP. Such a rank exists. For unequal words, choose their first unequal letter and use LET, if there is one. If one word is a proper

prefix, mark the first extra position in the longer word. Use LEN_A when A is longer, and LEN_B otherwise. These choices give $R = 0$. Set φ to the unequal input pairs. Lemma H.7 constructs an accepting vector satisfying all guards. The finite-memory equivalence follows from Lemma H.6. □

Lemma 6.4 still holds for all-DEAD cases with $\varphi = \emptyset$. Indeed, its common-word induction uses (K3) and an accepting run on that word. Lemma H.7 supplies both hypotheses.

Lemma H.9 (Offline word obstruction)**.** *Let* OFFW *mean that every* $u \in \mathrm{dom}(B)$ *has an accepting run of* B *with output word* $\lfloor [\![\rho_A(u)]\!] \rfloor$*. Under the* $K^\natural$*-promise, the following are equivalent:*

(1) OFFW *fails.*
(2) Some case with $\varphi = \emptyset$ *and no* GAP *leaf has an accepting vector satisfying all its* $R = 0$ *guards.*
(3) One case and one vector witness $\mathrm{BAD}^\natural(d, H)$ *for every pair* (d, H)*.*

Each condition implies $\neg J^{\natural,\infty}$*.*

Proof. The second condition implies the third because its guards contain neither parameter.

Suppose the third holds. A fixed G-coordinate cannot exceed every H, so there are no GAP leaves. Likewise a fixed finite E-coordinate cannot exceed every d. Thus $\varphi = \emptyset$, and all leaves have one input word $u \in \mathrm{dom}(B)$.

Take any accepting run γ on u. The induction used in Lemma 6.4 finds a leaf whose run equals γ. At each position, γ's next edge has an accepting continuation on the common remaining word. Condition (K3) therefore prevents this edge from being omitted by all compatible leaves. Following those leaves through the successive branching positions gives $\beta_\lambda = \gamma$. This leaf accepts, so it has a letter or length type. Its zero R-weight makes its output word differ from A's. Since γ was arbitrary, OFFW fails.

Conversely, choose a word u witnessing that failure. Take all accepting runs of B on u. There are between one and K. Their prefix tree, with nonbranching portions contracted, gives a shape and branching positions satisfying (K1)–(K2). To check (K3), consider an omitted edge at a represented prefix. If it had an accepting continuation on u, that prefix, edge, and continuation would form another accepting run on u. That run is already among the leaves and would represent the edge at the same prefix. Hence every omitted edge is dead there.

The tree has at most K leaves. Along any path, each sibling subtree accounts for a distinct leaf, so $\sum(\deg - 1) \le K - 1$. Its shape therefore lies in $\mathfrak{S}_K^\flat$. Every leaf has a wrong output word. Choose its letter or length type as in Lemma H.8. All input words equal u, so $\varphi = \emptyset$. Lemma H.7 gives the required vector. Finally, Lemma H.8 rules out every resource pair. □

H.6. **Cutoff and extraction.**

Proposition H.10 (Cutoff under $\mathbf{G}^\natural$)**.** *Let $Z \geq 1$, $W \geq 1$, and m bound the state counts, edge-weight norms, and coordinate counts of all cases. Put*

$$T_*^\natural = m!\,(Z^2W+1)^m + 1, \qquad H_*^\natural(d) = m!\,(Z^2W+d+1)^m + 1.$$

Then

$$J^\natural \iff \mathrm{GOOD}^\natural(T_*^\natural, T_*^\natural), \qquad Q_d^\natural \iff \mathrm{GOOD}^\natural(d, H_*^\natural(d)).$$

Both equivalences also hold with arbitrary memory.

Proof. By Lemma 6.5, each case has a finite union of weight sets

$$a + \mathbb{N}V, \qquad \|a\|_\infty \leq Z^2W, \quad \|v\|_\infty \leq ZW \quad (v \in V).$$

Accepting states have no outgoing rows, as required by that lemma.

Fix one piece, and write its weight as $y = a + \sum_v n_v v$. For diagonal resources, introduce a nonnegative variable t and nonnegative slack variables. The guards become

$$\begin{aligned} y_{R_\lambda} &= 0, \\ y_{G_\lambda} - t - \sigma_\lambda &= 1, \\ y_{E_\pi} - t - \sigma_\pi &= 0. \end{aligned}$$

There are at most m rows. Coefficients and constants have absolute value at most $Z^2W + 1$.

Apply the integer-ray and vertex arguments from Proposition 6.7. If this system has an integer point and t is unbounded on its real solutions, its recession cone contains an integer ray with positive t-coordinate. Adding multiples of that ray to the integer point gives bad vectors at arbitrarily large diagonal resources. Downward closure then gives $\mathrm{BAD}^\natural(d, H)$ for every d, H.

Otherwise Lemma 6.6 bounds every feasible value by

$$t \leq m!\,(Z^2W+1)^m.$$

Thus a bad vector at $T_*^\natural$ must belong to an unbounded piece. By Lemma H.8, such a piece rules out $J^{\natural,\infty}$. If no bad vector exists there, Lemma H.6 gives the required finite-memory controller.

A piece with no GAP leaf and $\varphi = \emptyset$ has no row containing t. This case remains valid. From any integer solution, adding e_t gives another solution. The coordinate vector e_t itself is an integer recession ray. Lemma H.9 identifies the corresponding obstruction.

For fixed d, keep the equations $y_{E_\pi} - \sigma_\pi = d$ and let t represent H. The constants are bounded by $Z^2W + d + 1$. The same argument bounds t by $H_*^\natural(d) - 1$ unless it is unbounded. This includes systems with no row containing t, even when they have finite-gap guards. Downward closure and Lemma H.8 give the second equivalence. □

Proof of Theorem H.2. Apply Lemma H.3. Weber and Seidl's ambiguity procedure [WS91], together with the run-count test of Lemma 8.21, computes K_O for the underlying edge automaton of $B \times D_O$. Enumerate the finitely many witness cases and compute common bounds Z, W, m.

For $J^\natural$, solve the game of Lemma H.6 at $(T_*^\natural, T_*^\natural)$. For $Q_d^\natural$, solve it at $(d, H_*^\natural(d))$. Proposition H.10 proves correctness. A winning positional strategy yields the controller and its stated memory, using the safe-edge quotient in Lemma H.6. Projection adds Q_O for the original machine. The same lemma proves the claims for fixed resources and arbitrary memory. □

H.7. **Complexity.**

Proposition H.11 (Complexity under $\mathbf{G}_O^\natural$)**.** *Assume the $K^\natural$-promise with $K = K_O$, and put $N = 2^K$. Let s_O be the size of $(A, B \times D_O)$. Thus $s_O = O(s|Q_O|)$; when $O = \mathrm{dom}(B)$ is treated directly, take $s_O = s$. Use k, μ_A, μ_B, μ as in Section 8. The resource parameters may be supplied in binary.*

(1) The witness automata admit bounds

$$\log_2 Z \le 7N s_O + 2N^2, \qquad W \le \max(\mu, 1), \qquad m \le 2N + \binom{N}{2}.$$

Consequently,

$$\log_2 T_*^\natural = O(N^3 s_O + N^4), \qquad \log_2 H_*^\natural(d) = O(N^3 s_O + N^4 + N^2 \log(d+1)).$$

(2) The decision bounds are

$$\begin{aligned} \mathrm{BAD}^\natural(d,H) &\in \mathrm{NSPACE}\big(O(N s_O + N^2 + N\log(H+2) + N^2\log(d+2))\big),\\ Q_d^\natural &\in \mathrm{NSPACE}\big(O(N^4 s_O + N^5 + N^3\log(d+2))\big),\\ J^\natural &\in \mathrm{NSPACE}\big(O(N^5 s_O + N^6)\big). \end{aligned}$$

For fixed K, and also for $K = O(\log s_O)$, all three problems $\mathrm{GOOD}^\natural$, $Q_d^\natural$, and $J^\natural$ lie in PSPACE. For fixed K,

$$J^\natural \in \mathrm{NSPACE}(O(s_O)) \subseteq \mathrm{DTIME}(2^{O(s_O)}).$$

(3) For every fixed $K \ge 1$, $\mathrm{GOOD}^\natural(d, H)$ is PSPACE-complete when H is supplied as input. Every lower bound transferred in Proposition 8.17 also transfers here, with the same ambiguity bound and $O = \Sigma^*$.

(4) On positive instances, controller memory is at most

$$3(|Q_A|+1)|Q_O|\, 2^{|\mathcal{N}_{T_*^\natural}|} = 2^{2^{O(N^3 s_O + N^4)}},$$

where $\mathcal{N}_H$ is the full four-type monitor. The factor $|Q_O|$ is omitted when $O = \mathrm{dom}(B)$ is treated directly.

Proof. We first work after the product reduction. At an output leaf, store its status, two control states, C_λ, the use bits, endpoint maps, and at most two birth flags. A LET leaf also stores $\hat{\gamma}$. Use bits contribute 2^k choices and endpoint maps at most 9^k. The stored letter contributes $|\Gamma|+1$ choices. The type belongs to the case.

With the size convention of Section 8, these fields need at most $7s_O$ bits per output leaf. A DEAD leaf needs at most $1+2|Q_B|$ bits. The unchanged node and pair phases contribute at most $2N^2$ bits. This gives the stated bound on Z. Each R changes by at most μ in one row; each G and E changes by at most one. The coordinate count was established above. Substitution in Proposition H.10 gives the two logarithmic bounds.

For fixed resources, use the implicit graph from Proposition 8.18, with the new leaf types. Keep exact counters

$$R_\lambda \in [-\mu_B H, \mu_A H]$$

at output leaves. Saturate each G_λ at $H+1$ and each E_π at d. A target is an accepting state with every $R_\lambda = 0$, every $G_\lambda = H+1$, and every $E_\pi = d$. A path to a target gives a bad vector, since saturated values certify the required lower bounds.

For the converse, the choice of ranks matters. Take any bad vector and its coherent witness. At each output leaf choose the least bad rank: the first rank with different letters or birth difference greater than H. If no common rank is bad and the lengths differ, choose the first rank beyond the shorter word. Retype the leaf by this failure, choosing either applicable type when letter and birth failures coincide. Changing the type is allowed because cases are existentially quantified.

Every smaller rank has equal letters and birth difference at most H. After row r, the exact rank counter is the difference between the numbers of these smaller ranks already born in the two clones. Lemma 4.1 bounds it by $[-\mu_B H, \mu_A H]$. For LEN_A, the marked rank is $c = N^B + 1$. Every position of the unmarked B-clone is therefore below c. For LEN_B, it is $c = N^A + 1$, and every position of the unmarked A-clone is below c. Thus the same counter bound applies to both length types. All counters end at zero. Lemma H.7 realizes these choices inside the bounded graph, proving reachability.

A graph state uses

$$O(Ns_O + N^2 + N\log(H+2) + N^2\log(d+2))$$

bits. The extra stored letters require only $O(N \log s_O)$ bits, and a letter comparison uses $O(\log s_O)$. Guessing a path with a graph-size clock proves the first space bound. Nondeterministic space is closed under complement at these bounds. Using Lemma H.8 and substituting the cutoffs gives the displayed bounds for $Q_d^\natural$ and $J^\natural$. If K is fixed, N is constant. If $K = O(\log s_O)$, N is polynomial in s_O. The PSPACE and fixed-K time conclusions follow.

For hardness, take a **G**-instance and set $O = \Sigma^*$. All accepting runs already produce A's word, so its successful controllers are exactly the original successful controllers. The ambiguity is unchanged. Theorem 8.10 and Corollary 8.11 therefore give PSPACE-hardness with H in the input, even for fixed d and ambiguity one. The same argument transfers the remaining lower bounds in Proposition 8.17. In particular, Theorem 3.4 and Lemma H.6 give RE-completeness at linear ambiguity.

Finally, the monitor of Theorem 4.2 has

$$|\mathcal{N}_H| \le 16 \cdot 18^k(|\Gamma| + 1)((\mu_A + \mu_B)H + 1)(H + 2) + 1.$$

Its full deterministic checker has at most $3 \cdot 2^{|\mathcal{N}_H|}$ states. Lemma H.6 gives the first memory bound. The estimate for $T_*^\natural$ gives the double exponential. Projection back to B multiplies memory by $|Q_O|$. □

H.8. **Two examples.**

Example H.12 (Lookahead for the output word)**.** Let $\Sigma = \Gamma = \{a, b\}$, with one register X in each machine. The reference has states p_0, p_1, p_2, starts at p_0, and accepts only at p_2, with final expression X. For $x \in \Sigma$, its edges are

$$p_0 \xrightarrow{x/\mathrm{id}} p_1, \qquad p_1 \xrightarrow{x/X:=Xx} p_2, \qquad p_2 \xrightarrow{x/\mathrm{id}} p_2.$$

The machine B starts at s_0 and has states t_g, f_g for $g \in \{a, b\}$. Only the f_g are final, with final expression X. Its edges are

$$s_0 \xrightarrow{x/X:=Xg} t_g, \qquad t_g \xrightarrow{x/\mathrm{id}} f_g, \qquad f_g \xrightarrow{x/\mathrm{id}} f_g.$$

Both domains are $\Sigma^{\ge 2}$, and every obligation word has two accepting runs.

At lookahead zero, the first window is identical on aa and ab. Choosing $g = a$ fails on ab, while choosing $g = b$ fails on aa. This defeats arbitrary memory for every H. Equivalently, these two failing runs form a coherent two-leaf LET witness, branching at position one with gap zero.

With lookahead one, choose g equal to the second input letter. The resulting output is correct. Its birth is one, while the reference birth is two. Hence $\mathrm{GOOD}^\natural(1, 1)$ holds. Every correct run has this birth difference, so $\mathrm{GOOD}^\natural(d, 0)$ fails for all d.

Example H.13 (One word defeats every controller)**.** Keep the reference from Example H.12, and retain only the $g = a$ branch of B. The ambiguity is one. On ab, the sole accepting run outputs a, while A outputs b. A single LET leaf at rank one has $R = 0$ and $\varphi = \emptyset$. Its vector satisfies every resource pair, as in Lemma H.9.

Use of AI tools

The author set the questions and the scope, and checked the statements and the key steps of the proofs. AI referees checked the full proofs: GPT-6.1 Sol, GPT-6 Astra and ChatGPT Pro from OpenAI, and Claude Opus 5.5 from Anthropic. Claude Opus 5.5 agents also drafted proofs and ran

computations for the research record. The idea notes behind the prose were drafted by an AI from the author's briefing, then reviewed and adopted by the author. The prose was drafted with AI from those notes. For each section and appendix the author chose among drafts by GPT-6 Astra, Claude Fable 5.1, Gemini 4 and ChatGPT Pro; the text uses the Astra and Fable drafts. ChatGPT Pro reviewed the assembled text, and Claude Opus 5.5 applied the corrections the author accepted. The AI tools are not authors. Responsibility for the contents rests with the author.

References

[AD11] R. Alur and J. V. Deshmukh. Nondeterministic streaming string transducers. In *ICALP 2011*, volume 6756 of *LNCS*, pages 1–20, 2011. doi:10.1007/978-3-642-22012-8_1.

[BKM+19] S. Bose, S. N. Krishna, A. Muscholl, V. Penelle, and G. Puppis. On synthesis of resynchronizers for transducers. In *MFCS 2019*, volume 138 of *LIPIcs*, pages 69:1–69:14, 2019. arXiv:1906.08688v2, doi:10.4230/LIPIcs.MFCS.2019.69.

[BMPP18] S. Bose, A. Muscholl, V. Penelle, and G. Puppis. Origin-equivalence of two-way word transducers is in PSPACE. In *FSTTCS 2018*, volume 122 of *LIPIcs*, pages 22:1–22:18, 2018. arXiv:1807.08053v3, doi:10.4230/LIPIcs.FSTTCS.2018.22.

[Boj14] M. Bojańczyk. Transducers with origin information. In *ICALP 2014*, volume 8573 of *LNCS*, pages 26–37, 2014. arXiv:1309.6124, doi:10.1007/978-3-662-43951-7_3.

[Dud23] A. Dudenhefner. Constructive many-one reduction from the halting problem to semi-unification (extended version). *Logical Methods in Computer Science*, 19(4):22, 2023. arXiv:2208.13428, doi:10.46298/lmcs-19(4:22)2023.

[EFJ18] L. Exibard, E. Filiot, and I. Jecker. The complexity of transducer synthesis from multi-sequential specifications. In *MFCS 2018*, volume 117 of *LIPIcs*, pages 46:1–46:16, 2018. arXiv:1905.03560, doi:10.4230/LIPIcs.MFCS.2018.46.

[FJL+25] E. Filiot, I. Jecker, C. Löding, A. Muscholl, G. Puppis, and S. Winter. Finite-valued streaming string transducers. *TheoretiCS*, 4:1, 2025. Conference version: LICS 2024. arXiv:2405.08171v6.

[FJLW23] E. Filiot, I. Jecker, C. Löding, and S. Winter. A regular and complete notion of delay for streaming string transducers. In *STACS 2023*, volume 254 of *LIPIcs*, page 32, 2023. arXiv:2205.04287v3.

[HKT10] M. Holtmann, Ł. Kaiser, and W. Thomas. Degrees of lookahead in regular infinite games. In *FoSSaCS 2010*, volume 6014 of *LNCS*, pages 252–266, 2010. Journal version: Logical Methods in Computer Science 8(3:24), 2012, doi:10.2168/LMCS-8(3:24)2012. doi:10.1007/978-3-642-12032-9_18.

[KM20] D. Kuperberg and J. Martens. Regular resynchronizability of origin transducers is undecidable. In *MFCS 2020*, volume 170 of *LIPIcs*, pages 58:1–58:14, 2020. arXiv:2002.07558v3, doi:10.4230/LIPIcs.MFCS.2020.58.

[KMRW26] S. Kiefer, F. Mazowiecki, A. Ryzhikov, and A. Wiśniewski. Stronger bounds on the degree of ambiguity of finite automata, 2026. Preprint, not refereed. arXiv:2610.08525v1.

[KZ16] F. Klein and M. Zimmermann. How much lookahead is needed to win infinite games? *LMCS*, 12(3:4), 2016. Corrected version. Conference version: ICALP 2015. arXiv:1412.3701v5.

[LR13] C. Löding and S. Repke. Decidability results on the existence of lookahead delegators for NFA. In *FSTTCS 2013*, volume 24 of *LIPIcs*, pages 327–338,

2013. To be read with the erratum [Lö23]. doi:10.4230/LIPIcs.FSTTCS.2013.327.

[Lö23] C. Löding. Errata for: Decidability results on the existence of lookahead delegators for NFA. Errata document, linked from the author's publication page at www.lics.rwth-aachen.de, 2023. 22 February 2023.

[RS07] B. Ravikumar and N. Santean. On the existence of lookahead delegators for NFA. *International Journal of Foundations of Computer Science*, 18(5):949–973, 2007. doi:10.1142/S0129054107005078.

[Win21] S. Winter. Decision problems for origin-close top-down tree transducers. In *MFCS 2021*, volume 202 of *LIPIcs*, pages 90:1–90:16, 2021. arXiv:2107.02591v1, doi:10.4230/LIPIcs.MFCS.2021.90.

[WS91] A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991. doi:10.1016/0304-3975(91)90381-B.