Optimal Sequential Flows

Hugo Gimbert
CNRS, LaBRI, Université de Bordeaux, France

Corto Mascle
MPI-SWS, Kaiserslautern, Germany

Patrick Totzke
University of Liverpool, UK

Abstract

We provide a new algebraic technique to solve the sequential flow problem in polynomial space. The task is to maximize the flow through a graph where edge capacities can be changed over time by choosing a sequence of capacity labelings from a given finite set. Our method is based on a novel factorization theorem for finite semigroups that, applied to a suitable flow semigroup, allows to derive small witnesses. This generalizes to multiple in/output vertices, as well as regular constraints.

2012 ACM Subject Classification Theory of computation → Network flows; Computing methodologies → Symbolic and algebraic algorithms; Computing methodologies → Algebraic algorithms; Theory of computation → Algebraic language theory

Keywords and phrases Network Flow, Sequential Flow, Semigroup Factorization

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Determining the maximal flow through a network under capacity constraints is a classical optimisation problem [10].

The Sequential flow problem is a dynamic variant in which the edge capacities are not static but change with (discrete) time. At any date one can select the labelling of edges by capacities from finitely many options: We are given a finite set $A \subseteq (\mathbb{N} \cup \{\omega\})^{V \times V}$ of which each element $a \in A$ prescribes capacities for every edge in the directed graph[1]. Every length-ℓ capacity word $a_1 a_2 \cdots a_\ell \in A^*$ determines a *pipeline*, a graph of size $(\ell + 1) \cdot |V|$ together with edge capacities where at time $1 \leq i \leq \ell$, the edge $v \to v'$ has capacity $a_i(v, v')$. The Sequential flow problem asks to determine the supremum of flow values through any such pipeline.

Notice that there is no limit on the length k of the capacity words and therefore, the optimal sequential flow can be unbounded even if there is no finite word witnessing this.

▶ **Example 1.** Consider the graph with vertices $V = \{v_1, v_2, v_3, v_4\}$, with source $v_s = v_1$ and target $v_t = v_4$, and capacity constraints $A = \{a, b\}$ as depicted on the top half of Figure 1. In both capacities a and b, there is no path from the source v_1 to the target v_4, it is necessary to combine them sequentially in order to enable positive flow from v_1 to v_4. This can be achieved for instance using the capacity word $abba$ as illustrated on the right of Figure 1.

Even more flow can be transported from v_1 to v_4 through longer pipelines. For every $n > 0$ the pipeline for capacity word $ab^n a$ has a flow of (maximal) value n, as depicted in Figure 2 The *sequential* flow for A is therefore unbounded.

[1] A capacity / flow value ω means unbounded, i.e., finite but arbitrarily large.

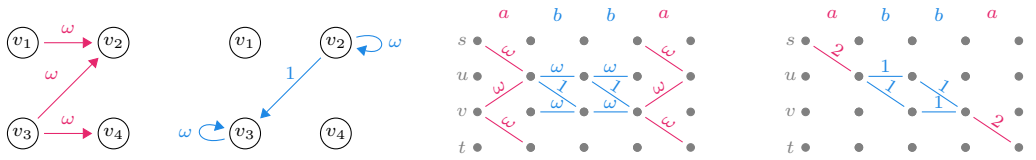


Figure 1 The two capacities a, b from Example 1, the pipeline, and the optimal flow for $abba$.

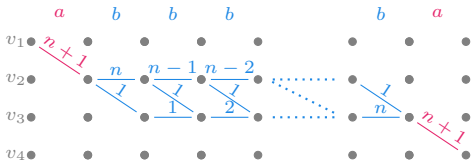


Figure 2 A flow of value $n+1$ through the pipeline $ab^{n+1}a$.

Background Early works on *dynamic flows* considers flows in directed networks where edges have fixed capacities as well as transit times, and where associated costs are minimised (see. e.g. [9] and [2] for a survey and applications).

Closest to our setting, Akrida et al. [1] compute maximal flows through temporal networks [15, 16], where the edge capacities themselves are a function of (discrete and bounded) time, i.e., a fixed capacity word. They devise a polynomial-time algorithm to compute the maximal *temporal* flow that satisfies these constraints, and show a temporal version of the max-flow min-cut theorem.

In contrast to these works, in the Sequential flow problem neither the time horizon nor the capacities at given times are fixed. Instead, akin to planning or scheduling problems, we can choose the capacity word to maximise the flow through the network. Sequential flows in the form discussed here have been introduced in [5] in the context of distributed computing and has applications for controlling populations of Markov Decision Problems [11]. They have further applications in logics: it can be observed that the *commutative lossy tiling problem* defined and shown decidable in [3] is inter-reducible with the Sequential flow problem. Colcombet et al. [6] showed that it is PSPACE-hard, and decidable in exponential space wether the optimal sequential flow is unbounded[2]. The upper bound was achieved by an exponential reduction to the unboundedness problem of distance automata, for which a PSPACE upper bound is known [17].

Contributions We provide a new, simple and optimal solution for the Sequential flow problem. We show how to compute the precise maximal sequential flow values in polynomial space (Theorem 22), thus matching the lower bound of [6] with an upper bound for a much more general problem. Our technique is adapted to further generalisations: We derive the same upper bounds for the Sequential flow problem where sequential flows must be witnessed by capacity words from a given regular language (Theorem 37). This setting directly generalizes the classical Max flow problem as well as the setting in Akrida et al. [1]. We also show how to solve a generalization called the Fair sequential flow

[2] This is called the "simple sequential flow problem" in [6].

PROBLEM , where multiple source-target pairs are given and the flow should be routed equally along all those edges (Theorem 36).

Our contributions are based on new algebraic contributions of independent interest: In particular, we provide a factorization technique for general finite semigroups, where we show the existence of small *summaries* (Theorem 26). We also show the existence of $\sharp$-*summaries* of polynomial height for elements of the *flow semigroup*, a particular stabilization monoid [4] used to witness unbounded sequential flows. This allows to ultimately obtain optimal upper bounds on the largest possible value of a finite solution to the SEQUENTIAL FLOW PROBLEM (Theorem 16).

2 Sequential Flows

We first recall the definition of flows and their optimisation problem.

Flows Fix a finite set of vertices V and two distinct vertices v_s and v_t referred to as *source* and *target*, respectively. A *flow* $f \in \mathbb{R}^{V\times V}$ is a mapping of edges to non-negative reals which satisfies capacity– and flow conservation constraints as follows.

A *capacity constraint* $a \in (\mathbb{N} \cup \{\omega\})^{V\times V}$ assigns to each edge a capacity, respecting that all incoming edges to the source and from the target have capacity 0.

$$\forall (v, v') \in V^2, (v' = v_s \vee v = v_t) \implies a(v, v') = 0 \ . \tag{1}$$

A flow f satisfies the capacity constraint a if

$$\forall v, v' \in V, \quad f(v, v') \leq a(v, v') \tag{2}$$

It satisfies the *flow conservation* constraints if

$$\forall v \in V \setminus \{v_s, v_t\}, \quad \mathrm{out}(f)(v) = \mathrm{in}(f)(v) \tag{3}$$

where $\mathrm{out}(f)(v) = \sum_{v'\in V} f(v, v')$ and $\mathrm{in}(f)(v) = \sum_{v'\in V} f(v', v)$.

Intuitively, a flow determines rate of goods flowing along each of the edges, and the capacity of an edge (v, v') is a predetermined bound on the admissible rate that can flow from v to v'. A capacity of $a(v, v') = 0$ means that nothing at all can flow, and a capacity of $a(v, v') = \omega$ means that an arbitrary finite amount can flow.

The *value* of a flow f is $|f| = \mathrm{out}(f)(v_s)$, the cumulative flow out of the source vertex. The MAX FLOW PROBLEM asks to compute the maximal value of any flow.

MAX FLOW PROBLEM
Given a capacity constraint $a \in (\mathbb{N} \cup \{\omega\})^{V\times V}$.
Maximise $|f|$ under constraints in equations (2) and (3).

Due to the presence of edges with unbounded capacities and because every flow must have a finite value, there may not exist flows of maximal value.

Sequential flows The SEQUENTIAL FLOW PROBLEM is a dynamic variant of this setting in which the maximizer gets to pick a fresh capacity constraint at any unit time. Intuitively, a sequential flow still represents the rate of goods flowing along the edges of the graph, but at any moment in time, both capacity constraints and flow conservation dynamically reflect maximizer's current choice of capacities.

Assume a finite set $A \subseteq (\mathbb{N} \cup \{\omega\})^{V \times V}$ of capacity constraints. A sequence $f = f_1 f_2 \dots f_\ell \in \left(\mathbb{R}^{V \times V}\right)^*$ is a *sequential flow* if $\forall 0 < i \leq \ell$,

$$\exists a_i \in A, \forall v, v' \in V, \quad f_i(v, v') \leq a_i(v, v') \tag{2'}$$

$$\forall v \in V \setminus \{v_s, v_t\}, \quad \text{in}(f_i)(v) = \text{out}(f_{i+1})(v) \tag{3'}$$

A sequential flow $f = f_1 f_2 \dots f_\ell$ dictates (at least) one *capacity word* $w = a_1 a_2 \dots a_\ell \in A^*$ that witnesses f satisfying the sequential capacity conditions (2′). We will refer to f as a sequential flow *over* capacity word w. The *value* of f is

$$|f| = \text{in}(f_\ell)(v_t) = \text{out}(f_1)(v_s),$$

the input flow to the target at the latest time ℓ and the output of the source at time 1. We want to optimize the supremum value of any sequential flow.

SEQUENTIAL FLOW PROBLEM
Given a finite set $A \subseteq (\mathbb{N} \cup \{\omega\})^{V \times V}$ of capacity constraints.
Determine the *optimal sequential flow* optSeqFlow $= \sup\{|f| : f \text{ is a sequential flow}\}$.

There is a natural connection between the SEQUENTIAL FLOW PROBLEM and the MAX FLOW PROBLEM. Every capacity word $a_1 \dots a_\ell$ defines an instance of MAX FLOW PROBLEM in the corresponding pipeline, with source $(v_s, 0)$ and target (v_t, ℓ). This instance has a maximal flow, and optSeqFlow is simply the supremum of those maximal flows over all capacity words $a_1 \dots a_\ell$.

However, the SEQUENTIAL FLOW PROBLEM is *not* a linear program because ℓ is not bounded, and hence the number of constraints (2′) and (3′) is not bounded a priori.

Despite their apparent similarity, there is no simple algorithmic reduction between these two optimisation problems. Indeed, for an instance of the SEQUENTIAL FLOW PROBLEM with singleton set of capacities $\{c\}$, the optimal sequential flow value can be the same, strictly larger, or strictly greater than the maximal flow value if c is interpreted as an instance of the classical MAX FLOW PROBLEM.

▶ **Example 2.** Let $V = \{v_1, v_2, v_3, v_4\}$ and consider capacity letters a, b, c as depicted in Figure 3. If c is considered as an instance of MAX FLOW then only 1 unit of flow can be transported from source $v_s = v_1$ to target $v_t = v_4$, using the edge (v_2, v_3) at its maximal capacity. The values of maximal flows through capacity words $c, cc, ccc, cccc$ are $0, 0, 1, 2$, respectively, and the max flow through any capacity word of the form $c^n, n > 4$ remains 2. The optimal sequential flow given set of capacities $A = \{c\}$ is therefore 2.

Consider now only capacity d. A flow with maximal value $|f| = 2$ from the source $v_s = v_1$ to the target $v_t = v_4$ is $f(v_1, v_3) = f(v_2, v_4) = f(v_1, v_2) = f(v_3, v_4) = 1$. Similarly, the sequential flow ff for capacity word $dd \in A^*$ has value $|ff| = 2$. However, the capacity word ddd only admits the trivial sequential flow of value 0. The optimal sequential flow given set of capacities $A = \{d\}$ is therefore 2.

Consider now only capacity e. The maximal value of a flow from v_1 to v_4 is 2 whereas the optimal sequential flow is only 1, because for any $n \geq 0$ there is at most one path of length n, and the minimal capacity along these paths is 1. The optimal sequential flow given set of capacities $A = \{e\}$ is therefore 1.

Finally, to demonstrate that combining different capacity letters may be required for the optimal sequential flow, consider the set of capacities $A = \{c, e\}$. The optimal sequential

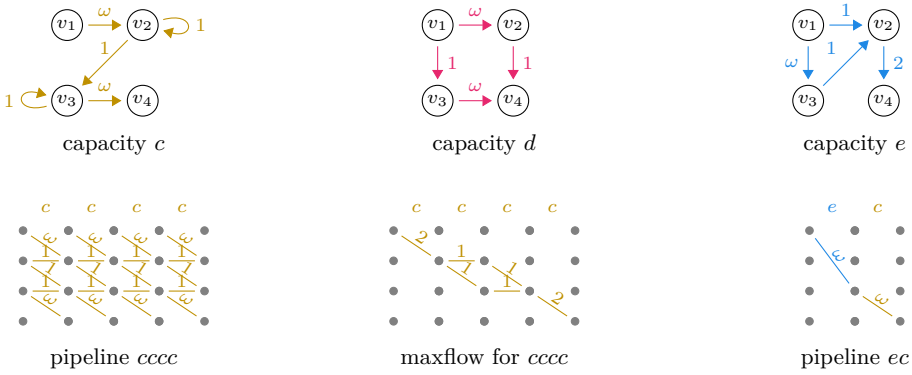
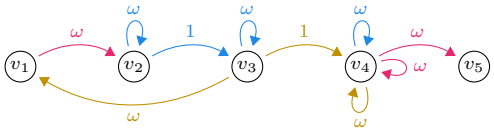


Figure 3 Capacities c, d, e from Example 2, pipeline $cccc$ and its maximal flow, and pipeline ec.

flow value ω can be witnessed by a single capacity word ec and a family of sequential flows $f_n = f_{n,1}f_{n,1}$ with $f_{n,1} = f_{n,2} = n$ of value $|f_n| = n$.

As established in both examples so far, there may not exist sequential flows of maximal value because there are in fact sequential flows of arbitrarily high values. In that case we call the optimal sequential flow *unbounded* and write optSeqFlow $= \omega$. This can be witnessed in two ways: either by a single pipeline such as at the end of Example 2, or, as in Example 1, by a family of pipelines of growing length and maximal flow value. In Example 1, the optimal sequential flow value of ω cannot be witnessed by any finite pipeline and is instead witnessed by a family $ab^n a$ of words that iterate the capacity constraint b arbitrarily, but finitely often. Our final example demonstrates that such witnesses may require more complex nested iterations.

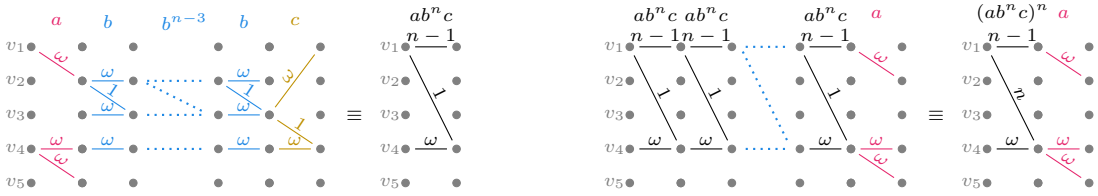
▶ **Example 3.** Take sets $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $A = \{a, b, c\}$ of vertices and capacity constraints as depicted below, where non-zero capacities of a, b, c are shown in red, blue, and yellow.



The optimal sequential flow from source $v_s = v_1$ to target $v_t = v_5$ is ω, yet no finite capacity word witnesses this. To witness a sequential flow of value n, a capacity word must be of the form $(ab^{\geq n}c)^{\geq n}a$. The combined capacities for the word $ab^n c$ is shown in Figure 4 (left). This allows a flow of n from v_1 to v_3 (using ab^n); then to transfer one unit to v_3 (using c, which empties v_3). Iterating this prefix n times allows a flow of n units to v_4, at which point all can flow in one step towards the target v_5 (via a, see the right half of Figure 4).

3 Solving the Sequential flow problem

We present a solution to the Sequential flow problem in two stages. The first stage, described in Section 3, is qualitative: we determine whether the instance is unbounded, i.e., whether there exists sequential flows of arbitrarily high value. The key for that is to abstract the exact computation of values by mean of a finite algebraic structure called the *flow semigroup*. The elements of this semigroup are enumerated using Algorithm 1, which



pipeline ab^nc and its combined effect

pipeline for $(ab^nc)^na$

Figure 4 The pipelines from Example 3. The pipeline for ab^nc and its shortened representation (seen on the left) is iterated another n times in the pipeline for $(ab^nc)^na$ (seen on the right).

searches for a witness of unboundedness. The second stage is quantitative: it is performed by Algorithm 2, which computes the maximal value of sequential flows, assuming they are bounded. Both stages can be carried out in polynomial space. A key component for this upper bound in the second stage is the proof that when sequential flows are bounded the supremum is at most exponential in the number of vertices.

For the qualitative stage, we will abstract the exact values of capacity constraints and only consider wether those values are 0, finite or ω.

We make use of the *maxmin semiring*

$$\mathbb{M} = (\{0, 1, \omega\}, \max, \min) \text{ with } 0 < 1 < \omega$$

There is a natural structure of semigroup on $\mathbb{M}^{V\times V}$, using the usual matrix product over this semiring. For $x, y \in \mathbb{M}^{V\times V}$, and $v, v' \in V$,

$$(x \cdot y)(v, v') = \max_{v''\in V} \left(\min\left(x\left(v, v''\right), y\left(v'', v'\right)\right)\right) \quad .$$

Every capacity constraint $a \in \mathbb{N}^{V\times V}$ is naturally abstracted as a matrix $x_a \in \mathbb{M}^{V\times V}$ by losing precision: 0 and ω are preserved while finite positive numbers are mapped to 1.

▶ **Example 4.** In Example 1 there are two capacity constraints a and b. Their abstractions, as well as the product thereof, are as follows:

$$x_a = \begin{pmatrix} 0 & \omega & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \omega & 0 & \omega \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad x_b = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \omega & 1 & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad x_a \cdot x_b = \begin{pmatrix} 0 & \omega & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \omega & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Notice that $a = x_a$ and $b = x_b$ coincide with their abstractions since all finite coordinates are equal to 0 or 1.

The matrix product allows to keep track of which paths have finite or ω-capacity. For every $n \geq 1$, the product $x_a x_b^n x_a$ can be computed easily: since x_b is idempotent, i.e. $x_b^2 = x_b$, we have

$$x_a \cdot x_b^n \cdot x_a = x_a \cdot x_b \cdot x_a = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

This simple computation tells us that starting from the source v_s (the first line), any sequential flow through the pipeline $x_a x_b^n x_a$ can carry only a finite flow to the target v_t (the last column),

because the top right coefficient is 1. However, this finite representation of the pipeline $x_a x_b^n x_a$ misses an important point: although the flow from v_s to v_t is finite for every $n > 0$, it is actually unbounded and grows with n (cf. Figure 2).

In order to take this phenomenon into account, we introduce an extra operation on idempotent elements, i.e. the elements $e \in \mathbb{M}^{V\times V}$ such that $e = e^2$. This operation computes a new idempotent element $e^\sharp \in \mathbb{M}^{V\times V}$. Intuitively, the element $e^\sharp$ is an abstraction of the sequence of pipelines $(e^n)_{n\in\mathbb{N}}$ and should be thought of as "using e many times".

Before we proceed to define this operation, let's observe that the possible effect that iterating idempotents can have on the maximal flow between any two vertices.

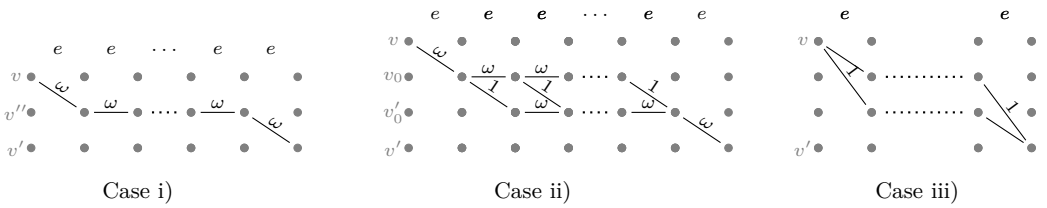


Figure 5 The figure illustrates Lemma 5, which classifies the three possible long-term behaviours of an edge (v, v') of an idempotent e, when $e(v, v') \neq 0$. Case i) is $e(v, v') = \omega$ and case ii) and iii) occur when $e(v, v') = 1$.

▶ **Lemma 5** (Flow-carrying Edges of idempotent elements). *Let $e \in \mathbb{M}^{V\times V}$ such that $e = e^2$, and $v, v' \in V$ such that $e(v, v') > 0$. For $n \geq 1$, let K_n denote the optimal flow value from v to v' in the pipeline e^n. Exactly one of the following holds.*

i) $K_n = \omega$, for every $n \geq 1$. This holds iff $e(v, v') = \omega$ and there exists $v'' \in V$ such that

$$\omega = e(v, v'') = e(v'', v'') = e(v'', v') \ .$$

ii) $n - 2 \leq K_n \leq n|V|$, for every $n \geq 1$. This holds iff $e(v, v') = 1$ and $e(v, v') = 1$ and there exists v_0, v_0' such that

$$e(v_0, v_0') = 1 \text{ and } \omega = e(v, v_0) = e(v_0, v_0) = e(v_0', v_0') = e(v_0', v') \ .$$

iii) $1 \leq K_n \leq 2|V|$, for every $n \geq 1$. This holds iff $e(v, v') = 1$ and $e(v, v') = 1$ and for every $v_0, v_0' \in V$,

$$e(v_0, v_0') \geq 1 \implies (e(v, v_0) \leq 1 \text{ or } e(v_0', v') \leq 1) \ .$$

The following definition of iterations of idempotents explicitly distinguishes the three cases of Lemma 5 to summarise an "ever growing" finite maxflow (case ii) as a new ω.

▶ **Definition 6** (iteration of an idempotent). *Let $e = e^2$ be an idempotent element of $\mathcal{F}$. A pair $(v, v') \in V^2$ such that $e(v, v') = 1$ is called* unstable *iff there exists $v_0, v_0' \in V$ such that $e(v, v_0) = \omega$, $e(v_0, v_0') = 1$ and $e(v_0', v') = \omega$, and* stable *otherwise.*

Then the iteration *of e, denoted $e^\sharp$, is defined by*

$$e^\sharp(v, v') = \begin{cases} e(v, v') & \text{if } e(v, v') \in \{0, \omega\} \\ 1 & \text{if } e(v, v') = 1 \text{ and } (v, v') \text{ is stable in } e \\ \omega & \text{if } e(v, v') = 1 \text{ and } (v, v') \text{ is unstable in } e. \end{cases}$$

An idempotent e is called unstable *if it has an unstable pair, and* stable *otherwise.*

Note that if an edge is unstable then it does not satisfy condition iii) of Lemma 5, thus it satisfies condition ii) of the same Lemma. Note also that e is unstable if and only if $e \neq e^\sharp$.

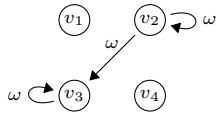
▶ **Lemma 7.** *The iteration of an idempotent e is stable, i.e., $(e^\sharp)^\sharp = e^\sharp$.*

Our main algebraic tool is the flow semigroup, a finite structure obtained by application of the product and iteration to the abstract capacities until saturation. This flow semigroup is in fact an example of a *stabilisation monoid* (upon addition of a neutral element), which are monoids with a stabilisation operator [4].

▶ **Definition 8** (flow semigroup). *The flow semigroup, denoted $\mathcal{F}$, is the smallest subset of $\mathbb{M}^{V\times V}$ which contains all abstracted capacity constraints $\{x_a \mid a \in A\}$ and is closed under the matrix product and the iteration operation.*

▶ **Example 9.** Continue with Example 4. The flow semigroup $\mathcal{F}$ contains x_a and x_b and since $x_b^2 = x_b$, it also contains $x_b^\sharp$. The only capacity-1 edge in x_b is unstable since $x_b(v_2, v_2) = \omega, x_b(v_2, v_3) = 1$ and $x_b(v_3, v_3) = \omega$. Therefore, $\mathcal{F}$ contains the iteration

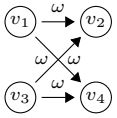
$$x_b^\sharp = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \omega & \omega & 0 \\ 0 & 0 & \omega & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



Intuitively, b allows only one unit of flow from v_2 to v_3, but arbitrarily much flow can remain both in v_2 and v_3. If this is iterated then the sum of those single units of flows from v_2 to v_3 grows, which is represented by a new capacity ω on this edge in the capacity constraint $b^\sharp$.

Finally, $\mathcal{F}$ contains the product

$$x_a x_b^\sharp x_a = \begin{pmatrix} 0 & \omega & 0 & \omega \\ 0 & 0 & 0 & 0 \\ 0 & \omega & 0 & \omega \\ 0 & 0 & 0 & 0 \end{pmatrix}$$



The top right coordinate in $x_a x_b^\sharp x_a$, which corresponds to the edge (v_1, v_4) from the source to the target, is ω. This suggests that an arbitrary amount of flow can be transported from $v_s = v_1$ to $v_4 = v_t$. Such an element is called an *unboundedness witness*.

▶ **Definition 10** (Unboundedness witness). *An unboundedness witness is an element of $x \in \mathcal{F}$ such that*

$$x(v_s, v_t) = \omega \ .$$

The existence of such elements in $\mathcal{F}$ is a sufficient and necessary condition for the existence of sequential flows carrying an arbitrary large amount of flow from the source to the target.

▶ **Theorem 11** (characterization of the unboundedness case). *An instance of the Sequential flow problem is unbounded iff there exists an unboundedness witness in the corresponding flow semigroup.*

We will now present the structure of the proof this theorem.

To begin with, we observe that since our finite capacities have integer values the integral flow theorem [9] guarantees that every sequence of capacity constraints has a flow with integer coefficients and maximum value. This allows use to work with *token flows*, in which an explicit set of named tokens is fixed, and the trajectory of every token is precisely described.

▶ **Definition 12** (Token flows). *Fix a finite set $\mathcal{T}$ of tokens. A token flow of $\mathcal{T}$ of length k over a capacity word w is a mapping $d \in V^{\mathcal{T}\times(0\dots k)}$ which describes the position of every token at every date, and which satisfies capacity constraints. Formally, for every $i \in [1,k]$ and $v, v' \in V$*

$$|\{\tau \in \mathcal{T} \mid d(\tau, i-1) = v \wedge d(\tau, i) = v'\}| \leq a_i(v, v').$$

The outcome of a token flow is described by a *global flow*, which accounts for the number of tokens traveling along every pair of states.

▶ **Definition 13** (Global flows). *With every token flow d of length k is associated a global flow denoted $g(d) \in \mathbb{N}^{V\times V}$, which measures the number of tokens moving between every pair of vertices between the dates 0 and k, formally defined as:*

$$g(d)(v, v') = |\{\tau \in \mathcal{T} \mid d(\tau, 0) = v \wedge d(\tau, k) = v'\}|$$

for all $v, v' \in V$.

The proof of Theorem 11 is in two steps: Lemma 14 establishes that the condition is sufficient and Lemma 17 that is it is necessary.

▶ **Lemma 14** (sufficient condition). *If there exists an unboundedness witness, the answer to the* Sequential flow problem *is ω.*

The proof makes use of the notion of paths.

▶ **Definition 15** (Paths). *A path in a capacity word $w = a_1 \dots a_k \in A^*$ is a sequence of vertices $\pi : \{0, \dots, k\} \to V$ that may be followed by a token, i.e., such that $a_i(\pi(i-1), \pi(i)) \geq 1$ for all $i \in \{1, \dots, k\}$.*

Proof sketch of Lemma 14. The full proof is presented in Appendix A.3. The key idea is to establish that the following property of element x in the flow semigroup is invariant by product and iteration, and satisfied by the abstract capacities. The property is satisfied by $x \in \mathcal{F}$ if for every $N \geq 1$, there exists a token flow $d(x)$ over a capacity word w such that the corresponding global flow $g(d(x))$ satisfies

$$\forall v, v' \in V, x(v, v') = \omega \Rightarrow g(d(x))(v, v') \geq N \ ,$$

as well as

$$\forall v, v' \in V, x(v, v') = 1 \Rightarrow \text{ there is a path in } w \text{ from } v \text{ to } v'.$$

The proof is by induction on the flow semigroup. For abstract capacity constraints x_a, the capacity word is simply a and we have N tokens following each edge (v, v') with $x_a(v, v') = \omega$.

For a product $x \cdot y$ the token flow $d(x \cdot y)$ for N is obtained by considering two token flows $d(x)$ and $d(y)$ for $|V|^2 N$ tokens, renaming and deleting some of the tokens and concatenating the resulting sequential flows.

The last case is the iteration $f = e^\sharp$. We start by showing that we can decompose $e^\sharp$ as a product of e with so-called simple unstable idempotents, where all non-zero entries are self-loops, except for a single unstable pair. This simple structure is used to craft token flows carrying arbitrarily large amount of tokens along unstable pairs. ◀

For the other direction, we show that we can evaluate capacity words in the flow semigroup so that the flow between pairs of vertices which the resulting element of $\mathcal{F}$ maps to 0 or 1 are uniformly bounded.

▶ **Theorem 16.** *Let K be the largest finite constant appearing in a capacity constraint of A. For every token flow d, there exists an element $x \in \mathcal{F}$ such that, for every $v, v' \in V$,*

$$x(v,v') = 0 \implies g(d)(v,v') = 0 \qquad \text{and}$$
$$x(v,v') = 1 \implies g(d)(v,v') \leq K \cdot |V|^{536 \cdot |V|^{12}}.$$

This result relies on a subtle and careful analysis of the flow semigroup, performed in Section 4. The first step is a general result about finite semigroups, showing that every element of a semigroup $\mathcal{S}$ has a finite representation as a binary tree whose height is polynomial in $L(\mathcal{S})$, the regular $\mathcal{J}$-length of the semigroup (Theorem 26). The second step is specific to the flow semigroup $\mathcal{F}$, and shows that for $\mathcal{F}$ this parameter is polynomial in $|V|$ (Theorem 33). We then extend these trees to incorporate the $\sharp$ operator, and prove that the resulting trees still have polynomial height. That leads to Theorem 16. By contraposition, a consequence of Theorem 16 is the following.

▶ **Lemma 17** (necessary condition)**.** *If the answer to the* Sequential flow problem *is ω then there is an unboundedness witness in $\mathcal{F}$.*

Proof. Since we have sequential flows of unbounded values between v_s and v_t, there exists a capacity word w with a sequential flow of value greater than $K \cdot |V|^{536 \cdot |V|^{12}}$. By the integral flow theorem [9], we also have a token flow d over w such that $g(d)(v_s, v_t) > K \cdot |V|^{536 \cdot |V|^{12}}$. We apply Theorem 16 to d. By case inspection, the only possible value of $x(v,v')$ is ω, thus x is an unboundedness witness. ◀

In order to test unboundedness for the Sequential flow problem we can compute the entire flow semigroup $\mathcal{F}$, starting with the capacity abstractions $\{x_a \mid a \in A\}$ and closing it by product and $\sharp$, and then check if it contains an unboundedness witness. The correctness of this algorithm follows directly from the definition of $\mathcal{F}$, as well as Lemma 14 and Lemma 17. The resulting algorithm runs in exponential time and space, essentially bounded by the size of $\mathcal{F}$.

This can be improved to polynomial space: Instead of explicitly enumerating elements of the flow group we can enumerating so-called $\sharp$-expressions, which represent elements of $\mathcal{F}$.

▶ **Definition 18.** *A $\sharp$-expression of an element $x \in \mathcal{F}$ is a finite $\mathcal{F}$-labeled ordered tree such that the root node is labeled by x, and every node ν is of one of three possible types:*

- *either a leaf node labeled by an abstract capacities $x_a, a \in A$;*
- *it has a single-child labeled by e, in which case $e = e^2$ is idempotent and ν is labeled by $e^\sharp$,*
- *it has two children labeled by x_1 and x_2, in which case ν is labeled by $x_1 \cdot x_2$.*

The recursive nature of this definition dictates a recursive algorithm to determine if a given element $x \in \{0,1,\omega\}^{V \times V}$ has a $\sharp$-expression of at most a given height h. By convention, the height of a single leaf node is 0.

▶ **Lemma 19.** *Given capacities A, element $x \in \{0,1,\omega\}^{V \times V}$, and $h \geq 0$, Algorithm 1 returns* true *iff x has a $\sharp$-expression of height at most h.*

Notice that Algorithm 1 still runs in (deterministic) exponential time due to the enumerations in lines 5 and 8. However, it only requires space polynomial in $|V|$ and h due to the explicit bound on the recursion depth.

The central argument for showing that unboundedness can be tested in polynomial space is a polynomial bound on the number of nested applications of the $\sharp$ operator necessary to produce an unboundedness witness, provided by the following theorem.

Algorithm 1 Check if x has a $\sharp$-expression of height at most h.

```
1: function IsIn-Rec(x, h)
2:     if x ∈ {x_a | a ∈ A} then
3:         return true
4:     if h=0 then
5:         return false
6:     for every y, z ∈ {0,1,ω}^{V×V} with y · z = x do
7:         if IsIn-Rec(y, h − 1) and IsIn-Rec(z, h − 1) then
8:             return true
9:     for every e ∈ {0,1,ω}^{V×V} with e · e = e and e^♯ = x do
10:        if IsIn-Rec(e, h − 1) then
11:            return true
12:    return false
```

▶ **Theorem 20.** *Every element of the flow semigroup $\mathcal{F}$ is generated by a $\sharp$-expression of height at most $2n^4$.*

We make use of proof techniques used for designing polynomial-space algorithms in other contexts, in particular for checking limitedness of desert automata [17] and the value 1 problem of probabilistic automata [8, 7]. According to Theorem 20, every element of $\mathcal{F}$ has a $\sharp$-expression of height at most $2n^4$. We can therefore check in polynomial space if a given element x is in the flow semigroup $\mathcal{F}$ and whether there exist unboundedness witnesses.

▶ **Theorem 21** (Checking unboundedness)**.** *Checking whether the optimal sequential flow is unbounded is decidable in polynomial space.*

Proof. By Theorem 20, any positive instance admits an unboundedness witnessed x that has $\sharp$-expressions of height at most $h = 2n^4$. It therefore suffices to enumerate (in polynomial space, using Algorithm 1) all $\sharp$-expressions of such-bouned height and for each check if the represented element $x \in \mathcal{F}$ constitutes an unboundedness witness, i.e., that $x(v_s, v_t) = \omega$. ◀

It is natural to ask what maximal finite sequential flow value is, in case it exists. This can be performed in polynomial space thanks to Theorem 16.

▶ **Theorem 22.** *Given capacities A, the optimal sequential flow can be computed in polynomial space.*

Proof. By Theorem 21, we can check in polynomial space whether the optimal sequential flow is unbounded. Suppose now that it is bounded and let $n = |V|$.

We proceed by binary search (see Algorithm 2). According to Theorem 16 and the constant fixed in line 1, Algorithm 2 preserves an invariant: the finite optimal sequential flow belongs to the interval $[m, M-1]$. Since this interval is halved at every step of the loop, this loop terminates in at most $\left\lceil \log_2 \left(K \cdot n^{536 \cdot n^{12}} \right) \right\rceil$ steps.

The loop requires to test the existence of a sequential flow of value C in polynomial space. We make use of two notions. A C-configuration is a map $V \to \{1, \dots, C\}$ counting the number of tokens per vertex, whose sum of coordinates is C. For two C-configurations x_s, x_t, a sequential integral flow from x_s to x_t of length ℓ is a sequence $f_1 f_2 \dots f_\ell \in (0 \dots C)^{V \times V}$ satisfying conditions (2′) and (3′) and such that $\forall v \in V, x_s(v) = \mathrm{out}(f_1)(v)$ and $x_t(v) = \mathrm{in}(f_\ell)(v)$. If such a sequential integral flow exists, we write $x_s \to_\ell x_t$.

Since our finite capacities have integer values the integral flow theorem [9] guarantees that the existence of sequential flow of value C is equivalent to the existence of ℓ such that $x_s \to_\ell x_t$, where x_s has coordinate C on v_s and 0 elsewhere and x_t has coordinate C on v_t and 0 elsewhere.

Notice that $x_s \to_\ell x_t$ iff there exists a sequence $x_s \to_1 x_1 \to_1 \ldots \to_1 x_\ell \to_1 x_t$. Therefore, checking $x_s \to_\ell x_t$ amounts to a reachability question in the space of K-configurations, for the relation $\to_1$. There is no need to traverse twice the same C-configuration, hence the search can be limited to $\ell \in 1 \ldots C^n$. Checking whether $x \to_1 x'$ reduces to solving $|A|$ instances of Max flow problem, in polynomial time, details are given in the next subsection.

Thus, checking $x_s \to_\ell x_t$ with $\ell \in 1 \ldots C^n$ can be performed non-deterministically in PSPACE, by storing triplets of C-configurations and the length ℓ.

A deterministic PSPACE algorithm is given by Savitch's theorem [19]: it consists in implementing a dichotomic search for the path, which requires only polynomial space in $n = |V|$, $|A|$ and $\log_2(C) \leq \log_2\left(K \cdot n^{536 \cdot n^{12}}\right)$. ◀

Algorithm 2 Computing the optimal sequential flow

$M \leftarrow K \cdot |V|^{536 \cdot |V|^{12}} + 1$
$m \leftarrow 0$
repeat
  $C \leftarrow \lceil (m + M)/2 \rceil$
  if there is a sequential flow f such that $|f| = C$ **then**
    $m \leftarrow C$
  else
    $M \leftarrow C$
until $M \leq m + 1$
return m

4 Diving into the flow semigroup

This section is dedicated to the proof of two bounds which are crucial to show that the Sequential flow problem can be solved in PSPACE. The first one is Theorem 20, which gives a polynomial upper-bound on the maximal depth of a $\sharp$-expression generating elements of $\mathcal{F}$. The second one is Theorem 16, which establishes a polynomial upper bound on the optimal sequential flow, in case it is finite.

The central tool for these proofs are summaries and $\sharp$-summaries, which provide a bounded-size representation of words of arbitrary length.

4.1 A general factorization theorem for finite semigroups

We rely on a form of factorization of words in a finite semigroups, which we call *summaries*. Let $(\mathcal{S}, \cdot)$ be a finite semigroup[3]. An element $e \in \mathcal{S}$ is called *idempotent* if $e \cdot e = e$. The set of idempotent elements of $\mathcal{S}$ is denoted $E(\mathcal{S})$.

[3] We choose to work with semigroups in this paper since they are slightly more general than monoids. Note that [13] formulates everything for finite monoids, but the existence of a neutral element is never used in the paper. Every statement from that paper holds for finite semigroups as well.

We define a morphism $\varphi_{\mathcal{S}} : \mathcal{S}^* \to \mathcal{S}$ that evaluates sequences of elements of $\mathcal{S}$ by applying the semigroup (product) operation: $\varphi_{\mathcal{S}}(x_0 \dots x_k) = x_0 \cdot x_1 \cdots x_k$.

We recall the Green relations, introduced in [12], $\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{H}$ on $\mathcal{S}$, starting with the following partial orders:

- $x \leq_{\mathcal{J}} y$ if there exist $a, b \in \mathcal{S}$ such that $x = ayb$
- $x \leq_{\mathcal{L}} y$ if there exist $a \in \mathcal{S}$ such that $x = ay$
- $x \leq_{\mathcal{R}} y$ if there exist $b \in \mathcal{S}$ such that $x = yb$
- $x \leq_{\mathcal{H}} y$ if $x \leq_{\mathcal{L}} y$ and $x \leq_{\mathcal{R}} y$

These partial orders can be thought of as reachability relations on $\mathcal{S}$, and the corresponding equivalence classes as strongly connected components.

We use well-established notations for the relations $\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{H}$, the equivalence relations induced by those partial orders. Formally, for each $\mathcal{X} \in \{\mathcal{J}, \mathcal{L}, \mathcal{R}, \mathcal{H}\}$ we define the relations $\mathcal{X} = \leq_{\mathcal{X}} \cap \geq_{\mathcal{X}}$ and $<_{\mathcal{X}} = \leq_{\mathcal{X}} \setminus \geq_{\mathcal{X}}$. Note that $\mathcal{J}$ is coarser than $\mathcal{L}$ and $\mathcal{R}$, themselves coarser than $\mathcal{H}$.

The *regular $\mathcal{J}$-length* [4] of a semigroup $\mathcal{S}$, denoted $L(\mathcal{S})$, is defined as $\sup\{k \in \mathbb{N} \mid \exists e_1 <_{\mathcal{J}} \cdots <_{\mathcal{J}} e_k \in E(\mathcal{S})\}$.

The *Ramsey function* of $\mathcal{S}$ is the function $R_{\mathcal{S}} : \mathbb{N} \to \mathbb{N}$ where $R_{\mathcal{S}}(k)$ is the minimal number n such that every word $w \in \mathcal{S}^*$ of length n contains an infix of the form $u_1 \cdots u_k$ with $\varphi(u_1) = \cdots = \varphi(u_k) = e$ for some $e \in E(\mathcal{S})$. The existence of such a number n for all k can be inferred from Ramsey's theorem, but the following theorem gives much more precise bounds.

A core element of the construction is the following result, which guarantees the existence of consecutive idempotent factors in sufficiently long words over $\mathcal{S}$. Note that the bound is only exponential in the regular $\mathcal{J}$-length of the semigroup, not its size.

▶ **Theorem 23** ([13, Theorem 1]). *For all $k \in \mathbb{N}$,*

$$k^{L(\mathcal{S})} \leq R_{\mathcal{S}}(k) \leq (k|\mathcal{S}|^4)^{L(\mathcal{S})}.$$

We can now define the central object of our proofs. The theorem gives a way to summarize a word with respect to a finite semigroup. A summary abstract sequences of idempotent infixes by only keeping the first and last ones. We do so in a way that ensures that the remaining idempotent factors are "short", so that the number of letters we keep from the initial word is polynomial in the size of the semigroup and $R_{\mathcal{S}}(3)$. The theorem gives a way to summarize a word with respect to a finite

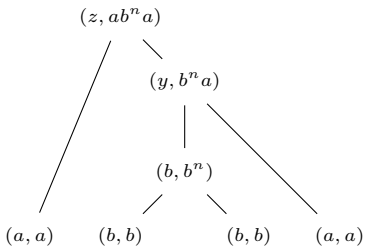
▶ **Definition 24.** *A summary of a word $w \in \mathcal{S}^*$ is a $\mathcal{S} \times \mathcal{S}^*$-labeled ordered binary tree with three types of nodes:*

- *a leaf has no children and a label (x, x) for some $x \in \mathcal{S}$*
- *a product node labeled (x, w) has two children labeled (y_1, u_1) and (y_2, u_2) such that $y_1 \cdot y_2 = x$ and $u_1 u_2 = w$.*
- *an idempotent node labeled (e, w) has two children labeled (e, u_1) and (e, u_2) such that we have $w = u_1 w' u_2$, $e \in E(\mathcal{S})$ and $\varphi_{\mathcal{S}}(u_1) = \varphi_{\mathcal{S}}(u_2) = \varphi_{\mathcal{S}}(w') = e$.*

Moreover, the root is labeled by (x, w), for some $x \in F$, called the result of the summary.

[4] Called regular $\mathcal{D}$-length in [13]. For finite semigroups, $\mathcal{D} = \mathcal{J}$, and we use $\mathcal{J}$ here since it is more common. Note that the paper introduces it with a different definition, but the two are proven equivalent in the extended version [14, Appendix B].

▶ **Example 25.** Consider the flow semigroup $\mathcal{F}$ for Example 1. Remember that $a = x_a$ and $b = x_b$: capacities match their abstractions because finite constants are 0 or 1. Let $z, y \in \mathcal{F}$ the products $y = b \cdot a$ and $z = a \cdot b \cdot a$. Since $b = b^2$, the following tree is a summary of $ab^n a$, for any $n > 2$.



Independently of their length, all words $(ab^n a)_{n\geq 2}$ have this summary of height 4 and size 7. The existence of summaries of constant depth is true in general, as shown in Theorem 26.

This definition resembles the one of Simon's factorization forests [20]. However, an important difference is that Simon's trees are meant to factorize the entire word, while ours omit a lot of information by skipping intermediate idempotents. This lets us obtain better bounds on the height of the tree, since Simon's trees have linear height in the size of the semigroup, not just its regular $\mathcal{J}$-length. Those bounds are essential to obtain singly-exponential bounds, and then a polynomial-space algorithm, in the quantitative setting.

▶ **Theorem 26.** *For all $w \in \mathcal{S}^*$ there exists a summary whose result is $\varphi_{\mathcal{S}}(w)$ and of height at most $L(\mathcal{S})(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4)$.*

Proof sketch. The full proof is presented in Appendix B.1. We first define the regular $\mathcal{J}$-length of an element of the semigroup, as the one of the subsemigroup of elements $\leq_{\mathcal{J}}$-below it. The proof goes through an induction on the regular $\mathcal{J}$-length of elements of the semigroup. We start by cutting the word in minimal blocks of maximal regular $\mathcal{J}$-length. We factorize these blocks as a single letter and a block of smaller regular $\mathcal{J}$-length, for which we get a factorization by induction. Then we consider the word obtained by replacing each block with its value in the semigroup.

We cut this word into infixes, each long enough to guarantee that it contains idempotents. We describe an operation that lets us merge some blocks where the same idempotent appears. We then use properties of the Green relations to bound the number of blocks obtained this way. ◀

4.2 Application to the flow semigroup and iterations

This section is dedicated to the proof of two bounds which are crucial to show that the Sequential flow problem can be solved in polynomial space. The first one is Theorem 20, which gives a polynomial upper-bound on the maximal depth of a $\sharp$-expression generating elements of $\mathcal{F}$. The second one is Theorem 16, which establishes a polynomial upper bound on the optimal sequential flow, in case it is finite.

Let $n = |V|$, and let $\mathcal{B} = \mathbb{B}^{n\times n}$ be the finite semigroup of Boolean matrices of dimension n, equipped with the $(\vee, \wedge)$ matrix product. The following result bounds its regular $\mathcal{J}$-length by a polynomial in its dimension.

▶ **Theorem 27** ([13, Theorem 2])**.** *The regular $\mathcal{J}$-length of $\mathcal{B}$ is at most $(n^2 + n + 2)/2$.*

Let us now take a look at the flow semigroup, $\mathcal{F} \subseteq \mathbb{M}^{n\times n}$ with $\mathbb{M} = (\{0,1,\omega\}, \max, \min)$.

▶ **Lemma 28.** *$\mathcal{F}$ is isomorphic to a sub-semigroup of $\mathcal{B}^2$.*

Proof. Define $\psi : \mathcal{F} \to \mathcal{B}^2$ such that for all matrix $x \in \mathcal{F}$, $\psi(x) = (\mu_1, \mu_\omega)$ with, for all $i, j \in [1, n]$,

$$\mu_1(i,j) = \begin{cases} \top \text{ if } x(i,j) \geq 1 \\ \bot \text{ otherwise} \end{cases} \qquad \text{and} \qquad \mu_\omega(i,j) = \begin{cases} \top \text{ if } x(i,j) = \omega \\ \bot \text{ otherwise.} \end{cases}$$

This is an injective function, and it is easily verified that this is a morphism. ◀

▶ **Theorem 29.** *Every word $w \in \mathcal{F}^*$ admits a summary of height at most $536 \cdot n^{10}$.*

Proof. As observed above, $\mathcal{F}$ is isomorphic to a subsemigroup of $\mathcal{B}^2$, which has regular $\mathcal{J}$-length bounded by $(n^2+n+2)/2$. The regular $\mathcal{J}$-length of $\mathcal{F}$ is then at most $(n^2+n+2)^2/4$, while its size is 3^{n^2}. By Theorem 23, we obtain $R_{\mathcal{F}}(3) \leq 3^{(n^2+1)(n^2+n+2)^2} \leq 3^{32n^6}$ for $n \geq 1$. Then, by Theorem 26 we have that every word over $\mathcal{F}$ has a summary of height at most

$$\begin{aligned} & L(\mathcal{F})(\log_2(|\mathcal{F}|) + 2\log_2(R_{\mathcal{F}}(3)) + 4) \\ \leq\ & \frac{(n^2+n+2)^2}{4}(n^2\log_2(3) + 2(32n^6\log_2(3)) + 4) \\ \leq\ & 4n^4(2n^2 + 128n^6 + 4) \\ \leq\ & 536n^{10} \end{aligned}$$ ◀

We now need to integrate the $\sharp$ operator in this construction. We do this by following a proof of Simon [21, Theorem 9] on a different semigroup.

▶ **Lemma 30.** *Let $m \geq n^2$ and $e_1, \ldots, e_m \in E(\mathcal{F})$ idempotents of $\mathcal{F}$ such that $e_{i+1} \leq_{\mathcal{J}} e_i^\sharp$ for all i. Then there exists i such that $e_i^\sharp = e_i$.*

The proof of this lemma is presented in Appendix B.2. Observe that Example 3 can be generalized to obtain instances where we need a linear number of nested $\sharp$ in order to obtain some elements of $\mathcal{F}$.

That result has two interesting consequences, which are the keys to obtain PSPACE algorithms for the Sequential flow problem. The first consequence is that small "$\sharp$-expressions" (Definition 18) are enough to generate all elements in the flow semigroup, as stated in Theorem 20.

Proof of Theorem 20. Clearly by definition of $\mathcal{F}$ every element is generated by a $\sharp$-expression. Define the $\sharp$-*height* of a $\sharp$-expression as the maximal number of idempotent nodes along a branch. To begin with, observe that every element of $\mathcal{F}$ is generated by a $\sharp$-expression of $\sharp$-height at most $n^2 - 1$: Take a $\sharp$-expression E of $\sharp$-height n^2 or more. There are idempotent nodes $\nu_1, \ldots, \nu_{n^2}$ along a branch of E (from leaf to root), and idempotents $e_1, \ldots, e_{n^2}$ such that ν_i is labeled $e_i^\sharp$ and has a single child labeled e_i.

Notice that in E, if a node labeled y is the child of a node labeled z, then $z \leq_{\mathcal{J}} y$. It is trivial for a product node. For an idempotent node, it follows from the fact that $e^\sharp \cdot e = e^\sharp$, which can be checked by a simple computation. It follows by transitivity of $\leq_{\mathcal{J}}$ that this still holds when y labels a descendant of z, and thus that $e_{i+1} \leq_{\mathcal{J}} e_i^\sharp$ for all i.

Hence there is a ν_i such that $e_i = e_i^\sharp$ by Lemma 30. As a result, we can reduce E by replacing the subtree rooted in ν_i by the one rooted in its child.

We have shown that all elements of $\mathcal{F}$ are generated by a $\sharp$-expression of $\sharp$-height at most n^2-1. We now prove that for all $h>0$, if an element has a $\sharp$-expression of $\sharp$-height at most h then it has one of height at most $(2n^2+1)h$. Take x generated by E of $\sharp$-height h, assume we have shown the property for all lower heights. Then $x = x_1 \dots x_m$ with for all i either $x_i = x_a$ for some $a \in A$ or $x_i = y_i^\sharp$ with y_i generated by a $\sharp$-expression of $\sharp$-height $< h$. By induction hypothesis, every x_i in the second case can be generated by a $\sharp$-expression E_i of height $\leq (2n^2+1)(h-1)$.

We can assume that $m \leq |\mathcal{F}| \leq 3^{n^2}$. Otherwise, there must exist $i<j$ with $x_1 \cdots x_i = x_1 \cdots x_j$ and we can shorten x as $x = x_1 \cdots x_i x_{j+1} \cdots x_m$. Thus, we can obtain x from $x_1, \dots, x_m$ with a tree of product nodes of height at most $\log_2(|\mathcal{F}|) \leq 2n^2$. The leaves of this tree are labeled by $x_1, \dots, x_m$. For all x_i that is not in $\{x_a \mid a \in A\}$, there exist y_i generated by E_i such that $x_i = y_i^\sharp$. We append E_i below the corresponding leaf labeled x_i. We obtain a $\sharp$-expression of height at most $(2n^2+1)(h-1) + 2n^2 + 1 = (2n^2+1)h$.

This concludes our induction. To obtain the corollary, it suffices to apply it with $h = n^2-1$: every element has a $\sharp$-expression of height at most $(2n^2+1)(n^2-1) \leq 2n^4$. ◀

The second consequence of Lemma 30 is that every capacity word can be represented as a small $\sharp$-summary.

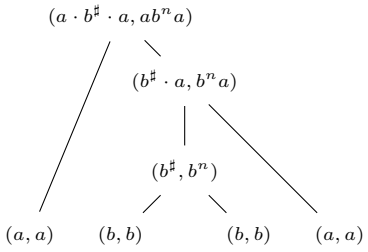
We define $\sharp$-*summaries*, where idempotent nodes for $e \in E(\mathcal{S})$ are labeled with $e^\sharp$ instead of e.

▶ **Definition 31.** *A $\sharp$-summary of a word $w \in \mathcal{F}^*$ is a $\mathcal{F} \times \mathcal{F}^*$-labeled ordered binary tree, with three types of nodes:*

- *A leaf is labeled by (x,x) for some $x \in \mathcal{F}$,*
- *A product node has two children. If their labels are (x_1,u_1) and (x_2,u_2) then its label is $(x_1 \cdot x_2, u_1u_2)$*
- *An idempotent node has two children. If their labels are (x_1,u_1) and (x_2,u_2) then $x_1 = x_2 = e$ is an idempotent of $\mathcal{F}$ and the label of the node is $(e^\sharp, u_1w_1 \dots w_mu_2)$ for some $w_1, \dots, w_m \in \mathcal{F}^*$ such that all w_i have a $\sharp$-summary whose root is labeled (e, w_i). In the case that $e = e^\sharp$ we say that the node is a stable idempotent node, and an unstable idempotent node otherwise.*

Moreover, the root is labeled by (x,w) for some $x \in \mathcal{F}$, which is called the result of the $\sharp$-summary.

▶ **Example 32.** Consider Example 1, again. Since $b = b^2$, the following tree is a $\sharp$-summary of ab^na.



This $\sharp$-summary bears some similarity with the summary of Example 25. However, there is a crucial difference: the root of the $\sharp$-summary is labelled by $a \cdot b^\sharp \cdot a$, which is an unboundedness witness (see details before Definition 10). This is not the case in the summary of Example 25: the root is labelled by $z = aba$ which is not an unboundedness witness since $z(v_1, v_4) = 1$.

Independently of their length, all words $(ab^n a)_{n\geq 2}$ have this $\sharp$-summary of height 4 and size 7. The existence of $\sharp$-summaries of constant height (and size), whatever the size of the word, is true in general, as shown in Theorem 33.

Our next step is to show that all words have a $\sharp$-summary of polynomial height in the number of vertices $n = |V|$. We take inspiration from two existing proofs: First, one by Kirsten [17] to show that the number of unstable nodes along a branch of a $\sharp$-summary (or a $\sharp$-expression) is bounded by a polynomial in n, the number of vertices[5]. Second, one by Simon [20] to show that every word has a $\sharp$-summary where the distance between consecutive unstable nodes along a branch is bounded by another polynomial in n. Combining those two arguments yields the result.

▶ **Theorem 33.** *For all $w \in \mathcal{F}^*$ there is a $\sharp$-summary of height at most $536 \cdot n^{12}$.*

Proof of Theorem 33. This proof is inspired by [21, Lemma 10]. We start by observing that a branch of a $\sharp$-summary τ can have at most $n^2 - 1$ unstable idempotent nodes. Suppose the contrary, let $\nu_1, \dots, \nu_{n^2}$ be unstable idempotent nodes along a branch, from leaf to root, with each ν_i labeled by $e_i^\sharp$ and with a single child labeled by e_i.

Analogously to the proof of Theorem 20, if a node labeled y is the descendant of a node labeled z, then $z \leq_{\mathcal{J}} y$. Hence $e_{i+1} \leq_{\mathcal{J}} e_i^\sharp$ for all i. By Lemma 30, one of those nodes must be stable, a contradiction.

We now prove that every word w admits a $\sharp$-summary where along every branch, sequences of consecutive nodes without any leaf or unstable idempotent node have length at most $536 \cdot n^{10} - 1$.

By induction on $|w|$. Let w be a word and τ a summary of w of height at most $536 \cdot n^{10}$ (which exists by Theorem 26). We distinguish two cases:
Case 1: all idempotent nodes in τ are stable. Then τ is a $\sharp$-summary and the property holds trivially since τ has height at most $536 \cdot n^{10}$.
Case 2: τ has an idempotent node ν labeled (v, e) with $e^\sharp \neq e$. If τ has several such nodes, we pick one of maximal depth. Consider the subtree τ_ν rooted at ν. Let τ'_v be the tree obtained from τ_ν by replacing the label of ν with $(v, e^\sharp)$. It is a $\sharp$-summary as we took ν of maximal depth and thus every idempotent node below it is stable. Further, it has height at most $536 \cdot n^{10}$, since τ_ν is a subtree of τ. Now consider the word w' where the factor v has been replaced with a single letter $e^\sharp$. It is shorter than w, thus it admits a $\sharp$-summary τ' satisfying the property. To obtain the desired $\sharp$-summary for w', we start from τ' and replace the $e^\sharp$ leaf with τ_v. The result is a $\sharp$-summary, which satisfies the property since the tree below ν has height at most $536 \cdot n^{10}$, ν is an unstable idempotent node and the rest of the tree satisfies that property.

We can now bring in the bound on the number of unstable idempotent nodes along a branch: since every branch has at most $n^2 - 1$ unstable idempotent nodes, and the length of sequences of nodes without those is bounded by $536 \cdot n^{10} - 1$, the height is at most $536 \cdot n^{12}$. ◀

We make use of Theorem 33 to prove Theorem 16.

Proof of Theorem 16. The proof makes use of the usual notion of *cuts*.

[5] A polynomial bound in $\mathcal{O}(n^4)$ can be obtained by proving that $e^\sharp <_{\mathcal{J}} e$ for all $e \in \mathcal{F}$, and using the bound on the regular $\mathcal{J}$-length. We prefer to use Lemma 30, which gives $\mathcal{O}(n^2)$.

▶ **Definition 34.** *Given two sets of vertices V_s and V_t, a cut between V_s and V_t in a capacity word $w = a_1 \dots a_k$ of length k is a sequence $\mathbf{c} = c_1 \dots c_k$ of sets of edges such that for every path $v_0 \dots v_k$ in w with $v_0 \in V_s$ and $v_k \in V_t$ there exists i such that $(v_{i-1}, v_i) \in c_i$.*

The cost of this cut is the sum of the capacities of its edges:

$$cost(\mathbf{c}) = \sum_{i=1}^{k} \sum_{(v,v' \in c_i)} a_i(v, v').$$

By the max flow-min cut theorem [10], we know that in a capacity word w the minimal cost of a cut between two vertices is exactly the maximal value of a flow between those vertices.

Set $n = |V|$. We prove the following statement by induction on h the height of a $\sharp$-summary τ of $x_{a_1} \dots x_{a_m}$: there is a cut between s and t in $w = a_1 \dots a_m$ of cost at most Kn^h.

1. if $x(s,t) = 0$ then there is no path in w from s to t, and
2. if $x(s,t) \leq 1$ then there is a cut between s and t in w of cost at most Kn^h.

If $h = 0$ then τ is just a leaf and therefore w is a single letter a. Since $x(s,t) \leq 1$, we have $a(s,t) < \omega$ and thus we have a cut of value $\leq K$ between s and t.

If the root of τ is a product node, let $(y_1, u_1), (y_2, u_2) \in \mathcal{F} \times \mathcal{F}^*$ be the labels of its two children. Then $x = y_1 \cdot y_2$ and $u_1 = x_{a_1} \dots x_{a_\ell}$ and $u_2 = x_{a_{\ell+1}} \dots x_{a_k}$. Since $x(s,t) \leq 1$, for all $v \in V$, we have $y_1(s,v) \leq 1$ or $y_2(v,t) \leq 1$. By induction hypothesis, this means that for all $v \in V$, we have a cut of value at most n^{h-1} between s and v in $a_1 \dots a_\ell$ or between v and t in $a_{\ell+1} \dots a_k$. We take the union of all these cuts. This forms a cut between s and t in w, of value at most $nKn^{h-1} \leq Kn^h$.

If the root of τ is an idempotent node, it has two children labeled $(e, u_1), (e, u_k)$ with $u_1, u_k \in \mathcal{F}^*$ and $e \in E(\mathcal{F})$. Then $x = e^\sharp$ and $u_1 = x_{a_1} \dots x_{a_\ell}$ and $u_2 = x_{a_{\ell'}} \dots x_{a_k}$ for some $1 \leq \ell < \ell' \leq k$. By induction hypothesis, for all $v \in V$ such that $e(s,v) \leq 1$ we have a cut in $a_1 \dots a_\ell$ between s and v of value $\leq Kn^{h-1}$. Similarly, for all $v' \in V$ such that $e(v',t) \leq 1$ then we have a cut in $a_{\ell'} \dots a_k$ between v' and t of value $\leq Kn^{h-1}$.

Consider the union of those cuts: its value is at most $2Kn^{h-1} \leq Kn^h$ (we set aside the very specific and trivial case $n = 1$). We only have to prove that it is a cut between s and t in w. Since x is the root of τ then $x = e^\sharp$ for some idempotent, hence $x^\sharp = x$ (cf. Lemma 7). In particular, (s,t) is a stable node in x. According to Lemma 5, every path between s and t must either be in a vertex v with $e(s,v) \leq 1$ after going through u_1, or be in a vertex v' with $e(v',t) \leq 1$ before going through u_k. As a consequence, it must go through the constructed cut.

According to Theorem 33, the height can be bounded by $536 \cdot n^{12}$, hence the result. This ends our proof. ◀

5 Extensions

Our approach to solve the Sequential flow problemcan be extended to further generalizations that consider sequential flows between sets of source and target vertices and under regular constraints on the witnessing capacity words.

5.1 Fair flows along multiple edges

The previous results and algorithms can be adapted to solve a more general problem: the *Fair sequential flow problem* . In that setting, instead of a single source-target pair

(v_s, v_t), the problem comes with a collection $E \subseteq V \times V$ of edges, and one wishes to carry as much flow as possible along those edges.

There is a fairness constraint: the objective is *not* to maximize the total amount of flow through the edges, which would easily be reformulated as an instance of the Sequential flow problem. Such a reformulation could lead to one of the edges being unused, in the sense where it would carry no flow at all.

Instead, there is a fairness constraint, which is better expressed using a token flow d and its associated global flow $g(d)$) (cf. Definition 12). We want the flow d to carry simultaneously many tokens through every edge in E, by maximizing the value

$$|d| = \min\{g(d)(v_s, v_t) \mid (v_s, v_t) \in E\} \ ,$$

and the Fair sequential flow problem requires to compute the value

$$\sup_{\text{token flow } d} |d| \ .$$

This problem can be solved similarly to the Sequential flow problem, except one looks for a different kind of witnesses in the flow semigroup $\mathcal{F}$.

▶ **Definition 35.** *A fair unboundedness witness is an element of $x \in \mathcal{F}$ such that*

$$\forall (v_s, v_t) \in E, x(v_s, v_t) = \omega \ .$$

▶ **Theorem 36.** *The Fair sequential flow problem can be solved in polynomial space.*

Proof. The unboundedness of the Fair sequential flow problem is equivalent to the existence of a fair unboundedness witness in the flow semigroup $\mathcal{F}$, the proof is a straightforward adaptation of the proofs of Lemma 14 and 17. Such a witness can be looked for in polynomial space for using Algorithm 1. If no such witness exists then the value $\sup_d |d|$ is finite and even bounded by $B = K \cdot |V|^{536 \cdot |V|^{12}}$ according to Theorem 16. Then a variant of Algorithm 2 can be used in order to optimize $|d|$, by looking for a token flow d moving exactly k tokens along every edge in E, where k is optimized by dichotomic search in the interval $0 \dots B$. ◀

5.2 Regular constraints

The *Fair sequential flow problem with regular constraints* generalizes the Fair sequential flow problem by requiring that we only consider capacity words within a given regular language. Formally, consider a finite set $A \subseteq (\mathbb{N} \cup \{\omega\})^{V \times V}$ of capacities, a set $E \subseteq V \times V$ and a regular language $L \subseteq A^*$ recognized by a non-deterministic automaton with m states. Let $n = |V|$ and K be the largest finite capacity in any element of A. The problem is to compute the optimal fair token flow

$$\sup_{\text{token flow } d} |d| \ .$$

over capacity words in L.

▶ **Theorem 37.** *The Fair sequential flow problem with regular constraints can be solved in polynomial space. Furthermore if the answer is bounded then it is at most $K(2|V|)^{(170 \log_2(m) + 835)|V|^{12}}$.*

This can be shown with the technique discussed in Sections 3 and 5.1 with a slightly extended definition of the flow semigroup. We describe the necessary adjustments below; more detailed proofs are presented in Appendix C.

Fix a non-deterministic finite automaton $\mathcal{A}$ with Q the set of m control states, $I, F \subseteq Q$ sets of initial and final states, and $\Delta \subseteq Q \times A \times Q$ the transitions over the alphabet A. We define the semigroup $\mathcal{B}_\mathcal{A}$ made of the set of triples in $Q \times \{0,1,\omega\}^{V\times V} \times Q$, plus an element $\bot$, and where the product $\star$ is defined as follows:

$$(q_1, x, q_1') \star (q_2, y, q_2') = \begin{cases} (q_1, x \cdot y, q_2') \text{ if } q_1' = q_2 \\ \bot \text{ otherwise.} \end{cases}$$

We set $\bot \star \bot = \bot$ and $(q,x,q') \star \bot = \bot \star (q,x,q') = \bot$ for all $(q,x,q') \in Q \times \{0,1,\omega\}^{V\times V} \times Q$.

To lift the iteration operation, notice that, except for $\bot$, an idempotent of $\mathcal{B}_\mathcal{A}$ is of the form (q,e,q) with an idempotent matrix e. Define $(q,e,q)^\sharp = (q, e^\sharp, q)$ for all $q \in Q$ and idempotent e, and let $\bot^\sharp = \bot$.

The *labeled flow semigroup* $\mathcal{F}_\mathcal{A}$ is defined as the smallest sub-semigroup of $\mathcal{B}_\mathcal{A}$ which contains $\{(q, x_a, q') \mid (q,a,q') \in \Delta\}$ and is stable by $\star$ and $\sharp$.

It then suffices to go through the same steps as in Sections 3 and 4, with some minor changes to accommodate the state constraints. We explain the necessary changes in Appendix C.

Conclusion

We provide a new algebraic technique to solve the Sequential flow problem in PSPACE. We mention two promising directions to utilize the results shown here. First, we aim to adapt our techniques to graphs generated by graph grammars. Second, we plan to extend our framework to settings with asynchronous flows, with applications to asynchronous distributed computing.

References

1 Eleni C. Akrida, Jurek Czyzowicz, Leszek Gąsieniec, Łukasz Kuszner, and Paul G. Spirakis. Temporal flows in temporal networks. *Journal of Computer and System Sciences*, 103:46–60, 2019. doi:10.1016/j.jcss.2019.02.003.

2 Jay E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20(1):1–66, December 1989. doi:10.1007/BF02216922.

3 Achim Blumensath, Thomas Colcombet, and Pawel Parys. On a fragment of AMSO and tiling systems. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 47 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.STACS.2016.19.

4 Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Log. Methods Comput. Sci.*, 9(3), 2013. doi:10.2168/LMCS-9(3:3)2013.

5 Thomas Colcombet, Nathanaël Fijalkow, and Pierre Ohlmann. Controlling a random population. In *International Conference on Foundations of Software Science and Computational Structures (FoSSaCS)*, volume 12077 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2020. doi:10.1007/978-3-030-45231-5_7.

6 Thomas Colcombet, Nathanaël Fijalkow, and Pierre Ohlmann. Controlling a random population. *Logical Methods in Computer Science*, 17(4), 2021. doi:10.46298/LMCS-17(4:12)2021.

7 Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Denis Kuperberg. Stamina: Stabilisation monoids in automata theory. In *International Conference on Implementation and*

Application of Automata, volume 10329 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 2017. doi:10.1007/978-3-319-60134-2_9.

8 Nathanaël Fijalkow, Hugo Gimbert, Edon Kelmendi, and Youssouf Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. *Log. Methods Comput. Sci.*, 11(2), 2015. doi:10.2168/LMCS-11(2:12)2015.

9 L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958. doi:10.1287/opre.6.3.419.

10 Lester Randolph Ford and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

11 Hugo Gimbert, Corto Mascle, and Patrick Totzke. Optimally controlling a random population, 2025. arXiv:2411.15181.

12 James Alexander Green. On the structure of semigroups. *Annals of Mathematics*, 54(1):163–172, 1951. doi:10.2307/1969317.

13 Ismaël Jecker. A Ramsey theorem for finite monoids. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 187 of *LIPIcs*, pages 44:1–44:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.STACS.2021.44.

14 Ismaël Jecker. A ramsey theorem for finite monoids. *CoRR*, abs/2101.05895, 2021. arXiv:2101.05895.

15 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *Symposium on Theory of Computing (STOC)*, STOC '00, page 504–513. Association for Computing Machinery, 2000. doi:10.1145/335305.335364.

16 David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002. doi:10.1006/JCSS.2002.1829.

17 Daniel Kirsten. Distance desert automata and the star height problem. *RAIRO Theor. Informatics Appl.*, 39(3):455–509, 2005. doi:10.1051/ITA:2005027.

18 Jean-Eric Pin. *Varieties of formal languages*. North Oxford, London et Plenum, New-York, 1986. doi:10.5555/576708.

19 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

20 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990. doi:10.1016/0304-3975(90)90047-L.

21 Imre Simon. On semigroups of matrices over the tropical semiring. *Informatique Théorique et Applications*, 28(3-4):277–294, 1994.

Appendix

A Proofs of Section 3

A.1 Proof of Lemma 5

Proof of Lemma 5. The three cases are clearly distinct: in case i) K_n is ω, in case it is finite but unbounded, in case iii) it is bounded.

Case i). Assume that $e(v, v') = \omega$ and let $n \geq 1$. Since $e^n = e$, then $e^n(v, v') = \omega$. By definition of the product, there exists a sequence $v = v_0, v_1, \ldots, v_n = v'$ such that $e(v_{i-1}, v_i) = \omega, i \in 1 \ldots n$. Those edges can carry an arbitrarily large amount of flow, hence the first statement of i) holds. Moreover, for $n = |V|$, there must exist $0 \leq i < j \leq n$ such that $v_i = v_j$. Set $v'' = v_i = v_j$. Then $e(v'', v'') = e^{j-i}(v'', v'') \geq \min(e(v_i, v_{i+1}), \ldots, e(v_{j-1}, v_j)) = \omega$. Conversely, if such a pair v', v'' exists then $e(v, v') = e^2(v, v') \geq \min(e(v, v''), e(v'', v'))) = \omega$.

Cases ii) and iii). Assume now that $e(v, v') = 1$ and let $n \geq 1$. Analogously to the previous case we must have $\forall n, K_n \geq 1$.

Introduce

$$P = \{(v_0, v'_0) \in V^2 \mid e(v, v_0) = \omega \wedge e(v_0, v'_0) = 1 \wedge e(v'_0, v') = \omega\}.$$

The alternative conditions for cases (ii) and (iii) simply check whether P is non-empty or empty, respectively. We start with a preliminary remark. Let d_n be a flow from v to v' in the pipeline e^n. Since $e(v, v') = 1$, every sequence $v = v_0, v_1, \ldots, v_n = v'$ such that all $d_n(v_i, v_{i+1}) > 0$ carry positive flow must traverse a 1-labelled edge $e(v_0, v'_0) = 1$ at some point.

This remark shows $K_n \leq n|V|$: the cut which removes all edges of capacity 1 disconnects v from v'. This has cost at most $n|V|$ thus according to the maxflow-mincut theorem, $K_n \leq n|V|$.

Assume first that P is empty. Consider the cut which removes from the pipeline all 1-labelled edge in the first and last copy of e. Since P is empty, according to the previous remark, the cut disconnects v from v'. There are at most $2|V|$ such edges thus according to the maxflow-mincut theorem, $K_n \leq 2|V|$.

Assume now that P is not empty. A reasoning similar to case i) shows that P contains a pair (v_0, v'_0) such that $e(v_0, v_0) = e(v'_0, v'_0) = \omega$. Consider the following sequential flow of value n in the pipeline ee^ne. First all the flow traverses the ω-edge (v, v_0). Then at every step $1 \leq \ell \leq n$, one unit of flow traverses the 1-edge (v_0, v'_0) while the rest of the flow stays on the ω-loops on v_0 and v'_0. Finally all the flow travereses the ω-edge (v'_0, v'). This shows that $K_{n+2} \geq n$. ◀

A.2 Proof of Lemma 7

Proof of Lemma 7. The iteration operation does not create new unstable edges. Let (v, v') such that $e^\sharp(v, v') = 1$. To prove stability, we take any $v_0, v'_0 \in V$ such that $\omega = e^\sharp(v, v_0) = e^\sharp(v'_0, v')$ and prove that $e^\sharp(v_0, v'_0) = 0$, which is enough according to iii) of Lemma 5. By definition of $e^\sharp$, it is also enough to prove $e(v_0, v'_0) = 0$.

The first case is $\omega = e(v, v_0) = e(v'_0, v')$. Since $e^\sharp(v, v') = 1$ then $e(v, v') = 1$ hence (v, v') is stable in e. According to iii) of Lemma 5, $e(v_0, v'_0) = 0$.

The second case is $1 = e(v, v_0)$. Then (v, v_0) is unstable in e thus there exists w_0, w'_0 such that $\omega = e(v, w_0) = e(w'_0, v_0)$ and $e(w_0, w'_0) = 1$. Since (v, v') is stable in e then according to iii) of Lemma 5, $e(w_0, v'_0) = 0$. Then, $0 = e(w_0, v'_0) = e^3(w_0, v'_0) \geq \min(e(w_0, w'_0), e(w'_0, v_0), e(v_0, v'_0)) = \min(1, e(v_0, v'_0))$. That shows $e(v_0, v'_0) = 0$.

The third case is $1 = e(v'_0, v')$. It is similar to the second case. ◀

A.3 Proof of Lemma 14

We prove the following property on elements of the flow semigroup $\mathcal{F}$:

($\diamondsuit$) For all $x \in \mathcal{F}$, for all $N \in \mathbb{N}$, there exist a capacity word w and a token flow d over w such that for all $v, v' \in V$, the following two conditions are satisfied:

1. $x(v, v') = \omega \Rightarrow g(d)(v, v') \geq N$
2. $x(v, v') \geq 1 \Rightarrow$ there is a path in w from v to v' (in the sense of Definition 15).

This clearly implies the lemma, by applying property ($\diamondsuit$) (specifically, the first item) to an unboundedness witness.

The proof follows the inductive definition of the flow semigroup $\mathcal{F}$: first prove property ($\diamondsuit$) for the abstracted capacity constraints $\{x_a \mid a \in A\}$, then show that property ($\diamondsuit$) is preserved by product and iteration.

Let x_a be the abstraction of a capacity constraint $a \in \mathbb{N}^{V \times V}$. Let $N \in \mathbb{N}$. We simply use the capacity word of length one $w = a$, with a token flow d over w transferring N tokens from v to v' for all $v, v' \in V$ such that $x_a(v, v') = \omega$. Both conditions 1) and 2) clearly hold by definition of the abstraction x_a, hence the property ($\diamondsuit$) holds.

We now show that property ($\diamondsuit$) is preserved by the product operation. Let $x, y \in \mathcal{F}$ which both satisfy property ($\diamondsuit$) , and consider their product $x \cdot y$ and some integer N. To obtain the property for N for $x \cdot y$, we apply it with $N' = |V|^2 N$ for x and y, and obtain two capacity words w_x, w_y and token flows d_x, d_y which satisfy conditions 1) and 2) for N'. We use the concatenation $w = w_x w_y$ as our capacity word. By definition of the product, for each v, v' such that $(x \cdot y)(v, v') \geq 1$, there exists $\overline{v} \in V$ such that $x(v, \overline{v}) \geq 1$ and $y(\overline{v}, v') \geq 1$. Since condition 2) holds for both w_x and w_y, there is a path from v to $\overline{v}$ in w_x, and from $\overline{v}$ to v' in w_y, and thus from v to v' in w. We no define the token flow for $x \cdot y$. Take an ω-edge of $(x \cdot y)$ i.e. a pair v, v' such that $(x \cdot y)(v, v') = \omega$. By definition of the product, there exists $\overline{v} \in V$ such that $x(v, \overline{v}) = y(\overline{v}, v') = \omega$. We select N tokens from the token flow d_x going from v to $\overline{v}$ and N tokens from d_y going from $\overline{v}$ to v'. Since at least $|V|^2 N$ tokens are moving along each ω-edge in d_x and d_y, the selection can be performed so that the set of tokens are pairwise-disjoint. The N tokens selected for $(\overline{v}, v')$ in d_y are renamed so that they are equal to the N tokens selected for $(v, \overline{v})$ in d_x. All other tokens are deleted from d_x and d_y. The resulting token flows are concatenated, to get one over w where N tokens are moving along every ω-edge of $x \cdot y$. This terminates the inductive step for the product.

Finally, we prove that the property ($\diamondsuit$) is preserved by iteration, i.e. application of the $\sharp$ operator to idempotent elements. To begin with, we show that it suffices to prove that the property ($\diamondsuit$) is preserved while applying $\sharp$ on simple unstable idempotents, defined below.

▶ **Definition 38.** *A simple unstable idempotent is one which only has self-loops (1 or ω), plus a single unstable pair. Formally, it is an idempotent element $e \in \mathcal{F}$ such that there exist $v \neq v' \in V$ with $e(v, v') = 1$ and for all $\overline{v} \neq \overline{v}' \in V$, $(v, v') \neq (\overline{v}, \overline{v}') \Rightarrow e(\overline{v}, \overline{v}') = 0$.*

Given two elements x, y of $\mathcal{F}$, we write $x \leq y$ if $x(v, v') \leq y(v, v')$ for all $v, v' \in V$. Clearly, if y satisfies the property ($\diamondsuit$) , then so does x (take the same pipelines and token flows).

▶ **Lemma 39.** *For every unstable idempotent $e \in \mathcal{F}$, there exist some simple unstable idempotents $e_1, \ldots, e_m \leq e \in \mathcal{F}$ such that $e^\sharp \leq e e_1^\sharp \cdots e_m^\sharp e$.*

Proof. For every unstable pair $(\overline{v}, \overline{v}')$ of e such that $e(\overline{v}, \overline{v}) = e(\overline{v}', \overline{v}') = \omega$, let $e_{\overline{v},\overline{v}'}$ be the matrix such that $e_{\overline{v},\overline{v}'}(\overline{v}, \overline{v}') = 1$, $e_{\overline{v},\overline{v}'}(v, v) = e(v, v)$ for all $v \in V$ and all other pairs are mapped to 0. It is easily verified that $e_{\overline{v},\overline{v}'}$ is a simple unstable idempotent and that $e^\sharp_{\overline{v},\overline{v}'}$ is the idempotent identical to $e_{\overline{v},\overline{v}'}$ except that $e^\sharp_{\overline{v},\overline{v}'}(\overline{v}, \overline{v}') = \omega$.

Let m the number of unstable pairs of e such that $e(\overline{v}, \overline{v}) = e(\overline{v}', \overline{v}') = \omega$ and $e_1, \dots, e_m$ be the associated simple unstable idempotents, in any order. Define $x = ee^\sharp_1 \dots e^\sharp_m e$, we show that $e^\sharp \le x$. Since e is idempotent, then $e^3 = e$ and by definition of the product, for all $v, v' \in V$, there exist v_0 such that $e(v, v') = \min(e(v, v_0), e(v_0, v_0), e(v_0, v'))$. For every $\ell \in 1 \dots m$, $e(v_0, v_0) = e_\ell(v_0, v_0) \le e^\sharp_\ell(v_0, v_0)$. Therefore, by definition of the product,

$$e(v, v') \le \min(e(v, v_0), e^\sharp_1(v_0, v_0), \dots, e^\sharp_m(v_0, v_0), e(v_0, v')) \le x(v, v').$$

This holds for every edge v, v', thus $e \dots x$.

To show that $e^\sharp \le x$, it remains to prove that all unstable pairs of e are mapped to ω in x. Let (v, v') be one. By definition of (un)stability (condition ii) in Lemma 5), there exists v_0, v'_0 such that $e(v, v_0) = e(v_0, v_0) = e(v'_0, v'_0) = e(v'_0, v') = \omega$ and $e(v_0, v'_0) = 1$.

As a consequence, there is an index ℓ such that $e_\ell = e_{\overline{v},\overline{v}'}$ and thus $e^\sharp_\ell(\overline{v}, \overline{v}') = \omega$. By definition of the product, $x(v, v')$ is greater or equal than

$$\min(e(v, v_0), e^\sharp_1(v_0, v_0), \dots, e^\sharp_\ell(v_0, v'_0), \dots, e^\sharp_m(v'_0, v'_0), e(v'_0, v')$$

and all those terms are ω hence $x(v, v') \ge \omega$. Finally, $e^\sharp \le x$. ◀

We now focus on simple unstable idempotents.

▶ **Lemma 40.** *Let e a simple unstable idempotent with (v, v') its unstable pair. Suppose that for all N there is a capacity word w such that*

- *there is a token flow transferring N tokens along every ω-loop for all N, and*
- *there is a path in w from v to v'.*

Then, for all N, there is a token flow over a capacity word w', transferring N tokens along every ω-loop, plus 1 from s to t.

Proof. Let $N \in \mathbb{N}$. By hypothesis, there exists a capacity word w, a token flow d on w transferring $N + 2$ tokens along every ω-loop, and a path π from v to v' in w. Since (v, v') is unstable and e is a simple unstable idempotent, there are ω-loops on both v and v'.

We say that a token τ crosses path π at date δ if the $\delta + 1$th state in π is the $\delta + 1$th state visited by τ.

We cut d as follows: Let $0 = \delta'_0 < \delta_1 \le \delta'_1 < \delta_2 < \dots < \delta'_m$ be such that δ_i is the first date at which a token τ_i crosses path π strictly after δ'_{i-1}, v_i is the initial vertex of τ_i and δ'_i is the last date at which a token τ'_i with initial vertex v_i crosses π.

Define d' the token flow where tokens τ_i and τ'_i have been deleted. Let τ be a fresh token. For each $i \in [0, m]$ let d_i be the token flow such that before δ'_i we make τ mimic τ'_i (if $i > 0$), between δ'_i and δ_{i+1} we make τ follow π and after δ_i we make τ mimic τ_{i+1} (if $i < m$).

Observe that d_i transfers at least N tokens along each ω-loop, plus τ which is transferred from v_i to v_{i+1}.

The composition of these flows $d_0 \cdots d_m$ transfers N tokens along each ω-loop, plus τ which is transferred from $v_0 = s$ to $v_{m+1} = t$. ◀

▶ **Lemma 41.** *Let e a simple unstable idempotent with (v, v') its unstable pair. Let w a capacity word such that*

- *there is a token flow transferring N tokens along every ω-loop for all N, and*
- *there is a path in w from v to v'*

Then there is a token flow transferring N tokens along every ω-loop, plus N from v to v', for all N.

Proof. Take a token flow transferring N tokens along every ω-loop, plus one from v to v' (it exists by the previous lemma). By iterating it N times, we transfer N tokens from v to v', and still N tokens along every ω-edge. ◀

To conclude, let e be an idempotent satisfying the property $(\diamondsuit)$, and let $e_1, \dots, e_m$ be as in Lemma 39. Since $e_i \leq e$ for all i, each e_i satisfies the property $(\diamondsuit)$. By Lemma 40, so does each $e_i^\sharp$, hence also $ee_1^\sharp \dots e_m^\sharp e$ since property $(\diamondsuit)$ is preserved by product. As $e^\sharp \leq ee_1^\sharp \dots e_m^\sharp e$, we obtain that $e^\sharp$ satisfies property $(\diamondsuit)$.

We have shown that property $(\diamondsuit)$ holds on $\{x_a \mid a \in A\}$, and is preserved by product and application of $\sharp$. As a result, it holds on all elements of $\mathcal{F}$.

A.4 Proof of Lemma 19

Proof of Lemma 19. By induction on $h \geq 0$. *Case* $h = 0$. The only valid expressions of height 1 are (abstractions of) single capacity letters. In the algorithm then correctly returns whether this is the case for x, either on line 3 or 5. *Case* $h \geq 1$. If there is a $\sharp$-expression of height $h \geq 1$ then its root is either labelled by a product $x_1 \cdot x_2$ and there are $\sharp$-expressions of height $h-1$ for the two x_1, or the root is labelled by $e^\sharp$ for an idempotent e and there exists a $\sharp$-expression of height $h-1$ for e. Either way, the algorithm correctly returns *true* in line 8 or 11. Otherwise, if no such expression exists, the algorithm correctly returns *false* in line 12. ◀

A.5 Checking whether $x_s \to_\ell x_t$ in polynomial space

Let x_s, x_t two C-configurations.

In case $\ell = 1$, $x_s \to_1 x_t$ reduces to solving $|A|$ instances of Max flow problem with maximal capacity C, which can be solved in time polynomial in $|Vl$ and the length of the binary representation of C. For every $a \in A$, the Max flow problem instance is denoted $I(a, x_s, x_t)$. It is obtained by adding a new source vertex v'_s with capacity constraints $x_s(v)$ on each edge $(v'_s, v), v \in V$ and a new target vertex v'_t with capacity constraints $x_t(v)$ on each edge $(v, v'_t), v \in V$, and checking the existence of a flow of maximal value C from v'_s to v'_t.

The existence of a sequential integral flow is performed by Algorithm 3. The calling depth height is $\leq \lceil \log_2(\ell) \rceil$hence it can be implemented and a stack of up to $\lceil \log_2(\ell) \rceil$ triplets of C-configurations and binary counter $\leq \ell$. This requires polynomial space in $\log_2(\ell)$, $|V|$ and $\log_2(C)$, plus the space for the $\ell = 1$ case, which is polynomial in $|A|$, $|V|$ and $\log_2(C)$.

▶ Remark 42. We obtain an alternate way to check unboundedness of flow values: simply check wether there is a sequential flow of value $K \cdot |V|^{536 \cdot |V|^{12}} + 1$ (see the proof of Theorem 22 for a way to do this in polynomial space). If no such sequential flow exists, then we have a bound, otherwise there exist sequential flows with unbounded values by Theorem 16 and 11.

Algorithm 3 Checking whether $x_s \to_\ell x_t$

```
1:  function IsIntSeqFlow(x_s, x_t, ℓ)
2:      if ℓ = 0 then
3:          return x_s = x_t
4:      if ℓ = 1 then
5:          for every capacity a ∈ A do
6:              if the instance I(a, x_s, x_t) has a flow of value C then
7:                  return true
8:          return false
9:      for every C-configuration x do
10:         b_s ← IsIntSeqFlow(x_s, x, ⌈ℓ/2⌉)
11:         b_t ← IsIntSeqFlow(x, x_t, ⌊ℓ/2⌋)
12:         if b_s and b_t then
13:             return true
14:     return false
```

B Proofs of Section 4

B.1 Proof of Theorem 26

We extend the definition of regular $\mathcal{J}$-length to elements of the semigroup.

▶ **Definition 43.** *The regular $\mathcal{J}$-length of an element $x \in \mathcal{S}$ is the maximal number h such that there exist idempotents $e_1, \dots, e_h \in E(\mathcal{S})$ such that $e_1 <_\mathcal{J} \dots <_\mathcal{J} e_h \leq_\mathcal{J} x$.*

Equivalently, it is the regular $\mathcal{J}$-length of the semigroup obtained by restricting $\mathcal{S}$ to $\{y \in \mathcal{S} \mid y \leq_\mathcal{J} x\}$.

We write $L(x)$ for the regular $\mathcal{J}$-length of x.

Let us recall two facts on Green relations. For an in-depth introduction to Green relations, see for instance [18].

▶ **Lemma 44.** *In a finite semigroup $\mathcal{S}$, every equivalence class for $\mathcal{H}$ contains at most one idempotent.*

▶ **Lemma 45.** *In a finite semigroup $\mathcal{S}$, for all $x, y \in \mathcal{S}$, if $x \mathcal{J} y$ and $x \leq_\mathcal{L} y$ (resp. $x \leq_\mathcal{R} y$) then $x \mathcal{L} y$ (resp. $x \mathcal{R} y$).*

We start by considering sequences of elements of $\mathcal{S}$ that remain within a layer of the semigroup of equal regular $\mathcal{J}$-length. We show that we can cut every such sequence into blocks, where every block is of the form $u\alpha\beta\gamma$, with u, α, γ short and α, β, γ evaluating to the same idempotent.

▶ **Lemma 46.** *Call a block a word of the form $u\alpha\beta\gamma$, with $\varphi_\alpha = \varphi_\beta = \varphi_\gamma \in E(\mathcal{S})$.*

Let $w \in \mathcal{S}^*$. A block decomposition of w is a sequence of tuples $(u_i, \alpha_i, \beta_i, \gamma_i)_{i \leq m}$, plus an extra word u_{m+1}, with:

- $w = u_1\alpha_1\beta_1\gamma_1 u_2 \dots \alpha_m\beta_m\gamma_m u_{m+1}$
- *All u_i, α_i and γ_i have length at most $R_\mathcal{S}(3)$*
- *For all i there exists $e_i \in E(\mathcal{S})$ such that $\varphi_\mathcal{S}(\alpha_i) = \varphi_\mathcal{S}(\beta_i) = \varphi_\mathcal{S}(\gamma_i) = e_i$.*

We start by cutting the word w in an unbounded number of blocks, and then merge some of those blocks to obtain a bounded number of them.

▶ **Lemma 47.** *Every word $w = x_1 \dots x_k \in \mathcal{S}^*$ has a block decomposition.*

Proof. Consider the following algorithm:

Algorithm 4 Algorithm to cut the word into blocks

```
i, j, p ← 1
while k − i > R_S(3) do
    while x_i ... x_j is not a block do
        j ← j + 1
    B_p ← x_i ... x_j
    i, j ← j + 1
    p ← p + 1
ū_p ← x_i ... x_k
```

Observe that the internal while loop must always terminate with $j \leq i + R_{\mathcal{S}}(3)$: by definition of $R_{\mathcal{S}}(3)$, if $j - i > R_{\mathcal{S}}(3)$, there must be three consecutive infixes in in $x_i \dots x_j$ evaluating to the same idempotent. As a consequence, $x_i \dots x_j$ must have a block as a prefix.

In particular, we cannot reach the end of the word while in the internal loop: before entering it, the external loop condition guarantees that $k - i > R_{\mathcal{S}}(3)$, hence we will find a block before reaching the end of the word.

Let r be the value of $p - 1$ at the end of the execution of Algorithm 4. We obtain a sequence of blocks $B_1, \dots, B_r$ and a suffix B_{r+1} such that $x_1 \dots x_k = B_1 \dots B_r \overline{u}_{r+1}$.

Since each B_i with $i \leq r$ is a block, we can define $\overline{u}_i, \overline{\alpha}_i, \overline{\beta}_i, \overline{\gamma}_i \in \mathcal{S}^*$ and $\overline{e}_i \in E(\mathcal{S})$ such that $B_i = \overline{u}_i \overline{\alpha}_i \overline{\beta}_i \overline{\gamma}_i$ and $\varphi_{\mathcal{S}}(\overline{\alpha}_i) = \varphi_{\mathcal{S}}(\overline{\beta}_i) = \varphi_{\mathcal{S}}(\overline{\gamma}_i) = \overline{e}_i$. Observe that all those words have length at most $R_{\mathcal{S}}(3)$. ◀

▶ **Lemma 48.** *Let $w = x_1 \dots x_k$ be such that $L(x_i) = L(\varphi_{\mathcal{S}}(w))$. Then w has a block decomposition with at most $|\mathcal{S}|R_{\mathcal{S}}(3)$ elements.*

Proof. In all that follows, given a block $B = (u, \alpha, \beta, \gamma)$, we write $\overline{B}$ for the word $u\alpha\beta\gamma$.

Let $B_1, \dots, B_m$ be a block decomposition, with $B_i = (u_i, \alpha_i, \beta_i, \gamma_i)$. We say that we can reduce it if there exist $i < j$ such that

$$\varphi_{\mathcal{S}}(\alpha_i) = \varphi_{\mathcal{S}}(\alpha_j) = \varphi_{\mathcal{S}}(\beta_i \gamma_i \overline{B}_{i+1} \dots \overline{B}_{j-1} u_j \alpha_j \beta_j).$$

In that case it reduces to the block decomposition

$$B_1, \dots, B_{i-1}, (u_i, \alpha_i, \beta_i \gamma_i \overline{B}_{i+1} \dots \overline{B}_{j-1} u_j \alpha_j \beta_j, \gamma_j), B_{j+1}, \dots, B_k.$$

Note that this sequence is also a block decomposition of w, with less elements than the initial one.

By Lemma 47, w has a block decomposition. We can reduce it until we cannot anymore. Let $B_1, \dots, B_m$ be a block decomposition of w that cannot be reduced, with $B_i = (u_i, \alpha_i, \beta_i, \gamma_i)$.

For all j let $e_j = \varphi_{\beta_j} \in E(\mathcal{S})$. We show that $m \leq |\mathcal{S}|R_{\mathcal{S}}(3)$, by contradiction. Suppose $m > |\mathcal{S}|R_{\mathcal{S}}(3)$. Then by the pigeonhole principle there exists e an idempotent which is equal to e_j for at least $R_{\mathcal{S}}(3) + 1$ distinct values of j. Let $j(0) < \dots < j(\ell)$ be such that $e_{j(i)} = e$ for all i and $\ell \geq R_{\mathcal{S}}(3)$.

For all $i \in [1, \ell]$ let $y_i = \gamma_{j(i-1)} \overline{B}_{j(i-1)+1} \dots \overline{B}_{j(i)-1} \alpha_{j(i)} \beta_{j(i)}$. Since $\ell \geq R_{\mathcal{S}}(3) \geq R_{\mathcal{S}}(1)$, by definition of $R_{\mathcal{S}}(1)$ there exist $i < i'$ such that $\varphi_{\mathcal{S}}(y_i \dots y_{i'})$ is an idempotent $f \in E(\mathcal{S})$.

We have $f = \varphi_{\mathcal{S}}(y_i \dots y_{i'})$ and thus

$$\begin{aligned} f &= \varphi_{\mathcal{S}}(\gamma_{j(i-1)})\varphi_{\mathcal{S}}(\overline{B}_{j(i-1)+1} \dots \overline{B}_{j(i')-1})\varphi_{\mathcal{S}}(\alpha_{j(i')}\beta_{j(i')}) \\ &= e\varphi_{\mathcal{S}}(\overline{B}_{j(i-1)+1} \dots \overline{B}_{j(i')-1})e. \end{aligned}$$

Hence $f <_{\mathcal{H}} e$.

Let ℓ be the regular $\mathcal{J}$-length of $x_1 \dots x_k$ (and of all x_i). Since e and f are the values of infixes of $x_1 \dots x_k$, they must both have regular $\mathcal{J}$-length ℓ, and thus $f \mathcal{J} e$. By Lemma 45, we have $f \mathcal{H} e$. Since an $\mathcal{H}$-class contains at most one idempotent by Lemma 44, we have $f = e$.

This means that $e = \varphi_{\mathcal{S}}(\gamma_{j(i-1)}\overline{B}_{j(i-1)+1} \dots \overline{B}_{j(i')-1}\alpha_{j(i')}\beta_{j(i')})$. Since $e(j(i)) = ej(i')$. As a consequence, the block decomposition is reducible, a contradiction. Hence $m \leq |\mathcal{S}|R_{\mathcal{S}}(3)$. ◀

▶ **Corollary 49.** *Let $w = x_1 \dots x_k \in \mathcal{S}^*$ be a word so that $L(\varphi_{\mathcal{S}}(x_0 \cdots x_k)) = L(x_i)$ for all i. There exists a summary of $x_1 \dots x_k$ of height at most $(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3))) + 3$.*

Proof. We use the decomposition from Lemma 47, and the same notations. We use a tree of height at most $\log(2m+1) \leq \log(2|\mathcal{S}|R_{\mathcal{S}}(3)+1)$ of product nodes to obtain one leaf for each u_i and one for each $\alpha_i\beta_i\gamma_i$. Then we apply idempotent nodes for each $\alpha_i\beta_i\gamma_i$ so that all leaves are labeled by some u_i, α_i or γ_i. We can then apply $\log_2(2R_{\mathcal{S}}(3))$ product nodes on each to obtain a tree where all leaves are labeled by some x_i.

In total we have built a tree of height at most $\log(2|\mathcal{S}|R_{\mathcal{S}}(3)+1) + 1 + \log_2(2R_{\mathcal{S}}(3)) \leq \log_2(|\mathcal{S}|) + 2\log_2(|R_{\mathcal{S}}(3)|) + 3$. ◀

We now show how to use this lemma to prove Theorem 26.

Proof of Theorem 26. We prove by strong induction on ℓ, the regular $\mathcal{J}$-length of $\varphi_{\mathcal{S}}(w)$, that w has a summary of height at most $\ell(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4)$.

Let $w \in \mathcal{S}^*$ such that $\varphi_{\mathcal{S}}(w)$ has regular $\mathcal{J}$-length ℓ, suppose we have the lemma for all words of regular $\mathcal{J}$-length $< \ell$.

We cut w into $w_1 a_1 \dots w_k a_k w_{k+1}$ so that w_i is the largest prefix of $w_i a_i \dots w_{k+1}$ of regular $\mathcal{J}$-length $< \ell$. Note that since w has regular $\mathcal{J}$-length ℓ, $w_i a_i$ must have regular $\mathcal{J}$-length ℓ as well.

For each i let $x_i = \varphi_{\mathcal{S}}(w_i a_i)$. All x_i have the same regular $\mathcal{J}$-length ℓ as $\varphi_{\mathcal{S}}(x_1 \dots x_k)$. By Corollary 49 there exists a summary τ of $x_1 \dots x_k$ of height bounded by $\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 3$.

Furthermore, by induction hypothesis we have a summary for each w_i, of height at most $(\ell-1)(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4)$. An additional product node gives us a summary τ_i for $w_i a_i$, of height at most $(\ell-1)(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4) + 1$.

We append τ_i at the leaf corresponding to x_i in τ, for all i. We obtain a summary of w, of height at most $\ell(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4)$.

The induction is proven. To obtain the theorem, we observe that ℓ is at most $L(\mathcal{S})$. By Theorem 23, we have $L(\mathcal{S}) \leq \log_2(R_{\mathcal{S}}(2)) \leq \log_2(R_{\mathcal{S}}(3))$. In conclusion, for every word $w \in \mathcal{S}^*$ there exists a summary of height at most $\log_2(R_{\mathcal{S}}(3))(\log_2(|\mathcal{S}|) + 2\log_2(R_{\mathcal{S}}(3)) + 4)$. ◀

B.2 Proof of Lemma 30

The following proof is inspired by [17]. For this section it is convenient to visualise elements of $\mathcal{F}$ as graphs over V. Recall that n is the number of vertices $|V|$. Given an idempotent element $e \in \mathcal{F}$, its *ω-graph* is the directed graph whose set of vertices is V, with an edge

from v to v' if and only if $e(v,v') = \omega$. Strongly connected components of this graph are called *ω-SCCs* of e. An ω-SCC C is *non-trivial* if there is at least one edge $e(v,v') = \omega$ with $v, v' \in C$. The set of non-trivial ω-SCCs is denoted $SCC_\omega(e)$. We also define the relation $R_\omega(e)$ over $SCC_\omega(e)$ such that $(C_1, C_2) \in R_e$ if there exist $v_1 \in C_1$ and $v_2 \in C_2$ with $e(v_1, v_2) = \omega$.

▶ Remark 50. The ω-graph of an idempotent e is necessarily transitive: for all $v_1, v_2, v_3 \in S$, if $e(v_1,v_2) = e(v_2,v_3) = \omega$ then, since $e \cdot e = e$, $e(v_1,v_3) = \omega$. As a consequence, given a non-trivial ω-SCC C, we necessarily have $e(v,v') = \omega$ for all $v, v' \in C$ (in particular, every state in C has a self-loop).

Another consequence is that if $(C_1, C_2) \in R_\omega(e)$ then $e(v_1, v_2) = \omega$ for all $v_1 \in C_1$ and $v_2 \in C_2$.

In order to bound the number of unstable idempotent nodes along a branch of a factorisation, we show that a certain quantity can only decrease or stay the same among idempotents along a branch, and has to decrease at every unstable idempotent node. The first component of that quantity is the number of ω-SCCs.

▶ **Lemma 51.** *For all idempotents e, f, if $e \leq_{\mathcal{J}} f$ then $|SCC_\omega(e)| \leq |SCC_\omega(f)|$.*

Proof. This proof is inspired by the one of [17, Proposition 5.5]. Since $e \leq_{\mathcal{J}} f$, there exist $x, y \in M$ such that $e = xfy$. We can assume without loss of generality that $x = xf$ and $y = fy$; otherwise we replace x by $x' = xf$ and y by $y' = fy$: we then have $x' = x'f$, $y' = fy'$ and $e = x'fy'$.

We define an injective function $\texttt{yields} : SCC_\omega(e) \to SCC_\omega(f)$ as follows. For each non-trivial ω-SCC C of e, we select any vertex $v_C \in C$. Since C is non-trivial, $e(v_C, v_C) = \omega$ by Remark 50. We select a vertex v such that $x(v_C, v) = f(v,v) = y(v, v_C) = \omega$ and define $\texttt{yields}(C)$ as the ω-SCC of v, which is non-trivial since $f(v,v) = \omega$.

Let us first argue that such a v always exists. Since $e = xfy$, there exist v', v'' such that $x(v_C, v') = f(v', v'') = y(v'', v_C) = \omega$. Hence by Lemma 5 there exists $\overline{v} \in S$ such that $f(v', \overline{v}) = f(\overline{v}, \overline{v}) = f(\overline{v}, v'')$. Since $x = xf$ and $y = fy$, we obtain $x(v_C, \overline{v}) = f(\overline{v}, \overline{v}) = y(\overline{v}, v_C) = \omega$.

It remains to show that this function is injective. Let C, C' be two ω-SCCs of e such that $\texttt{yields}(C) = \texttt{yields}(C')$. By definition of $\texttt{yields}$, we have two vertices $v \in C, v' \in C'$ and two others $\overline{v}, \overline{v}' \in \texttt{yields}(C) = \texttt{yields}(Ct)$ such that $x(v, \overline{v}) = x(v', \overline{v}') = y(\overline{v}, v) = y(\overline{v}', v') = \omega$. Since $\overline{v}, \overline{v}'$ are in the same ω-SCC of f, we have $f(\overline{v}, \overline{v}') = f(\overline{v}', \overline{v}) = \omega$ by Remark 50. As a consequence, since $e = xfy$, we obtain $e(v, v') = e(v', v) = \omega$, hence v and v' are in the same ω-SCC of e. ◀

The second component of that quantity is the size of the relation $R_\omega(e)$ over ω-SCCs.

▶ **Lemma 52.** *For all idempotents e and f such that $e \leq_{\mathcal{J}} f$ and $|SCC_\omega(e)| = |SCC_\omega(f)|$, we have $|R_\omega(e)| \geq |R_\omega(f)|$.*

Proof. If $|SCC_\omega(e)| = |SCC_\omega(f)|$ then the function $\texttt{yields}$ defined in the proof of Lemma 51 is actually a bijection.

We show that for all (C, C') in $R_\omega(f)$, $(\texttt{yields}^{-1}(C), \texttt{yields}^{-1}(C')) \in R_\omega(e)$, which implies the lemma. Let (C, C') in $R_\omega(f)$. Let $v \in \texttt{yields}^{-1}(C)$ and $v' \in \texttt{yields}^{-1}(C')$. By definition of $\texttt{yields}$, there exist $\overline{v} \in C$ and $\overline{v}' \in C'$ such that $x(v, \overline{v}) = f(\overline{v}, \overline{v}) = y(\overline{v}, v)$ and $x(v', \overline{v}') = f(\overline{v}', \overline{v}') = y(\overline{v}', v')$.

Furthermore, since $(C, C') \in R_\omega(f)$, there exist $v'' \in C, v''' \in C'$ such that $f(v'', v''') = \omega$. In light of Remark 50, and since both $\overline{v}, v''$ and $\overline{v}', v'$ share the same SCC, we obtain $f(\overline{v}, \overline{v}') = \omega$, and thus $e(v, v') = \omega$. As a result, we have $(\texttt{yields}^{-1}(C), \texttt{yields}^{-1}(C')) \in R_\omega(e)$. ◀

▶ **Lemma 53.** *For all idempotent $e \in M$, if $e \neq e^\sharp$ then either:*
- $|SCC_\omega(e^\sharp)| < |SCC_\omega(e)|$, *or*
- $|SCC_\omega(e^\sharp)| = |SCC_\omega(e)|$ *and* $|R_\omega(e^\sharp)| > |R_\omega(e)|$

Proof. For all $v, v' \in S$, if $e(v, v') = \omega$ then $e^\sharp(v, v') = \omega$. In other words, the ω-graph of e is a subgraph of the one of $e^\sharp$, and since both e and $e^\sharp$ are idempotent, both graphs are transitive. As a consequence, $|SCC_\omega(e)| \geq |SCC_\omega(e^\sharp)|$ and if $|SCC_\omega(e)| = |SCC_\omega(e^\sharp)|$, then actually the two graphs have exactly the same SCCs. In that case, since $e \neq e^\sharp$, there exist v, v_0, v_0', v' such that $e(v, v_0) = e(v_0', v') = \omega$ and $e(v_0, v_0') = 1$. By Lemma 5, we have v_1, v_1' such that $e(v, v_1) = e(v_1, v_1) = e(v_1, v_0) = \omega$ and $e(v_0', v_1') = e(v_1', v_1') = e(v_1', v') = \omega$. Since e is idempotent and $e(v_0, v_0') = 1$, we have $e(v_1, v_1') \leq 1$ and similarly since $e^\sharp(v_0, v_0') = \omega$ we have $e^\sharp(v_1, v_1') = \omega$. Hence the ω-graph of $e^\sharp$ must have an extra edge between non-trivial ω-SCCs. Since the graphs are transitive, this extra edge has to be from an ω-SCC C to another C' such that there are no edges from C to C' in e. As a result, $|R_\omega(e^\sharp)| > |R_\omega(e)|$. ◀

Proof of Lemma 30. Let $e_1, \ldots, e_m$ be idempotents in $\mathcal{F}$ such that $e_i^\sharp \leq_{\mathcal{J}} e_{i+1}$ for all i. For each i we define

$$(k_i, p_i) = (n - |SCC_\omega(e_i)|, |R_\omega(e_i)|).$$

Observe that k_i ranges from 0 to n, since the number of ω-SCCs cannot exceed the number of nodes. As for p_i, it ranges from 0 to $n-1$: if we had $p_i > n-1$, then also $p_i > |SCC_\omega(e_i)| - 1$ and the relation $R_\omega(e_i)$ would have a cycle. This is impossible as we would then have two distinct ω-SCCs connected in both directions.

By Lemmas 51 and 52, $(k_i, p_i)_{1 \leq i \leq k}$ is non-increasing for the lexicographic ordering. As a result, it can strictly decrease only $n(n-1) \leq n^2 - 1$ times. As a consequence, there exists i such that $(k_i, p_i) = (k_{i+1}, p_{i+1})$, implying that $e_i = e_i^\sharp$ by Lemma 53. ◀

C Regular constraints

The aim of this section is to prove Theorem 37.

We extend the notion of fair unboundedness witness: A *fair unboundedness witness for $\mathcal{A}$* is an element $(q, x, q') \in \mathcal{F}_\mathcal{A}$ such that $q \in I$, $q' \in F$ and $x(v_s, v_t) = \omega$ for all $(v_s, v_t) \in E$,

▶ **Lemma 54** (Adapted from Lemma 14)**.** *If the labeled flow semigroup contains an element (q_i, x, q_f) with $q_i \in I$, $q_f \in F$ and $x(v_s, v_t) = \omega$ for all $(v_s, v_t) \in E$ then the answer to the* Fair sequential flow problem with regular constraints *is ω.*

Proof. We show that all elements of $\mathcal{F}_\mathcal{A}$ besides $\bot$ satisfy the following property: For all $(q, x, q') \in \mathcal{F}_\mathcal{A}$, for all $N \in \mathbb{N}$, there exist a capacity word w and a token flow d over w such that w labels a run of $\mathcal{A}$ from q to q' and for all $v, v' \in V$, the following two conditions are satisfied:

1. $x(v, v') = \omega \Rightarrow g(d)(v, v') \geq N$
2. $x(v, v') \geq 1 \Rightarrow$ there is a path in w from v to v'

The proof is then a straightforward adaptation of Appendix A.3. ◀

For the other direction, we need to bound the regular $\mathcal{J}$-length of $\mathcal{F}_\mathcal{A}$. Although bounds on the Ramsey function can be inferred through Theorem 23, we obtain better bounds by a simple pigeonhole argument.

▶ **Lemma 55.** *The labeled flow semigroup satisfies the following inequalities:*

1. $|\mathcal{F}_\mathcal{A}| \leq m^2|\mathcal{F}| + 1 \leq m^2 3^{n^2} + 1$,
2. $L(\mathcal{F}_\mathcal{A}) \leq L(\mathcal{F}) + 1 \leq (n^2+n+2)^2/4 + 1$, *and*
3. *for all* $k \in \mathbb{N}$, $R_{\mathcal{F}_\mathcal{A}}(k) \leq R_\mathcal{F}(k(m+1))$. *In particular,* $R_{\mathcal{F}_\mathcal{A}}(3) \leq (3(m+1)|\mathcal{F}|^4)^{L(\mathcal{F})} \leq (m+1)^{16n^4} 3^{32n^6}$

Proof. 1. The bound on the size follows from the simple observation that for every element of $\mathcal{F}_\mathcal{A}$ of the form (q, x, q'), by definition of $\mathcal{F}_\mathcal{A}$, x can be obtained from the $(x_a)_{a\in A}$ using product and $\sharp$. Hence $x \in \mathcal{F}$, and thus $|\mathcal{F}_\mathcal{A}| \leq |Q|^2|\mathcal{F}| + 1$.

2. It was shown by Jecker that the regular $\mathcal{J}$-length $L(\mathcal{S})$ of a finite semigroup $\mathcal{S}$ is equal to the largest m such that the max-monoid H_m can be embedded in $\mathcal{S}$ [14, Appendix B]. We use this to show that the regular $\mathcal{J}$-length of $\mathcal{F}_\mathcal{A}$ is at most the one of $\mathcal{F}$ plus one.

   Let $m \in \mathbb{N}$, suppose we have an injective morphism $\psi : H_{m+1} \to \mathcal{F}_\mathcal{A}$. At most one element can be mapped to $\bot$, and if there is one it must be $m+1$. We exhibit an injective morphism $H_m \to \mathcal{F}$.

   For all $i \in [1, m]$, we must have $\psi(i) \star \psi(i) = \psi(\max(i, i)) = \psi(i)$, hence all $\psi(i)$ are idempotents, of the form (q_i, e_i, q_i). Further, for all $i < j$ we have $\psi(i) \star \psi(j) = \psi(\max(i, j)) = \psi(j)$, hence $q_i = q_j$. As a result, there is a state q such that $\psi(i) = (q, e_i, q)$ for all i. Since ψ is injective, all e_i must be distinct.

   It suffices to observe that the function $\psi' : H_m \to \mathcal{F}$ with $\psi'(i) = e_i$ is an injective morphism.

   As a consequence, $L(\mathcal{F}_\mathcal{A}) \leq L(\mathcal{F}) + 1$.

3. For the Ramsey function, we use a simple pigeonhole argument. Let $\pi : \mathcal{F}_\mathcal{A} \to \mathcal{F}$ be the morphism projecting each triple $(q, x, q') \in \mathcal{F}_\mathcal{A}$ to its middle component $x \in \mathcal{F}$. Let $k \in \mathbb{N}$, let $w \in \mathcal{F}_\mathcal{A}^*$ be a word of length $kR_\mathcal{F}(k|Q|)$. We show that w contains k consecutive infixes evaluating to the same idempotent.

   We first cut w in k parts of length $R_\mathcal{F}(k|Q|)$: $w = w_1 \dots w_k$. If $\varphi_{\mathcal{F}_\mathcal{A}}(w_j) = \bot$ for all j, then we have the desired consecutive infixes. Otherwise, there is some p such that $\varphi_{\mathcal{F}_\mathcal{A}}(w_j) \neq \bot$.

   Since w_p has length at least $R_\mathcal{F}(k|Q|)$, by definition of the Ramsey function it contains an infix $u_1 \dots u_{k(|Q|)}$ such that there is an idempotent $e \in \mathcal{F}$ with $\varphi_\mathcal{F}(\pi(u_i)) = e$ for all i. Since $\varphi_{\mathcal{F}_\mathcal{A}}(w_j) \neq \bot$, there are states $q_0, \dots, q_{k|Q|}$ such that we have $\varphi_\mathcal{F}(\pi(u_i)) = (q_i, e_i, q_{i+1})$ for all i. By the pigeonhole principle, there exist $i_0 < \dots < i_k$ such that $q_{i_0} = \dots = q_{i_k}$. Let q be that state. As a result, the consecutive infixes $u_{i_j} \dots u_{i_{j+1}-1}$ all evaluate to the same idempotent (q, e, q). ◀

▶ **Lemma 56** (Adapted from Theorem 16). *For all capacity words w, for all $q, q' \in Q$, if there is a run reading w from q to q' in $\mathcal{A}$ then there exists x such that $(q, x, q') \in \mathcal{F}_\mathcal{A}$ and for all $v, v' \in V$,*

$$x(v, v') = 0 \implies g(d)(v, v') = 0 \qquad \text{and}$$
$$x(v, v') = 1 \implies g(d)(v, v') \leq K(2|V|)^{(170\log_2(m)+835)n^{12}}.$$

Proof. To begin with, we can apply Theorem 26 to $\mathcal{F}_\mathcal{A}$:

Every word $w \in \mathcal{F}_\mathcal{A}^*$ has a summary of height at most

$$L(\mathcal{F}_\mathcal{A})(\log_2(|\mathcal{F}_\mathcal{A}|) + 2\log_2(R_{\mathcal{F}_\mathcal{A}}(3)) + 4).$$

By Lemma 55, this is bounded by

$$
\begin{aligned}
&(\frac{(n^2+n+2)^2}{4}+1)(\log_2(|Q|^2 3^{n^2}+1)+2\log_2((|Q|+1)^{16n^4}3^{32n^6})+4)\\
\leq&(5n^4)(2\log_2(|Q|)+2n^2+1+32n^4\log_2(|Q|)+32n^4+128n^6+4)\\
\leq&(170\log_2(|Q|)+835)n^{10}
\end{aligned}
$$

We extend the $\sharp$-summaries to the labeled flow semigroup. A *labeled $\sharp$-summary* is defined analogously to a $\sharp$-summary, but the labels are in $\mathcal{F}_\mathcal{A} \times \mathcal{F}_\mathcal{A}^*$ instead of $\mathcal{F} \times \mathcal{F}^*$.

Lemma 30 can be extended to $\mathcal{F}_\mathcal{A}$: Let $(q_1, e_1, q_1), \ldots, (q_{n^2}, e_{n^2}, q_{n^2})$ be idempotents of $\mathcal{F}_\mathcal{A}$ (different from $\bot$) such that

$$(q_i, e_i, q_i)^\sharp \leq_\mathcal{J} (q_{i+1}, e_{i+1}, q_{i+1})$$

for all i. Then, for all i we have $e_i^\sharp \leq_\mathcal{J} e_{i+1}$. By Lemma 30, there exists i such that $e_i^\sharp = e_i$. As a consequence, $(q_i, e_i, q_i)^\sharp = (q_i, e_i^\sharp, q_i) = (q_i, e_i, q_i)$.

Thus, we can repeat the proof of Theorem 33 for $\mathcal{F}_\mathcal{A}$. We obtain that each word has a labeled $\sharp$-summary of height at most $(170\log_2(|Q|)+835)n^{12}$.

This lets us in turn adapt Theorem 16: if K is the largest finite coordinate in capacities, and w has a labeled $\sharp$-summary whose result is x with $x(s,t) \leq 1$ then the maximal flow in w from s to t is bounded by $K(2n)^{(170\log_2(|Q|)+835)n^{12}}$. ◀

This leads to the following corollary which allows to compute the optimal sequential flows in polynomial space just like in the previous cases.

▶ **Lemma 57** (Adapted from Lemma 17). *If the answer to the* Fair sequential flow problem with regular constraints *is ω then there is a fair unboundedness witness for $\mathcal{A}$ in $\mathcal{F}_\mathcal{A}$.*

Proof. Since we have capacity words in L with token flows of unbounded values, in particular there exists a capacity word $w \in L$ with a token flow d transferring more than $K(2|V|)^{(170\log_2(m)+835)n^{12}}$ tokens between each pair of vertices in E. Let $q \in I, q' \in F$ such that w labels a run from q to q' in $\mathcal{A}$. We apply Lemma 56 to d. By case inspection, the only possible value of $x(v,v')$ is ω, thus (q, x, q') is a fair unboundedness witness for $\mathcal{A}$. ◀