Verifying Unboundedness via Amalgamation

Ashwani Anand
ashwani@mpi-sws.org
Max-Planck-Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

Sylvain Schmitz
schmitz@irif.fr
Université Paris Cité, CNRS, IRIF
Paris, France

Lia Schütze
lschuetze@mpi-sws.org
Max-Planck-Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

Georg Zetzsche
georg@mpi-sws.org
Max-Planck-Institute for Software Systems (MPI-SWS)
Kaiserslautern, Germany

ABSTRACT

Well-structured transition systems (WSTS) are an abstract family of systems that encompasses a vast landscape of infinite-state systems. By requiring a well-quasi-ordering (wqo) on the set of states, a WSTS enables generic algorithms for classic verification tasks such as coverability and termination. However, even for systems that are WSTS like vector addition systems (VAS), the framework is notoriously ill-equipped to analyse reachability (as opposed to coverability). Moreover, some important types of infinite-state systems fall out of WSTS' scope entirely, such as pushdown systems (PDS).

Inspired by recent algorithmic techniques on VAS, we propose an abstract notion of systems where the set of *runs* is equipped with a wqo and supports *amalgamation* of runs. We show that it subsumes a large class of infinite-state systems, including (reachability languages of) VAS and PDS, and even all systems from the abstract framework of valence systems, except for those already known to be Turing-complete.

Moreover, this abstract setting enables simple and general algorithmic solutions to *unboundedness problems*, which have received much attention in recent years. We present algorithms for the (i) simultaneous unboundedness problem (which implies computability of downward closures and decidability of separability by piecewise testable languages), (ii) computing priority downward closures, (iii) deciding whether a language is bounded, meaning included in $w_1^* \cdots w_k^*$ for some words $w_1, \ldots, w_k$, and (iv) effective regularity of unary languages. This leads to either drastically simpler proofs or new decidability results for a rich variety of systems.

CCS CONCEPTS

• **Theory of computation** → **Formal languages and automata theory**; *Verification by model checking*.

KEYWORDS

Verification, well-quasi-order, valence systems, vector addition system, simultaneous unboundedness, separability, downward closure

1 INTRODUCTION

Well Structured Transition Systems (WSTS for short) form an abstract family of systems, for which generic verification algorithms are available off-the-shelf [1, 25]. They were invented in the late 1980s [23] in an attempt to abstract and generalise from techniques originally designed to reason about vector addition systems (VASS) and their variants, but they have since been shown to yield decision procedures outside the field of formal verification, for various logics, automata models, or proof systems.

One reason for their success is the simplicity of the abstract definition: those are transition systems, where the configurations are equipped with a *well-quasi-ordering* (wqo) [41, 44, 58], which is "compatible" with transition steps: if there is a step $c \to c'$ and a configuration $d \geq c$, then there is a configuration $d' \geq c'$ reached by a step $d \to d'$. From this simple definition and under basic effectiveness assumptions, one can decide for instance whether starting from a source configuration one can reach in finitely many steps a configuration larger or equal to a target—aka the *coverability problem*, or restated in terms of formal languages, the emptiness problem for coverability languages [29].

Despite their wide applicability and the filiation from vector addition systems, WSTS have a few essential limitations. One is that the framework is of no avail for the *reachability problem*, which is actually undecidable for many classes of WSTS [20]—though famously decidable for vector addition systems [43, 45, 46, 51]. Another is that pushdown systems provide a notable exception to the applicability of the definition [25, end of Sec. 7]—which might help explain why we still have very limited understanding of vector addition systems extended by a pushdown store (PVASS) [50, 59].

Towards Unboundedness Problems. The case of vector addition systems is worth further attention: their reachability problem can be decided through the construction of a so-called *KLM decomposition*, named after Kosaraju [43], Lambert [45], and Mayr [51]: this is a structure capturing all the possible runs of the system between the source and target configurations. Beyond reachability, Lambert [45, Sec. 5] already observed that this decomposition provided considerably more information, allowing in particular to derive a pumping lemma for VASS (reachability) languages. Habermehl et al. [33] also show that the *downward closure* of VASS languages can be computed from the KLM decomposition.

These applications of the KLM decomposition were pushed further by Czerwiński et al. [19] to show the decidability of a family of *unboundedness problems* on VASS languages—informally, of decision problems for formal languages where one asks for the existence of infinitely many words of some shape.

- One example is provided by *separability problems* [e.g., 16–18, 31], where we are given two languages K and L and we want to decide whether there exists a language R such that $K \subseteq R$ and $L \cap R = \emptyset$. Here, R is constrained to belong to a class $\mathcal{S}$ of *separators*, usually the regular languages or a subclass thereof. Deciding separability can be seen as an unboundedness problem: Suppose that for two words u, v, one defines their distance by taking the minimal size (of a finite monoid) of a language in $\mathcal{S}$ separating u and v. Then for most classes $\mathcal{S}$, we have that two languages K and L are inseparable by $\mathcal{S}$ if and only if this distance between K and L is unbounded.

- Another, already mentioned example is computing *downward closures*. A well known consequence of Higman's Lemma [41] is that for every language L, the set $L\!\downarrow$ of all scattered subwords of L is a regular language [38]. However, effectively computing an automaton for $L\!\downarrow$ is often a difficult task [15, 33]. In [63], it was shown that computing downward closures often reduces to the *simultaneous unboundedness problem* (SUP), which asks for a given language $L \subseteq a_1^* \cdots a_n^*$, whether for every $k \in \mathbb{N}$, there exists a word $a_1^{x_1} \cdots a_n^{x_n} \in L$ with $x_1, \ldots, x_n \geq k$. Downward closure computation (also beyond the ordinary subword ordering [66]) based on the SUP have been studied in several papers in recent years [4, 36, 55]. A refinement of this problem recently investigated in [2] is to work with a *priority embedding*, allowing to represent congestion control policies based on priorities assigned to messages [32], instead of the scattered subword ordering.
- A third example is deciding whether a given language $L \subseteq \Sigma^*$ is *(language) bounded*, meaning whether there exist a $k \in \mathbb{N}$ and $w_1, \ldots, w_k \in \Sigma^*$ such that $L \subseteq w_1^* \cdots w_k^*$. Deciding language boundednes is motivated by the fact that bounded languages have pleasant decidability properties [e.g., 6, 12].

A Generic Approach. The previous applications of KLM decompositions show the decidability of a whole range of properties besides reachability in vector addition systems. Our main motivation in this paper is to study how to generalise the approach beyond vector addition systems. In this, we take our inspiration from Leroux and Schmitz [49], who showed that KLM decompositions could be recast as computing the downward closure of the set of runs between source and target configuration with respect to an embedding relation between runs. Crucially, this embedding relation is a well-quasi-ordering, and enjoys an *amalgamation* property—a notion from model theory, where embeddings of a structure A into structures B and C can be combined into embeddings into a superstructure D of B and C. As the runs of PVASS can be equipped with an embedding relation enjoying the same properties [48], this opens the way for an abstract framework orthogonal to WSTS, that generalises from vector addition systems, and where well-quasi-ordered run embeddings and amalgamation play a role akin to well-quasi-ordered configurations and compatibility.

1.1 Contributions

We introduce in Section 3 a general notion of *(concatenative) amalgamation systems* that consist of a set of *runs* equipped with a wqo and such that any two runs with a common subrun can be *amalgamated*. We show that our notion of amalgamation systems both (i) permits extremely simple decidability arguments for the unboundedness problems mentioned above, but also (ii) applies to a wide variety of infinite-state systems. In particular, while run amalgamation has been used for concrete types of systems to prove structural properties [24, 48] or in specialised subroutines of complex procedures [13, 18], we identify amalgamation as a powerful algorithmic tool that is often sufficient on its own for solving prominent problems. Let us elaborate on (i) and (ii).

Algorithmic Properties. Regarding the computational properties of amalgamation systems, we show that under mild effectiveness assumptions, for concatenative amalgamation systems with decidable emptiness, downward closures are computable, priority downward closures are computable, whether the accepted language is a bounded language is decidable, and all languages over one letter are effectively regular. More specifically, we show in Section 4 that if we assume that our class of systems is effective and closed under rational transductions [7], then all these effectiveness results hold as soon as emptiness is decidable.

Main Theorem A. *For every language class that supports concatenative amalgamation and is effectively closed under rational transductions, the following are equivalent:*

1. *The simultaneous unboundedness problem is decidable.*
2. *Downward closures are computable.*
3. *Separability by piecewise testable languages is decidable.*
4. *Language boundedness is decidable.*
5. *Unary languages are effectively regular.*
6. *Priority downward closures are computable.*
7. *Emptiness is decidable.*

Here, by *supports concatenative amalgamation*, we mean that every language in the class is recognised by some (concatenative) amalgamation system (see Section 3 for the full definition).

problems (1)–(6) are usually considered much more difficult for infinite-state systems than emptiness (7). For instance, emptiness is decidable for lossy channel systems and lossy counter machines [11, 52], but e.g., downward closures are not effective [11] and language boundedness is undecidable [12]—incidentally, these examples show that amalgamation systems are incomparable with WSTS. Moreover, when applied to examples of amalgamation systems, Main Theorem A often yields new or drastically simpler alternative proofs of decidability (see below for consequences).

Amalgamation Systems Everywhere! Regarding examples of amalgamation systems, we investigate the class of *valence automata* over graph monoids [65, 67]. These form an abstract model of systems with a finite-state control and a storage mechanism, which is usually infinite-state. The storage mechanism is specified by an undirected graph $\Gamma = (V, E)$ where self-loops are allowed. By choosing suitable graphs, one recovers various concrete infinite-state models from the literature. Examples include Turing machines, VASS, integer VASS, pushdown automata, and combinations, like pushdown VASS (PVASS). Valence automata have been studied over the last decade [26, 64, 65, 67, 68], and identifying the graphs Γ leading to a decidable emptiness problem is a challenging open question.

If one rules out graphs that are known to result in Turing-completeness, then the remaining storage mechanisms can be classified into three classes, dubbed SC^-, $\mathsf{SC}^\pm$, and SC^+ in [65, p. 185], obtained by *adding counters* and *building stacks*. Adding counters means that we take a storage mechanism and combine it with additional counters: either "blind" $\mathbb{Z}$-counters (which can go below zero) or "partially blind" $\mathbb{N}$-counters (which have to stay non-negative, like in a VASS). Building stacks means that we take a storage mechanism and define a new one that allows stacks, where each entry is a configuration of the original storage mechanism. One can operate on the top-most entry as specified by the original mechanism; by a push operation, one can create a fresh empty entry on top, and using a pop, one can remove an empty topmost entry. For SC^- and $\mathsf{SC}^\pm$, emptiness is decidable [67], while for SC^+ it is open.

We show in Section 5 that for every graph Γ, valence automata over Γ are amalgamation systems, *unless* reachability is already known to be undecidable. Note that it is unavoidable to rule out certain graphs: for instance, Turing machines cannot be amalgamation systems. By showing that valence automata over all the remaining graphs (namely in SC^+) lead to amalgamation systems, we obtain a very general characterisation of decidable unboundedness problems over the entire class of valence automata.

Main Theorem B. *For every graph Γ, the following are equivalent for valence automata over Γ:*

(1) The simultaneous unboundedness problem is decidable.
(2) Downward closures are computable.
(3) Separability by piecewise testable languages is decidable.
(4) Language boundedness is decidable.
(5) Unary languages are effectively regular.
(6) Priority downward closures are computable.
(7) Emptiness is decidable.

1.2 Consequences

There are several examples of infinite-state systems where our approach either yields new results or new (much simpler) proofs. Let us mention some of them.

Vector Addition Systems. A first important insight is that full-fledged KLM decompositions are not required in order to solve the unboundedness problems of Main Theorem A: our proofs show precisely that a black-box access to an oracle for reachability along with simple reasoning on amalgamated runs suffice. This should be contrasted with the rather more involved arguments used in the case of VASS to show the computability of downward closures [33], decidability of PTL-separability [17], decidability of language boundedness [19], and effectively regular unary languages [40]. Also, now that the proofs are decorrelated from the KLM decomposition, one can use any algorithm for VASS reachability (like Leroux's simple invariant-based algorithm [46]) to derive the decidability of these problems. Moreover, the computability of priority downward closures for VASS languages is a new result.

Models with Decidable Emptiness. Beyond VASS, there is a hierarchy of infinite-state systems, within the framework of valence automata, where decidability of emptiness is known, namely the classes SC^- and $\mathsf{SC}^\pm$ [67]. The difference between SC^- and $\mathsf{SC}^\pm$ is that when we apply "building stacks" and "adding counters," then in SC^-, we can only ever add $\mathbb{Z}$-counters, while in $\mathsf{SC}^\pm$, we can first add $\mathbb{N}$-counters, then build stacks, but afterwards only add $\mathbb{Z}$-counters (in alternation with building stacks). In particular, by starting with $\mathbb{N}$-counters and then building stacks, one obtains automata with a stack, where each entry can contains $\mathbb{N}$-counters; these are equivalent to the *sequential recursive Petri nets* of Haddad and Poitrenaud [34, 35] and generalise both pushdown automata and VASS.

These classes differ in what was already known about decidability. For SC^-, we knew that emptiness was decidable and that downward closures were computable [63, 64], and effective regularity of unary languages was known for SC^-, because they have semilinear Parikh images (this is not the case for $\mathsf{SC}^\pm$) [9]. The decidability of language boundedness was known for the smaller SC^--subclass of pushdown automata with reversal-bounded counters [6] (which model recursive programs with numeric data types [37]). For $\mathsf{SC}^\pm$, we knew that emptiness was decidable [67], and the decidability of the SUP could be derived from that proof. Main Theorem B implies decidability of all problems (1)–(6) for the entire class $\mathsf{SC}^\pm$.

PVASS and their Restrictions. In SC^+, one can arbitrarily alternate between adding $\mathbb{N}$-counters and building stacks. Since $\mathbb{Z}$-counters can always be simulated by $\mathbb{N}$-counters, SC^+ is more powerful than $\mathsf{SC}^\pm$, but the decidability of emptiness is open, the simplest open example being one-dimensional PVASS [67, Prop. 3.6]—also known as the *Finkel Problem.* For all these models, our results imply that decidable emptiness will immediately imply the other properties of Main Theorem A.

Furthermore, since PVASS are amalgamation systems, so are all systems that have a (language-)equivalent PVASS. Two examples where emptiness is known to be decidable come to mind: VASS with *nested zero tests* ($\mathsf{VASS}_{\mathrm{nz}}$) [3], and PVASS where the stack behaviour is *oscillation bounded* [27].

- A $\mathsf{VASS}_{\mathrm{nz}}$ is a VASS that has for each i an operation that tests all counters $1, \dots, i$ for zero at the same time. Reachability is decidable for $\mathsf{VASS}_{\mathrm{nz}}$ [8, 57] and the clover computation of Bonnet [8, Thm. 16] can be used to show decidability of the SUP (and thus downward closure computability and PTL separability); the remaining (4)–(6) in Main Theorem A are new results.
- PVASS with an oscillation bounded behaviour [27] are equivalent to PVASS where the stack behaviour is specified by a finite-index context-free language, and thus emptiness is decidable [3]—while the latter decidability result relies on $\mathsf{VASS}_{\mathrm{nz}}$, these PVASS appear to be more expressive in terms of accepted languages. Thus, all the algorithmic properties of Main Theorem A apply to these as well.

The paper is structured as follows. After some preliminaries in Section 2, we define the notion of an amalgamation system in Section 3. To illustrate the notion, we also present a few example systems. In Section 4, we present simple one-size-fits-all algorithms for the unboundedness properties in Main Theorem A for general amalgamation systems. Finally, in Section 5, we show that valence automata with graphs in SC^+ are amalgamation systems and prove Main Theorem B.

2 WELL-QUASI-ORDERS

We recall in this section the basic definitions for well-quasi-orders [41, 44, 58] and introduce some of the notations used in the paper.

Quasi-orders. A *quasi-order* (qo) is a pair $(X, \leq)$ where X is a set and $\leq \subseteq X \times X$ is a transitive reflexive binary relation. We write

$x < x'$ if $x \leq x'$ but $x' \not\leq x$. If $Y \subseteq X$, then it defines an *induced* qo when using the quasi-ordering $\leq \cap\, Y \times Y$.

Well-quasi-orders. A (finite or infinite) sequence $x_0, x_1, \ldots$ of elements from X is *good* if there exist $i < j$ such that $x_i \leq x_j$; the sequence is otherwise called *bad*. A qo $(X, \leq)$ is a *well-quasi-order* (wqo) if bad sequences are finite. Well-quasi-orders also enjoy the *finite basis property*: every subset $Y \subseteq X$ has a finite subset $B \subseteq Y$ such that for every $y \in Y$ there is $x \in B$ with $x \leq y$.

For instance, if S is a finite set, then $(S, =)$ is a wqo by the Pigeonhole Principle. For another example, $(\mathbb{N}, \leq)$ with the usual ordering is a wqo because bad sequences are strictly descending and the ordering is well-founded. By definition, a qo induced inside a wqo is also a wqo.

Vectors. By Dickson's Lemma, if $(X, \leq_X)$ and $(Y, \leq_Y)$ are two wqos, then so is their Cartesian product $X \times Y$ with the product (i.e., componentwise) ordering defined by $(x, y) \leq (x', y')$ if $x \leq_X x'$ and $y \leq_Y y'$. In particular, vectors $\mathbf{u} \in \mathbb{N}^d$ are well-quasi-ordered by the product ordering.

Words. Let X be a set; we write X^* for the set of finite sequences (or *words*) with letters taken from X. We write ε for the empty word, and define $X_\varepsilon \stackrel{\text{def}}{=} X \cup \{\varepsilon\}$ when we want to treat it as a letter.

If $(X, \leq)$ is a wqo, then by Higman's Lemma, so is $(X^*, \leq_*)$ where $w \leq_* w'$ if there exists a (scattered word) *embedding*, i.e., a strictly monotone map $f\colon [1, |w|] \to [1, |w'|]$ such that $w(i) \leq w'(f(i))$ for all $i \in [1, |w|]$. For instance, over the wqo $(\{a, b, c, r\}, =)$, $acab \leq_*$ $\underline{a}br\underline{a}\underline{c}a\underline{d}\underline{a}\underline{b}ra$, where we have underlined the positions in the image of the embedding.

Trees. Let X be a set. The set of (finite, ordered) X*-labelled trees* $T(X)$ is the smallest set such that, if $x \in X$, $n \in \mathbb{N}$, and $t_1, \ldots, t_n \in T(X)$ then the tree with root labelled by x and with immediate subtrees $t_1, \ldots, t_n$, denoted by $x[t_1, \ldots, t_n]$, is in $T(X)$. If $t \in T(X)$ and $p \in \mathbb{N}^*$, the *subtree of* t *at* p, written t/p, if it exists, is defined inductively by $t/\varepsilon \stackrel{\text{def}}{=} t$ and $x[t_1, \ldots, t_n]/(i \cdot p') \stackrel{\text{def}}{=} t_i/p'$ when $i \in [1, n]$.

If $(X, \leq)$ is a wqo, then $(T(X), \leq_T)$ is also a wqo by Kruskal's Tree Theorem, where $\leq_T$ denotes the *homeomorphic tree embedding* relation: $x[t_1, \ldots, t_n] \leq_T t$ if there exists a subtree $t/p = x'[t'_1, \ldots, t'_m]$ of t for some p, such that (i) $x \leq x'$ and (ii) there exist $1 \leq j_1 < \cdots < j_n \leq m$ such that $t_i \leq_T t'_{j_i}$ for all $i \in [1, n]$—this second condition corresponds to finding a word embedding between $t_1 \cdots t_n$ and $t'_1 \cdots t'_m$ with respect to $\leq_T$.

3 AMALGAMATION SYSTEMS

Broadly speaking, an amalgamation system consists of an infinite set of runs that are ordered by an embedding relation. Moreover, under certain circumstances, we can combine multiple runs into a new one. We define amalgamation systems in the forthcoming Section 3.1, before illustrating the concept with some concrete examples in sections 3.2 to 3.4—some of the proofs details that they are indeed amalgamation systems are deferred to in Section 5. For these basic examples, Table 1 presents the known results pertaining to Main Theorem A.

Table 1: Known effectiveness results for regular (Section 3.2), VASS (Section 3.3), and context-free (Section 3.4) languages.

	Reg	VASS	CFL
(1) SUP	[63]	[63]	[63]
(2) $L{\downarrow}$	folklore	[33]	[15, 60]
(3) PTL sep.	[56]	[17]	[17]
(4) boundedness	[30]	[19]	[30]
(5) unary eff. reg.	[54]	[40]	[54]
(6) $L{\downarrow}_{\mathrm{P}}$	[2]	Thm. A	[2]
(7) emptiness	folklore	[43, 46, 51]	folklore

3.1 Concatenative Amalgamation Systems

A *(concatenative) amalgamation system* is a tuple $S = (\Sigma, R, E, \mathsf{can})$, where Σ is a finite alphabet and R is a (usually infinite) set of *runs*. Each run $\rho \in R$ has an associated *canonical decomposition* $\mathsf{can}(\rho) = u_1 \cdots u_n \in \Sigma_\varepsilon^*$ of some length $|\rho|_{\mathsf{can}} = n$, with every $u_i \in \Sigma_\varepsilon$. The corresponding *accepted word* $\mathsf{yield}(\rho) \in \Sigma^*$ is obtained by concatenating the u_i's; the *language* accepted by the system is then $L(S) \stackrel{\text{def}}{=} \bigcup_{\rho \in R} \mathsf{yield}(\rho)$. A language class *supports amalgamation* if, for every language L in the class, there exists an amalgamation system S such that $L(S) = L$.

3.1.1 Admissible Embeddings and Gaps. Furthermore, for any two runs ρ, σ, $E(\rho, \sigma)$ is a set of *admissible embeddings* between their canonical decompositions. Here, an embedding over the alphabet $(\Sigma_\varepsilon, =)$ is defined as in Section 2: if $\mathsf{can}(\rho) = u_1 \cdots u_n$ and $\mathsf{can}(\sigma) = v_1 \cdots v_m$ are the canonical decompositions of the two runs, then an embedding of ρ in σ is a strictly monotone map $f\colon [1, n] \to [1, m]$ with $v_{f(i)} = u_i$ for every $i \in [1, n]$. For each embedding $f \in E(\rho, \sigma)$ and $i \in [0, n]$, we define the *gap word* $\mathrm{G}_{i,f} \in \Sigma^*$, such that $v_1 \cdots v_m = \mathrm{G}_{0,f} u_1 \mathrm{G}_{1,f} \cdots u_n \mathrm{G}_{n,f}$. Formally,

$$
\begin{aligned}
\mathrm{G}_{0,f} &\stackrel{\text{def}}{=} v_1 \cdots v_{f(1)-1}\,, \\
\mathrm{G}_{i,f} &\stackrel{\text{def}}{=} v_{f(i)+1} \cdots v_{f(i+1)-1} \qquad \text{for all } i \in [1, n-1], \\
\mathrm{G}_{n,f} &\stackrel{\text{def}}{=} v_{f(n)+1} \cdots v_m\,.
\end{aligned}
$$

If the set $E(\rho, \sigma)$ is non-empty, we write $\rho \trianglelefteq \sigma$; if we wish to refer to a specific $f \in E(\rho, \sigma)$, we write $\rho \trianglelefteq_f \sigma$. Note that $\rho \trianglelefteq \sigma$ implies $\mathsf{can}(\rho) \leq_* \mathsf{can}(\sigma)$ but the converse might not hold: $E(\sigma, \rho)$ need not contain all the possible embeddings between the canonical decompositions of ρ and σ.

3.1.2 Conditions. Finally, we require the following:

composition if $f \in E(\rho, \sigma)$ and $g \in E(\sigma, \tau)$, then $f \circ g \in E(\rho, \tau)$,
wqo $(R, \trianglelefteq)$ is a well-quasi-order, and
(concatenative) amalgamation if $\rho_0 \trianglelefteq_f \rho_1$ and $\rho_0 \trianglelefteq_g \rho_2$ with canonical decomposition $\mathsf{can}(\rho_0) = w_1 \cdots w_n$, then for every choice of $i \in [0, n]$, there exists a run $\rho_3 \in R$ such that $\rho_1 \trianglelefteq_{f'} \rho_3$, $\rho_2 \trianglelefteq_{g'} \rho_3$ with $f' \circ f = g' \circ g$ (we write h

for this composition) and

$$\mathrm{G}_{j,h} \in \{\mathrm{G}_{j,f}\mathrm{G}_{j,g},\ \mathrm{G}_{j,g}\mathrm{G}_{j,f}\} \qquad \text{for every } j \in [0,n], \text{ and}$$
$$\mathrm{G}_{i,h} = \mathrm{G}_{i,f}\mathrm{G}_{i,g} \qquad \text{for the chosen } i.$$

Thus the embedding h of ρ_0 into ρ_3 has the property that each gap word is the concatenation of the two gap words from the embeddings into ρ_1 and ρ_2, in some order. Moreover, in the particular gap i, we know that the gap word from ρ_1 comes first, and then the gap word from ρ_2. This tell us that, given a gap $i \in [1,n]$, we can choose a run ρ_3 such that the concatenation in gap i happens in an order of our choice.

3.1.3 Gap Languages. Given an amalgamation system and some run $\rho \in R$ we also define the *gap language* for a gap $i \in [0, |\rho|_{\mathsf{can}}]$

$$L_{\rho,i} \stackrel{\text{def}}{=} \{\mathrm{G}_{i,f} \mid \exists \sigma \in R\colon \rho \trianglelefteq_f \sigma\}\ .$$

In other words, $L_{\rho,i}$ is the set of all words that can be inserted in the i-th gap of ρ's canonical decomposition when ρ embeds into some larger run. A language $L \subseteq \Sigma^*$ is a *subsemigroup* if for any two words $u, v \in L$, we have $uv \in L$. The following is a direct consequence of concatenative amalgamation.

Observation 3.1. For each run ρ and i, the language $L_{\rho,i}$ is a subsemigroup.

3.1.4 Effectiveness. In order to derive the algorithmic results of Section 4, we need to make some effectiveness assumptions. These are always clear from our constructions and will be used tacitly in the algorithms. Specifically, we assume that each run in R has some finite representation such that

(i) the set R is recursively enumerable,
(ii) the function $\mathsf{can}(\cdot)$ is computable, and
(iii) for any two runs ρ, σ we can compute the set $E(\rho,\sigma)$ of admissible embeddings (and hence decide whether $\rho \trianglelefteq \sigma$ and compute the various gap words).

3.2 Example: Regular Languages

Let us start with a very simple example, namely a non-deterministic finite automaton (NFA). Suppose we have an NFA $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ with finite state set Q, input alphabet Σ, transition set $\Delta \subseteq Q \times \Sigma_\varepsilon \times Q$, initial states $I \subseteq Q$, and final states $F \subseteq Q$. Then we can view it as an amalgamation system by taking $R \subseteq \Delta^*$ to be the set of all finite sequences $(q_0, a_1, q_1)(q_1, a_2, q_2) \cdots (q_{n-1}, a_n, q_n) \in \Delta^*$ starting in some $q_0 \in I$ and ending in some $q_n \in F$, and where the end state of each individual element of the sequence is the same as first state of the next element. Such a transition sequence can therefore actually be executed by the automaton. The canonical decomposition of such a run is simply $a_1 \cdots a_n$, and $L(\mathcal{A}) = \bigcup_{\rho \in R} \mathsf{yield}(\rho)$ as desired.

For two runs $\rho = r_1 \cdots r_n \in R$ and $\sigma = s_1 \cdots s_m \in R$, we then let $E(\rho,\sigma)$ be the set of strictly monotone maps $f\colon [1,n] \longrightarrow [1,m]$ such that $s_{f(i)} = r_i$ for every $i \in [1,n]$. Thus $(R, \trianglelefteq)$ is the qo induced by $R \subseteq \Delta^*$ inside the word embedding qo $(\Delta^*, \leq_*)$ over the alphabet $(\Delta, =)$, and the composition and wqo conditions follow.

Concatenative amalgamation also holds. Let ρ_0, ρ_1, ρ_2 be runs with $\rho_0 \trianglelefteq_f \rho_1$ and $\rho_0 \trianglelefteq_g \rho_2$, where the canonical decomposition of $\mathsf{can}(\rho_0)$ has n transitions. Because the individual transitions of ρ_0 must be compatible with each other, we know that the i-th transition ends in the same state q_i that the $(i+1)$-st transition begins in. However, that means that the gap words $\mathrm{G}_{i,f}$ and $\mathrm{G}_{i,g}$ must be read on loops from q_i to q_i. Therefore, we can concatenate them in any order we wish to obtain a new run ρ_3 as required.

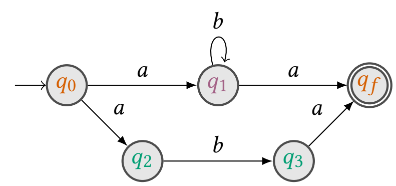


Figure 1: A finite automaton for Example 3.2.

Example 3.2. As a concrete example, consider the automaton in Fig. 1. A possible run is $\rho_0 \stackrel{\text{def}}{=} (q_0, a, q_1)(q_1, a, q_f)$. Another run is $\rho \stackrel{\text{def}}{=} (q_0, a, q_2)(q_2, b, q_3)(q_3, a, q_f)$. Although $\mathsf{can}(\rho_0) = aa \leq_* aba = \mathsf{can}(\rho)$, there is no a run embedding between the two: the transition (q_0, a, q_1) of ρ_0 cannot be mapped to a corresponding transition in ρ.

If we consider the run $\rho_1 \stackrel{\text{def}}{=} (q_0, a, q_1)(q_1, b, q_1)(q_1, a, q_f)$, then we can embed ρ_0 into ρ_1 via the map $\{1 \mapsto 1, 2 \mapsto 3\}$. Indeed, we get a decomposition of the yield of ρ_1 into $\mathrm{G}_{0,f}a\mathrm{G}_{1,f}a\mathrm{G}_{2,f}$ with $\mathrm{G}_{0,f} = \mathrm{G}_{2,f} = \varepsilon$ and $\mathrm{G}_{1,f} = b$. And clearly we can take the b-labelled loop several times, yielding for instance the run $\rho_2 \stackrel{\text{def}}{=} (q_0, a, q_1)(q_1, b, q_1)(q_1, b, q_1)(q_1, a, q_f)$.

3.3 Example: Vector Addition Systems

A d-dimensional (labelled) *vector addition system with states (VASS)* is a finite automaton $\mathcal{V} = (Q, \Sigma, \Delta, I, F)$ with transition labels in $\Sigma_\varepsilon \times \mathbb{Z}^d$: Δ is now a finite subset of $Q \times \Sigma_\varepsilon \times \mathbb{Z}^d \times Q$. We revisit here the results of [49] to show that VASS are amalgamation systems.

3.3.1 Configurations and Semantics. Let $\mathit{Confs} \stackrel{\text{def}}{=} Q \times \mathbb{N}^d$. A *configuration* is a pair $q(\mathbf{u}) \in \mathit{Confs}$ of a state and the values of the d counters. Configurations are ordered through the product ordering: $q(\mathbf{u}) \leq q'(\mathbf{u}')$ if $q = q'$ and $\mathbf{u} \leq \mathbf{u}'$. If $c = q(\mathbf{u})$ is a configuration and $\mathbf{d} \in \mathbb{N}^d$, we write $c + \mathbf{d}$ for the configuration $q(\mathbf{u} + \mathbf{d})$; note that $c \leq c'$ if and only if there exists $\mathbf{d} \in \mathbb{N}^d$ such that $c' = c + \mathbf{d}$.

The counters can be incremented and decremented along transitions, but not tested for zero: for configurations $c, c' \in \mathit{Confs}$ and a transition $t \in \Delta$, we write $c \xrightarrow{t} c'$ if $t = (q, x, \mathbf{v}, q')$, $c = q(\mathbf{c})$, and $c' = q'(\mathbf{c} + \mathbf{v})$, and extend this notation to sequences of transitions: $c_0 \xrightarrow{t_1 \cdots t_n} c_n$ if there exist $c_1, \ldots, c_{n-1}$ such that $c_{i-1} \xrightarrow{t_i} c_i$ for all $i \in [1,n]$. VASS transitions are *monotonic*, in that if $c \xrightarrow{t} c'$ and $\mathbf{d} \in \mathbb{N}^d$, then $c + \mathbf{d} \xrightarrow{t} c' + \mathbf{d}$.

3.3.2 Runs and Admissible Embeddings. A *run* is a sequence $\rho = (c_0, t_1, c_1) \cdots (c_{n-1}, t_n, c_n)$ such that $c_0 = q_0(\mathbf{0})$ for some $q_0 \in I$, $c_n = q_f(\mathbf{0})$ for some $q_f \in F$, and $c_{i-1} \xrightarrow{t_i} c_i$ at each step. Let t_i be $(q_{i-1}, a_i, \mathbf{v}_i, q_i)$ for each i in this run; the associated canonical decomposition is $\mathsf{can}(\rho) \stackrel{\text{def}}{=} a_1 \cdots a_n$ and we recover the usual notion of a VASS (reachability) language: $L(\mathcal{V}) = \bigcup_{\rho \in R} \mathsf{yield}(\rho)$.

Let $(R, \trianglelefteq)$ be the qo induced by $R \subseteq (\mathit{Confs} \times \Delta \times \mathit{Confs})^*$ inside the word embedding qo $((\mathit{Confs} \times \Delta \times \mathit{Confs})^*, \leq_*)$ over the product alphabet $(\mathit{Confs}, \leq) \times (\Delta, =) \times (\mathit{Confs}, \leq)$. Thus for two runs $\rho_0 = (c_0, t_1, c_1) \cdots (c_{n-1}, t_n, c_n)$ and $\rho_1 = (c'_0, t'_1, c'_1) \cdots (c'_{m-1}, t'_m, c'_m)$, we have $\rho_0 \trianglelefteq \rho_1$ if there exists a strictly monotone map $f : [1, n] \to [1, m]$ such that, for all $i \in [1, n]$, $c_{i-1} \leq c'_{f(i)-1}$, $t_i = t'_{f(i)}$, and $c_i \leq c'_{f(i)}$. By definition, the composition and wqo conditions follow. Also, because ρ_1 is a run, there exist $\mathbf{d}_1, \dots, \mathbf{d}_n \in \mathbb{N}^d$ such that for all $i \in [1, n]$,

$$c'_{f(i)-1} = c_{i-1} + \mathbf{d}_i \xrightarrow{t_i} c_i + \mathbf{d}_i = c'_{f(i)} \ . \tag{1}$$

Furthermore, letting $\mathbf{d}_0 \stackrel{\text{def}}{=} \mathbf{0}$, $\mathbf{d}_{n+1} \stackrel{\text{def}}{=} \mathbf{0}$, $f(0) \stackrel{\text{def}}{=} 0$, and $f(n+1) \stackrel{\text{def}}{=} m+1$, then each $\mathrm{G}_{i,f}$ for $i \in [0, n]$ is the yield of the transition sequence

$$c_i + \mathbf{d}_i = c'_{f(i)} \xrightarrow{t'_{f(i)+1} \cdots t'_{f(i+1)-1}} c'_{f(i+1)-1} = c_i + \mathbf{d}_{i+1} \ . \tag{2}$$

3.3.3 Concatenative Amalgamation. Assume $\rho_0 \trianglelefteq_f \rho_1$ as above, and $\rho_0 \trianglelefteq_g \rho_2 = (c''_0, t''_1, c''_1) \cdots (c''_{p-1}, t''_p, c''_p)$. Then there exist $\mathbf{d}_1, \dots, \mathbf{d}_n$ (resp. $\mathbf{d}'_1, \dots, \mathbf{d}'_n$) such that we can decompose ρ_1 (resp. ρ_2) as in (1) and (2). We can amalgamate to a third run ρ_3 by observing that, by monotonicity, $c_i + \mathbf{d}_i + \mathbf{d}'_i = c'_{f(i)} + \mathbf{d}'_i \xrightarrow{t'_{f(i)+1} \cdots t'_{f(i+1)-1}} c'_{f(i+1)-1} + \mathbf{d}'_i = c_i + \mathbf{d}_{i+1} + \mathbf{d}'_i = c''_{g(i)} + \mathbf{d}_{i+1} \xrightarrow{t''_{g(i)+1} \cdots t''_{g(i+1)-1}} c''_{g(i+1)-1} + \mathbf{d}_{i+1} = c_i + \mathbf{d}_{i+1} + \mathbf{d}'_{i+1}$ yields $\mathrm{G}_{i,f}\mathrm{G}_{i,g}$ for all $i \in [0, n]$.

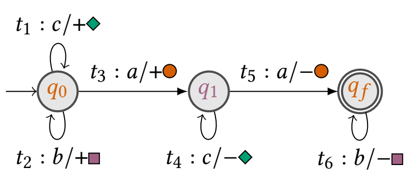


Figure 2: A VASS for Example 3.3.

Example 3.3. As a concrete example, consider the VASS in Fig. 2. The three runs we will consider are

$\rho_0 \stackrel{\text{def}}{=} (q_0(), t_3, q_1(\bullet))\ (q_1(\bullet), t_5, q_f())$

$\rho_1 \stackrel{\text{def}}{=} (q_0(), t_2, q_0(\blacksquare))\ (q_0(\blacksquare), t_3, q_1(\bullet\blacksquare))\ (q_1(\bullet\blacksquare), t_5, q_f(\blacksquare))\ (q_f(\blacksquare), t_6, q_f())$

$\rho_2 \stackrel{\text{def}}{=} (q_0(), t_1, q_0(\blacklozenge))\ (q_0(\blacklozenge), t_3, q_1(\bullet\blacklozenge))\ (q_1(\bullet\blacklozenge), t_4, q_1(\bullet))\ (q_1(\bullet), t_5, q_f())$

We have $\rho_0 \trianglelefteq_f \rho_1$ and $\mathrm{G}_{0,f} = b, \mathrm{G}_{1,f} = \varepsilon$, and $\mathrm{G}_{2,f} = b$. Similarly, we have $\rho_0 \trianglelefteq_g \rho_2$ and $\mathrm{G}_{0,g} = c, \mathrm{G}_{1,g} = c$, and $\mathrm{G}_{2,g} = \varepsilon$. We can amalgamate to obtain $\rho_3 = (q_0(), t_2, q_0(\blacksquare))\ (q_0(\blacksquare), t_1, q_0(\blacksquare\blacklozenge))\ (q_0(\blacksquare\blacklozenge), t_3, q_1(\bullet\blacksquare\blacklozenge))\ (q_1(\bullet\blacksquare\blacklozenge), t_4, q_1(\bullet\blacksquare))\ (q_1(\bullet\blacksquare), t_5, q_f(\blacksquare))\ (q_f(\blacksquare), t_6, q_f())$. One can check that $\rho_0 \trianglelefteq_h \rho_3$ and $\mathrm{G}_{i,h} = \mathrm{G}_{i,f}\mathrm{G}_{i,g}$ for all $i \in [0, 2]$.

VASS can be construed as "adding counters" to finite automata; see Section 5.2 for a general construction showing that this can be achieved more generally in amalgamation systems.

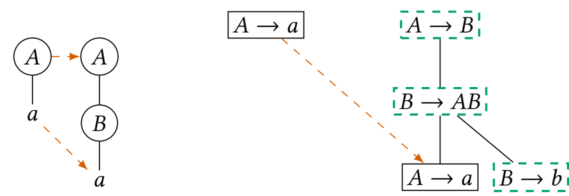
3.4 Example: Context-Free Languages

A *context-free grammar* (CFG) is a tuple $\mathcal{G} = (N, \Sigma, S, \Delta)$ with finite non-terminal alphabet N, finite terminal alphabet Σ, initial non-terminal $S \in N$, and finite set of productions $\Delta \subseteq N \times (N \cup \Sigma)^*$; each such production is written $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^*$.

We mostly consider grammars from the perspective of their derivation trees. In a slight divergence from the usual definition (but consistent with e.g., [48, App. A]), we label nodes in derivation trees not with symbols from $N \cup \Sigma_\varepsilon$, but with productions from Δ. Indeed, the usual homeomorphic tree embedding (c.f. Section 2) over trees labelled by $(N \cup \Sigma_\varepsilon, =)$ is a well-quasi-ordering, but this labelling is not sufficient for amalgamation and we shall rather rely on the homeomorphic tree embedding over trees labelled by $(\Delta, =)$.

Example 3.4. Consider the grammar with $N \stackrel{\text{def}}{=} \{A, B\}$, $\Sigma \stackrel{\text{def}}{=} \{a, b\}$, and $\Delta \stackrel{\text{def}}{=} \{A \to a, A \to B, B \to a, B \to b, B \to AB\}$ and the two trees in Fig. 3a. The left tree homeomorphically embeds into the right one, but no larger trees can be derived from this ordering, making it unsuitable for amalgamation.

Let us assume wlog. that the productions in Δ are either non-terminal productions of the form $A \to B_1 \cdots B_k$ with $k > 0$ and $B_1, \dots, B_k \in N$, or terminal productions of the form $A \to a$ with $a \in \Sigma_\varepsilon$. We call a tree in $T(\Delta)$ *A-rooted* if its root label is a production $A \to \alpha$ for some α. Using the notations from Section 2, a *derivation tree* is either a leaf $(A \to a)[]$ labelled by a terminal production $(A \to a) \in \Delta$, or a tree $(A \to B_1 \cdots B_k)[t_1, \dots, t_k]$ where $(A \to B_1 \cdots B_k) \in \Delta$ is a non-terminal production and for all $i \in [1, k]$, t_i is a B_i-rooted derivation tree.



(a) The left tree embeds into the right tree, but amalgamation is not possible. **(b) The embedding between the left and the right trees can be used to create successively larger trees.**

Figure 3: Derivation tree embeddings for Example 3.4.

Observe that, if two A-rooted derivation trees t and t' homeomorphically embed into each other, i.e., $t \leq_T t'$, then t has a production $A \to \alpha$ as root label, and there exists a subtree $t'/p = (A \to \alpha)[t'_1, \dots, t'_k]$ for some k. Then, putting a hole at position p in t' gives rise to an *A-context*, i.e., a context where plugging any A-rooted derivation tree will produce an A-rooted derivation tree; this context can thus be iterated at will. Furthermore, if $A \to \alpha$ is a non-terminal production, then $\alpha = B_1 \cdots B_k$, $t = (A \to B_1 \cdots B_k)[t_1, \dots, t_k]$, and $t'/p = (A \to B_1 \cdots B_k)[t'_1, \dots, t'_k]$, and inductively for each $i \in [1, k]$ t_i and t'_i are two B_i-rooted derivation trees with $t_i \leq_T t'_i$, allowing to repeat the same reasoning.

Example 3.4 (continued). In Fig. 3b the nodes in dashed boxes in the right tree form an A-context corresponding to the derivation $A \Rightarrow B \Rightarrow AB \Rightarrow Ab$. This A-context can be iterated (and thus the derivation as well) to derive larger and larger trees before plugging the node $A \to a$ from the image of the left tree.

Finally, the canonical decomposition of a derivation tree is defined inductively by $\mathsf{can}((A \to a)[]) \stackrel{\text{def}}{=} \varepsilon a \varepsilon$ for $a \in \Sigma_\varepsilon$ and $\mathsf{can}((A \to \alpha)[t_1, \dots, t_n]) \stackrel{\text{def}}{=} \varepsilon \mathsf{can}(t_1) \cdots \mathsf{can}(t_n) \varepsilon$ otherwise. A *run* is then an S-rooted derivation tree, and this matches the usual definition of the language of a context-free grammar: $L(\mathcal{G}) = \bigcup_{\rho \in R} \mathsf{yield}(\rho)$. A comment is in order for those explicit ε's. This can be seen as a transformation of the grammar into its associated parenthetical grammar (followed by an erasure of the parentheses), so that the extra ε reflect the tree structure. In turn, this serves to break up gap words that would otherwise span several levels of the derivation trees, and forces them to reflect how the trees embed. We can then interleave these smaller gap words as may be required for the concatenative amalgamation of trees. We will generalise this whole construction in Section 5.3, where we consider algebraic extensions of amalgamation systems.

4 ALGORITHMS FOR AMALGAMATION SYSTEMS

In this section, we prove that amalgamation systems have all the algorithmic properties in Main Theorem A. We work with amalgamation systems satisfying the implicit effectiveness assumptions of Section 3.1.4, and language classes that are effectively closed under rational transductions (and thus under morphisms and intersection with regular languages)—also known as *full trios* [see, e.g., 7].

4.1 The Simultaneous Unboundedness Problem

We now prove the first main result of this work, about the *simultaneous unboundedness problem* (SUP) for formal languages.

> **Given** An alphabet $\{a_1, \ldots, a_n\}$ and a language $L \subseteq a_1^* \cdots a_n^*$.
> **Question** Is it true that for every $k \in \mathbb{N}$, there exist $x_1, \ldots, x_n \in \mathbb{N}$ such that $x_1, \ldots, x_n \geq k$ and $a_1^{x_1} \cdots a_n^{x_n} \in L$?

There has been some interest in this problem because in [63, Thm. 1], it was shown that for any full trio C, downward closures are computable if and only if the SUP is decidable for C, thus under the hypotheses of Main Theorem A, "(1) $\Leftrightarrow$ (2)". Moreover, in [17, Thm. 2.6], it was shown that for any full trio C, separability by piecewise testable languages is decidable if and only if the SUP is decidable, thus "(1) $\Leftrightarrow$ (3)". In fact, analogous results hold also for some orderings beyond the subword ordering [66]. Given this motivation, the SUP is known to be decidable for VASS [19, 33], higher-order pushdown automata [36], and even higher-order recursion schemes [5, 14, 55].

Proof of "(7) $\Rightarrow$ (1)". Let us first define the *Parikh image* $\Psi(w)$ of a word $w \in \{a_1, \ldots, a_n\}^*$ as the vector in $\mathbb{N}^n$ where $\Psi(w)(i)$ is the number of occurrences of a_i inside w. We also write $\Psi(\rho)$ as shorthand for $\Psi(\mathsf{yield}(\rho))$. For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{N}^n$, we write $\mathbf{u} \ll \mathbf{v}$ if, for all $i \in [1, n]$, $\mathbf{u}(i) < \mathbf{v}(i)$.

Our algorithm consists of two semi-decision procedures. The first enumerates $k \in \mathbb{N}$ and then checks whether for some $i \in [1, n]$, we have $L \subseteq a_1^* \cdots a_{i-1}^* a_i^{\leq k} a_{i+1}^* \cdots a_n^*$, which can be decided in a full trio with an oracle for the emptiness problem by checking whether $L \cap (\Sigma^* \setminus a_1^* \cdots a_{i-1}^* a_i^{\leq k} a_{i+1}^* \cdots a_n^*) = \emptyset$. The other one enumerates pairs of runs ρ, σ and checks whether (i) $\rho \trianglelefteq \sigma$ and (ii) $\Psi(\rho) \ll \Psi(\sigma)$. Clearly, if the first semi-decision procedure terminates, then our system cannot be a positive instance of the SUP, because in any word in L, there are at most k occurrences of a_i. Moreover, if the second semi-decision procedure finds ρ and σ as above, then by amalgamation, we obtain runs $\sigma_1, \sigma_2, \ldots$ such that $\Psi(\sigma_k)(i) \geq k$, meaning we have a positive instance.

It remains to argue that one of the two procedures will terminate. This is trivial if our system is a negative instance. Conversely, if our system is a positive instance, then there is an infinite sequence of runs $\rho_1, \rho_2, \ldots$ such that $\Psi(\rho_k)(i) \geq k$ for every $i \in [1, n]$. This sequence has an infinite subsequence $\rho'_1, \rho'_2, \ldots$ such that $\Psi(\rho'_{k+1})(i) > \Psi(\rho'_k)(i)$ for every $i \in [1, n]$. Since $(R, \trianglelefteq)$ is a wqo, we can find $j < \ell$ such that $\rho'_j \trianglelefteq \rho'_\ell$. By definition of the ρ'_k, this pair satisfies $\Psi(\rho'_j) \ll \Psi(\rho'_\ell)$. ◻

Proof of "(1) $\Rightarrow$ (7)". Assuming decidable SUP, for any $L \subseteq \Sigma^*$, take the rational transduction $T \stackrel{\text{def}}{=} \Sigma^* \times \{a_1\}^*$ and observe that $TL \subseteq a_1^*$ is a positive instance of the SUP if and only if $L \neq \emptyset$. ◻

4.2 Language Boundedness

Our next main result is about deciding language boundedness.

> **Given** A language $L \subseteq \Sigma^*$.
> **Question** Does there exist $k \in \mathbb{N}$ and words $w_1, \ldots, w_k \in \Sigma^*$ such that $L \subseteq w_1^* \cdots w_k^*$?

The decidability of language boundedness was known for pushdown automata since the 1960's [e.g., 30]—and is even in polynomial time [28]. The question was open for many years in the case of reversal-bounded counter machines (RBCM) [10, 21] before it was settled for VASS in [19]. For RBCM with a pushdown, it was settled even more recently [6]. The proof here is substantially simpler. Our proof also easily yields that all amalgamation systems enjoy a *growth dichotomy*: their languages either have polynomial growth (if they're bounded) or otherwise exponential growth (see [28] for a precise definition). After being open for RBCM for a long time [42], this was shown in [6] for RBCM and for pushdown RBCM.

We begin with a simple characterisation in the case of subsemigroups.

Lemma 4.1. *Let $L \subseteq \Sigma^*$ be a subsemigroup. Then exactly one of the following holds: (i) L is bounded or (ii) L contains two prefix-incomparable words.*

Proof. First, suppose that (i) and (ii) both hold. Then $L \subseteq w_1^* \cdots w_k^*$, which implies that for every $n \in \mathbb{N}$, L contains at most n^{k-1} words of length n. On the other hand, if $u, v \in L$ are prefix-incomparable, meaning that neither is a prefix of the other, then the two words uv and vu have equal length, but are distinct. Moreover, as L is a subsemigroup, we have $\{uv, vu\}^n \subseteq L$ for every n. But $\{uv, vu\}^n$ contain 2^n distinct words of length $|uv| \cdot n$. Yet $2^n \leq (|uv| \cdot n)^{k-1}$ cannot hold for every $n \in \mathbb{N}$. Hence, (i) and (ii) are mutually exclusive. Now suppose (ii) does not hold. If $L = \emptyset$ or $L = \{\varepsilon\}$, then L is bounded. Otherwise, let $w \in L$ be a word of minimal length; since $L \neq \{\varepsilon\}$ and (ii) does not hold, $w \neq \varepsilon$. As L is a subsemigroup, $w^n \in L$ for all $n > 0$, and since any two words in L are prefix-comparable, we have $L \subseteq \mathsf{prefixes}(w^*)$. Since w^* is bounded, $\mathsf{prefixes}(w^*)$ and thus L are bounded as well. ◻

Proof of "(7) $\Rightarrow$ (4)". We again provide two semi-decision procedures. The procedure for positive instances simply enumerates expressions $w_1^* \cdots w_k^*$ and checks whether $L \subseteq w_1^* \cdots w_k^*$, which is decidable for a full trio with an oracle for the emptiness problem. The more interesting case is the procedure for negative instances, which looks for the following *non-boundedness witness*: three runs

ρ_0, ρ_1, and ρ_2 with $\rho_0 \trianglelefteq_f \rho_1$ and $\rho_0 \trianglelefteq_g \rho_2$ such that for some i, the words $\mathrm{G}_{i,f}$ and $\mathrm{G}_{i,g}$ are prefix-incomparable. Let us show that non-boundedness witnesses characterise negative instances.

First, suppose there is a non-boundedness witness. Then the gap language $L_{\rho_0,i}$ contains two prefix-incomparable words. Moreover, by Observation 3.1, $L_{\rho_0,i}$ is a subsemigroup, and thus by Lemma 4.1, the language $L_{\rho_0,i}$ is not bounded. However, every word in $L_{\rho_0,i}$ appears as a factor in a word of L. Then L is not bounded, as otherwise the set of factors of L would be bounded and thus also $L_{\rho_0,i}$.

Conversely, suppose there is no non-boundedness witness. As $(R, \trianglelefteq)$ is a wqo, R itself has a finite basis $\{\rho_1, \ldots, \rho_m\}$. Let $\mathrm{can}(\rho_i) = a_{i,1} \cdots a_{i,n_i}$ be the canonical decomposition of ρ_i. Then we have

$$L \subseteq \bigcup_{i=1}^{m} L_{\rho_i,0} a_{i,1} L_{\rho_i,1} \cdots a_{i,n_i} L_{\rho_i,n_i} \; . \tag{3}$$

Since there is no non-boundedness witness, each language $L_{\rho_i,j}$ is linearly ordered by the prefix ordering. Moreover, by Observation 3.1 each language $L_{\rho_i,j}$ is a subsemigroup and thus Lemma 4.1 implies that $L_{\rho_i,j}$ is bounded. As boundedness is preserved by finite products, finite unions, and taking subsets, L must be bounded. ◻

Proof of "(4) ⟹ (7)". Given a language $L \subseteq \Sigma^*$, consider the rational transduction $T \stackrel{\text{def}}{=} \Sigma^* \times \{a, b\}^*$. Then TL is a bounded language (actually, the empty set) if and only if $L = \emptyset$ (as otherwise, $TL = \{a, b\}^*$ is not bounded). ◻

4.3 Unary Languages

We now come to results on languages over single-letter alphabets $\Sigma = \{a\}$. To simplify the exposition, we slightly abuse notation and identify each word a^k with the number $k \in \mathbb{N}$ and thus assume that each $\mathrm{yield}(\rho)$ for a run ρ is a natural number.

First, we provide in Section 4.3.1 a very simple proof due to Leroux [47] that shows that all VASS languages over a single letter are regular. We then show how to make the proof effective in Section 4.3.2; the resulting proof is still markedly simpler that the one of effective regularity in VASS by Hauschildt and Jantzen [40], which relies on Hauschildt's dissertation [39].

4.3.1 Regularity. There is a proof due to Leroux which shows, only using amalgamation, that all unary VASS languages are regular [47], and happens to apply to our notion of amalgamation systems. It relies on a folklore result from number theory (see, e.g. [61]).

Lemma 4.2 (folklore). *If $S \subseteq \mathbb{N}$ is a subsemigroup of $\mathbb{N}$, then S is ultimately periodic. Moreover, S is ultimately identical to $\mathbb{N} \cdot \gcd(S)$.*

Since Leroux's proof has not been published, we reproduce it here, in a general form for amalgamation systems.

Lemma 4.3 ([47]). *Every unary language accepted by an amalgamation system is regular.*

Proof. Let $L \subseteq \mathbb{N}$ be the language of an amalgamation system $(\{a\}, R, E, \mathrm{can})$. We want to show that L is ultimately periodic. Since $(R, \trianglelefteq)$ is a wqo, the set R has a finite basis, say $\{\rho_1, \ldots, \rho_m\}$. For each ρ_i, consider the set

$$L_i \stackrel{\text{def}}{=} \{\mathrm{yield}(\sigma) - \mathrm{yield}(\rho_i) \mid \exists \sigma \in R \colon \rho_i \trianglelefteq \sigma\} \; . \tag{4}$$

Since every run of R embeds one of the runs $\rho_1, \ldots, \rho_m$, we have $L = (\mathrm{yield}(\rho_1) + L_1) \cup \cdots \cup (\mathrm{yield}(\rho_m) + L_m)$. By concatenative amalgamation, each L_i is a subsemigroup of $\mathbb{N}$. By Lemma 4.2, this implies that L_i is ultimately periodic, and thus so is L. ◻

4.3.2 Effectiveness. Unfortunately, Leroux's proof in Lemma 4.3 is not effective: even for VASS, one cannot compute a basis of the set of runs (see Appendix A). Therefore, we prove an effective version, which works by enumeration. It enumerates certain combinations of runs that yield an ultimately periodic subset of numbers. Moreover, we will show that for every amalgamation system, there exists such a combination of runs that yields exactly its entire language.

Proof of "(7) ⟹ (5)". Define a *unary witness* as a pair (F, T), where $F \subseteq R$ is a finite set of runs and $T \subseteq R \times R \times R$ is a finite set of triples (ρ, σ, τ) of runs such that $\rho \trianglelefteq \sigma$ and $\rho \trianglelefteq \tau$. The set *represented by* (F, T) is defined as

$$S(F,T) \stackrel{\text{def}}{=} \{\mathrm{yield}(\rho) \mid \rho \in F\} \;\cup \bigcup_{(\rho,\sigma,\tau) \in T} S(\rho, \sigma, \tau) \; ,$$

where, for runs ρ, σ, τ with $\rho \trianglelefteq \sigma$ and $\rho \trianglelefteq \tau$,

$$\begin{aligned} S(\rho,\sigma,\tau) \stackrel{\text{def}}{=} \mathrm{yield}(\rho) \; &+ \; \mathbb{N} \cdot (\mathrm{yield}(\sigma) - \mathrm{yield}(\rho)) \\ &+ \; \mathbb{N} \cdot (\mathrm{yield}(\tau) - \mathrm{yield}(\rho)) \; . \end{aligned}$$

Our algorithm works as follows. It enumerates unary witnesses (F, T) and for each of them, checks whether $L \subseteq S(F, T)$. The latter is decidable because $S(F, T)$ is an effectively regular language and we can check if the set $L \cap (\mathbb{N} \setminus S(F, T))$ is empty in a full trio with an oracle for emptiness. Since we always have $S(F, T) \subseteq L$ by construction, this algorithm is correct: if it finds (F, T) with $L \subseteq S(F, T)$, then we know $L = S(F, T)$. It remains to show termination.

Claim 4.4. *There is a unary witness (F, T) with $L = S(F, T)$.*

To prove Claim 4.4, let $\{\rho_1, \ldots, \rho_m\}$ be a finite basis of the wqo $(R, \trianglelefteq)$ and define the sets L_i as in (4). Since each L_i is a semigroup, Lemma 4.2 tells us that L_i ultimately agrees with $\mathbb{N} \cdot \gcd(L_i)$. In particular, there are $k, \ell \in L_i$ with $k - \ell = \gcd(L_i)$. This means that there are runs σ_i and τ_i with $\rho_i \trianglelefteq \sigma_i$ and $\rho_i \trianglelefteq \tau_i$ with $\mathrm{yield}(\sigma_i) = \mathrm{yield}(\rho_i) + k$ and $\mathrm{yield}(\tau_i) = \mathrm{yield}(\rho_i) + \ell$. We choose $T \stackrel{\text{def}}{=} \{(\rho_i, \sigma_i, \tau_i) \mid i \in [1, m]\}$. We now claim that the set $L \setminus S(\emptyset, T)$ is finite. Note that if this is true, we are done, because we can choose F by picking a run for each number in $L \setminus S(\emptyset, T)$. For finiteness of $L \setminus S(\emptyset, T)$, it suffices to show finiteness of $L_i \setminus G_i$ for each i, where

$$G_i \stackrel{\text{def}}{=} \mathbb{N} \cdot (\mathrm{yield}(\sigma_i) - \mathrm{yield}(\rho_i)) + \mathbb{N} \cdot (\mathrm{yield}(\tau_i) - \mathrm{yield}(\rho_i)).$$

To show that $L_i \setminus G_i$ is finite, we claim that $\gcd(G_i)$ divides $\gcd(L_i)$. This will imply finiteness of $L_i \setminus G_i$ because G_i is a subsemigroup of $\mathbb{N}$ and thus ultimately agrees with $\mathbb{N} \cdot \gcd(G_i)$.

Since $\mathrm{yield}(\sigma_i) - \mathrm{yield}(\rho_i)$ and $\mathrm{yield}(\tau_i) - \mathrm{yield}(\rho_i)$ both belong to G_i, we know that $\gcd(G_i)$ divides both numbers, and therefore $\gcd(G_i)$ also divides their difference, which is $\mathrm{yield}(\sigma_i) - \mathrm{yield}(\tau_i) = \gcd(L_i)$ by the choice of σ_i and τ_i. The claim is established. ◻

Proof of "(5) ⟹ (7)". Given $L \subseteq \Sigma^*$, consider the rational transduction $T \stackrel{\text{def}}{=} \{(w, a^{|w|}) \mid w \in \Sigma^*\}$. Then $TL \subseteq \{a\}^*$ is a unary language. Moreover, $TL = \emptyset$ if and only if $L = \emptyset$. Thus, we can decide emptiness of L using an NFA for TL constructed by the oracle for effectively regular unary languages accepted by amalgamation systems. ◻

4.4 Computing Priority Downward Closures

Motivated by the verification of systems that communicate via channels with congestion control, Anand and Zetzsche [2] consider the problem of computing downward closures with respect to the *priority ordering*, which was introduced in [32]. In that setting, one has an alphabet Σ with associated priorities in $[0,d]$, specified by a priority map $p\colon \Sigma \to [0,d]$. Then $u \preccurlyeq_{\mathrm{P}} v$ holds if $u = u_1 \cdots u_n$, $u_1,\ldots,u_n \in \Sigma$, and $v = v_1u_1 \cdots v_nu_n$, $v_1,\ldots,v_n \in \Sigma^*$, such that for each i, the letters in v_i have priority at most $p(u_i)$. In other words, letters can only be dropped from v if they are followed by some (undropped) letter of higher or equal priority. In a channel with congestion control, sending message sequences from a set $L \subseteq \Sigma^*$ will result in received messages in the *priority downward closure*

$$L{\downarrow_{\mathrm{P}}} = \{u \in \Sigma^* \mid \exists v \in L\colon u \preccurlyeq_{\mathrm{P}} v\}\ . \tag{5}$$

As with the ordinary word embedding, because $\preccurlyeq_{\mathrm{P}}$ well-quasi-orders the set of words with priorities [32, Lem. 3.2], the language $L{\downarrow_{\mathrm{P}}}$ is regular for any language $L \subseteq \Sigma^*$, hence the problem of *computing priority downward closures*, i.e., computing an NFA for $L{\downarrow_{\mathrm{P}}}$ for an input language L.

The proof of "(6) $\Rightarrow$ (7)" consists in observing that $L = \emptyset$ if and only if $L{\downarrow_{\mathrm{P}}} = \emptyset$, the latter being straightforward with an NFA recognising $L{\downarrow_{\mathrm{P}}}$. Here, we only want to sketch the proof of "(7) $\Rightarrow$ (6)", and point to Appendix B for the actual proof.

To compute the priority downward closure of a language $L \subseteq \Sigma^*$, the algorithm uses a strategy from [63]. It essentially enumerates $\preccurlyeq_{\mathrm{P}}$-downward closed sets D, decompose them into finitely many ideals as $D = I_1 \cup \cdots \cup I_n$, and then decides (i) whether L is included in D and (ii) whether each I_i is included in $L{\downarrow_{\mathrm{P}}}$. Here, in a wqo $(X,\leq)$, an *ideal* is a non-empty subset $I \subseteq X$ that is downwards closed and *upwards directed*, meaning that for any $x, y \in I$, there is a $z \in I$ with $x \leq z$ and $y \leq z$. It is a general property of wqos that every downwards closed set decomposes into finitely many ideals [e.g., 31, 49]. In this algorithm, deciding (i) is easy, because D is already a regular language, and since we assume decidable emptiness and closure under rational transductions, we can decide the emptiness of $L \cap (\Sigma^* \setminus D)$.

The challenging part is deciding (ii). For ordinary downward closures, this problem reduces to the SUP [63]. For priority downward closures, this also leads to an unboundedness problem, but a more intricate one. Instead of some measures (in the SUP: the number of occurrences of each letter) to be unbounded simultaneously, we here also need to decide *nested unboundedness*. However, using amalgamation, it will be possible to detect such nested unboundedness properties using certain "run constellations."

4.4.1 Nested Unboundedness. Let us illustrate this with an example. A particular unboundedness property that is required for ideal inclusion is whether in a language $L \subseteq \{0,1\}^*$, for every $k \geq 0$, there is a word $w \in L$ with $\geq k$ factors, each containing $\geq k$ contiguous 0's, and they are separated by 1's. In other words, we need to find arbitrarily many arbitrarily long blocks of 0's. Let us call this property *nested unboundedness*. Intuitively, this is more complicated that the SUP.

However, using run amalgamation, this amounts to checking a simple kind of witness. First, we need to slightly transform our language. Consider the language

$$\begin{aligned} K = \{u_1 \cdots u_n \mid\ & n \in \mathbb{N},\ v_0u_1v_1 \cdots u_nv_n \in L,\ \forall i \in [0,n]\colon \\ & \text{either (a) } v_i \in 0^*, \text{ (b) } v_i \in (10^*)^* \text{ and } u_{i+1} \in (10^*)^+, \text{ or} \\ & \text{(c) } i = n \text{ and } v_i \in (10^*)^+\} \end{aligned}$$

Hence, words in K are obtained from words in L by removing (a) factors in 0^* or (b) factors from $(10^*)^*$, but the latter only if that factor was followed by a non-removed 1, or is a suffix in $(10^*)^+$. This means that we can either make individual blocks of 0's smaller, or remove maximal factors of 0's, including exactly one neighbouring 1. One can see that K can be obtained from L using a rational transduction and thus we can construct an amalgamation system for K. Moreover, K has the nested unboundedness property if and only if L does. However, the advantage of working with K is that if nested unboundedness holds, then K contains every word in $(10^*)^*$.

4.4.2 Witnesses for Nested Unboundedness. In an amalgamation system for K, we have a simple kind of witness for our property: runs $\rho_0 \trianglelefteq \rho_1 \trianglelefteq \rho_2$ such that (i) some gap of $\rho_0 \trianglelefteq \rho_1$ between some positions $i < j$ of ρ_1 contains a 1 and (ii) some gap of $\rho_1 \trianglelefteq \rho_2$, which is between i and j in ρ_1, belongs to 0^+. Then, by amalgamating ρ_2 with itself above ρ_1 again and again, we can create arbitrarily long blocks of contiguous 0's. The resulting run ρ_2' still embeds ρ_1 and thus ρ_0, such that one gap of ρ_2' in ρ_0 contains both our long block of 0's and also a 1. Thus, amalgamating ρ_2' again and again above ρ_0, we obtain arbitrarily many long blocks of 0's.

4.4.3 Existence of Witnesses. Of course, we need to show that if nested unboundedness is satisfied, then the witness exists. Let S be an amalgamation system for K and suppose K has nested unboundedness. Let M be the maximal length of the canonical decompositions of runs in a finite basis of S. Consider the sequence $w_1, w_2, \ldots$ with $w_i = (0^i1)^{2M}$ for $i \geq 1$. Then $w_1, w_2, \ldots \in K$, so there must be runs ρ_1 and ρ_2 with $\rho_1 \trianglelefteq \rho_2$ such that $\mathsf{yield}(\rho_1) = w_i$ and $\mathsf{yield}(\rho_2) = w_j$. Then every non-empty gap of ρ_2 in ρ_1 belongs to 0^+. Moreover, any embedding of a minimal run ρ_0 of S into ρ_1 will have some gap containing two 1's, and thus have 10^i1 as a factor. Thus the runs ρ_0, ρ_1, ρ_2 constitute a witness.

Other Concepts. The full proof in Appendix B involves several steps. First, to simplify the exposition, we work with a slightly different ordering, called the *simple block ordering* and denoted $\preccurlyeq_{\mathrm{S}}$, such that downward closure computability of $L{\downarrow_{\mathrm{S}}}$ with respect to $\preccurlyeq_{\mathrm{S}}$ implies that of $L{\downarrow_{\mathrm{P}}}$ with respect to $\preccurlyeq_{\mathrm{P}}$. We then provide a syntactic description of the ideals of $\preccurlyeq_{\mathrm{S}}$. To avoid some technicalities, we introduce the related notion of *pseudo-ideals* and devise our algorithm to work with those. We define a notion of I-witness for each pseudo-ideal I, which is a particular constellation of run embeddings in a system for $L{\downarrow_{\mathrm{S}}}$ such that we have $I \subseteq L{\downarrow_{\mathrm{S}}}$ if and only if an amalgamation system for $L{\downarrow_{\mathrm{S}}}$ possesses an I-witness. There, we use that $L{\downarrow_{\mathrm{S}}}$ is obtained using a rational transduction from L.

4.5 Factor Universality

Finally, we want to mention that while amalgamation allows us to perform many algorithmic tasks (and applies to a significant class of systems—see the next section), it is not quite enough to cover the axiomatically defined class of *unboundedness predicates*

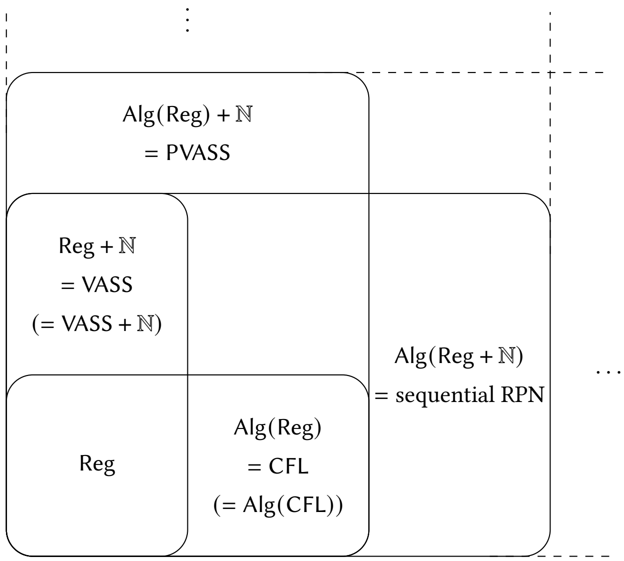


Figure 4: The hierarchy of classes obtained by the operations $\cdot + \mathbb{N}$ and $\mathrm{Alg}(\cdot)$.

of formal languages, as introduced by Czerwiński et al. [19]. That paper shows that for each unboundedness predicate, if it can be decided for regular languages, then it can even be decided for VASS.

Consider the *factor universality problem*: given a language $L \subseteq \Sigma^*$, does every word from Σ^* appear as a factor in L? This is an unboundedness predicate in the sense of [19], but it is not decidable for all amalgamation systems. Indeed, as observed in [19], factor universality is undecidable for one-counter automata, which, as a subclass of context-free languages, support amalgamation.

5 EXAMPLES OF AMALGAMATION SYSTEMS

In this section, we show that amalgamation systems are not limited to the few examples from Section 3. On the contrary, we present a whole hierarchy of language classes that support amalgamation (and satisfy the implicit effectiveness assumptions of Section 3.1.4), starting from regular languages as already discussed in Section 3.2 (see Section 5.1), by repeatedly applying the operations of "adding counters" (see Section 5.2) and "building stacks" (see Section 5.3)—those operations are denoted by $\cdot + \mathbb{N}$ and $\mathrm{Alg}(\cdot)$, respectively, in Fig. 4. This hierarchy includes some well-known classes—for example VASS languages and context-free languages as already mentioned in sections 3.3 and 3.4—, along with perhaps lesser-known classes—like PVASS languages or the languages of sequential recursive Petri nets (RPN) [34, 35].

As explained in the introduction, this entire hierarchy can also be envisioned from the perspective of *valence automata* by choosing appropriate graph monoids [65, 68]. In that framework, barring the classes that are already known to have an undecidable emptiness problem, one can show that our hierarchy actually exhausts all the remaining cases, yielding a proof of Main Theorem B; see Section 5.4 for a sketch and Appendix E for more details.

Well-quasi-ordered Decorations. Our constructions in this section actually capture more than the hierarchy of Fig. 4: we show in sections 5.2 and 5.3 how to "add counters" and "build stacks" to any language class C that supports amalgamation—provided an additional technical requirement is met by that class. To show that the classes $C + \mathbb{N}$ and $\mathrm{Alg}(C)$ support amalgamation, we need the property that C supports "wqo decorations," as we define now.

Given a run ρ and a set X, an X-*decoration* of ρ is a pair (ρ, w) where w is a word in X^* of length $|w| = |\mathrm{yield}(\rho)|$. Intuitively, one might think of a decoration as adding additional information from X to every letter in $\mathrm{yield}(\rho)$. For a set of runs R, we write $\mathrm{Deco}^X(R)$ for the set of all X-decorations of runs from R. If $(X, \leq)$ is a qo, we define the set of admissible embeddings between X-decorated runs by $E^X((\rho, u), (\sigma, v)) \stackrel{\text{def}}{=} \{f \in E(\rho, \sigma) \mid u_i \leq v_{f(i)}$ for all $i \in [1, |\mathrm{yield}(\rho)|]\}$; this defines a quasi-ordering $\trianglelefteq^X$ on $\mathrm{Deco}^X(R)$ such that $(\rho, u) \trianglelefteq^X (\sigma, v)$ if and only if there is f such that $\rho \trianglelefteq_f \sigma$ and $u_i \leq v_{f(i)}$ for all $i \in [1, |\mathrm{yield}(\rho)|]$.

Definition 5.1 (wqo decorations). An amalgamation system with set of runs $(R, \trianglelefteq)$ supports *wqo decorations* if, for *every* wqo $(X, \leq)$, the set of decorated runs $(\mathrm{Deco}^X(R), \trianglelefteq^X)$ is also a wqo.

By extension, we say a class of languages (that supports amalgamation) supports wqo decorations if for every language L in the class, there exists an amalgamation system that supports wqo decorations and whose language is L. In practice, supporting wqo decorations is not a major restriction: as part of our proofs, we will show that regular languages support wqo decorations, and furthermore that this property is maintained by the operations $\cdot + \mathbb{N}$ and $\mathrm{Alg}(\cdot)$.

Remark 5.2. Not every amalgamation system supports wqo decorations. Here is an example: let $\Sigma \stackrel{\text{def}}{=} \{a\}$ and define $R \stackrel{\text{def}}{=} \Sigma^*$ with $\mathrm{can}(w) \stackrel{\text{def}}{=} w$ for all $w \in \Sigma^*$. If $n \leq m$, we define the identity function $f\colon i \mapsto i$ for every $i \in [1, n]$ as the sole admissible embedding between a^n and a^m; all the gap words are ε except possibly $G_{n,f} = a^{m-n}$. Thus $\trianglelefteq$ is the prefix ordering, but over a unitary alphabet it gives rise to a wqo $(R, \trianglelefteq)$ isomorphic to $(\mathbb{N}, \leq)$. Also, those embeddings compose, and if $a^n \trianglelefteq a^r$ and $a^n \trianglelefteq a^s$ (thus with $n \leq r$ and $n \leq s$), then letting $m \stackrel{\text{def}}{=} r + s - n$ allows to amalgamate into a^m.

Consider now the wqo $X \stackrel{\text{def}}{=} (\{A, B\}, =)$, i.e., the finite set $\{A, B\}$ with the equality relation. Then $(\mathrm{Deco}^X(R), \trianglelefteq^X)$ is isomorphic with the prefix ordering over $\{A, B\}^*$, which is not a wqo. Indeed, among the decorated runs in $\mathrm{Deco}^X(R)$, one finds the decorated pairs $(a^n, B^{n-1}A)$ for all $n \geq 2$, and those decorated pairs form an infinite antichain: whenever $n < m$, when attempting to compare $(a^n, B^{n-1}A)$ with $(a^m, B^{m-1}A)$, the embedding $f \in E(a^n, a^m)$ maps n to itself, but the nth letter in the decoration of the first pair is A and the nth letter in the decoration of the second pair is B, thus $(a^n, B^{n-1}A) \not\trianglelefteq^X (a^m, B^{m-1}A)$.

5.1 Regular Languages

We already discussed the case of regular languages in Section 3.2. Let us provide here a more formal statement.

Theorem 5.3. *The class of regular languages supports amalgamation and wqo decorations.*

Proof. The class of regular languages is produced exactly by finite-state automata, which we can assume wlog. to be ε-free. The definitions of runs, their canonical decompositions, admissible embeddings, and how to amalgamate them were already given in Section 3.2. It remains to show that ε-free finite-state automata support wqo decorations. Let $(X, \leq)$ be a wqo, and (ρ, w) an X-decoration of a run $\rho = (q_0, a_1, q_1) \cdots (q_{n-1}, a_n, q_n) \in R$. Since the automaton is ε-free, w is of length $|w| = n$ and can be written as $w = x_1 \cdots x_n$. Then the map $r\colon \mathrm{Deco}^X(R) \to (\Delta \times X)^*$ defined by $r\colon (\rho, w) \mapsto ((q_0, a_1, q_1), x_1) \cdots ((q_{n-1}, a_n, q_n), x_n)$ is an *order reflection*, in that if $r(\rho, w) \leq_* r(\rho', w')$, then $(\rho, w) \trianglelefteq^X (\rho', w')$. By Dickson's and Higman's lemmata, $((\Delta \times X)^*, \leq_*)$ is a wqo, and this order reflection shows that $(\mathrm{Deco}^X(R), \trianglelefteq^X)$ is also a wqo: r pointwise maps bad sequences $(\rho_0, w_0), (\rho_1, w_1), \ldots$ over $(\mathrm{Deco}^X(R), \trianglelefteq^X)$ to bad sequences $r(\rho_0, w_0), r(\rho_1, w_1), \ldots$ of the same length over $((\Delta \times X)^*, \leq_*)$, thus bad sequences over $(\mathrm{Deco}^X(R), \trianglelefteq^X)$ must be finite. ◻

5.2 Counter Extension

Vector addition systems were presented in Section 3.3 as finite-state automata that additionally modify a set of d counters. A natural question is whether we can generalise this operation of "adding counters" to arbitrary amalgamation systems. To that end, we first define in Section 5.2.1 a generic operator that takes a language class C and forms a language class $C + \mathbb{N}$ of languages in C extended with $d > 0$ counters, such that for instance $\mathrm{Reg} + \mathbb{N} = \mathrm{VASS}$. We then show in Section 5.2.2 that, if C supports amalgamation and wqo decorations, then so does $C + \mathbb{N}$.

5.2.1 Extending Languages. Fix $d > 0$ the number of counters we wish to add. We use a finite alphabet $\mathbb{U}_d \subseteq \mathbb{Z}^d$ of unit updates, defined by $\mathbb{U}_d \stackrel{\text{def}}{=} \{\mathbf{0}\} \cup \{\mathbf{e}_i, -\mathbf{e}_i \mid i \in [1, d]\}$ where each $\mathbf{e}_i$ is the unit vector such that $\mathbf{e}_i(i) = 1$ and $\mathbf{e}_i(j) = 0$ for all $j \neq i$. Then the morphism $\delta\colon \mathbb{U}_d^* \to \mathbb{Z}^d$ maps words over $\mathbb{U}_d$ to their effect, and is defined by $\delta(\mathbf{u}_1 \cdots \mathbf{u}_n) \stackrel{\text{def}}{=} \sum_{j=1}^n \mathbf{u}_j$. Finally, let $N_d \subseteq \mathbb{U}_d^*$ be the language of all $\mathbb{N}$-counter-like words over $\mathbb{U}_d$; formally,

$$N_d \stackrel{\text{def}}{=} \{w \in \mathbb{U}_d^* \mid \delta(v) \geq \mathbf{0} \text{ for all prefixes } v \text{ of } w \text{ and } \delta(w) = \mathbf{0}\} .$$

Put differently, N_d is the language of the VASS with a single state q and a transition $(q, \mathbf{u}, \mathbf{u}, q)$ for each $\mathbf{u} \in \mathbb{U}_d$.

Let Δ be a finite alphabet; a morphism $\eta\colon \Delta^* \to \mathbb{U}_d^*$ is *tame* if, for all $a \in \Delta$ and all $i \in [1, d]$, all the occurrences of $\mathbf{e}_i$ in $\eta(a)$ occur before all the occurrences of $-\mathbf{e}_i$. Let Σ also be a finite alphabet; a morphism $\alpha\colon \Delta^* \to \Sigma^*$ is *alphabetic* if $\alpha(\Delta) \subseteq \Sigma_\varepsilon$.

Definition 5.4 (Counter extension). Let L be a language over a finite alphabet Δ. For a tame morphism $\eta\colon \Delta^* \to \mathbb{U}_d^*$ and an alphabetic morphism $\alpha\colon \Delta^* \to \Sigma^*$ into a finite alphabet Σ, let $L_{\eta,\alpha}$ be the language $\alpha(\eta^{-1}(N_d) \cap L)$. For a class of languages C, $C + \mathbb{N}$ is the class of languages $L_{\eta,\alpha}$ when L ranges over C.

Very informally, for $C = \mathrm{Reg}$, $L \subseteq \Delta^*$ describes transition sequences, η the effect of each transition encoded as a word over $\mathbb{U}_d$, and α its label in Σ_ε, resulting in $L_{\eta,\alpha}$ being a VASS language.

5.2.2 Extending Amalgamation Systems. We are going to show that this construction is well behaved in the following sense.

Theorem 5.5. *If C is a class of languages that supports concatenative amalgamation and wqo decorations, then so does $C + \mathbb{N}$.*

To this end, fix $d > 0$ and let $S = (\Delta, R, E, \mathrm{can})$ be an amalgamation system supporting wqo decorations and accepting a language $L \in C$, and let η and α be morphisms as in Definition 5.4. Our goal is to define an amalgamation system $S_{\eta,\alpha}$ that supports wqo decorations such that $L(S_{\eta,\alpha}) = L_{\eta,\alpha}$.

We decorate runs $\rho \in R$ with pairs of counter valuations from $P \stackrel{\text{def}}{=} \mathbb{N}^d \times \mathbb{N}^d$. Consider a P-decorated run (ρ, w) and let $a_1 \cdots a_n = \mathrm{yield}(\rho)$ be the word accepted by ρ. Then $w = (\mathbf{u}_1, \mathbf{v}_1) \cdots (\mathbf{u}_n, \mathbf{v}_n)$. We say that (ρ, w) is *coherent* if $\mathbf{v}_i = \mathbf{u}_i + \delta(\eta(a_i))$ for all $i \in [1, n]$ and $\mathbf{v}_i = \mathbf{u}_{i+1}$ for all $i \in [1, n-1]$. We say that (ρ, w) is *accepting* if it is coherent and additionally the initial counters are $\mathbf{u}_1 = \mathbf{0}$ and the final counters are $\mathbf{v}_n = \mathbf{0}$.

Let R_η be the set of accepting decorated runs in $\mathrm{Deco}^P(R)$. If the canonical decomposition of a run $\rho \in R$ is $a_1 \cdots a_m$ with each $a_i \in \Delta_\varepsilon$, then the canonical decomposition of a decorated run $(\rho, w) \in R_\eta$ is defined as $\alpha(a_1) \cdots \alpha(a_m)$, with each $\alpha(a_i) \in \Sigma_\varepsilon$ by definition of α. Let $S_{\eta,\alpha} \stackrel{\text{def}}{=} (\Sigma, R_\eta, E^P, \alpha \circ \mathrm{can})$. The following claim, proven in Appendix C, shows that this yields the intended language.

Claim 5.6. *If S is an amalgamation system and L is its language, then $L(S_{\eta,\alpha}) = L_{\eta,\alpha}$.*

In order to complete the proof of Theorem 5.5, it remains to show that $S_{\eta,\alpha}$ satisfies the conditions of Section 3.1.2 (see Claim 5.7) and supports wqo decorations in the sense of Definition 5.1 (see Claim 5.8).

Claim 5.7. *If S is an amalgamation system that supports wqo decorations, then $S_{\eta,\alpha}$ is an amalgamation system.*

Proof. We show that $S_{\eta,\alpha}$ satisfies the conditions of Section 3.1.2.
Composition. As we have defined above, the admissible embeddings in $E^P((\rho, u), (\sigma, v))$ of two runs in R_η are those of $E(\rho, \sigma)$ that respect the ordering on decorations in P. These embeddings compose, because the admissible embeddings of S compose and because additionally $\leq$ on $P \stackrel{\text{def}}{=} \mathbb{N}^d \times \mathbb{N}^d$ is transitive.
Well-quasi-order. Because we assume S supports wqo decorations and $(P, \leq)$ is a wqo, $(\mathrm{Deco}^P(R), \trianglelefteq^P)$ is a wqo, and the induced $(R_\eta, \trianglelefteq^P)$ as well.
Amalgamation. The construction mirrors the one presented in Section 3.3.3. Let $(\rho_0, w_0) \in R_\eta$ be a run with $a_1 \cdots a_n$ the accepted word of ρ_0 and $w_0 = (\mathbf{u}_1, \mathbf{v}_1) \cdots (\mathbf{u}_n, \mathbf{v}_n)$. Assume $(\rho_0, w_0) \trianglelefteq_f^P (\rho_1, w_1)$ and $(\rho_0, w_0) \trianglelefteq_g^P (\rho_2, w_2)$ for $(\rho_1, w_1), (\rho_2, w_2) \in R_\eta$. By definition of our decorated embedding, we have $\rho_0 \trianglelefteq_f \rho_1$ and $\rho_0 \trianglelefteq_g \rho_2$, and for each $i \in [1, n]$ there exists $\mathbf{c}_i, \mathbf{d}_i \in \mathbb{N}^d$ such that the $f(i)$th pair in w_1 is $(\mathbf{u}_i + \mathbf{c}_i, \mathbf{v}_i + \mathbf{c}_i)$ and the $g(i)$th pair in w_2 is $(\mathbf{u}_i + \mathbf{d}_i, \mathbf{v}_i + \mathbf{d}_i)$. Because C is an amalgamation system, there exists a run $\rho_3 \in R$ with $\rho_1 \trianglelefteq_{f'} \rho_3$, $\rho_2 \trianglelefteq_{g'} \rho_3$, and $\rho_0 \trianglelefteq_h \rho_3$ with $h = f \circ f' = g \circ g'$ that satisfies the concatenative amalgamation condition. It remains to show that we can decorate ρ_3 in an accepting way to also satisfy amalgamation.

To do so, we let the $h(i)$th pair of w_3 be $(\mathbf{u}_i + \mathbf{c}_i + \mathbf{d}_i, \mathbf{v}_i + \mathbf{c}_i + \mathbf{d}_i)$. As $\mathbf{u}_i + \delta(\eta(a_i)) = \mathbf{v}_i$, we also have $\mathbf{u}_i + \mathbf{c}_i + \mathbf{d}_i + \delta(\eta(a_i)) = \mathbf{v}_i + \mathbf{c}_i + \mathbf{d}_i$ as desired. Consider now the ith gap $\mathrm{G}_{i,h}$ and assume wlog. that $\mathrm{G}_{i,h} =$

$\mathrm{G}_{i,f}\mathrm{G}_{i,g}$. Observe that in w_1, we have $\mathbf{u}_i+\mathbf{c}_i+\delta(\eta(\mathrm{G}_{i,f})) = \mathbf{u}_i+\mathbf{c}_{i+1}$. Similarly, in w_2 we have $\mathbf{u}_i+\mathbf{d}_i+\delta(\eta(\mathrm{G}_{i,g})) = \mathbf{u}_i+\mathbf{d}_{i+1}$. Then $\mathbf{u}_i+\mathbf{c}_i+\mathbf{d}_i+\delta(\eta(\mathrm{G}_{i,f})) = \mathbf{u}_i+\mathbf{c}_{i+1}+\mathbf{d}_i$ and by monotonicity we can decorate w_3 coherently along $\mathrm{G}_{i,f}$ by adding $\mathbf{d}_i$ to all the pairs from w_1 along that segment, and then $\mathbf{u}_i+\mathbf{c}_{i+1}+\mathbf{d}_i+\delta(\eta(\mathrm{G}_{i,g})) = \mathbf{u}_i+\mathbf{c}_{i+1}+\mathbf{d}_{i+1}$ as desired and again by monotonicity we can decorate w_3 coherently along $\mathrm{G}_{i,g}$ by adding $\mathbf{c}_{i+1}$ to all the pairs from w_2 along that segment. The produced decorated run (ρ_3, w_3) is coherent by construction and satisfies $(\rho_1, w_1) \trianglelefteq^P_{f'} (\rho_3, w_3)$ and $(\rho_2, w_2) \trianglelefteq^P_{g'} (\rho_3, w_3)$ as desired. Using the same abuse of notation as in Section 3.3.3 for the border cases, because we are dealing with accepting runs, in the case of $i = 0$ we additionally have $\mathbf{c}_0 = \mathbf{d}_0 = \mathbf{0}$, and analogously in the case of $i = n$ we have $\mathbf{c}_{n+1} = \mathbf{d}_{n+1} = \mathbf{0}$, thus (ρ_3, w_3) is also accepting. ◻

Claim 5.8. *If S supports wqo decorations, then so does $S_{\eta,\alpha}$.*

Proof. Observe that a decoration of a run $(\rho, w_1 \cdots w_n) \in R_\eta$ with a sequence $x_1 \cdots x_n \in X^*$ over a wqo X is equivalent to a decoration of the run $\rho \in R$ with the sequence $(w_1, x_1)\cdots(w_n, x_n) \in (P \times X)^*$ over the wqo alphabet $\mathbb{N}^d \times \mathbb{N}^d \times X$. ◻

5.3 Algebraic Extension

We introduce now, as a generalisation of the case of context-free languages presented in Section 3.4, how to support amalgamation in the algebraic closure of a class of languages.

Given a class of languages C, a *C-grammar* is a tuple $\mathcal{G} = (N, \Sigma, S, \{L_A\}_{A\in N})$, where $S \in N$ and each L_A is a language from C with $L_A \subseteq (N \cup \Sigma)^*$. We write that $uAv \Rightarrow uwv$ if $w \in L_A$. The language of a C-grammar is the set $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$. For example, every context-free grammar can be seen as a Fin-grammar, where Fin is the class of finite languages. The class of context-free languages is also defined by the so-called “extended” context-free grammars allowing regular expressions in their productions, i.e., Reg-grammars.

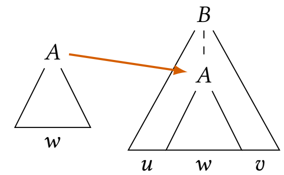
Definition 5.9 (Algebraic extension). Given a class of languages C, we denote by $\mathsf{Alg}(C)$ the *algebraic extension* of C, that is, the class of all languages recognised by C-grammars.

We are going to show that, if C is well-behaved, then so is $\mathsf{Alg}(C)$.

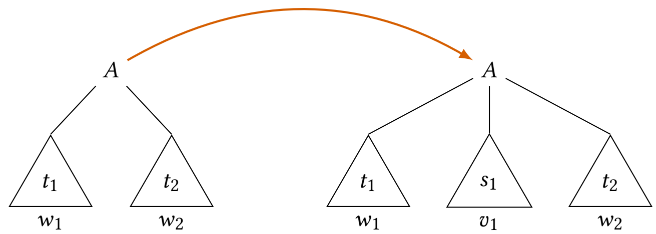
Theorem 5.10. *If C is a class of languages that supports concatenative amalgamation and wqo decorations, then so does its algebraic extension $\mathsf{Alg}(C)$.*

Let us fix a C-grammar $\mathcal{G}$ and write $\mathcal{M}_A$ for the amalgamation system with wqo decorations that recognises the language L_A. We assume wlog. that all words in each L_A are either single terminal letters from Σ_ε or only contain non-terminal symbols.

Just like with context-free grammars in Section 3.4, the derivations of a C-grammar can be viewed as trees, with nodes labelled either ε or with pairs of non-terminals A and runs ρ of $\mathcal{M}_A$. We call ρ the *explanation* of the expansion of the non-terminal A at this step. More specifically, if ρ is a run in $\mathcal{M}_A$ with $\mathsf{yield}(\rho) \in \Sigma^*$, then $(A \to \rho)[]$ is a tree. Otherwise, let $X_1X_2\cdots X_n$ be the projection of $\mathsf{yield}(\rho)$ to N and $t_1, \ldots, t_n$ be X_1-, …, X_n-rooted trees. Then $(A \to \rho)[t_1, \ldots, t_n]$ is a tree as well. For the remainder of this section, if we write $\mathsf{yield}(\rho)$ or $\mathsf{can}(\rho) = u_0X_1u_1\cdots X_nu_n$, we



(a) Mapping to a subtree adds letters to the left and right.



(b) Larger runs in the underlying system $\mathcal{M}_A$ interleave letters.

Figure 5: Gaps in the canonical decomposition of trees.

assume that $X_i \in N$ and $u_j \in \Sigma^*$. We write $\mu_\rho(i)$ for the map $[1, n] \to [1, |\rho|_{\mathsf{can}}]$ that associates to every X_i its position in the canonical decomposition.

Canonical decompositions. Let $\tau = (A \to \rho_0)[\ldots]$ and $\pi = (B \to \rho_1)[\ldots]$ be trees of $\mathcal{G}$ and assume that $\tau \trianglelefteq \pi$. Additional letters in the output of π can come from one of two sources: From the mapping of τ to a subtree of π (Figure 5a) or from the image of τ in π being explained by a larger run (Figure 5b). Separating these two sources motivates the canonical decomposition for derivation trees.

Definition 5.11. Assume $\tau = (A \to \rho)[t_1, \ldots, t_n]$ and $\mathsf{can}(\rho) = u_0X_1u_1\cdots X_nu_n$. We define $\mathsf{can}(\tau) = \varepsilon \cdot u_0 \cdot \varepsilon\mathsf{can}(t_1)\varepsilon \cdot u_1 \cdots u_{n-1} \cdot \varepsilon\mathsf{can}(t_n)\varepsilon \cdot u_n \cdot \varepsilon$.

Intuitively, we wrap the canonical decomposition of τ itself and of each child t_i in ε on either side to delimit gap words produced by a mapping of t_i to a non-trivial descendant from those obtained by runs larger than ρ in the image of τ.

This also yields the expected definition of $\mathsf{yield}(\cdot)$, being $\mathsf{yield}(\tau) = \mathsf{yield}(\rho)$ if τ is a leaf node, and $\mathsf{yield}(\tau) = u_0\mathsf{yield}(t_1)u_1\cdots\mathsf{yield}(t_n)u_n$ otherwise. We write $\mathcal{T}(\mathcal{G})$ for the set of all the trees of $\mathcal{G}$, and $R_{\mathcal{G}}$ for the S-rooted ones. Then $L(\mathcal{G}) = \bigcup_{\rho\in R_{\mathcal{G}}} \mathsf{yield}(\rho)$ as desired.

The embedding between trees is similar to the one we used for context-free grammars in Section 3.4, but needs to be generalised slightly: when mapping $\tau_1 = (A \to \rho)[\ldots]$ to $\tau_2/p = (A \to \sigma)[\ldots]$, we require that ρ embeds into σ such that the corresponding subtrees also embed. Formally, let τ_1 and τ_2 be trees from $\mathcal{T}(\mathcal{G})$. Denote the run embeddings between runs of the various systems $\{\mathcal{M}_A\}_{A\in N}$ by $\preceq$. We write $\tau_1 \trianglelefteq \tau_2$ if there exists a subtree τ_2/p such that

(1) $\tau_1 = (A \to \rho)[t_1, \ldots, t_n]$, $\tau_2/p = (A \to \sigma)[t'_1, \ldots, t'_k]$ and $\rho \preceq_f \sigma$, and
(2) $t_i \trianglelefteq t'_{g(i)}$ for all $i \in [1, n]$, where $g = \mu_\sigma^{-1} \circ f \circ \mu_\rho$.

For the details of the following statements that together show Theorem 5.10, we refer to Appendix D.

Lemma 5.12. *$(\mathcal{T}(\mathcal{G}), \trianglelefteq)$ is a well-quasi-order.*

Lemma 5.13. *$\mathcal{G}$ supports well-quasi-ordered decorations.*

Lemma 5.14. *If $\tau_0 \trianglelefteq_f \tau_1$ and $\tau_1 \trianglelefteq_g \tau_2$, then $\tau_0 \trianglelefteq_{g \circ f} \tau_2$.*

Lemma 5.15. *If τ_0, τ_1, τ_2 are all A-rooted trees such that $\tau_0 \trianglelefteq_f \tau_1$ and $\tau_0 \trianglelefteq_g \tau_2$, then for every choice of $i \in [0, |\tau_0|_{\mathsf{can}}]$ there exists an A-rooted tree τ_3 with $\tau_1 \trianglelefteq_{f'} \tau_3$ and $\tau_2 \trianglelefteq_{g'} \tau_3$ such that*

(1) $f' \circ f = g' \circ g$ (we write h for this composition),
(2) $\mathrm{G}_{j,h} \in \{\mathrm{G}_{j,f}, \mathrm{G}_{j,g}, \mathrm{G}_{j,g}\mathrm{G}_{j,f}\}$ for every $j \in [0, |\tau_0|_{\mathsf{can}}]$, and in particular
(3) $\mathrm{G}_{i,h} = \mathrm{G}_{i,f}\mathrm{G}_{i,g}$ for the chosen i.

5.4 Valence Automata

In this subsection, we give a rough sketch of why, armed with the two aforementioned mechanisms of *adding counters* and *algebraic extensions*, we cover the entire class of valence automata over graph monoids, except for those where Turing-completeness is known.

In a *valence automaton*, we have finitely many control states, and a (potentially infinite-state) storage mechanism that is specified by a monoid. In the framework of valence automata over graph monoids (see [65, 68] for overviews), one considers valence automata where the monoid is defined by a finite (undirected) graph Γ that may have self-loops. The resulting *graph monoid* is denoted $\mathbb{M}\Gamma$. The exact definition of valence automata and graph monoids can be found in Appendix E, but will not be necessary for this sketch.

In [67], it is shown that if Γ contains certain induced subgraphs (essentially: a path on four nodes or a cycle on four nodes), then valence automata over $\mathbb{M}\Gamma$ accept all recursively enumerable languages. Thus, for Main Theorem B, we may assume that these do not occur. It follows from [67] that all the *remaining* monoids $\mathbb{M}\Gamma$ belong to the class PD (for "potentially decidable"), where PD (i) contains the trivial monoid 1 and (ii) for all monoids M, N in PD, the class PD also contains $M * N$, $M \times \mathbb{B}$, and $M \times \mathbb{Z}$. Here, $\mathbb{B}$ is the so-called *bicyclic monoid*, which corresponds to a single so-called "partially blind," $\mathbb{N}$-valued, counter: valence automata over $\mathbb{B}$ are essentially 1-dimensional VASS. Moreover, $M * N$ denotes the free product of monoids. Without giving an exact definition, they can be thought of as *stacks*: valence automata over $M * N$ are essentially valence automata with stacks, where each entry is a configuration of the storage mechanisms described by M or N. Moreover, $\mathbb{Z}$ corresponds to a so-called "blind," $\mathbb{Z}$-valued counter.

It follows from known results on valence automata that going from M and N to $M * N$ results in languages in the algebraic extension $\mathsf{Alg}(\mathsf{VA}(M) \cup \mathsf{VA}(N))$, where $\mathsf{VA}(M)$ is the class of languages accepted by valence automata over M. Moreover, going from M to $M \times \mathbb{B}$ can be seen as *adding a counter* as in our definition of $C + \mathbb{N}$ for language classes C. Valence automata over $M \times \mathbb{Z}$ can be simulated by valence automata over $M \times \mathbb{B} \times \mathbb{B}$, so that $M \times \mathbb{Z}$ can be treated similarly. As this exhausts all the potentially decidable graph monoids and $\mathsf{VA}(M)$ is always a full trio [22, Thm. 4.1], Main Theorem B follows; see Appendix E for more details.

6 CONCLUSION

We hope that we have demonstrated the surprisingly flexible nature of amalgamation systems. Their structure is at once simple enough to be a fit for several computational models, and powerful enough to be able to answer a number of open problems.

We think that this approach merits further investigation. In particular, we are interested in the following questions:

(a) Which other problems are decidable for amalgamation systems?
(b) Are there amalgamation systems that are not valence systems?
(c) Is there a natural, non-trivial class that subsumes amalgamation systems and their algorithmic properties?
(d) Is there a generic approach to the complexity of the algorithmic problems of Main Theorem A?

ACKNOWLEDGMENTS

Funded by the European Union (ERC, FINABIS, 101077902) and DFG (Project 389792660 TRR 248–CPEC). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

We are grateful to Jérôme Leroux for the discussion about [47].

REFERENCES

[1] Parosh A. Abdulla, Karlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. 2000. Algorithmic analysis of programs with well quasi-ordered domains. *Inform. and Comput.* 160, 1–2 (2000), 109–127. https://doi.org/10.1006/inco.1999.2843
[2] Ashwani Anand and Georg Zetzsche. 2023. Priority downward closures. In *Proc. Concur 2023 (Leibniz Intl. Proc. Informatics, Vol. 279)*. LZI, 39:1–39:18. https://doi.org/10.4230/LIPIcs.CONCUR.2023.39
[3] Mohamed Faouzi Atig and Pierre Ganty. 2011. Approximating Petri net reachability along context-free traces. In *Proc. FSTTCS 2011 (Leibniz Intl. Proc. Informatics, Vol. 13)*. LZI, 152–163. https://doi.org/10.4230/LIPICS.FSTTCS.2011.152
[4] David Barozzini, Lorenzo Clemente, Thomas Colcombet, and Paweł Parys. 2022. Cost automata, safe schemes, and downward closures. *Fundam. Informaticae* 188, 3 (2022), 127–178. https://doi.org/10.3233/FI-222145
[5] David Barozzini, Paweł Parys, and Jan Wroblewski. 2022. Unboundedness for recursion schemes: A simpler type system. In *Proc. ICALP 2022 (Leibniz Intl. Proc. Informatics, Vol. 229)*. LZI, 112:1–112:19. https://doi.org/10.4230/LIPIcs.ICALP.2022.112
[6] Pascal Baumann, Flavio D'Alessandro, Oscar Ibarra, Moses Ganardi, Ian McQuillan, Lia Schütze, and Georg Zetzsche. 2023. Unboundedness problems for machines with reversal-bounded counters. In *Proc. FoSSaCS 2023 (Lect. Notes Comput. Sci., Vol. 13992)*. Springer, 240–264. https://doi.org/10.1007/978-3-031-30829-1_12
[7] Jean Berstel. 1979. *Transductions and Context-Free Languages*. Teubner. https://doi.org/10.1007/978-3-663-09367-1
[8] Rémi Bonnet. 2013. *Theory of well-structured transition systems and extended vector-addition systems*. Ph. D. Dissertation. ENS Cachan, France.
[9] P. Buckheister and Georg Zetzsche. 2013. Semilinearity and Context-Freeness of Languages Accepted by Valence Automata. In *Proc. MFCS 2013 (Lecture Notes in Computer Science, Vol. 8087)*. Springer, 231–242. https://doi.org/10.1007/978-3-642-40313-2_22
[10] Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. 2012. Bounded Parikh automata. *Int. J. Found. Comput. Sci.* 23, 8 (2012), 1691–1710. https://doi.org/10.1142/S0129054112400709
[11] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. 1996. Unreliable channels are easier to verify than perfect channels. *Inf. Comput.* 124, 1 (1996), 20–31. https://doi.org/10.1006/inco.1996.0003
[12] Pierre Chambart, Alain Finkel, and Sylvain Schmitz. 2016. Forward analysis and model checking for trace bounded WSTS. *Theor. Comput. Sci.* 637 (2016), 1–29. https://doi.org/10.1016/J.TCS.2016.04.020
[13] Lorenzo Clemente, Wojciech Czerwinski, Slawomir Lasota, and Charles Paperman. 2017. Separability of reachability sets of vector addition systems. In *Proc. of STACS 2017 (Leibniz Intl. Proc. Informatics, Vol. 66)*. LZI. https://doi.org/10.4230/LIPICS.STACS.2017.24

[14] Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. 2016. The diagonal problem for higher-order recursion schemes is decidable. In *Proc. LICS 2016*. ACM, 96–105. https://doi.org/10.1145/2933575.2934527

[15] Bruno Courcelle. 1991. On constructing obstruction sets of words. *Bull. EATCS* 44 (1991), 178–186.

[16] Wojciech Czerwinski and Sławomir Lasota. 2019. Regular separability of one counter automata. *Log. Methods Comput. Sci.* 15, 2 (2019). https://doi.org/10.23638/LMCS-15(2:20)2019

[17] Wojciech Czerwinski, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. 2017. A characterization for decidable separability by piecewise testable languages. *Discret. Math. Theor. Comput. Sci.* 19, 4 (2017). https://doi.org/10.23638/DMTCS-19-4-1

[18] Wojciech Czerwinski and Georg Zetzsche. 2020. An approach to regular separability in vector addition systems. In *Proc. LICS 2020*. ACM, 341–354. https://doi.org/10.1145/3373718.3394776

[19] Wojciech Czerwiński, Piotr Hofman, and Georg Zetzsche. 2018. Unboundedness problems for languages of vector addition systems. In *Proc. ICALP 2018 (Leibniz Intl. Proc. Informatics, Vol. 107)*. LZI. https://doi.org/10.4230/LIPIcs.ICALP.2018.119

[20] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. 1998. Reset nets between decidability and undecidability. In *Proc. ICALP 1998 (Lect. Notes Comput. Sci., Vol. 1443)*. Springer, 103–115. https://doi.org/10.1007/BFB0055044

[21] Joey Eremondi, Oscar H. Ibarra, and Ian McQuillan. 2015. On the density of context-free and counter languages. In *Proc. DLT 2015 (Lect. Notes Comput. Sci., Vol. 9168)*. Springer, 228–239. https://doi.org/10.1007/978-3-319-21500-6_18

[22] Henning Fernau and Ralf Stiebe. 2002. Sequential grammars and automata with valences. *Theor. Comput. Sci.* 276, 1-2 (2002), 377–405. https://doi.org/10.1016/S0304-3975(01)00282-1

[23] Alain Finkel. 1987. A generalization of the procedure of Karp and Miller to well structured transition systems. In *Proc. ICALP 1987 (Lect. Notes Comput. Sci., Vol. 267)*. Springer, 499–508. https://doi.org/10.1007/3-540-18088-5_43

[24] Alain Finkel, Shankara Narayanan Krishna, Khushraj Madnani, Rupak Majumdar, and Georg Zetzsche. 2023. Counter machines with infrequent reversals. In *Proc. FSTTCS 2023 (Leibniz Intl. Proc. Informatics, Vol. 284)*. LZI. https://doi.org/10.4230/LIPICS.FSTTCS.2023.42

[25] Alain Finkel and Philippe Schnoebelen. 2001. Well-structured transition systems everywhere! *Theor. Comput. Sci.* 256, 1–2 (2001), 63–92. https://doi.org/10.1016/S0304-3975(00)00102-X

[26] Moses Ganardi, Rupak Majumdar, and Georg Zetzsche. 2022. The complexity of bidirected reachability in valence systems. In *Proc. of LICS 2022*. ACM. https://doi.org/10.1145/3531130.3533345

[27] Pierre Ganty and Damir Valput. 2016. Bounded-oscillation pushdown automata. In *Proc. GandALF 2016 (Elec. Proc. Theor. Comput. Sci., Vol. 226)*. 178–197. https://doi.org/10.4204/EPTCS.226.13

[28] Pawel Gawrychowski, Dalia Krieger, Narad Rampersad, and Jeffrey O. Shallit. 2010. Finding the growth rate of a regular or context-free language in polynomial time. *Int. J. Found. Comput. Sci.* 21, 4 (2010), 597–618. https://doi.org/10.1142/S0129054110007441

[29] Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. 2007. Well-structured languages. *Acta Inf.* 44, 3-4 (2007), 249–288. https://doi.org/10.1007/S00236-007-0050-3

[30] Seymour Ginsburg and Edwin H. Spanier. 1964. Bounded Algol-like languages. *Trans. Amer. Math. Soc.* 113, 2 (1964), 333–368. https://doi.org/10.2307/1994067

[31] Jean Goubault-Larrecq and Sylvain Schmitz. 2016. Deciding piecewise testable separability for regular tree languages. In *Proc. ICALP 2016 (Leibniz Intl. Proc. Informatics, Vol. 55)*. LZI, 97:1–97:15. https://doi.org/10.4230/LIPIcs.ICALP.2016.97

[32] Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. 2014. The power of priority channel systems. *Log. Meth. Comput. Sci.* 10, 4 (2014). https://doi.org/10.2168/LMCS-10(4:4)2014

[33] Peter Habermehl, Roland Meyer, and Harro Wimmel. 2010. The downward-closure of Petri net languages. In *Proc. ICALP 2010 (Lect. Notes Comput. Sci., Vol. 6199)*. Springer, 466–477. https://doi.org/10.1007/978-3-642-14162-1_39

[34] Serge Haddad and Denis Poitrenaud. 2001. Checking linear temporal formulas on sequential recursive Petri nets. In *Proc. TIME 2001*. IEEE Computer Society, 198–205. https://doi.org/10.1109/TIME.2001.930718

[35] Serge Haddad and Denis Poitrenaud. 2007. Recursive Petri nets. *Acta Inf.* 44, 7-8 (2007), 463–508. https://doi.org/10.1007/S00236-007-0055-Y

[36] Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. 2016. Unboundedness and downward closures of higher-order pushdown automata. In *Proc. POPL 2016*. ACM, 151–163. https://doi.org/10.1145/2837614.2837627

[37] Matthew Hague and Anthony Widjaja Lin. 2011. Model checking recursive programs with numeric data types. In *Proc. CAV 2011 (Lect. Notes Comput. Sci., Vol. 6806)*. Springer, 743–759. https://doi.org/10.1007/978-3-642-22110-1_60

[38] Leonard H Haines. 1969. On free monoids partially ordered by embedding. *J. Comb. Theory* 6, 1 (1969), 94–98. https://doi.org/10.1016/S0021-9800(69)80111-0

[39] Dirk Hauschildt. 1990. *Semilinearity of the reachability set is decidable for Petri nets*. Ph. D. Dissertation. University of Hamburg, Germany. https://d-nb.info/911413707

[40] Dirk Hauschildt and Matthias Jantzen. 1994. Petri net algorithms in the theory of matrix grammars. *Acta Inf.* 31, 8 (1994), 719–728. https://doi.org/10.1007/BF01178731

[41] Graham Higman. 1952. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.* s3-2, 1 (1952), 326–336. https://doi.org/10.1112/plms/s3-2.1.326

[42] Oscar H. Ibarra and Bala Ravikumar. 1986. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *Proc. STACS 1986 (Lect. Notes Comput. Sci., Vol. 210)*. Springer, 171–179. https://doi.org/10.1007/3-540-16078-7_74

[43] S. Rao Kosaraju. 1982. Decidability of reachability in vector addition systems. In *Proc. STOC 1982*. ACM, 267–281. https://doi.org/10.1145/800070.802201

[44] Joseph B. Kruskal. 1972. The theory of well-quasi-ordering: A frequently discovered concept. *J. Comb. Theory A* 13, 3 (1972), 297–305. https://doi.org/10.1016/0097-3165(72)90063-5

[45] Jean-Luc Lambert. 1992. A structure to decide reachability in Petri nets. *Theor. Comput. Sci.* 99, 1 (1992), 79–104. https://doi.org/10.1016/0304-3975(92)90173-D

[46] Jérôme Leroux. 2012. Vector addition systems reachability problem (a simpler solution). In *Turing-100—The Alan Turing Centenary (EPiC Series in Computing, Vol. 10)*. EasyChair, 214–228. https://doi.org/10.29007/BNX2

[47] Jérôme Leroux. 2019. Personal communication.

[48] J. Leroux, M. Praveen, Ph. Schnoebelen, and G. Sutre. 2019. On functions weakly computable by pushdown Petri nets and related systems. *Log. Methods Comput. Sci.* 15, 4 (2019). https://doi.org/10.23638/LMCS-15(4:15)2019

[49] Jérôme Leroux and Sylvain Schmitz. 2015. Demystifying reachability in vector addition systems. In *Proc. LICS 2015*. IEEE Computer Society, 56–67. https://doi.org/10.1109/LICS.2015.16

[50] Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. 2015. On the coverability problem for pushdown vector addition systems in one dimension. In *Proc. ICALP 2015 (Lect. Notes Comput. Sci., Vol. 9135)*. Springer, 324–336. https://doi.org/10.1007/978-3-662-47666-6_26

[51] Ernst W. Mayr. 1981. An algorithm for the general Petri net reachability problem. In *Proc. STOC 1981*. 238–246. https://doi.org/10.1145/800076.802477

[52] Richard Mayr. 2003. Undecidable problems in unreliable computations. *Theor. Comput. Sci.* 297, 1–3 (2003), 337–354. https://doi.org/10.1016/S0304-3975(02)00646-1

[53] Crispin St. John Alvah Nash-Williams. 1963. On well-quasi-ordering finite trees. *Math. Proc. Cambridge Phil. Soc.* 59, 4 (1963), 833–835. https://doi.org/10.1017/S0305004100003844

[54] Rohit J. Parikh. 1966. On context-free languages. *J. ACM* 13, 4 (1966), 570–581. https://doi.org/10.1145/321356.321364

[55] Paweł Parys. 2017. The complexity of the diagonal problem for recursion schemes. In *Proc. FSTTCS 2017 (Leibniz Intl. Proc. Informatics, Vol. 93)*. LZI, 45:1–45:14. https://doi.org/10.4230/LIPIcs.FSTTCS.2017.45

[56] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. 2013. Separating regular languages by piecewise testable and unambiguous languages. In *Proc. MFCS 2013 (Lect. Notes Comput. Sci., Vol. 8087)*. Springer, 729–740. https://doi.org/10.1007/978-3-642-40313-2_64

[57] Klaus Reinhardt. 2008. Reachability in Petri nets with inhibitor arcs. In *Proc. RP 2008 (Elec. Notes Theor. Comput. Sci., Vol. 223)*. Elsevier, 239–264. https://doi.org/10.1016/j.entcs.2008.12.042

[58] Sylvain Schmitz and Philippe Schnoebelen. 2012. Algorithmic Aspects of WQO Theory. Lecture notes. http://cel.archives-ouvertes.fr/cel-00727025

[59] Sylvain Schmitz and Georg Zetzsche. 2019. Coverability is undecidable in one-dimensional pushdown vector addition systems with resets. In *Proc. RP 2019 (Lect. Notes Comput. Sci., Vol. 11674)*. Springer, 193–201. https://doi.org/10.1007/978-3-030-30806-3_15

[60] Jan van Leeuwen. 1978. Effective constructions in well-partially-ordered free monoids. *Discr. Math.* 21, 3 (1978), 237–252. https://doi.org/10.1016/0012-365X(78)90156-5

[61] Herbert S Wilf. 1978. A circle-of-lights algorithm for the "money-changing problem". *Amer. Math. Monthly* 85, 7 (1978), 562–565.

[62] Georg Zetzsche. 2013. Silent transitions in automata with storage. In *Proc. ICALP 2013 (Lect. Notes Comput. Sci., Vol. 7966)*. Springer, 434–445. https://doi.org/10.1007/978-3-642-39212-2_39

[63] Georg Zetzsche. 2015. An approach to computing downward closures. In *Proc. ICALP 2015 (Lect. Notes Comput. Sci., Vol. 9135)*. Springer, 440–451. https://doi.org/10.1007/978-3-662-47666-6_35

[64] Georg Zetzsche. 2015. Computing downward closures for stacked counter automata. In *Proc. STACS 2015 (Leibniz Intl. Proc. Informatics, Vol. 30)*. LZI, 743–756. https://doi.org/10.4230/LIPICS.STACS.2015.743

[65] Georg Zetzsche. 2016. *Monoids as Storage Mechanisms*. Ph. D. Dissertation. Kaiserslautern University of Technology, Germany. https://kluedo.ub.rptu.de/frontdoor/index/index/docId/4400

[66] Georg Zetzsche. 2018. Separability by piecewise testable languages and downward closures beyond subwords. In *Proc. of LICS 2018*. ACM, 929–938. https://doi.org/10.1145/3209108.3209201

[67] Georg Zetzsche. 2021. The emptiness problem for valence automata over graph monoids. *Inf. Comput.* 277 (2021), 104583. https://doi.org/10.1016/J.IC.2020.104583

[68] Georg Zetzsche. 2021. Recent advances on reachability problems for valence systems (invited talk). In *Proc. RP 2021 (Lect. Notes Comput. Sci., Vol. 13035)*. Springer, 52–65. https://doi.org/10.1007/978-3-030-89716-1_4

A MINIMAL RUNS OF VASS ARE NOT COMPUTABLE

In this appendix, we prove that there is no algorithm to compute a basis for the set of runs of a (two-dimensional) VASS. Note that the run embedding for VASS is a partial ordering, so there is always a finite set of minimal runs, and computing a basis is equivalent to computing this set.

We use a reduction from reachability in two-counter machines. These are 2-VASS with zero tests meaning, they have two additional types of edge labels: zero_1 and zero_2, which test the first, resp. second, counter for zero, with the obvious semantics.

Given a two-counter machine (Q, q_0, Δ, q_f), where Q is the set of states, q_0 is the initial state, finite transition set $\Delta \subseteq Q \times (\mathbb{Z}^2 \cup \{\mathsf{zero}_1, \mathsf{zero}_2\}) \times Q$, and final state $q_f \in Q$, it is well-known to be undecidable whether there is a run from the configuration $(q_0, 0, 0)$ to $(q_f, 0, 0)$.

Given a two-counter machine as above, we define a 2-VASS with state set Q, initial state q_0, and final state q_f as follows. The input alphabet is $\Sigma = \{z_1, z_2\}$ and it has the following transitions:

$$
\begin{aligned}
&(p, \mathbf{u}, \varepsilon, q) && \text{for every transition } (p, \mathbf{u}, q) \text{ in } \mathcal{A}\\
&(p, (0,0), z_1, q) && \text{for every transition } (p, \mathsf{zero}_1, q) \text{ in } \mathcal{A}\\
&(p, (0,0), z_2, q) && \text{for every transition } (p, \mathsf{zero}_2, q) \text{ in } \mathcal{A}.
\end{aligned}
$$

Thus, the 2-VASS entirely ignores the zero tests, but it reads a letter z_i whenever the two-counter machine performs a zero test on counter i. We say that a run of the 2-VASS is *faithful* if all the transitions labelled z_1 or z_2 are actually executed in a configuration where the first, resp. second counter is zero. Then clearly, the two-counter machine has a run if and only if our 2-VASS has a faithful run.

Claim A.1. *The* 2*-VASS has a faithful run if and only if one of its minimal runs is faithful.*

If this is shown, it clearly follows that the minimal runs are not computable: Otherwise, we could compute them and check if one of them is faithful. The "if" direction is trivial, so suppose none of the minimal runs is faithful. Then each of the minimal runs contains a step of one of the following forms:

$$
\begin{aligned}
&((x_1, x_2), t, (x_1, x_2)) && \text{where } t \text{ is labelled by } z_1 \text{ and } x_1 > 0, \text{ or}\\
&((x_1, x_2), t, (x_1, x_2)) && \text{where } t \text{ is labelled by } z_2 \text{ and } x_2 > 0
\end{aligned}
$$

However, this implies that *every* run of our 2-VASS contains such a step. In particular, none of them can be faithful.

B RESULTS ON PRIORITY DOWNWARD CLOSURES

In this appendix, we prove the results about priority downward closures.

B.1 Overview

For the implication "(7) $\Rightarrow$ (6)" of Main Theorem A, we show that if emptiness is decidable, we can compute priority downward closures. To compute the priority downward closure of an input language L, we need to show that $L{\downarrow_{\mathrm{P}}} = D$ for some downward closure regular language D. The difficult part of this computation is to decide whether our input language L satisfies $D \subseteq L{\downarrow_{\mathrm{P}}}$. Our algorithm uses a strategy from [63], namely to decompose D into ideals. Somewhat more precisely, we do the above ideal decompositions not for the general priority ordering, but for the simple block order, which we define now.

The simple block order. To strip away some technicalities, we will work with a slightly different ordering for which downward closure computation is an equivalent problem. For each $d \in \mathbb{N}$, we define the alphabet $\Sigma_d = [0, d]$ and the *simple block ordering* over Σ_d^*. We think of Σ_d as an alphabet with priorities $[0, d]$, except that there is only one letter of each priority. If $u, v \in \Sigma_d^*$ and $m \in \Sigma_d$, $m > 0$, is the largest letter occurring in u and v, then we define $u \preccurlyeq_{\mathrm{S}} v$ if and only if

$$u = u_0 m u_1 \cdots m u_k,\ v = v_0 m v_1 \cdots m v_\ell$$

with $u_0, \ldots, u_k, v_0, \ldots, v_\ell \in \Sigma_{m-1}^*$ and there is a strictly monotone map $\varphi\colon [0,k] \to [0,\ell]$ with $u_i \preccurlyeq_S v_{\varphi(i)}$ for every $i \in [0,k]$. (In particular, $k = 0$ means that m does not occur in u) Thus, $\preccurlyeq_S$ is defined recursively w.r.t. the occurring priorities. To cover the base case, if $u, v \in 0^*$, then we simply have $u \preccurlyeq_S v$ if and only if $|u| \leq |v|$. For a word $w_1 m w_2 m \cdots m w_n m$, we call the words w_i which are enclosed between two consecutive m's, m-blocks. Thus intuitively, u is *simple block smaller* than v, if on splitting both words along the highest priority letter m, the m blocks from u are monotonically and recursively simple block smaller than those from v.

Example B.1. For $d = 1$, we have $0 \preccurlyeq_S 00 \not\preccurlyeq_S 010 \preccurlyeq_S 1010$, but $010 \not\preccurlyeq_S 1010$.

Then for $L \subseteq \Sigma_d^*$, we define as above

$$L{\downarrow_S} = \{u \in \Sigma_d^* \mid \exists v \in L\colon u \preccurlyeq_S v\}.$$

The task of computing an NFA for $L{\downarrow_S}$ for a given language L is called *computing simple block downward closures.*

In the following, we show that the simple block order is a rational relation.

Lemma B.2. *Let $\Sigma = [0,d]$, a rational transducer T can be constructed in polynomial time, such that for every language $L \subseteq \Sigma^*$, $L{\downarrow_S} = TL$.*

Proof. For the simple block order consider the transducer that has two states l_{out} and l_{skip} for every letter l, and a sink state t, and for every state p it reads a letter l, and for every letter l,

- if the state is l_{out}, it represents highest last outputted letter was l. From this state, there are transitions,
  - if the input letter is l, it stays at l, on outputting and skipping the letter.
  - otherwise, on reading a letter p, goes to p_{out} and p_{skip}, respectively on outputting and skipping the letter
- if the state is l_{skip}, it represents highest last skipped letter was l. From this state, there are transitions,
  - if the input letter is smaller, it stays at l_{skip} on skipping, and goes to t on outputting the letter.
  - if the input letter is l, then stays at l_{skip} on skipping, and goes to l_{out} on outputting the letter.
  - if the input letter p is greater, it goes to p_{skip} on skipping, and goes to p_{out} on outputting the letter.

The starting state is 0_{skip}, and any run that does not end at t is accepting.

Intuitively, the transducer makes sure that on skipping a letter l, every subsequent lower letter is skipped until a letter equal or greater is output. This ensures that two l-blocks are not merged, i.e., between two consecutive letters which are not dropped, no bigger priority letter is dropped.

We argue that the transducer restricted to the state set $S_k = \{l_{out}, l_{skip} | l \leq k\} \cup \{t\}$ outputs the set of words smaller than a word $w \in \Sigma_k$.

The base case is trivial, as there will be no edge to the sink state. Now suppose that the claim holds for some S_{k-1}. Then for S_k, let $u \in \Sigma_k$. Then for any word v which is small block smaller than u, the $(k-1)$-blocks in v that map to that of u, can be recognized by S_{k-1}, and every $(k-1)$-block that is skipped can be skipped by S_{k-1}, and after skipping, outputting another $(k-1)$-block can happen via k_{out} by outputting a k.

For any word v that is not simple block smaller than u, then consider the first three consecutive letters in u, xyz, such that $x, z < y$, x, z are output and y is not. Then the run will reach y_{skip} and next output letter will be smaller, leading to the sink state, hence the transducer will not output v. ◻

Lemma B.3. *If C is a full trio, then priority downward closures are computable for C if and only if simple block downward closures are computable for C.*

We will show this in Appendix B.2. In [2], the authors also introduce a "block order" (slightly different from our simple block order) and also show that downward closure computation of it is equivalent to that for the priority order.

Ideals. An *ideal* in a WQO $(X, \leq)$ is a downward closed set $I \subseteq X$ where for any $u, v \in I$, there is a $w \in I$ with $u \leq w$ and $v \leq w$. What makes these useful is that in a WQO, every downward closed set can be written as a finite union of ideals. Moreover, we will see that establishing $I \subseteq L{\downarrow_S}$ for the language L of an amalgamation system can be done by enumerating runs. To this end, we rely on a syntax for specifying ideals.

Lemma B.4. *The ideals of* $(\Sigma_d, \preccurlyeq_S)$ *are precisely the sets in* Ideal_d*, where*

$$\begin{aligned}\mathsf{Atom}_d &= \{Id \cup \varepsilon \mid I \in \mathsf{Ideal}_{d-1}\} \cup \{(Dd)^* \mid D \in \mathsf{Down}_{d-1}\},\\ \mathsf{Ideal}_0 &= \{0^n \mid n \in \mathbb{N}\} \cup \{0^*\},\\ \mathsf{Ideal}_d &= \{X_1 \cdots X_n A \mid X_i \in \mathsf{Atom}_d,\ A \in \mathsf{Ideal}_{d-1}\},\end{aligned}$$

and Down_d *is the set of all downward closed subsets of* Σ_d^* *with respect to simple block order.*

See Appendix B.2 for a proof. Now an algorithm for computing the simple block downward closure of a language $L \subseteq \Sigma_d^*$ can do the following. It enumerates finite unions $I_1 \cup \cdots \cup I_n$ of ideals $I_1, \ldots, I_n$. For each such union, it checks two inclusions: $L{\downarrow_S} \subseteq I_1 \cup \cdots \cup I_n$ and $I_1 \cup \cdots \cup I_n \subseteq L{\downarrow_S}$. The former inclusion is easy to check: Since $I_1 \cup \cdots \cup I_n$ is a regular language, we can just use decidable emptiness and closure under regular intersection to check whether $L{\downarrow_S} \cap \Sigma_d^* \setminus (I_1 \cup \cdots \cup I_n) = \emptyset$. The inclusion $I_1 \cup \cdots \cup I_n \subseteq L{\downarrow_S}$ is significantly harder to establish, and it will be the focus of the remainder of this subsection.

Pseudo-ideals. First, observe that it suffices to decide $I \subseteq L{\downarrow_S}$ for an individual ideal I. As the second step, we will simplify ideals even further to pseudo-ideals. A *pseudo-ideal of priority* 0 is an ideal of the form $0^{\leq n}$ or 0^*. A *pseudo-ideal of priority* $d > 0$ is an ideal of the form $(Id)^*$ or $I_1 d \cdots I_n d$, where I and $I_1, \ldots, I_n$ are pseudo-ideals of priority $d-1$. Thus, intuitively, we rule out subterms of the form $(Dd)^*$ with some downward-closed $D \subseteq \Sigma_{d-1}^*$. Despite being less expressive, deciding inclusion of pseudo-ideals is sufficient for inclusion of arbitrary ideals:

Proposition B.5. *Given an ideal* $I \subseteq \Sigma_d^*$ *and a class* $\mathcal{C}$ *of languages closed under rational transduction, we can construct finitely many pseudo-ideals* $J_j \subseteq \Sigma^*$ *and a rational transduction* T *such that for any language* $L \in \mathcal{C}$ *and* $L \subseteq \Sigma_d^*$*, we have* $I \subseteq L\downarrow_S$ *if and only if* $\cup_j J_j \subseteq TL\downarrow_S$*.*

This implies that if we have an algorithm to decide $J \subseteq L{\downarrow_S}$ for pseudo-ideals J, then this can even be done for arbitrary ideals. Essentially, the idea is to emulate subterms $I = (Dd)^*$ with $D = I_1 \cup \cdots \cup I_n$ by a new term $J = ((I_1 d \cdots I_n d)e)^*$, where $e > d$ is a fresh priority. The transduction modifies the words in L so that after every occurrence of d, an occurrence of e is potentially inserted. Note that then, in order for J to be included, each of the ideals $I_1, \ldots, I_n$ need to occur arbitrarily often, which corresponds to inclusion of $I = (Dd)^*$. We will show this in Appendix B.2.

Ideal inclusion via amalgamation. Let us now see how to establish an inclusion $I \subseteq L{\downarrow_S}$ for pseudo-ideals I using run embeddings. We begin with an example. Suppose we want to verify that the ideal $(0^*1)^*$ in included in our language. Here, we need to check if for every $k \geq 0$, there is a word with $\geq k$ factors, each containing $\geq k$ contiguous 0's, and they are separated by 1's. Intuitively, this is more complicated that the SUP and a proof using e.g. grammars seems involved. However, using run amalgamation, this amounts to checking a simple kind of witness: Namely, three runs $\rho_0 \trianglelefteq \rho_1 \trianglelefteq \rho_2$ such that (i) some gap of $\rho_0 \trianglelefteq \rho_1$ between some positions $i < j$ of ρ_1 contains a 1 and (ii) some gap of $\rho_1 \trianglelefteq \rho_2$, which is between i and j in ρ_1, belongs to 0^+. Notice that then, by amalgamating ρ_2 with itself above ρ_1 again and again, we can create arbitrarily long blocks of contiguous 0's. The resulting run ρ_2' still embeds ρ_1 and thus ρ_0, such that one gap of ρ_2' in ρ_0 contains both our long block of 0's and also a 1. This means, if we amalgamate ρ_2' again and again above ρ_0, we obtain arbitrarily many blocks of our long 0-blocks. This yields runs that cover all words in $(0^*1)^*$.

Witnesses for ideal inclusion. We will now see how inclusions $I \subseteq L{\downarrow_S}$ can always be verified by such run constellations, which we call "I-witnesses". For a word $w \in \Sigma \cup \{\varepsilon\}$, $w|_\Sigma$ denotes the restriction of w over Σ. We call a word w' a factor of $w = w_1 \cdots w_n \in \Sigma \cup \{\varepsilon\}$, if $w|_\Sigma \in \Sigma^* w' \Sigma^*$, i.e., w' is an infix of restriction of w over Σ. For a word $w = w_0 w_1 \cdots w_n$, by $w[i, j]$ we denote the word $w_i \cdots w_j$ for $i < j$.

A finite subset $\mathcal{W}$ of runs R is called an I-witness if

- for $I = 0^{\leq k}$ for some $k \in \mathbb{N}$, there exists is a run $\rho \in \mathcal{W}$ such that 0^k is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$, for some $0 \leq \overleftarrow{l}, \overrightarrow{l} \leq |\rho|_{\mathsf{can}}$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$.
- for $I = 0^*$, there exist runs $\rho, \psi \in \mathcal{W}$ such that $\psi \trianglelefteq_f \rho$ and $\mathrm{G}_{i,f} = 0^l$ for some $l > 0$, $i \in [0, |\psi|_{\mathsf{can}}]$ and $f \in E(\psi, \rho)$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$, if $\mathrm{G}_{i,f}$ is a factor of is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$.
- for $I = I_1 a \cdots a I_n a$, there exists a run $\rho \in \mathcal{W}$ and $\overleftarrow{l} = l_0 \leq l_1 < l_2 < \cdots < l_n \leq \overrightarrow{l}$ such that ρ is an I_i-witness between l_{i-1} and l_i, and $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}] \in \Sigma_a^*$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$.
- for $I = (I'a)^*$, there exist runs $\rho, \psi \in \mathcal{W}$ such that $\psi \trianglelefteq_f \rho$ and ρ is a witness for $(I'a)^l$ between $f(i)$ and $f(i+1)$, and $\rho[f(i), f(i+1)] \in \Sigma_a^*$, for some $l > 0$, $i \in [0, |\psi|_{\mathsf{can}} - 1]$ and $f \in E(\psi, \rho)$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$, if $\mathrm{G}_{i,f}$ is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$.

With this notion of I-witnesses, we can prove:

Lemma B.6. *For every pseudo-ideal I and every amalgamation system for $L{\downarrow}_{\mathrm{S}}$, we have $I \subseteq L{\downarrow}_{\mathrm{S}}$ if and only if the system possesses an I-witness.*

To illustrate the proof idea, let us see how to show that there is always a witness for $I = (0^*1)^*$. Suppose we have an amalgamation system S for $L{\downarrow}_{\mathrm{S}}$ and we know $I \subseteq L{\downarrow}_{\mathrm{S}}$. Let M be the maximal number of factors in the canonical decomposition of minimal runs of S. Consider the sequence $w_1, w_2, \ldots$ with $w_i = (0^i 1)^{2M}$ for $i \geq 1$. Then $w_1, w_2, \ldots \in I \subseteq L{\downarrow}_{\mathrm{S}}$, so there must be runs ρ_1 and ρ_2 with $\rho_1 \trianglelefteq \rho_2$ with $\mathsf{can}(\rho_1) = w_i$ and $\mathsf{can}(\rho_2) = w_j$. Then clearly, every non-empty gap of ρ_2 in ρ_1 belongs to 0^+. Moreover, any embedding of a minimal run ρ_0 of S into ρ_1 will have some gap containing two 1's, and thus have 10^i1 as a factor. Then the runs ρ_0, ρ_1, ρ_2 clearly constitute an I-witness. We will show this in Appendix B.2.

The algorithm. We now have all the ingredients to show that decidable emptiness in a class of amalgamation systems that forms a full trio implies computable priority downward closures. First, by Lemma B.3, it suffices to compute $L{\downarrow}_{\mathrm{S}}$ for a given language L. Second, according to Proposition B.5 and Lemma B.6, deciding whether $I \subseteq L{\downarrow}_{\mathrm{S}}$ for an ideal I is recursively enumerable. Therefore, we proceed as follows. We enumerate all finite unions $I_1 \cup \cdots \cup I_n$ of ideals and try to establish the inclusion $I_1 \cup \cdots \cup I_n \subseteq L{\downarrow}_{\mathrm{S}}$. Once we find such a finite union where inclusion holds, we check whether $L{\downarrow}_{\mathrm{S}} \subseteq I_1 \cup \cdots \cup I_n$. The latter is decidable: By closure under rational transductions, we can construct an amalgamation system for $L{\downarrow}_{\mathrm{S}} \cap (\Sigma^* \setminus (I_1 \cup \cdots \cup I_n))$ and check it for emptiness. Since we know that for every downward-closed set, there exists a finite union of ideals, our algorithm will eventually discover a finite union $I_1 \cup \cdots \cup I_n$ with $L{\downarrow}_{\mathrm{S}} = I_1 \cup \cdots \cup I_n$.

Finally, note that "(6) $\Rightarrow$ (7)" holds as well, because $L \neq \emptyset$ if and only if $L{\downarrow}_{\mathrm{P}} \neq \emptyset$, meaning we decide emptiness of L by computing an NFA for $L{\downarrow}_{\mathrm{P}}$ and check that for emptiness.

B.2 Detailed proofs

B.2.1 Equivalence of Simple Block Order and Priority Order.

Lemma B.3. *If C is a full trio, then priority downward closures are computable for C if and only if simple block downward closures are computable for C.*

Proof. In [2], it was shown that priority downward closure can be computed if and only if block downward closure can be computed, where we say $u \preccurlyeq_{\mathrm{B}} v$, if

i. if $u, v \in \Sigma_{=p}^*$, and $u \preceq v$ (u is subword smaller than v), or

ii. if

$$\begin{aligned} u &= u_0 x_0 u_1 x_1 \cdots x_{n-1} u_n \\ \text{and, } v &= v_0 y_0 v_1 y_1 \cdots y_{m-1} v_m \end{aligned}$$

where $x_0, \ldots x_{n-1}, y_0, \ldots, y_{m-1} \in \Sigma_{=p}$, and for all $i \in [0, n]$, we have $u_i, v_i \in \Sigma^*_{\leq p-1}$ (the u_i and v_i are called p blocks), and there exists a strictly monotonically increasing map $\varphi : [0, n] \to [0, m]$, which we call the *witness block map*, such that

(a) $u_i \preccurlyeq_{\mathrm{B}} v_{\varphi(i)}, \forall i$,
(b) $\varphi(0) = 0$,
(c) $\varphi(n) = m$, and
(d) $x_i \preceq v_{\varphi(i)} y_{\varphi(i)} v_{\varphi(i)+1} \cdots v_{\varphi(i+1)}, \forall i \in [0, n-1]$.

Intuitively, we say that u is *block smaller* than v, if either

- both words have letters of same priority, and u is a subword of v, or,
- the largest priority occurring in both words is p. Then we split both words along the priority p letters, to obtain sequences of p blocks of words, which have words of strictly less priority. Then by item iia, we embed the p blocks of u to those of v, such that they are recursively block smaller. Then with items iib and iic, we ensure that the first (and last) p block of u is embedded in the first (resp., last) p block of v. We will see later that this constraint allows the order to be multiplicative. Finally, by item iid, we ensure that the letters of priority p in u are preserved in v, i.e. every x_i indeed occurs between the embeddings of the p block u_i and u_{i+1}.

Then we now show that block downward closures can be computed if and only if simple block downward closures can be computed.

Since $u \preccurlyeq_{\mathrm{B}} v \implies u \preccurlyeq_{\mathrm{S}} v$, it is trivial that $L \downarrow_S$ is computable if $L \downarrow_B$ is computable.

For the other direction, suppose that the alphabet is $\Sigma = [0, d]$. Then we consider a new alphabet $\Sigma' = \{0, 0', 0^\varepsilon, \ldots d, d', d^\varepsilon\}$ such that $0 < 0' < 0^\varepsilon < 1 < 1' < 1^\varepsilon < \cdots < d < d' < d^\varepsilon$. By $\mathcal{B}$, we denote $\Sigma' \backslash \Sigma$. Let for a word $w = w_1 a w_2 a \cdots a w_n$, where $w_i \in \Sigma^*_{a-1}$ and a is the highest priority letter in w, by $border(w)$ we define the word $b_1 \cdot border(w_1) \cdot b_1 \cdot a \cdot border(w_2) \cdot a \cdots a \cdot border(w_n - 1) \cdot a \cdot b_n \cdot border(w_n) \cdot b_n$, where b_1 (and b_n) is $(a-1)^\varepsilon$ if $w_1 = \varepsilon$ (resp., $w_n = \varepsilon$), else it is $(a-1)'$. Moreover, $border(w) = w$ if $w \in 0^*$. Intuitively, we bound the first and the last a blocks of the word recursively, and we also distinguish whether these blocks are ε or not.

Let $border(L) = \{border(w) \mid w \in L\}$. Then we claim that $L \downarrow_B = (border(L) \downarrow_S \cap \mathfrak{R}_d)|_\Sigma$, where $\mathfrak{R}_d$ is defined recursively as follows.

$$\begin{aligned} \mathfrak{R}_0 &= 0^* \\ \mathfrak{R}_a &= (a^\varepsilon \varepsilon a^\varepsilon + a' \mathfrak{R}_{a-1} a') \cdot (a \mathfrak{R}_{a-1})^* a \cdot (a^\varepsilon \varepsilon a^\varepsilon + a' \mathfrak{R}_{a-1} a') \end{aligned}$$

Note that $L' = border(L) \downarrow_S \cap \mathfrak{R}_d$ is the language of simple block smaller words where $0', 0^\varepsilon \cdots d', d^\varepsilon$ are not dropped. Furthermore, the $L|_\Sigma$ is restriction of words in L to letters in Σ.

Now we prove the claim. For one direction suppose $u \in L \downarrow_B$. Then there exists a word $v \in L$ such that $u \preccurlyeq_{\mathrm{B}} v$. It suffices to show by induction on the maximum priority letter d in u and v (which has to be the same by definition of block order) that $border(u) \preccurlyeq_{\mathrm{S}} border(v)$. For the base case, i.e. $d = 0$, since $border(u) = u$ for any $u \in 0^*$, this trivially holds.

Then for the induction step, let the statement holds true for some $d-1 \in \mathbb{N}$. Then assuming $u = u_1 d u_2 \cdots d u_k$ and $v = v_1 d v_2 \cdots d v_l$, there exists a $\preccurlyeq_{\mathrm{B}}$ witness $\varphi : [1, k] \to [1, l]$, with $\varphi(1) = 1$ and $\varphi(k) = l$. Then we show that φ is also a witness for $border(u) \preccurlyeq_{\mathrm{S}} border(v)$. Let $border(u) = u'_1 d u'_2 \cdots d u'_k$ and $border(v) = v'_1 d v'_2 \cdots d v'_l$. Induction hypothesis immediately implies that $u'_i \preccurlyeq_{\mathrm{S}} v'_{\varphi(i)}$ for $i \in [2; k-1]$. We argue that $u'_1 \preccurlyeq_{\mathrm{S}} v'_{\varphi(1)} = v'_1$ (argument is analogous for the last d block). If $u'_1 = d^\varepsilon \varepsilon d^\varepsilon$, then $v'_1 = d^\varepsilon \varepsilon d^\varepsilon$, since ε is only block smaller than ε. Then clearly, $u'_1 \preccurlyeq_{\mathrm{S}} v'_1$. Now if $u'_1 = d' x d'$, where $x \in \mathfrak{R}_{a-1}$, then $v'_1 = d' y d'$, where $y \in \mathfrak{R}_{a-1}$. Again by induction hypothesis, $x \preccurlyeq_{\mathrm{S}} y$, and hence $u'_1 \preccurlyeq_{\mathrm{S}} v'_1$. Hence, $border(u) \preccurlyeq_{\mathrm{S}} border(v)$.

Now for the other direction, we suppose that $u \in border(L) \downarrow_S \cap \mathfrak{R}_d$. Then there exist $u' \in \Sigma^*$, and $v, v' \in \Sigma'^*$, such that

- $v' = border(u'), v = border(u)$

- $v \preccurlyeq_S v'$, and
- $v'|_{\mathcal{B}} = v|_{\mathcal{B}}$, i.e. no bordering letters are dropped.

Then we need to show that $u \preccurlyeq_B u'$: then since $u' \in L$ (by definition of u) implying that $u \in L\downarrow_B$. In the rest of the proof we will show that $u' \preccurlyeq_B u$. Since $v \preccurlyeq_S v'$, there exists a witness φ. We show that φ is a witness for $u \preccurlyeq_B u$, by induction on d. The base case is trivial as $border(x) = x$ when $x \in 0^*$. Now, for the induction step let the highest letter in u and u' is d.

We show that the first (analogously, last) d block of v maps to that of v', i.e. $\varphi(1) = 1$. Since v and v' are bordered words, the borders d' and d^{ε} are added in the first and the last blocks of both words. Then the first block of v must be mapped to that of v' to map these borders, i.e. $\varphi(1) = 1$. Then assuming $v = v_1 d \cdots dv_k$ and $v' = v'_1 d \cdots dv'_l$, we have that $v_1 \preccurlyeq_S v'_1$, then $v_1|_\Sigma \preccurlyeq_B v'_1|_\Sigma$. Similarly, $v_k|_\Sigma \preccurlyeq_B v'_l|_\Sigma$. For other block, the induction hypothesis, immediately implies $v_i \preccurlyeq_B v_{\varphi(i)}$. Then φ is a witness for $u \preccurlyeq_B u'$. Hence, $u \in L\downarrow_B$. This completes the proof of the lemma. □

B.2.2 Simple Block Order Ideals.

Upward closure. Let $(X, \leq)$ be a WQO. Let $Y \subseteq X$, then upward closure of Y, denoted as $Y\uparrow_{\leq}$, is defined as the set of all the elements in X which are bigger than an element in Y, i.e., $Y\uparrow_{\leq} := \{x \in X | \exists y \in Y, y \leq x\}$. For the purpose of this appendix, we will mean $Y\uparrow_{\leq}$ with $Y\uparrow$.

We now show that the downward closed sets are finite union of ideals which are defined as follows.

Ideals. Let $(X, \leq)$ be a WQO. A subset I of X is called an ideal, if it is

- downward closed, i.e. if $u \in I$ and $v \leq u$, then $v \in I$, and
- up-directed, i.e. $\forall\, u, v \in I, \exists\, z \in I$ such that $u \leq z$ and $v \leq z$.

Lemma B.7. *Let $\mathcal{J}$ be a subset of ideals, such that*

(1) the complement of any filter can be written as a finite union of ideals from $\mathcal{J}$, and
(2) the intersection of any two ideals of $\mathcal{J}$ is a finite union of ideals from $\mathcal{J}$.

Then $\mathcal{J}$ is the set of all the ideals.

Lemma B.4. *The ideals of $(\Sigma_d, \preceq)$ are precisely the sets in Ideal_d, where*

$$\begin{aligned}\mathsf{Atom}_d &= \{Id \cup \varepsilon \mid I \in \mathsf{Ideal}_{d-1}\} \cup \{(Dd)^* \mid D \in \mathsf{Down}_{d-1}\},\\ \mathsf{Ideal}_0 &= \{0^n \mid n \in \mathbb{N}\} \cup \{0^*\},\\ \mathsf{Ideal}_d &= \{X_1 \cdots X_n A \mid X_i \in \mathsf{Atom}_d,\ A \in \mathsf{Ideal}_{d-1}\},\end{aligned}$$

and Down_d is the set of all downward closed subsets of Σ_d^* with respect to simple block order.

Proof. Let $Ideals_a$ be the set of all the ideals with respect to simple block order for alphabet $\Sigma_a = [0, a]$. We first show that every set in Id_d is an ideal, i.e., $Id_d \subseteq Ideals_d$. Observe that the elements of Id_0 are indeed ideals, because for singletons, the subword order and the simple block order coincide. So, now assume that $Id_{a-1} = Ideals_{a-1}$.

We now show that $Id_a \subseteq Ideals_a$. Let $X = (X_1 \cdots X_n A) \in Id_a$. We need to show that X is downward closed and up-directed.

Downward closed. Suppose $u = u_1 u_2 \cdots u_n u_A \in X$ such that $u_i \in X_i$ and $u_A \in A$, and let $v = v_1 a v_2 a \cdots a v_k$ with $v \preccurlyeq_S u$. Then there exists a strictly monotonically increasing map φ that maps a blocks of v to those of u. Then consider the map $\psi : [1, k] \to \{1, \ldots, n, A\}$ that maps each a block v_i to j if $\varphi(i)$-th a block of u lies in u_j. Note that this map is well defined as each u_i terminates with an a, so no a block of v can be mapped to a a block of u that splits between two u_i's. Then each $v_i a \cdots a v_j a$ such that $\psi(i) = \psi(j) = p$ is in X_p, by induction hypothesis. Now suppose $S = \{i | \psi(j) = i,$ for some $j\}$. Then $v \in X_{i_1} \cdots X_{i_q} \subseteq X_1 \cdots X_n A$, where $S = \{i_1, \ldots i_q\}$.

Upward directed. Let $u, v \in X$. Suppose $u = u_1 \dots u_n u_A$ and $v = v_1 \dots v_n v_A$, such that $u_i, v_i \in X_i$ and $u_A, v_A \in A$. Construct the word $z = z_1 \cdots z_n z_A$ as follows

$$\begin{aligned} z_i &= \begin{cases} u_i v_i, & \text{if } X_i = (Da)^* \\ w_i, & \text{if } X_i = Ia \text{ and } u_i, v_i \preccurlyeq_{\mathrm{B}} w_i \end{cases} \\ z_A &= w_A, \text{ where } u_A, v_A \preccurlyeq_{\mathrm{B}} w_A \end{aligned}$$

It is again easy to notice that $u, v \preccurlyeq_{\mathrm{B}} z$. Hence, X is up-directed.

We have shown above that $Id_d \subseteq Ideals_d$. To show that $Id_d = Ideals_d$, we use the Lemma B.4, and show that Id_d satisfies the two preconditions of the lemma.

- (Complement of a filter is a finite union of Id_a ideals) Let $u \in \Sigma_a^*$, and we need to show that $\Sigma_a^* \backslash u \uparrow$ is a finite union of ideals from Id_a. The proof is by an induction on the number of a blocks in u.
  When u has only 1 a block, $u \in \Sigma_{a-1}^*$. Then $\Sigma_a^* \backslash u \uparrow = \Sigma_{a-1}^* \backslash u \uparrow = \cup_{i=1}^k I_i$, where $I_i \in Id_{a-1} \subseteq Id_a$ and $k \in \mathbb{N}$. Hence, the base case holds. For the induction hypothesis, assume that the required result holds when u has $n-1$ a blocks.
  Let u has n many a blocks. Then u can be written as vaw where $v \in \Sigma_{a-1}^*$ and w has $n-1$ many a blocks.

  Claim B.8.
  $$\begin{aligned} \Sigma_a^* \backslash\, vaw \uparrow &= \big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* \backslash\, v \uparrow) \\ &\quad \cup \Big(\big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* a)(\Sigma_a^* \backslash\, w \uparrow)\Big) \downarrow_S \end{aligned}$$

  Proof of Claim B.8. ($\subseteq$) Let $z \in \Sigma_a^* \backslash\, vaw \uparrow$. Then $u = vaw \not\preccurlyeq_{\mathrm{S}} z$, and we have the following two cases,
  1. Case 1: If v can not be mapped to a a block of z, then all the a blocks of z come from $(\Sigma_{a-1}^* \backslash\, v \uparrow)$. Hence,
  $$z \in \big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* \backslash\, v \uparrow) \subseteq RHS.$$
  2. Case 2: If v can be mapped to a a block of z then let this be the i^{th} block of $z = z_1 a z_2 a \cdots a z_k$. Hence, the first $i-1$ blocks come from $(\Sigma_{a-1}^* \backslash\, v \uparrow)$, and the i^{th} block comes from Σ_{a-1}^*. This implies that $z_1 a z_2 a \cdots a z_i a \in \big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* a)$.
  Since $u \not\preccurlyeq_{\mathrm{S}} z$ and $va \preccurlyeq_{\mathrm{S}} z_1 a z_2 a \cdots z_i a$, $w \not\preccurlyeq_{\mathrm{S}} z_{i+1} a \cdots a z_n$, i.e. $w \in (\Sigma_a^* \backslash\, w \uparrow)$. Hence,
  $$z \in \big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* a)(\Sigma_a^* \backslash\, w \uparrow) \subseteq RHS.$$

  Hence, $LHS \subseteq RHS$.
  ($\supseteq$) This containment can be seen by going backwards in the arguments for the other containment. ◻

  Since $(\Sigma_{a-1}^* \backslash\, v \uparrow)$ is a finite union of ideals from Id_{a-1},
  $$\begin{aligned} \big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* \backslash\, v \uparrow) &= (\cup_i I_i) a (\cup_j I_j) \\ &= \bigcup_{i,j} I_i a I_j \end{aligned}$$
  where $I_i, I_j \in Id_{a-1}$. Since $I_i a I_j$ is an element in Id_a, $\Sigma_{a-1}^* \backslash\, v \uparrow$ is a finite union of ideals from Id_{a-1}.
  Using similar arguments, it can be shown that
  $\big((\Sigma_{a-1}^* \backslash\, v \uparrow) a\big)^* (\Sigma_{a-1}^* a)(\Sigma_a^* \backslash\, w \uparrow)$ is also a finite union of ideals from Id_a.
  Hence, the complement of a filter is a finite union of Id_a ideals.
- (Intersection of Id_a ideals is a finite union of Id_a ideals) We prove a stronger property, where ideals are defined over $Atom'_a = Atom_a \cup Ia | I \cap Id_{a-1}$. When $\Sigma = \{0\}$, the ideals are of the form 0^* or $\{0^i | 0 \leq i \leq n\}$ for some n, and the intersection of two ideals can only be another ideal.
  Suppose that the statement holds for some Σ_{a-1}. Then let $I_1 = X_1 X_2 \cdots X_k A$ and $I_2 = Y_1 Y_2 \cdots Y_l B$ be two ideals from Id_a. We then show by induction on the sum of the number of atoms in each ideal, i.e. $s = k + l$. The base cases are:

(1) when $s = 0$: then $I_1 = A \in Id_{a-1}$ and $I_2 = B \in Id_{a-1}$, then by induction hypothesis, $I_1 \cap I_2$ a finite union of ideals.
(2) when $s = 1$: then $I_1 = A \in Id_{a-1}$ and $I_2 = X_1 B$ where $X_1 \in Atom'_a$ and $B \in Id_{a-1}$. Then $I_1 \cap I_2 = (A \cap B) \cup (A \cap B_1)$, where $X_1 = B_1 a$ or $X_1 = (B_1 a)^*$. But both intersections are finite union of ideals.

Then suppose the statement holds for all ideals $I_1 = X_1 X_2 \cdots X_k A$ and $I_2 = Y_1 Y_2 \cdots Y_l B$, i.e. for some $s = k + l$. Now suppose the sum of numbers of atoms in ideals be $s + 1$. Then without loss of generality let $I_1 = X_1 X_2 \cdots X_k X_{k+1} A$ and $I_2 = Y_1 Y_2 \cdots Y_l B$. To reduce notational clutter, we write $I_1 = X_1 X A$ and $I_2 = Y_1 Y B$, with canonical X and Y.

Then we show that $I_1 \cap I_2$ is a finite union of ideals. We consider the following cases, depending on the types of atoms X_1 and Y_1:
(1) if $X_1 = A_1 a$ and $Y_1 = B_1 a$,
then $I_1 \cap I_2 = (A_1 \cap B_1) a (XA \cap YB)$
(2) if $X_1 = A_1 a$ and $Y_1 = (B_1 a \cup \{\varepsilon\})$,
then $I_1 \cap I_2 = (A_1 a XA \cap B_1 a YA) \cup (A_1 a XA \cap YB) = ((A_1 \cap B_1) a (XA \cap YB)) \cup (A_1 a XA \cap YB)$,
(3) if $X_1 = A_1 a$ and $Y_1 = (B_1 a)^*$,
then $I_1 \cap I_2 = I_1 \cap (YB \cup B_1 a I_2) = (I_1 \cap YB) \cup (I_1 \cap B_1 a I_2) = (I_1 \cap YB) \cup ((A_1 \cap B_1) a (XA \cap I_2))$,
(4) if $X_1 = (A_1 a \cup \{\varepsilon\})$ and $Y_1 = (B_1 a \cup \{\varepsilon\})$,
then $I_1 \cap I_2 = [(A_1 \cap B_1) a (XA \cap YB) \cup (A_1 a XA \cap YB) \cup (XA \cap B_1 a YB) \cup (XA \cap YB)]$,
(5) if $X_1 = (A_1 a \cup \{\varepsilon\})$ and $Y_1 = (B_1 a)^*$,
then $I_1 \cap I_2 = (I_1 \cap YB) \cup ((A_1 \cap B_1) a (XA \cap I_2)) \cup (XA \cap I_2)$,
(6) if $X_1 = (A_1 a)^*$ and $Y_1 = (B_1 a)^*$,
then $I_1 \cap I_2 = (XA \cup A_1 a I_1) \cap (YB \cup B_1 a I_2) = (XA \cap YB) \cup (XA \cap B_1 a I_2) \cup (A_1 a I_1 \cap YB) \cup (A_1 a I_1 \cap B_1 a I_2)$.

The equalities above follow from basic distributivity of unions and intersections. Since each intersection is between ideals with sum of atoms at most s, then using the induction hypothesis, we have that $I_1 \cap I_2$ are finite union of ideals.

Hence, by Lemma B.7, we have that $Id_a = Ideals_a$. □

B.2.3 From ideals to pseudo-ideals.

Proposition B.5. *Given an ideal $I \subseteq \Sigma_d^*$ and a class C of languages closed under rational transduction, we can construct finitely many pseudo-ideals $J_j \subseteq \Sigma^*$ and a rational transduction T such that for any language $L \in C$ and $L \subseteq \Sigma_d^*$, we have $I \subseteq L\downarrow_S$ if and only if $\cup_j J_j \subseteq TL\downarrow_S$.*

Proof. Since pseudo-ideals are special cases of general ideals, one direction is trivial. For the other direction, suppose that the containment is decidable for pseudo-ideals.

From ideals to flat ideals. We call an ideal $I = X_1 X_2 \cdots X_n A$ flat, if X_i is of the form Id or $(I_1 d \cdots d I_k d)^*$. We first show that for a general ideal I, there exist flat ideals J_j, such that $I \subseteq L\downarrow_S$ if and only if $\cup_j J_j \subseteq L'\downarrow_S$, for some language L'.

Let $I = X_1 \cdots X_n A$ be an ideal, where $X_i \in Atom_d$, $A \in Id_{d-1}$. First, since some atoms are of the form $(I'_d \cup \varepsilon)$, by distributivity of concatenation over union, we have that I is finite union of sets J_j of the form $Y_1 \cdot Y_k A$, where $k \leq n$ and Y_i's are of form $I'd$ or $(Dd)*$. Then $I \subseteq L\downarrow_S$ if and only if $J_j \subseteq L\downarrow_S$ for all j. For each J_j, we give construct a flat ideal. So for simplicity of notation, we assume that X_i's are of the form $I'd$ or $(Dd)^*$. Moreover, since $I \subseteq L\downarrow_S$ if and only if $Id \subseteq Ld\downarrow_S$, we may also assume that $I = X_1 \cdots X_n$.

Since every downward closed set is a finite union of ideals, then if $X_i = (Dd)^*$, then we may replace X_i with $(I_1 d I_2 d \ldots I_k d)^*$, where $D = \cup_{j \in [1,k]} I_j$ is the ideal decomposition of D. Suppose we obtain I' by replacing such X_is, then it is easy to see that $I \subseteq L\downarrow_S$ if and only if $I' \subseteq L\downarrow_S$. So we may assume that X_i is of the form Id or $(I_1 d \ldots d I_k d)^*$.

Now consider the alphabet $\Sigma'_d = \{i, \bar{i} | i \in \Sigma\}$, such that $0 < \bar{0} < 1 < \cdots < d < \bar{d}$. Consider a transducer T_i that arbitrarily adds a $\bar{i}$ after an occurrence of i. Then consider the ideal

$I' = X_1 \cdot \overline{d} \dots \overline{d} \cdot X_n \cdot \overline{d}$. And consider the language $L' = T_1 \cdots T_a L$. Since C is closed under rational transduction, $L' \in C$.

By induction on highest letter a in I, we will show that $I \subseteq L \downarrow_S$ if and only if $I' \subseteq L' \downarrow_S$.

For the base case, suppose $a = 1$. Then let $I \subseteq L \downarrow_S$, and $u = u_1\overline{1} \cdots u_k\overline{1} \in I'$, where $u_i \in §_1$. Then $u' = u_1 \cdots u_k \in I$, and there exists $v' \in L$ such that u' embeds in a 2-block of v' with witness map φ. Then on adding a $\overline{1}$ after every 1-block where the last 1-block of each u_i embeds to, to obtain v, we get that $u \preccurlyeq_S v$.

For the reverse direction, let $I' \subseteq L' \downarrow_S$ and $u = u_1 \cdots u_k \in I$, where $u_i \in X_i$. Then $v = u_1\overline{1} \cdots \overline{1}u_k \in I'$, and hence there exists $v' = v_1\overline{1} \cdots \overline{1}v_l \in L'$ such that $v \preceq v'$. Then since each $\overline{1}$ block in v recursively embeds in $\overline{1}$ blocks in v', the 1-blocks recursively embed too. Then $u' = v_1 v_2 \cdots v_l \in L$ and $u \preceq u'$.

Then we assume that for some $a-1$, $I \subseteq L \downarrow_S$ if and only if $I' \subseteq L' \downarrow_S$, where $L' = T_1 \cdots T_{a-1}L$. Now suppose highest letter in I is a, and $L' = T_a \cdots T_1 L$. Also, let $I' = X_1 \cdot \overline{a} \dots \overline{a} \cdot X_n \cdot \overline{a}$. We first show that $I \subseteq L \downarrow_S$ iff $I' \subseteq T_a L \downarrow_S$.

First, let $I \subseteq L \downarrow_S$, and $u = u_1\overline{d} \cdots \overline{d}u_k \in I'$, where $u_i \in X_i$. Then $v = u_1 u_2 \cdots u_k \in I$. Then there exists $v' \in L$ such that and $v \preceq v'$, with a witness map φ. Consider $u' = v_1\overline{d} \cdots \overline{d}v_k$ such that $v_i \in \Sigma_d \cdot w_i \cdot d$ where w_i is the $\varphi(i)$-th d-block of u. Note that $v' \in T_a L$ and $u \preceq u'$, since $u_i \preceq v_i$.

Now, suppose that $I' \subseteq T_a L \downarrow_S$. Then let $u = u_1 \cdots u_k \in I$, where $u_i \in X_i$. Then $v = u_1\overline{d} \cdots \overline{d}u_k \in I'$, and hence there exists $v' = v_1\overline{d} \cdots \overline{d}v_l \in T_a L$ such that $v \preceq v'$. Then since each $\overline{d}$ block in v recursively embeds in $\overline{d}$ blocks in v', the d-blocks recursively embed too. Then $u' = v_1 v_2 \cdots v_l \in L$ and $u \preceq u'$.

Since X_i's are ideals enclosed between $\overline{d}$'s in I', then by induction hypothesis $X_i \subseteq T_a L \downarrow_S$ iff $X_i' \subseteq T_1 \cdots T_{a-1} T_a L \downarrow_S$, where X_i' is a flat ideal obtained by eliminating the downward closed sets in the Kleene stars. Hence, $I \subseteq L \downarrow_S$ if and only if $I'' = X_1'\overline{d} \cdots X_n'\overline{d} \subseteq T_1 \cdots T_{a-1} T_a L \downarrow_S$.

From flat ideals to pseudo-ideals. Since flat ideals are almost in form of pseudo-ideals, except only when X_i is of the form $(I_1 d I_2 d \cdots I_k d)^*$, for the simplicity of the proof, we show how we reduce from $(I_1 d I_2 d \cdots I_k d)^*$ to $(Id)^*$. The generalization is simple extension.

Consider the alphabet $\Sigma_d' = \{\underline{i}, i, \overline{i} | i \in \Sigma\}$, such that $\underline{0} < 0 < \overline{0} < \underline{1} < 1 < \cdots < \underline{d} < d < \overline{d}$. Consider a transducer T_i that arbitrarily adds a $\overline{i}$ after an occurrence of i, and a transducer T_i' that in arbitrary $\overline{i}$-blocks, replaces every i with an $\underline{i}$, and adds a i after the final replacement in the $\overline{i}$-block. For example, on a word 0101010, one of the outputs of T_1 is $01\overline{1}0101\overline{1}0$, and on this word, T_1' outputs $01\overline{1}0\underline{1}01\underline{1}\overline{1}0$.

Let $X = (I_1 d I_2 d \cdots I_k d)^*$, where I_i's are flat ideals over Σ_{d-1}, then consider $X' = (I_1 \underline{d} I_2 \cdots \underline{d} I_k \underline{d} d)^*$. Note that X' is a pseudo-ideal. Then we show by induction over the highest letter a in flat ideals, that $X \subseteq L \downarrow_S$ if and only if $X' \subseteq T_a' T_{a-1}' \cdot T_1' L \downarrow_S$.

For the base case, when $a = 1$, then $X = (I_1 1 \cdots 1 I_k 1)^*$, where I_i is either 0^* or $0^{\leq k}$, and $L' = (I_1 \underline{1} \cdots \underline{1} I_k \underline{1} 1)^*$. If $X \subseteq L \downarrow$, and $u = u_1 1 \cdots u_n 1 \in X'$, where $u_i \in \Sigma_1^* \underline{1}$. Then $u' = u_1' 1 \cdots u_n' 1$, such that every $\underline{1}$ is replaced with 1, and the last $\underline{1}$ before every 1 is dropped. Then $u' \in I$. Then there is a word $v' = v_1' v_2' v_3'$, where v_2' is a 2-block such that $u' \preccurlyeq_S v_2'$, with a witness map φ. Let's say u_i has s_i many $\underline{1}$-blocks. Then on replacing all the 1's with $\underline{1}$'s in v_2', except the ones that appear at $s_1 + s_2 + \ldots + s_i$-th 1's for every i, and adding $\underline{1}$, before them, we get a word v. Then it is easy to observe that $u \preccurlyeq_S v$, since u embeds within v_2: map i-th 1-block of u to i-th 1-block of v_2, and recursively map $\underline{1}$-blocks respecting φ.

If $I' \subseteq T_1' L \downarrow_S$ then going backward in the argument above, we get that $I \subseteq T_1' L \downarrow_S$.

Then for some a, the argument is similar with induction hypothesis over flat ideals of smaller highest letter, with the observation that adding, removing, and replacing $\underline{i}$ as per T_i', preserves the blocks of $(i-1)$'s, and never splits them, which can continue to embed respecting their original embedding. Also, observe now that X' is a pseudo-ideal.

Now to see this generalizes to any flat ideals, we observe that flat ideals are of the form $X_1 X_2 \cdots X_n$ where X_i are of the form $Id\overline{d}$ or $(I_1 d I_2 d \cdots I_x d)^* \overline{d}$, i.e. every X_i is enclosed with highest priority letter in the ideal. Then within each $\overline{d}$-block the embedding is respected in the transformation from flat ideals to pseudo-ideals. So, we just replace X_i with X_i' as defined above, and apply $T_a' \cdots T_1'$ to $T_a \cdots T_1 L$.

The two transformation reduce the containment problem of general ideals in a downward closed language to that for psuedo-ideals (via flat ideals).

□

B.2.4 Proof that pseudo-ideal is contained in downward closure if and only if I-witness exists. A finite subset $\mathcal{W}$ of runs R is called an I-witness if

- for $I = 0^{\leq k}$ for some $k \in \mathbb{N}$, there exists is a run $\rho \in \mathcal{W}$ such that 0^k is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$, for some $0 \leq \overleftarrow{l}, \overrightarrow{l} \leq |\rho|_{\mathsf{can}}$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$.
- for $I = 0^*$, there exist runs $\rho, \psi \in \mathcal{W}$ such that $\psi \trianglelefteq_f \rho$ and $\mathrm{G}_{i,f} = 0^l$ for some $l > 0$, $i \in [0, |\psi|_{\mathsf{can}}]$ and $f \in E(\psi, \rho)$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$, if $\mathrm{G}_{i,f}$ is a factor of is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$.
- for $I = I_1 a \cdots a I_n a$, there exists a run $\rho \in \mathcal{W}$ and $\overleftarrow{l} = l_0 \leq l_1 < l_2 < \cdots < l_n \leq \overrightarrow{l}$ such that ρ is an I_i-witness between l_{i-1} and l_i, and $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}] \in \Sigma_a^*$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$.
- for $I = (I'a)^*$, there exist runs $\rho, \psi \in \mathcal{W}$ such that $\psi \trianglelefteq_f \rho$ and ρ is a witness for $(I'a)^l$ between $f(i)$ and $f(i+1)$, and $\rho[f(i), f(i+1)] \in \Sigma_a^*$, for some $l > 0$, $i \in [0, |\psi|_{\mathsf{can}} - 1]$ and $f \in E(\psi, \rho)$.
  Then ρ is said to witness I between $\overleftarrow{l}$ and $\overrightarrow{l}$, if $\mathrm{G}_{i,f}$ is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$.

For a word $w \in \Sigma \cup \{\varepsilon\}$, $w|_\Sigma$ denotes the restriction of w over Σ. We call a word w' a factor of $w = w_1 \cdots w_n \in \Sigma \cup \{\varepsilon\}$, if $w|_\Sigma \in \Sigma^* w' \Sigma^*$, i.e., w' is an infix of restriction of w over Σ. For a word $w = w_0 w_1 \cdots w_n$, by $w[i, j]$ we denote the word $w_i \cdots w_j$ for $i < j$. Given a set of runs S, by the amalgamation closure of S we mean the set of runs that can be produced by amalgamating runs in S.

Lemma B.6. *For every pseudo-ideal I and every amalgamation system for $L{\downarrow_S}$, we have $I \subseteq L{\downarrow_S}$ if and only if the system possesses an I-witness.*

Proof. Let I be a pseudo-ideal over $\Sigma = \Sigma_a$. Suppose $I \subseteq L\downarrow_S$.

- If $I = 0^{\leq k}$, then for $u = 0^k$ there exists a run ρ in the system recognizing L, such that $u \preccurlyeq_S \mathsf{yield}(\rho)|_\Sigma = v$. Then 0^k is a factor of $\mathsf{can}(\rho)[\overleftarrow{l}, \overrightarrow{l}]$ for some $\overleftarrow{l}$ and $\overrightarrow{l}$. Hence, $\{\rho\}$ is an I-witness between 0 and $|\rho|_{\mathsf{can}}$.
- If $I = 0^*$, then suppose there is no I-witness, i.e. for every pair of runs $\psi \trianglelefteq_f \rho$, every gap word is either ε^* or it contains a letter $p > 0$. If every gap word is ε^*, clearly $I \not\subseteq L\downarrow_S$. Then suppose $r \in \mathbb{N}$ $(r > 0)$ be the maximum number such that 0^r is a factor of a gap word. Then since $I \subseteq L\downarrow_S$, 0^{3r+2} must be a factor of a run ρ, then for any run that embeds in to ρ, the factor 0^{3r+2} splits over at least 3 gap words, due to the maximality of r. But then there would be a gap word which is 0^l for some $l > 0$, which is a contradiction to non-existence of an I-witness.
- If $I = I_1 a \cdots a I_n a$, then consider the set of runs $R' \subseteq R$ that yield simple block bigger words than any word in I (we say R' covers I). If R' is finite, then there is no Kleene star in the pseudo-ideal. Hence there is a run among R' which yields $maximal(I)$ and this run witnesses I. Otherwise, if R' is an infinite set, then since $(R, \trianglelefteq)$ is a WQO, R' has finitely many minimal runs. Among these minimal runs, consider a minimal set of these minimal runs whose amalgamation closure R'' covers I. Then due to the up-directedness of pseudo-ideals, we can choose a sequence of runs $\rho_1 \trianglelefteq \rho_2 \trianglelefteq \cdots$ from R'' such that $\{\rho_1, \rho_2, \ldots\}$ covers I: for ρ_1 take the smallest run that embeds each run from the minimal set of minimal run (which corresponds to the join of yields of minimal runs).
  Then observe that each run in R'' yields n many a's. So we can construct an amalgamation system A_i that produces only the i-th a block of the yields of the runs in R'' for every $i \in [1, n]$: this can be done since amalgamation systems are closed under

rational transduction. Then since R'' covers I, it also covers I_i for $i \in [1, n]$. So, by induction hypothesis, there is a run that witnesses I_i in A_i, for every $i \in [1, n]$. Then there is a run ρ'_i in R'' which witnesses I_i. But every run in R'' embeds ρ_1, hence ρ_1 is a witness for I.
- If $I = (I'a)^*$, then consider the set of runs $R' \subseteq R$ that yield $I \cap L\downarrow_S = I$. Since $I \subseteq L\downarrow_S$, hence $(I'a)^k \subseteq L\downarrow_S$ for $k \in \mathbb{N}$. Let R'_i be the set of runs that witness $(I'a)^i$ which is an pseudo-ideal of the type above.
  Then again consider the sequence of runs $\rho_1, \rho_1 \cdots$ such that $\rho_i \in R'_i$. Since the set of runs is a WQO over run embeddings, there is a subsequence $\rho'_1, \rho'_2 \cdots$ such that every run embeds into the next run. Now consider a run ρ'_t in this sequence which belongs to R'_t, where $t > |\rho'_1|_{\mathsf{can}}$. Let $\rho'_1 \trianglelefteq_f \rho'_t$. Then there must exist $i \in [1, |\rho'_1|_{\mathsf{can}}]$, such that ρ'_t witnesses $(I'a)^l$ for some l in an interval of a gap. Moreover, by the definition of R', the gap interval only contains the letters from Σ_a. And hence, ρ'_t witnesses I between the interval of the gap word.

Now for the other direction, suppose that I-witness exists in the system recognizing $L\downarrow_S$.
- If $I = 0^{\leq k}$ for some $k \in \mathbb{N}$, then the I-witness yields a word w that contains 0^k as a factor. Then for any word $u \in I$, $u \preccurlyeq_{\mathsf{S}} w$, implying that $u \in L\downarrow_S$.
- If $I = 0^*$, then the I-witness contains two runs ρ, ψ such that $\psi \trianglelefteq_f \rho$ and $\mathrm{G}_{i,f} = 0^l$ for some i. Suppose $u = 0^k \in I$, then can construct a sequence of runs $\rho = \rho_1, \rho_2, \ldots$, ρ_i is obtained by amalgamating ρ_{i-1} with ψ. Then the yield of ρ_i contains 0^{il} as a factor. Then there exists i such that $i \times l \geq k$, and then yield of ρ_i is simple block bigger than u.
- If $I = I_1 a I_2 a \cdots a I_n a$, then there is a run ρ that witnesses I_i between some l_i and l_{i+1}. Let $u = w_1 a w_2 a \cdots a w_n a \in I$ such that $w_i \in I_i$. Then since ρ witnesses I_1 between l_1 and l_2, we can obtain a run ρ_1 such that $\rho \trianglelefteq_f \rho_1$ that yields a simple block bigger word than w_1 between some $f(i)$ and $f(i+1)$, and witnesses $I_2 a \cdots I_n a$ in an interval after $f(i+1)$. Continuing the same way, we obtain runs $\rho_2, \ldots \rho_n$ such that ρ_i contains a simple block bigger word than $w_1 a w_2 a \cdots w_i a$ before position j and witnesses $I_{i+1} a \cdots I_n a$ after position j. Then $u \preccurlyeq_{\mathsf{S}} \mathsf{yield}(\rho_n)$, implying $I \subseteq L\downarrow_S$.
- if $I = (I'a)^*$, then there exist two runs ρ, ψ in the witness set such that $\psi \trianglelefteq_f \rho$ and ρ is a witness for $(I'a)^l$ between $f(i)$ and $f(i+1)$, and $\rho[f(i), f(i+1)] \in \Sigma_a^*$, for some $l > 0$, $i \in [0, |\psi|_{\mathsf{can}} - 1]$ and $f \in E(\psi, \rho)$. Let $u = w_1 a w_2 a \cdots w_n a \in I$. Then we can amalgamate ρ k many times with ψ to obtain a run that witnesses $I'a$ in k disjoint intervals. Then with the arguments as above, we can get a run ρ' such that $u \preccurlyeq_{\mathsf{S}} \mathsf{yield}(\rho')$.

□

C DETAILS ON COUNTER EXTENSIONS

Claim 5.6. *If S is an amalgamation system and L is its language, then $L(S_{\eta,\alpha}) = L_{\eta,\alpha}$.*

Proof. By definition, $L(S_{\eta,\alpha}) = \bigcup_{(\rho,w)\in R_\eta} \alpha(\mathsf{yield}(\rho))$ and $L_{\eta,\alpha} = \alpha(\eta^{-1}(N_d) \cap L)$, thus it suffices to show that $\bigcup_{(\rho,w)\in R_\eta} \mathsf{yield}(\rho) = \eta^{-1}(N_d) \cap L$. Consider a word $w = a_1 \cdots a_n$ from $L = L(S)$. There is a run $\rho \in R$ such that $w = \mathsf{yield}(\rho)$ is accepted by ρ. Let us show that $\eta(w) \in N_d$ (by showing that it has an accepting run in the VASS for the language N_d) if and only if there exists an accepting decoration u of ρ.

If (ρ, u) is an accepting decorated run for some $u = (\mathbf{u}_1, \mathbf{v}_1) \cdots (\mathbf{u}_n, \mathbf{v}_n)$, then $q(\mathbf{u}_i) \xrightarrow{\eta(a_i)} q(\mathbf{v}_i)$ for all $i \in [1, n]$ in our VASS because $\mathbf{v}_i = \mathbf{u}_i + \delta(\eta(a_i))$ and η was assumed to be tame, $q(\mathbf{v}_i) = q(\mathbf{u}_{i+1})$ for all $i \in [1, n-1]$ because $\mathbf{v}_i = \mathbf{u}_{i+1}$, $q(\mathbf{u}_1) = q(\mathbf{0})$ because $\mathbf{u}_1 = \mathbf{0}$, and $q(\mathbf{v}_n) = q(\mathbf{0})$ because $\mathbf{v}_n = \mathbf{0}$. Thus there is a run of the VASS since

$$q(\mathbf{0}) = q(\mathbf{u}_0) \xrightarrow{\eta(a_1)} q(\mathbf{u}_1) \xrightarrow{\eta(a_2)} q(\mathbf{u}_2) \cdots q(\mathbf{u}_{n-1}) \xrightarrow{\eta(a_n)} q(\mathbf{u}_n) = q(\mathbf{0}) \,. \tag{6}$$

This shows that $\eta(w) = \eta(a_1) \cdots \eta(a_n) \in N_d$.

Conversely, if $\eta(w) \in N_d$, then there is a run of the VASS for N_d such that (6) holds, and we can decorate ρ with the sequence of pairs $u \stackrel{\text{def}}{=} (\mathbf{u}_0, \mathbf{u}_1)(\mathbf{u}_1, \mathbf{u}_2) \cdots (\mathbf{u}_{n-1}, \mathbf{u}_n)$; then $(\rho, u) \in R_\eta$ is an accepting decorated run. □

D DETAILS ON ALGEBRAIC EXTENSIONS

Let $\mathcal{C}$ be a class of languages with concatenative amalgamation and well-quasi-ordered decorations and let $\mathcal{G} = (N, T, S, \{L_A\}_{A\in N})$ be a $\mathcal{C}$ grammar. For each $A \in N$, let $\mathcal{M}_A$ be the amalgamation system with wqo decorations recognising L_A. Let $\preceq$ be the embedding in $\mathcal{M}_A$.

D.1 Well-Quasi-Orderedness and Decorations

Let τ_1 and τ_2 be trees of $\mathcal{G}$. We recall the definition of the embedding $\trianglelefteq$ in $\mathcal{G}$, being $\tau_1 \trianglelefteq \tau_2$ if there exists a subtree τ_2/p such that

(1) $\tau_1 = (A \to \rho)[t_1, \dots, t_n]$, $\tau_2/p = (A \to \sigma)[t'_1, \dots, t'_k]$ and $\rho \preceq_f \sigma$, and
(2) $t_i \trianglelefteq t'_{g(i)}$ for all $i \in [1, n]$, where $g = \mu_\sigma^{-1} \circ f \circ \mu_\rho$.

Lemma 5.12. *$(\mathcal{T}(\mathcal{G}), \trianglelefteq)$ is a well-quasi-order.*

Proof. We rely on Nash-Williams's *minimal bad sequence argument* [53]. Assume for the sake of contradiction that $(\mathcal{T}(\mathcal{G}), \trianglelefteq)$ is not a wqo. Then we can construct a *minimal* infinite bad sequence of trees $t_0, t_1, \dots$, where minimality means that for all i, any sequence $t_0, t_1, \dots, t_{i-1}, t, \dots$ where t is a (strict) subtree of t_i, i.e., $t = t_i/p$ for some $p \neq \varepsilon$, is good. To construct such a sequence, we start by selecting a tree t_0 minimal for the subtree ordering among all those that may start an infinite bad sequence; this t_0 exists because the subtree ordering is well-founded. We continue adding to this sequence by selecting a minimal t_i for the subtree ordering among all the trees $t \in \mathcal{T}(\mathcal{G})$ such that there exists an infinite bad sequence starting with $t_0, t_1, \dots, t_{i-1}, t, \dots$. At every step, the constructed sequence $t_0, \dots, t_i$ is bad. The infinite sequence remains bad: for every $i < j$, $t_0, \dots, t_j$ is a bad sequence, hence $t_i \not\trianglelefteq t_j$.

Let $S_i \stackrel{\text{def}}{=} \{t \in \mathcal{T}(\mathcal{G}) \mid \exists p \neq \varepsilon . t = t_i/p\}$ be the set of subtrees of t_i and $S \stackrel{\text{def}}{=} \bigcup_{i\geq 0} S_i$.

Claim D.1. *$(S, \trianglelefteq)$ is a wqo.*

Proof. Assume $(S, \trianglelefteq)$ is not a wqo. Then there is an infinite bad sequence $s_0, s_1, \dots$. Let i be minimal such that $s_0 \in S_i$. Since $\bigcup_{j\leq i} S_j$ is finite, without loss of generality we may assume that each s_k originates from a set S_ℓ with $\ell \geq i$.

Consider the sequence $t_0, \dots, t_{i-1}, s_0, s_1, \dots$. As s_0 is a strict subtree of t_i, by the minimality assumption of t_i, this sequence is good. Since the sequences $t_1, \dots, t_{i-1}$ and $s_0, s_2, \dots$ are both bad, there must exist j, k with $j < i$ such that $t_j \trianglelefteq s_k$. However, this means that there exists a subtree s_k/p satisfying the conditions of $\trianglelefteq$. As $s_k \in S_\ell$ for some $\ell \geq i$, there exists $p' \neq \varepsilon$ such that $s_k/p = t_\ell/p'p$. Thus $t_j \trianglelefteq t_\ell$ with $j < i \leq \ell$, a contradiction with the fact that $t_0, t_1, \dots, t_\ell$ is bad. ◻

We return to the proof that $(\mathcal{T}(\mathcal{G}), \trianglelefteq)$ is a well-quasi-order.

As there are only finitely many symbols in N, there is $A \in N$ and an infinite bad subsequence $t_{i_0}, t_{i_1}, \dots$ of $(t_i)_i$ where all the t_{i_j}'s are A-rooted; let us write ρ_j for the run of $\mathcal{M}_A$ labelling the root of $t_{i_j} = (A \to \rho_j)[\dots]$.

If any ρ_j has $\mathsf{can}(\rho_j) \in \Sigma_\varepsilon^*$, then $t_{i_j} = (A \to \rho_j)[]$ is a leaf. Because $\mathcal{M}_A$ is an amalgamation system and therefore $\preceq$ is a wqo, there exists $\ell > j$ such that $\rho_j \preceq \rho_\ell$. Then $t_{i_j} \trianglelefteq t_{i_\ell}$ because condition (1) holds by assumption and condition (2) is vacuous in this case, a contradiction.

We therefore assume that each ρ_j has $\mathsf{can}(\rho_j) = u_{j,0}B_{j,1}u_{j,1} \cdots B_{j,k_j}u_{j,k_j}$ for some $k_j > 0$ and non-terminals $B_{j,1} \cdots B_{j,k_j} \in N$; then $t_{i_j} = (A \to \rho_j)[t_{i_j}/1, \dots, t_{i_j}/k_j]$. We decorate each ρ_j with its sequence of children $w_j \stackrel{\text{def}}{=} t_{i_j}/1 \cdots t_{i_j}/k_j$, which all belong to S. This gives rise to an infinite sequence $(\rho_j, w_j)_j$ of decorated runs in $\mathrm{Deco}^S(R)$. Because $(S, \trianglelefteq)$ is a wqo by Claim D.1 and $\mathcal{M}_A$ supports wqo decorations by assumption, there is a pair $j < \ell$ with (1) $\rho_{i_j} \preceq_f \rho_{i_\ell}$ and (2) $t_{i_j}/k \trianglelefteq t_{i_k}/f(k)$ for all $k \in [1, k_j]$. This however implies that $t_{i_j} \trianglelefteq t_{i_\ell}$, again a contradiction. ◻

We can at this point show that $\mathcal{G}$ also supports wqo decorations.

Lemma 5.13. *$\mathcal{G}$ supports well-quasi-ordered decorations.*

Proof. Assume we decorate every terminal symbol of a tree with symbols from a wqo X. This is equivalent to decorating the runs ρ with the same symbols, introducing a new symbol 1 incomparable from all the elements of X for the output letters from N. Then $X \cup \{1\}$ is also a wqo.

Then the embedding $\trianglelefteq^X$ between decorated trees of $\mathcal{G}$ can be defined by replacing the order $\leq$ on the inner runs with $\leq^{X\cup\{1\}}$. Since C supports wqo decorations, $\leq^{X\cup\{1\}}$ is a well-quasi-ordering. The proof of Lemma 5.12 shows that $\trianglelefteq^X$ is a well-quasi-order as well. ◻

D.2 Admissible Embeddings

Recall the definition of the canonical decomposition of trees of C-grammars:

Definition 5.11. Assume $\tau = (A \to \rho)[t_1, \ldots, t_n]$ and $\mathsf{can}(\rho) = u_0 X_1 u_1 \cdots X_n u_n$. We define $\mathsf{can}(\tau) = \varepsilon \cdot u_0 \cdot \varepsilon\mathsf{can}(t_1)\varepsilon \cdot u_1 \cdots u_{n-1} \cdot \varepsilon\mathsf{can}(t_n)\varepsilon \cdot u_n \cdot \varepsilon$.

Intuitively, we wrap the canonical decomposition of τ itself and of each child t_i in ε on either side to delimit gap words produced by a mapping of t_i to a non-trivial descendant from those obtained by runs larger than ρ in the image of τ.

The nested structure of the decomposition requires us to define some additional notation to address the letters contributed by specific subtrees to the canonical decomposition of some tree. We define $\mathsf{ix}_p(\tau)$ as the offset of the canonical decomposition of τ/p in the decomposition of τ. That is, if the canonical decomposition of τ is $a_1 \cdots a_n$ and the canonical decomposition of τ/p is $b_1 \cdots b_m$ then b_i corresponds to $a_{\mathsf{ix}_p(\tau)+i}$.

Recall that we write μ_ρ for the map associating with every occurence of a non-terminal X_i in $\mathsf{yield}(\rho)$ its position in the canonical decomposition of ρ. Then if $p = \varepsilon$, we have

$$\mathsf{ix}_\varepsilon(\tau) = 0.$$

If $p = i \cdot p'$, we have

- the length of the canonical decomposition of ρ up to X_i,
- the combined length of the canonical decomposition of every t_j with $j < i$,
- less the individual letters $X_1, \ldots X_{i-1}$,
- two ε markers for every u_j with $j \leq i$,
- and finally the position of p' in the canonical decomposition of t_i.

Put together, we have

$$\mathsf{ix}_{i\cdot p'}(\tau) = \mu_\rho(i) + \left(\sum_{j=1}^{i-1} |t_j|_{\mathsf{can}}\right) + i + \mathsf{ix}_{p'}(t_i)$$

More generally, we have $\mathsf{ix}_{p\cdot q}(\tau) = \mathsf{ix}_p(\tau) + \mathsf{ix}_q(\tau/p)$. Observe also that if $\tau/(p \cdot i)$ and $\tau/(p \cdot [i+1])$ are defined, then $\mathsf{ix}_{p\cdot(i+1)}(\tau) = \mathsf{ix}_{p\cdot i}(\tau) + |\tau/(p\cdot i)|_{\mathsf{can}} + (\mu_\rho(i+1) - \mu_\rho(i)) + 1$.

If $\tau/p \trianglelefteq \tau'/p'$ and $f \in E(\tau/p, \tau'/p')$, we write $\hat{f}^{\tau/p}_{\tau'/p'}$ for the lifting of f to τ and τ', a function from $[\mathsf{ix}_p(\tau) + 1, \mathsf{ix}_p(\tau) + |\tau/p|_{\mathsf{can}}]$ to $[\mathsf{ix}_{p'}(\tau') + 1, \mathsf{ix}_{p'}(\tau') + |\tau'/p'|_{\mathsf{can}}]$ given by $\hat{f}^{\tau/p}_{\tau'/p'}(i + \mathsf{ix}_p(\tau)) = f(i) + \mathsf{ix}_{p'}(\tau')$.

If τ and τ' are trees from $\mathcal{G}$, than the set of admissible embeddings $E(\tau, \tau')$ is isomorphic to all the ways to embed τ into τ'.

Let $\tau = (A \mapsto \rho)[t_1, \ldots, t_l]$. Each $p \in \mathbb{N}^*$ such that $\tau'/p = (A \to \rho')[t'_1, \ldots, t'_k]$, $\rho \leq_\varphi \rho'$, and $t_i \trianglelefteq t'_{\varphi(i)}$ corresponds to a set of admissible embeddings. Let $f_i \in E(t_i, t'_{\varphi(i)})$. We write f'_i for $\hat{f_i}^{\tau/i}_{\tau'/p\cdot\varphi(i)}$. Note that the domains and co-domains of each f'_i are necessarily disjoint. Therefore we may take their union $g = \bigcup f_i$. This g induces a unique (partial) admissible embedding in $E(\tau, \tau')$. We are missing the mapping for the terminal letters in the canonical decomposition of ρ, as well as the ε-components, which we assign as follows:

Let $\nu \stackrel{\text{def}}{=} \mu_{\rho'}^{-1} \circ \varphi \circ \mu_\rho : [1, l] \to [1, k]$ be the subtree index map between $t_1, \ldots, t_l$ and $t'_1, \ldots, t'_k$. Let $\beta_\rho(i) \stackrel{\text{def}}{=} \min(\{j-1 \mid i < \mu_\rho(j)\} \cup \{l\})$ be the *block index* of a non-terminal letter at position i of the canonical decomposition of ρ. For convenience, we assume $\mathsf{ix}_0(\tau) = 0$ and $\mu_\rho(l+1) = |\rho|_{\mathsf{can}} + 1$

We have

$$\begin{aligned}
1 &\mapsto \mathsf{ix}_p(\tau') + 1 \\
|\tau|_{\mathsf{can}} &\ \mathsf{ix}_p(\tau') + |\tau'/p|_{\mathsf{can}} \\
\mathsf{ix}_i(\rho) &\mapsto \mathsf{ix}_p(\tau') + \mathsf{ix}_{\nu(i)}(\tau'/p) && (1 \le i \le l) \\
\mathsf{ix}_i(\rho) + |t_i|_{\mathsf{can}} + 1 &\mapsto \mathsf{ix}_p(\tau') + \mathsf{ix}_{\nu(i)}(\tau'/p) + |t'_{\nu(i)}|_{\mathsf{can}} + 1 && (1 \le i \le l) \\
\mathsf{ix}_i(\rho) + |t_i|_{\mathsf{can}} + j &\mapsto \mathsf{ix}_p(\tau') \cdot \mathsf{ix}_x(\tau'/p) + (\varphi(d) - \mu_{\rho'}(x)) + 1 \\
&\qquad (1 \le i \le l, 2 \le j \le (\mu_\rho(i+1) - \mu_\rho(i))) \\
&\qquad (\text{where } d = \mu_\rho(i) + j - 1, x = \beta_{\rho'}(\varphi(d))
\end{aligned}$$

Intuitively, we may assume that the ε-component directly to the left of the i-th gets mapped to the one directly to the left of the image of i, and the one directly to the right gets mapped to the one directly to the right of the image. Non-terminals get mapped to the corresponding non-terminal in ρ' by φ.

Note that different choices of $p \in \mathbb{N}^*$ induce different assignments for these values. If we assume that we have two different embeddings of τ but the same path p for both, then either the underlying run embedding φ must be different, which leads to a distinct lifting for the subtrees, or at least one subtree t_i has a different embedding into the same subtree $t'_{\varphi(i)}$ and by induction we may assume that this corresponds to a distinct embedding in $E(t_i, t'_{\varphi(i)})$. In brief, we may conclude that there is a one-to-one correspondence between tree embeddings and admissible embeddings.

D.3 Composition

Lemma 5.14. *If $\tau_0 \trianglelefteq_f \tau_1$ and $\tau_1 \trianglelefteq_g \tau_2$, then $\tau_0 \trianglelefteq_{g \circ f} \tau_2$.*

Let $\tau_0 = (A \to \rho_0)[\dots], \tau_1, \tau_2$ be trees from $\mathcal{G}$ and $\tau_0 \trianglelefteq_f \tau_1 \trianglelefteq_g \tau_2$. The maps f and g correspond to specific embeddings between τ, τ' and τ''. In particular, let p be the path corresponding to the mapping of τ_0 into τ_1. We have $\tau_1/p = (A \to \rho_1)[\dots]$ and $\rho_0 \preceq_\varphi \rho_1$. Let q be the path corresponding to the mapping of τ_1/p into τ_2. We have $\tau_2/q = (A \to \rho_2)[\dots]$ and $\rho_1 \preceq_\psi \rho_2$. Due to the structure of $\mathcal{M}_A$, we know that φ and ψ can be composed such that $\rho_0 \preceq_{\psi\circ\varphi} \rho_2$. Then the path q and the embedding of the children of τ along $\psi \circ \varphi$ is also a valid embedding of τ_0 into τ_2.

If we expand the composition of f and g, we get

$$\begin{aligned}
g(f(i)) &= g(\widehat{f'}^{\tau_0}_{\tau_1/p}(i)) && (\text{for } f' \in E(\tau_0, \tau_1/p)) \\
&= g(f'(i) + \mathsf{ix}_p(\tau_1)) \\
&= \widehat{g'}^{\tau_1/p}_{\tau_2/q}(f'(i) + \mathsf{ix}_p(\tau_1)) && (\text{for } g' \in E(\tau_1/p, \tau_2/q)) \\
&= g'(f'(i)) + \mathsf{ix}_q(\tau_2) \\
&= \widehat{(g' \circ f')}^{\tau}_{\tau_2/q}(i)
\end{aligned}$$

which is what we would get from the direct mapping of τ into τ_2/q.

An analogous line of reasoning holds for the case of $\tau_0 = (\varepsilon)[]$

D.4 Concatenative Amalgamation

Lemma 5.15. *If τ_0, τ_1, τ_2 are all A-rooted trees such that $\tau_0 \trianglelefteq_f \tau_1$ and $\tau_0 \trianglelefteq_g \tau_2$, then for every choice of $i \in [0, |\tau_0|_{\mathsf{can}}]$ there exists an A-rooted tree τ_3 with $\tau_1 \trianglelefteq_{f'} \tau_3$ and $\tau_2 \trianglelefteq_{g'} \tau_3$ such that*

(1) $f' \circ f = g' \circ g$ (we write h for this composition),
(2) $\mathrm{G}_{j,h} \in \{\mathrm{G}_{j,f}, \mathrm{G}_{j,g}, \mathrm{G}_{j,g}\mathrm{G}_{j,f}\}$ for every $j \in [0, |\tau_0|_{\mathsf{can}}]$, and in particular
(3) $\mathrm{G}_{i,h} = \mathrm{G}_{i,f}\mathrm{G}_{i,g}$ for the chosen i.

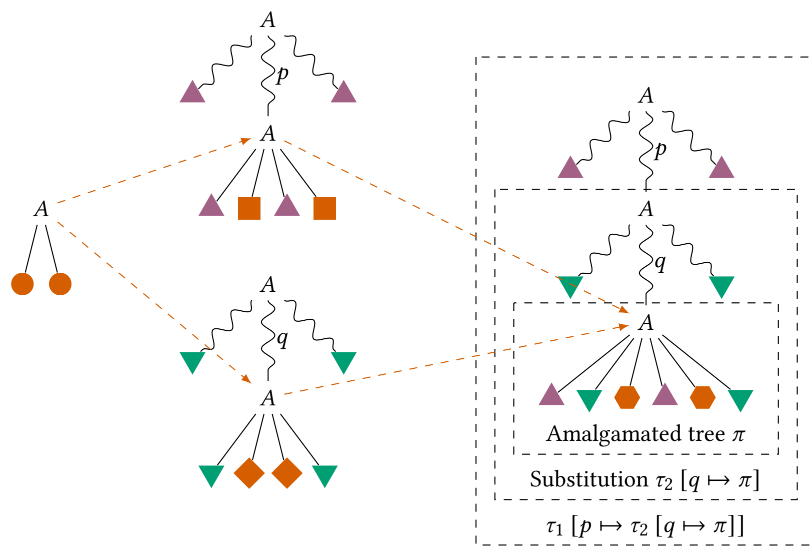


Figure 6: Amalgamation of trees τ_1 and τ_2 (middle) over base tree τ_0 (left).

If $\tau_0 = (\varepsilon)[]$ and therefore also τ_1 and τ_2, the statement trivially holds. We therefore consider the interesting case.

As an intuition for the amalgamation of trees, see Figure 6. To make the construction of the large tree easier, we introduce notation for the substitution of subtrees, as in [48, Sec. 3]. If $\tau = (A \to \rho)[\dots]$, π are trees and $\pi/p = (A \to \rho')[\dots]$, we inductively define $\pi\,[p \mapsto \tau]$ as

$$\pi\,[\varepsilon \mapsto \tau] = \tau$$
$$(B \to \rho)[t_1, \dots, t_n]\left[(i \cdot p') \mapsto \tau\right] = (B \to \rho)[t_1, \dots, t_{i-1}, t_i\left[p' \mapsto \tau\right], t_{i+1}, \dots, t_n]$$

Note that this operation maintains all labels along the path p and both τ and π/p are A-rooted. The result is therefore a valid tree of $\mathcal{G}$ again.

Substituting a subtree by a larger tree makes the entire tree larger:

Lemma D.2. *If $\pi/p \trianglelefteq \tau$ and τ and π/p are both A-rooted, then $\pi \trianglelefteq \pi\,[p \mapsto \tau]$ (and π and $\pi\,[p \mapsto \tau]$ are both B-rooted).*

Proof. If $p = \varepsilon$, then this is trivially true. Otherwise let $p = i \cdot p'$. Let t_i be the i-th child of p. By induction, we have $t_i \trianglelefteq t_i\,[p' \mapsto \tau]$. Then the definition of the tree embedding means we have $\pi \trianglelefteq \pi\,[i \mapsto t_i\,[p' \mapsto \tau]] = \pi\,[p \mapsto \tau]$. $\square$

We can now proceed with the proof of Lemma 5.15. Refer also to Fig. 6 for a visual example.

Proof. Let τ_1, τ_2 be trees such that $\tau_0 \trianglelefteq_f \tau_1$ and $\tau_0 \trianglelefteq_g \tau_2$. Recall that this means there are p, p' such that $\tau_1/p = (A \to \rho_1)[t'_1, \dots, t'_{k'}]$, $\rho_0 \preceq_\varphi \rho_1$ and $t_i \trianglelefteq t'_{\varphi(i)}$; similarly $\tau_2/p' = (A \to \rho_2)[t''_1, \dots, t''_{k''}]$, $\rho_0 \preceq_\psi \rho_2$ and $t_i \trianglelefteq t''_{\psi(i)}$.

As C has concatenative amalgamation, we can construct a run ρ_3 in the system associated with A such that $\rho_1 \preceq_{\varphi'} \rho_3$ and $\rho_2 \preceq_{\psi'} \rho_3$ such that $\varphi' \circ \varphi = \psi' \circ \psi$.

We now construct a tree $\pi = (A \to \rho_3)[t'''_1, \dots, t'''_{k'''}]$. For each j in $[1, k''']$, we can distinguish three cases:

- j is in the image of φ' and not in the image of ψ'. Then we set $t'''_j = t''_{\varphi'^{-1}(j)}$ (corresponding to the upward triangular nodes in Fig. 6).
- j is not in the image of φ', but is in the image of ψ'. Then we set $t'''_j = t''_{\psi'^{-1}(j)}$ (corresponding to the lower triangular nodes in Fig. 6).

- j is in the image of both φ' and ψ'. Then j is in the image of $\varphi' \circ \varphi$. We have $t_{(\varphi'\circ\varphi)^{-1}(j)} \trianglelefteq t'_{\varphi'^{-1}(j)}$ and $t_{(\varphi'\circ\varphi)^{-1}(j)} \trianglelefteq t''_{\psi'^{-1}(j)}$ and all three are X-rooted. We can construct, by induction, an X-rooted tree t'''_j such that $t'_{\varphi'^{-1}(j)} \trianglelefteq t'''_j$ and $t''_{\psi'^{-1}(j)} \trianglelefteq t'''_j$ (corresponding to the hexagonal nodes in Fig. 6).

We have $\tau_1/p \trianglelefteq \pi$ and both are A-rooted. By Lemma D.2, we can substitute π for τ_1/p. Then we have $\pi \trianglelefteq \tau_1\,[p \mapsto \pi]$. Analogously we have $\tau_2/p' \trianglelefteq \tau_1\,[p \mapsto \pi]$ and both subtrees are A-rooted. We can therefore again substitute $\tau_1\,[p \mapsto \pi]$ for τ_2/p'. We get that τ_1 and τ_2 both embed into $\tau_2\,[p' \mapsto \tau_1\,[p \mapsto \pi]]$. Equivalently, we may swap the order of substitutions to obtain $\tau_1\,[p \mapsto \tau_2\,[p' \mapsto \pi]]$.

Let i be a gap of τ_0. We distinguish the following cases:

- i is the first or the last gap. By the definition of the admissible embeddings, the first and last ε in the canonical decomposition of τ_0 gets mapped to the first or last ε-component of π. It follows that the content of the gap comes only from the context of the final substitutions and depending on the order of these substitutions we have $\mathrm{G}_{h,0} = \mathrm{G}_{f,0}\mathrm{G}_{g,0}$ or vice versa and equivalently for $\mathrm{G}_{h,|\tau_0|_{\mathrm{can}}}$.
- i is the gap between two symbols that occur on the level of τ_0 and not one of its children. Then the content of the gap is induced by the gap language $\mathrm{G}_{\varphi'\circ\varphi,k}$, which satisfies the concatenative amalgamation property.
- i is the gap between an ε-component and a letter of a child t_j, or occurs entirely within t_j. Then a simple inductive argument shows that the concatenative amalgamation property holds.

□

E DETAILS ON VALENCE AUTOMATA

Recall that a *monoid* is a set with a binary associative operation and a neutral element. Intuitively, in a valence automaton over a monoid M, each edge is labelled by an input word and an element of the monoid. Then, an execution from an initial state to a final state is valid if the product of the monoid elements is the identity. Unless stated otherwise, we denote the operation by juxtaposition and the neutral element by 1.

E.1 Valence Automata

Formally, a *valence automaton* over a monoid M is an automaton $\mathcal{A} = (Q, \Sigma, M, \Delta, q_0, F)$, where Q is a finite set of *states*, Σ is a finite alphabet, $\Delta \subseteq Q \times \Sigma^* \times M \times Q$ is a finite set of *edges*, $q_0 \in Q$ is its *initial state*, and $F \subseteq Q$ is its set of *final states*. Towards defining the language of $\mathcal{A}$, we consider the following relation. For $(q, w, m), (q', w', m')$, we write $(q, w, m) \to (q', w', m')$ if there is an edge $(q, u, x, q') \in \Delta$ such that $w' = wu$ and $m' = mx$. Then, language of $\mathcal{A}$ is defined as

$$L(\mathcal{A}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \exists q \in F \colon (q_0, \varepsilon, 1) \xrightarrow{*} (q, w, 1)\},$$

where $\xrightarrow{*}$ is the reflexive, transitive closure of $\to$.

E.1.1 Graphs Monoids. Here, we are interested in the case where the monoid M is defined by a finite graph. In the following, by a *graph* we mean a finite undirected graph $\Gamma = (V, E)$ where self-loops are allowed. Hence, V is a finite set of vertices, and $E \subseteq \{e \subseteq V \mid |e| \le 2\}$ is its set of edges. To each graph Γ, we associate a monoid as follows. Consider the alphabet $X_\Gamma \stackrel{\text{def}}{=} \{a_v, \bar{a}_v \mid v \in V\}$, i.e., we create two letters $a_v, \bar{a}_v$ for each vertex $v \in V$.

Intuitively, we think of the letters a_v as *increment* or *push* instructions and each $\bar{a}_v$ as the corresponding *decrement* or *pop* instructions. Let us make this formal. On the set X_Γ^* of words, we define an equivalence relation. Consider the relation

$$\begin{aligned} R_\Gamma \stackrel{\text{def}}{=}\ & \{(a_v\bar{a}_v, \varepsilon) \mid v \in V\} \\ & \cup \{(xy, yx) \mid x \in \{a_u, \bar{a}_u\},\ y \in \{a_v, \bar{a}_v\},\ \{u, v\} \in E\}. \end{aligned}$$

We now write $w \equiv_\Gamma w'$ if w' can be obtained from w by repeatedly replacing factors x by x' such that $(x, x') \in R_\Gamma$. First, this means we can always delete $a_v\bar{a}_v$ for any $v \in V$: this reflects the fact that $\bar{a}_v$ is the inverse operation of a_v. Moreover, if u and v are adjacent in

Γ, then the letters of u (i.e., a_u and $\bar{a}_u$) commute with the letters of v (i.e., a_v and $\bar{a}_v$). As another example, if the edge v has a self-loop in Γ, then we may commute a_v and $\bar{a}_v$, because $(a_v\bar{a}_v, \bar{a}_v a_v) \in R_\Gamma$. Finally, we define the monoid $\mathbb{M}\Gamma$ as the quotient $X_\Gamma^*/{\equiv_\Gamma}$. Thus, $\mathbb{M}\Gamma$ is the set of equivalence classes of X_Γ^* modulo $\equiv_\Gamma$ and multiplication is via $[x][y] \stackrel{\text{def}}{=} [xy]$ (this is well-defined since $\equiv_\Gamma$ is a congruence by definition).

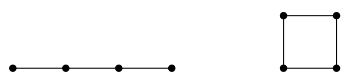
For example, if Γ consists of a single vertex, then $\mathbb{M}\Gamma$ is called the *bicyclic monoid* and is denoted by $\mathbb{B}$. Since then $X_\Gamma = \{a_v, \bar{a}_v\}$ and the only pair in R_Γ is $(a_v\bar{a}_v, \varepsilon)$, it is not difficult to see that for $w \in X_\Gamma^*$, we have $w \equiv_\Gamma \varepsilon$ if and only if w is a well-bracketed word where a_v and $\bar{a}_v$ are the opening and closing brackets. Hence, valence automata over $\mathbb{B}$ are automata with one $\mathbb{N}$-counter.

E.1.2 Valence Automata over Graphs. This allows us to define valence automata over graphs: For a graph Γ, a *valence automaton over* Γ is a valence automaton over the monoid $\mathbb{M}\Gamma$. By $\mathsf{VA}(\Gamma)$, we denote the class of languages accepted by valence automata over Γ.

E.2 Valence Automata as Amalgamation Systems

In order to deduce Main Theorem B from Main Theorem A, we need to show that if the emptiness problem is decidable for $\mathsf{VA}(\Gamma)$, then the language class $\mathsf{VA}(\Gamma)$ is a class of amalgamation systems. To this end, we show that $\mathsf{VA}(\Gamma)$ belongs to a language class obtained from the regular languages by repeatedly applying the operators $\cdot + \mathbb{N}$ and $\mathsf{Alg}(\cdot)$. This will follow from results in [67].

For a graph Γ, let Γ^- denote the graph where all self-loops are removed. In [67], it was shown that if Γ has one of the two graphs



(which are denoted P4 and C4, respectively) as an induced subgraph, then $\mathsf{VA}(\Gamma)$ is the class of all recursively enumerable languages, and in particular, the emptiness problem is undecidable for $\mathsf{VA}(\Gamma)$. Thus, for Main Theorem B, we only need to consider those graphs Γ for which Γ^- does not contain P4 and C4 as induced subgraphs. These graphs have been described in [67].

E.2.1 Graphs without P4 *and* C4. Let PD be the smallest (isomorphism-closed) class of monoids such that

(i) the trivial monoid 1 belongs to PD,
(ii) for every monoid M in PD, we also have $M \times \mathbb{B}$, $M \times \mathbb{Z}$ in PD, and
(iii) for any monoids M and N in PD, we also have $M * N$ in PD. Here, $M * N$ denotes the *free product* of monoids. The precise definition is not needed here—we will only need the following: in [62, Lem. 2], it is shown that for any monoids M, N, we have $\mathsf{VA}(M * N) \subseteq \mathsf{Alg}(\mathsf{VA}(M) \cup \mathsf{VA}(N))$.

A consequence of Theorem 3.3, Proposition 3.5, and Proposition 3.6 in [67] is that, if Γ^- does not contain P4 or C4 as an induced subgraph, then $\mathbb{M}\Gamma$ belongs to PD.

Proof of Main Theorem B. By the previous discussion, it remains to show that for every language in $\mathsf{VA}(M)$ with M in PD, we can construct an amalgamation system. For (i) the trivial monoid 1, $\mathsf{VA}(1)$ is just the class of regular languages, so this follows from Theorem 5.3.

Moreover, for the subcase of (ii) of monoids $M \times \mathbb{Z}$, a classic construction for counter systems yields the inclusion $\mathsf{VA}(M \times \mathbb{Z}) \subseteq \mathsf{VA}(M \times \mathbb{B} \times \mathbb{B})$. Indeed, a single $\mathbb{Z}$-counter can be simulated by two $\mathbb{N}$-counters.

Thus, it suffices to show that if $\mathsf{VA}(M)$ and $\mathsf{VA}(N)$ have concatenative amalgamation, then so do (ii) $\mathsf{VA}(M \times \mathbb{B})$ and (iii) $\mathsf{VA}(M * N)$. In the latter case, we know that $\mathsf{VA}(M * N) \subseteq \mathsf{Alg}(\mathsf{VA}(M) \cup \mathsf{VA}(N))$ [62, Lem. 2] and thus we may apply Theorem 5.10. Finally, for $M \times \mathbb{B}$, it is entirely straightforward to prove that $\mathsf{VA}(M \times \mathbb{B})$ is included in the language class $\mathsf{VA}(M) + \mathbb{N}$, where $\mathsf{VA}(M) + \mathbb{N}$ is defined as in Section 5.2. Indeed, a valence automaton over $M \times \mathbb{B}$ can be viewed has having three inscriptions on each edge, namely an element of M, an element of $\mathbb{B}$ (which is a counter update for an $\mathbb{N}$-counter), and an input word. This can be directly encoded into a language in $\mathsf{VA}(M) + \mathbb{N}$.

This shows that for every M in PD, all the languages in $\mathsf{VA}(M)$ have concatenative amalgamation systems. Finally, since each class $\mathsf{VA}(M)$ is a full trio [22, Thm. 4.1] this shows that Main Theorem B follows from Main Theorem A. $\square$