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, 24]. They were invented in the late 1980s [22] 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) [40, 43, 57], which is “compatible” with transition steps: if there is a step \( c \rightarrow c’ \) and a configuration \( d \geq c \), then there is a configuration \( d’ \geq c’ \) reached by a step \( d \rightarrow d’ \). 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 [28].

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 [19]—though famously decidable for vector addition systems [42, 44, 45, 50]. Another is that pushdown systems provide a notable exception to the applicability of the definition [24, 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) [49, 58].

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 [42], Lambert [44], and Mayr [50]: this is a structure capturing all the possible runs of the system between the source and target configurations. Beyond reachability, Lambert [44, 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. [32] 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. [18] 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., 15–17, 30], 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 $S$ 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 $S$ separating $u$ and $v$. Then for most classes $S$, we have that two languages $K$ and $L$ are inseparable by $S$ 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 [40] is that for every language $L$, the set $L^\downarrow$ of all scattered subwords of $L$ is a regular language [37]. However, effectively computing an automaton for $L^\downarrow$ is often a difficult task [14, 32]. In [62], it was shown that computing downward closures often reduces to the simultaneous unboundedness problem (SUP), which asks for a given language $L \subseteq \Sigma^*$, whether for every $k \in \mathbb{N}$, there exists a word $a_1^k \cdots a_n^k \in L$ with $x_1, \ldots, x_n \geq k$. Downward closure computation (also beyond the ordinary subword ordering) [65] based on the SUP have been studied in several papers in recent years [4, 35, 54]. 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 [31], 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 $w_1, \ldots, w_N \in \Sigma^*$ such that $L \subseteq w_1^* \cdots w_N^*$. Deciding language boundedness is motivated by the fact that bounded languages have pleasant decidability properties [e.g., 6, 11].

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 [48], 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 [47], 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 [23, 47] or in specialised subroutines of complex procedures [12, 17], 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).

The generality of this equivalence comes as a surprise, because the 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 [10, 51], but e.g., downward closures are not effective [10] and language boundedness is undecidable [11]—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 [64, 66]. 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 [25, 63, 64, 66, 67], and identifying the graphs $\Gamma$ 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 $\mathsf{SC}^-$, $\mathsf{SC}^+$, and $\mathsf{SC}^*$ in [64, 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 $\mathsf{SC}^-$ and $\mathsf{SC}^*$, emptiness is decidable [66], while for $\mathsf{SC}^*$ it is open.

We show in Section 5 that for every graph $\Gamma$, valence automata over $\Gamma$ 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 $\mathsf{SC}^*$) 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 $\Gamma$, the following are equivalent for valence automata over $\Gamma$:

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 [32], decidability of PTL-separability [16], decidability of language boundedness [18], and effectively regular unary languages [39]. Also, now that the proofs are decolored from the KLM decomposition, one can use any algorithm for VASS reachability (like Leroux’s simple invariant-based algorithm [45]) 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 $\mathsf{SC}^-$ and $\mathsf{SC}^*$ [66]. The difference between $\mathsf{SC}^-$ and $\mathsf{SC}^*$ is that when we apply “building stacks” and “adding counters,” then in $\mathsf{SC}^-$, we can only ever add $\mathbb{Z}$-counters, while in $\mathsf{SC}^*$, 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 contain $\mathbb{N}$-counters; these are equivalent to the sequential recursive Petri nets of Haddad and Pouzet [33, 34] and generalise both pushdown automata and VASS.

These classes differ in what was already known about decidability. For $\mathsf{SC}^-$, we knew that emptiness was decidable and that downward closures were computable [62, 63], and effective regularity of unary languages was known for $\mathsf{SC}^-$, because they have semilinear Parikh images [63] (this is not the case for $\mathsf{SC}^*$). The decidability of language boundedness was known for the smaller $\mathsf{SC}^-$ subclass of pushdown automata with reversal-bounded counters [6] (which model recursive programs with numeric data types [36]). For $\mathsf{SC}^*$, we knew that emptiness was decidable [66], 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 $\mathsf{SC}^*$.

**PVASS and their Restrictions.** In $\mathsf{SC}^*$, one can arbitrarily alternate between adding $\mathbb{N}$-counters and building stacks. Since $\mathbb{Z}$-counters can always be simulated by $\mathbb{N}$-counters, $\mathsf{SC}^*$ is more powerful than $\mathsf{SC}^*$, but the decidability of emptiness is open, the simplest open example being one-dimensional PVASS [66, 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 (VASS$_{\text{nz}}$) [3], and PVASS where the stack behaviour is oscillation bounded [26].

- A VASS$_{\text{nz}}$ is a VASS that has for each $i$ an operation that tests all counters $1, \ldots, i$ for zero at the same time. Reachability is decidable for VASS$_{\text{nz}}$ [8, 56] and the clever 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 [26] 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 VASS$_{\text{nz}}$, 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 [40, 43, 57] 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 \( e \) for the empty word, and define \( X^0 \equiv X \cup \{e\} \) when we want to treat it as a letter.

If \( (X, \leq) \) is a wqo, then by Higman’s Lemma, so is \( (X^*, \leq^{\star}) \) where \( w \leq w' \) if there exists a (scattered word) embedding, i.e., a strictly monotone map \( f : [1, |w|] \to [1, |w'|] \) such that \( w(i) \leq w'(f(i)) \) for all \( i \in [1, |w|] \). For instance, over the wqo \( (a, b, c, r) =:, acab \leq_B aabca 
\), 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 there root with label \( x \) and with immediate subtrees \( t_1, \ldots, t_n \) is in \( T(X) \). If \( t \in T(X) \) then there exists a subtree of \( t \) at \( p \), written \( t/p \), if it exists, is defined inductively by \( t/e \equiv t \) and \( x[t_1, \ldots, t_n](i \cdot p') \equiv t_i/p' \) where \( i \in [1, n] \).

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 \( t_i \leq_T t_{i_j} \), 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 Section 5. For these basic examples, Table 1 presents the known results pertaining to Main Theorem A.

### 3.1 Concatenative Amalgamation Systems

A (concatenative) amalgamation system is a tuple \( S = (\Sigma, R, E, \text{can}) \), where \( \Sigma \) is a finite alphabet and \( R \) is a (usually infinite) set of runs. Each run \( \rho \in R \) has an associated canonical decomposition \( \text{can}(\rho) = u_1 \cdots u_n \in \Sigma^* \) of some length \( |\rho|_{\text{can}} = n \), with every \( u_i \in \Sigma^* \). The corresponding accepted word \( \text{yield}(\rho) \in \Sigma^* \) is obtained by concatenating the \( u_i \)'s; the language accepted by the system is then \( L(S) \equiv \bigcup_{\rho \in R} \text{yield}(\rho) \). 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 \( \rho, \sigma, E(\rho, \sigma) \) is a set of admissible embeddings between their canonical decompositions. Here, an embedding over the alphabet \( (\Sigma, \leq) \) is defined as in Section 2: if \( \text{can}(\rho) = u_1 \cdots u_n \) and \( \text{can}(\sigma) = v_1 \cdots v_m \) are the canonical decompositions of the two runs, then an embedding of \( \rho \) in \( \sigma \) is a strictly monotone map \( f : [1, n] \to [1, m] \) with \( v_i = u_i \) for every \( i \in [1, n] \). For each embedding \( f \in E(\rho, \sigma) \) and \( i \in [0, n] \), we define the gap word \( G_{i,f} \in \Sigma^* \), such that \( v_0 \cdots v_m = G_{0,f}u_1G_{1,f} \cdots u_nG_{n,f} \).

- \( G_{0,f} \equiv u_1 \cdots u_f(1)-1 \),
- \( G_{i,f} \equiv u_f(i)1 \cdots u_{f(i)+1}-1 \) for all \( i \in [1, n-1] \),
- \( G_{n,f} \equiv u_f(n)1 \cdots u_m \).

| 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 | [62] | [62] | [62] |
| (2) \( L^\downarrow \) | folklore | [32] | [14, 59] |
| (3) PTL sep. | [55] | [16] | [16] |
| (4) boundedness | [29] | [18] | [29] |
| (5) unary eff. reg. | [53] | [39] | [53] |
| (7) emptiness | folklore | [42, 45, 50] | folklore |
If the set $E(\rho, \sigma)$ is non-empty, we write $\rho \preceq \sigma$; if we wish to refer to a specific $f \in E(\rho, \sigma)$, we write $\rho \trianglelefteq f \trianglerighteq \sigma$. Note that $\rho \preceq \sigma$ implies $\text{can}(\rho) \subseteq \text{can}(\sigma)$ but the converse might not hold: $E(\rho, \sigma)$ need not contain all the possible embeddings between the canonical decompositions of $\rho$ and $\sigma$.

### 3.1.2 Conditions

Finally, we require the following:

- Composition: if $f \in E(\rho, \sigma)$ and $g \in E(\sigma, \tau)$, then $f \circ g \in E(\rho, \tau)$.
- $\text{wqo}$ (w.r.t. $\preceq$) is a well-quasi-order, and
- (concatenative) amalgamation: if $\rho_0 \not\sqsubset f \trianglerighteq \rho_1$ and $\rho_0 \not\sqsubset g \trianglerighteq \rho_2$ with canonical decomposition $\text{can}(\rho_0) = \mathcal{w}_1 \cdots \mathcal{w}_n$, then for every choice of $i \in \{0, n\}$, there exists a run $\rho_3 \in R$ such that $\rho_1 \sqsubset f \trianglerighteq \rho_3 \sqsubset g \trianglerighteq \rho_2$ with $f' \circ f = g' \circ g$ (we write $h$ for this composition) and

$$G_{j,h} \in \{G_{j,f}G_{j,g}, G_{j,g}G_{j,f}\} \quad \text{for every } j \in \{0, n\}, \text{and}$$

$$G_{i,h} = G_{i,f}G_{i,g} \quad \text{for the chosen } i.$$

Thus the embedding $h$ of $\rho_0$ into $\rho_3$ has the property that each gap word is the concatenation of the two gap words from the embeddings into $\rho_1$ and $\rho_2$, in some order. Moreover, in the particular gap $i$, we know that the gap word from $\rho_1$ comes first, and then the gap word from $\rho_2$. This tells us that, given a gap $i \in \{1, n\}$, we can choose a run $\rho_3$ 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 $i \in [0, |\text{can}|]$

$$L_{\rho,i} \triangleq \{G_{i,h} \mid \exists \sigma \in R: \rho \not\sqsubset f \trianglerighteq \sigma\}.$$

In other words, $L_{\rho,i}$ is the set of all words that can be inserted in the $i$-th gap of $\rho$’s canonical decomposition when $\rho$ 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 $\rho$ 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 $\text{can}(\cdot)$ is computable, and

(iii) for any two runs $\rho, \sigma$ we can compute the set $E(\rho, \sigma)$ of admissible embeddings (and hence decide whether $\rho \preceq \sigma$ 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 $A = (Q, \Sigma, \Delta, I, F)$ with finite state set $Q$, input alphabet $\Sigma$, transition set $\Delta \subseteq Q \times \Sigma \times Q$, 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 $(q_0, a_1, q_1)\cdots(q_0, a_n, q_n) \in \Delta^*$ 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 $L(A) = \bigcup_{r \in R} \text{yield}(r)$ 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 $f: [1, n] \rightarrow [1, m]$ such that $s_{f(i)} = r_i$ for every $i \in [1, n]$. Thus $(R, \preceq)$ is the qo induced by $\rho \subseteq \sigma$ inside the word embedding $\text{qo} (\Delta^*, \preceq)$ over the alphabet ($\Delta = \Sigma$), and the composition and wqo conditions follow.

Concatenative amalgamation also holds. Let $\rho_0, \rho_1, \rho_3$ be runs with $\rho_0 \not\sqsubset f \trianglerighteq \rho_1$ and $\rho_0 \not\sqsubset g \trianglerighteq \rho_2$, where the canonical decomposition of $\text{can}(\rho_0)$ has $n$ transitions. Because the individual transitions of $\rho_0$ 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 $G_{i,f}$ and $G_{i,g}$ 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 $\rho_3$ as required.

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

If we consider the run $\rho_1 \equiv (q_0, a, q_1)(q_1, b, q_2)(q_2, a, q_f)$, then we can embed $\rho_0$ into $\rho_1$ via the map $\{1 \mapsto 1, 2 \mapsto 3\}$. Indeed, we get a decomposition of the yield of $\rho_1$ into $G_{0, f}aG_{1, f}aG_{2, f}$ with $G_{0, f} = G_{2, f} = e$ and $G_{1, f} = b$. And clearly we can take the $b$-labelled loop several times, yielding for instance the run $\rho_2 \equiv (q_0, a, q_1)(q_1, b, q_2)(q_2, a, q_f)$.

### 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 $\Sigma \times \mathbb{Z}^d$. $\Delta$ is now a finite subset of $Q \times \Sigma \times \mathbb{Z}^d \times Q$. We revisit here the results of [48] to show that VASS are amalgamation systems.

#### 3.3.1 Configurations and Semantics

Let $\text{Confs} \equiv Q \times \mathbb{N}^d$. A configuration is a pair $q(u) \in \text{Confs}$ of a state and the values of the $d$ counters. Configurations are ordered through the product ordering: $q(u) \leq q'(u')$ if $q = q'$ and $u \leq u'$. If $c = q(u)$ is a configuration and $\delta \in \mathbb{N}^d$, we write $c + \delta$ for the configuration $q(u + \delta)$; note that $c_1 \leq c_2$ if and only if there exists $\delta \in \mathbb{N}^d$ such that $c_2 = c_1 + \delta$.

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

3.3.2 Runs and Admissible Embeddings. A run is a sequence \( \rho = (c_0, t_0, t_1, c_1) \) such that \( c_0 = q_0(0) \) for some \( q_0 \in I \), \( c_n = q_f(F) \) for some \( q_f \in F \), and \( c_{i-1} \rho d c_i \) at each step. Let \( t_i \) be \( \{q_i-1, a_i, v_i, q_i\} \) for each \( i \) in this run; the associated canonical decomposition is \( \rho = a_1 \cdots a_n \) and we recover the usual notion of a VASS (reachability) language: \( \mathcal{L}(V^\prime) = \bigcup_{\rho \in R} \mathcal{R}(\rho) \).

Let \( (R, \leq) \) be the qo induced by \( R \subseteq (\text{Confs} \times \Delta \times \text{Confs}) \) inside the word embedding \( \text{qo} \) of \( (\text{Confs} \times \Delta \times \text{Confs})^\ast \) over the product alphabet \( (\text{Confs}, \leq) \times (\Delta, \leq) \times (\text{Confs}, \leq) \). Thus for two runs \( \rho_0 = (c_0, t_0, t_1, c_1) \) and \( \rho_1 = (c_0', t_0', t_1', c_1') \), we have \( \rho_0 \leq \rho_1 \) if there exists a strictly monotone map \( f : [1, n] \to [1, m] \) such that, for all \( i \in [1, n] \), \( c_i \leq c_i' \).

Similarly, we have \( \rho \) (1) and (2). We can amalgamate to a third run \( \rho_3 \) as above, i.e.,

\[
\begin{align*}
\rho_3 &= (c_0, t_0, t_1, c_1) \\
&\leq (c_0', t_0', t_1', c_1') \\
&\leq (c_0'', t_0'', t_1'', c_1'') \\
&\leq (c_0'''', t_0'''', t_1'''', c_1'''')
\end{align*}
\]

We mostly consider grammars from the perspective of their derivation trees. In a slight divergence from the usual definition (but consistent with e.g., [47, App. A]), we label nodes in derivation trees not with symbols from \( \Sigma \) but with productions from \( \Delta \). Indeed, the usual homeomorphic tree embedding (c.f. Section 2) over trees labelled by \( (\Sigma \cup \Delta) = \) a well-ordering, but this labelling is not sufficient for amalgamation and we shall rather rely on the homeomorphic tree embedding over trees labelled by \( (\Delta, \leq) \).

Example 3.4. Consider the grammar with \( N \equiv \{A, B\}, \Sigma \equiv \{a, b\} \), and \( \Delta \equiv \{A \to a, A \to B, B \to a, B \to b, A \to AB\} \) 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 \( \Delta \) are either non-terminal productions of the form \( A \to B_1 \cdots B_k \) with \( k > 0 \) and \( B_1, \ldots, B_k \in N \), or terminal productions of the form \( A \to a \) with \( a \in \Sigma_e \). We call a tree in \( T(\Delta) \) \( A \)-rooted if its root label is a production \( A \to a \) for some \( a \). Using the notations from Section 2, a derivation tree is either a left \( (A \to a) \) \( k \)-labelled by a terminal production \( (A \to a) \in \Delta \), or a tree \( (A \to B_1 \cdots B_k)[t_1, \ldots, t_k] \) 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.

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 \( G = (N, \Sigma, S, \Delta) \) with finite non-terminal alphabet \( N \), finite terminal alphabet \( \Sigma \), initial non-terminal \( S \in N \), and finite set of productions \( \Delta \subseteq N \times (\Sigma \cup \{\epsilon\}) \); each such production is written \( A \to \alpha \) with \( A \in N \) and \( \alpha \in (N \cup \Sigma)^* \).

(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.

We have \( \rho_0 \leq \rho_1 \) and \( G_{\rho_0} = b, G_{\rho_1} = \epsilon, \) and \( G_{\rho_2} = b \). Similarly, we have \( \rho_0 \leq \rho_2 \) and \( G_{\rho_0} = b, G_{\rho_2} = \epsilon, \) and \( G_{\rho_2} = b \).

We can amalgamate to obtain \( \rho_1 = (q_0(1), t_0, q_0(1)) \) (resp. \( q_1(1), t_0, q_1(1), q_f(1), t_0, q_f(1) \)).

We have \( \rho_0 \leq \rho_2 \) and \( G_{\rho_0} = b, G_{\rho_2} = b \). Similarly, we have \( \rho_0 \leq \rho_2 \) and \( G_{\rho_0} = c, G_{\rho_2} = c, \) and \( G_{\rho_2} = c \).

We can amalgamate to obtain \( \rho_1 = (q_0(1), t_0, q_0(1)) \) (resp. \( q_1(1), t_0, q_1(1), q_f(1), t_0, q_f(1) \)).

We can amalgamate to obtain \( \rho_1 = (q_0(1), t_0, q_0(1)) \) (resp. \( q_1(1), t_0, q_1(1), q_f(1), t_0, q_f(1) \)).

One can check that \( \rho_0 \leq \rho_2 \) and \( G_{\rho_0} = b, G_{\rho_2} = b \) for all \( i \in [0, 2] \).
Example 3.4 (continued). In Fig. 3b the nodes in dashed boxes in the right tree form an A-context corresponding to the derivation $A \Rightarrow B \Rightarrow AB \Rightarrow A b$. This A-context can be iterated (and thus the derivation as well) to derive larger and larger trees before plugging the node $A \Rightarrow a$ from the image of the left tree.

Finally, the canonical decomposition of a derivation tree is defined inductively by $\text{can}(A \Rightarrow a[i]) \triangleq \text{rAR}$ for $a \in \Sigma$ and $\text{can}(A \Rightarrow a[t_1, \ldots, t_n]) \triangleq \text{rcan}(t_1) \cdots \text{rcan}(t_n)$ otherwise. A run is then an S-rooted derivation tree, and this matches the usual definition of the language of a context-free grammar: $L(G) = \bigcup_{\rho \in \mathcal{R}} \text{yield}(\rho)$. A comment is in order for those explicit $\varepsilon$'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 $\varepsilon$ reflects 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 extend. 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 $x_1 \cdots x_n \geq k$ and $a_1^{x_1} \cdots a_n^{x_n} \in L$?

There has been some interest in this problem because in [62, 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) $\Rightarrow$ (2)” Moreover, in [16, 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) $\Rightarrow$ (3)” In fact, analogous results hold also for some orderings beyond the subword ordering [65]. Given this motivation, the SUP is known to be decidable for VASS [18, 32], higher-order pushdown automata [35], and even higher-order recursion schemes [5, 13, 54].

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 $\Psi(\text{yield}(\rho))$. For two vectors $u, v \in \mathbb{N}^n$, we write $u \prec v$ if, for all $i \in [1, n], u(i) < v(i)$.

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 $L \subseteq a_1^{x_1} \cdots a_i^{x_i} a_{i+1}^{x_{i+1}} \cdots a_n^{x_n}$, which can be decided in a full trio with an oracle for the emptiness problem by checking whether $L \cap (\Sigma \setminus a_1^{x_1} \cdots a_i^{x_i} a_{i+1}^{x_{i+1}} \cdots a_n^{x_n}) = \emptyset$. The other one enumerates pairs of runs $\rho, \sigma$ and checks whether (i) $\rho \preceq \sigma$ 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 $\rho$ and $\sigma$ as above, then by amalgamation, we obtain runs $a_1, a_2, \ldots$, such that $\Psi(a_i(i)) \geq k$, 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 $\Psi(\rho'_k(i)) \geq k$ for every $i \in [1, n]$. Since $(\mathbb{N}, \preceq)$ is a wqo, we can find $j < t$ such that $\rho'_j \preceq \rho'_t$. By definition of the $\rho'_k$, this pair satisfies $\Psi(\rho'_j) \ll \Psi(\rho'_t)$. □

Proof of “(1) $\Rightarrow$ (7)” Assuming decidable SUP, for any $L \subseteq \Sigma^*$, take the rational transduction $T \triangleq \{a_1\}^*$ 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., 29]—and is even in polynomial time [27]. The question was open for many years in the case of reversal-bounded counter machines (RBCM) [9, 20] before it was settled for VASS in [18]. 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 [27] for a precise definition). After being open for RBCM for a long time [41], this was shown in [6] for RBCM and for pushdown RBCM.

We begin with a simple characterisation in the case of subsemi-}


alonge.
prefix-comparable, we have $L \subseteq \text{prefixes}(w^*)$. Since $w^*$ is bounded, \text{prefixes}(w^*) and thus $L$ are bounded as well. \hfill □

**Proof of “(7) ⇒ (4)”**. We again provide two semi-decision procedures. The procedure for positive instances simply enumerates expressions $w_1^* \cdots w_n^*$ and checks whether $L \subseteq w_1^* \cdots w_n^*$, 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 $p_0, p_1, p_2$ with $p_0 \not\leq_f p_1$ and $p_0 \not\leq_g p_2$ such that for some $i$, the words $G_{i,f}$ and $G_{i,g}$ 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 $L_{p_0,i}$ contains two prefix-incomparable words. Moreover, by Observation 3.1, $L_{p_0,i}$ is a subsemigroup, and thus by Lemma 4.1, the language $L_{p_0,i}$ is not bounded. However, every word in $L_{p_0,i}$ 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 $L_{p_0,i}$.

Conversely, suppose there is no non-boundedness witness. As $(R, \not\leq)$ is a wqo, $R$ itself has a finite basis $(p_1, \ldots, p_m)$. Let $\text{can}(p_i) = a_{i,1} \cdots a_{i,n_i}$ be the canonical decomposition of $p_i$. Then we have

$$L \subseteq \bigcup_{i=1}^m L_{p_0,0}a_{i,1}L_{p_1,1} \cdots L_{p_n,n_i}L_{p_0,n_i}.$$  

(3) Since there is no non-boundedness witness, each language $L_{p_0,i}$ is linearly ordered by the prefix ordering. Moreover, by Observation 3.1 each language $L_{p_0,i}$ is a subsemigroup and thus Lemma 4.1 implies that $L_{p_0,i}$ is bounded. As boundedness is preserved by finite products, finite unions, and taking subsets, $L$ must be bounded. \hfill □

**Proof of “(4) ⇒ (7)”**. Given a language $L \subseteq \Sigma^*$, consider the rational transduction $T = \Sigma^* \times \{a, b\}^*$. Then $TL$ is a bounded language (actually, the empty set) if and only if $L = \emptyset$ (as otherwise, $TL = \{a, b\}^*$ is not bounded). \hfill □

### 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 yield($\rho$) for a run $\rho$ is a natural number.

First, we provide in Section 4.3.1 a very simple proof due to Leroux [46] 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 [39], which relies on Hauschildt’s dissertation [38].

#### 4.3.1 Regularity.

There is a proof due to Leroux which shows, only using amalgamation, that all unary VASS languages are regular [46], and happens to apply to our notion of amalgamation systems. It relies on a folklore result from number theory (see, e.g. [60]).

**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 $\mathbb{N} \cdot \text{gcd}(S)$.

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

**Lemma 4.3 ([46]).** Every unary language accepted by an amalgamation system is regular.

**Proof.** Let $L \subseteq \mathbb{N}$ be the language of an amalgamation system $((a), R, E, \text{can})$. We want to show that $L$ is ultimately periodic. Since $(R, \not\leq)$ is a wqo, the set $R$ has a finite basis, say $(\rho_1, \ldots, \rho_m)$. For each $\rho_i$, consider the set

$$L_i \triangleq \{\text{yield}(\sigma) - \text{yield}(\rho_i) \mid \exists \sigma \in R: \rho_i \not\leq_\sigma \}.$$  

(4) Since every run of $R$ embeds one of the runs $\rho_1, \ldots, \rho_m$, we have $L = \{\text{yield}(\rho_1) + L_1 \cup \cdots \cup \{\text{yield}(\rho_m) + L_m\}$. 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 $L$. \hfill □

**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 $(\rho, \sigma, \tau)$ of runs such that $\rho \not\leq \sigma$ and $\rho \not\leq \tau$. The set represented by $(F, T)$ is defined as

$$S(F, T) \triangleq \{\text{yield}(\rho) \mid \rho \in F\} \cup \bigcup_{(\rho, \sigma, \tau) \in T} S(\rho, \sigma, \tau).$$

where, for runs $\rho, \sigma, \tau$ with $\rho \not\leq \sigma$ and $\rho \not\leq \tau$,

$$S(\rho, \sigma, \tau) \triangleq \text{yield}(\rho) + \mathbb{N} \cdot (\text{yield}(\sigma) - \text{yield}(\rho))$$

$$+ \mathbb{N} \cdot (\text{yield}(\tau) - \text{yield}(\rho)).$$

Our algorithm works as follows. It enumerates unary witnesses $(F, T)$ 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 $(R, \not\leq)$ 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 $\mathbb{N} \cdot \text{gcd}(L_i)$. In particular, there are $k, \ell \in L_i$ with $k - \ell = \text{gcd}(L_i)$. This means that there are runs $\sigma_i$ and $\tau_i$ with $\rho_i \not\leq_\sigma \sigma_i$ and $\rho_i \not\leq_\tau \tau_i$ with $\text{yield}(\sigma_i) = \text{yield}(\rho_i) + k$ and $\text{yield}(\tau_i) = \text{yield}(\rho_i) + \ell$. We choose $T \triangleq \{(\rho_i, \sigma_i, \tau_i) \mid i \in \{1, m\}\}$. We now claim that the set $L \setminus S(0, T)$ is finite. Note that if this is true, we are done, because we can choose $F$ by picking a run for each number in $L \setminus S(0, T)$. For finiteness of $L \setminus S(0, T)$, it suffices to show finiteness of $L_i \setminus G_i$ for each $i$, where

$$G_i \triangleq \mathbb{N} \cdot (\text{yield}(\sigma_i) - \text{yield}(\rho_i)) + \mathbb{N} \cdot (\text{yield}(\tau_i) - \text{yield}(\rho_i)).$$

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 $\mathbb{N} \cdot \text{gcd}(G_i)$.

Since $\text{yield}(\sigma_i) - \text{yield}(\rho_i)$ and $\text{yield}(\tau_i) - \text{yield}(\rho_i)$ 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 yield$(\sigma_i) - \text{yield}(\tau_i) = \text{gcd}(L_i)$ by the choice of $\sigma_i$ and $\tau_i$. The claim is established. \hfill □
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 [31]. In that setting, one has an alphabet $\Sigma$ with associated priorities in $[0, d]$, specified by a priority map $p : \Sigma \rightarrow [0, d]$. Then if $u <_p v$ holds if $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$ 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

$$L|_p = \{ u \in \Sigma^* \mid \exists v \in L : u <_p v \}.$$  

As with the ordinary word embedding, because $<_p$ well-quasi-orders the set of words with priorities [31, Lem. 3.2], the language $L|_p$ is regular for any language $L \subseteq \Sigma^*$, hence the problem of computing priority downward closures, i.e., computing an NFA for $L|_p$ for an input language $L$.

The proof of “(6) $\Rightarrow$ (7)” consists in observing that $L = \emptyset$ if and only if $L|_p = \emptyset$, the latter being straightforward with an NFA recognising $L|_p$. 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 [62]. It essentially enumerates $<_p$-downward closed sets $D$, decompose them into finitely many ideals as $D = I_1 \cdot \cdots \cdot I_n$, and then decides (i) whether $L$ is included in $D$ and (ii) whether each $I_j$ is included in $L|_p$. Here, in a wqo $(X, \preceq)$, 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 $x \preceq z$ and $y \preceq z$. It is a general property of wqos that every downwards closed set decomposes into finitely many ideals [e.g., 30, 48]. 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 [62]. 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

$$K = \{ u_1 \cdots u_n \mid n \in \mathbb{N}, u_0 u_1 \cdots u_n \in L, \forall i \in [0, n] :$$

- either $(a) v_i \in (0)^*, (b) v_i \in (10)^*$ and $u_{i+1} \in (10)^*$, or
- $(c) i = n$ and $v_i \in (10)^*$

Hence, words in $K$ are obtained from words in $L$ by removing (a) factors in 0’s 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 $(10)^*$. This

4.4.2 Witnesses for Nested Unboundedness. In an amalgamation system for $K$, we have a simple kind of witness for our property: runs $\rho_0 \sqsubseteq \rho_1 \sqsubseteq \rho_2$ such that (i) some gap of $\rho_0 \sqsubseteq \rho_1$ between some positions $i < j$ of $\rho_1$ contains a 1 and (ii) some gap of $\rho_1 \sqsubseteq \rho_2$, which is between $i$ and $j$ in $\rho_1$, belongs to 0*. Then, by amalgamating $\rho_2$ with itself above $\rho_1$ again and again, we can create arbitrarily long blocks of contiguous 0’s. The resulting run $\rho'_2$ still embeds $\rho_1$ and thus $\rho_0$, such that one gap of $\rho'_2$ in $\rho_0$ contains both our long block of 0’s and also a 1. Thus, amalgamating $\rho'_2$ again and again above $\rho_0$, 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 $w_i = (0^i 1)^{2M}$ for $i \geq 1$. Then $w_1, w_2, \ldots \in K$, so there must be runs $\rho_1$ and $\rho_2$ with $\rho_1 \sqsubseteq \rho_2$ such that $\text{yield}(\rho_1) = w_i$ and $\text{yield}(\rho_2) = w_j$. Then every non-empty gap of $\rho_2$ in $\rho_1$ belongs to 0*. Moreover, any embedding of a minimal run $\rho_0$ of $S$ into $\rho_1$ will have some gap containing two 1’s, and thus have 10^i 1 as a factor. Thus the runs $\rho_0, \rho_1, \rho_2$ 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 $<_\Sigma$, such that downward closure computability of $L|_\Sigma$ with respect to $<_\Sigma$ implies that of $L|_p$ with respect to $<_p$. We then provide a syntactic description of the ideals of $<_\Sigma$. 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 $L|_\Sigma$ such that we have $I \subseteq L|_\Sigma$ if and only if an amalgamation system for $L|_\Sigma$ possesses an I-witness. There, we use that $L|_\Sigma$ is obtained using a rational transduction from $L$. 
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.

5.0.1 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 + N and 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| = |yield(ρ)|. Intuitively, one might think of a decoration as adding additional information from X to every letter in yield(ρ). For a set of runs R, we write DecoX(R) for the set of all X-decorations of runs from R. If (X, ≤) is a qo, we define the set of admissible embeddings between X-decorated runs by $E^X((ρ, u), (σ, v)) \equiv \{(f \in E(ρ, σ)) | u_i \leq σ_f(i) \text{ for all } i \in [1, |yield(ρ)|]\}; this defines a quasi-ordering $\preceq^X$ on DecoX(R) such that $(ρ, u) \preceq^X (σ, v)$ if and only if there is f such that $ρ \preceq_f σ$ and $u_i \leq σ_f(i)$ for all $i \in [1, |yield(ρ)|]$.

Definition 5.1 (wqo decorations). An amalgamation system with set of runs $(R, \preceq)$ supports wqo decorations if, for every wqo $(X, \leq)$, the set of decorated runs $(\text{Deco}^X(R), \preceq^X)$ 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. As it turns out, supporting wqo decorations is in practice 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 $\cdot + N$ and Alg(·).

Remark 5.2. Not every amalgamation system supports wqo decorations. Here is an example: let $Σ = \{a\}$ and define $R \equiv Σ$ with can(w) $\equiv \{w\}$ for all $w \in Σ^*$. If $n \leq m$, we define the identity function $f : i \mapsto i$ 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 $≤$ is the prefix ordering, but over a unitary alphabet it gives rise to a wqo $(R, \preceq)$ isomorphic to $(N, \leq)$. Also, those embeddings compose, and if $a^n \preceq a^t$ and $a^n \preceq a^t$ (thus with $n \leq r$ and $n \leq s$), then letting $m \equiv r + s - n$ allows to amalgamate into $a^m$.

Consider now the wqo $X \equiv \{(A, B), =\}$, i.e., the finite set $(A, B)$ with the equality relation. Then $(\text{Deco}^X(R), \preceq^X)$ is isomorphic with the prefix ordering over $(A, B)^*$, which is not a wqo. Indeed, among the decorated runs in DecoX(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 $(a^n, B^{n-1}A) \not{\preceq}^X (a^m, B^{m-1}A)$.
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 $(X, ≤)$ be a wqo, and $(ρ, w)$ an $X$-decoration of a run $ρ = (q₀, a₁, q₁) · · · (qᵦ₋₁, aᵦ, qᵦ) ∈ R$. Since the automaton is $ε$-free, $w$ is of length $|w| = n$ and can be written as $w = x₁ · · · xₙ$. Then the map $r: \text{Deco}^X(R) → (Δ × X)^*$ defined by $r: (ρ, w) ↦ ((q₀, a₁, q₁), x₁) · · · ((qᵦ₋₁, aᵦ, qᵦ), xₙ)$ is an order reflection, in that if $r(ρ, w) ≤ r(ρ', w')$, then $(ρ, w) ≤^X (ρ', w')$. By Dickson’s and Higman’s lemmata, $(Δ × X)^*, ≤_≈$ is a wqo, hence this order reflection shows that $(\text{Deco}^X(R), ≤_≈)$ is also a wqo. Indeed, $r$ pointwise maps bad sequences $(ρ₀, w₀), (ρ₁, w₁), · · ·$ over $(\text{Deco}^X(R), ≤_≈)$ to bad sequences $r(ρ₀, w₀), r(ρ₁, w₁), · · ·$ of the same length over $(Δ × X)^*, ≤_≈$, hence bad sequences over $(\text{Deco}^X(R), ≤_≈)$ 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 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 $\text{Reg} + \mathbb{N} = \text{VASS}$. 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 $U_d ⊆ \mathbb{Z}^d$ of unit updates, defined by $U_d = \{0\} ∪ \{e₁, · · · , e_d \ | \ i ∈ [1, d]\}$ where each $eᵢ$ is the unit vector such that $eᵢ(j) = 1$ and $eᵢ(j) = 0$ for all $j ≠ i$. Then the morphism $δ: U_d → \mathbb{Z}^d$ maps words over $U_d$ to their effect, and is defined by $δ(u₁ · · · uₙ) = Σᵢ⁺¹ uᵢ$. Finally, let $N_d ⊆ U_d$ be the language of all $\mathbb{N}$-counter-like words over $U_d$; formally,

$N_d = \{ w ∈ U_d \ | \ δ(w) ≥ 0 \}$

for all prefixes $w$ of $u$ and $δ(w) = 0$.

Put differently, $N_d$ is the language of the VASS with a single state $q$ and a transition $(q, u, u, q)$ for each $u ∈ U_d$.

Let $Δ$ be a finite alphabet; a morphism $η: Δ^* → U_d$ is tame if, for all $a ∈ Δ$ and all $i ∈ [1, d]$, all the occurrences of $eᵢ$ in $η(a)$ occur before all the occurrences of $–eᵢ$. Let $Σ$ also be a finite alphabet; a morphism $α: Δ^* → Σ^*$ is alphabetic if $α(Δ) ⊆ Σ^*$. 

**Definition 5.4 (Counter extension).** Let $L$ be a language over a finite alphabet $Δ$. For a tame morphism $η: Δ^* → U_d$ and an alphabetic morphism $α: Δ^* → Σ^*$ into a finite alphabet $Σ$, let $L_ηα$ be the language $η⁻¹(N_d) \cap L$. For a class of languages $C$, $C + \mathbb{N}$ is the class of languages $L_{ηα}$, when $L$ ranges over $C$.

Very informally, for $C = \text{Reg}, L ⊆ Δ^*$ describes transition sequences, $η$ the effect of each transition encoded as a word over $U_d$, and $α$ its label in $Σ$, resulting in $L_{ηα}$ 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 $S = (Δ, R, E)$ can be an amalgamation system supporting wqo decorations and accepting a language $L ∈ C$, and let $η$ and $α$ be morphisms as in Definition 5.4. Our goal is to define an amalgamation system $S_{ηα}$ that supports wqo decorations such that $L(S_{ηα}) = L_{ηα}$.

We decorate runs $ρ ∈ R$ with pairs of counter valuations from $P \subseteq \mathbb{N}^d × \mathbb{N}^d$. Consider a $P$-decorated run $(ρ, w)$ and let $a₁ · · · aₙ = yield(ρ)$ be the word accepted by $ρ$. Then $w = (u₁, v₁) · · · (uᵦ, vᵦ)$. We say that $(ρ, w)$ is coherent if $vᵢ = uᵢ + δ(η(αᵢ))$ for all $i ∈ [1, n]$. By $vᵢ = uᵢ + δ(η(αᵢ))$ for all $i ∈ [1, n]$. We say that $(ρ, w)$ is accepting if it is coherent and additionally the initial counters are $u₁ = 0$ and the final counters are $vᵦ = 0$.

Let $R_η$ be the set of accepting decorated runs in $\text{Deco}^P(R)$. If the canonical decomposition of a run $ρ ∈ R$ is $a₁ · · · aₙ$ with each $αᵢ ∈ Δ_ε$, then the canonical decomposition of a decorated run $(ρ, w) ∈ R_η$ is defined as $η(α₁) · · · η(αₙ)$, with each $η(αᵢ) ∈ Σ$ by definition of $η$. Let $S_{ηα} = ((Σ, R_η, P), α \circ \eta \circ \text{can})$. 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 $L(S_{ηα}) = L_{ηα}$.

In order to complete the proof of Theorem 5.5, it remains to show that $S_{ηα}$ 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 $S_{ηα}$ is an amalgamation system.

Proof. We show that $S_{ηα}$ satisfies the conditions of Section 3.1.2.

**Composition.** As we have defined above, the admissible embeddings in $E^P((ρ, u), (σ, v))$ of two runs in $R_η$ are those of $E(ρ, σ)$ that respect the ordering on decorations in $P$. These embeddings compose, because the admissible embeddings of $S$ compose and because additionally $≤_P$ on $P \subseteq \mathbb{N}^d × \mathbb{N}^d$ is transitive.

**Well-quasi-order.** Because we assume $S$ supports wqo decorations and $(P, ≤_P)$ is a wqo, $(\text{Deco}^P(R), ≤_P)$ is a wqo, and the induced $(R_η, ≤_P)$ as well.

**Amalgamation.** The construction mirrors the one presented in Section 3.3.3. Let $(ρ₀, w₀) ∈ R_η$ be run with $a₁ · · · aₙ$ the accepted word of $ρ₀$ and $w₀ = (u₁, v₁) · · · (uᵦ, vᵦ)$. Assume $(ρ₀, w₀) ≤_P (ρ₁, w₁)$ and $(ρ₀, w₀) ≤_P (ρ₂, w₂)$ for $(ρ₁, w₁), (ρ₂, w₂) ∈ R_η$. By definition of our decorated embedding, we have $ρ₀ ≤_P ρ₁$ and $ρ₀ ≤_P ρ₂$, and for each $i ∈ [1, n]$ there exists $cᵢ, dᵢ ∈ \mathbb{N}^d$ such that the $f(𝑖)$th pair in $w₁$ is $(uᵢ + cᵢ, vᵢ + cᵢ)$ and the $g(𝑖)$th pair in $w₂$ is $(uᵢ + dᵢ, vᵢ + dᵢ)$. Because $C$ is an amalgamation system, there exists a run $p₃ ∈ R$ with $ρ₁ ≤_P ρ₃, ρ₂ ≤_P ρ₃$, and $ρ₀ ≤_P ρ₃$ with $h = f ⊆ δ\rceil_{uᵦ}$.
where letters from \( L \) assume wlog. that all words in each extension \( G \). We are going to show that, if \( C \) is a class of languages that supports concatenative amalgamation condition, then \( \rho' \) is also defined by the so-called “extended” context-free grammars. Observe that in \( C \), we have \( u_i + c_i + d_i + \delta(\eta(c_i)) = u_i + c_i + d_i \) as desired. Consider now the \( \delta \) gap \( G_i,j_k \) and assume wlog. that \( G_i,j_k = G_{i,j} \). Observe that in \( w_1 \), we have \( u_i + c_i + d_i + \delta(\eta(c_i)) = u_i + c_i + d_i \). Similarly, in \( w_2 \) we have \( u_i + c_i + d_i + \delta(\eta(c_i)) = u_i + c_i + d_i \). Then \( u_i + c_i + d_i + \delta(\eta(c_i)) = u_i + c_i + d_i + d_i \) and by monotonicity we can decorate \( w_3 \) coherently along \( G_{i,j} \) by adding \( d_i \) to all the pairs from \( w_1 \) along that segment, and then \( u_i + c_i + d_i + \delta(\eta(c_i)) = u_i + c_i + d_i + d_i \) as desired and again by monotonicity we can decorate \( w_3 \) coherently along \( G_{i,j} \) by adding \( c_i + d_i \) to all the pairs from \( w_2 \) along that segment. The produced decorated run \( (\rho_1, w_3) \) is coherent by construction and satisfies \( (\rho_1, w_1) \leq \rho \), \( (\rho_3, w_3) \) and \( (\rho_2, w_2) \leq \rho \). Additional letters in the case of \( i = 0 \) we additionally have \( c_0 = d_0 = 0 \), and analogously in the case of \( i = n \) we have \( c_{n+1} = d_{n+1} = 0 \), thus \( (\rho_3, w_3) \) is also accepting. □

**Claim 5.8.** If \( S \) supports wqo decorations, then so does \( S_{\eta,\tau} \).

Proof. Observe that a decoration of a run \( (\rho, w_1 \cdots w_n) \in R_q \) 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 \( \exists d \times \exists d \times X \).

### 5.3 Algebraic Extension

We introduce now, as a generalisation of the 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 \( G = (N, \Sigma, S, \{ L_A \}_{A \in \mathbb{N}} \) where \( S \in N \) and each \( L_A \) is a language from \( C \) with \( L_A \subseteq (N \cup \Sigma)^* \). We write that \( wA \Rightarrow uwo \) if \( w \in L_A \). The language of a \( C \)-grammar is the set \( \{ w \in \Sigma^* | S \Rightarrow w \} \). For example, every context-free grammar can be seen as a Fin-grammar, where \( \text{Fin} \) 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 \( \text{Alg}(C) \) 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 \( \text{Alg}(C) \).

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

Let us fix a \( C \)-grammar \( G \) and write \( M_A \) 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 \( \Sigma \) 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 \( \varepsilon \) or with pairs of non-terminals \( A \) and runs \( \rho \) of \( M_A \). We call \( \rho \) the explanation of the expansion of the non-terminal \( A \) at this step. More specifically, if \( \rho \) is a run in \( M_A \) with yield(\( \rho \)) \( \in \Sigma^* \), then \( (A \rightarrow \rho)[\cdot] \) is a tree. Otherwise, let \( X_1X_2 \cdots X_n \) be the projection of yield(\( \rho \)) to \( N \) and \( t_1, \ldots, t_n \) be \( X_1, \ldots, X_n \)-rooted trees. Then \( (A \rightarrow \rho)[\{t_1, \ldots, t_n\}] \) is a tree as well. For the remainder of this section, if we write yield(\( \rho \)) or can(\( \rho \)) = \( u_0X_1u_1 \cdots X_nu_n \), we assume that \( X_i \in N \) and \( u_j \in \Sigma^* \). We write \( \rho_{\eta}(i) \) for the map \( [1, n] \rightarrow [1, |\rho_{\eta}|] \) that associates to every \( X_i \) its position in the canonical decomposition.

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

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

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

This also yields the expected definition of yield(\( \cdot \)), being yield(\( \tau \)) = yield(\( \rho \)) if \( \tau \) is a leaf node, and yield(\( \tau \)) = \( u_0 \text{yield}(t_1)u_1 \cdots \text{yield}(t_n)u_n \). Otherwise we write \( T(G) \) for the set of all the trees of \( G \), and \( R(G) \) for the \( S \)-rooted ones. Then \( L(G) = \bigcup_{\rho \in R(G)} \text{yield}(\rho) \) 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 \( t_1 = (A \rightarrow \rho)[\ldots] \) to \( t_2/\rho = (A \rightarrow \sigma)[\ldots] \), we require that \( \rho \) embeds into \( \sigma \) such that the corresponding subtrees also embed. Formally, let \( t_1 \) and \( t_2 \) be trees from \( T(G) \). Denote the run embeddings between runs of the various systems
be treated similarly. As this exhausts all the potentially decidable graph monoids and $\mathcal{V}(M)$ is always a full trio \cite[Thm. 4.1]{main}, 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?

REFERENCES

\begin{thebibliography}{1}


\bibitem{7} Jean Berstel. 1979. Transductions and Context-Free Languages. Teubner. \url{https://doi.org/10.1007/978-3-663-85367-1}


\bibitem{13} Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. 2016. The diagonal problem for higher-order recursion schemes is decidable. In Proc. LICS 2016. ACM, 96–105. \url{https://doi.org/10.1145/2933757.2934527}


\end{thebibliography}
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: zero₁ and zero₂, which test the first, resp. second, counter for zero, with the obvious semantics.

Given a two-counter machine \((Q, q₀, Δ, q_f)\), where \(Q\) is the set of states, \(q₀\) is the initial state, finite transition set \(Δ \subseteq Q \times (\mathbb{Z}^₂ \cup \{\text{zero}_1, \text{zero}_2\}) \times Q\), and final state \(q_f \in Q\), it is well-known to be undecidable whether there is a run from the configuration \((q₀, 0, 0)\) to \((q_f, 0, 0)\).

Given a two-counter machine as above, we define a 2-VASS with state set \(Q\), initial state \(q₀\), and final state \(q_f\) as follows. The input alphabet is \(Σ = \{z₁, z₂\}\) and it has the following transitions:

\[
\begin{align*}
(p, u, ε, q) & \quad \text{for every transition } (p, u, q) \text{ in } A \\
(p, (0, 0), z₁, q) & \quad \text{for every transition } (p, \text{zero}_1, q) \text{ in } A \\
(p, (0, 0), z₂, q) & \quad \text{for every transition } (p, \text{zero}_2, q) \text{ in } A.
\end{align*}
\]

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 \emph{faithful} if all the transitions labelled \(z₁\) or \(z₂\) 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:

\[
\begin{align*}
((x₁, x₂), t, (x₁, x₂)) & \quad \text{where } t \text{ is labelled by } z₁ \text{ and } x₁ > 0, \text{ or} \\
((x₁, x₂), t, (x₁, x₂)) & \quad \text{where } t \text{ is labelled by } z₂ \text{ and } x₂ > 0
\end{align*}
\]

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) ⇒ (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 \(L\) is a directed regular language \(D\). The difficult part of this computation is to decide whether our input language \(L\) satisfies \(D \subseteq L\). Our algorithm uses a strategy from [62], 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 \(Σ_d = \{0, d\}\) 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 \(u, v \in Σ_d^*\) and \(m \in Σ_d\), \(m > 0\), is the largest letter occurring in \(u\) and \(v\), then we define \(u ≲ v\) if and only if

\[u = u₀m₁ \cdots m_k, \quad v = v₀m₁ \cdots m_l\]
with $u_0, \ldots, u_k, v_0, \ldots, v_r \in \Sigma_{m-1}^*$ and there is a strictly monotone map $\varphi : [0,k] \to [0,l]$ with $u_i \lessdot_{\varphi(i)} v_i$ for every $i \in [0,k]$ (in particular, $k = 0$ means that $m$ does not occur in $u$). Thus, $\lessdot$ is defined recursively w.r.t. the occurring priorities. To cover the base case, if $u, v \in \Sigma^*$, then we simply have $u \lessdot v$ if and only if $|u| \leq |v|$. For a word $w_1 w_2 \cdots w_q m$, 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 $0 \lessdot 00 \not\lessdot 010 \lessdot 1010$, but $010 \not\lessdot 1010$.

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

$$L|_S = \{ u \in \Sigma_d^* \mid \exists v \in L : u \lessdot_S v \}.$$  

The task of computing an NFA for $L|_S$ 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 $L \subseteq \Sigma^*$, $L|_S = TL$.

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

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

The starting state is $0_{\text{skip}}$, 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 $S_k = \{ l_{\text{out}}, l_{\text{skip}} \mid l \leq k \} \cup \{ t \}$ 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 $k_{\text{out}}$ 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 $y_{\text{skip}}$ 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 \((\Sigma_d, \leq_S)\) are precisely the sets in \(\text{Ideal}_d\), where

\[
\begin{align*}
\text{Atom}_d &= \{ld \cup r \mid l \in \text{Ideal}_{d-1}\} \cup \{(Dd)^* \mid D \in \text{Down}_{d-1}\}, \\
\text{Ideal}_0 &= \{0^n \mid n \in \mathbb{N}\} \cup \{\varepsilon\}, \\
\text{Ideal}_d &= \{X_1 \cdots X_nA \mid X_i \in \text{Atom}_d, A \in \text{Ideal}_{d-1}\},
\end{align*}
\]

and \(\text{Down}_d\) is the set of all downward closed subsets of \(\Sigma_d^*\) 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 \(0^*\varepsilon\) or \(0^n\varepsilon\). A pseudo-ideal of priority \(d > 0\) is an ideal of the form \((Id)^*\) or \(I_d \cup \cdots \cup I_n\) where \(I_1, \ldots, I_n\) 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 \(C\) of languages closed under rational transduction, we can construct finitely many pseudo-ideals \(J_1 \subseteq \Sigma^*\) 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 \(J_1 \subseteq TL\downarrow S\).

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 \(l = (Dd)^*\) with \(D = I_1 \cup \cdots \cup I_n\) by a new term \(J = ((I_1d \cdots I_nd)\varepsilon)^*\), where \(\varepsilon > d\) is a fresh priority. The transduction modifies the words in \(L\) so that after every occurrence of \(d\), an occurrence of \(\varepsilon\) 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 \((0^*1)^*\varepsilon\) 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 \(\rho_0 \subseteq \rho_1 \subseteq \rho_2\) such that (i) some gap of \(\rho_0 \subseteq \rho_1\) between some positions \(i < j\) of \(\rho_1\) contains a 1 and (ii) some gap of \(\rho_1 \subseteq \rho_2\), which is between \(i\) and \(j\) in \(\rho_1\), belongs to \(0^*\). Notice that then, by amalgamating \(\rho_2\) with itself above \(\rho_1\) again and again, we can create arbitrarily long blocks of contiguous \(0\)'s. The resulting run \(\rho_2'\) still embeds \(\rho_1\) and thus \(\rho_0\), such that one gap of \(\rho_2'\) in \(\rho_0\) contains both our long block of \(0\)'s and also a \(1\). This means, if we amalgamate \(\rho_2'\) again and again above \(\rho_0\), we obtain arbitrarily many blocks of our long 0-blocks. This yields runs that cover all words in \((0^*1)^*\varepsilon\).

**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 \(\Sigma\). We call a word \(w'\) a factor of \(w = w_1 \cdots w_n \in \Sigma \cup \{\varepsilon\},\)
if \( w[1] \in \Sigma^* \Sigma^* \), i.e., \( w' \) is an infix of restriction of \( w \) over \( \Sigma \). For a word \( w = w_0 w_1 \ldots w_n \), by \( w[i,j] \) we denote the word \( w_i \ldots w_j \) for \( i < j \).

A finite subset \( \mathcal{W} \) of runs \( R \) is called an \( l \)-witness if

- for \( I = \emptyset \leq^k \) for some \( k \in \mathbb{N} \), there exists a run \( \rho \in \mathcal{W} \) such that \( 0^k \) is a factor of \( \text{can}(\rho)[\overline{I}, \overline{I}] \), for some \( 0 \leq \overline{I}, \overline{I} \leq |\rho|_{\text{can}} \).
  Then \( \rho \) is said to witness \( I \) between \( \overline{I} \) and \( \overline{I} \).
- for \( I = 0^s \), there exist runs \( \rho, \psi \in \mathcal{W} \) such that \( \psi \triangleleft_{f} \rho \) and \( G_{i,f} = 0^t \) for some \( l > 0 \), \( i \in [0,|\psi|_{\text{can}}] \) and \( f \in E(\psi, \rho) \).
  Then \( \rho \) is said to witness \( I \) between \( \overline{I} \) and \( \overline{I} \), if \( G_{i,f} \) is a factor of \( \text{can}(\rho)[\overline{I}, \overline{I}] \).

With this notion of \( l \)-witnesses, we can prove:

**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 \( l \)-witness.

To illustrate the proof idea, let us see how to show that there is always a witness for \( I = (0^s 1^t)^{\ast} \). Suppose we have an amalgamation system \( S \) for \( L \downarrow S \) and we know \( I \subseteq L \downarrow S \). 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 \( w_i = (0^s 1^t)^{\ast} \) for \( i \geq 1 \). Then \( w_1, w_2, \ldots \) \( \in \downarrow L \downarrow S \), so there must be runs \( \rho_1 \) and \( \rho_2 \) with \( \rho_1 \triangleleft \rho_2 \) with \( \text{can}(\rho_1) = w_1 \) and \( \text{can}(\rho_2) = w_f \). Then clearly, every non-empty gap of \( \rho_2 \) in \( \rho_1 \) belongs to \( 0^s \). Moreover, any embedding of a minimal run \( \rho_0 \) of \( S \) into \( \rho_1 \) will have some gap containing two \( 1 \)'s, and thus have \( 10^t 1 \) as a factor. Then the runs \( \rho_0, \rho_1, \rho_2 \) clearly constitute an \( l \)-witness. We will show this in Appendix B.2.

**The algorithm.** We now have all the ingredients to show that decidability 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 \( L \downarrow S \) for a given language \( L \). Second, according to Proposition B.5 and Lemma B.6, deciding whether \( I \subseteq L \downarrow S \) for an ideal \( I \) is recursively enumerable. Therefore, we proceed as follows. We enumerate all finite unions \( I_1 \cup \cdots \cup I_\alpha \) of ideals and try to establish the inclusion \( I_1 \cup \cdots \cup I_\alpha \subseteq L \downarrow S \). Once we find such a finite union where inclusion holds, we check whether \( L \downarrow S \subseteq I_1 \cup \cdots \cup I_\alpha \). The latter is decidable: By closure under rational transductions, we can construct an amalgamation system for \( L \downarrow S \cap (\Sigma^* \setminus (I_1 \cup \cdots \cup I_\alpha)) \) 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 \( I_1 \cup \cdots \cup I_\alpha \) with \( L \downarrow S = I_1 \cup \cdots \cup I_\alpha \).

Finally, note that \(^{(6)} \Rightarrow (7)^{\ast} \) holds as well, because \( L \neq \emptyset \) if and only if \( L \downarrow L \neq \emptyset \), meaning we decide emptiness of \( L \) by computing an NFA for \( L \downarrow L \) 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 \( u \triangleleft_{B} v \), if

i. if \( u, v \in \Sigma_{np} \), and \( u \triangleleft v \) (\( u \) is subword smaller than \( v \)), or
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 \( u_i, v_i \in \Sigma_{=p-1} \) (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 \textit{witness block map}, such that

(a) \( u_i \leq_B v_{\varphi(i)}, \forall i \),

(b) \( \varphi(0) = 0 \),

(c) \( \varphi(n) = m \), and

(d) \( x_i \leq_{\varphi(i)} y_{\varphi(i)} y_{\varphi(i)+1} \cdot \cdot \cdot y_{\varphi(i+1)}, \forall i \in [0, n-1] \).

Intuitively, we say that \( u \) is \textit{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 iib, 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 only simple block downward closures can be computed.

Since \( u \leq_B v \implies u \leq_S v \), 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 \( \Sigma' = \{ 0', 0'', \ldots, d', d'' \} \) such that \( 0 < 0' < 0'' < 1 < 1' < 1'' < \cdot \cdot \cdot < d < d' < d'' \). By \( B \), we denote \( \Sigma' \cup \Sigma \). Let for a word \( w = w_1 w_2 a_1 \cdots a_n \), where \( w_i \in \Sigma_{=p-1} \) and \( a \) is the highest priority letter in \( w \), by \( \text{border}(w) \) we define the word \( b_1 \cdot \text{border}(w_1) \cdot b_1 \cdot a \cdot \text{border}(w_2) \cdot a \cdot \cdot \cdot a \cdot \text{border}(w_n - 1) \cdot a \cdot h_n \cdot \text{border}(w_n) \cdot h_n \), where \( b_1 \) and \( h_n \) is \((a-1)'\) if \( w_0 = \epsilon \) (resp., \( w_n = \epsilon \)), else it is \((a-1)''\). Moreover, \( \text{border}(w) = w \) if \( w = 0' \). Intuitively, we bound the first and the last \( a \) blocks of the word recursively, and we also distinguish whether these blocks are \( \epsilon \) or not.

Let \( B \downarrow_B = \{ \text{border}(w) \mid w \in L \} \). Then we claim that \( L \downarrow_B = (L \downarrow_S \cap \mathcal{R}_d)(L) \), where \( \mathcal{R}_d \) is defined recursively as follows.

\[
\mathcal{R}_0 = 0^* \\
\mathcal{R}_d = (a^*e^*a^* + a^*a_{=1}a') \cdot (a^*e^*a^* + a^*a_{=1}a')
\]

Note that \( L' = \text{border}(L) \downarrow_S \cap \mathcal{R}_d \) is the language of simple block smaller words where \( 0'', 0' \cdot \cdot \cdot a'' \cdot d'', \cdot d'' \) are not dropped. Furthermore, the \( L \downarrow_S \) is restriction of words in \( L \) to letters in \( \Sigma \).

Now we prove the claim. For one direction suppose \( u \in L \downarrow_B \). Then there exists a word \( v \in L \) such that \( u \leq_B v \). 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 \( \text{border}(u) \leq_S \text{border}(v) \).

For the base case, i.e. \( d = 0 \), since \( \text{border}(u) = u \) for any \( u = 0' \), this trivially holds.

Then for the induction step, let the statement holds true for some \( d-1 \in \mathbb{N} \). Then assuming \( u = u_1 d_{i-2} \cdots d_{i-1} u_1 \) and \( v = v_1 d_{i-2} \cdots d_{i-1} v_1 \), there exists a \( \leq_B \) witness \( \varphi : [1, k] \to [1, l] \), with \( \varphi(1) = 1 \) and \( \varphi(k) = l \). Then we show that \( v \) is also a witness for \( \text{border}(u) \leq_S \text{border}(v) \).

Let \( \text{border}(u) = u_1' d_{i-2}' \cdots d_{i-1}' u_1' \) and \( \text{border}(v) = v_1' d_{i-2}' \cdots d_{i-1}' v_1' \). Induction hypothesis immediately implies that \( u_1' \leq_S \varphi(i) \cdot \varphi(i) \cdot \cdot \cdot \varphi(i) \cdot \epsilon \cdot \cdot \cdot \epsilon \cdot \cdot \cdot \epsilon \) (argument is analogous for the last \( d \) block). If \( u_1' = d' \cdot d'' \), then \( \epsilon = d' \cdot d'' \), since \( \epsilon \) is only block smaller than \( d \). Then clearly, \( u_1' \leq_S v_1' \). Now if \( u_1' = d' \cdot d'' \), where \( x \in \mathcal{R}_{d-1} \), then \( v_1' = d' \cdot d'' \), and \( y \in \mathcal{R}_{d-1} \). Again by induction hypothesis, \( x \leq_S y \), and hence \( u_1' \leq_S v_1' \). Hence, \( \text{border}(u) \leq_S \text{border}(v) \).

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

- \( v' = \text{border}(u') \), \( v = \text{border}(u) \).
Then we need to show that \( u \preceq_B u' \); 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 \( u' \preceq_B u \). Since \( v \preceq_S v' \), there exists a witness \( \varphi \). We show that \( \varphi \) is a witness for \( u \preceq_B u' \), by induction on \( d \). The base case is trivial as \( \text{border}(x) = x \) when \( x \neq 0' \). Now, for the induction step let the highest letter in \( u \) and \( u' \) be \( d \).

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 \( d'' \) 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 \( v = v_1 d \cdots d_{k-1} v_k \) and \( v' = v'_1 d' \cdots d'_{k-1} v'_k \), we have that \( v_1 \preceq_S v'_1 \), then \( v_1 \mid_S \preceq_B v'_1 \mid_S \). Similarly, \( v_k \mid_S \preceq_B v'_k \mid_S \). For other block, the induction hypothesis, immediately implies \( v_i \preceq_B v_{\varphi(i)} \). Then \( \varphi \) is a witness for \( u \preceq_B u' \). Hence, \( u \in L \downarrow_B \). This completes the proof of the lemma.

\[ \square \]

\subsection*{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.

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

\begin{itemize}
  \item downward closed, i.e. if \(u \in I\) and \(v \leq u\), then \(v \in I\), and
  \item up-directed, i.e. \(\forall u, v \in I, \exists z \in I\) such that \(u \leq z\) and \(v \leq z\).
\end{itemize}

\textbf{Lemma B.7.} Let \(J\) be a subset of ideals, such that

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

Then \(J\) is the set of all the ideals.

\textbf{Lemma B.4.} The ideals of \((\Sigma_d, \leq)\) are precisely the sets in \(\text{Ideal}_d\), where

\[\begin{align*}
\text{Atom}_d &= \{ (D \cup \varepsilon) \mid I \in \text{Ideal}_{d-1} \} \cup \{(Dd)^* \mid D \in \text{Down}_{d-1}\}, \\
\text{Ideal}_0 &= \{ \emptyset \mid n \in \mathbb{N} \} \cup \{ \emptyset \}, \\
\text{Ideal}_d &= \{ X_1 \cdots X_n A \mid X_i \in \text{Atom}_d, A \in \text{Ideal}_{d-1} \},
\end{align*}\]

and \(\text{Down}_d\) is the set of all downward closed subsets of \(\Sigma_d\) with respect to simple block order.

**Proof.** Let \(\text{Ideal}_d\) be the set of all the ideals with respect to simple block order for alphabet \(\Sigma_d = \{ 0, a \}\). We first show that every set in \(\text{Id}_d\) is an ideal, i.e., \(\text{Id}_d \subseteq \text{Ideal}_d\).

Observe that the elements of \(\text{Id}_d\) are indeed ideals, because for singletons, the subword order and the simple block order coincide. So, now assume that \(\text{Id}_{d-1} = \text{Ideal}_{d-1}\).

We now show that \(\text{Id}_d \subseteq \text{Ideal}_d\). Let \(X = (X_1 \cdots X_n A) \in \text{Id}_d\). 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 \(v = v_1 a_1 v_2 \cdots \) with \(v \preceq_S u\). Then there exists a strictly monotonically increasing map \(\varphi\) that maps \(a\) blocks of \(v\) to those of \(u\). Then consider the map \(\psi : [1, k] \rightarrow \{1, \ldots, n, A\}\) that maps each \(a\) block \(v_j \) to \(j\) if \(\varphi(i)\)-th \(a\) block of \(u\) lies in \(u_j\). Note that this map is well defined as each \(u_j\) 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 \(v_i \cdots v_{q-1} a\) such that \(\psi(i) = \psi(j) = p\) is in \(X_p\), by induction hypothesis. Now suppose \(S = \{ i | \psi(i) = i\), for some \(j\}\). Then \(v \in X_i \cdots X_q \subseteq X_1 \cdots X_n A\), where \(S = \{ i_1, \ldots, i_q \}\).
Upward directed. Let \( u, v \in X \). Suppose \( u = u_1 \ldots u_n u_A \) and \( v = v_1 \ldots v_n v_A \), such that \( u_i, v_i \in X_i \) and \( u_A, v_A \in A \). Construct the word \( z = z_1 \ldots z_n z_A \) as follows

\[
\begin{align*}
z_i &= \begin{cases} u_i v_i, & \text{if } X_i = (Da)^* \\ w_i, & \text{if } X_i = 1a \text{ and } u_i, v_i \leq_B w_i \end{cases} \\
z_A &= w_A, \text{ where } u_A, v_A \leq_B w_A
\end{align*}
\]

It is again easy to notice that \( u, v \leq_B z \). Hence, \( X \) is up-directed.

We have shown above that \( Id_A \subseteq \text{Ideals}_A \). To show that \( Id_A = \text{Ideals}_A \), we use the Lemma B.4, and show that \( Id_A \) 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 \( \Sigma_A^* \setminus u \uparrow \) 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, \( u \in \Sigma_A^{a-1} \). Then \( \Sigma_A^* \setminus u \uparrow = \Sigma_A^{a-1} \setminus u \uparrow = \cup_{i=1}^{k} I_i \), 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 \( oaw \) where \( o \) from \( \Sigma_A^{a-1} \) and \( w \) has \( n-1 \) many \( a \) blocks.

Claim B.8.

\[
\Sigma_A^* \setminus oaw \uparrow = (\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) \cup (\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) (\Sigma_A^* \setminus w \uparrow) \downarrow_S
\]

Proof of Claim B.8. (\( \subseteq \)) Let \( z \in \Sigma_A^* \setminus oaw \uparrow \). Then \( u = oaw \not\in S \), and we have the following two cases,

1. Case 1: If \( o \) can not be mapped to a \( a \) block of \( z \), then all the \( a \) blocks of \( z \) come from \( (\Sigma_A^{a-1} \setminus o \uparrow) \). Hence,

\[
z \in ((\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow)) \subseteq \text{RHS}.
\]

2. Case 2: If \( o \) 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 \( (\Sigma_A^{a-1} \setminus o \uparrow) \), and the \( i \)th block comes from \( \Sigma_A^{a-1} \). This implies that \( z_1 a z_2 a \cdots a z_k a \in ((\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus a) \).

Since \( u \not\in S \) \( z \) and \( va \not\in S \) \( z_1 a z_2 a \cdots z_i a \), \( w \not\in S \) \( z_{i+1} a \cdots a z_n \), i.e. \( w \in (\Sigma_A^* \setminus w \uparrow) \).

Hence,

\[
z \in ((\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow)) \subseteq \text{RHS}.
\]

Hence, \( \text{LHS} \subseteq \text{RHS} \).

(2) This containment can be seen by going backwards in the arguments for the other containment.

Since \( (\Sigma_A^{a-1} \setminus o \uparrow) \) is a finite union of ideals from \( Id_{a-1} \),

\[
(\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) = (\cup_i I_i a) (\cup_i I_j)
\]

where \( I_i, I_j \in Id_{a-1} \). Since \( I_i a I_j \) is an element in \( Id_A \), \( \Sigma_A^{a-1} \setminus o \uparrow \) is a finite union of ideals from \( Id_{a-1} \).

Using similar arguments, it can be shown that

\[
(\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) a^* (\Sigma_A^{a-1} \setminus o \uparrow) w \uparrow) \downarrow_S
\]

is also a finite union of ideals from \( Id_{a-1} \).

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 \( \text{Atom}_A = \text{Atom}_A \cup 1aI \cap Id_{a-1} \). When \( \Sigma = \{0\} \), the ideals are of the form \( 0^* \) or \( \{0^* \mid 0 \leq i \leq n) \) for some \( n \), and the intersection of two ideals can only be another ideal.

Suppose that the statement holds for some \( \Sigma_{a-1} \). Then let \( I_1 = X_1X_2 \cdots X_k A \) and \( I_2 = Y_1Y_2 \cdots Y_l B \) 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 $I_1 = A \in \Id_{d-1}$ and $I_2 = X_1B$ where $X_1 \in \Atom_d$ and $B \in \Id_d$.

Then $I_1 \cap I_2 = (A \cap B) \cup (A \cap B_1)$, where $X_1 = B_1a$ or $X_1 = (B_1a)^*$. But both intersections are finite union of ideals.

Then suppose the statement holds for all ideals $I_1 = X_1X_2 \cdots X_kA$ and $I_2 = Y_1Y_2 \cdots Y_lB$, 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 $I_1 = X_1X_2 \cdots X_kX_{k+l}A$ and $I_2 = Y_1Y_2 \cdots Y_lB$. To reduce notational clutter, we write $I_1 = X_1XA$ and $I_2 = Y_1YB$, 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 $X_1 = A_1a$ and $Y_1 = B_1a$, then $I_1 \cap I_2 = (A_1 \cap B_1)a(XA \cap YB)$.

2. if $X_1 = A_1a$ and $Y_1 = (B_1a \cup \{e\})$, then $I_1 \cap I_2 = (A_1aXA \cap B_1aYA) \cup (A_1aXA \cap YB) = ((A_1 \cap B_1)a(XA \cap YB) \cup (A_1aXA \cap YB))$.

3. if $X_1 = A_1a$ and $Y_1 = (B_1a)^*$, then $I_1 \cap I_2 = I_1 \cap (YB \cup B_1al_2) = (I_1 \cap YB) \cup (I_1 \cap B_1al_2) = (I_1 \cap YB) \cup ((A_1 \cap B_1)a(XA \cap I_2))$.

4. if $X_1 = (A_1a \cup \{e\})$ and $Y_1 = (B_1a \cup \{e\})$, then $I_1 \cap I_2 = [(A_1 \cap B_1)a(XA \cap YB) \cup (A_1aXA \cap YB) \cup (XA \cap B_1aYB) \cup (XA \cap YB)]$.

5. if $X_1 = (A_1a \cup \{e\})$ and $Y_1 = (B_1a)^*$, 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 $X_1 = (A_1a)^*$ and $Y_1 = (B_1a)^*$, then $I_1 \cap I_2 = (XA \cup A_1al_1) \cap (YB \cup B_1al_2) = (XA \cup YB) \cup (XA \cap B_1al_2) \cup (A_1al_1 \cap YB) \cup (A_1al_1 \cap B_1al_2)$.

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 $\Id_d = \Ideals_d$.

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 $J_j \subseteq \Sigma^*_d$ 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 $\cup J_j \subseteq TL \downarrow_S$.

**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 $I = X_1X_2 \cdots X_nA$ flat, if $X_i$ is of the form $Id$ or $(1d \cdot \cdots \cdot dl_d)^*$. 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 $\cup J_j \subseteq L \downarrow_S$, for some language $L'$.

Let $I = X_1 \cdots X_nA$ be an ideal, where $X_i \in \Atom_{d-1}, A \in \Id_{d-1}$. First, since some atoms are of the form $(I'_j \cup \{e\})$, by distributivity of concatenation over union, we have that $I$ is finite union of sets $J_j$ of the form $Y_1 \cdot Y_2A$, where $k \leq n$ and $Y_k$'s are of form $Id$ or $(Dd)^*$. Then $I \subseteq L \downarrow_S$ if and only if $\cup J_j \subseteq L \downarrow_S$ for all $j$. For each $J_j$, we give construct a flat ideal. So for the purpose of making a notation, we assume that $X_i$'s are of the from $Id$ or $(Dd)^*$. Moreover, since $I \subseteq L \downarrow_S$ if and only if $\cup Id \subseteq L \downarrow_S$, 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 $(l_1d \cdots l_d)^*$, where $D = \cup_{\{1 \leq k \leq l\}} I_j$ is the ideal decomposition of $D$. Suppose we obtain $I'$ by replacing such $X_i$'s, 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 $(l_1d \cdots l_d)^*$.

Now consider the alphabet $\Sigma'_d = \{i, i | i \in \Sigma\}$, such that $0 < \bar{0} < 1 < \cdots < d < \bar{d}$. Consider a transducer $T_i$ that arbitrarily adds a $\bar{a}$ after an occurrence of $i$. Then consider the ideal $I' = X_1 \cdot \bar{d} \cdots \bar{d} \cdot X_n \cdot \bar{d}$ and consider the language $L' = T_1 \cdots T_dL$. Since $C$ is closed under rational transduction, $L' \in C$.

By induction on highest letter $a$, 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 \downarrow \text{\( x \)} \), and \( u = u_1 \overline{1} \cdots u_k \overline{1} \in I' \), where \( u_i \in \mathfrak{S} \). Then \( u' = u_1 \cdots u_k \in I \), and there exists \( v' \in I \) such that \( u' \) embeds in a 2-block of \( v' \) with witness map \( \varphi \). Then on adding a \( \overline{1} \) after every \( \overline{1} \) block of each \( u_i \) embeds to, we obtain \( v \), we get that \( u \preceq \downarrow \text{\( x \)} v \).

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

Now we assume that for some \( a - 1 \), \( I \subseteq \downarrow \text{\( x \)} I \) if and only if \( I' \subseteq \downarrow \text{\( x \)} I \), where \( L' = T_1 \cdots T_{a-1} I \). Now suppose highest letter in \( I \) is \( a \), and \( L' = T_a \cdots T_L \). Also, let \( I' = X_1 \overline{1} \cdots \overline{1} \overline{X} \overline{1} X_2 \cdots \overline{1} \). We first show that \( I \subseteq \downarrow \text{\( x \)} I \) iff \( I' \subseteq \downarrow \text{\( x \)} I \).

First, let \( I' \subseteq \downarrow \text{\( x \)} I \), and \( u = u_1 \overline{1} \cdots u_k \in I \), where \( u_i \in X_i \). Then \( v = u_1 u_2 \cdots u_k \in I \). Then there exists \( \varphi' \in \downarrow \text{\( x \)} I \) such that \( v \preceq \downarrow \text{\( x \)} \varphi' \), with a witness map \( \varphi \). Consider \( u' = v_1 \overline{1} \cdots \overline{1} \varphi v_k \) such that \( v_1 \in \Sigma_d \cdot w_1 \cdot \overline{1} \) where \( w_1 \) is the \( \varphi' \)-th \( \overline{1} \)-block of \( u \). Note that \( v' \preceq \downarrow \text{\( x \)} v \) and \( u \preceq u' \), since \( u_i \preceq v_i \).

Now, suppose that \( I' \subseteq T_a \downarrow \text{\( x \)} I \). Then let \( u = u_1 \cdots u_k \in I \), where \( u_i \in X_i \). Then \( v = u_1 \overline{1} \cdots \overline{1} \overline{X} \overline{1} v_k \) \( \in I' \), and hence there exists \( \varphi' = v_1 \overline{1} \cdots \overline{1} \overline{v} v_k \) \( \in \downarrow \text{\( x \)} I \) such that \( v \preceq \downarrow \text{\( x \)} v' \). Then since each \( \overline{1} \)-block in \( v \) recursively embeds in \( \overline{1} \)-blocks in \( v' \), the \( \overline{1} \)-blocks recursively embed too. Then \( u' = v_1 v_2 \cdots v_k \in L \) and \( u \preceq u' \).

Since \( X_i \)'s are ideals enclosed between \( \overline{1} \)s \( \in I' \), then by induction hypothesis \( X_i \subseteq \downarrow \text{\( x \)} T_a \downarrow \text{\( x \)} I \) if and only if \( I'' = X_i' \cdots X_{i-1}' \subseteq T_1 \cdots T_{a-1} \downarrow \text{\( x \)} I \), where \( X_i' \) is a flat ideal obtained by eliminating the downclosed closed sets in the Kleene stars. Hence, \( I \subseteq \downarrow \text{\( x \)} I \) if and only if \( I'' \subseteq T_1 \cdots T_{a-1} \downarrow \text{\( x \)} I \).

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 \((i_1 i_2 \cdots i_k d)^*\), for the simplicity of the proof, we show how we reduce from \((i_1 i_2 \cdots i_k d)^*\) to \((i d)^*\). The generalization is simple extension.

Consider the alphabet \( \Sigma' = \{i, j \mid i, j \in \Sigma\} \), such that \( 0 < 0 < 1 < 1 < \cdots < 1 < \cdots < 1 < \cdots < 1 \). Consider a transducer \( T_i \) that arbitrarily adds a \( \overline{1} \) after an occurrence of \( i \), and a transducer \( T_i' \) that in arbitrary \( \overline{1} \)-blocks, replaces every \( i \) with an \( j \), and adds a \( i \) after the final replacement in the \( \overline{1} \)-block. For example, on a word 0101010, one of the outputs of \( T_i \) is 010101010, and on this word, \( T_{i}' \) outputs 0101011010.

Let \( X = (i_1 i_2 \cdots i_k d)^* \), where \( i_k \)'s are flat ideals over \( \Sigma_{d-1} \), then consider \( X' = (i_1 i_2 \cdots i_k d)^* \). Note that \( X' \) is a pseudo-ideal. Then we show by induction over the highest letter in flat ideals, that \( X \subseteq \downarrow \text{\( x \)} I \) if and only if \( X' \subseteq T_{a} T_{a-1} \cdots T_{1} \downarrow \text{\( x \)} I \).

For the base case, when \( a = 1 \), then \( X = (i_1 \cdots i_k 1)^* \), where \( i_k \) is either \( 0^* \) or \( 0 \leq k \), and \( L' = (i_1 \cdots i_k 1)^* \). If \( X \subseteq \downarrow \text{\( x \)} I \), and \( u = u_1 \cdots u_k \in X \), where \( u_i \in \Sigma_{d-1} \). Then \( u' = u_1 \cdots u_k \overline{1} \), such that every \( i \) is replaced with \( 1 \), and the last \( 1 \) before every \( 1 \) is dropped. Then \( u' \in I \) and in \( u, v' \) is a word \( v' = v_1 v_2 \overline{1} \), where \( v_1 \) is a 2-block such that \( u \preceq v_1 \), with a witness map \( \varphi \). Let’s say \( u_i \) has \( i_j \) many \( 1 \)-blocks. Then on replacing all the \( 1 \)-blocks with \( 1 \)'s in \( v_1 \), except the ones that appear at \( s_1 + s_2 + \cdots + s_i \)-th \( 1 \)'s for every \( i \), and adding \( 1 \) before them, we get a word \( v \). Then it is easy to observe that \( u \preceq v \), since \( u \) embeds within \( v_2 \) map \( i \)-th \( \overline{1} \)-block of \( u \) to \( (i-1) \)-th \( \overline{1} \)-block of \( v_2 \), and recursively map \( \overline{1} \)-blocks respecting \( \varphi \).

If \( I' \subseteq T_{i} \downarrow \text{\( x \)} I \), then going backward in the argument above, we get that \( I \subseteq T_{i} \downarrow \text{\( x \)} I \).

Then for some \( a \), the argument is similar with induction hypothesis over flat ideals of smaller highest letter, with the observation that adding, removing \( i \) 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 \((i d_1 \cdots i d_2)^* \), i.e. every \( X_i \) is enclosed with highest priority letter in the ideal. Then within each \( \overline{1} \)-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 \( T_{a} T_{a-1} \cdots T_{1} \) to \( T_d \cdots T_1 \).

The two transformation reduce the containment problem of general ideals in a downward closed language to that for pseudo-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 \( W \) of runs \( R \) is called an \( I \)-witness if

- for \( I = 0^\leq k \) for some \( k \in \mathbb{N} \), there exists a run \( \rho \in \mathcal{W} \) such that \( 0^k \) is a factor of \( \text{can}(\rho)[\overleftarrow{T}, \overrightarrow{T}] \), for some \( 0 \leq \overleftarrow{T}, \overrightarrow{T} < |\rho|_{\text{can}} \).

Then \( \rho \) is said to witness \( I \) between \( \overleftarrow{T} \) and \( \overrightarrow{T} \).

- for \( I = 0^* \), there exist runs \( \rho, \psi \in \mathcal{W} \) such that \( \psi \preceq_f \rho \) and \( G_{i,f} = 0^l \) for some \( l > 0 \), \( i \in [0, |\psi|_{\text{can}} - 1] \) and \( f \in E(\psi, \rho) \).

Then \( \rho \) is said to witness \( I \) between \( \overleftarrow{T} \) and \( \overrightarrow{T} \), if \( G_{i,f} \) is a factor of \( \text{can}(\rho)[\overleftarrow{T}, \overrightarrow{T}] \).

- for \( I = 1_{n-1}a_{n,0} \), there exists a run \( \rho \in \mathcal{W} \) and \( \overleftarrow{T} = l_0 \leq l_1 \leq \cdots < l_n \leq \overrightarrow{T} \) such that \( \rho \) is an \( I \)-witness between \( l_{i-1} \) and \( l_i \), and \( \text{can}(\rho)[\overleftarrow{T}, \overrightarrow{T}] \in \Sigma_a \).

Then \( \rho \) is said to witness \( I \) between \( \overleftarrow{T} \) and \( \overrightarrow{T} \).

- for \( I = (l')^* \), there exist runs \( \rho, \psi \in \mathcal{W} \) such that \( \psi \preceq_f \rho \) and \( \rho \) is a witness for \( (l')^l \) between \( f(i) \) and \( f(i+1) \), and \( \rho[f(i), f(i+1)] \in \Sigma_a \), for some \( l > 0 \), \( i \in [0, |\psi|_{\text{can}} - 1] \) and \( f \in E(\psi, \rho) \).

Then \( \rho \) is said to witness \( I \) between \( \overleftarrow{T} \) and \( \overrightarrow{T} \), if \( G_{i,f} \) is a factor of \( \text{can}(\rho)[\overleftarrow{T}, \overrightarrow{T}] \).

For a word \( w \in \Sigma \cup \{\varepsilon\} \), \( w|_{\Sigma} \) denotes the restriction of \( w \) over \( \Sigma \). We call a word \( w' \) a factor of \( w = w_1 \cdots w_n \in \Sigma \cup \{\varepsilon\} \), if \( w|_{\Sigma} = \Sigma^* w' \Sigma^c \), i.e., \( w' \) is an infix of restriction of \( w \) over \( \Sigma \). 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 \( L_{|S} \), we have \( I \subseteq L_{|S} \) if and only if the system possesses an \( I \)-witness.

**Proof.** Let \( I \) be a pseudo-ideal over \( \Sigma = \Sigma_a \). Suppose \( I \subseteq L_{|S} \).

- If \( I = 0^\leq k \), then for \( u = 0^k \) there exists a run \( \rho \) in the system recognizing \( L \), such that \( u \preceq_S \text{yield}(\rho)|_{\Sigma} = u \). Then \( 0^k \) is a factor of \( \text{can}(\rho)[\overleftarrow{T}, \overrightarrow{T}] \) for some \( \overleftarrow{T} \) and \( \overrightarrow{T} \). Hence, \( \{\rho\} \) is an \( I \)-witness between \( 0 \) and \( |\rho|_{\text{can}} \).

- If \( I = 0^* \), then suppose there is no \( I \)-witness, i.e. for every pair of runs \( \psi \preceq_f \rho \), every gap word is either \( \varepsilon \) or it contains a letter \( p \) greater than \( 0 \). If every gap word is \( \varepsilon \), clearly \( I \not\subseteq L_{|S} \). Then suppose \( r \in \mathbb{N} (r > 0) \) be the maximum number such that \( \varnothing^r \) is a factor of a gap word. Then since \( I \subseteq L_{|S} \), \( 0^{3r+2} \) must be a factor of a run \( \rho \), then for any run that embeds in to \( \rho \), 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 \( 0^r \) for some \( l > 0 \), which is a contradiction to non-existence of an \( I \)-witness.

- If \( I = 1_{n-1}a_{n,0} \), 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 \( (R, \preceq) \) 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 \( \rho_1 \preceq \rho_2 \preceq \cdots \) from \( R'' \) such that \( \{\rho_1, \rho_2, \ldots\} \) covers \( I \): for \( \rho_1 \) 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 \( \rho_i' \) in \( R'' \) which witnesses \( I_i \). But every run in \( R'' \) embeds \( \rho_1 \), hence \( \rho_1 \) is a witness for \( I \).
Verifying Unboundedness via Amalgamation

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

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 \( R'_k \) be the set of runs that witness \((I'a)^k\) which is a pseudo-ideal of the type above.

Then again consider the sequence of runs \( \rho_1, \rho_1 \cdots \rho_i \) such that \( \rho_i \in R'_i \). Since the set of runs is a WQO over run embeddings, there is a subsequence \( \rho'_{i_1}, \rho'_{i_2} \cdots \) such that every run embeds into the next run. Now consider a run \( \rho'_i \) in this sequence which belongs to \( R'_i \), where \( t > |\rho'_i|_{\text{can}} \). Let \( \rho'_i \preceq \rho'_{i+1} \). Then there must exist \( i \in [1, |\rho'_i|_{\text{can}}] \), such that \( \rho'_i \) 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 \( \Sigma_a \). And hence, \( \rho'_i \) 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 \( I = 0^\omega \) 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 \( u \in I, u \preceq_S w \), implying that \( u \in L \downarrow_S \).

- If \( I = 0^\omega \), then the \( I \)-witness contains two runs \( \psi, \psi' \) such that \( \psi \preceq_f \rho \) and \( G_{i,f} = 0^l \) for some \( i \). Suppose \( u = 0^k \in I \), then can construct a sequence of runs \( \rho = \rho_1, \rho_2, \ldots, \rho_i \) obtained by amalgamating \( \rho_{i-1} \) with \( \psi \). Then the yield of \( \rho_i \) contains \( 0^l \) as a factor. Then there exists such that \( i \times l \geq k \), and then yield of \( \rho_i \) is simple block bigger than \( u \).

- If \( I = 1_i \alpha_1 a \cdots \alpha_n a \), then there is a run \( \rho \) that witnesses \( l_i \) between some \( l_i \) and \( l_{i+1} \). Let \( u = \alpha_1 w_2 a \cdots w_n a \in I \) such that \( w_1 \in l_i \). Then since \( \rho \) witnesses \( l_i \) between \( l_i \) and \( l_{i+1} \), we can obtain a run \( \rho_{i+1} \) such that \( \rho \preceq_f \rho_{i+1} \) that yields a simple block bigger word than \( w_1 \) between some \( f(i) \) and \( f(i+1) \), and witnesses \( \alpha_1 w_2 a \cdots \alpha_n a \) in an interval after \( f(i+1) \). Continuing the same way, we obtain \( \rho_{i+1} \cdots \rho_n \) such that \( \rho_i \) contains a simple block bigger word than \( w_1 w_2 a \cdots w_n a \) before position \( j \) and witnesses \( l_{i+1} \alpha_1 a \cdots l_{n-1} a \) after position \( j \). Then \( u \preceq_S \text{yield} \( (\rho_n) \), implying \( I \subseteq L \downarrow_S \).

- If \( I = (I'a)^l \), then there exist two runs \( \rho, \psi \) in the witness set such that \( \psi \preceq_f \rho \) and \( \rho \) is a witness for \((I'a)^l\) between \( f(i) \) and \( f(i+1) \), and \( \rho[f(i), f(i+1)] \in \Sigma_a^l \) for some \( l > 0 \), \( i \in [0, |\psi|_{\text{can}} - 1] \) and \( f \in E(\psi, \rho) \). Let \( u = \alpha_1 w_2 a \cdots w_n a \in I \). Then we can amalgamate \( \rho \) \( k \) many times with \( \psi \) to obtain a run that witnesses \( I'a \) in \( k \) disjoint intervals. Then with the arguments as above, we can get a run \( \rho' \) such that \( u \preceq_S \text{yield} \( (\rho') \).

\[ \square \]

C DETAILS ON COUNTER EXTENSIONS

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

**Proof.** By definition, \( L(S_{\eta,a}) = \bigcup_{(\rho, w) \in R_\eta} \alpha(\text{yield}(\rho)) \) and \( L_{\eta,a} = \alpha(\eta^{-1}(N_d) \cap L) \), thus it suffices to show that \( \bigcup_{(\rho, w) \in R_\eta} \text{yield}(\rho) = \eta^{-1}(N_d) \cap L \). Consider a word \( w = a_1 \cdots a_n \) from \( L = L(S) \). There is a run \( \rho \in R \) such that \( w = \text{yield}(\rho) \) is accepted by \( \rho \). 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 \( \rho \).

If \((\rho, u)\) is an accepting decorated run for some \( u = (u_1, v_1) \cdots (u_n, v_n) \), then \( q(u_i) \overset{\eta(a_i)}{\longrightarrow} q(v_i) \) for all \( i \in [1, n] \) in our VASS because \( v_i = u_i + \delta(\eta(a_i)) \) and \( \eta \) was assumed to be tame, \( q(v_i) = q(u_{i+1}) \) for all \( i \in [1, n - 1] \) because \( v_i = u_{i+1}, q(u_1) = q(0) \) because \( u_1 = 0 \), and \( q(v_n) = q(0) \) because \( v_n = 0 \). Thus there is a run of the VASS since

\[
q(0) = q(u_0) \overset{\eta(a_1)}{\longrightarrow} q(u_1) \overset{\eta(a_2)}{\longrightarrow} q(u_2) \cdots q(u_{n-1}) \overset{\eta(a_n)}{\longrightarrow} q(u_n) = q(0).
\]

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 \( \rho \) with the sequence of pairs \( u \equiv (u_0, u_1) (u_1, u_2) \cdots (u_{n-1}, u_n) \); then \( (\rho, u) \in R_\eta \) is an accepting decorated run.

\[ \square \]
D DETAILS ON ALGEBRAIC EXTENSIONS

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

D.1 Well-Quasi-Orderedness and Decorations

Let $t_1$ and $t_2$ be trees of $\mathcal{G}$. We recall the definition of the embedding $\preceq$ in $\mathcal{G}$, being $t_1 \preceq t_2$ if there exists a subtree $t_2/p$ such that

1. $t_1 = (A \to \rho)[t_1, \ldots, t_n]$,
2. $t_2/p = (A \to \sigma)[t_1', \ldots, t_n']$ and $\rho \preceq \sigma$, and
3. $t_i \preceq t'_i$ for all $i \in [1, n]$, where $g = \mu^{-1} \circ f \circ \mu$.

Lemma D.1. $(\mathcal{T}(\mathcal{G}), \preceq)$ is a well-quasi-order.

Proof. We rely on Nash-Williams’s minimal bad sequence argument [52]. Assume for the sake of contradiction that $(\mathcal{T}(\mathcal{G}), \preceq)$ is not a wqo. Then we can construct a minimal infinite bad sequence of trees $t_0, t_1, \ldots$, where minimality means that for all $i$, any sequence $t_0, t_1, \ldots, t_{i-1}, t_i, \ldots$ 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 $t_0, t_1, \ldots, t_i, t, \ldots$. At each step, the constructed sequence $t_0, t_1, \ldots, t_i$ is bad. The infinite sequence remains bad: for every $i < j$, $t_0, t_1, \ldots, t_j$ is a bad sequence, hence $t_i \not\preceq t_j$.

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

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

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

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

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

As there are only finitely many symbols in $N$, there is $A \in \mathbb{N}$ and an infinite bad subsequence $t_{u_1}, t_{u_2}, \ldots$ of $(t_{u_1})$, where all the $t_{u_i}$'s are $A$-rooted; let us write $\rho_j$ for the run of $M_A$ labelling the root of $t_{u_j} = (A \to \rho_j)[\ldots]$.

If any $\rho_j$ has $\{\rho_j\} \in \Sigma^*$, then $t_{u_j} = (A \to \rho_j)[\ldots]$ is a leaf. Because $M_A$ is an amalgamation system and therefore $\preceq$ is a wqo, there exists $\ell > j$ such that $\rho_j \preceq \rho_\ell$. Then $t_{u_j} \preceq t_{u_\ell}$, because condition (1) holds by assumption and condition (2) is vacuous in this case, a contradiction.

We therefore assume that each $\rho_j$ has $\{\rho_j\} = u_{j,0}B_{j,1}u_{j,1} \cdots B_{j,k_j}u_{j,k_j}$ for some $k_j > 0$ and non-terminals $B_{j,1}, \ldots, B_{j,k_j} \in N$; then $t_{u_j} = (A \to \rho_j)[t_{ij}/1, \ldots, t_{ij}/k_j]$. We decorate each $\rho_j$ with its sequence of children $w_j \equiv t_{ij}/1 \cdots t_{ij}/k_j$, which all belong to $S$. This gives rise to an infinite sequence $(\rho_j, w_j)$ of decorated runs in $\text{Deco}^\mathbb{N}(R)$. Because $(S, \preceq)$ is a wqo by Claim D.1 and $M_A$ supports wqo decorations by assumption, there is a pair $j < \ell$ with (1) $\rho_j \preceq \rho_\ell$, and (2) $t_{ij}/k \preceq t_{i\ell}/f(k)$ for all $k \in [1, k_j]$. This however implies that $t_{ij} \preceq t_{i\ell}$, 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 $\rho$ 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 $\preceq_X$ between decorated trees of $\mathcal{G}$ can be defined by replacing the order $\preceq$ on the inner runs with $\preceq_{X \cup \{1\}}$. Since $C$ supports wqo decorations, $\preceq_{X \cup \{1\}}$ is a well-quasi-ordering. The proof of Lemma 5.12 shows that $\preceq_X$ is a well-quasi-order as well. $\square$

### D.2 Admissible Embeddings

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

**Definition 5.11.** Assume $\tau = (A \rightarrow \rho)[t_1, \ldots, t_n]$ and $\text{can}(\rho) = u_0X_1u_1 \cdots X_nu_n$. We define $\text{can}(\tau) = \varepsilon \cdot u_0 \cdot \varepsilon \text{can}(t_1) \cdot u_1 \cdot \varepsilon \text{can}(t_2) \cdot \cdots \cdot \varepsilon \text{can}(t_n) \cdot \varepsilon \cdot u_n \cdot \varepsilon$.

Intuitively, we wrap the canonical decomposition of $\tau$ itself and of each child $t_i$ in $\varepsilon$ 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 $\rho$ in the image of $\tau$.

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 $\text{ix}_\rho(\tau)$ as the offset of the canonical decomposition of $\tau/p$ in the decomposition of $\tau$. That is, if the canonical decomposition of $\tau$ is $a_1 \cdots a_n$ and the canonical decomposition of $\tau/p$ is $b_1 \cdots b_m$, then $b_1$ corresponds to $a_{\text{ix}_\rho(\tau)+1}$.

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

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

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

- the length of the canonical decomposition of $\rho$ up to $X_i$,
- the combined length of the canonical decomposition of every $t_j$ with $j < i$,
- less the individual letters $X_i, \ldots, X_{i-1}$,
- two $\tau$ 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

$$\text{ix}_{i \cdot p'}(\tau) = \mu_p(i) + \sum_{j=1}^{i-1} |t_j|_{\text{can}} + i + \text{ix}_{p'}(t_i).$$

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

If $\tau/p \preceq \tau'/p'$ and $f \in E(\tau/p, \tau'/p')$, we write $f_{\tau/p}^{\tau'/p'}$ for the lifting of $f$ to $\tau$ and $\tau'$, a function from $[\text{ix}_p(\tau) + 1, \text{ix}_p(\tau) + |\tau/p|_{\text{can}}]$ to $[\text{ix}_{p'}(\tau') + 1, \text{ix}_{p'}(\tau') + |\tau'/p'|_{\text{can}}]$ given by

$$f_{\tau/p}^{\tau'/p'}(i + \text{ix}_p(\tau)) = f(i) + \text{ix}_{p'}(\tau').$$

If $\tau$ and $\tau'$ are trees from $\mathcal{G}$, then the set of admissible embeddings $E(\tau, \tau')$ is isomorphic to all the ways to embed $\tau$ into $\tau'$. Let $\tau = (A \rightarrow \rho)[t_1, \ldots, t_1]$. Each $p \in \mathbb{N}^+$ such that $\tau'/p = (A \rightarrow \rho')[t'_1, \ldots, t'_1]$, $\rho \preceq \rho'$, and $t_i \preceq t'_{\varphi(i)}$ corresponds to a set of admissible embeddings. Let $f_i \in E(t_i, t'_{\varphi(i)})$. We write $f_{\tau'/p}^{\tau}$ for $f_i f_{\tau'/p-\varphi(i)}$. Note that the domains and co-domains of each $f_i'$ are necessarily disjoint. Therefore we may take their union $g = \bigsqcup f_i$. 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 $\rho$, as well as the $\tau$-components, which we assign as follows:

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

We have

\[
1 \mapsto i \times_{p}(\tau') + 1
\]

\[
|\tau|_{\text{can}} \times_{p}(\tau') + |\tau'/p|_{\text{can}}
\]

\[
i_{\times}(\rho) \mapsto i \times_{p}(\tau') + i_{\times}(\tau'/p)
\]

\[
(1 \leq i \leq l)
\]

\[
i_{\times}(\rho) + |\tau|_{\text{can}} + 1 \mapsto i \times_{p}(\tau') + i_{\times}(\tau'/p) + |\tau'/\times_{p}(i)|_{\text{can} + 1}
\]

\[
(1 \leq i \leq l)
\]

\[
i_{\times}(\rho) + |\tau|_{\text{can}} + j \mapsto i \times_{p}(\tau') \cdot i_{\times}(\tau'/p) + (\varphi(d) - \mu_{\rho}(x)) + 1
\]

\[
(1 \leq i \leq l, 2 \leq j \leq (\mu_{\rho}(i + 1) - \mu_{\rho}(i))
\]

\[
(\text{where } d = \mu_{\rho}(i) + j - 1, x = \beta_{\rho}(\varphi(d))
\]

Intuitively, we may assume that the \( \varepsilon \)-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 \( \rho' \) by \( \varphi \).

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 \( \tau \) but the same path \( p \) for both, then either the underlying run embedding \( \varphi \) 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 \( \tau_{0} \leq_{f} \tau_{1} \) and \( \tau_{1} \leq_{g} \tau_{2} \), then \( \tau_{0} \leq_{g \circ f} \tau_{2} \).

Let \( \tau_{0} = (A \rightarrow \rho_{0})[\ldots] \), \( \tau_{1} \) and \( \tau_{2} \) be trees from \( \mathcal{G} \) and \( \tau_{0} \leq_{f} \tau_{1} \leq_{g} \tau_{2} \). The maps \( f \) and \( g \) correspond to specific embeddings between \( \tau \), \( \tau' \) and \( \tau'' \). In particular, let \( p \) be the path corresponding to the mapping of \( \tau_{0} \) into \( \tau_{1} \). We have \( \tau_{1}/p = (A \rightarrow \rho_{1})[\ldots] \) and \( \rho_{0} \leq_{g} \rho_{1} \). Let \( q \) be the path corresponding to the mapping of \( \tau_{1}/p \) into \( \tau_{2} \). We have \( \tau_{2}/q = (A \rightarrow \rho_{2})[\ldots] \) and \( \rho_{1} \leq_{\psi} \rho_{2} \). Due to the structure of \( \mathcal{M}_{A} \), we know that \( \varphi \) and \( \psi \) can be composed such that \( \rho_{0} \leq_{g \circ \varphi} \rho_{2} \). Then the path \( q \) and the embedding of the children of \( \tau \) along \( \psi \circ \varphi \) is also a valid embedding of \( \tau_{0} \) into \( \tau_{2} \).

If we expand the composition of \( f \) and \( g \), we get

\[
g(f(i)) = g(f'_{\tau_{1}/p}(i))
\]

\[
= g(f'_{i}) + i_{\times}(\tau_{1})
\]

\[
= g'_{\tau_{1}/p}(f'(i) + i_{\times}(\tau_{1}))
\]

\[
= g'_{f'(i)} + i_{\times}(\tau_{2})
\]

\[
= (g'_{f'})_{\tau_{2}/q}(i)
\]

which is what we would get from the direct mapping of \( \tau \) into \( \tau_{2}/q \).

An analogous line of reasoning holds for the case of \( \tau_{0} = (\varepsilon)[\ldots] \).

### D.4 Concatenative Amalgamation

**Lemma 5.15.** If \( \tau_{0}, \tau_{1}, \tau_{2} \) are all \( A \)-rooted trees such that \( \tau_{0} \leq_{f} \tau_{1} \) and \( \tau_{0} \leq_{g} \tau_{2} \), then for every choice of \( i \in \{0, |\tau_{0}|_{\text{can}}\} \) there exists an \( A \)-rooted tree \( \tau_{3} \) with \( \tau_{1} \leq_{f'} \tau_{3} \) and \( \tau_{2} \leq_{g'} \tau_{3} \) such that

1. \( \rho' \circ f = g' \circ g \) (we write \( h \) for this composition).
2. \( G_{j,h} \in \{G_{j,\rho}, G_{j,\rho} G_{j,\rho} G_{j,\rho} \} \) for every \( j \in \{0, |\tau_{0}|_{\text{can}}\} \), and in particular
3. \( G_{i,h} = G_{i,\rho} G_{i,\rho} \) for the chosen \( i \).
If $\tau_0 = (\epsilon)[\cdot]$ and therefore also $\tau_1$ and $\tau_2$, 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 [47, Sec. 3]. If $\tau = (A \to p)[\cdot \cdot \cdot]$, $p$ are trees and $\pi/p = (A \to \rho)[\cdot \cdot \cdot]$, we inductively define $\pi[p \mapsto \tau]$ as

$$
\pi[p \mapsto \tau] = \tau
$$

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

Note that this operation maintains all labels along the path $p$ and both $\tau$ and $\pi/p$ 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 $\pi/p \leq \tau$ and $\tau$ and $\pi/p$ are both $A$-rooted, then $\pi \leq \pi[p \mapsto \tau]$ (and $\pi$ and $\pi[p \mapsto \tau]$ are both $B$-rooted).

**Proof.** If $p = \epsilon$, 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 $t_i \leq t_i[p' \mapsto \tau]$. Then the definition of the tree embedding means we have $\pi \leq \pi[i \mapsto t_i[p' \mapsto \tau]] = \pi[p \mapsto \tau]$. $\square$

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

**Proof.** Let $\tau_1, \tau_2$ be trees such that $\tau_0 \not\leq_{f} \tau_1$ and $\tau_0 \not\leq_{g} \tau_2$. Recall that this means there are $p, p'$ such that $\tau_1/p = (A \to \rho_1)[t_1', \ldots, t_k']$, $p_0 \not\leq_{f} \rho_1$ and $t_i \leq t_i[\psi(i)]$; similarly $\tau_2/p' = (A \to \rho_2)[t_1'', \ldots, t'_k'']$, $p_0 \not\leq_{g} \rho_2$ and $t_i \leq t_i[\psi(i)]$.

As $C$ has concatenative amalgamation, we can construct a run $\rho_3$ in the system associated with $A$ such that $\rho_1 \not\leq_{\psi} \rho_3$ and $p_3 \not\leq_{\psi'} \rho_3$ such that $\psi' \circ \psi = \psi' \circ \psi$. We now construct a tree $\pi = (A \to \rho_3)[t_1''', t_2'''', \ldots]$. For each $j$ in $[1, k''']$, we can distinguish three cases:

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

Figure 6: Amalgamation of trees $\tau_1$ and $\tau_2$ (middle) over base tree $\tau_0$ (left).
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, $\Sigma$ 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 $(q, w, m), (q', w', m') \in \Delta$, 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

\[ L(\mathcal{A}) \overset{\triangleq}{=} \{ w \in \Sigma^* \mid \exists q \in F : (q_0, \epsilon, 1) \xrightarrow{\ast} (q, w, 1) \}, \]

where $\xrightarrow{\ast}$ 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 $E \subseteq \{ e \subseteq V \mid |e| \leq 2 \}$ is its set of edges. To each graph $\Gamma$, we associate a monoid as follows. Consider the alphabet $X_\Gamma \overset{\triangleq}{=} \{ a_v, \bar{a}_v \mid v \in V \}$, 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 $X_\Gamma^*$ of words, we define an equivalence relation. Consider the relation

\[ R_\Gamma \overset{\triangleq}{=} \{ (a_v\bar{a}_v, \epsilon) \mid v \in V \} \]

\[ \cup \{ (xy, yx) \mid x \in \{ a_v, \bar{a}_v \}, y \in \{ a_v, \bar{a}_v \}, \{ u, v \} \in E \}. \]

We now write $w \equiv w'$ 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
\( \Gamma \), 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 \( e \) has a self-loop in \( \Gamma \), then we may commute \( a_{e} \) and \( \bar{a}_{e} \), because \( (a_{e}, \bar{a}_{e}, \bar{a}_{e} a_{e}) \in R_{\Gamma} \). Finally, we define the monoid \( \mathcal{M}_\Gamma \) as the quotient \( X_{\Gamma}^* / \equiv_{\Gamma} \). Thus, \( \mathcal{M}_\Gamma \) is the set of equivalence classes of \( X_{\Gamma}^* \) modulo \( \equiv_{\Gamma} \) and multiplication is via \([x][y] \equiv [xy] \) (this is well-defined since \( \equiv_{\Gamma} \) is a congruence by definition).

For example, if \( \Gamma \) consists of a single vertex, then \( \mathcal{M}_\Gamma \) is called the \textit{bicyclic monoid} and is denoted by \( \mathcal{B} \). Since then \( X_{\Gamma} = \{ a_{e}, \bar{a}_{e} \} \) and the only pair in \( R_{\Gamma} = (a_{e}, \bar{a}_{e}, e) \), it is not difficult to see that for \( w \in X_{\Gamma}^* \), we have \( w \equiv_{\Gamma} e \) if and only if \( w \) is a well-bracketed word where \( a_{e} \) and \( \bar{a}_{e} \) are the opening and closing brackets. Hence, valence automata over \( \mathcal{B} \) 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 \( \Gamma \), a \textit{valence automaton} over \( \Gamma \) is a valence automaton over the monoid \( \mathcal{M}_\Gamma \). By \( \text{VA}(\Gamma) \), we denote the class of languages accepted by valence automata over \( \Gamma \).

#### 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 \( \text{VA}(\Gamma) \), then the language class \( \text{VA}(\Gamma) \) is a class of amalgamation systems. To this end, we show that \( \text{VA}(\Gamma) \) belongs to a language class obtained from the regular languages by repeatedly applying the operators \( \cdot + \mathbb{N} \) and \( \text{Alg}(\cdot) \). This will follow from results in [66].

For a graph \( \Gamma \), let \( \Gamma^- \) denote the graph where all self-loops are removed. In [66], it was shown that if \( \Gamma \) has one of the two graphs

\[
\begin{array}{c}
\text{\( \bullet \)} \\
\text{\( \bullet \)}
\end{array}
\]

(which are denoted \( P4 \) and \( C4 \), respectively) as an induced subgraph, then \( \text{VA}(\Gamma) \) is the class of all recursively enumerable languages, and in particular, the emptiness problem is undecidable for \( \text{VA}(\Gamma) \). Thus, for Main Theorem B, we only need to consider those graphs \( \Gamma \) for which \( \Gamma^- \) does not contain \( P4 \) and \( C4 \) as induced subgraphs. These graphs have been described in [66].

#### E.2.1 Graphs without \( P4 \) and \( C4 \)

Let \( \text{PD} \) be the smallest (isomorphism-closed) class of monoids such that

(i) the trivial monoid \( 1 \) belongs to \( \text{PD} \),
(ii) for every monoid \( M \) in \( \text{PD} \), we also have \( M \times \mathbb{B}, M \times \mathbb{Z} \) in \( \text{PD} \), and
(iii) for any monoids \( M \) and \( N \) in \( \text{PD} \), we also have \( M + N \) in \( \text{PD} \). Here, \( M + N \) denotes the \textit{free product} of monoids. The precise definition is not needed here—we will only need the following: in [61, Lem. 2], it is shown that for any monoids \( M, N \), we have \( \text{VA}(M + N) \subseteq \text{Alg}(\text{VA}(M) \cup \text{VA}(N)) \).

A consequence of Theorem 3.3, Proposition 3.5, and Proposition 3.6 in [66] is that, if \( \Gamma^- \) does not contain \( P4 \) or \( C4 \) as an induced subgraph, then \( \mathcal{M}_\Gamma \) belongs to \( \text{PD} \).

**Proof of Main Theorem B.** By the previous discussion, it remains to show that for every language in \( \text{VA}(M) \) with \( M \) in \( \text{PD} \), we can construct an amalgamation system. For (i) the trivial monoid \( 1 \), \( \text{VA}(1) \) 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 \( \text{VA}(M \times \mathbb{Z}) \subseteq \text{VA}(M \times \mathbb{B} \times \mathbb{B}) \). Indeed, a single \( \mathbb{Z} \)-counter can be simulated by two \( \mathbb{N} \)-counters.

Thus, it suffices to show that if \( \text{VA}(M) \) and \( \text{VA}(N) \) have concatenative amalgamation, then so do (ii) \( \text{VA}(M \times \mathbb{B}) \) and (iii) \( \text{VA}(M + N) \). In the latter case, we know that \( \text{VA}(M + N) \subseteq \text{Alg}(\text{VA}(M) \cup \text{VA}(N)) \) [61, Lem. 2] and thus we may apply Theorem 5.10. Finally, for \( M \times \mathbb{B} \), it is entirely straightforward to prove that \( \text{VA}(M \times \mathbb{B}) \) is included in the language class \( \text{VA}(M) + \mathbb{N} \), where \( \text{VA}(M) + \mathbb{N} \) is defined as in Section 5.2. Indeed, a valence automaton over \( M \times \mathbb{B} \) can be viewed as 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 $\mathcal{VA}(M) + \mathbb{N}$.

This shows that for every $M$ in $\mathcal{PD}$, all the languages in $\mathcal{VA}(M)$ have concatenative amalgamation systems. Finally, since each class $\mathcal{VA}(M)$ is a full trio [21, Thm. 4.1] this shows that Main Theorem B follows from Main Theorem A. □