


# Regular Separability of Well-Structured Transition Systems

**Wojciech Czerwiński**<sup>1</sup>

University of Warsaw, Poland


wczerwin@mimuw.edu.pl

 <https://orcid.org/0000-0002-6169-868X>

**Sławomir Lasota**<sup>2</sup>

University of Warsaw, Poland


sl@mimuw.edu.pl

 <https://orcid.org/0000-0001-8674-4470>

**Roland Meyer**

TU Braunschweig, Germany


roland.meyer@tu-bs.de

 <https://orcid.org/0000-0001-8495-671X>

**Sebastian Muskalla**

TU Braunschweig, Germany

s.muskalla@tu-bs.de

 <https://orcid.org/0000-0001-9195-7323>

**K. Narayan Kumar**<sup>3</sup>

Chennai Mathematical Institute and UMI RELAX, India

kumar@cmi.ac.in

**Prakash Saivasan**

TU Braunschweig, Germany

p.saivasan@tu-bs.de

---

## Abstract

We investigate the languages recognized by well-structured transition systems (WSTS) with upward and downward compatibility. Our first result shows that, under very mild assumptions, every two disjoint WSTS languages are regular separable: There is a regular language containing one of them and being disjoint from the other. As a consequence, if a language as well as its complement are both recognized by WSTS, then they are necessarily regular. In particular, no subclass of WSTS languages beyond the regular languages is closed under complement. Our second result shows that for Petri nets, the complexity of the backwards coverability algorithm yields a bound on the size of the regular separator. We complement it by a lower bound construction.

**2012 ACM Subject Classification** Theory of computation → Models of computation, Theory of computation → Formal languages and automata theory, Theory of computation → Regular languages, Theory of computation → Parallel computing models

**Keywords and phrases** regular separability, wsts, coverability languages, Petri nets

**Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2018.35

---

<sup>1</sup> Supported by the Polish National Science Centre under grant 2016/21/D/ST6/01376.

<sup>2</sup> Partially supported by the European Research Council (ERC) project Lipa under the EU Horizon 2020 research and innovation programme (grant agreement No. 683080).

<sup>3</sup> Partially supported by the Indo-French project AVeCSO, the Infos Foundation, and DST-VR Project P-02/2014.



© Wojciech Czerwiński, Sławomir Lasota, Roland Meyer, Sebastian Muskalla, K. Narayan Kumar, and Prakash Saivasan;

licensed under Creative Commons License CC-BY

29th International Conference on Concurrency Theory (CONCUR 2018).

Editors: Sven Schewe and Lijun Zhang; Article No. 35; pp. 35:1–35:18

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Related Version** The full version is available as technical report on arXiv [18], <https://arxiv.org/abs/1702.05334>.

**Acknowledgements** We thank an anonymous referee for pointing out Part (2) of Corollary 8. We thank Sylvain Schmitz for helpful discussions.

## 1 Introduction

We study the languages recognized by well-structured transition systems (WSTS) [24, 25, 5, 1, 28]. WSTS form a framework subsuming several widely-studied models, like Petri nets [23] and their extensions with transfer [22], data [54], and time [4], graph rewriting systems [36], depth-bounded systems [47, 57, 21], ad-hoc networks [3], process algebras [13], lossy channel systems (LCS) [5], and programs running under weak memory models [6, 7]. Besides their applicability, the importance of WSTS stems from numerous decidability results. Finkel showed the decidability of termination and boundedness [24, 25]. Abdulla came up with a backward algorithm for coverability [5], for which a matching forward procedure was found only much later [31]. Several simulation and equivalence problems are also decidable for WSTS [28]. The work on WSTS even influenced algorithms for regular languages [58] and recently led to the study of new complexity classes [55].

Technically, a WSTS is a transition system equipped with a quasi order on the configurations that satisfies two properties. It is a well quasi order and it is (upward or downward) compatible with the transition relation in the sense that it forms a simulation relation. For our language-theoretic study, we assume the transitions to be labeled and the WSTS to be equipped with sets of initial and final configurations. The set of final configurations is supposed to be upward or downward closed wrt. the quasi order of the WSTS. When specialized to VAS, this yields the so-called *covering languages*.

For WSTS languages, we study the problem of regular separability. Given two languages  $\mathcal{L}$  and  $\mathcal{K}$  over the same alphabet, a *separator* is a language  $\mathcal{R}$  that contains one of the languages and is disjoint from the other,  $\mathcal{L} \subseteq \mathcal{R}$  and  $\mathcal{R} \cap \mathcal{K} = \emptyset$ . The separator is regular if it is a regular language. Separability has recently attracted considerable attention. We discuss the related work in a moment.

Disjointness is clearly necessary for regular separability. We show that for most WSTS, disjointness is also sufficient. Our main result is the following:

Any two disjoint WSTS languages are regular separable.

The only assumption we need is that, in the case of upward-compatible WSTS resp. downward-compatible WSTS, one of the WSTS is finitely branching resp. deterministic.

The proof proceeds in two steps. In the first step, we link inductive invariants from verification [43] to separability in formal languages. More precisely, we show that any inductive invariant (of the product of the given systems) gives rise to a regular separator – provided it can be finitely represented. We do not even need WSTS here, but only upward compatibility. An inductive invariant is a set of configurations that contains the initial ones, is closed under the transition relation, and is disjoint from the final configurations.

In a second step, we show that finitely-represented invariants always exist. To this end, we use ideal completions from lattice theory [37, 9, 27]. The insight is that, in a WSTS, any inductive invariant can be finitely represented by its ideal decomposition. This ideal decomposition yields states in the ideal completion of the WSTS, and the first step applies.

The result has theoretical as well as practical applications. On the theoretical side, recall the following about Petri nets from [49, 48]: Every two Petri net covering languages that are

complements of each other are necessarily regular. The result not only follows from ours, but the same applies to other classes of WSTS, for instance to the languages of LCS, and actually to *all* WSTS languages fulfilling the above-mentioned assumptions. For instance, if the covering language of a Petri net is the complement of the language of an LCS, they are necessarily regular; and if the languages are just disjoint, they are regular separable.

The result is also important in verification. In 2016 and 2017, the Software Verification Competition was won by so-called language-theoretic algorithms [33]. These algorithms replace the classical state-space search by proofs of language disjointness (between a refinement of the control-flow language and the language of undesirable behavior). Regular separators are precisely what is needed to prove disjointness. In this setting, regular separators seem to play the role that inductive invariants play for safety verification [43]. Indeed, our results establishes a first link between the two.

We accompany our main result by two more findings. The first ones are determinization results that broaden the applicability of our results. For upward compatibility, we show that every finitely branching WSTS can be determinized. For downward compatibility, we show that every WSTS can be determinized if the quasi order is an  $\omega^2$ -wqo. In fact all examples from the literature are  $\omega^2$ -WSTS, hence they determinize, and in consequence satisfy the assumptions of our results.

Our second accompanying result is on the size of regular separators for Petri nets. We show how to construct a regular separator in the form of a non-deterministic automaton of size triply exponential in size of the given nets. With the main result at hand, the result amounts to giving a bound on the size of a finite representation of an inductive invariant. As inductive invariant, we use the complement of the configurations backward reachable from the final ones. The estimation starts from a result on the size of a basis for the backward reachable configurations [40] and reasons about the complementation. There is a matching lower bound for deterministic automata.

**Outline.** Section 2 recalls the basics on WSTS. The determinization results can be found in Section 3. They prepare the main result in Section 4. The state complexity of separators for Petri nets is in Section 5. Section 6 concludes the paper.

**Related Work.** Separability is a widely-studied problem in Theoretical Computer Science. A classical result says that every two co-recursively enumerable languages are recursively separable, i.e. separable by a recursive language [30]. In the area of formal languages, separability of regular languages by subclasses thereof was investigated most extensively as a decision problem: Given two regular languages, decide whether they are separable by a language from a fixed subclass. For the following subclasses, among others, the separability problem of regular languages is decidable: The piecewise-testable languages, shown independently in [19] and [51], the locally testable and locally threshold-testable languages [50], the languages definable in first-order logic [53], and the languages of certain higher levels of the first-order hierarchy [52].

Regular separability of classes larger than the regular languages attracted little attention until recently. As a remarkable example, already in the 70s, the undecidability of regular separability of context-free languages has been shown [56] (see also a later proof [34]); then the undecidability has been strengthened to visibly pushdown languages [38] and to languages of one-counter automata [17].

An intriguing problem, to the best of our knowledge still open, is the decidability of regular separability of Petri net languages, under the proviso that acceptance is by *reaching* a

distinguished final configuration. As for now, positive answers are known only for subclasses of VAS languages: PSPACE-completeness for one-counter nets (i.e. one-dimensional vector addition systems with states) [17], and elementary complexity for languages recognizable by Parikh automata (or, equivalently, by integer vector addition systems) [14]. Finally, regular separability of *commutative closures* of VAS languages has been shown to be decidable in [15]. As a consequence of this paper, regular separability of two VAS languages reduces to disjointness of the same two VAS languages (and is thus trivially decidable), given that acceptance is by *covering* a distinguished final configuration.

Languages of upward-compatible WSTS were investigated e.g. in [32], where interesting closure properties have been shown, including a natural pumping lemma. Various subclasses of languages of WSTS have been considered, e.g. in [20, 2, 45].

## 2 Well structured transition systems

**Well Quasi Orders.** A quasi order  $(X, \preceq)$ , i.e. a set  $X$  equipped with a reflexive and transitive binary relation  $\preceq$ , is called *well quasi order* (wqo) if for every infinite sequence  $x_1, x_2, \dots \in X$  there are indices  $i < j$  such that  $x_i \preceq x_j$ . It is folklore that  $(X, \preceq)$  is wqo iff it admits neither an infinite descending sequence (i.e. it is well-founded) nor an infinite antichain (i.e. it has the finite antichain property).

We will be working either with wqos, or with  $\omega^2$ -wqos, a strengthening of wqos. We prefer not to provide the technical definition of  $\omega^2$ -wqo (which can be found, e.g. in [44]), as it would not serve our aims. Instead, we take the characterization provided by Lemma 2 below as a working definition. The class of  $\omega^2$ -wqos provides a framework underlying the forward WSTS analysis developed in [26, 27, 31]. Both classes, namely wqos and  $\omega^2$ -wqos, are stable under various operations like taking the Cartesian product, the lifting to finite multisets (multiset embedding), and the lifting to finite sequences (Higman ordering).

A subset  $U \subseteq X$  is *upward closed* with respect to  $\preceq$  if  $u \in U$  and  $u' \succeq u$  implies  $u' \in U$ . Similarly, one defines *downward closed* sets. Clearly,  $U$  is upward closed iff  $X \setminus U$  is downward closed. The *upward* and *downward closure* of a set  $U \subseteq X$  are defined as:

$$\uparrow U = \{x \in X \mid \exists u \in U, x \succeq u\} \quad \text{and} \quad \downarrow U = \{x \in X \mid \exists u \in U, x \preceq u\} .$$

The family of all upward-closed resp. downward-closed subsets of  $X$  we denote by  $\mathcal{P}^\uparrow(X)$  resp.  $\mathcal{P}^\downarrow(X)$ . If  $(X, \preceq)$  is a wqo then every upward closed set is the upward closure of a finite set, namely of the set of its minimal elements. This is not the case for downward closed set; we thus distinguish a subfamily  $\mathcal{P}_{\text{fin}}^\downarrow(X) \subseteq \mathcal{P}^\downarrow(X)$  of *finitary* downward closed subsets of  $X$ , i.e. downward closures of finite sets. In general, these are not necessarily finite sets (e.g. consider the set  $\mathbb{N} \cup \{\omega\}$  with  $\omega$  bigger than all natural numbers, and the downward closure of  $\{\omega\}$ ). The set  $\mathcal{P}_{\text{fin}}^\downarrow(X)$ , ordered by inclusion, is a wqo whenever  $(X, \preceq)$  is:

► **Lemma 1.**  $(\mathcal{P}_{\text{fin}}^\downarrow(X), \subseteq)$  is a wqo iff  $(X, \preceq)$  is a wqo.

This property does not necessarily extend to the whole set  $\mathcal{P}^\downarrow(X)$  of all downward closed subsets of  $X$ . As shown in [35]:

► **Lemma 2.**  $(\mathcal{P}^\downarrow(X), \subseteq)$  is a wqo iff  $(X, \preceq)$  is an  $\omega^2$ -wqo.

As a matter of fact, [35] considers the reverse inclusion order on upward closed sets, which is clearly isomorphic to the inclusion order on downward closed sets.

**Labeled Transition Systems.** In the sequel we always fix a finite alphabet  $\Sigma$ . A labeled transition system (LTS)  $\mathcal{W} = (S, T, I, F)$  over  $\Sigma$  consists of a set of *configurations*  $S$ , a set of *transitions*  $T \subseteq S \times \Sigma \times S$ , and subsets  $I, F \subseteq S$  of *initial* and *final* configurations. We write  $s \xrightarrow{a} s'$  instead of  $(s, a, s') \in T$ . A path from configuration  $s$  to configuration  $s'$  over a word  $w = a_0 \cdots a_{k-1}$  is a sequence of configurations  $s = s_0, s_1, \dots, s_{k-1}, s_k = s'$  such that  $s_i \xrightarrow{a_i} s_{i+1}$  for all  $i \in \{0, \dots, k-1\}$ . We write  $s \xrightarrow{w} s'$ . For a subset  $X \subseteq S$  of configurations and a word  $w \in \Sigma^*$  we write

$$\begin{aligned} \text{REACH}_{\mathcal{W}}(X, w) &= \{s \in S \mid \exists x \in X : x \xrightarrow{w} s\}, \\ \text{REACH}_{\mathcal{W}}^{-1}(X, w) &= \{s \in S \mid \exists x \in X : s \xrightarrow{w} x\} \end{aligned}$$

for the set of all configurations reachable (resp. reversely reachable) from  $X$  along  $w$ . Note that we have  $\text{REACH}_{\mathcal{W}}(X, \varepsilon) = X = \text{REACH}_{\mathcal{W}}^{-1}(X, \varepsilon)$ . Important special cases will be the set of all  $a$ -successors (resp.  $a$ -predecessors) for  $a \in \Sigma$ , i.e. configurations reachable along a one-letter word  $a$ , and the configurations reachable from the initial configurations  $I$  (resp. reversely reachable from the final configurations  $F$ ):

$$\begin{aligned} \text{SUCC}_{\mathcal{W}}(X, a) &= \text{REACH}_{\mathcal{W}}(X, a) & \text{REACH}_{\mathcal{W}}(w) &= \text{REACH}_{\mathcal{W}}(I, w) \\ \text{PRED}_{\mathcal{W}}(X, a) &= \text{REACH}_{\mathcal{W}}^{-1}(X, a) & \text{REACH}_{\mathcal{W}}^{-1}(w) &= \text{REACH}_{\mathcal{W}}^{-1}(F, w) \end{aligned}$$

satisfying the following equalities for all  $w \in \Sigma^*$  and  $a \in \Sigma$ :

$$\text{REACH}_{\mathcal{W}}(w.a) = \text{SUCC}_{\mathcal{W}}(\text{REACH}_{\mathcal{W}}(w), a) \tag{1}$$

$$\text{REACH}_{\mathcal{W}}^{-1}(a.w) = \text{PRED}_{\mathcal{W}}(\text{REACH}_{\mathcal{W}}^{-1}(w), a). \tag{2}$$

We also establish the notation for the whole set of (reversely) reachable configurations:

$$\text{REACH}_{\mathcal{W}} = \bigcup_{w \in \Sigma^*} \text{REACH}_{\mathcal{W}}(w) \quad \text{REACH}_{\mathcal{W}}^{-1} = \bigcup_{w \in \Sigma^*} \text{REACH}_{\mathcal{W}}^{-1}(w).$$

An LTS  $\mathcal{W} = (S, T, I, F)$  is *finitely branching* if  $I$  is finite and for every configuration  $s \in S$  and each  $a \in \Sigma$  there are only finitely many configurations  $s' \in S$  such that  $s \xrightarrow{a} s'$ . Furthermore,  $\mathcal{W}$  is *deterministic* if it has exactly one initial configuration and for every  $s \in S$  and each  $a \in \Sigma$  there is exactly one  $s' \in S$  such that  $s \xrightarrow{a} s'$ . If  $\mathcal{W}$  is deterministic, we write  $s' = \text{SUCC}_{\mathcal{W}}(s, a)$  (resp.  $s' = \text{REACH}_{\mathcal{W}}(w)$ ) instead of  $\{s'\} = \text{SUCC}_{\mathcal{W}}(s, a)$  (resp.  $\{s'\} = \text{REACH}_{\mathcal{W}}(w)$ ).

The language recognized by  $\mathcal{W}$ , denoted  $\mathcal{L}(\mathcal{W})$ , is the set of words which occur on some path starting in an initial configuration and ending in a final one, i.e.

$$\mathcal{L}(\mathcal{W}) = \{w \in \Sigma^* \mid \exists i \in I, f \in F : i \xrightarrow{w} f\}.$$

We call two LTS  $\mathcal{W}, \mathcal{W}'$  *equivalent* if their languages are the same.

Note that we did not allow for  $\varepsilon$ -steps in transition systems. Even if  $\varepsilon$ -steps can be eliminated by pre-composing and post-composing every transition  $s \xrightarrow{a} s'$  with the reflexive-transitive closure of  $\xrightarrow{\varepsilon}$ , this transformation does not necessarily preserve finite branching.

**Synchronized Products.** Consider LTS  $\mathcal{W} = (S, T, I, F)$  and  $\mathcal{W}' = (S', T', I', F')$ . Their *synchronized product* is the LTS  $\mathcal{W} \times \mathcal{W}' = (S_{\times}, T_{\times}, I_{\times}, F_{\times})$  defined as follows: The configurations are tuples of configurations,  $S_{\times} = S \times S'$ , and the initial and final configurations are  $I_{\times} = I \times I'$  and  $F_{\times} = F \times F'$ , respectively. The transition relation is defined by

$$(s, s') \xrightarrow{a} (r, r') \text{ in } \mathcal{W} \times \mathcal{W}' \quad \text{if} \quad \begin{aligned} &s \xrightarrow{a} r \text{ in } \mathcal{W} \\ &\text{and } s' \xrightarrow{a} r' \text{ in } \mathcal{W}'. \end{aligned}$$

It is immediate from the definition that the language of the product is the intersection of the languages, i.e.  $\mathcal{L}(\mathcal{W} \times \mathcal{W}') = \mathcal{L}(\mathcal{W}) \cap \mathcal{L}(\mathcal{W}')$ . If  $\mathcal{W}$  and  $\mathcal{W}'$  both are finitely branching, then so is their product.

**Upward-Compatible Well-Structured Transition Systems.** Now we define a labeled version of *well-structured transition systems* as described in [28], here called upward-compatible well-structured transition system (UWSTS). We start by defining the more general notions of quasi ordered LTS and ULTS.

By a *quasi-ordered LTS*  $\mathcal{W} = (S, T, \preceq, I, F)$  we mean an LTS  $(S, T, I, F)$  extended with a quasi order  $\preceq$  on configurations.

An upward-compatible LTS (ULTS) is a quasi-ordered LTS such that the set  $F$  of final configurations  $F$  is upward closed<sup>4</sup> with respect to  $\preceq$ , and the following upward compatibility<sup>5</sup> is satisfied: whenever  $s \preceq s'$  and  $s \xrightarrow{a} r$ , then  $s' \xrightarrow{a} r'$  for some  $r' \in S$  such that  $r \preceq r'$ . In other words,  $\preceq$  is a simulation relation. Upward compatibility extends to words:

► **Lemma 3.** *For  $w \in \Sigma^*$ ,  $s \preceq s'$  with  $s \xrightarrow{w} r$ , we have  $s' \xrightarrow{w} r'$  for some  $r' \in S$  with  $r \preceq r'$ .*

If the order  $(S, \preceq)$  in a ULTS  $\mathcal{W} = (S, T, \preceq, I, F)$  is a wqo, we call  $\mathcal{W}$  a UWSTS.

As  $F$  is upward closed,  $\mathcal{W}$  is equivalent to its downward closure  $\downarrow\mathcal{W}$ , obtained from  $\mathcal{W}$  by replacing the set  $I$  by its (not necessarily finite) downward closure  $\downarrow I$  with respect to  $\preceq$ , and by extending the transition relation as follows:  $s \xrightarrow{a} r$  in  $\downarrow\mathcal{W}$  if  $s \xrightarrow{a} r'$  in  $\mathcal{W}$  for some  $r' \succeq r$ . Note that with respect to the extended transition relation,  $\text{Succ}_{\downarrow\mathcal{W}}(X, a)$  is downward closed for every  $X \subseteq S$ . One easily checks that  $\downarrow\mathcal{W}$  still satisfies upward compatibility, and every word accepted by  $\mathcal{W}$  is also accepted by  $\downarrow\mathcal{W}$ . The converse implication follows by the following simulation of  $\downarrow\mathcal{W}$  by  $\mathcal{W}$ :

► **Lemma 4.** *Let  $w \in \Sigma^*$ . Whenever  $s \preceq s'$  and  $s \xrightarrow{w} r$  in  $\downarrow\mathcal{W}$ , then  $s' \xrightarrow{w} r'$  in  $\mathcal{W}$  for some  $r' \in S$  such that  $r \preceq r'$ .*

The synchronized product of two ULTS  $(S, T, \preceq, I, F)$  and  $(S', T', \preceq', I', F')$  is still a ULTS with respect to the product order  $\preceq_{\times}$  defined by  $(x, x') \preceq_{\times} (y, y')$  iff  $x \preceq y$  and  $x' \preceq' y'$ . Indeed,  $F \times F'$  is upward closed wrt.  $\preceq_{\times}$  and the transition relation satisfies upward compatibility. Since the product order of two wqos is again a wqo, the synchronized product of two UWSTS is a UWSTS.

When  $\preceq$  is a  $\omega^2$ -wqo, the UWSTS  $\mathcal{W}$  is called  $\omega^2$ -UWSTS. When the LTS  $(S, T, I, F)$  is finitely branching (resp. deterministic), the UWSTS  $\mathcal{W}$  is called *finitely-branching UWSTS* (resp. *deterministic UWSTS*). In the sequel we speak shortly of UWSTS-languages (resp.  $\omega^2$ -UWSTS-languages, finitely-branching UWSTS-languages, etc.).

**Downward-Compatible Well-Structured Transition Systems.** A downward-compatible well-structured transition system (DWSTS) is defined like its upward-compatible counterpart, with two modifications. First, we assume the set of final configurations  $F$  to be downward closed, instead of being upward closed. Second, instead of upward compatibility, we require its symmetric variant, namely *downward compatibility*: Whenever  $s' \preceq s$  and  $s \xrightarrow{a} r$ , then  $s' \xrightarrow{a} r'$  for some  $r' \in S$  such that  $r' \preceq r$ . In other words, the inverse of  $\preceq$  is a simulation relation. Downward compatibility extends to words, which can be shown similar to Lemma 3. Symmetrically to the downward closure of a UWSTS, we may define the upward closure  $\uparrow\mathcal{W}$  of a DWSTS  $\mathcal{W}$  that recognizes the same language.

<sup>4</sup> Languages defined by upward-closed sets of final configurations are usually called *coverability languages*.

<sup>5</sup> In the terminology of [28], this is strong compatibility.

As above, we also speak of finitely-branching DWSTS, or  $\omega^2$ -DWSTS. We jointly call UWSTS and DWSTS just WSTS.

**Examples of WSTS.** Various well known and intensively investigated models of computation happen to be either an UWSTS or DWSTS. The list of natural classes of systems which are UWSTS contains, among the others: vector addition systems (VAS) resp. Petri nets and their extensions (e.g. with reset arcs or transfer arcs); lossy counter machines [10]; string rewriting systems based on context-free grammars; lossy communicating finite state machines (aka lossy channel systems, LCS) [12]; and many others. In the first two models listed above the configurations are ordered by the multiset embedding, while in the remaining two ones the configurations are ordered by Higman's subsequence ordering. The natural examples of UWSTS, including all models listed above, are  $\omega^2$ -UWSTS and, when considered without  $\varepsilon$ -transitions, finitely-branching.

DWSTS are less common. A natural source of examples is *gainy* models, like gainy counter system machines or gainy communicating finite state machines. For an overview, see e.g. page 31 of [28].

### 3 Expressibility

Our proof of regular separability assumes one of the WSTS to be deterministic. In this section, we show that this is no strong restriction. We compare the languages recognized by different classes of WSTS, in particular deterministic ones. The findings are summarized in Theorem 5, where we use  $\subseteq$  to say that every language of a WSTS from one class is also the language of a WSTS from another class; and we use  $\subseteq_{\text{rev}}$  to say that every language of a WSTS from one class is the reverse of the language of a WSTS from another class.

► **Theorem 5.** *The following relations hold between the WSTS language classes:*

$$\begin{aligned} \omega^2\text{-UWSTS} &\subseteq \text{deterministic UWSTS} = \text{finitely-branching UWSTS} \subseteq \text{all UWSTS} , \\ \omega^2\text{-DWSTS} &\subseteq \text{deterministic DWSTS} \subseteq \text{finitely-branching DWSTS} = \text{all DWSTS} , \\ \omega^2\text{-UWSTS} &\subseteq_{\text{rev}} \text{deterministic DWSTS} , \\ \omega^2\text{-DWSTS} &\subseteq_{\text{rev}} \text{deterministic UWSTS} . \end{aligned}$$

In short,  $\omega^2$ -UWSTS and  $\omega^2$ -DWSTS determinize and reverse-determinize; finitely-branching UWSTS determinize too; and (unrestricted) DWSTS are equivalent to finitely-branching DWSTS.

### 4 Regular Separability

We now show our first main results: Under mild assumptions, disjoint DWSTS resp. disjoint UWSTS are regular separable. Both theorems follow from a technical result that establishes a surprising link between verification and formal language theory: Every inductive invariant (of a suitable product WSTS) that has a finite representation can be turned into a regular separator. With this, the proofs of regular separability are invariant constructions.

**Main Results.** We say that two languages  $\mathcal{L}$  and  $\mathcal{K}$  over the same alphabet are *regular separable* if there is a regular language  $\mathcal{R}$  that satisfies  $\mathcal{L} \subseteq \mathcal{R}$  and  $\mathcal{R} \cap \mathcal{K} = \emptyset$ . For two WSTS  $\mathcal{W}$  and  $\mathcal{W}'$ , we say that they are regular separable if so are their languages. Disjointness is clearly necessary for regular separability. Our first main results show that for most WSTS disjointness is also sufficient:

► **Theorem 6.** *Every two disjoint DWSTS, one deterministic, are regular separable.*

► **Theorem 7.** *Every two disjoint UWSTS, one finitely branching, are regular separable.*

The results imply that the complement of a non-regular WSTS language cannot be a WSTS language. They also show that there is no subclass of WSTS languages beyond the regular languages that is closed under complement. More formally, for a class of languages  $\mathcal{C}$ , we call a language *doubly*  $\mathcal{C}$ , if the language as well as its complement are in  $\mathcal{C}$ . We obtain the following corollary, generalizing earlier results for Petri net coverability languages [49, 48].

► **Corollary 8.** *(1) Every doubly deterministic DWSTS language resp. every doubly finitely-branching UWSTS language is regular. (2) No subclass of finitely-branching UWSTS languages resp. deterministic DWSTS languages beyond REG is closed under complement.*

The rest of the section is devoted to the proofs. We will use that the product of two disjoint WSTS is again a WSTS with the empty language. Whenever the language of a WSTS is empty, we can find an inductive invariant, a downward-closed set of configurations separating the reachability set from the final configurations. Given a finite representation for such an invariant, we show how to turn it into a regular separator, provided one of the WSTS is deterministic. This is our key technical insight, formulated as Theorem 11 below.

The proof of Theorem 6 follows directly from this result. For Theorem 7, we consider the ideal completion of an UWSTS, an extended system in which every downward-closed set has a finite representation. This in particular applies to inductive invariants, as we show in the form of Proposition 21: Any inductive invariant in the original UWSTS induces an inductive invariant in the ideal completion that has a finite representation. Combining this result with Theorem 11 yields the desired proof.

**Turning Inductive Invariants into Regular Separators.** Inductive invariants are a standard tool in the safety verification of programs [43]. Technically, an inductive invariant (of a program for a safety property) is a set of program configurations that includes the initial ones, is closed under the transition relation, and is disjoint from the set of undesirable states. The following definition lifts the notion to WSTS (actually to the more general ULTS), where it is natural to require inductive invariants to be downward-closed.

► **Definition 9.** An *inductive invariant* for a ULTS  $\mathcal{W}$  with configurations  $S$  is a downward-closed set  $X \subseteq S$  with the following three properties:

$$I \subseteq X, \tag{3}$$

$$F \cap X = \emptyset, \tag{4}$$

$$\text{Succ}_{\mathcal{W}}(X, a) \subseteq X \text{ for all } a \in \Sigma. \tag{5}$$

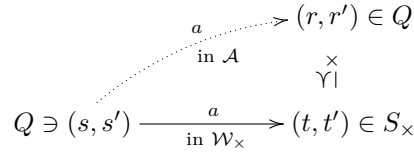
An inductive invariant  $X$  is *finitely-represented* if  $X = \downarrow Q$  for a finite set  $Q \subseteq S$ .

By (3) and (5), the invariant has to contain the whole reachability set. By (4) and (5), it has to be disjoint from the predecessors of the final configurations:

$$\text{REACH}_{\mathcal{W}} \subseteq X, \quad \text{REACH}_{\mathcal{W}}^{-1} \cap X = \emptyset.$$

This means every inductive invariant shows language emptiness. Even more, inductive invariants are complete for proving emptiness, like inductive invariants for programs are (relatively) complete for proving safety [16].





■ **Figure 1** The transition relation of  $\mathcal{A}$ .

► **Lemma 10.** *Consider ULTS  $\mathcal{W}$ . Then  $\mathcal{L}(\mathcal{W}) = \emptyset$  iff there is an inductive invariant for  $\mathcal{W}$ .*

For completeness, observe that  $X = \downarrow \text{REACH}_{\mathcal{W}}$  is an inductive invariant. It is the least one wrt. inclusion. There is also a greatest inductive invariant, namely the complement of  $\text{REACH}_{\mathcal{W}}^{-1}$ . Note that, due to upward compatibility,  $\text{REACH}_{\mathcal{W}}^{-1}$  is always upward-closed.

Other invariants may have the advantage of being easier to represent. We will be particularly interested in invariants that are finitely-represented in the sense that they form the downward closure of a finite set.

Here is the core result. Consider two disjoint ULTS. Any finitely-represented inductive invariant for the product can be turned into a regular separator. We will comment on the assumed determinism in a moment.

► **Theorem 11.** *Let  $\mathcal{W}$  and  $\mathcal{W}'$  be disjoint ULTS, one of them deterministic, such that  $\mathcal{W} \times \mathcal{W}'$  admits a finitely-represented inductive invariant  $\downarrow Q$ . Then  $\mathcal{W}$  and  $\mathcal{W}'$  are regular separable by the language of a finite automaton with states  $Q$ .*

For the definition of the separator, let  $\mathcal{W} = (S, T, \preceq, I, F)$  be an arbitrary ULTS and let  $\mathcal{W}' = (S', T', \preceq', I', F')$  be a deterministic one such that their languages are disjoint. Let

$$\mathcal{W}_\times = \mathcal{W} \times \mathcal{W}' = (S_\times, T_\times, \preceq_\times, I_\times, F_\times)$$

be their synchronized product. By the disjointness of  $\mathcal{W}$  and  $\mathcal{W}'$  we know that  $\mathcal{L}(\mathcal{W}_\times) = \emptyset$ . Let  $Q \subseteq S_\times$  be a finite set such that  $\downarrow Q$  is an inductive invariant.

We define a finite automaton  $\mathcal{A}$  with states  $Q$  whose language will contain  $L(\mathcal{W})$  while being disjoint from  $L(\mathcal{W}')$ . The idea is to over-approximate the configurations of  $\mathcal{W}_\times$  by the elements available in  $Q$ . The fact that  $\text{REACH}_{\mathcal{W}_\times} \subseteq \downarrow Q$  guarantees that every configuration  $(s, s') \in S_\times$  has such a representation. Since we seek to approximate the language of  $\mathcal{W}$ , the final states only refer to the  $\mathcal{W}$ -component. Transitions are approximated existentially.

► **Definition 12.** We define the *separating automaton induced by  $Q$*  to be  $\mathcal{A} = (Q, \rightarrow, Q_I, Q_F)$ . A state is initial if it dominates some initial configuration of  $\mathcal{W}_\times$ ,  $Q_I = \{(s, s') \in Q \mid (i, i') \preceq_\times (s, s') \text{ for some } (i, i') \in I_\times\}$ . As final states we take pairs whose  $\mathcal{W}$ -component is final,  $Q_F = \{(s, s') \in Q \mid s \in F\}$ . Finally, the transition relation in  $\mathcal{A}$  is an over-approximation of the transition relation in  $\mathcal{W}_\times$ :

$$(s, s') \xrightarrow{a} (r, r') \text{ in } \mathcal{A} \quad \text{if } (s, s') \xrightarrow{a} (t, t') \text{ in } \mathcal{W}_\times \text{ for some } (t, t') \preceq_\times (r, r').$$

Figure 1 illustrates the construction.

To show separation, we need to prove  $\mathcal{L}(\mathcal{W}) \subseteq \mathcal{L}(\mathcal{A})$  and  $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{W}') = \emptyset$ . We begin with the former. As  $\mathcal{W}'$  is deterministic,  $\mathcal{W}_\times$  contains all computations of  $\mathcal{W}$ . Due to upward compatibility,  $\mathcal{A}$  over-approximates the computations in  $\mathcal{W}_\times$ . Combining these two insights, which are summarized in the next lemma, yields the result.

► **Lemma 13.** (1) For every  $s \in \text{REACH}_{\mathcal{W}}(w)$  there is some  $(s, s') \in \text{REACH}_{\mathcal{W}_\times}(w)$ . (2) For every  $(s, s') \in \text{REACH}_{\mathcal{W}_\times}(w)$  there is some  $(r, r') \in \text{REACH}_{\mathcal{A}}(w)$  with  $(s, s') \preceq_\times (r, r')$ .

► **Proposition 14.**  $\mathcal{L}(\mathcal{W}) \subseteq \mathcal{L}(\mathcal{A})$ .

It remains to prove disjointness of  $\mathcal{L}(\mathcal{A})$  and  $\mathcal{L}(\mathcal{W}')$ . The key observation is that, due to determinism,  $\mathcal{W}'$  simulates the computations of  $\mathcal{A}$  – in the following sense: If upon reading a word  $\mathcal{A}$  reaches a state  $(s, s')$ , then the unique computation of  $\mathcal{W}'$  will reach a configuration dominated by  $s'$ .

► **Lemma 15.** For every  $w \in \Sigma^*$  and every  $(s, s') \in \text{REACH}_{\mathcal{A}}(w)$  we have  $\text{REACH}_{\mathcal{W}'}(w) \preceq' s'$ .

With this lemma we can show disjointness. Towards a contradiction, suppose some word  $w$  satisfies  $w \in \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{W}')$ . As  $w \in \mathcal{L}(\mathcal{A})$ , there is a configuration  $(s, s') \in \text{REACH}_{\mathcal{A}}(w)$  with  $s \in F$ . As  $w \in \mathcal{L}(\mathcal{W}')$ , the unique configuration  $\text{REACH}_{\mathcal{W}'}(w)$  belongs to  $F'$ . With the previous lemma and the fact that  $F'$  is upward-closed, we conclude  $s' \in F'$ . Together,  $(s, s') \in F_\times$ , which contradicts the fact that  $\downarrow Q$  is an inductive invariant, Property (4).

► **Proposition 16.**  $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{W}') = \emptyset$ .

Together, Proposition 14 and 16 show Theorem 11. With Theorem 11 at hand, the proof of regular separability for DWSTS follows easily.

**Proof of Theorem 6.** Consider an arbitrary DWSTS  $\mathcal{W} = (S, T, \preceq, I, F)$  and a deterministic one  $\mathcal{W}' = (S', T', \preceq', I', F')$ . We start with the observation that the inversed versions of  $\mathcal{W}$  and  $\mathcal{W}'$ , namely with the orders  $\preceq^{-1}$  and  $(\preceq')^{-1}$  and denoted by  $\mathcal{W}^{-1}$  and  $(\mathcal{W}')^{-1}$ , are ULTS. We claim that these ULTS satisfy the assumptions of Theorem 11. The language of  $\mathcal{W}_\times^{-1} = \mathcal{W}^{-1} \times (\mathcal{W}')^{-1}$  is empty since the language of  $\mathcal{W}_\times = \mathcal{W} \times \mathcal{W}'$  is empty and inversion does not change the language,  $\mathcal{L}(\mathcal{W}) = \mathcal{L}(\mathcal{W}^{-1})$  and similar for  $\mathcal{W}'$ . Inversion also does not influence determinism.

It remains to find an inductive invariant of  $\mathcal{W}_\times^{-1}$  that is finitely represented. We claim that  $X = \downarrow_{-1} \text{REACH}_{\mathcal{W}_\times^{-1}}$  is a suitable choice. The subscript indicates that the downward closure is computed relative to the quasi order of  $\mathcal{W}_\times^{-1}$ . As the language of  $\mathcal{W}_\times^{-1}$  is empty,  $X$  is an inductive invariant by Lemma 10. For the finite representation, note that inversion does not change the transition relation. Hence,  $\mathcal{W}_\times$  and  $\mathcal{W}_\times^{-1}$  reach the same configurations,  $\text{REACH}_{\mathcal{W}_\times^{-1}} = \text{REACH}_{\mathcal{W}_\times} = Z$ . With the definition of inversion,  $X = \downarrow_{-1} Z = \uparrow Z$  holds. Moreover,  $\uparrow Z = \uparrow \min(Z)$ , with minimum and upward closure computed relative to  $\mathcal{W}_\times$ . Since the configurations of  $\mathcal{W}_\times$  are well quasi ordered,  $\min(Z)$  is finite. Another application of inversion yields  $X = \uparrow \min(Z) = \downarrow_{-1} \min(Z)$ . Hence,  $X$  is a finitely-represented downward-closed subset of  $\mathcal{W}_\times^{-1}$ .

By Theorem 11, the languages of  $\mathcal{W}^{-1}$  and  $(\mathcal{W}')^{-1}$  are regular separable and so are the languages of  $\mathcal{W}$  and  $\mathcal{W}'$ . ◀

**Ideal Completions of UWSTS.** The proof of regular separability for UWSTS is more involved. Here, we need the notion of ideal completions [9, 27]. We show that any invariant for a WSTS yields a finitely-represented invariant for the corresponding ideal completion. Theorem 7 follows from this.

An *ideal* in a wqo  $(X, \preceq)$  is a non-empty downward-closed subset  $Z \subseteq X$  which is directed: For every  $z, z' \in Z$  there is a  $z'' \in Z$  with  $z \preceq z''$  and  $z' \preceq z''$ . Every downward-closed set decomposes into finitely many ideals. In fact, the finite antichain property is sufficient and necessary for this.

► **Lemma 17** ([37, 27, 41]). *In a wqo, every downward-closed set is a finite union of ideals.*

We use  $\text{ID-DEC}_X(Z)$  to denote the set of inclusion-maximal ideals in  $Z$ . By the above lemma,  $\text{ID-DEC}_X(Z)$  is always finite and

$$Z = \bigcup \text{ID-DEC}_X(Z). \quad (6)$$

We will also make use of the fact that ideals are irreducible in the following sense.

► **Lemma 18** ([37, 27, 41]). *Let  $(X, \preceq)$  be a wqo. If  $Z \subseteq X$  is downward-closed and  $I \subseteq Z$  is an ideal, then  $I \subseteq J$  for some  $J \in \text{ID-DEC}_X(Z)$ .*

The *ideal completion*  $(\overline{X}, \subseteq)$  of  $(X, \preceq)$  has as elements all ideals in  $X$ . The order is inclusion. The ideal completion  $\overline{X}$  can be seen as extension of  $X$ ; indeed, every element  $x \in X$  is represented by  $\downarrow\{x\} \in \overline{X}$ , and inclusion among such representations coincides with the original quasi order  $\preceq$ . Later, we will also need general ideals that may not be the downward closure of a single element.

In [27, 9], the notion has been lifted to WSTS  $\mathcal{W} = (S, T, \preceq, I, F)$ . The ideal completion of  $\mathcal{W}$  is the ULTS  $\overline{\mathcal{W}}$ , where the given wqo is replaced by its ideal completion. The initial configurations are the ideals in the decomposition of  $\downarrow I$ . The transition relation is defined similarly, by decomposing  $\downarrow \text{SUCC}_{\mathcal{W}}(X, a)$ , with  $X$  an ideal. The final configurations are the ideals that intersect  $F$ .

► **Definition 19** ([27, 9]). For an UWSTS  $\mathcal{W} = (S, T, \preceq, I, F)$ , we define its *ideal completion*  $\overline{\mathcal{W}} = (\overline{S}, \overline{T}, \subseteq, \overline{I}, \overline{F})$ , where  $(\overline{S}, \subseteq)$  is the ideal completion of  $(S, \preceq)$ , the transition relation is defined by  $\text{SUCC}_{\overline{\mathcal{W}}}(X, a) = \text{ID-DEC}_S(\downarrow \text{SUCC}_{\mathcal{W}}(X, a))$ ,  $\overline{I} = \text{ID-DEC}_S(\downarrow I)$ , and  $\overline{F} = \{X \in \overline{S} \mid X \cap F \neq \emptyset\}$ .

Using upward compatibility in  $\mathcal{W}$ , language equivalence holds and determinism is preserved.

► **Lemma 20.** *The ideal completion  $\overline{\mathcal{W}}$  of an UWSTS  $\mathcal{W}$  is a ULTS. We have  $\mathcal{L}(\overline{\mathcal{W}}) = \mathcal{L}(\mathcal{W})$ . If  $\mathcal{W}$  is deterministic, then so is  $\overline{\mathcal{W}}$ .*

As a matter of fact,  $\overline{\mathcal{W}}$  is even finitely branching, but we do not need this property.

The purpose of using ideal completions is to make it easier to find inductive invariants that are finitely represented. Assume the given UWSTS  $\mathcal{W}$  has an inductive invariant  $X$ , not necessarily finitely represented. By definition,  $X$  is downward-closed. Thus, by Lemma 17,  $X$  is a *finite* union of ideals. These ideals are configurations of the ideal completion  $\overline{\mathcal{W}}$ . To turn  $\text{ID-DEC}_S(X)$  into an inductive invariant of  $\overline{\mathcal{W}}$ , it remains to take the downward closure of the set. As the order among ideals is inclusion, this does not add configurations. In short, an inductive invariant for  $\mathcal{W}$  induces a finitely-represented inductive invariant for  $\overline{\mathcal{W}}$ .

► **Lemma 21.** *If  $X \subseteq S$  is an inductive invariant of  $\mathcal{W}$ ,  $\downarrow \text{ID-DEC}_S(X)$  is a finitely-represented inductive invariant of  $\overline{\mathcal{W}}$ .*

**Proof.** Define  $Q = \text{ID-DEC}_S(X)$ . Since  $Q$  contains all ideals  $Y \subseteq X$  that are maximal wrt. inclusion,  $\downarrow Q$  contains all ideals  $Y \subseteq X$ . We observe that  $X \stackrel{(6)}{=} \bigcup Q = \bigcup \downarrow Q$ . By Lemma 17,  $Q$  is finite and thus  $\downarrow Q$  is finitely-represented. It remains to check that  $\downarrow Q$  satisfies the Properties (3), (4), and (5).

We have  $I \subseteq X$  by Property (3), and since  $X$  is downward-closed, we obtain  $\downarrow I \subseteq X$ . Consequently, any ideal that is a subset of  $\downarrow I$  is also a subset of  $X$ , and  $\downarrow Q$  contains all such ideals. For Property (4), assume towards a contradiction that  $\downarrow Q$  contains an ideal  $Y$  that is final in  $\overline{\mathcal{W}}$ . This means  $Y$  contains a final configuration. Since  $Y \subseteq X$ , we obtain a

contradiction to  $X \cap F = \emptyset$ , Property (4). To check the inclusion  $\text{SUCC}_{\overline{\mathcal{W}}}(\downarrow Q, a) \subseteq \downarrow Q$ , we pick an ideal  $Y \in \downarrow Q$  and show  $\text{SUCC}_{\overline{\mathcal{W}}}(Y, a) \subseteq \downarrow Q$ . Recall the definition  $\text{SUCC}_{\overline{\mathcal{W}}}(Y, a) = \text{ID-DEC}_S(\downarrow \text{SUCC}_{\mathcal{W}}(Y, a))$ . Thus, any element of  $\text{SUCC}_{\overline{\mathcal{W}}}(Y, a)$  is an ideal that is a subset of  $\downarrow \text{SUCC}_{\mathcal{W}}(Y, a)$ . We have  $\text{SUCC}_{\mathcal{W}}(X, a) \subseteq X$  by Property (5). This implies  $\text{SUCC}_{\mathcal{W}}(Y, a) \subseteq X$  as  $Y \subseteq X$ , and even  $\downarrow \text{SUCC}_{\mathcal{W}}(Y, a) \subseteq X$  as  $X$  is downward-closed. Hence, any ideal that is a subset of  $\downarrow \text{SUCC}_{\mathcal{W}}(Y, a)$  is also subset of  $X$ , and thus an element of  $\downarrow Q$ . ◀

Theorem 11 expects invariants for UWSTS of a particular shape, namely products  $\mathcal{W} \times \mathcal{W}'$ . We now show that the operation of ideal completion commutes with taking products of UWSTS, a fact that will be key to the proof of Theorem 7. We start by recalling that the ideals in a product wqo  $X \times Y$  are precisely the products of the ideals in  $X$  and in  $Y$ .

► **Lemma 22** ([37, 27, 41]). *A set  $Z \subseteq X \times Y$  is an ideal iff  $Z = I \times J$ , where  $I \subseteq X$  and  $J \subseteq Y$  are ideals.*

Lemma 22 yields the mentioned commutativity.

► **Lemma 23.** *For two UWSTSes  $\mathcal{W}$  and  $\mathcal{W}'$ ,  $\overline{\mathcal{W}} \times \overline{\mathcal{W}'}$  and  $\overline{\mathcal{W} \times \mathcal{W}'}$  are isomorphic.*

We are now prepared to apply Theorem 11 once more to establish our second main result.

**Proof of Theorem 7.** Let  $\mathcal{W} = (S, T, \preceq, I, F)$  and  $\mathcal{W}' = (S', T', \preceq', I', F')$  be disjoint UWSTS and  $\mathcal{W}'$  finitely branching. By Theorem 5 we can assume  $\mathcal{W}'$  is deterministic.

We would like to construct a finitely-represented inductive invariant in the synchronized product of the ideal completions  $\overline{\mathcal{W}} \times \overline{\mathcal{W}'}$  and then apply Theorem 11. Indeed, by Lemma 20 we know that the ideal completions are disjoint ULTS, and that the latter one is still deterministic, so they satisfy the assumptions.

Relying on Lemma 23 we prefer to show the existence of a finitely-represented inductive invariant in  $\overline{\mathcal{W} \times \mathcal{W}'}$ . Using Proposition 21, it is sufficient to find any inductive invariant in  $\mathcal{W} \times \mathcal{W}'$ , it does not have to be finitely-represented. We know that such an inductive invariant exists by Lemma 10, since we assume  $\mathcal{L}(\mathcal{W} \times \mathcal{W}') = \mathcal{L}(\mathcal{W}) \cap \mathcal{L}(\mathcal{W}') = \emptyset$ . ◀

**Effective Representation.** The states of the separating automaton in the proof of Theorem 7 are ideals in the product systems. With Lemma 22, these are tuples of ideals in the original systems. For most types of UWSTS, it is known how ideals can be effectively represented, i.e. how to obtain finite representations on which the successors can be computed. We briefly mention such a construction for Petri nets in Lemma 28, see e.g. [9] for more examples. In general, one may exploit the fact that ideals are downward-closed sets, which in turn are complements of upward-closed sets that can be represented by finitely many minimal elements – an idea first proposed in [29]. Note that in the proof of Theorem 7, we invoke Theorem 5 to determinize the given finitely-branching UWSTS. The states of the resulting UWSTS are finitary downward-closed sets of states of the original one. For most types of UWSTS, this construction can be avoided. We demonstrate this for the case of Petri nets in the proof of Proposition 30.

## 5 Separator Size: The Case of Petri Nets

The UWSTS associated to Petri nets are finitely branching. Hence, Theorem 7 applies: Whenever the coverability languages of two Petri nets are disjoint, they are regular separable. We now show how to construct a triply-exponential non-deterministic finite automaton (NFA) separating two such languages, provided they are disjoint. Moreover, for deterministic finite automata (DFA), we show that this size cannot be avoided.

► **Theorem 24.** *Let  $\mathcal{L}(N_1)$ ,  $\mathcal{L}(N_2)$  be disjoint Petri net coverability languages. There is an NFA  $\mathcal{A}$  of size triply exponential in  $|N_1| + |N_2|$  such that  $\mathcal{L}(\mathcal{A})$  separates  $\mathcal{L}(N_1)$  and  $\mathcal{L}(N_2)$ .*

► **Theorem 25.** *In general, Petri net coverability languages cannot be separated by DFA of less than triply-exponential size.*

Instead of invoking Theorem 7, which uses Theorem 5 to determinize, we directly show how to construct an equivalent instance of the separability problem in which one of the nets is deterministic. In this setting, we prove an upper bound that combines Theorem 11 with a size estimation for an ideal decomposition. We then show how to handle non-determinism. The lower bound combines a classical result from automata theory, showing that minimal DFA may have exponentially many states [39], with a Petri net construction due to Lipton [42].

**Petri Nets.** A Petri net over the alphabet  $\Sigma$  is a tuple  $N = (P, T, F, \lambda, M_0, M_f)$  where  $P$  is a finite set of places,  $T$  is a finite set of transitions with  $P \cap T = \emptyset$ ,  $F: (P \cup T) \times (P \cup T) \rightarrow \mathbb{N}$  is a flow function, and  $\lambda: T \rightarrow \Sigma$  is a labeling of the transitions. The runtime behavior of Petri nets is defined in terms of so-called *markings* from  $M \in \mathbb{N}^d$  with  $d = |P|$ . If  $M(p) = k > 0$ , we say place  $p$  carries  $k$  tokens. We assume to be given an initial and a final marking,  $M_0, M_f \in \mathbb{N}^d$ . Markings are changed by firing transitions: A transition  $t \in T$  is *enabled* in marking  $M \in \mathbb{N}^d$ , if  $M(p) \geq F(p, t)$  for all places  $p$ . An enabled transition can be *fired* leading to the marking  $M'$  with  $M'(p) = M(p) - F(p, t) + F(t, p)$ , denoted  $M[t]M'$ . Note that enabledness and firing are upward compatible with the componentwise ordering  $\leq$  on markings, in the following sense. If  $M_1 \leq M_2$  and  $M_1[t]M'_1$ , then  $M_2[t]M'_2$  with  $M'_1 \leq M'_2$ .

Relying on this compatibility, we can define the *UWSTS induced by  $N$*  to be  $\mathcal{W}_N = (\mathbb{N}^P, T', \leq, \{M_0\}, \uparrow M_f)$ . The transition relation is defined by  $(M, a, M') \in T'$  if there is a transition  $t \in T$  such that  $M[t]M'$  and  $\lambda(t) = a$ . The language of  $\mathcal{W}_N$  is also called the (*coverability*) *language of  $N$* <sup>6</sup>, and denoted by  $\mathcal{L}(N)$ . We call  $N$  *deterministic* if  $\mathcal{W}_N$  is.

We use a *product operation* on Petri nets  $N_i = (P_i, T_i, F_i, \lambda_i, M_{0,i}, M_{f,i})$ ,  $i = 1, 2$ . The product Petri net is obtained by putting the places of  $N_1$  and  $N_2$  side by side and creating a new transition for all pairs of transitions in  $T_1 \times T_2$  that carry the same label. Formally,  $N_1 \times N_2 = (P, T, F, \lambda, M_0, M_f)$  with  $P = P_1 \cup P_2$ ,  $T = \{(t_1, t_2) \in T_1 \times T_2 \mid \lambda(t_1) = \lambda(t_2)\}$ . We have  $\lambda(t_1, t_2) = \lambda(t_1) = \lambda(t_2)$ . The flow function is defined by the flow functions of the component Petri nets,  $F(p, (t_1, t_2)) = F_x(p, t_x)$  and  $F((t_1, t_2), p) = F_x(t_x, p)$ , where  $x = i$  if  $p \in P_i$ . We have  $M_0(p) = M_{0,i}(p)$  for  $p \in P_i$ , and similar for  $M_f$ . The product operation on Petri nets coincides with the product on UWSTS.

► **Lemma 26.**  $\mathcal{W}_{N_1 \times N_2}$  is isomorphic to  $\mathcal{W}_{N_1} \times \mathcal{W}_{N_2}$ .

We will need the size of a Petri net. It is defined using a binary encoding of the values in the range of the flow function and in the markings. Define the *infinity norm* of a vector  $M \in \mathbb{N}^d$  to be  $\|M\|_\infty = \max_{p \in P} M(p)$ . We extend this notion to matrices, sets of vectors, and functions by taking the maximum over all entries, elements, and elements in the range, respectively. The *size* of the Petri net  $N$  is now  $|N| = |P| |T| (1 + \lceil \log_2(1 + \|F\|_\infty) \rceil) + |M_0| + |M_f|$ . The size of a marking  $M$  is  $|M| = |P|(1 + \lceil \log_2(1 + \|M\|_\infty) \rceil)$ .

**An Upper Bound Assuming Determinism.** Theorem 11 assumes that one of the UWSTS is deterministic. We now show that for Petri nets, in this case, the regular separator is (an NFA of size) at most doubly exponential in the size of the input Petri nets.

<sup>6</sup> We consider covering the final marking as acceptance condition, i.e. a sequence of transitions is accepting if it reaches some marking  $M'$  with  $M'(p) \geq M_f(p)$  for all  $p \in P$ .

To prove the result, we show how a size estimation for the basis of  $\text{REACH}_{\mathcal{W}}^{-1}$  with  $\mathcal{W} = \mathcal{W}_{N_1 \times N_2}$  can be turned into a size estimation for the ideal decomposition of the complement. The size estimation of the basis is the following result. It is obtained by inspecting Abdulla's backward search [1].

► **Theorem 27** (Bozzelli & Ganty [11]). *Consider a Petri net  $N$  with final marking  $M_f$ . Then  $\text{REACH}_{\mathcal{W}_N}^{-1} = \uparrow\{v_1, \dots, v_k\}$ , where  $k$  as well as  $\|\{v_1, \dots, v_k\}\|_\infty$  are bounded from above by*

$$g = (|T| \cdot (\|F\|_\infty + \|M_0\|_\infty + \|M_f\|_\infty + 2))^{2^{\mathcal{O}(|P| \cdot \log |P|)}}.$$

By Lemma 10,  $\mathbb{N}^d \setminus \text{REACH}_{\mathcal{W}}^{-1}$  is an inductive invariant of  $\mathcal{W}$  (provided the language is empty). We can now apply Lemma 17 to finitely represent this set by its ideal decomposition. To represent this ideal decomposition in turn, we have to explicitly represent ideals in  $\mathbb{N}^d$ . The following lemma gives such a representation.

Let  $\mathbb{N}_\omega$  denote  $\mathbb{N}$  extended by a new top element  $\omega$ . Every ideal in  $\mathbb{N}^d$  is the downward closure  $\downarrow u$  of a single vector  $u \in \mathbb{N}_\omega^d$ . The lemma moreover shows how to compute the intersection of two ideals and how to obtain the ideal decomposition of the complement  $\mathbb{N}^d \setminus \uparrow v$  of the upward closure of a vector  $v \in \mathbb{N}^d$ .

► **Lemma 28** (see e.g. [40]). *(1) The ideals in  $\mathbb{N}^d$  have the shape  $\downarrow u$  for  $u \in \mathbb{N}_\omega^d$ . (2) For two ideals  $\downarrow u_1, \downarrow u_2$  of  $\mathbb{N}^d$ , the intersection is  $\downarrow u_1 \cap \downarrow u_2 = \downarrow u$  with  $u(i) = \min\{u_1(i), u_2(i)\}$ . (3) For  $v \in \mathbb{N}^d$ , we have  $\text{ID-DEC}(\mathbb{N}^d \setminus \uparrow v) = \{\downarrow u_{<v(j)} \mid j \in [1..d]\}$ , where  $u_{<v(j)}(j) = v(j) - 1$  and  $u_{<v(j)}(i) = \omega$  for  $i \neq j$ .*

We can now combine Theorem 27 and Lemma 28 to obtain our upper bound.

► **Proposition 29.** *Let  $N_1$  be an arbitrary Petri net and let  $N_2$  be deterministic. If  $N_1$  and  $N_2$  are disjoint, they can be separated by an NFA of size doubly exponential in  $|N_1| + |N_2|$ .*

**A General Upper Bound.** The previous result yields a doubly-exponential separator in the case where  $N_2$  is deterministic. We now show how to get rid of this assumption and construct a separator in the general case.

► **Proposition 30.** *Let  $N_1$  and  $N_2$  be disjoint Petri nets. Then they are separable by an NFA of size triply exponential in  $|N_1| + |N_2|$ .*

The proof transforms  $N_1$  and  $N_2$  into  $N_{-\lambda}$  and  $N_{det}$  so that  $N_{det}$  is deterministic, invokes Proposition 29, and then turns the resulting separator for  $N_{-\lambda}$  and  $N_{det}$  into a separator for  $N_1$  and  $N_2$ . The approach is inspired by [14].

Let  $N_2$  be non-deterministic with labeling function  $\lambda: T_2 \rightarrow \Sigma$ . We define  $N_{det}$  to be a variant of  $N_2$  that is labeled by the identity function, i.e.  $N_{det}$  is a Petri net over the alphabet  $T_2$ . We have  $\mathcal{L}(N_2) = \lambda(\mathcal{L}(N_{det}))$ , where we see  $\lambda$  as a homomorphism on words. We furthermore define  $N_{-\lambda}$  to be the  $T_2$ -labeled Petri net obtained from  $N_1$  as follows. For each  $a$ -labeled transition  $t_1$  of  $N_1$  and each  $a$ -labeled transition  $t$  of  $N_2$ ,  $N_{-\lambda}$  contains a  $t$ -labeled copy  $t_1^t$  of  $t_1$  with the same input-output behavior. Transition  $t_1$  itself is removed.

► **Lemma 31.**  $\mathcal{L}(N_1 \times N_2) = \lambda(\mathcal{L}(N_{-\lambda} \times N_{det}))$ .

With this lemma, and since  $N_1$  and  $N_2$  are disjoint,  $N_{-\lambda}$  and  $N_{det}$  have to be disjoint. As  $N_{det}$  is deterministic, we can apply Proposition 29 and obtain a separator for  $N_{-\lambda}$  and  $N_{det}$ . Let  $\mathcal{A}$  be the doubly-exponential NFA over the alphabet  $T_2$  with  $\mathcal{L}(N_{-\lambda}) \subseteq \mathcal{L}(\mathcal{A})$  and  $\mathcal{L}(N_{det}) \cap \mathcal{L}(\mathcal{A}) = \emptyset$ . We show how to turn  $\mathcal{A}$  into a separator for  $N_1$  and  $N_2$ . The first

step is to determine the complement automaton  $\mathcal{A}^c$ , which satisfies  $\mathcal{L}(N_{det}) \subseteq \mathcal{L}(\mathcal{A}^c)$  and  $\mathcal{L}(N_{-\lambda}) \cap \mathcal{L}(\mathcal{A}^c) = \emptyset$ . The second step is to apply  $\lambda$  to  $\mathcal{A}^c$ . Let  $\mathcal{B} = \lambda(\mathcal{A}^c)$  be the automaton obtained from  $\mathcal{A}^c$  by relabeling each  $t$ -labeled transition to  $\lambda(t)$ . The following lemma shows that  $\mathcal{B}$  is a separator for the original nets. The observation that the size of  $\mathcal{A}^c$  and hence the size of  $\mathcal{B}$  is at most exponential in the size of  $\mathcal{A}$  concludes the proof of Proposition 30

► **Lemma 32.**  $\mathcal{L}(N_2) \subseteq \mathcal{L}(\mathcal{B})$  and  $\mathcal{L}(N_1) \cap \mathcal{L}(\mathcal{B}) = \emptyset$ .

Note that  $\lambda(\mathcal{A})$  is not necessarily a separator: There might be  $u \in \mathcal{L}(\mathcal{A})$ ,  $u \notin \mathcal{L}(N_{det})$  such that there is  $u' \in \mathcal{L}(N_{det})$  with  $\lambda(u) = \lambda(u')$ . Thus,  $\lambda(u) \in \lambda(\mathcal{L}(\mathcal{A})) \cap \mathcal{L}(N_2)$ .

**A Lower Bound.** We now consider separation by *deterministic* finite automata (DFA). In this case, we can show a triply-exponential lower bound on the size of the separator.

► **Proposition 33.** *For all  $n \in \mathbb{N}$ , there are disjoint Petri nets  $N_0(n)$  and  $N_1(n)$  of size polynomial in  $n$  such that any separating DFA has size at least triply exponential in  $n$ .*

Our proof relies on the classical result that for each  $x \in \{0, 1\}$  and each  $k \in \mathbb{N}$ , the minimal DFA for the language  $\mathcal{L}_{x@k} = \{w \in \{0, 1\}^{\geq k} \mid \text{the } k\text{-last letter in } w \text{ is } x\}$  needs at least  $2^k$  states [39]. To obtain the desired lower bound, we will show how to generate  $\mathcal{L}_{x@k}$  for a doubly-exponential number  $k$  by a polynomially-sized Petri net. To this end, we make use of Lipton’s proof of EXPSPACE-hardness for coverability [42].

## 6 Conclusion

We have shown that, under mild assumptions, disjointness of WSTS languages implies their regular separability. In particular, we have shown that if one of two disjoint upward-compatible WSTS is finitely branching, they are regular separable. Using our expressibility results, it is also sufficient if the underlying order for one of the two is an  $\omega^2$ -wqo. A similar result holds for downward-compatible WSTS assuming that one of them is deterministic or the underlying order is an  $\omega^2$ -wqo. As WSTS are typically  $\omega^2$ -WSTS, our result already implies the decidability of regular separability for almost all WSTS of practical relevance.

Our work brings together research on inductive invariants and regular separability. We show that a finite representation of an inductive invariant for the product system can be transformed into a regular separator. For Petri nets, one may use any representation of the coverability set. As we show, it is beneficial in terms of the worst-case size, to use an inductive invariant obtained from the backward coverability algorithm [1]. For lossy channel systems, the coverability set is not computable [46], but one can obtain a finitely-represented inductive invariant e.g. from the EEC-algorithm [31].

We leave some questions without answer. It is not clear whether the assumptions of Theorems 7 and 6 are necessary; we were neither able to drop the assumptions, nor to provide a counterexample. Similarly, we do not know whether the inclusions in Theorem 5 are strict. Finally, in the case of Petri nets, closing the gap between the triply-exponential size of the NFA separator and the triply-exponential lower bound for DFA remains an open problem.

As future work, one could consider the *well-behaved transition systems (WBTS)* of [8], a generalization of WSTS where only the finite-antichain property is required.

## References

- 1 P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *LICS*, pages 313–321, 1996.
- 2 P. A. Abdulla, G. Delzanno, and L. Van Begin. Comparing the expressive power of well-structured transition systems. In *CSL*, pages 99–114, 2007.
- 3 P. A. Abdulla, G. Delzanno, O. Rezine, A. Sangnier, and R. Traverso. On the verification of timed ad hoc networks. In *FORMATS*, volume 6919 of *LNCS*, pages 256–270. Springer, 2011.
- 4 P. A. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *LICS*, pages 345–354. IEEE, 2004.
- 5 P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *LICS*, pages 160–170. IEEE, 1993.
- 6 M. F. Atig, A. Bouajjani, S. Burckhardt, and M. Musuvathi. On the verification problem for weak memory models. In *POPL*, pages 7–18. ACM, 2010.
- 7 M. F. Atig, A. Bouajjani, S. Burckhardt, and M. Musuvathi. What’s decidable about weak memory models? In *ESOP*, volume 7211 of *LNCS*, pages 26–46. Springer, 2012.
- 8 M. Blondin, A. Finkel, and P. McKenzie. Well behaved transition systems. *Logical Methods in Computer Science*, 13(3), 2017.
- 9 M. Blondin, A. Finkel, and P. McKenzie. Handling infinitely branching well-structured transition systems. *Inf. Comput.*, 258:28–49, 2018.
- 10 A. Bouajjani and R. Mayr. Model checking lossy vector addition systems. In *STACS*, pages 323–333, 1999.
- 11 L. Bozzelli and P. Ganty. Complexity analysis of the backward coverability algorithm for VASS. In *RP*, pages 96–109, 2011.
- 12 D. Brand and P. Zafropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
- 13 N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication, and iteration in process calculi. In *ICALP*, volume 3142 of *LNCS*, pages 307–319. Springer, 2004.
- 14 L. Clemente, W. Czerwiński, S. Lasota, and C. Paperman. Regular separability of Parikh automata. In *ICALP*, pages 117:1–117:13, 2017.
- 15 L. Clemente, W. Czerwiński, S. Lasota, and C. Paperman. Separability of reachability sets of vector addition systems. In *STACS 2017*, pages 24:1–24:14, 2017.
- 16 S. A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7(1):70–90, 1978.
- 17 W. Czerwiński and S. Lasota. Regular separability of one counter automata. In *LICS*, pages 1–12, 2017.
- 18 W. Czerwiński, S. Lasota, R. Meyer, S. Muskalla, K. Narayan Kumar, and P. Saivasan. Regular separability of well structured transition systems. *CoRR*, abs/1702.05334, 2018. [arXiv:1702.05334](https://arxiv.org/abs/1702.05334).
- 19 W. Czerwiński, W. Martens, and T. Masopust. Efficient separability of regular languages by subsequences and suffixes. In *ICALP*, pages 150–161, 2013.
- 20 G. Delzanno and F. Rosa-Velardo. On the coverability and reachability languages of monotonic extensions of Petri nets. *Theoretical Computer Science*, 467:12–29, 2013.
- 21 E. D’Osualdo. *Verification of Message Passing Concurrent Systems*. PhD thesis, University of Oxford, 2015.
- 22 C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *ICALP*, volume 1443 of *LNCS*, pages 103–115. Springer, 1998.
- 23 J. Esparza. Decidability and complexity of Petri net problems - an introduction. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, pages 374–428. Springer, 1998.



- 24 A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. In *ICALP*, pages 499–508, 1987.
- 25 A. Finkel. Reduction and covering of infinite reachability trees. *Inf. Comput.*, 89(2):144–179, 1990.
- 26 A. Finkel and J. Goubault-Larrecq. Forward analysis for WSTS, part I: completions. In *STACS*, pages 433–444, 2009.
- 27 A. Finkel and J. Goubault-Larrecq. Forward analysis for WSTS, part II: complete WSTS. *Logical Methods in Computer Science*, 8(3), 2012.
- 28 A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
- 29 P. Ganty, J.-F. Raskin, and L. Van Begin. A complete abstract interpretation framework for coverability properties of WSTS. In *VMCAI*, pages 49–64, 2006.
- 30 W. Gasarch. A survey of recursive combinatorics. In *Handbook of recursive mathematics, Vol. 2*, volume 139 of *Stud. Logic Found. Math.*, page 1041–1176. Amsterdam: North-Holland, 1998.
- 31 G. Geeraerts, J.-F. Raskin, and L. Van Begin. Expand, enlarge and check: New algorithms for the coverability problem of WSTS. *Journal of Computer and System Sciences*, 72(1):180–203, 2006.
- 32 G. Geeraerts, J.-F. Raskin, and L. Van Begin. Well-Structured Languages. *Acta Informatica*, 44(3-4):249–288, 2007.
- 33 M. Heizmann, J. Hoenicke, and A. Podelski. Nested interpolants. In *POPL*, pages 471–482. ACM, 2010.
- 34 H. B. Hunt III. On the decidability of grammar problems. *Journal of the ACM*, 29(2):429–447, 1982.
- 35 P. Jancar. A note on well quasi-orderings for powersets. *Inf. Process. Lett.*, 72(5-6):155–160, 1999.
- 36 S. Joshi and B. König. Applying the graph minor theorem to the verification of graph transformation systems. In *CAV*, volume 5123 of *LNCS*, pages 214–226. Springer, 2008.
- 37 M. Kabil and M. Pouzet. Une extension d’un théorème de P. Jullien sur les âges de mots. *ITA*, 26:449–484, 1992.
- 38 E. Kopczynski. Invisible pushdown languages. In *LICS*, pages 867–872, 2016.
- 39 D. Kozen. *Automata and computability*. Undergraduate texts in Computer Science. Springer, 1997.
- 40 R. Lazic and S. Schmitz. The ideal view on rackoff’s coverability technique. In *RP*, pages 76–88, 2015.
- 41 J. Leroux and S. Schmitz. Demystifying reachability in vector addition systems. In *LICS*, pages 56–67, 2015.
- 42 R. J. Lipton. The reachability problem requires exponential space. Technical report, Yale University, Department of Computer Science, 1976.
- 43 Z. Manna and A. Pnueli. *Temporal verification of reactive systems - safety*. Springer, 1995.
- 44 A. Marccone. Foundations of bqo theory. *Transactions of the American Mathematical Society*, 345(2):641–660, 1994.
- 45 M. Martos-Salgado and F. Rosa-Velardo. Dynamic networks of timed Petri nets. In *Petri nets*, pages 294–313, 2014.
- 46 R. Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 1-3(297):337–354, 2003.
- 47 R. Meyer. On boundedness in depth in the pi-calculus. In *TCS*, volume 273 of *IFIP*, pages 477–489. Springer, 2008.
- 48 M. Mukund, K. N. Kumar, J. Radhakrishnan, and M. A. Sohoni. Robust asynchronous protocols are finite-state. In *ICALP*, pages 188–199, 1998.

- 49 M. Mukund, K. N. Kumar, J. Radhakrishnan, and M. A. Sohoni. Towards a characterisation of finite-state message-passing systems. In *ASIAN*, pages 282–299, 1998.
- 50 T. Place, L. van Rooijen, and M. Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. In *FSTTCS*, pages 363–375, 2013.
- 51 T. Place, L. van Rooijen, and M. Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *MFCS*, pages 729–740, 2013.
- 52 T. Place and M. Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *ICALP*, pages 342–353, 2014.
- 53 T. Place and M. Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12(1), 2016.
- 54 F. Rosa-Velardo and M. Martos-Salgado. Multiset rewriting for the verification of depth-bounded processes with name binding. *Inf. Comput.*, 215:68–87, 2012.
- 55 S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman’s lemma. In *ICALP*, volume 6756 of *LNCS*, pages 441–452. Springer, 2011.
- 56 T. G. Szymanski and J. H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976.
- 57 T. Wies, D. Zufferey, and T. A. Henzinger. Forward analysis of depth-bounded processes. In *FOSSACS*, volume 6014 of *LNCS*, pages 94–108. Springer, 2010.
- 58 M. De Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In *CAV*, volume 4144 of *LNCS*, pages 17–30. Springer, 2006.