Fundamenta Informaticae 189(3-4) : 261–296 (2022)
Available at IOS Press through:
https://doi.org/10.3233/FI-222161

On the Complexity of Techniques That Make Transition Systems Implementable by Boolean Nets

Raymond Devillers
Département d'Informatique
Université Libre de Bruxelles
Boulevard du Triomphe
C.P. 212, B-1050 Bruxelles, Belgium
raymond.devillersl@ulb.be

Ronny Tredup*
Geschwister-Scholl-Gymnasium Löbau
Pestalozzistraße 21, 02708 Löbau Germany
tredupron@gsg-loebau.lernsax.de

Abstract. Let us consider some class of (Petri) nets. The corresponding *Synthesis* problem consists in deciding whether a given labeled transition system (TS) A can be implemented by a net N of that class. In case of a negative decision, it may be possible to convert A into an implementable TS B by applying various modification techniques, like relabeling edges that previously had the same label, suppressing edges/states/events, etc. It may however be useful to limit the number of such modifications to stay close to the original problem, or optimize the technique. In this paper, we show that most of the corresponding problems are NP-complete if the considered class corresponds to so-called flip-flop nets or some flip-flop net derivatives.

Keywords: Petri net, Boolean Types, Label-Splitting, Edge-Removal, Event-Removal, State-Removal, Synthesis, Complexity

*Address for correspondence: Geschwister-Scholl-Gymnasium Löbau, Pestalozzistraße 21, 02708 Löbau Germany

1. Introduction

The so-called synthesis problem for nets of some class τ (τ-synthesis, for short) consists in deciding whether for a given labeled transition system (TS, for short) A there is a net N of class τ (a τ-net, for short) that implements A. In case of a positive decision, N should be constructed, possibly while minimizing some features (number of places, arc weights, ...); in case of a negative decision, some reason(s) should be given.

τ-Synthesis is used to, for example, extract concurrency from sequential specifications like TS and languages [1] and has applications in, for example, process discovery [2], supervisory control [3] and the synthesis of speed independent circuits [4].

However, whether N exists depends crucially on the kind of implementation we are striving at, that is, whether N should be an (exact) *realization* (meaning that A and N's reachability graph are isomorphic), or a *language-simulation* (meaning that A and N have the same language) or an *embedding* (meaning that N preserves the distinctness of states of A). Unfortunately, whatever the kind of implementation, a solution does not always exist. This observation motivates the search for techniques that modify the given TS (as little as possible) so that the result is an implementable behavior.

For instance, *label-splitting* has been considered in [5, 6, 7, 8]: This approach may convert a non-implementable TS A into an implementable one A' by relabeling differently some edges that previously had the same label. However, the new events produced by the label-splitting increase the complexity of the derived net, since each new copy will be transformed into a new transition. Hence, it is desired to find a label-splitting that induces the minimal number of transitions in the sought net. This allows to consider *τ-label-splitting* as a decision problem with input A and $\kappa \in \mathbb{N}$; the question is whether there is a TS B implementable by a τ-net that, firstly, is derived from A by splitting labels and, secondly, has at most κ labels (then, by a dichotomic search, an optimal splitting may be found). Recently, in [9], it has been shown that τ-label-splitting aiming at embedding is NP-complete if τ equals the class of *weighted Place/Transition*-nets. Moreover, in [10], we have shown that τ-label-splitting aiming at language-simulation or realization is also NP-complete for this type.

Since label-splitting is intractable (at least for weighted Place/Transition nets), other techniques with better worst-case complexity would be preferable. This led to the idea of simplifying an unimplementable TS by removing some edges, events or states until the result is implementable. Indeed, the removal of components is a strong technique that always leads to implementable behavior, since the result is implementable when only the initial state is left at the end. However, such an extreme modification is certainly not desirable. Instead, we aim to remove the minimum number κ of components so that the result is implementable. This justifies to consider the decision problems *edge-, event-, and state-removal*: given a TS A, and a number κ, the task is to decide whether we can modify A to an implementable TS B by the removal of at most κ edges, events or states, respectively. Unfortunately, it has been shown that these removal techniques are also NP-complete for all implementations - embedding, language-simulation and realization - if we target (weighted) Place/Transition nets [11, 12].

Naturally, it raises the question whether the decision problems, and thus the corresponding modification techniques, are of a different complexity provided the net class sought is different.

A whole family of net classes for which such investigations are certainly of interest is defined by the so-called *Boolean types of nets* [5, 13, 14, 15, 16, 17, 18], since the respective nets are widely accepted as excellent tools for modeling concurrent and distributed systems.

Boolean nets allow at most one token on each place p in every reachable marking. Therefore, p is considered a Boolean condition that is *true* if p is marked and *false* otherwise. A place p and a transition t of a Boolean net N are related by one of the following Boolean *interactions* (partial functions): *no operation* (nop), *input* (inp), *output* (out), *unconditionally set to true* (set), *unconditionally reset to false* (res), *inverting* (swap), *test if true* (used), and *test if false* (free). The relation between p and t determines which conditions p must satisfy to allow t's firing (the corresponding partial function must be defined), and which impact has the firing of t on p (the partial function determines what should be the new value of p if t is executed). Boolean nets are then classified by the interactions of $I = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{swap}, \mathsf{used}, \mathsf{free}\}$ that they apply or spare. More exactly, a subset $\tau \subseteq I$ is called a *Boolean type of net* and a net N is of type τ (a τ-net) if it applies at most the interactions of τ.

However, the determination of the complexity of TS modifications allowing τ-synthesis does not actually define an open problem for all Boolean types of nets. In particular, it is known that τ-synthesis aiming at language-simulation or realization is NP-complete for 84 out of the 128 Boolean types of nets containing nop (which allows some kind of independence between places and transitions) [19, 20, 21, 22, 20]. Likewise, it is known that τ-synthesis striving at embedding is NP-complete for 90 of the nop-equipped Boolean types [23]. Consequently, for these types, the NP-completeness of τ-synthesis implies already the NP-completeness of the corresponding problems *label-splitting*, and *edge-, event-*, and *state-removal*, for instance by choosing κ in a way that forbids any splitting, or removal, respectively. This puts the Boolean types with a tractable synthesis problem in focus. One of the most prominent Boolean type that fulfills this criterion is the so-called *flip-flop* type of nets, where $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$. Flip-flop nets have been originally introduced in [18], and their name is inspired by the interaction swap that allows a transition to unconditionally change the current marking of a place from 0 to 1 and from 1 to 0. The flip-flop nets are considered as the Boolean counterpart of the Place/Transition nets. This characterization is mainly based on the fact that synthesis aiming at flip-flop nets is solvable by a polynomial time algorithm [18] that is derived from the algorithm for synthesis aiming at Place/Transition nets, which has been introduced in [24]. In fact, the algorithm for flip-flop nets [18] is extendable to all types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$ [21], which makes their synthesis problem also tractable.

In this paper, for all 16 types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, hence in particular for the flip-flop nets, we investigate the computational complexity of τ-label-splitting and element removing for all introduced implementations: embedding, language-simulation and realization. In particular, we show that this problem aiming at embedding is NP-complete for all these types, unfortunately. Moreover, label-splitting aiming at language-simulation or realization is NP-complete if $\omega \neq \emptyset$, otherwise it is tractable.

Furthermore, for all 15 types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$ and $\omega \neq \emptyset$, we investigate the computational complexity of τ-edge-, event-, and state-removal, and show that these problems are NP-complete for all these types, regardless which of the implementations embedding, language-simulation and realization we are aiming at.

We obtain our NP-completeness results by reductions from a variant of the well-known vertex cover problem. Our current approach generalizes our methods from [10, 12] and tailors them to flip-flop nets and its aforementioned derivatives.

This paper, which is an extended version of [25], is organized as follows. The next Section 2 introduces necessary notions and definitions. After that, Sections 3 to 6 present our complexity results for label-splitting and the edge/event/state-removals. Finally, Section 7 briefly closes the paper.

2. Preliminaries

This section introduces the basic notions used throughout the paper.

Definition 2.1. (Transition Systems)
A *transition system* (TS) $A = (S, E, \delta)$ is a finite directed labeled graph with the set of nodes S (called *states*), the set of labels E (called *events*) and partial *transition function* $\delta : S \times E \longrightarrow S$.

We shall assume no event is useless, i.e., for every $e \in E$ there are states $s, s' \in S$ such that $\delta(s, e) = s'$. For convenience, with a little abuse of notation, we often identify δ and the set $\{(s, e, s') \in S \times E \times S \mid \delta(s, e) = s'\}$.

Event e *occurs* at s, denoted by $s \xrightarrow{e}$, if $\delta(s, e)$ is defined, otherwise we denote it by $s \xrightarrow{\neg e}$ or $\neg s \xrightarrow{e}$. We denote $\delta(s, e) = s'$ by $s \xrightarrow{e} s'$.

An *initialized* TS $A = (S, E, \delta, \iota)$ is a TS with a distinct *initial* state $\iota \in S$ such that every state $s \in S$ is *reachable* from ι by a directed labeled path. If $w = e_1 \dots e_n \in E^*$, by $\iota \xrightarrow{w}$ we denote that there are states $\iota = s_0, \dots, s_n \in S$ such that $s_i \xrightarrow{e_{i+1}} s_{i+1} \in A$ for all $i \in \{0, \dots, n-1\}$, in which case we also denote $\iota \xrightarrow{w} s_n$. We extend this notation by stating that $\iota \xrightarrow{\epsilon} \iota$.

The *language* of A is defined by $L(A) = \{w \in E^* \mid \iota \xrightarrow{w}\}$.

When not given explicitly, we shall refer to the components of a TS A by $S(A)$ (states), $E(A)$ (events), δ_A (transition function) and ι_A (initial state).

Definition 2.2. (Simulations)
Let A and B be initialized TS with the same set of events E. We say B *simulates* A, if there is a mapping $\varphi : S(A) \to S(B)$ such that $\varphi(\iota_A) = \iota_B$ and $s \xrightarrow{e} s' \in A$ implies $\varphi(s) \xrightarrow{e} \varphi(s') \in B$; such a mapping is called a *simulation* (between A and B).

φ is an *embedding*, denoted by $A \hookrightarrow B$, if it is injective; φ is a *language-simulation*, denoted by $A \triangleright B$, if $\varphi(s) \xrightarrow{e}$ implies $s \xrightarrow{e}$, implying $L(A) = L(B)$ [5, p. 67]; φ is an *isomorphism*, denoted by $A \cong B$, if it is both an embedding and a language simulation.

Definition 2.3. (Boolean Types [5])
A Boolean type is a subset τ of the 8 Boolean interactions

$$I = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{set}, \mathsf{res}, \mathsf{swap}, \mathsf{used}, \mathsf{free}\}$$

i.e., the partial functions $\{0,1\} \to \{0,1\}$ schematized in Figure 1. To each type τ, we associate the Boolean[1] TS $A_\tau = (\{0,1\}, E_\tau, \delta_\tau)$, where $E_\tau = \tau$ and, for each $x \in \{0,1\}$ and $i \in E_\tau$, $\delta_\tau(x,i)$ is defined if so is $i(x)$, in which case $\delta_\tau(x,i) = i(x)$. Since the correspondence is unique, we often identify τ with A_τ.

In fact there are $(2+1)^2 = 9$ such partial functions, but the empty one (defined nowhere) is of no interest here, since it may never occur as the label of an event.

x	$\mathsf{nop}(x)$	$\mathsf{inp}(x)$	$\mathsf{out}(x)$	$\mathsf{set}(x)$	$\mathsf{res}(x)$	$\mathsf{swap}(x)$	$\mathsf{used}(x)$	$\mathsf{free}(x)$
0	0		1	1	0	1		0
1	1	0		1	0	0	1	

Figure 1: All interactions i of I. If a cell is empty, then i is undefined on the respective x.

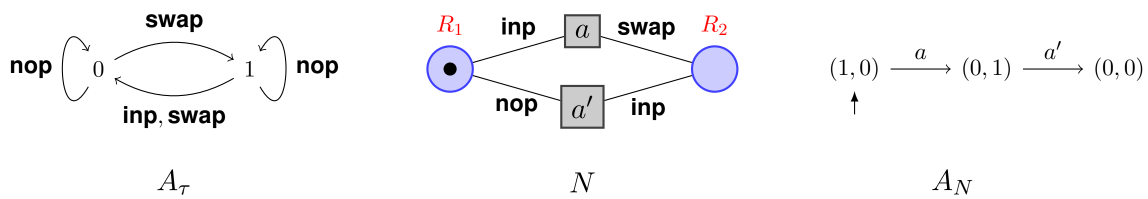

Figure 2: Left: A_τ for the the type $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{swap}\}$. Middle: A τ-net N (as usual, the initial marking is indicated by putting a token in the places p such that $M_0(p) = 1$). Right: The reachability graph A_N of N, where each state M is represented by its marking $(M(R_1), M(R_2))$, and the initial state is indicated by the arrow without source state.

Definition 2.4. **(τ-Nets)**
Let $\tau \subseteq I$. A Boolean net $N = (P, T, f, M_0)$ of type τ (a *τ-net*) is given by finite disjoint sets P of *places* and T of *transitions*, a (total) *flow-function* $f : P \times T \to \tau$, and an *initial marking* $M_0 : P \longrightarrow \{0,1\}$. A transition $t \in T$ can *fire* in a marking $M : P \longrightarrow \{0,1\}$ if $\delta_\tau(M(p), f(p,t))$ is defined for all $p \in P$. By firing, t produces the marking $M' : P \longrightarrow \{0,1\}$ where $M'(p) = \delta_\tau(M(p), f(p,t))$ for all $p \in P$, denoted by $M \xrightarrow{t} M'$. The behavior of τ-net N is captured by an initialized transition system A_N, called the *reachability graph* of N. The states set $S(A_N)$ of A_N will also be denoted $RS(N)$; it consists of all markings that can be reached from the initial state M_0 by sequences of transition firings; $E(A_N) = T$, $\iota_{A_N} = M_0$, and $(s, e, s') \in \delta_{A_N}$ iff $s \xrightarrow{e} s'$ in N.

Example 2.5. Figure 2 shows the transition system A_τ for the type $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{swap}\}$, the τ-net $N = (\{R_1, R_2\}, \{a, a'\}, f, M_0)$ with places R_1, R_2, flow-function $f(R_1, a) = f(R_2, a') = \mathsf{inp}$, $f(R_1, a') = \mathsf{nop}$, $f(R_2, a) = \mathsf{swap}$ and initial marking M_0 defined by $(M_0(R_1), M_0(R_2)) = (1, 0)$. Since $1 \xrightarrow{\mathsf{inp}} 0 \in \tau$ and $0 \xrightarrow{\mathsf{swap}} 1 \in \tau$, the transition a can fire in M_0, which leads to the

[1] Meaning that the state set $S = \{0,1\}$. The initial state is irrelevant here.

marking $M = (M(R_1), M(R_2)) = (0,1)$. After that, a' can fire, which results in the marking $M' = (M'(R_1), M'(R_2)) = (0,0)$. The reachability graph A_N of N is depicted on the right hand side of Figure 2.

Definition 2.6. (Implementations)
Let A be an initialized TS, τ be a Boolean type and N a τ-net. We say N is an (exact) *realization* of A if $A \cong A_N$. If $A \triangleright A_N$, then N is a *language-simulation* of A. If $A \hookrightarrow A_N$, then N is an embedding of A.

Remark 2.7. By definition, the reachability graph of a net is initialized. In the following, we shall thus only consider initialized transition systems without explicitly mention it every time.

Let τ be a Boolean type of nets. If a TS A is implementable by a τ-net N, then we want to construct N from the structure of A. Since A_N has to simulate A, N's transitions correspond to A's events. The connection between global states in TS and local states in the sought net is given by *regions of TS* that mimic places:

Definition 2.8. (τ-Regions)
A τ-region $R = (sup, sig)$ of $A = (S, E, \delta, \iota)$ consists of the *support* $sup : S \to \{0,1\}$ and the *signature* $sig : E \to E_\tau$ where every edge $s \xrightarrow{e} s'$ of A leads to an edge $sup(s) \xrightarrow{sig(e)} sup(s')$ of type τ. If $P = q_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} q_n$ is a path in A, then $P^R = sup(q_0) \xrightarrow{sig(e_1)} \dots \xrightarrow{sig(e_n)} sup(q_n)$ is a path in τ. We say P^R is the *image* of P (under R).

Notice that R is *implicitly* completely defined by $sup(\iota)$ and sig: Since A is initialized, for every state $s \in S(A)$, there is a path $\iota \xrightarrow{e_1} \dots \xrightarrow{e_n} s_n$ such that $s = s_n$. Thus, since δ_τ is a (partial) function, we inductively obtain $sup(s_{i+1})$ by $sup(s_{i+1}) = \delta_\tau(sup(s_i), sig(e_{i+1}))$ for all $i \in \{0, \dots, n-1\}$ and $s_0 = \iota$. Consequently, we can compute sup and thus R purely from $sup(\iota)$ and sig (and A), since when two paths lead to the same state the corresponding supports are the same (otherwise (sup, sig) does not define a region); this is illustrated by Example 2.14 and Figure 3.

$$\begin{array}{ccc} t_0 \xrightarrow{\quad a \quad} t_1 \xrightarrow{\quad a \quad} t_2 & \qquad t_0 \xrightarrow{\quad a \quad} t_1 \xrightarrow{\quad a' \quad} t_2 & \qquad 1 \xrightarrow{\quad \mathsf{inp} \quad} 0 \xrightarrow{\quad \mathsf{nop} \quad} 0 \\ \uparrow \qquad\qquad\qquad & \uparrow \qquad\qquad\qquad & \\ A & B & B^{R_1} \end{array}$$

Figure 3: Left: The TS A with event set $E = \{a\}$. Middle: The TS B with event set $E' = \{a, a'\}$. Right: The image B^{R_1} of the τ-region $R_1 = (sup_1, sig_1)$ of B, where $sup_1(t_0) = 1$, $sup_1(t_1) = sup_1(t_2) = 0$, $sig_1(a) = \mathsf{inp}$ and $sig_1(a') = \mathsf{nop}$, and $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{swap}\}$. Later, we shall also represent a region by a color convention indicating the support of each state.

Every set $\mathcal{R}$ of τ-regions of A implies a particular *synthesized* τ-net, where the regions model places and the associated part of the flow-function:

Definition 2.9. (Synthesized net)
Let $A = (S, E, \delta, \iota)$ be a TS, τ a Boolean type and $\mathcal{R}$ a set of τ-regions of A. The *synthesized net* (indicated by A and $\mathcal{R}$) is defined by $N_A^{\mathcal{R}} = (\mathcal{R}, E, f, M_0)$ where, for all $R = (sup, sig) \in \mathcal{R}$ and all $e \in E$, we have that $f(R, e) = sig(e)$ and $M_0(R) = sup(\iota)$.

It can be shown that there is always a (unique) simulation φ between A and the reachability graph $A_{N_A^{\mathcal{R}}}$ of the synthesized net $N_A^{\mathcal{R}}$ where, for all $s \in S(A)$ and all $R \in \mathcal{R}$, we have that $sup(s) = M(R)$ for the marking M of $N_A^{\mathcal{R}}$ that satisfies $\varphi(s) = M$. However, to ensure that φ is an embedding, we have to distinguish global states, and to ensure that φ is a language-simulation, we have to prevent the firings of transitions when their corresponding events are not present in TS. This is stated as *separation atoms* and *properties*.

Definition 2.10. (τ-State Separation and Property)
A pair (s, s') of distinct states of a A defines a *states separation atom* (SSA). A τ-region $R = (sup, sig)$ *solves* (s, s') if $sup(s) \neq sup(s')$; then this SSA is said τ-solvable. If every SSA of A is τ-solvable then A has the τ-*states separation property* (τ-SSP, for short).

Definition 2.11. (τ-Event Separation and Property)
A pair (e, s) of event $e \in E$ and state $s \in S$ where e does not occur, that is $\neg s \xrightarrow{e}$, defines an *event/state separation atom* (ESSA). A τ-region $R = (sup, sig)$ *solves* (e, s) if $sig(e)$ is not defined on $sup(s)$ in τ, that is, $\neg sup(s) \xrightarrow{sig(e)}$; then this ESSA is said τ-solvable. If every ESSA of A is τ-solvable then A has the τ-*event state separation property* (τ-ESSP, for short).

Definition 2.12. (τ-Witness)
A set $\mathcal{R}$ of τ-regions of A is called a τ-*witness* of A's τ-SSP, respectively τ-ESSP, if for each SSA, respectively ESSA, there is a τ-region R in $\mathcal{R}$ that solves it.

The next lemma ([5, p. 162], Proposition 5.10) establishes the connection between the existence of τ-witnesses and the existence of an implementing τ-net N for the witnessed property:

Lemma 2.13. ([5])
Let A be a TS, τ a Boolean type and N a τ-net. The following statements are true:

1. $A \hookrightarrow A_N$ if and only if there is a τ-witness $\mathcal{R}$ of the τ-SSP of A and $N = N_A^{\mathcal{R}}$.
2. $A \triangleright A_N$ if and only if there is a τ-witness $\mathcal{R}$ of the τ-ESSP of A and $N = N_A^{\mathcal{R}}$.
3. $A \cong A_N$ if and only if there is a τ-witness $\mathcal{R}$ of both the τ-SSP and the τ-ESSP of A and $N = N_A^{\mathcal{R}}$.

Example 2.14. Let τ be defined like in Figure 2 and A, B, B^{R_1} like in Figure 3. The TS A has neither the τ-SSP nor the τ-ESSP, since the atoms (t_0, t_2) and (a, t_2) are not τ-solvable. The TS B has both the τ-SSP and τ-ESSP. The region $R_1 = (sup_1, sig_1)$ that solves (t_0, t_1), (t_0, t_2), (a, t_1) and (a, t_2) is implicitly defined by $sup_1(t_0) = 1$, $sig_1(a) = \mathsf{inp}$ and $sig_1(a') = \mathsf{nop}$. We obtain sup_1 and thus R_1 explicitly by $sup_1(t_1) = \delta_\tau(1, \mathsf{inp}) = 0$ and $sup_1(t_2) = \delta_\tau(0, \mathsf{nop}) = 0$. B^{R_1} shows the image of B under R_1. The remaining (E)SSP atoms (t_1, t_2), (a', t_0) and (a', t_1) are solved by the following τ-region $R_2 = (sup_2, sig_2)$ that is implicitly defined by $sup_2(t_0) = 0$, $sig_2(a) = \mathsf{swap}$ and $sig_2(a') = \mathsf{inp}$. The set $\mathcal{R} = \{R_1, R_2\}$ is a witness for the τ-(E)SSP of A and the net $N_A^{\mathcal{R}}$ is exactly the net N that is depicted in Figure 2. N is a realization of A, since a bijective simulation φ between A and A_N is given by $\varphi(t_0) = (1, 0)$, $\varphi(t_1) = (0, 1)$ and $\varphi(t_2) = (0, 0)$.

3. The complexity of label-splitting

For a Boolean type τ, τ-synthesis is the task to find, for a given TS A, a τ-net N that implements A. Regardless of which of the implementations (τ-embedding, τ-language-simulation or τ-realization) we are aiming at, a suitable τ-net does not always exists. In this case, *label-splitting* might be a suitable technique to modify A into a TS B that is then implementable:

Definition 3.1. (Label-splitting)
Let $A = (S, E, \delta, \iota)$ be a TS and $e_1, \dots, e_n \in E$ be pairwise distinct events. The *label-splitting* of the events $e_1, \dots, e_n$ into the events $e_1^1, \dots, e_1^{m_1}, \dots, e_n^1, \dots, e_n^{m_n}$ (pairwise distinct, and distinct from the other events in $E \setminus \{e_1, \dots, e_n\}$), where $m_j \geq 2$ for all $j \in \{1, \dots, n\}$, yields the event set $E' = (E \setminus \{e_1, \dots, e_n\}) \cup \bigcup_{i=1}^{n} \{e_i^j \mid j \in \{1, \dots, m_j\}\}$.

A TS $B = (S, E', \delta', \iota)$ is an E'*-label-splitting* (E'-LS, for short) of A if $|\delta| = |\delta'|$ and, for all $s, s' \in S$ and all $e \in E$, the following is true: If $\delta(s, e) = s'$ and $e \notin \{e_1, \dots, e_n\}$, then $\delta'(s, e) = s'$; if $\delta(s, e) = s'$ and $e = e_i$ for some $i \in \{1, \dots, n\}$, then there is exactly one $\ell \in \{1, \dots, m_i\}$ such that $\delta'(s, e_i^\ell) = s'$. We say that $\mathfrak{L} = \{e_1, \dots, e_n\}$ is *the set of events of A that occur split in B*.

Note that, in practice, e_j^1 is usually chosen as the original event e_j for some or all $j \in \{1, \dots, n\}$.

Example 3.2. Let A and B be defined like in Figure 3. The TS B is an E'-label-splitting of A, where $E' = (E \setminus \{a\}) \cup \{a, a'\}$.

To be as close as possible to the original behavior A, a corresponding E'-label-splitting B of A should change A as little as possible, which means that the number of events of B should be as small as possible. This gives rise to consider label-splitting as a decision problem that, for a given TS A and a natural number κ, asks whether there is an E'-label-splitting B of A that is implementable and uses at most κ labels, i.e., $|E'| \leq \kappa$.

By Lemma 2.13, deciding the existence of an implementing net is equivalent to deciding if the input TS has the property that corresponds to the implementation. Finally, this leads to the following three decision problems that are the main subject of this section:

LS-τ-EMBEDDING	
Input:	a TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an E'-label-splitting B of A with $\lvert E'\rvert \leq \kappa$ that has the τ-SSP?

LS-τ-LANGUAGE-SIMULATION	
Input:	a TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an E'-label-splitting B of A with $\|E'\| \leq \kappa$ that has the τ-ESSP?

LS-τ-REALISATION	
Input:	a TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an E'-label-splitting B of A with $\|E'\| \leq \kappa$ that has both the τ-SSP and the τ-ESSP?

The following theorem presents the main result of this section:

Theorem 3.3. If $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, then

1. LS-τ-EMBEDDING is NP-complete,
2. LS-τ-LANGUAGE-SIMULATION, and LS-τ-REALISATION are NP-complete if $\omega \neq \emptyset$, otherwise they are in P.

First of all, we argue for the polynomial part, where we need the τ-ESSP: If $\tau = \{\mathsf{nop}, \mathsf{swap}\}$[2], then a TS $A = (S, E, \delta, \iota)$ has the τ-ESSP if and only if every event occurs at every state. Indeed, if every event occurs at every state there is no ESSA that needs to be solved. On the contrary, if some event e is missing from some state s, since all the allowed functions are defined on both 0 and 1, it is not possible to solve the ESSA (e, s) of A. This implies that an E'-label-splitting B of A that reflects an actual splitting, meaning that $|E'| > |E|$, can never yield a TS that has the τ-ESSP, since it would produce at least one unsolvable ESSA: If $|E'| > |E|$, then there is an event e_i in A for which there are edges $s \xrightarrow{e_i^j} s'$, and $t \xrightarrow{e_i^\ell} t'$ in B with $s \neq t$, and $e_i^j \neq e_i^\ell$ (since δ_A, and δ_B are functions, and every event has an occurrence by Definition 2.1), which were previously both labeled by e_i. This implies $\neg s \xrightarrow{e_i^\ell}$ (by δ_A being a function), and thus (s, e_i^ℓ) is not solvable. The same argument shows that, for the realization problem, if A has the ESSP (meaning that every event occurs at every state as explained before) but has an unsolvable SSA, then any label-splitting introduced to solve this atom would destroy the ESSP. Hence, for this Boolean type, the decision problems are trivial: either A is already implementable (which can be can be checked in polynomial time [18, 21]) or it has to be rejected. Thus, for the proof of Theorem 3.3(2), it remains to consider the cases when $\omega \neq \emptyset$, hence the NP-completeness results.

The decision problems LS-τ-EMBEDDING, LS-τ-LANGUAGE-SIMULATION as well as LS-τ-REALISATION are in NP: If a sought E'-label-splitting $B = (S, E', \delta', \iota)$ of a TS $A = (S, E, \delta, \iota)$ exists, then a Turing-machine M can compute B in a non-deterministic computation in time polynomial in the size $|\delta|$ of A, since $|\delta| = |\delta'|$. After that, M verifies in time polynomial in the size $|\delta'|$ of B (and thus of A) that it allows the sought implementation [18, 21].

[2]This remains true if $\tau \subseteq \{\mathsf{nop}, \mathsf{swap}, \mathsf{set}, \mathsf{res}\}$, but $\mathsf{set}/\mathsf{res}$ are not used in this paper.

Recall that a *non-directed graph* $G = (\mathfrak{U}, M)$ consists of a (finite) set $\mathfrak{U}$ of *vertices*, and a set M of (non-directed) *edges* over $\mathfrak{U}$, that is, for all $\mathfrak{e} \in M$, we have that $\mathfrak{e} \subseteq \mathfrak{U}$, and $|\mathfrak{e}| = 2$.

Our NP-completeness proofs are based on reductions from the following classical variant of the vertex cover (VC) problem [26, p. 190]:

3-BOUNDED VERTEX COVER (3BVC)	
Input:	a non-directed Graph $G = (\mathfrak{U}, M)$ such that every vertex $v \in \mathfrak{U}$ is a member of at most three distinct edges, and a natural number $\lambda \in \mathbb{N}$.
Question:	Does there exist a λ-vertex cover (λ-VC, for short) of G, that is a subset $\mathcal{S} \subseteq \mathfrak{U}$ with $|\mathcal{S}| \leq \lambda$ and $\mathcal{S} \cap \mathfrak{e} \neq \emptyset$ for all $\mathfrak{e} \in M$?

Example 3.4. (3BVC)
The instance $(G, 2)$, illustrated in Figure 4, where $G = (\mathfrak{U}, M)$ such that $\mathfrak{U} = \{v_0, v_1, v_2, v_3\}$ and $M = \{M_0, \dots, M_4\}$, where $M_0 = \{v_0, v_1\}, M_1 = \{v_0, v_2\}, M_2 = \{v_0, v_3\}, M_3 = \{v_1, v_2\}$, and $M_4 = \{v_2, v_3\}$, is a yes-instance of 3BVC, since $\mathcal{S} = \{v_0, v_2\}$ is a 2-VC of G.

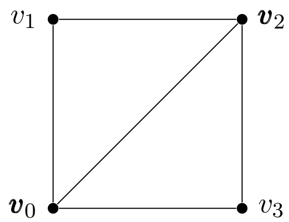

Figure 4: A running graph example G; a 2-VC is $\{\boldsymbol{v_0}, \boldsymbol{v_2}\}$ (in bold).

In the remainder of this paper, let (G, λ) be an input of 3BVC, where $G = (\mathfrak{U}, M)$ is a graph with n vertices $\mathfrak{U} = \{v_0, \dots, v_{n-1}\}$ and m edges $M = \{M_0, \dots, M_{m-1}\}$ such that $M_i = \{v_{i_0}, v_{i_1}\}$ and $i_0 < i_1$ for all $i \in \{0, \dots, m-1\}$.

For the proof of Theorem 3.3, we polynomially reduce (G, λ) to a pair (A_G, κ) of TS $A_G = (S, E, \delta, \bot_0)$ and natural number κ such that the following conditions are satisfied:

1. If there is an E'-label-splitting of A_G that satisfies $|E'| \leq \kappa$ and has the τ-SSP or the τ-ESSP, then G has a λ-VC.
2. If G has a λ-VC, then there is an E'-label-splitting of A_G that satisfies $|E'| \leq \kappa$ and has both the τ-SSP and the τ-ESSP.

Obviously, a polynomial-time reduction that satisfies Condition 1 and Condition 2 ensures that G allows a λ-VC if and only if A_G allows an E'-label-splitting that satisfies $|E'| \leq \kappa$ and has the τ-SSP, the τ-ESSP or both, according to which property is sought. Hence, it proves Theorem 3.3.

3.1. The proof of Theorem 3.3(1)

In the remainder of this section we assume that $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$, with an arbitrary but fixed subset $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$.

For a start, we define $\kappa = n + 2m - 1 + \lambda$, where $n + 2m - 1$ is the number of events of the aforementioned TS A_G. Hence, λ is the maximum number of events of A_G that could potentially be split in an E'-label-splitting B_G of A_G.

For every $i \in \{0, \dots, m-1\}$, the TS A_G has the following directed path T_i that uses the vertices v_{i_0} and v_{i_1} of the edge M_i as events:

$$T_i = t_{i,0} \xrightarrow{v_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2} \xrightarrow{v_{i_0}} t_{i,3} \xrightarrow{v_{i_1}} t_{i,4}$$

Finally, for all $i \in \{0, \dots, m-1\}$, we apply the edge $\bot_i \xrightarrow{w_i} t_{i,0}$ and, if $i < m - 1$, then also the edge $\bot_i \xrightarrow{\ominus_{i+1}} \bot_{i+1}$ to connect the paths $T_0, \dots, T_{m-1}$ into the TS A_G, cf. Figure 5. Let $\bot = \{\bot_0, \dots, \bot_{m-1}\}$ and $W = \{w_0, \dots, w_{m-1}\}$ and $\ominus = \{\ominus_1, \dots, \ominus_{m-1}\}$. The TS A_G has exactly $|V \cup W \cup \ominus| = n + 2m - 1$ events.

$$\begin{tikzcd}
{} \arrow[r] & \bot_0 \arrow[r, "w_0"] \arrow[d, "\ominus_1"'] & t_{0,0} \arrow[r, "v_0"] & t_{0,1} \arrow[r, "v_1"] & t_{0,2} \arrow[r, "v_0"] & t_{0,3} \arrow[r, "v_1"] & t_{0,4} \\
& \bot_1 \arrow[r, "w_1"] \arrow[d, "\ominus_2"'] & t_{1,0} \arrow[r, "v_0"] & t_{1,1} \arrow[r, "v_2"] & t_{1,2} \arrow[r, "v_0"] & t_{1,3} \arrow[r, "v_2"] & t_{1,4} \\
& \bot_2 \arrow[r, "w_2"] \arrow[d, "\ominus_3"'] & t_{2,0} \arrow[r, "v_0"] & t_{2,1} \arrow[r, "v_3"] & t_{2,2} \arrow[r, "v_0"] & t_{2,3} \arrow[r, "v_3"] & t_{2,4} \\
& \bot_3 \arrow[r, "w_3"] \arrow[d, "\ominus_4"'] & t_{3,0} \arrow[r, "v_1"] & t_{3,1} \arrow[r, "v_2"] & t_{3,2} \arrow[r, "v_1"] & t_{3,3} \arrow[r, "v_2"] & t_{3,4} \\
& \bot_4 \arrow[r, "w_4"] & t_{4,0} \arrow[r, "v_2"] & t_{4,1} \arrow[r, "v_3"] & t_{4,2} \arrow[r, "v_2"] & t_{4,3} \arrow[r, "v_3"] & t_{4,4}
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[r] & \bot_0 \arrow[r, "w_0"] \arrow[d, "\ominus_1"'] & t_{0,0} \arrow[r, "v_0'"] & t_{0,1} \arrow[r, "v_1"] & t_{0,2} \arrow[r, "v_0"] & t_{0,3} \arrow[r, "v_1"] & t_{0,4} \\
& \bot_1 \arrow[r, "w_1"] \arrow[d, "\ominus_2"'] & t_{1,0} \arrow[r, "v_0"] & t_{1,1} \arrow[r, "v_2"] & t_{1,2} \arrow[r, "v_0'"] & t_{1,3} \arrow[r, "v_2'"] & t_{1,4} \\
& \bot_2 \arrow[r, "w_2"] \arrow[d, "\ominus_3"'] & t_{2,0} \arrow[r, "v_0'"] & t_{2,1} \arrow[r, "v_3"] & t_{2,2} \arrow[r, "v_0"] & t_{2,3} \arrow[r, "v_3"] & t_{2,4} \\
& \bot_3 \arrow[r, "w_3"] \arrow[d, "\ominus_4"'] & t_{3,0} \arrow[r, "v_1"] & t_{3,1} \arrow[r, "v_2"] & t_{3,2} \arrow[r, "v_1"] & t_{3,3} \arrow[r, "v_2'"] & t_{3,4} \\
& \bot_4 \arrow[r, "w_4"] & t_{4,0} \arrow[r, "v_2'"] & t_{4,1} \arrow[r, "v_3"] & t_{4,2} \arrow[r, "v_2"] & t_{4,3} \arrow[r, "v_3"] & t_{4,4}
\end{tikzcd}$$

Figure 5: The transition systems A_G (top) and B_G (bottom) that originate from Example 3.4. They will serve for illustrating some region constructions in the proofs below.

In order to prove Theorem 3.3(1) we shall show that A_G allows a label-splitting B_G restricted by κ, and having the τ-SSP if and only if there is a λ-VC for G.

The following lemma implies that if a TS has any of the introduced paths, then it does not have the τ-SSP:

Lemma 3.5. Let A be a TS that has the path $P_0 = s_0 \xrightarrow{a} s_1 \xrightarrow{b} s_2 \xrightarrow{a} s_3 \xrightarrow{b} s_4$. If $R = (sup, sig)$ is a τ-region of A, then $sup(s_0) = sup(s_4)$.

Proof:
Let $R = (sup, sig)$ be an arbitrary but fixed region of A. If $sup(s_0) \neq sup(s_4)$, then the image P^R is a path from 0 to 1 or from 1 to 0 in τ. This implies that the number of state changes between 0 and 1 on P^R must be odd. Since $sig(a), sig(b) \in \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{swap}, \mathsf{used}, \mathsf{free}\}$ and both a and b occur twice, i.e. an even number of times, this is impossible. □

Note that this lemma would not be true if set and/or res would be allowed in τ: for instance, we have $0 \xrightarrow{\mathsf{set}} 1 \xrightarrow{\mathsf{nop}} 1 \xrightarrow{\mathsf{set}} 1 \xrightarrow{\mathsf{nop}} 1$.

Hence, if a TS has the path T_i for some $i \in \{0, \dots, m-1\}$, then the SSP atom $(t_{i,0}, t_{i,4})$ is not τ-solvable by Lemma 3.5. The following lemma states that this implies a λ-VC of G if a sought E'-label-splitting B_G of A_G exists:

Lemma 3.6. If there is an E'-label-splitting B_G of A_G such that $|E'| \leq \kappa$ that has the τ-SSP, then G has a λ-VC.

Proof:
Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed, and let $\mathfrak{L}$ be the set of events of A_G that occur split in B_G. By Lemma 3.5, the SSP atom $\alpha_i = (t_{i,0}, t_{i,4})$ is not τ-solvable by regions of A_G. However, since B_G has the τ-SSP, the atom α_i is τ-solvable in B_G. This implies $\{v_{i_0}, v_{i_1}\} \cap \mathfrak{L} \neq \emptyset$. Since i was arbitrary, this is simultaneously true for all paths $T_0, \dots, T_{m-1}$ and thus the set $\mathcal{S} = \mathfrak{U} \cap \mathfrak{L}$ intersects with every edge of G. Moreover, by $|E'| \leq \kappa = n + 2(m-1) + \lambda$, we have $|\mathcal{S}| \leq |\mathfrak{L}| \leq \lambda$. Hence, $\mathcal{S}$ defines a λ-VC of G. □

Conversely, let $\mathcal{S} = \{v_{j_0}, \dots, v_{j_{\lambda-1}}\} \subseteq \mathfrak{U}$ be a λ-VC of G. (Notice that a λ-VC with less than λ states implies one with exactly λ states as longs as $\lambda \leq |\mathfrak{U}|$, which can be reasonably assumed.) In the remainder of this section, we argue that there is a sought E'-label-splitting B_G of A_G. For every $i \in \{0, \dots, \lambda-1\}$, we split the event v_{j_i} into the two events v_{j_i} and v'_{j_i}. This yields $E' = (E \setminus \mathcal{S}) \cup \bigcup_{i=0}^{\lambda-1} \{v_{j_i}, v'_{j_i}\}$. To define the aforementioned E'-label-splitting $B_G = (S, E', \delta', \bot_0)$ of A_G, it suffices to define δ' on the states of $T_0, \dots, T_{m-1}$. In particular, for all $i \in \{0, \dots, m-1\}$, δ' restricted to $S(T_i)$ and $E(T_i)$ yields the path T'_i as follows:

- if $v_{i_0} \in \mathcal{S}$ and $v_{i_1} \notin \mathcal{S}$, then $T'_i = t_{i,0} \xrightarrow{v'_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2}, \xrightarrow{v_{i_0}} t_{i,3} \xrightarrow{v_{i_1}} t_{i,4}$;
- if $v_{i_0}, v_{i_1} \in \mathcal{S}$, then $T'_i = t_{i,0} \xrightarrow{v_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2}, \xrightarrow{v'_{i_0}} t_{i,3} \xrightarrow{v'_{i_1}} t_{i,4}$;
- if $v_{i_0} \notin \mathcal{S}$ and $v_{i_1} \in \mathcal{S}$, then $T'_i = t_{i,0} \xrightarrow{v_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2}, \xrightarrow{v_{i_0}} t_{i,3} \xrightarrow{v'_{i_1}} t_{i,4}$.

The following lemma essentially states that if $sup : S(B_G) \to \{0,1\}$ and $sig : E(B_G) \to \tau$ are mappings that define regions when restricted to $T'_0, \dots, T'_{m-1}$, then they can be extended suitably to a region of B_G:

Lemma 3.7. If $sup : S(B_G) \setminus \bot \to \{0,1\}$ and $sig : E' \setminus (W \cup \ominus) \to \tau$ are mappings such that $s \xrightarrow{e} s' \in B_G$ and $e \notin W \cup \ominus$ imply $sup(s) \xrightarrow{sig(e)} sup(s') \in A_\tau$, then there is a τ-region $R = (sup', sig')$ of B_G that preserves sup and sig as follows:

1. For all $s \in S(B_G)$, if $s \notin \bot$ then $sup'(s) = sup(s)$, otherwise $sup'(s) = 0$.
2. For all $e \in E'$, if $e \notin W \cup \ominus$ then $sig'(e) = sig(e)$; if $e \in \ominus$, then $sig(e) = \mathsf{nop}$; if $i \in \{0, \dots, m-1\}$ and $e = w_i$, if $sup(t_{i,0}) = 0$ then $sig(e) = \mathsf{nop}$, otherwise $sig(e) = \mathsf{swap}$.

Proof:
We argue that $s \xrightarrow{e} s' \in B_G$ implies $sup'(s) \xrightarrow{sig'(e)} sup'(s') \in \tau$. If $e \in W \cup \ominus$, then this is easy to see, since $sup'(\bot_i) = 0$ for all $i \in \{0, \dots, m-1\}$. For $e \notin W \cup \ominus$, the claim follows by the assumptions about sup and sig. □

By the next lemma, a τ-region of T'_i, where T'_i is considered as a TS, whose signature only uses nop and swap is always extendable to a region of B_G:

Lemma 3.8. Let $i \in \{0, \dots, m-1\}$. Let $sup : S(T'_i) \to \{0,1\}$ and $sig : E(T'_i) \to \{\mathsf{nop}, \mathsf{swap}\}$ be mappings such that $s \xrightarrow{e} s' \in T'_i$ implies $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$. There is a τ-region $R = (sup', sig')$ of B_G such that $sup'(s) = sup(s)$ and $sig'(e) = sig(e)$ for all $s \in S(T'_i)$ and $e \in E(T'_i)$.

Proof:
By Lemma 3.7, it suffices to argue that sup and sig are consistently extendable to $T'_0, \dots T'_{i-1}, T'_{i+1}, \dots, T'_{m-1}$. Let $j \in \{0, \dots, m-1\} \setminus \{i\}$, be arbitrary but fixed, and let
$T'_j = t_{j,0} \xrightarrow{e_{j,1}} t_{j,1} \xrightarrow{e_{j,2}} t_{j,2} \xrightarrow{e_{j,3}} t_{j,3} \xrightarrow{e_{j,4}} t_{j,4}$, where $e_{j,1}, \dots, e_{j,4} \in E(T'_j)$ in accordance to the definition of B_G. We obtain $R = (sup', sig')$ as follows. For all $e \in E(B_G) \setminus (W \cup \ominus)$, if $e \in E(T'_i)$, then $sig'(e) = sig(e)$ and otherwise $sig(e) = \mathsf{nop}$; for all $s \in S(T'_i)$, we define $sup'(s) = sup(s)$; for all $j \in \{0, \dots, m-1\} \setminus \{i\}$, we define $sup(t_{j,0}) = 0$ and inductively $sup(t_{j,\ell}) = \delta_G(sup(t_{j,\ell-1}), e_{j,\ell})$ for all $\ell \in \{1, \dots, 4\}$. Since sig maps to $\{\mathsf{nop}, \mathsf{swap}\}$, so does sig'. Thus, if $s \xrightarrow{e} s' \in T'_j$, then $sup'(s) \xrightarrow{sig'(e)} sup'(s') \in A_\tau$. Since j was arbitrary, this proves the lemma. □

Lemma 3.9. The TS B_G has the τ-SSP.

Proof:
It is easy to see that $(\bot_i, s)$ is τ-solvable for all $i \in \{0, \dots, m-1\}$ and all $s \in S(B_G) \setminus \{\bot_i\}$: one may choose $sup(\bot_i) = 1$, $sup(s) = 0$ if $s \neq \bot_i$, $sig(e) = \mathsf{swap}$ if $\xrightarrow{e} \bot_i$ or $\bot_i \xrightarrow{e}$, $sig(e) = \mathsf{nop}$ otherwise.

Similarly, the atom (s, s') where $s \in S(T'_i)$ and $s' \in S(B_G) \setminus S(T'_i)$ is τ-solvable for all $i \in \{0, \dots, m-1\}$: one may choose $sup(s) = 1$ if $s \in S(T_i)$ and 0 otherwise, $sig(e) = \textsf{swap}$ if $e = w_i$ and $\textsf{nop}$ otherwise.

Thus, it remains to argue that an atom (s, s') is also solvable if $s \neq s' \in S(T'_i)$, for all $i \in \{0, \dots, m-1\}$. By Lemma 3.8, it suffices to present corresponding regions for T'_i.

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed, and, for a start, let's consider the case where $v_{i_0} \in M$ and $v_{i_1} \notin M$. That is, $T'_i = t_{i,0} \xrightarrow{v'_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2} \xrightarrow{v_{i_0}} t_{i,2} \xrightarrow{v_{i_1}} t_{i,4}$. For all $\ell \in \{1, 2, 3\}$, let $R_\ell = (sup_\ell, sig_\ell)$ be a pair of mappings $sup_\ell : S(T'_i) \to \{0, 1\}$, $sig_\ell : E(T'_i) \to \{\textsf{nop}, \textsf{swap}\}$, (implicitly) defined by $sup_1(t_{i,0}) = 0$, $sig_1(v'_{i_0}) = sig_1(v_{i_0}) = \textsf{nop}$ and $sig_1(v_{i_1}) = \textsf{swap}$; and $sup_2(t_{i,0}) = 0$, $sig_2(v'_{i_0}) = sig_2(v_{i_1}) = \textsf{nop}$ and $sig_2(v_{i_0}) = \textsf{swap}$; and $sup_3(t_{i,0}) = 0$, $sig_3(v_{i_0}) = sig_3(v_{i_1}) = \textsf{nop}$ and $sig_3(v'_{i_0}) = \textsf{swap}$. The images of T'_i under R_1, R_2 and R_3 are as follows:

$$T_i'^{R_1} = 0 \xrightarrow{\textsf{nop}} 0 \xrightarrow{\textsf{swap}} 1 \xrightarrow{\textsf{nop}} 1 \xrightarrow{\textsf{swap}} 0 \qquad T_i'^{R_2} = 0 \xrightarrow{\textsf{nop}} 0 \xrightarrow{\textsf{nop}} 0 \xrightarrow{\textsf{swap}} 1 \xrightarrow{\textsf{nop}} 1$$

$$T_i'^{R_3} = 0 \xrightarrow{\textsf{swap}} 1 \xrightarrow{\textsf{nop}} 1 \xrightarrow{\textsf{nop}} 1 \xrightarrow{\textsf{nop}} 1$$

By Lemma 3.8, R_ℓ can be extended to a τ-region of B_G that preserves sup for all $\ell \in \{1, 2, 3\}$. Moreover, obviously, for every SSP atom (s, s') of T'_i, there is an $\ell \in \{1, 2, 3\}$ such that $sup_\ell(s) \neq sup_\ell(s')$. Thus, (s, s') is τ-solvable in B_G.

The arguments for the case $v_{i_0} \notin M$ and $v_{i_1} \in M$ is similar (the situation is symmetrical); the case $v_{i_0}, v_{i_1} \in M$ is simpler since no two events are the same.

By the arbitrariness of i, this proves the lemma. □

3.2. The proof of Theorem 3.3(2) when $\tau \cap \{\textsf{inp}, \textsf{out}\} \neq \emptyset$

Let $\tau = \{\textsf{nop}, \textsf{swap}\} \cup \omega$ be a type of nets such that $\omega \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}, \textsf{free}\}$ and $\omega \cap \{\textsf{inp}, \textsf{out}\} \neq \emptyset$, and let A_G and κ be defined as in Section 3.1.

Lemma 3.10. If there is an E'-label-splitting B_G of A_G such that $|E'| \leq \kappa$ that has the τ-ESSP, then G has a λ-VC.

Proof:
Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed and $\mathfrak{L}$ be the set of events of A_G that occur split in B_G. If $R = (sup, sig)$ is a τ-region of A_G, then $sup(t_{i,0}) = sup(t_{i,4})$ by Lemma 3.5. Thus, $\alpha_i = (v_{i_0}, t_{i,4})$ is not τ-solvable, since $sup(t_{i,0}) \xrightarrow{sig(v_{i_0})}$ implies $sup(t_{i,4}) \xrightarrow{sig(v_{i_0})}$. On the other hand, B_G has the τ-ESSP, implying the τ-solvability of α_i. This implies $\{v_{i_0}, v_{i_1}\} \cap \mathfrak{L} \neq \emptyset$. Since i was arbitrary, this is true for all $T_0, \dots, T_{m-1}$. Thus, just like for Lemma 3.6, we get that $\mathcal{S} = \mathfrak{L} \cap \mathfrak{U}$ defines a λ-VC of G. This proves the lemma. □

Conversely, let $\mathcal{S}$ be a λ-VC of G, and B_G be the E'-label-splitting of A_G as defined in Section 3.1. By the following lemma, B_G has an exact net realization:

Lemma 3.11. The TS B_G has the τ-SSP, and the τ-ESSP.

Proof:
By Lemma 3.9, the TS B_G has the τ-SSP. It remains to argue for the τ-ESSP. Without loss of generality, we assume that $\mathsf{inp} \in \tau$ and present τ-regions $R = (sup, sig)$ that only use nop, inp and swap. Indeed, if $\mathsf{inp} \notin \tau$, then $\mathsf{out} \in \tau$ and one gets corresponding (complement) regions $R' = (sup', sig')$ simply by $sup'(s) = 1 - sup(s)$, $sig'(e) = sig(e)$ if $sig(e) \in \{\mathsf{nop}, \mathsf{swap}\}$, and $sig'(e) = \mathsf{out}$ if $sig(e) = \mathsf{inp}$ for all $s \in S(B_G)$ and all $e \in E'$.

The general idea to solve an ESSA (e, s) is to choose $sup(s) = 0$, $sig(e) = \mathsf{inp}$, and $sig(e') \in \{\mathsf{nop}, \mathsf{swap}\}$ as needed, and $sup(s')$ accordingly for the other events e', and states s' of A_G. For instance, if $e \in W \cup \ominus$, one may choose $sup(s') = 1$ if $s' \xrightarrow{e}$ and $sup(s') = 0$ otherwise; $sig(e) = \mathsf{inp}$, $sig(e') = \mathsf{swap}$ when e' leads to or originates from the unique state with support 1, and nop otherwise: all the ESSAs (e, s) will then be solved.

For the other events, we proceed as follows. Let v be some node of the graph G.

If $v \notin \mathcal{S}$, it may only occur in at most three paths of B_G of the form $T'_i = t_{i,0} \xrightarrow{v} t_{i,1} \xrightarrow{v_i} t_{i,2} \xrightarrow{v} t_{i,3} \xrightarrow{v'_i} t_{i,4}$ or $T'_i = t_{i,0} \xrightarrow{v'_i} t_{i,1} \xrightarrow{v} t_{i,2} \xrightarrow{v_i} t_{i,3} \xrightarrow{v} t_{i,4}$, since the companion vertex v_i must be in $\mathcal{S}$; moreover, if v belongs to several edges of B_G, all the corresponding companions must be different. Also, v occurs twice, with a v_i in between. We may thus choose $sig(v) = \mathsf{inp}$, $sup(s) = 1$ and $sup(s') = 0$ if $s \xrightarrow{v} s'$, $sig(e) = \mathsf{swap}$ if $\xrightarrow{e} s$, in each path T'_i containing v. All the other events will have a signature nop. For any T'_j not containing v, hence only containing signatures nop and swap, the latter will introduce states with support 1 while we would like to have supports 0 to exclude v; if this occurs, we shall thus consider two choices for the support of $t_{j,0}$: 0 (region R_0) and 1 (region R_1, with $sig(w_j) = \mathsf{swap}$); then the corresponding supports for the various $t_{j,\ell}$ will be complementary too. In any case, we shall get regions separating v from all the needed states.

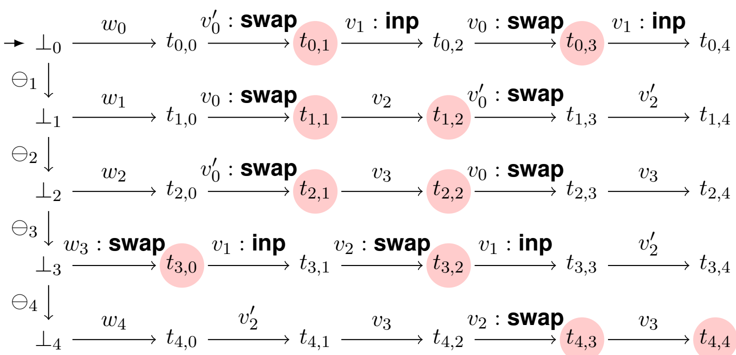

Figure 6: Region $R_0 = (sup, sig)$ for $v = v_1$. v occurs in T'_0 and T'_3. The supports of $t_{1,0}, t_{2,0}, t_{4,0}$ are chosen 0. The colored nodes have support 1, otherwise 0. When not indicated, signatures are nop.

If $v \in \mathcal{S}$, it may only occur in at most three paths of B_G of the form
$T'_i = t_{i,0} \xrightarrow{v'} t_{i,1} \xrightarrow{v_i} t_{i,2} \xrightarrow{v} t_{i,3} \xrightarrow{v_i} t_{i,4}$ or $T'_i = t_{i,0} \xrightarrow{v_i} t_{i,1} \xrightarrow{v} t_{i,2} \xrightarrow{v_i} t_{i,3} \xrightarrow{v'} t_{i,4}$,
or $T'_i = t_{i,0} \xrightarrow{v} t_{i,1} \xrightarrow{v_i} t_{i,2} \xrightarrow{v'} t_{i,3} \xrightarrow{v'_i} t_{i,4}$ or $T'_i = t_{i,0} \xrightarrow{v_i} t_{i,1} \xrightarrow{v} t_{i,2} \xrightarrow{v'_i} t_{i,3} \xrightarrow{v'} t_{i,4}$, depending on the

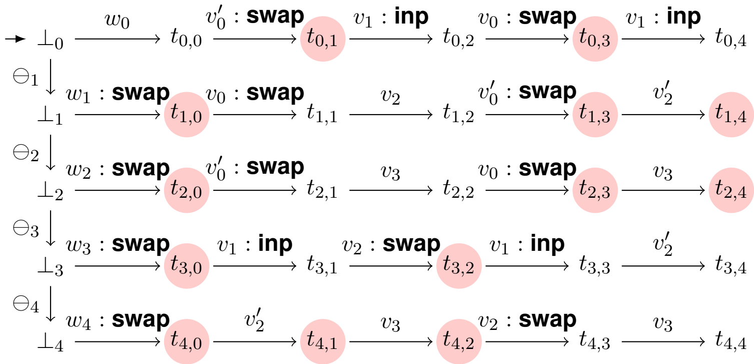

Figure 7: Region $R_1 = (sup, sig)$ for $v = v_1$. v occurs in T'_0 and T'_3. The supports of $t_{1,0}, t_{2,0}, t_{4,0}$ are chosen 1. The colored nodes have support 1, otherwise 0. When not indicated, signatures are nop.

fact that the companion vertex v_i belongs to $\mathcal{S}$ or not. Again, if v belongs to several edges, all the corresponding companions must be different. Also, v occurs only once, as well as v'. For v, we may thus choose $sig(v) = \mathsf{inp}$, $sup(s) = 1$ and $sup(s') = 0$ if $s \xrightarrow{v} s'$, $sig(e) = \mathsf{swap}$ if $\xrightarrow{e} s$, in each path T'_i containing v. All the other events will have a signature. For any T'_j not containing v, we may have some events with signature swap, hence some states with support 1 while we would like to only have supports 0 in order to exclude v; hence, if needed, we shall again consider two choices for the support of $t_{j,0}$: 0 (region R_2) and 1 (region R_3); then the corresponding supports for the various $t_{j,\ell}$ will be complementary too. In any case, we shall get regions separating v from all the needed states. However, for the first two kinds of configuration, since v is between two v_i's, after the second v_i (with signature swap), we shall have states with support 1 after v, while we would like to have supports 0 to exclude v; hence we will also use a region where w_i has signature swap and v_i has signature nop (region R_3). Note that, in this case, no event in any T'_j not containing v has a signature swap and all the states have support 0; hence we do not need here to introduce an additional region with $sig(w_j) = \mathsf{swap}$ to get complementary supports. In any case, we shall get regions separating v from all the needed states.

For v', we proceed similarly.

This proves the lemma. $\square$

3.3. The proof of Theorem 3.3(2), when $\tau \cap \{\mathsf{inp}, \mathsf{out}\} = \emptyset$

Since we already handled the case $\tau = \{\mathsf{nop}, \mathsf{swap}\}$, we may assume $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\emptyset \neq \omega \subseteq \{\mathsf{used}, \mathsf{free}\}$. Then, if we have to solve an ESSA (e, s), we have to try to find a τ-region where $sig(e) = \mathsf{used}$ and $\sup(s) = 0$, or $sig(e) = \mathsf{free}$ and $\sup(s) = 1$, but if $s' \xrightarrow{e} s \xrightarrow{\neg e}$, in either case $sup(s') = sup(s)$ and we may not solve (e, s).

Thus, there is no E'-label-splitting of the TS A_G of Section 3.1, that has the τ-ESSP. To overcome this obstacle, with as little effort as possible, we shall use transition systems $\overline{A}_G$ and $\overline{B}_G$ which extend

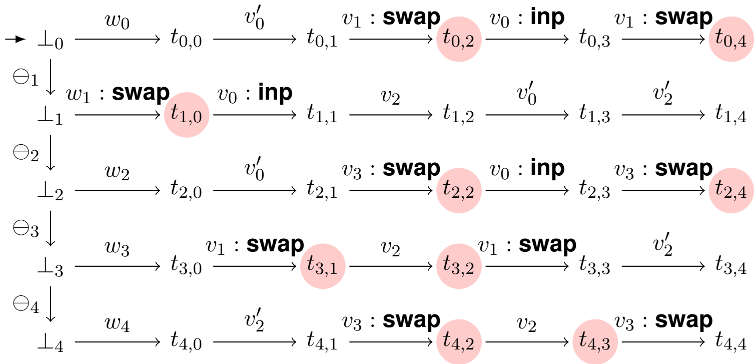

Figure 8: Region $R_2 = (sup, sig)$ for $v = v_0$. v occurs in T'_0, T'_1 and T'_2. The supports of $t_{3,0}, t_{4,0}$ are chosen 0. The colored nodes have support 1, otherwise 0. When not indicated, signatures are nop.

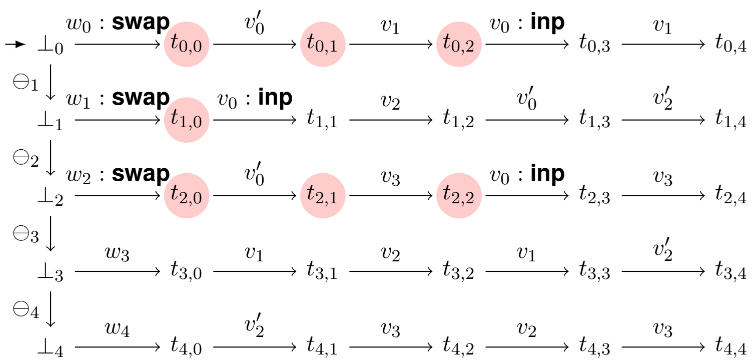

Figure 9: Region $R_3 = (sup, sig)$ for $v = v_0$. v occurs in T'_0, T'_1 and T'_2. The supports of $t_{3,0}, t_{4,0}$ are chosen 0. The colored nodes have support 1, otherwise 0. When not indicated, signatures are nop. Note that here no v_i with $i \neq 0$ needs to have a signature swap; hence we do not need to add a region with signature swap for w_j when T'_j does not contain v.

A_G and B_G by backward-edges (see Figure 10). Similarly, we shall denote by $\overline{T}_i$ the subsystem T_i with backward edges, for any i.

$$
\begin{tikzcd}
\bot_0 \arrow[r, leftrightarrow, "w_0"] \arrow[d, leftrightarrow, "\ominus_1"'] & t_{0,0} \arrow[r, leftrightarrow, "v_0"] & t_{0,1} \arrow[r, leftrightarrow, "v_1"] & t_{0,2} \arrow[r, leftrightarrow, "v_0"] & t_{0,3} \arrow[r, leftrightarrow, "v_1"] & t_{0,4} \\
\bot_1 \arrow[r, leftrightarrow, "w_1"] \arrow[d, leftrightarrow, "\ominus_2"'] & t_{1,0} \arrow[r, leftrightarrow, "v_0"] & t_{1,1} \arrow[r, leftrightarrow, "v_2"] & t_{1,2} \arrow[r, leftrightarrow, "v_0"] & t_{1,3} \arrow[r, leftrightarrow, "v_2"] & t_{1,4} \\
\bot_2 \arrow[r, leftrightarrow, "w_2"] \arrow[d, leftrightarrow, "\ominus_3"'] & t_{2,0} \arrow[r, leftrightarrow, "v_0"] & t_{2,1} \arrow[r, leftrightarrow, "v_3"] & t_{2,2} \arrow[r, leftrightarrow, "v_0"] & t_{2,3} \arrow[r, leftrightarrow, "v_3"] & t_{2,4} \\
\bot_3 \arrow[r, leftrightarrow, "w_3"] \arrow[d, leftrightarrow, "\ominus_4"'] & t_{3,0} \arrow[r, leftrightarrow, "v_1"] & t_{3,1} \arrow[r, leftrightarrow, "v_2"] & t_{3,2} \arrow[r, leftrightarrow, "v_1"] & t_{3,3} \arrow[r, leftrightarrow, "v_2"] & t_{3,4} \\
\bot_4 \arrow[r, leftrightarrow, "w_4"] & t_{4,0} \arrow[r, leftrightarrow, "v_2"] & t_{4,1} \arrow[r, leftrightarrow, "v_3"] & t_{4,2} \arrow[r, leftrightarrow, "v_2"] & t_{4,3} \arrow[r, leftrightarrow, "v_3"] & t_{4,4}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\bot_0 \arrow[r, leftrightarrow, "w_0"] \arrow[d, leftrightarrow, "\ominus_1"'] & t_{0,0} \arrow[r, leftrightarrow, "v_0'"] & t_{0,1} \arrow[r, leftrightarrow, "v_1"] & t_{0,2} \arrow[r, leftrightarrow, "v_0"] & t_{0,3} \arrow[r, leftrightarrow, "v_1"] & t_{0,4} \\
\bot_1 \arrow[r, leftrightarrow, "w_1"] \arrow[d, leftrightarrow, "\ominus_2"'] & t_{1,0} \arrow[r, leftrightarrow, "v_0"] & t_{1,1} \arrow[r, leftrightarrow, "v_2"] & t_{1,2} \arrow[r, leftrightarrow, "v_0'"] & t_{1,3} \arrow[r, leftrightarrow, "v_2'"] & t_{1,4} \\
\bot_2 \arrow[r, leftrightarrow, "w_2"] \arrow[d, leftrightarrow, "\ominus_3"'] & t_{2,0} \arrow[r, leftrightarrow, "v_0'"] & t_{2,1} \arrow[r, leftrightarrow, "v_3"] & t_{2,2} \arrow[r, leftrightarrow, "v_0"] & t_{2,3} \arrow[r, leftrightarrow, "v_3"] & t_{2,4} \\
\bot_3 \arrow[r, leftrightarrow, "w_3"] \arrow[d, leftrightarrow, "\ominus_4"'] & t_{3,0} \arrow[r, leftrightarrow, "v_1"] & t_{3,1} \arrow[r, leftrightarrow, "v_2"] & t_{3,2} \arrow[r, leftrightarrow, "v_1"] & t_{3,3} \arrow[r, leftrightarrow, "v_2'"] & t_{3,4} \\
\bot_4 \arrow[r, leftrightarrow, "w_4"] & t_{4,0} \arrow[r, leftrightarrow, "v_2'"] & t_{4,1} \arrow[r, leftrightarrow, "v_3"] & t_{4,2} \arrow[r, leftrightarrow, "v_2"] & t_{4,3} \arrow[r, leftrightarrow, "v_3"] & t_{4,4}
\end{tikzcd}
$$

Figure 10: The new transition systems $\overline{A}_G$ (top) and $\overline{B}_G$ (bottom) that originate from Example 3.4.

Lemma 3.12. If there is an E'-label-splitting of $\overline{A}_G$ such that $|E'| \leq \kappa$ that has the τ-ESSP, then G has a λ-VC.

Proof:
The proof is similar to the one of Lemma 3.10. □

Conversely, let $\mathcal{S}$ be a λ-VC of G, and let $\overline{B}_G = (S(B_G), E', \delta'', \bot_0)$ be the bi-directed extension of the TS $B_G = (S(B_G), E', \delta', \bot_0)$, which has been defined in Section 3.1. That is, for all $s, s' \in S(B_G)$ and all $e \in E'$ if $\delta'(s, e) = s'$, then $\delta''(s, e) = s'$ and $\delta''(s', e) = s$. To complete the proof of Theorem 3.3.2 it remains to argue that $\overline{B}_G$ has the τ-ESSP and the τ-SSP. Recall that the signatures of the regions that have been presented for the proof of Lemma 3.9 only use nop and swap. Thus, they can be directly applied to $\overline{B}_G$, which proves $\overline{B}_G$'s τ-SSP. Hence, it remains to argue for $\overline{B}_G$'s ESSP. The following lemma confirms both properties for $\overline{B}_G$ and thus completes the proof of Theorem 3.3.

Lemma 3.13. The TS $\overline{B}_G$ has the τ-SSP, and the τ-ESSP.

Proof:
First, we may observe that the signatures of the regions that have been presented for the proof of Lemma 3.9 only use $\textsf{nop}$ and $\textsf{swap}$. Thus, they can be directly applied to $\overline{B}_G$, which proves $\overline{B}_G$'s τ-SSP.

We now argue that $\overline{B}_G$ has the τ-ESSP. Without loss of generality, we assume $\textsf{used} \in \tau$. (Otherwise $\textsf{free} \in \tau$ and this case is similar.) We thus have to find, for each ESSA (e, s), a region $R = (sup, sig)$ such that $sig(e) = \textsf{used}$ and $sup(s) = 0$. The proof is very similar to the one for Lemma 3.11.

For instance, if $e \in W \cup \ominus$, one may choose $sup(s') = 1$ if $s' \xrightarrow{e}$ and $sup(s') = 0$ otherwise; $sig(e) = \textsf{used}$, $sig(e') = \textsf{swap}$ when e' leads to or originates from the unique state with support 1, and $\textsf{nop}$ otherwise: all the ESSAs (e, s) will then be solved. For the other events, we proceed as follows.

Let v be some node of the graph G.
If $v \notin \mathcal{S}$, it may only occur in at most three edges of G (in each case, the companion vertex must be in $\mathcal{S}$, and if v belongs to several edges of G, all the corresponding companions must be different), which leads to the two kinds of decorated paths in $\overline{B}_G$:

$$\bot_i : 0 \xleftrightarrow{w_i\,:\,\textsf{swap}} t_{i,0} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,1} : 1 \xleftrightarrow{v_i\,:\,\textsf{nop}} t_{i,2} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,3} : 1 \xleftrightarrow{v'_i\,:\,\textsf{swap}} t_{i,4} : 0$$

or $t_{i,0} : 0 \xleftrightarrow{v'_i\,:\,\textsf{swap}} t_{i,1} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,2} : 1 \xleftrightarrow{v_i\,:\,\textsf{nop}} t_{i,3} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,4} : 1.$

For any $\overline{T}_j$ not containing v, hence only containing signatures $\textsf{nop}$ and $\textsf{swap}$, this may introduce states with support 1 while we would like to have supports 0 to exclude v; if this occurs, we may consider two choices for the support of $t_{j,0}$: 0 (region R_0) and 1 (region R_1, with $sig(w_j) = \textsf{swap}$); then the corresponding supports for the various $t_{j,\ell}$ will be complementary too. In any case, we shall get regions separating v from all the needed states.

If $v \in \mathcal{S}$, this leads us to four kinds of decorated paths in $\overline{B}_G$, depending on the fact that the companion vertex v_i belongs to $\mathcal{S}$ or not, and if v is smaller than v_i or not:

$t_{i,0} : 0 \xleftrightarrow{v'\,:\,\textsf{nop}} t_{i,1} : 0 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,2} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,3} : 1 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,4} : 0$

or $t_{i,0} : 0 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,1} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,2} : 1 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,3} : 0 \xleftrightarrow{v'\,:\,\textsf{nop}} t_{i,4} : 0$

or $\bot_i : 0 \xleftrightarrow{w_i\,:\,\textsf{swap}} t_{i,0} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,1} : 1 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,2} : 0 \xleftrightarrow{v'\,:\,\textsf{nop}} t_{i,3} : 0 \xleftrightarrow{v'_i\,:\,\textsf{nop}} t_{i,4} : 0$

or $t_{i,0} : 0 \xleftrightarrow{v_i\,:\,\textsf{swap}} t_{i,1} : 1 \xleftrightarrow{v\,:\,\textsf{used}} t_{i,2} : 1 \xleftrightarrow{v'_i\,:\,\textsf{swap}} t_{i,3} : 0 \xleftrightarrow{v'\,:\,\textsf{nop}} t_{i,4} : 0.$

Again, if v belongs to several edges, all the corresponding companions must be different. Also, v occurs only once, as well as v'. For any $\overline{T}_j$ not containing v, we may have some events with signature $\textsf{swap}$ (the other ones having $\textsf{nop}$), hence some states with support 1 while we would like to only have supports 0 in order to exclude v; hence, if needed, we shall again consider two choices for the support of $t_{j,0}$: 0 (region R_2) and 1(region R_3); then the corresponding supports for the various $t_{j,\ell}$ will be complementary too. In any case, we shall get regions separating v from all the needed states.
For v', we proceed similarly.

Altogether, by the arbitrariness of v, we proved the τ-ESSP for $\overline{B}_G$ and this thus completes the proof of Lemma 3.13. $\square$

4. The complexity of edge-removal

In order to make a TS implementable, i.e., to satisfy SSP and/or ESSP, the removal of edges can also be an appropriate way of modification:

Definition 4.1. (Edge-Removal)
Let $A = (S, E, \delta, \iota)$ be a TS. A TS $B = (S, E, \delta', \iota)$ is an *edge-removal* of A if, for all $e \in E'$ and all $s, s' \in S'$, holds: if $s \xrightarrow{e} s' \in B$, then $s \xrightarrow{e} s' \in A$. By $\mathfrak{K} = \{s \xrightarrow{e} s' \in A \mid s \xrightarrow{e} s' \notin B\}$ we refer to the (set of) removed edges.

We would like to emphasize that B and A have the same set of states and events. Moreover, B is assumed to be a valid system, i.e., each state remains reachable from the initial one and each event occurs at least once in δ'.

τ-EDGE-REMOVAL FOR EMBEDDING	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an edge-removal B for A by $\mathfrak{K}$ that has the τ-SSP and satisfies $\lvert\mathfrak{K}\rvert \leq \kappa$?

τ-EDGE-REMOVAL FOR LANGUAGE-SIMULATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an edge-removal B for A by $\mathfrak{K}$ that has the τ-ESSP and satisfies $\lvert\mathfrak{K}\rvert \leq \kappa$?

τ-EDGE-REMOVAL FOR REALIZATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an edge-removal B for A by $\mathfrak{K}$ that has the τ-ESSP and the τ-SSP and satisfies $\lvert\mathfrak{K}\rvert \leq \kappa$?

The following theorem characterizes the complexity of the edge-removal problem for all implementations and types under consideration:

Theorem 4.2. If $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$, then

1. τ-EDGE-REMOVAL FOR EMBEDDING is NP-complete.
2. τ-EDGE-REMOVAL FOR LANGUAGE-SIMULATION and τ-EDGE-REMOVAL FOR REALIZATION are NP-complete if $\omega \neq \emptyset$, otherwise they are solvable in polynomial time.

First of all, we argue for the polynomial part: If $\tau = \{\mathsf{nop}, \mathsf{swap}\}$, then a TS $A = (S, E, \delta, \iota)$ has the τ-ESSP if and only if every event occurs at every state, since the functions nop,swap are defined on both 0 and 1. Thus, any ESSA (e, s) of A would be unsolvable. This implies that an edge-removal may neither render nor keep the τ-ESSP valid, since the removal of an edge would produce an unsolvable ESSA. Hence, the decision problems are polynomial, since either A is already implementable, which can be checked in polynomial time [21], or it has to be rejected. Thus, for the proof of Theorem 4.2, it remains to consider the NP-completeness results.

In order to prove Theorem 4.2, we present suitable reductions of 3BVC, where we reduce an input $G = (\mathfrak{U}, M)$ to an instance (A_G, κ), such that G has a λ-VC if and only if A_G allows an implementable edge-removal B_G that satisfies $|\mathfrak{K}| \leq \kappa$. However, due to their different ability to solve ESSAs, when it comes to language-simulation or realization, we have to distinguish again between the types τ that have at least one of inp or out, and the ones that do not have any of them.

4.1. The proof of Theorem 4.2(1), and the proof of Theorems 4.2(2) for the types with inp or out

In this section, we shall prove Theorem 4.2(1) for all types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and we prove Theorems 4.2(2) for the types that additionally satisfy $\omega \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$. We deal with these proofs simultaneously, since they use the same reduction. In particular, according to our general approach, we start from an input (G, λ) of 3BVC, and construct an instance TS (A_G, κ) as follows:

If $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, then A_G allows an edge-removal B_G that respects κ, and has the τ-SSP if and only if G has a λ-vertex-cover;

If $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and $\omega \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, then A_G allows an edge-removal B_G that respects κ, and has the τ-ESSP if and only if G has a λ-vertex-cover.

Hence, in the remainder of this section, let $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, whenever we deal with the τ-SSP, and let additionally $\omega \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, whenever we deal with the τ-ESSP (where the TS in question is clear from the context).

We now define the announced instance (A_G, κ). First of all, $\kappa = \lambda$. Moreover, for every $i \in \{0, \ldots, m-1\}$, the TS A_G has the following path T_i, that uses the vertices of $\mathfrak{e}_i = \{v_{i_0}, v_{i_1}\}$ (assuming $i_0 < i_1$) as events:

$$T_i = t_{i,0} \xrightarrow{v_{i_0}} t_{i,1} \xrightarrow{v_{i_1}} t_{i,2}$$

Furthermore, for every $i \in \{0, \ldots, n-1\}$, the TS A has the following gadget F_i that uses the node v_i as event, and, for all $j \in \{0, \ldots, \kappa\}$, has an a_j-labeled edge which directs in the same direction as the v_i-labeled edge:

$$F_i = \begin{tikzcd} f_{i,0} \arrow[rr, bend left=60, "a_\kappa"] \arrow[rr, bend left=30, "a_{\kappa-1}"] \arrow[rr, phantom, bend left=15, "\vdots"] \arrow[rr, "a_1"'] \arrow[rr, bend right=30, "a_0"'] \arrow[rr, bend right=60, "v_j"'] & & f_{i,1} \end{tikzcd}$$

The TS A_G has the initial state ι; for all $i \in \{0,\dots,m-1\}$, and all $j \in \{0,1,2\}$, it has the edge $\iota \xrightarrow{y_i^j} t_{i,j}$; finally, for all $\ell \in \{0,\dots,n-1\}$, it has the edge $\iota \xrightarrow{z_\ell} f_{\ell,0}$. The y_i^j-, and z_ℓ-labeled edges serve to ensure reachability. For the sake of simplicity, we summarize these events by $Y = \bigcup_{i=0}^{m-1}\{y_i^0, y_i^1, y_i^2\} \cup \{z_0,\dots,z_{n-1}\}$.

Lemma 4.3. If there is an edge-removal B_G of A_G that satisfies $|\mathfrak{K}| \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let B_G be an edge-removal of A_G that satisfies $|\mathfrak{K}| \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP, and let $\mathcal{S} = \mathfrak{U} \cap \{e \in E(A_G) \mid \exists s, s' \in S(A_G) : s \xrightarrow{e} s' \in \mathfrak{K}\}$ be the set of events of A_G that label an edge of A_G that is removed to obtain B_G. First of all, we note that $|\mathcal{S}| \leq |\mathfrak{K}| \leq \kappa = \lambda$. Moreover, in the following, we will argue that $\mathcal{S}$ defines a vertex cover of G.

Let $i \in \{0,\dots,m-1\}$ be arbitrary but fixed. We show that $\mathcal{S} \cap \{v_{i_0}, v_{i_1}\} \neq \emptyset$. If $t_{i,0} \xrightarrow{v_{i_0}} t_{i,1} \in \mathfrak{K}$ or $t_{i,1} \xrightarrow{v_{i_1}} t_{i,2} \in \mathfrak{K}$, then we are finished. Otherwise, since B_G has the τ-SSP, respectively the τ-ESSP, there is a τ-region $R = (sup, sig)$ that solves $(t_{i,0}, t_{i,2})$, respectively $(v_{i_0}, t_{i,2})$, and thus satisfies $sup(t_{i,0}) \neq sup(t_{i,2})$. This implies either $sup(t_{i,0}) = sup(t_{i,1})$, and thus $sig(v_{i_0}) \in \{\mathsf{nop}, \mathsf{free}, \mathsf{used}\}$, and $sig(v_{i_1}) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$, or $sup(t_{i,1}) = sup(t_{i,2})$, and thus $sig(v_{i_0}) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$, and $sig(v_{i_1}) \in \{\mathsf{nop}, \mathsf{free}, \mathsf{used}\}$.

We show that this implies $\mathfrak{K} \cap \{f_{i_0} \xrightarrow{v_{i_0}} f_{i_0,1}, f_{i_1} \xrightarrow{v_{i_1}} f_{i_1,1}\} \neq \emptyset$: Assume, for a contradiction, that the opposite is true. Since $|\mathfrak{K}| \leq \kappa$, there is a $j \in \{0,\dots,\kappa\}$, such that both $f_{i_0} \xrightarrow{a_j} f_{i_0,1}$, and $f_{i_1} \xrightarrow{a_j} f_{i_1,1}$ are present in B_G. If $sig(v_{i_0}) \in \{\mathsf{nop}, \mathsf{free}, \mathsf{used}\}$, and $sig(v_{i_1}) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$, we have $sup(f_{i_0,0}) = sup(f_{i_0,1})$, and $sup(f_{i_1,0}) \neq sup(f_{i_1,1})$, which simultaneously implies $sig(a_j) \in \{\mathsf{nop}, \mathsf{free}, \mathsf{used}\}$, and $sig(a_j) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$, which is a contradiction. Hence, we have $\mathfrak{K} \cap \{f_{i_0} \xrightarrow{v_{i_0}} f_{i_0,1}, f_{i_1} \xrightarrow{v_{i_1}} f_{i_1,1}\} \neq \emptyset$. Analogously, if $sig(v_{i_0}) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$, and $sig(v_{i_1}) \in \{\mathsf{nop}, \mathsf{free}, \mathsf{used}\}$, then we get $\mathfrak{K} \cap \{f_{i_0} \xrightarrow{v_{i_0}} f_{i_0,1}, f_{i_1} \xrightarrow{v_{i_1}} f_{i_1,1}\} \neq \emptyset$ as well.

By the arbitrariness of i, this implies that $\mathcal{S}$ is a vertex cover of G. □

For the converse direction, we have to show that the existence of a suitable vertex cover implies that A_G has an implementable edge-removal. So let $\mathfrak{S} = \{v_{\ell_0},\dots,v_{\ell_{\lambda-1}}\}$ be a vertex cover of G, and let B_G be the TS that originates from A_G be removing the edge $f_{\ell_i,0} \xrightarrow{v_{\ell_i}} f_{\ell_i,1}$ for all $i \in \{0,\dots,\lambda-1\}$, and nothing else. One easily verifies that B_G is a well-defined reachable edge-removal of A_G that satisfies $|\mathfrak{K}| \leq \kappa$.

In the following, we will show that B_G has the τ-SSP as well as the τ-ESSP, by presenting regions that altogether solve the individual separation atoms of B_G. Let $I = \{t_{0,0}, t_{1,0},\dots,t_{m-1,0}\} \cup \{f_{0,0}, f_{1,0},\dots,f_{n-1,0}\}$ be the set of the initial states of the gadgets of B_G, and $E = E(A_G) \setminus Y$ be the set of events in those gadgets (i.e., the vertices of G and the a_j's).

For the sake of simplicity, we often restrict the presentation of a region $R = (sup, sig)$ to the states of I and the events of E. This is justified, since we can easily extend R to B_G, when this is possible.

Indeed, choosing as we want $sup(\iota)$, for any $s \in I$ and $\iota \xrightarrow{e} s$ (then $e \in Y$ and is unique), we may choose $sig(e) = \mathsf{nop}$ if $sup(\iota) = sup(s)$ and $sig(e) = \mathsf{swap}$ otherwise; for any $i \in \{0, \dots, m-1\}$, from the support of $t_{i,0}$, the signatures of v_{i_0} and v_{i_1} determine the supports of $t_{i,1}$ and $t_{i,2}$ (if the functions of the signatures are defined for these supports); for any $i \in \{0, \dots, n-1\}$, if $f_{i,0} \xrightarrow{e} f_{i,1}$ in B_G, from the support of $f_{i,0}$, the signature of e determines the support of $f_{i,1}$ (if defined); however, if we also have $f_{i,0} \xrightarrow{e'} f_{i,1}$ with $e' \neq e$ in B_G, it is necessary that the signature of e' is "compatible" with the pair $sup(f_{i,0})$ and $sup(f_{i,1})$ (to go from 0 to 0, $sig(e')$ may be nop or free; to go from 0 to 1, $sig(e')$ may be swap or out; to go from 1 to 0, sig(e') may be swap or inp; to go from 1 to 1, $sig(e')$ may be nop or used).

In fact, the same is true if we choose as we want $sup(\iota)$ as well as $sup(s)$ when one chooses as we want some s in each T_i ($i \in \{0, \dots, m-1\}$) and each F_j ($j \in \{0, \dots, n-1\}$), and (coherently) signatures $sig(e)$ when $e \in E$. This is due to the fact that, for each partial function in τ, its inverse is also a partial function (this would not be true if we allowed set or res). Hence we may proceed backward as well as forward in the exploration of B_G.

Fact 4.4. The TS B_G has the τ-SSP.

Proof:
-Let $sup_0(\iota) = 0$, $sup_0(s) = 1$ when $s \in S(A_G) \setminus \{\iota\}$, $sig_0(e) = \mathsf{swap}$ for all $e \in Y$ and $sig_0(e) = \mathsf{nop}$ when $e \in E(A_G) \setminus Y$, then $R_0 = (sup_0, sig_0)$ is a region that solves ι.

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed.

- For all $s \in I$, let $sup_1(s) = 1$ if $s = t_{i,0}$, 0 otherwise, and let $sig_1(e) = \mathsf{nop}$ for all $e \in E$. This leads to a region $R_1 = (sup_1, sig_1)$ that solves (s, s') for all $s \in S(T_i)$, and all $s' \in S(A_G) \setminus S(T_i)$.

-If $sup_2(s) = 0$ for all $s \in I$ and $sig_2(e) = \mathsf{swap}$ for all $e \in E$, then $R_2 = (sup_2, sig_2)$ solves $(t_{i,0}, t_{i,1})$ and $(t_{i,1}, t_{i,2})$.

-If $sup_3(s) = 0$ for all $s \in I$ and, for all $e \in E$, if $e = v_{i_0}$ then $sig_3(e) = \mathsf{swap}$, nop otherwise, then $R_3 = (sup_3, sig_3)$ solves $(t_{i,0}, t_{i,2})$.

By the arbitrariness of i, this shows, for all $s \in \bigcup_{i=0}^{m-1} S(T_i)$, that s is solvable.

Similarly, one shows the solvability of each $s \in \bigcup_{i=0}^{n-1} S(F_i)$. The fact follows. □

In the rest of the subsection, we shall assume that $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$ with $\omega \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$.

Fact 4.5. The TS B_G has the τ-ESSP.

Proof:
We shall assume that $\mathsf{inp} \in \tau$ (the case where $\omega = \{\mathsf{out}\}$ is symmetrical).
-If $sup_0(\iota) = 1$, $sig_0(e) = \mathsf{inp}$ for all $e \in Y$ and $sig_0(e) = \mathsf{nop}$ for all $E(A_G) \setminus Y$, then $R_0 = (sup_0, sig_0)$ solves e for all $e \in Y$.

We proceed with the a_j's: Let $j \in \{0, \dots, \kappa\}$ be arbitrary but fixed.
-If $sup_1(\iota) = 0$, $sup_1(s) = 1$ for all $s \in I$, $sig_1(a_j) = \textsf{inp}$ and $sig_1(e) = \textsf{swap}$ if $e \in E(A_G) \setminus \{a_j\}$, then $R_1 = (sup_1, sig_1)$ solves (a_j, s) for all $s \in \bigcup_{i=0}^{n-1}\{f_{i,1}\} \cup \bigcup_{i=0}^{m-1}\{t_{i,1}\} \cup \{\iota\}$.

-If $sup_2(s) = 1$ for all $s \in \{f_{0,0}, \dots, f_{n-1,0}\}$, $sup_2(s) = 0$ for all $s \in \{t_{0,0}, \dots, t_{m-1,0}\}$, $sig_1(a_j) = \textsf{inp}$ and $sig_1(e) = \textsf{swap}$ if $e \in E \setminus \{a_j\}$, then $R_2 = (sup_2, sig_2)$ solves (a_j, s) for all $s \in \bigcup_{i=0}^{m-1}\{t_{i,0}, t_{i,2}\}$.
Since j was arbitrary, the solvability of the a_j's follows.

We proceed with the v_i's: Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed, and let L select the edges (at most three) that contain v_i, that is, $v_i \in \mathfrak{e}_\ell$ when $\ell \in L$.

We start with the case where $v_i \in \mathcal{S}$, implying that $f_{i,0}\xrightarrow{v_i}f_{i,1} \notin B_G$:

-For all $s \in I$, let $sup_3(s) = 1$ if $s \in \bigcup_{\ell \in L}\{t_{\ell,0}\}$ and 0 otherwise; for all $e \in E$, then $sig_3(e) = \textsf{inp}$ if $e = v_i$ and $\textsf{nop}$ otherwise. Then $R_3 = (sup_3, sig_3)$ solves (v_i, s) for all $s \in S \setminus \bigcup_{\ell \in L} S(T_\ell)$.

Let $\ell \in L$. The region R_3 also solves $(v_i, t_{\ell,2})$ and, if $t_{\ell,0}\xrightarrow{v_i}$, then it solves also $(v_i, t_{\ell,1})$. With respect to T_ℓ, it remains to address the case $t_{\ell,1}\xrightarrow{v_i}$, which requires to solve $(v_i, t_{\ell,0})$. The following region $R_4 = (sup_4, sig_4)$ does it: For all $s \in I$, if $s\xrightarrow{v_i}$ (implying that v_i is the "first" event in the corresponding edge of the graph), then $sup_4(s) = 1$, otherwise $sup_4(s) = 0$; for all $e \in E$, if $e = v_i$, then $sig_4(e) = \textsf{inp}$, otherwise $sig_4(e) = \textsf{swap}$ (note that, here, it is important that $f_{i,0}\xrightarrow{v_i}f_{i,1} \notin B_G$, otherwise we would lose the coherency in F_{v_i}). Since i and ℓ were arbitrary, this shows that solvability of vertex events that do belong to the vertex cover.

It remains to consider the case where $v_i \notin \mathcal{S}$, implying that $f_{i,0}\xrightarrow{v_i}f_{i,1} \in B_G$:

-Let $sup_5(s) = 1$ when $s \in I$ and, for all $e \in E$, $sig_5(e) = \textsf{inp}$ when $e \notin \mathcal{S}$, $\textsf{nop}$ otherwise. Then $R_5 = (sup_5, sig_5)$ solves (v_i, s) for all $s \in \bigcup_{i=0}^{n-1}\{f_{i,1}\}$. (Notice that never two events with an $\textsf{inp}$-signature occur one after the other, since $\mathcal{S}$ is a vertex cover.)

-If $sup_6(s) = 1$ for all $s \in \{f_{i,0}\} \cup \bigcup_{i=0}^{m-1}\{t_{i,0}\}$, $sup_6(s) = 0$ for all $s \in \bigcup_{j=0}^{n-1}\{f_{j,0}\} \setminus \{f_{i,0}\}$, and, for all $e \in E$, $sig_6(e) = \textsf{inp}$ if $e = v_i$, $sig_6(e) = \textsf{swap}$ if $e \in \{a_0, \dots, a_\kappa\} \cup \mathfrak{U} \setminus \mathcal{S}$, $sig_6(e) = \textsf{nop}$ otherwise, then $R_6 = (sup_6, sig_6)$ solves (v_i, s) for all $s \in \bigcup_{j=0}^{n-1}\{f_{j,0}\} \setminus \{f_{i,0}\}$.

In order to solve v_i within T_j when $j \notin L$, it suffices to define a region that is complementary to R_6 within the T_j's. The following region $R_7 = (sup_7, sig_7)$ accomplishes this: for all $s \in I$, if $s \in \{f_{i,0}\} \cup \bigcup_{l \in L}\{t_{l,0}\}$, then $sup_7(s) = 1$, otherwise $sup_7(s) = 0$; for all $e \in E$, $sig(e) = \textsf{inp}$ if $e = v_i$, $sig_7(e) = \textsf{swap}$ if $e \in \{a_0, \dots, a_\kappa\} \cup \mathfrak{U} \setminus \mathcal{S}$, otherwise $sig_7(e) = \textsf{nop}$.

It remains to solve v_i within T_j when $j \in L$. Let $j \in L$ be arbitrary but fixed. The ESSA $(v_i, t_{j,2})$ is solved by R_5, and if $t_{j,0}\xrightarrow{v_i}$, then R_5 solves $(v_i, t_{j,1})$ as well. Hence, it remains to consider the case $t_{j,1}\xrightarrow{v_i}$, which implies $v_{j_0} \in \mathcal{S}$, and requires to solve $(v_i, t_{j,0})$. The following region $R_8 = (sup_8, sig_8)$ accomplishes this: for all $s \in I$, if $s \in \{f_{i,0}\} \cup \bigcup_{l \in L}\{t_{l,0}\} \setminus \{t_{j,0}\}$, then $sup_8(s) = 1$, otherwise $sup_8(s) = 0$; for all $e \in E$, $sig_8(e) = \textsf{inp}$ if $e = v_i$, $sig_8(e) = \textsf{swap}$ if $e \in \{v_{j_0}\} \cup \{a_0, \dots, a_\kappa\} \cup \mathfrak{U} \setminus \mathcal{S}$, $sig_8(e) = \textsf{nop}$ otherwise.
Since i and j were arbitrary, this completes the solvability of the vertex events. The fact follows. □

Altogether, we get the following lemma:

Lemma 4.6. If there is a λ-VC for G, then there is an edge-removal B_G of A_G that satisfies $|\mathfrak{K}| \leq \kappa$, and has the τ-SSP as well as the τ-ESSP.

4.2. The proof of Theorem 4.2(2) for the types without inp and out

In this section, we assume that $\emptyset \neq \omega \subseteq \{\mathsf{free}, \mathsf{used}\}$, and more exactly $\mathsf{used} \in \omega$ (the case $\omega = \{\mathsf{free}\}$ may be handled symmetrically).

Note that, like in section 3.3, if A is a TS such that $s' \xrightarrow{e} s \xrightarrow{\neg e}$, then the ESSA (e, s) cannot be solved by a τ-region. The current reduction then extends the former one by simply adding the missing reverse edges. More exactly, we first define $\kappa = 2\lambda$. Then, for every $i \in \{0, \dots, m-1\}$, and for every $j \in \{0, \dots, n-1\}$, the new TS $\overline{A}_G$ has the following gadgets $\overline{T}_i$, and $\overline{F}_j$: (where $s \xrightarrow{e} s'$ implies $s' \xrightarrow{e} s$):

$$\overline{T}_i = t_{i,0} \xleftrightarrow{v_{i_0}} t_{i,1} \xleftrightarrow{v_{i_1}} t_{i,2} \qquad \overline{F}_j = \begin{tikzcd} f_{j,0} \arrow[r, leftrightarrow, bend left=60, "a_\lambda"] \arrow[r, leftrightarrow, bend left=30, "a_{\lambda-1}"] \arrow[r, phantom, bend left=10, "\vdots"] \arrow[r, leftrightarrow, "a_1"'] \arrow[r, leftrightarrow, bend right=30, "a_0"'] \arrow[r, leftrightarrow, bend right=60, "v_j"'] & f_{j,1} \end{tikzcd}$$

Again, the TS $\overline{A}_G$ has the initial state ι; for all $i \in \{0, \dots, m-1\}$ and all $j \in \{0, 1, 2\}$, it has the edges $\iota \xleftrightarrow{y_i^j} t_{i,j}$; finally, for all $\ell \in \{0, \dots, n-1\}$, it has the edges $\iota \xleftrightarrow{z_\ell} f_{\ell,0}$. The result is a TS $\overline{A}_G$, where the y_i^j-, and z_ℓ-labelled edges serve to ensure reachability even if we delete some v_j's. For the sake of simplicity, we summarize these events by $Y = \bigcup_{i=0}^{m-1} \{y_i^0, y_i^1, y_i^2\} \cup \{z_0, \dots, z_{n-1}\}$.

Lemma 4.7. If there is an edge-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{K}| \leq \kappa$, and has the τ-ESSP, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let $\overline{B}_G$ be an edge-removal of $\overline{A}_G$ that satisfies $|\mathfrak{K}| \leq \kappa$, and has the τ-ESSP, and let $\mathcal{S} = \{v \in \mathfrak{U} \mid s \xrightarrow{v} s' \in \mathfrak{K}\}$ select the vertex events that label an edge of $\overline{A}_G$ that is removed in $\overline{B}_G$. Note that $s \xrightarrow{e} s' \in \mathfrak{K}$ implies $s' \xrightarrow{e} s \in \mathfrak{K}$, since otherwise we would have an unsolvable ESSA (e, s') in $\overline{B}_G$, which is a contradiction. This particularly implies $|\mathcal{S}| \leq \frac{\kappa}{2} = \lambda$.

We argue that $\mathcal{S}$ defines a vertex cover for G:
Let $i \in \{0, \dots, m-1\}$ be arbitrary, but fixed. Similarly to the proof of Lemma 4.3 one argues that $\mathcal{S} \cap \{v_{i_0}, v_{i_1}\} = \emptyset$ implies a contradiction to the solvability of $(v_{i_0}, t_{i,2})$. Hence, by the arbitrariness of i, we have $\mathcal{S} \cap \mathfrak{e}_i \neq \emptyset$ for all $i \in \{0, \dots, m-1\}$. This proves the claim. $\square$

Lemma 4.8. If there a λ-VC for G, then there is an edge-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{K}| \leq \kappa$ and has both the τ-SSP and the τ-ESSP.

Proof:
Let $\mathcal{S} = \{v_{\ell_0}, \dots, v_{\ell_{\lambda-1}}\}$ be a suitable vertex cover, and let $\overline{B}_G$ be the TS that originates from $\overline{A}_G$ by the removal of the edges $f_{\ell_j,0} \xrightarrow{v_{\ell_j}} f_{\ell_j,1}$ and $f_{\ell_j,1} \xrightarrow{v_{\ell_j}} f_{\ell_j,0}$ for all $j \in \{0, \dots, \lambda-1\}$, and nothing else. Clearly, $\overline{B}_G$ is an edge-removal of $\overline{A}_G$ that satisfies $|\mathfrak{K}| = \kappa = 2\lambda$.

Note that every region presented for the proof of Lemma 4.4 can be directly transferred to the current edge-removal $\overline{B}_G$, since their signatures only use nop, and swap. Hence, $\overline{B}_G$ has the τ-SSP.

We argue for the τ-ESSP:

Let $I = \{t_{0,0}, t_{1,0}, \dots, t_{m-1,0}, f_{0,0}, f_{1,0}, f_{n-1,0}\}$ be the set of the initial states of the gadgets of $\overline{A}_G$ (when considered as TS).

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. In the following, we argue that y_i^0, and y_i^1, and y_i^2 are solvable at the necessary states of T_i. We start with y_i^0, and have to distinguish the case where v_{i_0} belongs to the vertex cover, and the case where it does not belong to the vertex cover, which implies that $\overline{F}_{i_0}$ is completely present in $\overline{B}_G$.

The following region $R_0 = (sup_0, sig_0)$ solves $(y_i^0, t_{i,1})$, and $(y_i^0, t_{i,2})$:

The case $v_{i_0} \in \mathcal{S}$ (implying that $f_{i_0,0} \xleftrightarrow{v_{i_0}} f_{i_0,1} \notin \overline{B}_G$): We let $sup(\iota) = sup(s) = 1$ for all $s \in I$, and, for all $e \in E(\overline{B}_G) \setminus (\bigcup_{i=0}^{m-1}\{y_i^1, y_i^2\})$, if $e = y_i^0$, then $sig(e) = \mathsf{used}$, if $e = v_{i_0}$, then $sig(e) = \mathsf{swap}$, and $sig(e) = \mathsf{nop}$ otherwise.

After that we compute the support completely: for every $j \in \{0, \dots, m-1\}$, and every $\ell \in \{0,1\}$, we let $sup(t_{j,\ell+1}) = 1 - sup(t_{j\ell})$ if $v_{j_\ell} = v_{i_0}$ (implying $sig(v_{j_\ell}) = \mathsf{swap}$), and otherwise $sup(t_{j,\ell+1}) = sup(t_{j\ell})$; for every $j \in \{0, \dots, n-1\}$, we let $sup(f_{j,1}) = 1$ (which is consistent, since $v_{i_0} \in \mathcal{S}$). Finally, we complete the signature: for all $e \in \bigcup_{i=0}^{m-1}\{y_i^1, y_i^2\}$, if $\xrightarrow{e} s \in \overline{B}_G$, and $sup(s) = 0$, then $sig(e) = \mathsf{swap}$, and $sig(e) = \mathsf{nop}$ otherwise.

The case $v_{i_0} \notin \mathcal{S}$ (implying that $f_{i_0,0} \xleftrightarrow{v_{i_0}} f_{i_0,1} \in \overline{B}_G$, and $v_{i_1} \in \mathcal{S}$, which implies $f_{i_1,0} \xleftrightarrow{v_{i_1}} f_{i_1,1} \notin \overline{B}_G$): We let $sup(\iota) = sup(s) = 1$ for all $s \in I$, and, for all $e \in E(\overline{B}_G) \setminus (\bigcup_{i=0}^{m-1}\{y_i^1, y_i^2\})$, if $e \in \{y_0^0, y_1^0, \dots, y_{m-1}^0\} \cup \{z_0, \dots, z_{n-1}\}$, then $sig(e) = \mathsf{used}$, if $e = v_{i_1}$, then $sig(e) = \mathsf{nop}$, and $sig(e) = \mathsf{swap}$ otherwise.

After that we compute the support completely: for every $j \in \{0, \dots, m-1\}$, and every $\ell \in \{0,1\}$, we let $sup(t_{j,\ell+1}) = 1 - sup(t_{j\ell})$ if $v_{j_\ell} = v_{i_0}$ (implying $sig(v_{j_\ell}) = \mathsf{swap}$), and otherwise $sup(t_{j,\ell+1}) = sup(t_{j,\ell})$; for every $j \in \{0, \dots, n-1\}$, we let $sup(f_{j,1}) = 0$. Finally, for all $e \in \bigcup_{i=0}^{m-1}\{y_i^1, y_i^2\}$, if $\xrightarrow{e} s \in \overline{B}_G$, and $sup(s) = 0$, then $sig(e) = \mathsf{swap}$, and $sig(e) = \mathsf{nop}$ otherwise.

It is easy to see, that we can define a region R that solves $(y_i^1, t_{i,0})$, and $(y_i^1, t_{i,2})$, respectively $(y_i^2, t_{i,0})$, and $(y_i^2, t_{i,1})$, in a similar way. We thus refrain from representing the tedious details, and consider y_i^0, y_i^1, y_i^2 solved with respect to the necessary states of $\overline{T}_i$.

-If $sup_1(\iota) = 1$ and, for all $e \in E(\overline{B}_G)$, if $e \in \{y_i^0, y_i^1, y_i^2\}$, then $sig_0(e) = \mathsf{used}$, if $e \in Y \setminus \{y_i^0, y_i^1, y_i^2\}$ then $sig_1(e) = \mathsf{swap}$, and $sig_1(e) = \mathsf{nop}$ otherwise, then the resulting region $R_1 = (sup_1, sig_1)$ solves (y_i^j, s) for all $s \in S(\overline{B}_G) \setminus (\{\iota\} \cup S(\overline{T}_i))$, and all $j \in \{0,1,2\}$.

By the arbitrariness of i, this proves the solvability of y for all $y \in \bigcup_{i=0}^{m-1}\{y_i^0, y_i^1, y_i^2\}$. Similarly, one argues that z is solvable for all $z \in \bigcup_{i=0}^{n-1}\{z_i\}$.

If $e \in \{a_0, \dots, a_\kappa\} \cup \mathfrak{U}$ and $s \in S(\overline{B}_G)$ belongs to a gadget that does not contain e, then it is easy to see that (e, s) is τ-solvable: One simply defines a region that maps exactly to 1 the states that belong to gadgets of $\overline{B}_G$ that contain e (and 0 to the others), maps e to used, choses a suitable signature of the reachability events from Y (either nop or swap), and maps all the other events e' with $e' \in E(\overline{B}_G) \setminus (\{e\} \cup Y)$ to nop. This particularly implies the solvability of a for all $a \in \{a_0, \dots, a_\lambda\}$. Hence, it remains to address the vertex events.

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. By the former discussion, if $f_{i_0,0} \xleftrightarrow{v_{i_0}} f_{i_0,1} \in \mathfrak{K}$ then v_{i_0} is solvable in $\overline{F}_{i_0}$, otherwise there is no need to solve it in $\overline{F}_{i_0}$. Similarly, the solvability of v_{i_1} in $\overline{F}_{i_1}$ needs no further discussion. Hence, it remains to argue for the solvability of v_{i_0} in $\overline{T}_i$ (the case of v_{i_1} will be similar). Let us first assume that $v_{i_0} \in \mathcal{S}$ (so that it does not occur in $\overline{F}_{i_0}$): The following region $R_2 = (sup_2, sig_2)$ solves $(v_{i_0}, t_{i,2})$ (and thus v_{i_0} in $\overline{T}_i$): $sup_2(\iota) = 0$; and for any $j \in \{0, \dots, m-1\}$,

if $t_{j,0} \xrightarrow{v_{i_0}} t_{j,1} \in \overline{T}_j$, then $sup_2(t_{j,0}) = sup_2(t_{j,1}) = 1$, and $sup_2(t_{j,2}) = 0$ else

if $t_{j,1} \xrightarrow{v_{i_0}} t_{j,2} \in \overline{T}_j$, then $sup_2(t_{j,0}) = 0$ and $sup_2(t_{j,1}) = sup_2(t_{j,2}) = 1$, otherwise $sup_2(t_{j,0}) = sup_2(t_{j,2}) = 0$ and $sup_2(t_{j,1}) = 1$; and, for every $j \in \{0, \dots, n-1\}$, $sup_2(f_{j,0}) = 0$ and $sup_2(f_{j,1}) = 1$; moreover, for all $e \in E(\overline{B}_G) \setminus Y$, if $e = v_{i_0}$ then $sig_2(e) = \mathsf{used}$, otherwise $sig_2(e) = \mathsf{swap}$; finally, for all $e \in Y$, if $\iota \xrightarrow{e} s$ and $sup(s) = 1$, then $sig_2(e) = \mathsf{swap}$, otherwise $sig_2(e) = \mathsf{nop}$.

Let us now assume that $v_{i_0} \notin \mathcal{S}$, so that $v_{i_1} \in \mathcal{S}$, since we have a vertex covering (then v_{i_0} occurs in $\overline{F}_{i_0}$, but the vertex cover events do not occur in the $\overline{F}$-gadgets). For all $e \in E(\overline{B}_G) \setminus Y$, if $e = v_{i_0}$ then $sig_3(e) = \mathsf{used}$, if $e \in \mathcal{S}$, then $sig_3(e) = \mathsf{swap}$, otherwise $sig_3(e) = \mathsf{nop}$; moreover, $sup(\iota) = 1$, and for any $j \in \{0, \dots, n-1\}$, $sup_3(f_{j,0}) = 1$, and for any $j \in \{0, \dots, m-1\}$, if $v_{j_0} = v_{i_0}$, then $sup_3(t_{j,0}) = 1$, otherwise $sup_3(t_{j,0}) = 0$; the other supports and signatures may then be derived coherently, delivering a region $R_3 = (sup_3, sig_3)$ which, as expected, solves $(v_{i_0}, t_{i,2})$.

Since i was arbitrary, this completes the proof. □

5. The complexity of event-removal

In this section, we deal with the following modification:

Definition 5.1. (Event-Removal)
Let $A = (S, E, \delta, \iota)$ be an TS. A TS $B = (S, E', \delta', \iota)$ with $E' \subseteq E$ is an *event-removal* of A if for all $e \in E'$ the following is true: $s \xrightarrow{e} s' \in B$ if and only if $s \xrightarrow{e} s' \in A$ for all $s, s' \in S$. By $\mathfrak{E} = E \setminus E'$ we refer to the (set of) removed events.

We want to stress that an event-removal B is an initialized TS by definition and Remark 2.7 systems, i.e., each state remains reachable from the initial one, and each remaining event occurs at least once in δ' since this was already the case in A.

τ-EVENT-REMOVAL FOR EMBEDDING	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an event-removal B for A by $\mathfrak{E}$ that has the τ-SSP and satisfies $\lvert\mathfrak{E}\rvert \leq \kappa$?

τ-EVENT-REMOVAL FOR LANGUAGE-SIMULATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an event-removal B for A by $\mathfrak{E}$ that has the τ-ESSP and satisfies $\lvert\mathfrak{E}\rvert \leq \kappa$?

τ-EVENT-REMOVAL FOR REALIZATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist an event-removal B for A by $\mathfrak{E}$ that has the τ-ESSP and the τ-SSP and satisfies $\lvert\mathfrak{E}\rvert \leq \kappa$?

The following theorem is the main result of this section:

Theorem 5.2. If $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$, then

1. τ-EVENT-REMOVAL FOR EMBEDDING is NP-complete.
2. τ-EVENT-REMOVAL FOR LANGUAGE-SIMULATION, and τ-EVENT-REMOVAL FOR REALIZATION are NP-complete if $\omega \neq \emptyset$.

5.1. The proof of Theorem 5.2(1), and the proof of Theorem 5.2(2) for the types with inp or out

Let $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and for the input (G, λ) of 3BVC, where $G = (\mathfrak{U}, M)$, let (A_G, κ) be defined as in Section 4.1. In this section, we show that G has a λ-VC if and only if A_G allows an event-removal B_G that respects κ, and has the τ-SSP. Moreover, we also show that if $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, then G has a λ-VC if and only if A_G allows an event-removal B_G that respects κ, and has the τ-ESSP. Altogether, this proves Theorem 5.2(1), and the statements of Theorem 5.2(2) for the types $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$.

For the sake of simplicity, in the remainder of this section, if we discuss aspects of the τ-ESSP (where the addressed TS is clear from the context), we always assume that $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$.

Lemma 5.3. If there is an event-removal B_G of A_G that satisfies $\lvert\mathfrak{E}\rvert \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let B_G be an event-removal of A_G that satisfies $\lvert\mathfrak{E}\rvert \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP, and let $\mathcal{S} = \mathfrak{E} \cap \mathfrak{U}$ be the set vertex-events that are removed from A_G.

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed.
Since $|\mathfrak{E}| \leq \kappa$, there is a $j \in \{0, \dots, \kappa\}$ such that the event a_j, and thus all a_j-labeled edges are present in B_G. Hence, similar to the arguments for the proof of Lemma 4.3, one argues that if $\{v_{i_0}, v_{i_1}\} \cap \mathfrak{E} = \emptyset$, implying that all v_{i_0}-labeled, and v_{i_1}-labeled edges are present in B, then $(t_{i,0}, t_{i,2})$ is an unsolvable SSA of B, respectively $(v_{i_0}, t_{i,2})$ is an unsolvable ESSA of B, which is a contradiction. By the arbitrariness of i, and since $|\mathcal{S}| \leq |\mathfrak{E}| \leq \kappa = \lambda$, the claim follows. □

Reversely, if G has a λ-VC, then A_G allows a suitable event-removal B_G that has the τ-SSP, respectively the τ-ESSP:

Lemma 5.4. If there is a λ-VC for $G = (\mathfrak{U}, M)$, then there is an event-removal B_G of A_G that satisfies $|\mathfrak{E}| \leq \kappa$, has the τ-SSP, and that has τ-ESSP if $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$.

Proof:
Let $\mathcal{S} = \{v_{\ell_0}, \dots, v_{\ell_{\lambda-1}}\}$ be a vertex cover for G, and let B_G be the TS that originates from A_G by the removal the events of $\mathcal{S}$ (and thus the edges labeled by these events), and nothing else. By the events of Y, B_G is a reachable TS, which has the same states as A_G, and obviously satisfies $|\mathfrak{E}| \leq \kappa = \lambda$.

In the following, we argue that the solvability of B_G follows from the regions presented for the proofs of Fact 4.4, and Fact 4.5 of Section 4.1:

Let B'_G be the TS that originates from A_G by removing, for all $i \in \{0, \dots, \lambda-1\}$, the edge $f_{\ell_i,0} \xrightarrow{v_{\ell_i}} f_{\ell_i,1}$, and nothing else (that is, B'_G corresponds to the edge-removal defined in Section 4.1). By definition, both B_G, and B'_G have the same states as A_G, and thus have the same SSA to solve. Moreover, if $s \xrightarrow{e} s'$ is an edge of B_G, then it is an edge of B'_G, since, for all $i \in \{0, \dots, \lambda-1\}$, B_G does not only miss the edge $f_{\ell_i,0} \xrightarrow{v_{\ell_i}} f_{\ell_i,1}$, but all v_{ℓ_i}-labeled edges (and nothing else). In particular, for every event $e \in E(B_G)$, which implies $e \notin \mathcal{S}$, all the (original) e-labeled edges (of A_G) are present in both B_G, and B'_G. Hence, if (e, s) is an ESSA of B_G, then it is an ESSA of B'_G. Finally, if $R = (sup, sig)$ is a region of B'_G, then its restriction to (the events of) B_G is also a region of B_G, since $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$ is implied for all $s \xrightarrow{e} s' \in B_G$, by $s \xrightarrow{e} s' \in B'_G$. Moreover, if R solves a (state or event) separation atom α of B'_G that is also present in B_G, then it also solves this atom in B_G. Consequently, since B'_G has the τ-SSP, and the τ-ESSP by the arguments of Fact 4.4, and Fact 4.5, respectively, we conclude that B_G has also these properties. □

5.2. The proof of Theorem 5.2(2) for the types without inp and out

Let $\emptyset \neq \omega \subseteq \{\mathsf{free}, \mathsf{used}\}$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$. In order to complete the proof of Theorem 5.2(2) (for τ), we reduce the input $G = (\mathfrak{U}, M)$ to the instance $(\overline{A}_G, \kappa)$, where $\overline{A}_G$ is the TS (with bi-directional edges) as defined in Section 4.2, and $\kappa = \lambda$, and show that G has a λ-VC if and only if $\overline{A}_G$ allows an event-removal $\overline{B}_G$ that respects κ, and has the τ-ESSP, respectively the τ-ESSP, and the τ-SSP. We would like to emphasize that we define $\kappa = \lambda$ here, in contrast to the definition of $\kappa = 2\lambda$ in section 4.2. This (current) definition is justified by the fact that the removal of an event already implies the removal of all its edges. For details see the following arguments.

Lemma 5.5. If there is an event-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{E}| \leq \kappa$, and has the τ-ESSP, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let $\overline{B}_G$ be an event-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{E}| \leq \kappa$, and has the τ-ESSP; let $\mathcal{S} = \mathfrak{U} \cap \mathfrak{E}$. Note that $|\mathcal{S}| \leq |\mathfrak{E}| \leq \kappa = \lambda$.

We argue that $\mathcal{S}$ is a vertex cover:

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. If $\{v_{i_0}, v_{i_1}\} \cap \mathcal{S} = \emptyset$, then $\overline{T}_i$, $\overline{F}_{i_0}$ and $\overline{F}_{i_1}$ are completely present in $\overline{B}_G$. Similar to the proof of Lemma 4.7 one argues that this contradicts the τ-ESSP of $\overline{B}_G$. Hence $\{v_{i_0}, v_{i_1}\} \cap \mathcal{S} \neq \emptyset$, and the arbitrariness of i implies the claim. □

Lemma 5.6. If there is a λ-VC for $G = (\mathfrak{U}, M)$, then there is an event-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{E}| \leq \kappa$ and has the τ-ESSP as well as the τ-SSP.

Proof:
Let $\mathcal{S} = \{v_{\ell_0}, \dots, v_{\ell_{\lambda-1}}\}$ be a suitable vertex cover of G, and let $\overline{B}_G$ be the event-removal that originates from $\overline{A}_G$ by removing the events of $\mathcal{S}$ (and the corresponding edges), and nothing else. One easily checks that $\overline{B}_G$ is a (well-defined) reachable TS, and satisfies $|\mathfrak{E}| \leq |\mathcal{S}|$.

In the following, we argue that the τ-SSP, and the τ-ESSP of $\overline{B}_G$ is implied by the regions presented for the proof of Lemma 4.8: Let $\overline{B}'_G$ be the TS that originates from $\overline{A}_G$ by the removal of the edges $f_{\ell_j,0} \xrightarrow{v_{\ell_j}} f_{\ell_j,1}$, and $f_{\ell_j,1} \xrightarrow{v_{\ell_j}} f_{\ell_j,0}$, and nothing else (that is, $\overline{B}'_G$ corresponds to the edge-removal of Section 4.2). One verifies that $\overline{B}_G$, and $\overline{B}'_G$ have the same states, and thus the same SSAs to solve. Moreover, if $s \xrightarrow{e} s'$ is an edge of $\overline{B}_G$, implying that $e \notin \mathcal{S}$, then $s \xrightarrow{e} s' \in \overline{B}'_G$. Hence, every ESSA of $\overline{B}_G$ is an ESSA of $\overline{B}'_G$. Moreover, for every region $R = (sup, sig)$ of $\overline{B}'_G$, the restriction of R to (the events of) $\overline{B}_G$ is a region of $\overline{B}_G$, since $sup(s) \xrightarrow{e} sup(s')$ is implied for every $s \xrightarrow{e} s'$ of $\overline{B}_G$ (by $s \xrightarrow{e} s' \in \overline{B}'_G$, and the region property of R). Hence, since the regions of Lemma 4.8 justify the τ-SSP, and the τ-ESSP, they particularly justify these properties for $\overline{B}_G$. □

6. The complexity of state-removal

In this section, we address the following modification:

Definition 6.1. (State-Removal)
Let $A = (S, E, \delta, \iota)$ be a TS. A TS $B = (S', E, \delta', \iota)$ with states $S' \subseteq S$ is a *state-removal* of A if the following two conditions are satisfied:

(1) for all $e \in E$ and all $s, s' \in S'$, $s \xrightarrow{e} s' \in B$ if and only if $s \xrightarrow{e} s' \in A$;

(2) if $s \xrightarrow{e} s' \in A$ and $s \xrightarrow{e} s' \notin B$, then $s \notin S'$ or $s' \notin S'$ (or both).

By $\mathfrak{S} = S \setminus S'$ we refer to the (set of) removed states.

In the following, we shall assume that B is a valid system, i.e., each state remains reachable from the initial one and each event occurs at least once in δ'.

In particular, we investigate the complexity of the following three decision problems:

τ-STATE-REMOVAL FOR EMBEDDING	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist a state-removal B for A by $\mathfrak{S}$ that has the τ-SSP and satisfies $\lvert\mathfrak{S}\rvert \leq \kappa$?

τ-STATE-REMOVAL FOR LANGUAGE-SIMULATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist a state-removal B for A by $\mathfrak{S}$ that has the τ-ESSP and satisfies $\lvert\mathfrak{S}\rvert \leq \kappa$?

τ-STATE-REMOVAL FOR REALIZATION	
Input:	A TS $A = (S, E, \delta, \iota)$, a natural number κ.
Question:	Does there exist a state-removal B for A by $\mathfrak{S}$ that has the τ-ESSP and the τ-SSP and satisfies $\lvert\mathfrak{S}\rvert \leq \kappa$?

The following theorem is the main result of this section:

Theorem 6.2. If $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$, then

1. τ-STATE-REMOVAL FOR EMBEDDING is NP-complete.
2. τ-STATE-REMOVAL FOR LANGUAGE-SIMULATION, and
τ-STATE-REMOVAL FOR REALIZATION are NP-complete if $\omega \neq \emptyset$, otherwise they are solvable in polynomial time.

First of all, we argue for the polynomial part: If $\tau = \{\mathsf{nop}, \mathsf{swap}\}$, then a TS $A = (S, E, \delta, \iota)$ has the τ-ESSP if and only if every event occurs at every state, since the functions nop,swap are defined on both 0 and 1. Thus, any ESSP atom (e, s) of A would be unsolvable. Again, we may determine the states $s \in S$ such that, for some event e, we have $\neg s \xrightarrow{e}$: they must be removed. Let $B = (S', E, \delta, \iota)$ be the (unique) result of this phase. For the language-simulation problem, this is enough and we simply have to check if B is valid and if the number k of removed states does not exceed κ. Let (s, s') be an unsolvable SSA of B. Since B is an initialized TS, there is a path $\iota \xrightarrow{e_1} s_1 \dots s_{n-1} \xrightarrow{e_n} s_n$ such that $s_n = s$ in B. If we remove s_n (and get, say, B'), then $\neg s_{n-1} \xrightarrow{e_n}$, since nop and swap are functions, and thus we have an unsolvable ESSA (s_{n_1}, e_n). Consequently, we have also to remove s_{n-1}, and get, say, B''. By the same arguments, we inductively obtain that $s_{n-1}, \dots, s_1$, and finally ι, have to be removed, which is a contradiction, since every state-removal of A has the initial state ι by Definition 6.1. Similarly, the removal of s' yields also a contradiction.

Thus, for the proof of Theorem 6.2, it remains to consider the NP-completeness results.

6.1. The proof of Theorem 6.2(1), and the proof of Theorem 6.2(2) for the types with inp or out

Let $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$ and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ be such that $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$.

In order to prove Theorem 6.2(1), and the statement of Theorem 6.2(2) for $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, we use the following reduction that transforms an input (G, λ), where $G = (\mathfrak{U}, M)$, to an instance (A_G, κ): $\kappa = \lambda$, and A_G is the TS that is defined just like the one in Section 4.1, but, for all $i \in \{0, \dots, m-1\}$, and for all $j \in \{1, 2\}$, it does not implement the edges $\iota \xrightarrow{y_i^j} t_{i,j}$. (The edge $\iota \xrightarrow{y_i^0} t_{i,0}$ is still present to ensure reachability.) By doing so, we ensure that if B_G is well-defined state-removal of A_G, then, for all $i \in \{0, \dots, m-1\}$, if $t_{i,1}$ is removed, then $t_{i,2}$ is also removed, and, moreover, if $t_{i,0}$ is removed, then also both $t_{i,1}$ and $t_{i,2}$ are removed. This can be seen as follows: By Definition 6.1, B_G is a TS, and thus is initialized; hence the initial state ι is present and its states are reachable from ι by a directed path. Hence, if, for example, the state $t_{0,1}$ is missing in B_G (compared with A_G), implying that the (only) edge $t_{0,1} \xrightarrow{v_{0_1}} t_{0,2}$ is removed too, then the state $t_{0,2}$ is also not present in B_G, since it would not be reachable by a directed path from ι otherwise, which is a contradiction.

Lemma 6.3. If there is a state-removal B_G of A_G that satisfies $|\mathfrak{S}| \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP with $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let B_G be a state-removal of A_G that satisfies $|\mathfrak{S}| \leq \kappa$, and has the τ-SSP, respectively the τ-ESSP. Let $\mathcal{S} = \{v \in \mathfrak{U} \mid \exists s \in \mathfrak{S} : \xrightarrow{v} s\}$. Note that $|\mathcal{S}| \leq |\mathfrak{S}|$, since every state of A is the target of at most one vertex event. We show that $\mathcal{S}$ defines a vertex cover: Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. Recall that, as argued above, ι is present in B_G, and, for every path $s_0 \xrightarrow{e_1} s_1 \dots \xrightarrow{e_n} s_n$ of A_G, if s_i is missing in B_G, then s_j is also missing for all $i < j \in \{0, \dots, n\}$, since B_G would have unreachable states otherwise. Hence, if $\mathcal{S} \cap \{v_{i_0}, v_{i_1}\} = \emptyset$, implying $\{t_{i,1}, t_{i,2}, f_{i_0,1}, f_{i_1,1}\} \subseteq S(B_G)$, then T_i, and F_{i_0}, and F_{i_1} are completely present in B_G. Consequently, similar to the proof of Lemma 4.3, one argues that this implies that B_G has the unsolvable SSA $(t_{i,0}, t_{i,2})$, respectively the unsolvable ESSA $(v_{i_0}, t_{i,2})$, which is a contradiction. By the arbitrariness of i, $\mathcal{S}$ is a suitable vertex cover, which proves the lemma. $\square$

For the converse direction, we state the following lemma:

Lemma 6.4. If there is a λ-VC for $G = (\mathfrak{U}, M)$, then there is a state-removal B_G of A_G that satisfies $|\mathfrak{S}| \leq \kappa$ and has the τ-SSP, and, since $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$, it has the τ-ESSP.

Proof:
Let $\mathcal{S} = \{v_{\ell_0}, \dots, v_{\ell_{\lambda-1}}\}$ be a vertex cover for G, and let B_G the TS that originates from A_G by removing the states $f_{\ell_0,1}, f_{\ell_1,1}, \dots, f_{\ell_{\lambda-1},1}$ (and thus the edges incident to these states), and nothing else. Notice that, for all $j \in \{0, \dots, \lambda-1\}$, this reduces F_{ℓ_j} to $f_{\ell_j,0}$, since, for all $a \in \{a_0, \dots, a_{\lambda-1}\}$, the edge $f_{\ell_j,0} \xrightarrow{a} f_{\ell_j,1}$ is removed by the removal of $f_{\ell_j,1}$. Obviously, $|\mathfrak{S}| \leq \kappa = \lambda$.

Let B'_G be the TS that result by the removal of the edge $f_{\ell_j,0}\xrightarrow{v_{\ell_j}}f_{\ell_j,1}$ for all $j \in \{0,\dots,\lambda-1\}$, that is, B'_G corresponds to the edge-removal defined in Section 4.1. Obviously, every edge of B_G is present in B'_G. Hence, every region $R = (sup, sig)$ of B'_G is a region of B_G, when restricted to its present states, and events. Furthermore, if (s, s') is an SSA of B_G, then it is also an SSA of B'_G, and solved by a corresponding region of Fact 4.4.

If (e, s) is an ESSA of B_G such that $e \notin \{a_0,\dots,a_\lambda\}$, or $s \notin \{f_{\ell_0},\dots,f_{\ell_{\lambda-1}}\}$, then (e, s) is also an ESSA of B'_G, and solved by a corresponding region of Fact 4.5.

Hence, it only remains to argue that an ESSA (e, s) is also solvable, if $e \in \{a_0,\dots,a_\lambda\}$ and $s \in \{f_{\ell_0},\dots,f_{\ell_{\lambda-1}}\}$. The following region $R = (sup, sig)$ accomplishes this: $sup(\iota) = 0$, and, for all $e \in E(B_G)$, if $e \in \{a_0,\dots,a_\lambda\}$, then $sig(e) = \mathsf{inp}$, and if $e \in \{z_{\ell_0},\dots,z_{\ell\lambda-1}\}$, then $sig(e) = \mathsf{nop}$, otherwise $sig(e) = \mathsf{swap}$. Notice that this definition implies $sup(f_{\ell_j,0}) = 0$ for all $j \in \{0,\dots,\lambda-1\}$ (by $sig(z_{\ell_j}) = \mathsf{nop}$), and $sup(f_{i,1}) = sup(f_{i,0}) - 1 = 0$ for all $i \in \{0,\dots,n-1\} \setminus \{\ell_0,\dots,\ell_{\lambda-1}\}$ (by $sig(z_i) = sig(v_i) = \mathsf{swap}$, respectively $sig(a_0) = \cdots = sig(a_\lambda) = \mathsf{inp}$) such that R is actually a well-defined region that solves the remaining ESSA. This proves the lemma. $\square$

6.2. Proof of Theorem 6.2(2) for the Types without inp and out

Let $\emptyset \neq \omega \subseteq \{\mathsf{free}, \mathsf{used}\}$, and $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$.

The following reduction transforms an input (G, λ), where $G = (\mathfrak{U}, M)$, to an instance $(\overline{A}_G, \kappa)$: $\kappa = \lambda$, and $\overline{A}_G$ is the TS that is defined just like the one in Section 4.2, but, for all $i \in \{0,\dots,m-1\}$, and for all $j \in \{1, 2\}$, does neither implement the edge $\iota\xrightarrow{y_i^j}t_{i,j}$ nor the edge $\iota\xrightarrow{y_i^j}t_{i,j}$. Hence, for every state-removal $\overline{B}_G$ of $\overline{A}_G$, if there is $i \in \{0,\dots,m-1\}$, such that $t_{i,1}$ is missing in $\overline{B}_G$ (implying that the edge $t_{i,1}\xrightarrow{v_{i_1}}t_{i,2}$ is also removed), then $t_{i,2}$ is also removed, since this state would not be reachable from the initial state ι otherwise, which is a contradiction. Similarly, if $t_{i,0}$ is removed, then so are $t_{i,1}$, and $t_{i,2}$.

In the following, we argue that $(\overline{A}_G, \kappa)$ is a yes-instance (i.e., it allows a fitting state-removal that has the ESSP, and, when realization is considered, the SSP) if and only if (G, λ) is a yes-instance.

Lemma 6.5. If there is a state-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{S}| \leq \kappa$ and has the τ-ESSP, then there is a λ-VC for $G = (\mathfrak{U}, M)$.

Proof:
Let $\overline{B}_G$ be a state-removal of $\overline{A}_G$ that satisfies $|\mathfrak{S}| \leq \kappa$ and has the τ-ESSP, and let $\mathcal{S} = \{v \in \mathfrak{U} \mid \exists s \in \mathfrak{S} : \xrightarrow{v} s\}$. Note that $|\mathcal{S}| \leq |\mathfrak{S}| \leq \kappa = \lambda$, which can be seen as follows: On the one hand, if $t_{i,1} \in \mathfrak{S}$ for some $i \in \{0,\dots,m-1\}$, which implies $v_{i_0}, v_{i_1} \in \mathcal{S}$, then $t_{i,2} \in \mathfrak{S}$. (Otherwise, $t_{i,2}$ would no reachable, which is excluded by the definition of TS.) On the other hand, there is no other kind of state that is adjacent to two different vertex events.

We argue that $\mathcal{S}$ is a vertex cover:
Let $i \in \{0,\dots,m-1\}$ be arbitrary but fixed. If $\{v_{i_0}, v_{i_1}\} \cap \mathcal{S} = \emptyset$, then $\overline{T}_i$, and $\overline{F}_{i_0}$, and $\overline{F}_{i_1}$ are completely present in $\overline{B}_G$. Similar to the proof of Lemma 4.7 one argues that this contradicts the τ-ESSP of $\overline{B}_G$. Hence $\{v_{i_0}, v_{i_1}\} \cap \mathcal{S} \neq \emptyset$, and the arbitrariness of i implies the claim. $\square$

For the converse direction, we present the following lemma:

Lemma 6.6. If there is a λ-VC for $G = (\mathfrak{U}, M)$, then there is a state-removal $\overline{B}_G$ of $\overline{A}_G$ that satisfies $|\mathfrak{S}| \leq \kappa$ and has the τ-ESSP, and the τ-SSP.

Proof:
Let $\mathcal{S} = \{v_{\ell_0}, \dots, v_{\ell_{\lambda-1}}\}$ be a vertex cover of G, and let $\overline{B}_G$ be the TS that originates from $\overline{A}_G$ by the removal of the states $f_{\ell_0,1}, f_{\ell_1,1}, \dots, f_{\ell_{\lambda-1},1}$ (and the corresponding adjacent edges), and nothing else. One finds our that $\overline{B}_G$ is a well-defined (i.e. reachable) TS, which satisfies $|\mathfrak{S}| \leq \kappa = \lambda$. Similar to the proof of Lemma 5.6, one argues that $\overline{B}_G$ has both the τ-SSP, and the τ-ESSP. The claim follows. □

7. Concluding remarks

In this paper, we characterized the computational complexity of finding a label-splitting of a TS A that allows implementing τ-net for all types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$ and all implementations previously considered in the literature. By the results of [20, 21, 22], the synthesis problem aiming at language-simulation and realization is NP-complete for all types $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$ and $\omega \cap \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\} \neq \emptyset$ and $\omega \cap \{\mathsf{set}, \mathsf{res}\} \neq \emptyset$. Hence, their corresponding label-splitting problem is also NP-complete. Moreover, similar to the proof of Theorem 3.3, one argues that label-splitting aiming at language-simulation or realization is trivial for $\tau = \{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ when $\omega \subseteq \{\mathsf{res}, \mathsf{set}\}$: the relabeling of an input TS A would result in some ESSAs, and since τ does not allow for solving ESSAs, either A already has the separation properties or it has to be rejected. Moreover, we already know that synthesis aiming at embedding is NP-complete for all Boolean types τ with $\{\mathsf{nop}, \mathsf{swap}\} \subseteq \tau$ and $\tau \cap \{\mathsf{res}, \mathsf{set}\} = \emptyset$ [23]. Hence, again their corresponding label-splitting problem is also NP-complete. Altogether, with the present work, the complexity of τ-label-splitting is characterized for all 64 Boolean types with $\{\mathsf{nop}, \mathsf{swap}\} \subseteq \tau$ and all implementations. It remains future work to determine the complexity of the label-splitting problem for the swap-free and nop-equipped Boolean types whose underlying synthesis problem is in P.

In order to avoid to deal with the intractability of label-splitting, we also analyzed the complexity of various element removal techniques to render a τ-implementable TS, for similar types $\tau \cup \omega$ with $\emptyset \neq \omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. Unfortunately, it turns out that these techniques are also NP-complete for (almost) all τ's, and all implementations. There is however a single case that, surprisingly, remains undetermined: the complexity for event-removal when $\omega = \emptyset$. Hence, it remains for future work to investigate this special case, and to search for other techniques that may have a better complexity to make a TS implementable, like for example the insertion of states, edges or events.

Moreover, one might consider κ as a parameter and investigate the discussed modification techniques from the point of view of parameterized complexity.

Acknowledgements.

We would like to thank the unknown reviewers of this paper and the previous versions of this paper for their valuable comments.

References

[1] Badouel E, Caillaud B, Darondeau P. Distributing Finite Automata Through Petri Net Synthesis. *Formal Asp. Comput.*, 2002. **13**(6):447–470. doi:10.1007/s001650200022.

[2] van der Aalst WMP. Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, 2011. doi:10.1007/978-3-642-19345-3.

[3] Holloway LE, Krogh BH, Giua A. A Survey of Petri Net Methods for Controlled Discrete Event Systems. *Discrete Event Dynamic Systems*, 1997. **7**(2):151–190. doi:10.1023/A:1008271916548.

[4] Cortadella J, Kishinevsky M, Kondratyev A, Lavagno L, Yakovlev A. A region-based theory for state assignment in speed-independent circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 1997. **16**(8):793–812. doi:10.1109/43.644602.

[5] Badouel E, Bernardinello L, Darondeau P. Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015. doi:10.1007/978-3-662-47967-4.

[6] Carmona J. The Label Splitting Problem. *Trans. Petri Nets Other Model. Concurr.*, 2012. **6**:1–23. doi: 10.1007/978-3-642-35179-2_1. URL `https://doi.org/10.1007/978-3-642-35179-2_1`.

[7] Cortadella J, Kishinevsky M, Lavagno L, Yakovlev A. Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 1998. **47**(8):859–882.

[8] Schlachter U, Wimmel H. Relabelling LTS for Petri Net Synthesis via Solving Separation Problems. *Trans. Petri Nets Other Model. Concurr.*, 2019. **14**:222–254. doi:10.1007/978-3-662-60651-3_9.

[9] Schlachter U, Wimmel H. Optimal Label Splitting for Embedding an LTS into an arbitrary Petri Net Reachability Graph is NP-complete. *CoRR*, 2020. **abs/2002.04841**. `2002.04841`.

[10] Tredup R. Finding an Optimal Label-Splitting to Make a Transition System Petri Net Implementable: a Complete Complexity Characterization. In: Proceedings of the 21st Italian Conference on Theoretical Computer Science, Ischia, Italy, September 14-16, 2020. 2020 pp. 131–144.

[11] Devillers RR, Tredup R. Some Basic Techniques allowing Petri Net Synthesis: Complexity and Algorithmic Issues. *CoRR*, 2021. **abs/2112.03605**. `2112.03605`, URL `https://arxiv.org/abs/2112.03605`.

[12] Tredup R. Edge, Event and State Removal: The Complexity of Some Basic Techniques that Make Transition Systems Petri Net Implementable. In: Application and Theory of Petri Nets and Concurrency - 42nd International Conference, PETRI NETS 2021, Virtual Event, June 23-25, 2021, Proceedings. 2021 pp. 253–273. doi:10.1007/978-3-030-76983-3_13.

[13] Badouel E, Darondeau P. Trace Nets and Process Automata. *Acta Inf.*, 1995. **32**(7):647–679. doi: 10.1007/BF01186645.

[14] Kleijn J, Koutny M, Pietkiewicz-Koutny M, Rozenberg G. Step semantics of boolean nets. *Acta Inf.*, 2013. **50**(1):15–39. doi:10.1007/s00236-012-0170-2.

[15] Montanari U, Rossi F. Contextual Nets. *Acta Inf.*, 1995. **32**(6):545–596. doi:10.1007/BF01178907.

[16] Pietkiewicz-Koutny M. Transition Systems of Elementary Net Systems with Inhibitor Arcs. In: ICATPN, volume 1248 of *Lecture Notes in Computer Science*. Springer, 1997 pp. 310–327. doi: 10.1007/3-540-63139-9_43.

[17] Rozenberg G, Engelfriet J. Elementary Net Systems. In: Petri Nets, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1996 pp. 12–121. doi:10.1007/3-540-65306-6_14.

[18] Schmitt V. Flip-Flop Nets. In: STACS, volume 1046 of *Lecture Notes in Computer Science*. Springer, 1996 pp. 517–528. doi:10.1007/3-540-60922-9_42.

[19] Badouel E, Bernardinello L, Darondeau P. The Synthesis Problem for Elementary Net Systems is NP-Complete. *Theor. Comput. Sci.*, 1997. **186**(1-2):107–134. doi:10.1016/S0304-3975(96)00219-8.

[20] Tredup R. The Complexity of Synthesizing nop-Equipped Boolean Petri Nets from g-Bounded Inputs. *Trans. Petri Nets Other Model. Concurr.*, 2021. **15**:101–125.

[21] Tredup R, Rosenke C. The Complexity of Synthesis for 43 Boolean Petri Net Types. In: TAMC, volume 11436 of *Lecture Notes in Computer Science*. Springer, 2019 pp. 615–634. doi:10.1007/978-3-030-\ linebreak14812-6.

[22] Tredup R, Rosenke C. On the Hardness of Synthesizing Boolean Nets. In: ATAED@Petri Nets/ACSD, volume 2371 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019 pp. 71–86.

[23] Tredup R, Erofeev E. The Complexity of Boolean State Separation. In: Theoretical Aspects of Computing - ICTAC 2020 - 17th International Colloquium, Macau, China, November 30 - December 4, 2020, Proceedings. 2020 pp. 123–142. doi:10.1007/978-3-030-64276-1_7.

[24] Badouel E, Bernardinello L, Darondeau P. Polynomial Algorithms for the Synthesis of Bounded Nets. In: TAPSOFT, volume 915 of *Lecture Notes in Computer Science*. Springer, 1995 pp. 364–378. doi: 10.1007/3-540-59293-8_207.

[25] Tredup R. The Complexity of the Label-Splitting-Problem for Flip-Flop-Nets. In: Reachability Problems - 14th International Conference, RP 2020, Paris, France, October 19-21, 2020, Proceedings. 2020 pp. 148–163. doi:10.1007/978-3-030-61739-4_10.

[26] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979. ISBN 0-7167-1044-7.