The Complexity of Boolean State Separation

Ronny Tredup[1(✉)] and Evgeny Erofeev[2]

[1] Institut für Informatik, Theoretische Informatik, Universität Rostock, Albert-Einstein-Straße 22, 18059 Rostock, Germany
ronny.tredup@uni-rostock.de

[2] Department of Computing Science, Carl von Ossietzky Universität Oldenburg, 26111 Oldenburg, Germany
evgeny.erofeev@informatik.uni-oldenburg.de

Abstract. For a Boolean type of nets τ, a transition system A is synthesizeable into a τ-net N if and only if distinct states of A correspond to distinct markings of N, and N prevents a transition firing if there is no related transition in A. The former property is called τ-state separation property (τ-SSP) while the latter – τ-event/state separation property (τ-ESSP). A is embeddable into the reachability graph of a τ-net N if and only if A has the τ-SSP. This paper presents a complete characterization of the computational complexity of τ-SSP for all Boolean Petri net types.

Keywords: Boolean Petri nets · Boolean state separation · Complexity characterization

1 Introduction

Providing a powerful mechanism for the modeling of conflicts, dependencies and parallelism, Petri nets are widely used for studying and simulating concurrent and distributed systems. In system analysis, one aims to check behavioral properties of system models, and many of these properties are decidable [7] for Petri nets and their reachability graphs, which represent systems' behaviors. The task of system synthesis is opposite: a (formal) specification of the system's behavior is given, and the goal is then to decide whether this behavior can be implemented by a Petri net. In case of a positive decision, such a net should be constructed.

Boolean Petri nets form a simple yet rich and powerful family of Petri nets [3,4,8,10,12,13,16], applied in asynchronous circuits design [25,27], concurrent constraint programs [9] and analysis of biological systems models [6]. In Boolean nets, each place contains at most one token, for any reachable marking. Hence, a place can be interpreted as a Boolean condition that is *true* if marked

E. Erofeev—Supported by DFG through grant Be 1267/16-1 ASYST.

V. K. I. Pun et al. (Eds.): ICTAC 2020, LNCS 12545, pp. 123–142, 2020.
https://doi.org/10.1007/978-3-030-64276-1_7

and *false* otherwise. A place p and a transition t of such a net are related by one of the Boolean *interactions* that define in which way p and t influence each other. The interaction inp (out) defines that p must be *true* (*false*) before and *false* (*true*) after t's firing; free (used) implies that t's firing proves that p is *false* (*true*); nop means that p and t do not affect each other at all; res (set) implies that p may initially be both *false* or *true* but after t's firing it is *false* (*true*); swap means that t inverts p's current Boolean value. Boolean Petri nets are classified by the sets of interactions that can be applied. A set τ of Boolean interactions is called a *type of net*, and a net N is of type τ (a τ-*net*) if it applies at most the interactions of τ. For a type τ, the τ-*synthesis* problem consists in deciding whether a specification given in the form of a labeled transition system (TS) is isomorphic to the reachability graph of some τ-net N, and in constructing N if it exists. The complexity of τ-synthesis has been studied in different settings [17,20,23], and varies substantially from polynomial [16] to NP-complete [2].

In order to perform synthesis, that is, to implement the behavior specified by the given TS with a τ-net, two general problems have to be resolved: The τ-net has to distinguish the global states of the TS, and the τ-net has to prevent actions at states where they are not permitted by the TS. In the literature [3], the former requirement is usually referred to as τ-*state separation property* (τ-SSP), while the latter – τ-*event/state separation property* (τ-ESSP). Both τ-SSP and τ-ESSP define decision problems that ask whether a given TS fulfills the respective property. The present work focuses exclusively on the computational complexity of τ-SSP. The interest to state separation is motivated in several ways. First, many synthesis approaches are very sensitive to the size of the input's state space. This raises the question if some initial, so-called *pre-synthesis* procedures [5,26] can be employed as a quick-fail mechanism, i.e., techniques with a small computational overhead that would gain some helpful information for the main synthesis, or reject the input if exact (up to isomorphism) synthesis is not possible. Since τ-synthesis allows a positive decision if and only if τ-SSP and τ-ESSP do [3], an efficient decision procedure for τ-SSP could serve as a quick-fail pre-process check. Second, if exact synthesis is not possible for the given TS, one may want to have a simulating model, i.e., a τ-net that over-approximates [14,15] the specified behavior with some possible supplement. Formally, the TS then has to be injectively embeddable into the reachability graph of a τ-net. It is well known from the literature [3] that a TS can be embedded into the reachability graph of a τ-net if and only if it has the τ-SSP. Finally, in comparison to τ-ESSP, so far the complexity of τ-SSP is known to be as hard [1,16,23,24] or actually less hard [18,19]. On the contrary, in this paper, for some types of nets, deciding the τ-SSP is proven harder (NP-complete) than deciding the τ-ESSP (polynomial), e.g., for $\tau = \{\mathsf{nop}, \mathsf{res}, \mathsf{set}\}$. From the contribution perspective, the τ-SSP has been previously considered only in the broader context of τ-synthesis, and only for selected types [16,19,22,23]. In this paper, we completely characterize the complexity of τ-SSP for all 256 Boolean types of nets and discover 150 new hard types, cf. §1–§3, §6, §9 of Fig. 1, and 4 new tractable types, cf. Fig. 1 §10,

and comprise the known results for the other 102 types as well, cf. Fig. 1 §4, §5, §7, §8. In particular, our characterization categorizes Boolean types with regard to their behavioral capabilities, resulting from the Boolean interactions involved. This reveals the internal organization of the entire class of Boolean nets, suggesting a general approach for reasoning about its subclasses.

This paper is organized as follows. In Sect. 2, all the necessary notions and definitions will be given. Section 3 presents NP-completeness results for the types with nop-interaction. τ-SSP for types without nop is investigated in Sect. 4. Concluding remarks are given in Sect. 5. Due to space restrictions, one proof is omitted but can be found in [21].

§	Type of net τ	Complexity	Quantity
1	$\{\mathsf{nop}, \mathsf{res}, \mathsf{set}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$	NP-complete	16
2	$\{\mathsf{nop}, \mathsf{res}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}, \mathsf{set}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$	NP-complete	32
3	$\{\mathsf{nop}, \mathsf{res}, \mathsf{set}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}, \mathsf{out}, \mathsf{res}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}, \mathsf{inp}, \mathsf{set}\} \cup \omega$ with $\omega \subseteq \{\mathsf{out}, \mathsf{used}, \mathsf{free}\}$	NP-complete	32
4	$\{\mathsf{nop}, \mathsf{res}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}, \mathsf{set}\} \cup \omega$ with $\omega \subseteq \{\mathsf{out}, \mathsf{used}, \mathsf{free}\}$	polynomial	16
5	$\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$ and $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{used}\}$	NP-complete	2
6	$\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{free}\}$ and $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}, \mathsf{inp}\} \cup \omega$ and $\{\mathsf{nop}, \mathsf{out}\} \cup \omega$ with $\omega \subseteq \{\mathsf{used}, \mathsf{free}\}$	NP-complete	10
7	$\{\mathsf{nop}, \mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, $\{\mathsf{nop}\} \cup \omega$ with $\omega \subseteq \{\mathsf{used}, \mathsf{free}\}$	polynomial	20
8	$\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$	polynomial	64
9	$\{\mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$ and $\omega \cap \{\mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\} \neq \emptyset$	NP-complete	60
10	$\{\mathsf{swap}\} \cup \omega$ with $\omega \subseteq \{\mathsf{inp}, \mathsf{out}\}$	polynomial	4

Fig. 1. Overview of the computational complexity of τ-SSP for Boolean types of nets τ. The gray area highlights the new results that this paper provides.

2 Preliminaries

In this section, we introduce necessary notions and definitions, supported by illustrations and examples, and some basic results that are used throughout the paper.

Transition Systems. A (finite, deterministic) *transition system* (TS, for short) $A = (S, E, \delta)$ is a directed labeled graph with the set of nodes S (called *states*), the set of labels E (called *events*) and partial *transition function* $\delta : S \times E \longrightarrow S$. If $\delta(s, e)$ is defined, we say that e *occurs* at state s, denoted by $s \xrightarrow{e}$. By $s \xrightarrow{e} s' \in A$, we denote $\delta(s, e) = s'$. This notation extends to paths, i.e., $q_0 \xrightarrow{e_1} \ldots \xrightarrow{e_n} q_n \in$

A denotes $q_{i-1} \xrightarrow{e_i} q_i \in A$ for all $i \in \{1, \ldots, n\}$. A TS A is *loop-free*, if $s \xrightarrow{e} s' \in A$ implies $s \neq s'$. A loop-free TS A is *bi-directed* if $s' \xrightarrow{e} s \in A$ implies $s \xrightarrow{e} s' \in A$. We say $s_0 \xleftrightarrow{e_1} \ldots \xleftrightarrow{e_n} s_n \in A$ is a simple bi-directed path if $s_i \neq s_j$ for all $i \neq j$ with $i, j \in \{0, \ldots, n\}$. An *initialized* TS $A = (S, E, \delta, \iota)$ is a TS with a distinct *initial* state $\iota \in S$, where every state $s \in S$ is *reachable* from ι by a directed labeled path.

Boolean Types of Nets [3]**.** The following notion of Boolean types of nets allows to capture *all* Boolean Petri nets in a *uniform* way. A *Boolean type of net* $\tau = (\{0,1\}, E_\tau, \delta_\tau)$ is a TS such that E_τ is a subset of the *Boolean interactions*: $E_\tau \subseteq I = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{swap}, \mathsf{used}, \mathsf{free}\}$. Each interaction $i \in I$ is a binary partial function $i : \{0,1\} \to \{0,1\}$ as defined in Fig. 2. For all $x \in \{0,1\}$ and all $i \in E_\tau$, the transition function of τ is defined by $\delta_\tau(x, i) = i(x)$. Notice

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

Fig. 2. All interactions i of I. If a cell is empty, then i is undefined on the respective x.

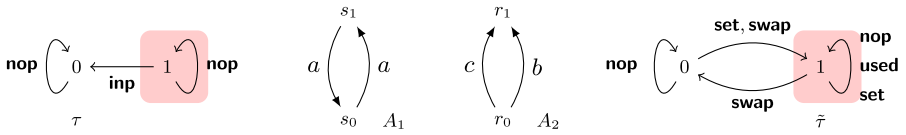


Fig. 3. Left: $\tau = \{\mathsf{nop}, \mathsf{inp}\}$. Right: $\tilde{\tau} = \{\mathsf{nop}, \mathsf{set}, \mathsf{swap}, \mathsf{used}\}$. The red colored area emphasizes the inside. The only SSP atom of A_1 is (s_0, s_1). It is $\tilde{\tau}$-solvable by $R_1 = (sup_1, sig_1)$ with $sup_1(s_0) = 0$, $sup_1(s_1) = 1$, $sig_1(a) = \mathsf{swap}$. Thus, A_1 has the $\tilde{\tau}$-separative set $\mathcal{R} = \{R_1\}$. The SSP atom (s_0, s_1) is not τ-solvable. The only SSP atom (r_0, r_1) in A_2 can be solved by $\tilde{\tau}$-region $R_2 = (sup_2, sig_2)$ with $sup_2(r_0) = 0$, $sup_2(r_1) = 1$, $sig_2(b) = \mathsf{set}$, $sig_2(c) = \mathsf{swap}$. Thus, A_2 has the $\tilde{\tau}$-SSP. The same atom can also be solved by τ-region $R_3 = (sup_3, sig_3)$ with $sup_3(r_0) = 1$, $sup_3(r_1) = 0$, $sig_3(b) = sig_3(c) = \mathsf{inp}$. Hence, A_2 has the τ-SSP, as well. (Color figure online)

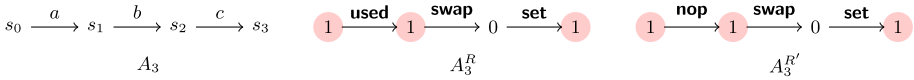


Fig. 4. Left: TS A_3, a simple directed path. If $\tilde{\tau}$ is defined as in Fig. 3, then $sup(\iota) = 1$, $sig(a) = \mathsf{used}$, $sig(b) = \mathsf{swap}$ and $sig(c) = \mathsf{set}$ implicitly defines the $\tilde{\tau}$-region $R = (sup, sig)$ of A_3 as follows: $sup(s_1) = \delta_{\tilde{\tau}}(1, \mathsf{used}) = 1$, $sup(s_2) = \delta_{\tilde{\tau}}(1, \mathsf{swap}) = 0$ and $sup(s_3) = \delta_{\tilde{\tau}}(0, \mathsf{set}) = 1$. Middle: The image A_3^R of A_3 (under R). One easily verifies that $\delta_{A_3}(s, e) = s'$ implies $\delta_{\tilde{\tau}}(sup(s), sig(e)) = sup(s')$, cf. Fig. 3. In particular, R is sound. For event b, the edge defined by $\delta_{A_3}(s_1, b) = s_2$ is mapped into $\delta_{\tilde{\tau}}(1, \mathsf{swap}) = 0$ under R, i.e., b makes a state change on the path; similar for $sig(c) = \mathsf{set}$. R is not normalized, since $sup(s_0) = sup(s_1)$, but $sig(a) = \mathsf{used} \neq \mathsf{nop}$. Right: The image $A_3^{R'}$ of A_3 under the normalized $\tilde{\tau}$-region R' that is similar to R but replaces used by nop.

that a type τ is completely determined by E_τ. Hence we often identify τ with E_τ, cf. Fig. 1. Moreover, since I captures all meaningful Boolean interactions [23, p. 617] and τ is defined by $E_\tau \subseteq I$, there are 256 Boolean types of nets at all. For a Boolean type of net τ, we say that its state 1 is *inside* and 0 is *outside*. An interaction $i \in E_\tau$ *exits* if $1 \xrightarrow{i} 0$, *enters* if $0 \xrightarrow{i} 1$, *saves 1* if $1 \xrightarrow{i} 1$ and *saves 0* if $0 \xrightarrow{i} 0$. Accordingly, we group interactions together by $\mathfrak{exit} = \{\mathsf{inp}, \mathsf{res}, \mathsf{swap}\}$, $\mathfrak{enter} = \{\mathsf{out}, \mathsf{set}, \mathsf{swap}\}$, $\mathfrak{save}_1 = \{\mathsf{nop}, \mathsf{set}, \mathsf{used}\}$, $\mathfrak{save}_0 = \{\mathsf{nop}, \mathsf{res}, \mathsf{free}\}$ and $\mathfrak{save} = \mathfrak{save}_1 \cup \mathfrak{save}_0$.

For a net of type τ (τ-net), the interactions of τ determine relations between *places* and *transitions* of the net. For instance, if a place p and a transition t are related via inp, then p has to be marked (*true*) to allow t to fire, and becomes unmarked (*false*) after the firing (cf. Fig. 2). Since we are only concerned with state separation, we omit the formal definition of τ-nets and rather refer to, e.g., [3] for a comprehensive introduction to the topic.

τ-Regions. The following notion of τ-regions is the key concept for state separation. A τ-region $R = (sup, sig)$ of TS $A = (S, E, \delta, \iota)$ consists of the mappings *support* $sup : S \to \{0, 1\}$ and *signature* $sig : E \to E_\tau$, such that for every edge $s \xrightarrow{e} s'$ of A, the edge $sup(s) \xrightarrow{sig(e)} sup(s')$ belongs to the type τ; we also say sup *allows* (sig and thus the region) R. 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). For region R and path P, event e_i with $1 \leq i \leq n$ is called *state changing* on the path P^R, if $sup(q_{i-1}) \neq sup(q_i)$ for $q_{i-1} \xrightarrow{e_i} q_i$ in P. Notice that R is *implicitly* defined by $sup(\iota)$ and sig: Since A is reachable, for every state $s \in S$, there is a path $\iota \xrightarrow{e_1} \dots \xrightarrow{e_n} s_n$ such that $s = s_n$. Thus, since τ is deterministic, we inductively obtain $sup(s_{i+1})$ by $sup(s_i) \xrightarrow{sig(e_{i+1})} sup(s_{i+1})$ for all $i \in \{0, \dots, n-1\}$ and $s_0 = \iota$. Hence, we can compute sup and thus R purely from $sup(\iota)$ and sig, cf. Fig. 4. If $\mathsf{nop} \in \tau$, then a τ-region $R = (sup, sig)$ of a TS A is called *normalized* if $sig(e) = \mathsf{nop}$ for as many events e as sup allows: for all $e \in E$, $sig(e) \notin \{\mathsf{used}, \mathsf{free}\}$ and if $sig(e) \in \mathfrak{exit} \cup \mathfrak{enter}$ then there is $s \xrightarrow{e} s' \in A$ such that $sup(s) \neq sup(s')$.

τ-State Separation Property. A pair (s, s') of distinct states of A defines a *state separation atom* (SSP atom). A τ-region $R = (sup, sig)$ *solves* (s, s') if $sup(s) \neq sup(s')$. If R exists, then (s, s') is called *τ-solvable*. If $s \in S_A$ and, for all $s' \in S_A \setminus \{s\}$, the atom (s, s') is τ-solvable, then s is called τ-solvable. A TS has the *τ-state separation property* (τ-SSP) if all of its SSP atoms are τ-solvable. A set $\mathcal{R}$ of τ-regions of A is called *τ-separative* if for each SSP atom of A there is a τ-region R in $\mathcal{R}$ that solves it. By the next lemma, if $\mathsf{nop} \in \tau$, then A has the τ-SSP if and only if it has a τ-separative set of normalized τ-regions:

Lemma 1. *Let A be a TS and τ be a nop-equipped Boolean type of nets. There is a τ-separative set $\mathcal{R}$ of A if and only if there is a τ-separative set of normalized τ-regions of A.*

Proof. The *if*-direction is trivial. *Only-if*: Let $R = (sup, sig)$ be a non-normalized τ-region, i.e., there is $e \in E_A$ such that $s \xrightarrow{e} s' \in A$ implies $sup(s) = sup(s')$. Since τ is nop-equipped, $sup(s) \xrightarrow{\mathsf{nop}} sup(s') \in \tau$ for all $s \xrightarrow{e} s' \in A$. Thus, a τ-region $R' = (sup, sig')$ can be constructed from R, where sig' is equal to sig except for $sig'(e) = \mathsf{nop}$. Since E_A is finite, a normalized region can be obtained from R by inductive application of this procedure. □

By the following lemma, τ-SSP and $\tilde{\tau}$-SSP are equivalent if τ and $\tilde{\tau}$ are isomorphic:

Lemma 2 (Without proof). *If τ and $\tilde{\tau}$ are isomorphic types of nets then a TS A has the τ-SSP if and only if A has the $\tilde{\tau}$-SSP.*

In this paper, we consider the τ-SSP also as decision problem that asks whether a given TS A has the τ-SSP. The decision problem τ-SSP is in NP: By definition, A has at most $|S|^2$ SSP atoms. Hence, a Turing-machine can (non-deterministically) guess a τ-separative set $\mathcal{R}$ such that $|\mathcal{R}| \leq |S|^2$ and (deterministically) check in polynomial time its validity if it exists.

In what follows, some of our NP-completeness results base on polynomial-time reductions of the following decision problem, which is known to be NP-complete [11]:

Cubic Monotone 1-in-3 3Sat. (CM 1-IN-3 3SAT) The input is a Boolean formula $\varphi = \{\zeta_0, \ldots, \zeta_{m-1}\}$ of negation-free three-clauses $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$, where $i \in \{0, \ldots, m-1\}$, with set of variables $X = \bigcup_{i=0}^{m-1} \zeta_i$; every variable $v \in X$ occurs in exactly three clauses, implying $|X| = m$. The question to decide is whether there is a (one-in-three model) $M \subseteq X$ satisfying $|M \cap \zeta_i| = 1$ for all $i \in \{0, \ldots, m-1\}$.

*Example 1 (*CM 1-IN-3 3SAT*).* The instance $\varphi = \{\zeta_0, \ldots, \zeta_5\}$ of CM 1-IN-3 3SAT with set of variables $X = \{X_0, \ldots, X_5\}$ and clauses $\zeta_0 = \{X_0, X_1, X_2\}$, $\zeta_1 = \{X_0, X_2, X_3\}$, $\zeta_2 = \{X_0, X_1, X_3\}$, $\zeta_3 = \{X_2, X_4, X_5\}$, $\zeta_4 = \{X_1, X_4, X_5\}$ and $\zeta_5 = \{X_3, X_4, X_5\}$ has the one-in-three model $M = \{X_0, X_4\}$.

3 Deciding the State Separation Property for nop-equipped Types

In this section, we investigate the computational complexity of nop-equipped Boolean types of nets. For technical reasons, we separately consider the types that include neither res nor set (§5–§7 in Fig. 1) and the ones that have at least one of them (§1–§4 in Fig. 1).

First of all, the fact that τ-SSP is polynomial for the types of §7 in Fig. 1 is implied by the results of [16] [23, p. 619]. Moreover, for the types of Fig. 1 §5 the NP-completeness of τ-SSP has been shown in [24] ($\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$), and in [19] ($\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{used}\}$, there referred to as 1-bounded P/T-nets). Thus, in order to complete the complexity characterization for the nop-equipped types

that neither contain res nor set, it only remains to ascertain the complexity of τ-SSP for the types Fig. 1 §6. The following Subsect. 3.1 proves that τ-SSP is NP-complete for these types.

Then, we proceed with the types of §1–§4 in Fig. 1. The fact that τ-SSP is polynomial for the types of §4 follows from [23, p. 619]. The NP-completeness of τ-SSP for the remaining types (§1–§3) will be demonstrated in Subsect. 3.2.

3.1 Complexity of τ-SSP for nop-equipped Types Without res and set

The following theorem summarizes the complexity for the types of §5–§7 in Fig. 1.

Theorem 1. *Let τ be a nop-equipped Boolean type of nets such that $\tau \cap \{res, set\} = \emptyset$. The τ-SSP is NP-complete if $\tau \cap \{inp, out\} \neq \emptyset$ and $swap \notin \tau$, otherwise it is polynomial.*

As just discussed in Sect. 3, to complete the proof of Theorem 1 it remains to characterize the complexity of τ-SSP for the types of Fig. 1 §6. Since τ-SSP is NP-complete if $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$ [24], by Lemma 1, the τ-SSP is also NP-complete if $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{free}\}$ and $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$.

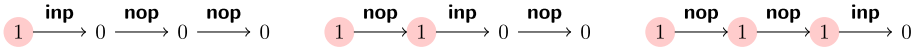
Thus, in what follows, we restrict ourselves to the types $\tau = \{\mathsf{nop}, \mathsf{inp}\} \cup \omega$ and $\tau = \{\mathsf{nop}, \mathsf{out}\} \cup \omega$, where $\omega \subseteq \{\mathsf{used}, \mathsf{free}\}$, and argue that their τ-SSP are NP-complete. To do so, we let $\tau = \{\mathsf{nop}, \mathsf{inp}\}$ and show the hardness of τ-SSP by a reduction of CM 1-IN-3 3SAT. By Lemma 1, this also implies the hardness of $\tau \cup \omega$-SSP, where $\omega \subseteq \{\mathsf{used}, \mathsf{free}\}$. Furthermore, by Lemma 2, the latter shows the NP-completeness of τ-SSP if $\tau = \{\mathsf{nop}, \mathsf{out}\} \cup \omega$ and $\omega \subseteq \{\mathsf{used}, \mathsf{free}\}$. The following paragraph introduces the intuition of our reduction approach.

The Roadmap of Reduction. Let $\tau = \{\mathsf{nop}, \mathsf{inp}\}$ and $\varphi = \{\zeta_0, \ldots, \zeta_{m-1}\}$ be an input of CM 1-IN-3 3SAT with variables $X = \{X_0, \ldots, X_{m-1}\}$ and clauses $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$ for all $i \in \{0, \ldots, m-1\}$. To show the NP-completeness of τ-SSP, we reduce a given input φ to a TS A_φ (i.e., an input of τ-SSP) as follows: For every clause $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$, the TS A_φ has a directed labeled path P_i that represents ζ_i by using its variables as events:

$$P_i = \; t_{i,0} \xrightarrow{X_{i_0}} t_{i,1} \xrightarrow{X_{i_1}} t_{i,2} \xrightarrow{X_{i_2}} t_{i,3}$$

We ensure by construction that A_φ has an SSP atom α such that if $R = (sup, sig)$ is a τ-region solving α, then $sup(t_{i,0}) = 1$ and $sup(t_{i,3}) = 0$ for all $i \in \{0, \ldots, m-1\}$. Thus, for all $i \in \{0, \ldots, m-1\}$, the path P_i^R is a path from 1 to 0 in τ. First, this obviously implies that there is an event $e \in \{X_{i_0}, X_{i_1}, X_{i_2}\}$ such that $sig(e) = \mathsf{inp}$. Second, it is easy to see that there is no path in τ on which inp occurs twice, cf. Fig. 3. The following figure sketches all possibilities of P_i^R, i.e., $sig(X_{i_0}) = \mathsf{inp}$ and $sig(X_{i_1}) = sig(X_{i_2}) = \mathsf{nop}$, $sig(X_{i_1}) = \mathsf{inp}$ and

$sig(X_{i_0}) = sig(X_{i_2}) = \mathsf{nop}$, and $sig(X_{i_2}) = \mathsf{inp}$ and $sig(X_{i_0}) = sig(X_{i_1}) = \mathsf{nop}$, respectively:



Hence, the event e is unique. Since this is simultaneously true for all paths $P_0, \ldots, P_{m-1}$, the set $M = \{e \in X \mid sig(e) = \mathsf{inp}\}$ selects exactly one variable per clause and thus defines a one-in-three model of φ. Altogether, this approach shows that if A_φ has the τ-SSP, which implies that α is τ-solvable, then φ has a one-in-three model.

Conversely, our construction ensures that if φ has a one-in-three model, then α and the other separation atoms of A_φ are τ-solvable, that is, A_φ has the τ-SSP.

The Reduction of A_φ for $\tau = \{\mathsf{nop}, \mathsf{inp}\}$. In the following, we introduce the announced TS A_φ, cf. Fig. 5. The initial state of A_φ is $t_{0,0}$. First of all, the TS A_φ has the following path P that provides the announced SSP atom $\alpha = (t_{m,0}, t_{m+1,0})$:

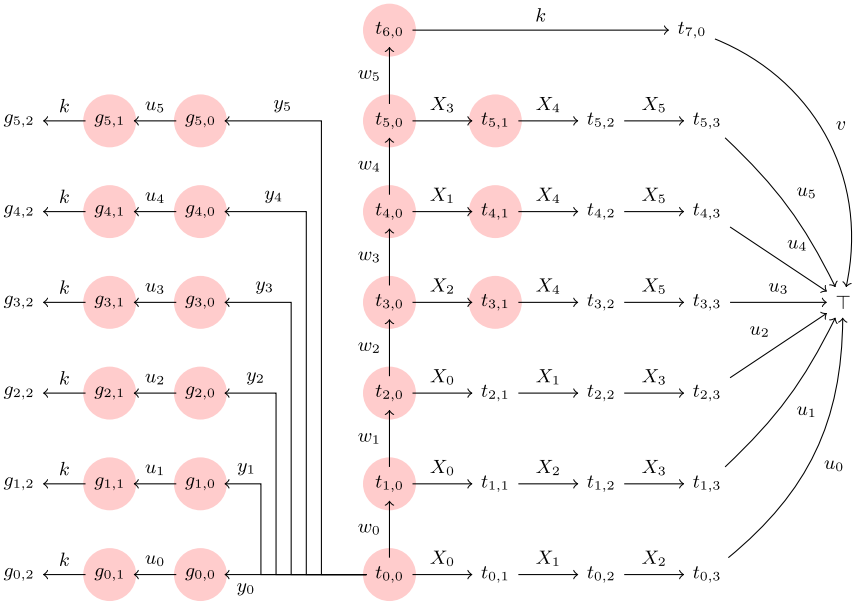
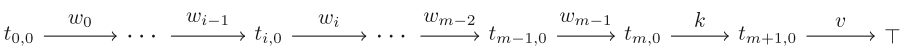


Fig. 5. The TS A_φ originating from the input φ of Example 1, which has the one-in-three model $M = \{X_0, X_4\}$. The colored area sketches the region R_M of Lemma 4 that solves $\alpha = (t_{6,0}, t_{7,0})$. (Color figure online)



Moreover, for every $i \in \{0, \ldots, m-1\}$, the TS A_φ has the following path T_i that uses the variables of $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$ as events and provides the sub-

path $P_i = t_{i,0} \xrightarrow{X_{i_0}} \ldots \xrightarrow{X_{i_2}} t_{i,3}$:

$$t_{i,0} \xrightarrow{X_{i_0}} t_{i,1} \xrightarrow{X_{i_1}} t_{i,2} \xrightarrow{X_{i_2}} t_{i,3} \xrightarrow{u_i} \top$$

Finally, the TS A_φ has, for all $i \in \{0, \ldots, m-1\}$, the following path G_i:

$$t_{0,0} \xrightarrow{y_i} g_{i,0} \xrightarrow{u_i} g_{i,1} \xrightarrow{k} g_{i,2}$$

Notice that the paths P and $T_0, \ldots, T_{m-1}$ have the same "final"-state $\perp$. Obviously, the size of A_φ is polynomial in the size of φ. The following Lemma 3 and Lemma 4 prove the validity of our reduction and thus complete the proof of Theorem 1.

Lemma 3. *Let $\tau = \{\mathsf{nop}, \mathsf{inp}\}$. If A_φ has the τ-SSP, then φ has a one-in-three model.*

Proof. Let $R = (sup, sig)$ be a τ-region that solves α, that is, $sup(t_{m,0}) \neq sup(t_{m+1,0})$. By $t_{m,0} \xrightarrow{k} t_{m+1,0}$ and $\tau = \{\mathsf{nop}, \mathsf{inp}\}$, this implies $sig(k) = \mathsf{inp}$, $sup(t_{m,0}) = 1$ and $sup(t_{m+1,0}) = 0$. If $s_0 \xrightarrow{e_0} \ldots \xrightarrow{e_i} s_i \xrightarrow{k} s_{i+1} \xrightarrow{e_{i+2}} \ldots \xrightarrow{e_n} s_n$ is a path in A_φ, then, by $sig(k) = \mathsf{inp}$, we get $sup(s_j) = 1$ for all $j \in \{0, \ldots, i\}$, $sup(s_j) = 0$ for all $j \in \{i+1, \ldots, n\}$ and $sig(e_j) = \mathsf{nop}$ for all $j \in \{1, \ldots, i, i+2, \ldots, n\}$. This implies $sup(\top) = 0$, $sig(v) = \mathsf{nop}$ as well as $sig(u_i) = \mathsf{nop}$ and $sup(t_{i,0}) = 1$ for all $i \in \{0, \ldots, m-1\}$. Furthermore, by $sup(\top) = 0$ and $sig(u_i) = \mathsf{nop}$, we get $sup(t_{i,3}) = 0$ for all $i \in \{0, \ldots, m-1\}$. Hence, for all $i \in \{0, \ldots, m-1\}$, the image P_i^R of P_i is a path from 1 to 0 in τ. Hence, as just discussed above, $M = \{e \in X \mid sig(e) = \mathsf{inp}\}$ selects exactly one variable per clause and defines a one-in-three model of φ. □

Lemma 4. *Let $\tau = \{\mathsf{nop}, \mathsf{inp}\}$. If φ has a one-in-three model, then A_φ has the τ-SSP.*

Proof. Let M be a one-in-three model of φ.

The following region $R_1 = (sup, sig)$ solves (s, s') for all $s \in \bigcup_{i=0}^{m-1} S(P_i)$ and all $s' \in \bigcup_{i=0}^{m-1} S(G_i)$, where $s' \neq t_{0,0}$: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{y_0, \ldots, y_{m-1}\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$.

Let $i \in \{0, \ldots, m-1\}$ be arbitrary but fixed. The following region $R_i^T = (sup, sig)$ solves (s, s') for all $s \in \{t_{i,0}, \ldots, t_{i,3}\}$ and all $s' \in \bigcup_{j=i+1}^{m-1} S(T_j) \cup \{t_{m,0}, t_{m+1,0}\}$: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{w_i\} \cup \{u_0, \ldots, u_i\} \cup \{y_{i+1}, \ldots, y_{m-1}\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$.

The following region $R_2 = (sup, sig)$ solves $(t_{m+1}, \top)$ and $(g_{i,0}, g_{i,1})$ and $(g_{i,0}, g_{i,2})$ for all $i \in \{0, \ldots, m-1\}$: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{v\} \cup \{u_0, \ldots, u_{m-1}\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$.

The following region $R_M = (sup, sig)$ uses the one-in-three model M of φ and solves α as well as $(g_{i,1}, g_{i,2})$ for all $i \in \{0, \ldots, m-1\}$: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{k\} \cup M$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$. Note Fig. 5 for an example of R_M.

Let $i \in \{0, \ldots, m-2\}$ be arbitrary but fixed. The following region $R_i^G = (sup, sig)$ solves (s, s') for all $s \in \{g_{i,0}, g_{i,1}, g_{i,2}\}$ and all $s' \in \bigcup_{j=i+1}^{m-1} S(G_j)$, where $s' \neq t_{0,0}$: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{y_{i+1}, \ldots, y_{m-1}\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$.

By the arbitrariness of i for R_i^T and R_i^G, it remains to show that $t_{i,0}, \ldots, t_{i,3}$ are pairwise separable for all $i \in \{0, \ldots, m-1\}$. Let $i \in \{0, \ldots, m-1\}$ be arbitrary but fixed. We present only a region $R_{i_0}^X = (sup, sig)$ that solves $(t_{i,0}, s)$ for all $s \in \{t_{i,1}, t_{i,2}, t_{i,3}\}$. It is then easy to see that the remaining atoms are similarly solvable. Let $j \neq \ell \in \{0, \ldots, m-1\} \setminus \{i\}$ select the other two clauses of φ that contain X_{i_0}, that is, $X_{i_0} \in \zeta_j \cap \zeta_\ell$. $R_{i_0}^X = (sup, sig)$ is defined as follows: $sup(t_{0,0}) = 1$; for all $e \in E_{A_\varphi}$, if $e \in \{X_{i_0}\} \cup \{v\} \cup \{u_0, \ldots, u_{m-1}\} \setminus \{u_i, u_j, u_\ell\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$.

Similarly, one gets regions where X_{i_1} or X_{i_2} has inp-signature. These regions solve the remaining atoms of $S(T_i) \setminus \{\top\}$. Since i was arbitrary, this completes the proof. Please note that the technical report [21] that corresponds to this paper provides graphical representations and examples for the just presented regions. □

3.2 Complexity of τ-SSP for nop-equipped Types with res or set

The next theorem states that τ-SSP is NP-complete for the nop-equipped types that have not yet been considered, cf Fig. 1 §1–§3. Moreover, it summarizes the complexity of τ-SSP for all types of Fig. 1 §1–§4:

Theorem 2. *Let τ and $\tilde{\tau}$ be Boolean type of nets and $\{\mathsf{nop}, \mathsf{res}\} \subseteq \tau$ and $\{\mathsf{nop}, \mathsf{set}\} \subseteq \tilde{\tau}$.*

1. *The τ-SSP is NP-complete if $\tau \cap \mathfrak{enter} \neq \emptyset$, otherwise it is polynomial.*
2. *The $\tilde{\tau}$-SSP is NP-complete if $\tilde{\tau} \cap \mathfrak{exit} \neq \emptyset$, otherwise it is polynomial.*

In this section we complete the proof of Theorem 2 as follows. Firstly, we let $\tau_0 = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$ and, by a reduction of τ_0-SSP, we show that the τ-SSP is NP-complete if $\tau = \{\mathsf{nop}, \mathsf{out}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$ or if $\{\mathsf{nop}, \mathsf{res}, \mathsf{set}\} \subseteq \tau$. By Lemma 2, the former also implies the NP-completeness of τ-SSP if $\tau = \{\mathsf{nop}, \mathsf{inp}, \mathsf{set}\} \cup \omega$ and $\omega \subseteq \{\mathsf{out}, \mathsf{used}, \mathsf{free}\}$. Altogether, this proves the claim for all the types listed in §1 and §3 of Fig. 1.

Secondly, we let $\tau_1 = \{\mathsf{nop}, \mathsf{inp}\}$ and reduce τ_1-SSP to τ-SSP, where $\tau = \{\mathsf{nop}, \mathsf{res}, \mathsf{swap}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$. Again by Lemma 2, this also implies the NP-completeness of τ-SSP if $\tau = \{\mathsf{nop}, \mathsf{set}, \mathsf{swap}\} \cup \omega$ and $\omega \subseteq \{\mathsf{out}, \mathsf{used}, \mathsf{free}\}$. Hence, this proves the claim for all the types listed in §2 of Fig. 1 and thus completes the proof of Theorem 2.

For the announced reductions, we use the following extensions of a TS A, cf. Fig. 6. Let $A = (S_A, E_A, \delta_A, \iota_A)$ be a loop-free TS, and let $\overline{E_A} = \{\overline{e} \mid e \in E_A\}$ be the set containing for every event $e \in E_A$ the unambiguous and fresh event $\overline{e}$ that is *associated with* e. *The backward-extension* $B = (S_A, E_A \cup \overline{E_A}, \delta_B, \iota_A)$ of A extends A by $\overline{E_A}$ and additional backward edges: for all $e \in E_A$ and all

$s, s' \in S_A$, if $\delta_A(s,e) = s'$, then $\delta_B(s,e) = s'$ and $\delta_B(s', \overline{e}) = s$. The *oneway loop-extension* $C = (S_A, E_A \cup \overline{E_A}, \delta_C, \iota_A)$ of a TS A extends B by some additional loops: for all $x \in E_A \cup \overline{E_A}$ and all $s \in S_A$, we define $\delta_C(s,e) = \delta_B(s,e)$ and, for all $e \in E_A$ and all $s, s' \in S_A$, if $\delta_A(s,e) = s'$, then $\delta_C(s',e) = s'$. Finally, the *loop-extension* $D = (S_A, E_A \cup \overline{E_A}, \delta_D, \iota_A)$ of A is an extension of C, where for all $x \in E_A \cup \overline{E_A}$ and all $s \in S_A$, we define $\delta_D(s,x) = \delta_C(s,x)$ and, for all $e \in E_A$ and all $s, s' \in S_A$, if $\delta_A(s,e) = s'$, then $\delta_D(s,\overline{e}) = s$.

Depending on the considered type τ, we let $\tilde{\tau} = \tau_0$ or $\tilde{\tau} = \tau_1$ and reduce a loop-free TS $A = (S_A, E_A, \delta_A, \iota_A)$ either to its backward-, oneway loop- or loop-extension and show that $sup : S_A \to \{0,1\}$ allows a $\tilde{\tau}$-region of A if and only if it allows a τ-region of the extension:

Lemma 5. *Let $\tau_0 = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$, $\tau_1 = \{\mathsf{nop}, \mathsf{inp}\}$, A a loop-free TS, $sup : S_A \to \{0,1\}$ and B, C and D the backward-, oneway loop- and loop-extension of A, respectively.*

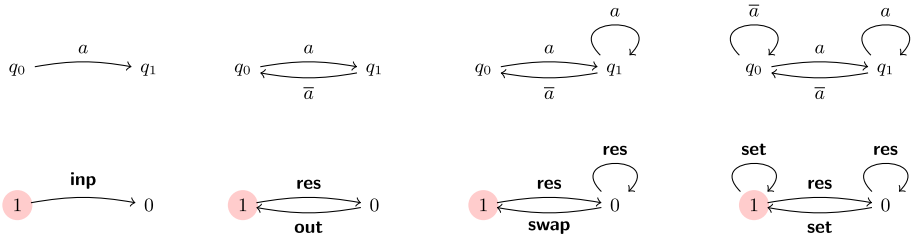


Fig. 6. Top, left to right: A TS A consisting of a single edge; backward-extension B of A; oneway loop-extension C of A; loop-extension D of A. Bottom, left to right: images of A and its extensions B, C, D under regions corresponding to the types of Lemma 5 solving (q_0, q_1): a $\{\mathsf{nop}, \mathsf{inp}\}$- ($\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$-) region of A; a $\{\mathsf{nop}, \mathsf{res}, \mathsf{out}\}$-region of B; a $\{\mathsf{nop}, \mathsf{res}, \mathsf{swap}\}$-region of C; a $\{\mathsf{nop}, \mathsf{res}, \mathsf{set}\}$-region of D.

1. *If $\tau = \{\mathsf{nop}, \mathsf{out}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, then sup allows a τ_0-region $R = (sup, sig)$ of A if and only if it allows a normalized τ-region $R' = (sup, sig')$ of B.*
2. *If $\tau \supseteq \{\mathsf{nop}, \mathsf{res}, \mathsf{set}\}$, then sup allows a τ_0-region $R = (sup, sig)$ of A if and only if it allows a normalized τ-region $R' = (sup, sig')$ of D.*
3. *If $\tau = \{\mathsf{nop}, \mathsf{res}, \mathsf{swap}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, then sup allows a τ_1-region $R = (sup, sig)$ of A if and only if it allows a normalized τ-region $R' = (sup, sig')$ of C.*

Proof. (1): *Only-if*: Let $R = (sup, sig)$ be a τ_0-region of A. Recall that $sig(e) = \mathsf{nop}$, $sig(e) = \mathsf{inp}$, $sig(e) = \mathsf{out}$ imply $sup(s) = sup(t)$, $sup(s) = 1$ and $sup(t) = 0$, $sup(s) = 0$ and $sup(t) = 1$ for all edges $s \xrightarrow{e} t$ of A, respectively. Thus, it is easy to see that R induces a normalized τ-region $R' = (sup, sig')$ of B as follows, cf. Fig. 6: For all $e \in E_A$ and its associated event $\overline{e} \in \overline{E_A}$, if $sig(e) = \mathsf{nop}$, then

$sig'(e) = sig'(\overline{e}) = \mathsf{nop}$; if $sig(e) = \mathsf{inp}$, then $sig'(e) = \mathsf{res}$ and $sig'(\overline{e}) = \mathsf{out}$; if $sig(e) = \mathsf{out}$, then $sig'(e) = \mathsf{out}$ and $sig'(\overline{e}) = \mathsf{res}$.

If: Let $R' = (sup, sig')$ be a normalized τ-region of B, and let $e \in E_A$ and $s \xrightarrow{e} t \in B$ and $s' \xrightarrow{e} t' \in B$ be arbitrary but fixed. First of all, we argue that if $sup(s) \neq sup(t)$, then $sup(s) = sup(s')$ and $sup(t) = sup(t')$: By definition of B, we have that $t \xrightarrow{\overline{e}} s$ and $t' \xrightarrow{\overline{e}} s'$ are present. If $sup(s) = 1$ and $sup(t) = 0$, then $s \xrightarrow{e} t$ and $t \xrightarrow{\overline{e}} s$ imply $sig'(e) \in \{\mathsf{inp}, \mathsf{res}\}$ and $sig'(\overline{e}) = \mathsf{out}$. By $\xrightarrow{e} t'$ and $\xrightarrow{\overline{e}} s'$, this immediately implies $sup(s') = 1$ and $sup(t') = 0$. Similarly, if $sup(s) = 0$ and $sup(t) = 1$, then $sig'(e) = \mathsf{out}$ and $sig'(\overline{e}) \in \{\mathsf{inp}, \mathsf{res}\}$, which implies $sup(s') = 0$ and $sup(t') = 1$. Consequently, since $s' \xrightarrow{e} t'$ was arbitrary, the claim follows. Note that this implies in return that if $sup(s) = sup(t)$, then $sup(s') = sup(t')$. Since both edges were arbitrary, it is easy to see that the following τ_0-region $R = (sup, sig)$ of A is well defined: for all $e \in E_A$, if there is $s \xrightarrow{e} t \in B$ such that $sup(s) \neq sup(t)$, then $sig(e) = \mathsf{inp}$ if $sup(s) = 1$ and $sup(t) = 0$, else $sig(e) = \mathsf{out}$; otherwise, $sig(e) = \mathsf{nop}$.

(2): *Only-if*: Recall that $0 \xrightarrow{\mathsf{res}} 0$ and $1 \xrightarrow{\mathsf{set}} 1$ are present in τ. Consequently, if $R = (sup, sig)$ is a τ_0-region of A, then the region $R' = (sup, sig')$, similarly defined to the one for the Only-if-direction of (1), but replacing out by set, is a τ-region of D, cf. Fig. 6.

If: If $R' = (sup, sig')$ is a normalized τ-region of D, then it holds $sig'(e) \in \{\mathsf{nop}, \mathsf{res}, \mathsf{set}\}$ for all $e \in E_D$. This is due to the fact that if $s \xrightarrow{x} s' \in D$, then $s' \xrightarrow{x} s' \in D$ for all $x \in E_D$ and all $s, t \in S_D$. Thus, $R = (sup, sig)$ is obtained from R' by the same arguments as the ones presented for the If-direction of (1).

(3): *Only-if*: Let $R = (sup, sig)$ be a τ_1-region of A. Recall that $sig(e) = \mathsf{nop}$ and $sig(e) = \mathsf{inp}$ imply $sup(s) = sup(t)$ and $sup(s) = 1, sup(t) = 0$ for all edges $s \xrightarrow{e} t$ of A, respectively. Moreover, $1 \xrightarrow{\mathsf{res}} 0$, $0 \xrightarrow{\mathsf{res}} 0$ and $0 \xrightarrow{\mathsf{swap}} 1$ are present in τ. Thus, we get a normalized τ-region $R' = (sup, sig')$ of C as follows, cf. Fig. 6: For all $e \in E_A$ and its associated event $\overline{e} \in \overline{E_A}$, if $sig(e) = \mathsf{nop}$, then $sig'(e) = sig'(\overline{e}) = \mathsf{nop}$; if $sig(e) = \mathsf{inp}$, then $sig'(e) = \mathsf{res}$ and $sig'(\overline{e}) = \mathsf{swap}$.

If: Let $R' = (sup, sig')$ be a normalized τ-region of C, and let $e \in E_A$ and $s \xrightarrow{e} t \in C$ and $s' \xrightarrow{e} t' \in C$ be arbitrary but fixed. We argue that $sup(s) \neq sup(t)$ implies $sup(s) = sup(s') = 1$ and $sup(t) = sup(t') = 0$: By definition of C and $s \xrightarrow{e} t \in C$, we get $t \xrightarrow{e} t \in C$. Thus, $sig'(e) \in \{\mathsf{nop}, \mathsf{res}\}$. Thus, if $sup(s) \neq sup(t)$, then $sig'(e) = \mathsf{res}$, which implies $sup(s) = 1$ and $sup(t) = 0$. Moreover, by $t \xrightarrow{\overline{e}} s \in C$, this also implies $sig'(\overline{e}) = \mathsf{swap}$. Finally, by $sig'(e) = \mathsf{res}$, $\xrightarrow{e} t'$, $sig'(\overline{e}) = \mathsf{swap}$ and $t' \xrightarrow{\overline{e}} s'$, we get $sup(t') = 0$ and $sup(s') = 1$. Consequently, the following definition of sig yields a well-defined τ_1-region $R = (sup, sig)$ of A: for all $e \in E_A$, if $sig'(e) = \mathsf{res}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{nop}$. □

Notice that the TS A_φ of Sect. 3.1 is loop-free. Furthermore, in [24], it has been shown that $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$-SSP is NP-complete even if A is a simple directed path. Moreover, the introduced extensions of A are constructible in polynomial

time. Thus, by Lemma 1 and Lemma 2, the following corollary, which is easily implied by Lemma 5, completes the proof of Theorem 2.

Corollary 1 (Without Proof). *Let $\tau_0 = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$ and $\tau_1 = \{\mathsf{nop}, \mathsf{inp}\}$, and let A be a loop-free TS and B, C and D its backward-, oneway loop- and loop-extension, respectively.*

1. *If $\tau = \{\mathsf{nop}, \mathsf{out}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, then A has the τ_0-SSP if and only if B has the τ-SSP.*
2. *If $\tau \supseteq \{\mathsf{nop}, \mathsf{res}, \mathsf{set}\}$, then A has the τ_0-SSP if and only if D has the τ-SSP.*
3. *If $\tau = \{\mathsf{nop}, \mathsf{res}, \mathsf{swap}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{used}, \mathsf{free}\}$, then A has the τ_1-SSP if and only if C has the τ-SSP.*

4 Deciding the State Separation Property for nop-free Types

The following theorem summarizes the complexity of τ-SSP for nop-free Boolean types:

Theorem 3. *Let τ be a nop-free type of nets and A a TS.*

1. *If $\mathsf{swap} \notin \tau$ or $\mathsf{swap} \in \tau$ and $\tau \cap \mathfrak{save} = \emptyset$, then deciding if A has the τ-SSP is polynomial.*
2. *If $\mathsf{swap} \in \tau$ and $\tau \cap \mathfrak{save} \neq \emptyset$, then deciding if A has the τ-SSP is NP-complete.*

The tractability of τ-SSP for nop-free types that are also swap-free has been shown in the broader context of τ-synthesis in [22]. Thus, restricted to Theorem 3.1, it remains to argue that τ-SSP is polynomial if $\tau = \{\mathsf{swap}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}\}$. The following lemma states that separable inputs of τ-SSP are trivial for these types and thus proves its tractability:

Lemma 6. *Let $\tau = \{\mathsf{swap}\} \cup \omega$, where $\omega \subseteq \{\mathsf{inp}, \mathsf{out}\}$, and $A = (S, E, \delta, \iota)$ be a TS. If A has the τ-SSP, then it has at most two states.*

Proof. If there is $s \xrightarrow{e} s \in A$, then A has no τ-regions. Hence, if such A has more than one state, then it does not have the τ-SSP.

Let's consider the case when A is loop-free, that is, $s \xrightarrow{e} s' \in A$ implies $s \neq s'$. First of all, note that there is at most one outgoing edge $\iota \xrightarrow{e} s$ at ι, since if $\iota \xrightarrow{e} s$, $\iota \xrightarrow{e'} s'$ and $s \neq s'$, then s and s' are not separable. This can be seen as follows: If $R = (sup, sig)$ is a τ-region that solves (s, s'), then $sup(s) = 0$ and $sup(s') = 1$ or $sup(s) = 1$ and $sup(s') = 0$. If $sup(s) = 0$ and $sup(s') = 1$, then $sup(\iota) = 0$ contradicts $sig(e) \in \tau$, and $sup(\iota) = 1$ contradicts $sig(e') \in \tau$. Similarly, $sup(s) = 1$ and $sup(s') = 0$ yields a contradiction. Thus, a separating region R does not exist, which proves the claim. Secondly, if $\iota \xrightarrow{e} s \xrightarrow{e'} s'' \in A$, then $\iota = s''$, since otherwise, ι and s'' are not separable. This can be seen as follows: If $R = (sup, sig)$ is a τ-region that solves (ι, s''), then $sup(\iota) = 0$ and

$sup(s'') = 1$ or $sup(\iota) = 1$ and $sup(s'') = 0$. If $sup(\iota) = 0$ and $sup(s'') = 1$, then $sup(s') = 0$ contradicts $sig(e) \in \tau$, and $sup(s') = 1$ contradicts $sig(e') \in \tau$. Similarly, $sup(\iota) = 1$ and $sup(s'') = 0$ yields a contradiction. This implies again that a separating region does not exist, which proves the claim and, moreover, proves the lemma. □

To complete the proof of Theorem 3, it remains to prove the NP-completeness of τ-SSP for the types listed in §9 of Fig. 1, which are exactly covered by Theorem Theorem 3.2. Thus, in the remainder of this section, if not stated explicitly otherwise, we let τ be a nop-free type such that $\mathsf{swap} \in \tau$ and $\tau \cap \mathfrak{save} \neq \emptyset$. Moreover, we reduce CM 1-IN-3 3SAT to τ-SSP again.

Basic Ideas of the Reduction. Similar to our previous approach we build a TS A_φ that has for every clause $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$, where $i \in \{0, \ldots, m-1\}$, a (bi-directed) path $P_i = \ldots \xleftrightarrow{X_{i_0}} \ldots \xleftrightarrow{X_{i_1}} \ldots \xleftrightarrow{X_{i_2}} \ldots$ on which the elements of ζ_i occur as events. Together, the corresponding paths $P_0, \ldots, P_{m-1}$ are meant to represent φ. However, the current types are more diverse than $\{\mathsf{nop}, \mathsf{inp}\}$, since they also allow swap and other interactions. Simultaneously, they are more restricted than $\{\mathsf{nop}, \mathsf{inp}\}$, since they lack of nop. One of the main obstacles that occur is that if $s \xrightarrow{e} s' \in A_\varphi$, then the current types basically allow $sup(s) = 1$ and $sup(s') = 0$ as well as $sup(s) = 0$ and $sup(s') = 1$ for a τ-region $R = (sup, sig)$ that solves (s, s'). It turns out that this requires a second representation of φ. To do so, we use a copy φ' that originates from φ by simply renaming its variables. That is, φ' originates from φ by replacing every variable $v \in X$ of φ by a unique and fresh variable v'.

Example 2 (Renaming of φ). The instance $\varphi' = \{\zeta'_0, \ldots, \zeta'_5\}$ that originates from φ of Example 1 is defined by $X' = \{X'_0, \ldots, X'_5\}$ and $\zeta'_0 = \{X'_0, X'_1, X'_2\}$, $\zeta'_1 = \{X'_0, X'_2, X'_3\}$, $\zeta'_2 = \{X'_0, X'_1, X'_3\}$, $\zeta'_3 = \{X'_2, X'_4, X'_5\}$, $\zeta'_4 = \{X'_1, X'_4, X'_5\}$ and $\zeta'_5 = \{X'_3, X'_4, X'_5\}$.

It is immediately clear that φ is one-in-three satisfiable if and only if φ' is one-in-three satisfiable. The TS A_φ additionally has for every clause $\zeta'_i = \{X'_{i_0}, X'_{i_1}, X'_{i_2}\}$, where $i \in \{0, \ldots, m-1\}$, of φ' also a (bi-directed) path $P'_i = \ldots \xleftrightarrow{X'_{i_0}} \ldots \xleftrightarrow{X'_{i_1}} \ldots \xleftrightarrow{X'_{i_2}} \ldots$ on which the elements of ζ'_i occur as events. Moreover, by the construction, the TS A_φ has a SSP atom $\alpha = (s, s')$ such that if a τ-region solves α, then either the signatures of the variable events X of φ define a one-in-three model of φ or the signatures of the variable events X' of φ' define a one-in-three model of φ'. Obviously, both cases imply the one-in-three satisfiability of φ.

Conversely, the construction ensures, if φ has a one-in-three model then A_φ has the τ-SSP.

Similar to our approach for Theorem 1, the TS A_φ is a composition of several gadgets. The next Lemma 7 introduces some basic properties of τ-regions in bi-directed TS for nop-free types that we use to prove the functionailty of A_φ's gadgets. After that, Lemma 8 introduces TS that are the (isomorphic) prototypes

of the gadgets of A_φ and additionally proves the essential parts of their intended functionality.

Lemma 7. *Let τ be a* nop*-free Boolean type of nets and A a bi-directed TS; let $s \xrightarrow{e} s'$ be an edge of A, $P_0 = s_0 \xleftrightarrow{e_1} \dots \xleftrightarrow{e_m} s_m$ and $P_1 = q_0 \xleftrightarrow{e_1} \dots \xleftrightarrow{e_m} q_m$ be two simple paths of A that both apply the same sequence $e_1 \dots e_m$ of events, and $R = (sup, sig)$ be a τ-region of A.*

1. *If $sig(e) \in$ save, then $sup(s) = sup(s') = sup(q) = sup(q')$ for every edge $q \xrightarrow{e} q' \in A$.*
2. *If $sup(s_m) \neq sup(q_m)$, then $sig(e_i) =$ swap for all $i \in \{1, \dots, m\}$.*
3. *If $sup(s_0) = sup(s_m)$, then $|\{e \in \{e_1, \dots, e_m\} \mid sig(e) =$ swap$\}|$ is even.*

Proof. (1): A is bi-directed and $\xrightarrow{i} p \in \tau$ and $\xrightarrow{i} p' \in \tau$ imply $p = p'$ for all $i \in$ save.

(2): By definition of τ, if $sup(s_m) \neq sup(q_m)$, then, by $\xrightarrow{e_m} s_m$ and $\xrightarrow{e_m} q_m$, we get $sig(e_m) =$ swap. Clearly, by $sup(s_m) \neq sup(q_m)$, this implies $sup(s_{m-1}) \neq sup(q_{m-1})$. Thus, the claim follows easily by induction on m.

(3): Since $sup(s_0) = sup(s_m)$, the image P_0^R of P_0 is a path of τ that starts and terminates at the same state. Consequently, the number of changes between 0 and 1 on P_0^R is even. Since A is bi-directed, $sup(s) \neq sup(s')$ if and only if $sig(e) =$ swap for all $s \xrightarrow{e} s' \in P_0$. Thus, the number of events of P_0 with a swap-signature must be even. Hence, the claim. □

Lemma 8 (Basic Components of A_φ). *Let τ be a* nop*-free Boolean type and A a bi-directed TS with the following paths G, F, T and Q, and let $R = (sup, sig)$ be a τ-region of A:*

$$G = g_0 \xleftrightarrow{v} g_1 \xleftrightarrow{w} g_2 \xleftrightarrow{k_0} g_3 \xleftrightarrow{k_1} g_4 \qquad F = f_0 \xleftrightarrow{v} f_1 \xleftrightarrow{w} f_2 \xleftrightarrow{k_1} f_3 \xleftrightarrow{k_0} f_4$$

$$T = t_0 \xleftrightarrow{k_0} t_1 \xleftrightarrow{v_0} t_2 \xleftrightarrow{v_1} t_3 \xleftrightarrow{X_0} t_4 \xleftrightarrow{v_2} t_5 \xleftrightarrow{X_1} t_6 \xleftrightarrow{v_3} t_7 \xleftrightarrow{X_2} t_8 \xleftrightarrow{v_4} t_9 \xleftrightarrow{v_5} t_{10} \xleftrightarrow{k_0} t_{11}$$

$$Q = q_0 \xleftrightarrow{X_0} q_1 \xleftrightarrow{v_6} q_2 \xleftrightarrow{X_2} q_3$$

1. *If R solves $\alpha = (g_2, g_4)$, then either $sig(k_0) \in$ save and $sig(k_1) =$ swap or $sig(k_0) =$ swap and $sig(k_1) \in$ save;*
2. *If $sig(k_0) \in$ save and $sig(k_1) =$ swap or $sig(k_0) =$ swap and $sig(k_1) \in$ save, then $sig(v) = sig(w) =$ swap;*
3. *If $sig(k_0) \in$ save and $sig(v_i) =$ swap for all $i \in \{0, \dots, 6\}$, then there is exactly one $X \in \{X_0, X_1, X_2\}$ such that $sig(X) \neq$ swap.*

Proof. Since A is bi-directed, we have $sig(e) \notin \{\mathsf{inp}, \mathsf{out}\}$ for all $e \in E_A$.

(1): R solves α, thus $sup(g_2) \neq sup(g_4)$. If $sig(k_0) \neq$ swap $\neq sig(k_1)$ or $sig(k_0) = sig(k_1) =$ swap, then $sup(g_2) = sup(g_4)$, a contradiction. Hence the claim, cf. Fig. 7.

(2): If $sig(k_0) \in \mathfrak{save}$ and $sig(k_1) = \mathsf{swap}$, then we have $sup(g_2) = sup(f_3) \neq sup(f_2)$, cf. Fig. 7. By symmetry, the latter is also true if $sig(k_0) = \mathsf{swap}$ and $sig(k_1) \in \mathfrak{save}$. Hence, the claim follows from Lemma 7.2.

(3): Since $sig(k_0) \in \mathfrak{save}$, we get $sup(t_1) = sup(t_{10})$. By Lemma 7.3, this implies that $|\{e \in E(T) \mid sig(e) = \mathsf{swap}\}|$ is even. Moreover, since $sig(v_0) = \ldots sig(v_5) = \mathsf{swap}$, this implies $|\{e \in \{X_0, X_1, X_2\} \mid sig(e) = \mathsf{swap}\}| \in \{0, 2\}$. If $|\{e \in \{X_0, X_1, X_2\} \mid sig(e) = \mathsf{swap}\}| = 0$ then, we get $sup(t_3) = sup(t_4) \neq sup(t_5) = sup(t_6) \neq sup(t_7) = sup(t_8)$ by Lemma 7.1. This particularly implies $sup(t_4) = sup(t_7)$ and, again by Lemma 7.1, also $sup(t_4) = sup(q_1) = sup(q_2)$ and contradicts $sig(v_6) = \mathsf{swap}$, cf. Fig. 8. Thus, we have $|\{e \in \{X_0, X_1, X_2\} \mid sig(e) = \mathsf{swap}\}| = 2$, which proves the claim, cf. Fig. 7. □

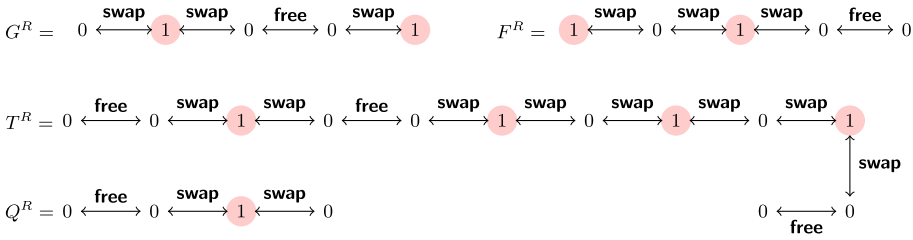


Fig. 7. Illustrations for Lemma 8. The images G^R, F^R, T^R and Q^R, where $R = (sup, sig)$ is a $\{\mathsf{swap}, \mathsf{free}\}$-region that solves (g_2, g_4) and satisfies $sig(k_0) = sig(X_0) = \mathsf{free}$ and $sig(k_1) = sig(v) = sig(w) = sig(X_1) = sig(X_2) = sig(v_i) = \mathsf{swap}$ for all $i \in \{0, \ldots, 6\}$.

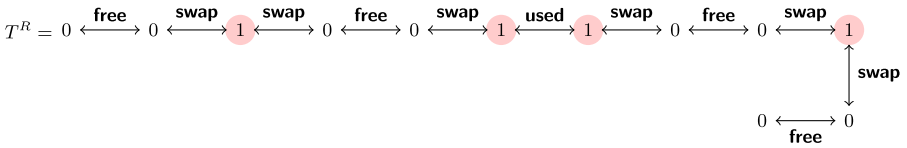


Fig. 8. Illustration for Lemma 8. A $\{\mathsf{swap}, \mathsf{used}, \mathsf{free}\}$-region $R = (sup, sig)$, restricted to T, where $sig(k_0) = sig(X_0) = sig(X_2) = \mathsf{free}$, $sig(X_1) = \mathsf{used}$ and $sig(v_i) = \mathsf{swap}$ for all $i \in \{0, \ldots, 5\}$ and, hence, $|\{e \in \{X_0, X_1, X_2\} \mid sig(e) = \mathsf{swap}\}| = 0$. R is not extendable to a region of a TS that has T and Q such that $sig(v_6) = \mathsf{swap}$, since $sig(v_6) = \mathsf{swap}$ would contradict $sup(q_0) = \cdots = sup(q_3) = 0$, which would be required by $sig(X_0) = sig(X_2) = \mathsf{free}$.

Let φ be an instance of CM 1-IN-3 3SAT with the set of variables $X = \{X_0, \ldots, X_{m-1}\}$ and φ' its renamed copy with event set $X' = \{X'_0, \ldots, X'_{m-1}\}$. In the following, we introduce the construction of A_φ.

Firstly, for every $i \in \{0, \ldots, 7m-1\}$, the TS A_i has the following gadgets G_i, F_i, G'_i and F'_i with starting states $g_{i,0}$, $f_{i,0}$, $g'_{i,0}$ and $f'_{i,0}$, respectively, providing the atom $\alpha = (g_{0,2}, g_{0,4})$:

$$G_i = g_{i,0} \xleftrightarrow{v_i} g_{i,1} \xleftrightarrow{w_i} g_{i,2} \xleftrightarrow{k_0} g_{i,3} \xleftrightarrow{k_1} g_{i,4} \qquad F_i = f_{i,0} \xleftrightarrow{v_i} f_{i,1} \xleftrightarrow{w_i} f_{i,2} \xleftrightarrow{k_1} f_{i,3} \xleftrightarrow{k_0} f_{i,4}$$

$$G'_i = g'_{i,0} \xleftrightarrow{v'_i} g'_{i,1} \xleftrightarrow{w'_i} g'_{i,2} \xleftrightarrow{k_1} g'_{i,3} \xleftrightarrow{k_0} g'_{i,4} \qquad F'_i = f'_{i,0} \xleftrightarrow{v'_i} f'_{i,1} \xleftrightarrow{w'_i} f'_{i,2} \xleftrightarrow{k_0} f'_{i,3} \xleftrightarrow{k_1} f'_{i,4}$$

Secondly, for every $i \in \{0, \dots, m-1\}$, the TS A_φ has the following gadgets $T_{i,0}, T_{i,1}$ and $T'_{i,0}, T'_{i,1}$ that use the elements of $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$ and $\zeta'_i = \{X'_{i_0}, X'_{i_1}, X'_{i_2}\}$ as events, respectively; their starting states are $t_{i,0,0}$, $t_{i,1,0}$, $t'_{i,0,0}$ and $t'_{i,1,0}$:

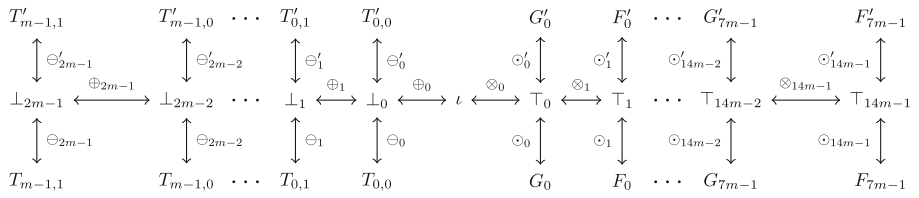
$$T_{i,0} = t_{i,0,0} \xleftrightarrow{k_0} t_{i,0,1} \xleftrightarrow{v_{7i}} t_{i,0,2} \xleftrightarrow{v_{7i+1}} t_{i,0,3} \xleftrightarrow{X_{i_0}} t_{i,0,4} \xleftrightarrow{v_{7i+2}} t_{i,0,5} \xleftrightarrow{X_{i_1}} t_{i,0,6} \xleftrightarrow{v_{7i+3}} t_{i,0,7} \xleftrightarrow{X_{i_2}} t_{i,0,8} \xleftrightarrow{v_{7i+4}} t_{i,0,9} \xleftrightarrow{v_{7i+5}} t_{i,0,10} \xleftrightarrow{k_0} t_{i,0,11}$$

$$T_{i,1} = t_{i,1,0} \xleftrightarrow{X_{i_0}} t_{i,1,1} \xleftrightarrow{v_{7i+6}} t_{i,1,2} \xleftrightarrow{X_{i_2}} t_{i,1,3}$$

$$T'_{i,0} = t'_{i,0,0} \xleftrightarrow{k_1} t'_{i,0,1} \xleftrightarrow{v'_{7i}} t'_{i,0,2} \xleftrightarrow{v'_{7i+1}} t'_{i,0,3} \xleftrightarrow{X'_{i_0}} t'_{i,0,4} \xleftrightarrow{v'_{7i+2}} t'_{i,0,5} \xleftrightarrow{X'_{i_1}} t'_{i,0,6} \xleftrightarrow{v'_{7i+3}} t'_{i,0,7} \xleftrightarrow{X'_{i_2}} t'_{i,0,8} \xleftrightarrow{v'_{7i+4}} t'_{i,0,9} \xleftrightarrow{v'_{7i+5}} t'_{i,0,10} \xleftrightarrow{k_1} t'_{i,0,11}$$

$$T'_{i,1} = t'_{i,1,0} \xleftrightarrow{X'_{i_0}} t'_{i,1,1} \xleftrightarrow{v'_{7i+6}} t'_{i,1,2} \xleftrightarrow{X'_{i_2}} t'_{i,1,3}$$

Finally, the gadgets are connected via their starting states to finally build A_φ as follows:



The following Lemma 9 and Lemma 10 prove the validity of our polynomial-time reduction and, hence, complete the proof of Theorem 3.

Lemma 9. *If A_φ has the τ-SSP, then φ is one-in-three satisfiable.*

Proof. Let G, F, T and Q be the paths defined in Lemma 8. First of all, we observe that $G \cong G_{7i+j} \cong G'_{7i+j}$ and $F \cong F_{7i+j} \cong F'_{7i+j}$ and $T \cong T_{i,0} \cong T'_{i,0}$ and $Q \cong T_{i,1} \cong T'_{i,1}$ for all $i \in \{0, \dots, m-1\}$ and $j \in \{0, \dots, 6\}$. Let $R = (sup, sig)$ be a τ-region that solves $\alpha = (g_{0,2}, g_{0,4})$ (which exists, since A_φ has the τ-SSP) and let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. By Lemma 8.1, we have either $sig(k_0) \in \mathfrak{save}$ and $sig(k_1) = \mathsf{swap}$ or $sig(k_0) = \mathsf{swap}$ and $sig(k_1) \in \mathfrak{save}$. This implies $sig(u_{7i}) = \dots = sig(u_{7i+6}) = sig(u'_{7i}) = \dots = sig(u'_{7i+6}) = \mathsf{swap}$ by Lemma 8.2. If $sig(k_0) \in \mathfrak{save}$ and $sig(k_1) = \mathsf{swap}$, then by Lemma 8.3, this implies that there is exactly one event $e \in \{X_{i_0}, X_{i_1}, X_{i_2}\}$ such that $sig(e) \neq \mathsf{swap}$. Consequently, since i was arbitrary, if $sig(k_0) \in \mathfrak{save}$ and $sig(k_1) = \mathsf{swap}$,

then $M = \{e \in X \mid sig(e) \neq \mathsf{swap}\}$ selects exactly one variable of every clause ζ_i for all $i \in \{0, \ldots, m-1\}$. Thus, M is a one in three-model of φ. Otherwise, if $sig(k_0) = \mathsf{swap}$ and $sig(k_1) \in \mathfrak{save}$, then we similarly obtain that $M' = \{e \in X' \mid sig(e) \neq \mathsf{swap}\}$ defines a one-in-three-model of φ', which also implies the one-in-three satisfiability of φ. □

Lemma 10. *If φ is one-in-three satisfiable, then A_φ has the τ-SSP.*

Due to space restrictions, the proof of Lemma 10 is omitted but can be found in the technical report that corresponds to this paper [21].

5 Conclusion

In this paper, we present the overall characterization of the computational complexity of the problem τ-SSP for all 256 Boolean types of nets τ. Our presentation includes 154 new complexity results (Fig. 1: §1–§3, §6, §9, §10) and 102 known results (Fig. 1: §4 [23], §5 [19,24], §7 [16,23], §8 [22]) and classifies them in the overall context of boolean state separation. Besides the new 150 hardness - and 4 tractability-results, this classification is one of the main contributions of this paper. First of all, it becomes apparent that the distinction between nop-free and nop-equipped types is meaningful: Within the class of nop-free types, τ-SSP turns out to be NP-complete if and only if $\mathsf{swap} \in \tau$ and $\tau \cap \mathfrak{save} \neq \emptyset$. Within the class of nop-equipped types, a differentiation between types τ that satisfy $\tau \cap \{\mathsf{res}, \mathsf{set}\} = \emptyset$ and the ones with $\tau \cap \{\mathsf{res}, \mathsf{set}\} \neq \emptyset$ is useful: τ-SSP for the former ones is NP-complete if and only if $\tau \cap \{\mathsf{inp}, \mathsf{out}\} \neq \emptyset$ and $\mathsf{swap} \notin \tau$. In particular, $\{\mathsf{swap}\} \cup \tau$-SSP becomes polynomial for all these types. On the other hand, for the latter ones, which include $i \in \{\mathsf{res}, \mathsf{set}\}$ such that $i \in \tau$, τ-SSP is NP-complete as long as there is also an interaction in τ that opposes i, that is, $\tau \cap \mathfrak{exit} \neq \emptyset$ if $i = \mathsf{set}$ and $\tau \cap \mathfrak{enter} \neq \emptyset$ if $i = \mathsf{res}$.

Moreover, our proofs discover that, up to isomorphism, there are essentially four hard kernels that indicate the NP-completeness of τ-SSP, namely $\{\mathsf{nop}, \mathsf{inp}\}$ and $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$ for the nop-equipped types and $\{\mathsf{swap}, \mathsf{free}\}$ and $\{\mathsf{swap}, \mathsf{free}, \mathsf{used}\}$ for the nop-free types. That means, for a nop-equipped type τ, the hardness of τ-SSP can either be shown by a reduction of $\{\mathsf{nop}, \mathsf{inp}\}$-SSP or $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}\}$-SSP (or their isomorphic types), which is basically done in the proof of Lemma 5, or τ-SSP is polynomial otherwise. Similarly, one finds out that for the nop-free types τ in question, the hardness of τ-SSP can be shown by a reduction of $\{\mathsf{swap}, \mathsf{free}\}$-SSP or $\{\mathsf{swap}, \mathsf{free}, \mathsf{used}\}$-SSP. Due to the space limitation, a reduction that covers *all* hard nop-free types on one blow is given instead of explicit proof.

For future work, it remains to completely characterize the computational complexity of deciding if a TS A is isomorphic to the reachability graph of a Boolean Petri net, instead of only being embeddable by an injective simulation map.

References

1. Badouel, E., Bernardinello, L., Darondeau, P.: Polynomial algorithms for the synthesis of bounded nets. In: Mosses, P.D., Nielsen, M., Schwartzbach, M.I. (eds.) CAAP 1995. LNCS, vol. 915, pp. 364–378. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59293-8_207
2. Badouel, E., Bernardinello, L., Darondeau, P.: The synthesis problem for elementary net systems is NP-complete. Theor. Comput. Sci. **186**(1–2), 107–134 (1997). https://doi.org/10.1016/S0304-3975(96)00219-8
3. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. TTCSAES. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47967-4
4. Badouel, E., Darondeau, P.: Trace nets and process automata. Acta Inf. **32**(7), 647–679 (1995). https://doi.org/10.1007/BF01186645
5. Best, E., Devillers, R.R.: Pre-synthesis of Petri nets based on prime cycles and distance paths. Sci. Comput. Program. **157**, 41–55 (2018). https://doi.org/10.1016/j.scico.2017.07.005
6. Chatain, T., Haar, S., Kolcák, J., Paulevé, L., Thakkar, A.: Concurrency in Boolean networks. Nat. Comput. **19**(1), 91–109 (2020). https://doi.org/10.1007/s11047-019-09748-4
7. Esparza, J., Nielsen, M.: Decidability issues for Petri nets - a survey. Bull. EATCS **52**, 244–262 (1994)
8. Kleijn, J., Koutny, M., Pietkiewicz-Koutny, M., Rozenberg, G.: Step semantics of Boolean nets. Acta Inf. **50**(1), 15–39 (2013). https://doi.org/10.1007/s00236-012-0170-2
9. Montanari, U., Rossi, F.: Contextual occurrence nets and concurrent constraint programming. In: Schneider, H.J., Ehrig, H. (eds.) Graph Transformations in Computer Science. LNCS, vol. 776, pp. 280–295. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57787-4_18
10. Montanari, U., Rossi, F.: Contextual nets. Acta Inf. **32**(6), 545–596 (1995). https://doi.org/10.1007/BF01178907
11. Moore, C., Robson, J.M.: Hard tiling problems with simple tiles. Discret. Comput. Geom. **26**(4), 573–590 (2001). https://doi.org/10.1007/s00454-001-0047-6
12. Pietkiewicz-Koutny, M.: transition systems of elementary net systems with inhibitor arcs. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 310–327. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63139-9_43
13. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) ACPN 1996. LNCS, vol. 1491, pp. 12–121. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-65306-6_14
14. Schlachter, U.: Over-approximative Petri net synthesis for restricted subclasses of nets. In: Klein, S.T., Martín-Vide, C., Shapira, D. (eds.) LATA 2018. LNCS, vol. 10792, pp. 296–307. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77313-1_23
15. Schlachter, U., Wimmel, H.: Optimal label splitting for embedding an LTS into an arbitrary Petri net reachability graph is NP-complete. CoRR abs/2002.04841 (2020). https://arxiv.org/abs/2002.04841
16. Schmitt, V.: Flip-flop nets. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 515–528. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60922-9_42

17. Tredup, R.: The complexity of synthesizing nop-equipped Boolean nets from g-bounded inputs (technical report) (2019)
18. Tredup, R.: Fixed parameter tractability and polynomial time results for the synthesis of b-bounded Petri nets. In: Donatelli, S., Haar, S. (eds.) PETRI NETS 2019. LNCS, vol. 11522, pp. 148–168. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21571-2_10
19. Tredup, R.: Hardness results for the synthesis of b-bounded Petri nets. In: Donatelli, S., Haar, S. (eds.) PETRI NETS 2019. LNCS, vol. 11522, pp. 127–147. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21571-2_9
20. Tredup, R.: Parameterized complexity of synthesizing b-bounded (m, n)-T-systems. In: Chatzigeorgiou, A., et al. (eds.) SOFSEM 2020. LNCS, vol. 12011, pp. 223–235. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38919-2_19
21. Tredup, R., Erofeev, E.: The complexity of Boolean state separation (technical report) (2020). submitted to arxive.org
22. Tredup, R., Erofeev, E.: On the complexity of synthesis of nop-free Boolean Petri nets. In: van der Aalst, W.M.P., Bergenthum, R., Carmona, J. (eds.) Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2020 Satellite event of the 41st International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2020, Virtual Workshop, June 24, 2020. CEUR Workshop Proceedings, vol. 2625, pp. 66–84. CEUR-WS.org (2020). http://ceur-ws.org/Vol-2625/paper-05.pdf
23. Tredup, R., Rosenke, C.: The complexity of synthesis for 43 boolean petri net types. In: Gopal, T.V., Watada, J. (eds.) TAMC 2019. LNCS, vol. 11436, pp. 615–634. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14812-6_38
24. Tredup, R., Rosenke, C., Wolf, K.: Elementary net synthesis remains NP-complete even for extremely simple inputs. In: Khomenko, V., Roux, O.H. (eds.) PETRI NETS 2018. LNCS, vol. 10877, pp. 40–59. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91268-4_3
25. Vogler, W., Semenov, A., Yakovlev, A.: Unfolding and finite prefix for nets with read arcs. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 501–516. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055644
26. Wimmel, H.: Presynthesis of bounded choice-free or fork-attribution nets. Inf. Comput. **271**, 104482 (2020)
27. Yakovlev, A., Koelmans, A., Semenov, A.L., Kinniment, D.J.: Modelling, analysis and synthesis of asynchronous control circuits using Petri nets. Integration **21**(3), 143–170 (1996). https://doi.org/10.1016/S0167-9260(96)00010-7