On the Complexity of Synthesis of nop-Free Boolean Petri Nets

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

[1] Universität Rostock, Institut für Informatik, Theoretische Informatik, Albert-Einstein-Straße 22, 18059, Rostock (`ronny.tredup@uni-rostock.de`)
[2] Department of Computing Science, Carl von Ossietzky Universität Oldenburg, D-26111 Oldenburg, Germany (`evgeny.erofeev@informatik.uni-oldenburg.de`)

Abstract. In a Boolean Petri net, the interaction nop allows places and transitions to be independent, so that the firing of a transition does not affect the marking of a place, and vice versa. Recently, the complexity of synthesis of nets equipped with nop has been investigated thoroughly, while the question for the rest 128 types of nets remains open. This work tackles the case of nop-free nets synthesis, that is, the Boolean nets where places and transitions are always related via interactions that are able to modify the marking of a place. In this paper, we show that, for nop-free nets, the absence of swap leads always to a polynomial time synthesis procedure. Moreover, we give a first hint, that the presence of swap might make the synthesis for these types NP-complete.

Keywords: Boolean Petri Net, Synthesis, transition systems, time complexity

1 Introduction

Boolean Petri nets is a well-known formalism for modeling and analysing of concurrent and distributed systems. This class of Petri nets assumes each place to be a Boolean condition which is *true* if the place is marked, i.e. contains a token, and *false* otherwise. Hence, the places can be treated as indicators for properties of the modeled system. The interconnections between places and transitions of a Boolean net are given through the Boolean *interactions*. These interactions are partial binary functions from $\{0,1\}$ to $\{0,1\}$, and the precise set of the interactions which are employed in a given net determines the *Boolean type* of the net.

By means of the notion of type of nets, Boolean Petri nets provide a uniform approach for defining of many classes of nets, e.g. event/condition nets [1], elementary net systems [1], SET nets [3], which allows researchers to comprise models of different kinds in a generic manner [1]. Eight binary functions are applied as interactions between places and transitions in Boolean nets: *no operation* (nop), *input* (inp), *output* (out), *unconditionally set to true* (set), *unconditionally reset*

* Supported by DFG through grant Be 1267/16-1 ASYST.

to false (res), *inverting* (swap), *test if true* (used), and *test if false* (free). These interactions define in which way a place p and a transition t influence each other: The interaction inp (out) defines that p must be *true* (*false*) before and *false* (*true*) after the firing of t; free (used) implies that the firing of t 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 the firing of t it is *false* (*true*); swap means that t inverts the current Boolean value of p. With these eight interactions, 256 different Boolean types of nets are possible in total.

In the previous works [9–11], the complexity of Boolean Petri net synthesis has been investigated for the types which allow the interaction nop. Among others, these types include some well-known classes, e.g. contextual nets $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$ [4], inhibitor nets $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{free}\}$ [6], trace nets $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{set}, \mathsf{res}, \mathsf{used}, \mathsf{free}\}$ [2], flip-flop nets $\{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{swap}\}$ [7]. Besides, for the types with nop, the complexity of synthesis has been studied in a special setting with restrictions on the synthesis input [8], which has some practical applications in hardware design. At the same time, the complexity of synthesis for 128 Boolean types which do not allow a place and a transition to be independent, i.e. the *nop-free* types of nets, remains an uncharted research area. This paper aims at compensating of this skewness and at filling the gap in theory. Hence, the paper focusses on the synthesis of nop-free types, and makes an effort to reach the complete characterization for the complexity of Boolean net synthesis. The limitations of the target net class imply a structural property of admissible behaviors, which we call the *single sink property.* The property requires all the edges that carry the same label, to have a unique common target state. Along the consideration, we especially distinguish between nop-free types which allow the interaction swap, and the ones which do not. In comparison to the other Boolean interactions, nop and swap are the only two of eight which on the one hand do not have a symmetrical dual interaction, and on the other hand, have 1 and 0 both as their input and their output. As we shall prove, this peculiarity indeed makes a big difference: while the synthesis of nop-free types without swap will be proved polynomial, we present a first hint that synthesis might be NP-complete at least for some nop-free types which admit swap.

The paper is organized as follows. After introducing of the necessary definitions in Section 2, the main results will be presented in Section 3. We begin with the polynomial cases in Section 3 and then continue with the hardness result in Section 4. Section 5 summarises the work and suggests an outlook of possible directions of the further research.

2 Preliminaries

Transition Systems. A (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* $\delta : S \times E \longrightarrow S$, where $\delta(s, e) = s'$ is interpreted as $s \xrightarrow{e} s'$. For $s \xrightarrow{e} s'$, we say that s is a *source* and s' is a *sink* of e, respectively. If $\delta(s, e)$ is defined, we say that event e *occurs* at state s, denoted by $s\xrightarrow{e}$. An *initialized* TS $A = (S, E, \delta, s_0)$ is a TS with a distinct *initial* state $s_0 \in S$ where every state $s \in S$ is *reachable* from s_0 by a directed labeled path.

Boolean Types of Nets [1]. 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*: $E_\tau \subseteq I = \{\mathsf{nop}, \mathsf{inp}, \mathsf{out}, \mathsf{set}, \mathsf{res}, \mathsf{swap}, \mathsf{used}, \mathsf{free}\}$. Each interaction $i \in I$ is a binary partial function $i : \{0,1\} \to \{0,1\}$ as defined in Figure 1. For all $x \in \{0,1\}$ and all $i \in E_\tau$, the transition function of τ is defined by $\delta_\tau(x, i) = i(x)$. Since a type τ is completely determined by E_τ, we often identify τ with E_τ, cf. Figure 2.

τ-Nets. Let $\tau \subseteq I$. A Boolean Petri 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 a transition system A_N, called the *reachability graph* of N. The states set $RS(N)$ of A_N consists of all markings that can be reached from initial state M_0 by sequences of transition firings.

τ-Regions. Let $\tau \subseteq I$. If an input A of τ-synthesis allows a positive decision, we want to construct a corresponding τ-net N. TS represents the behavior of a modeled system by means of *global states* (states of TS) and transitions between them (events). Dealing with a Petri net, we operate with *local states* (places) and their changing (transitions), while the global states of a net are markings, i.e., combinations of local states. Since A and A_N must be isomorphic, 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: A τ-region of $A = (S, E, \delta, s_0)$ is a pair (sup, sig) of *support* $sup : S \to \{0,1\}$ and *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. A region (sup, sig) models a place p and the associated part of the flow function f. In particular, $f(p, e) = sig(e)$ and $M(p) = sup(s)$, for marking $M \in RS(N)$ that corresponds to $s \in S(A)$. Every τ-region $R = (sup, sig)$ partitions the set of states S into two disjoint subsets: $S = S_0^R \cup S_1^R$, where $S_i^R = \{s \in S \mid sup(s) = i\}$, $i \in \{0,1\}$. Since each of S_0^R and S_1^R uniquely defines sup, we will often identify S_1^R with sup, for R.

Every set $\mathcal{R}$ of τ-regions of A defines the *synthesized τ-net* $N_A^{\mathcal{R}} = (\mathcal{R}, E, f, M_0)$ with $f((sup, sig), e) = sig(e)$ and $M_0((sup, sig)) = sup(s_0)$ for all $(sup, sig) \in \mathcal{R}, e \in E$. To ensure that the input behavior is captured by the synthesized net, we have to discern global states, and prevent the firings of transitions when their corresponding events are not present in TS. This is stated as so called *separation atoms* and *problems*. 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 every SSP atom of A is τ-solvable then A has the *τ-state separation property* (τ-SSP, for short). A pair (e, s) of event $e \in E$ and state $s \in S$ where e does not

x	nop(x)	inp(x)	out(x)	set(x)	res(x)	swap(x)	used(x)	free(x)
0	0		1	1	0	1		0
1	1	0		1	0	0	1	

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

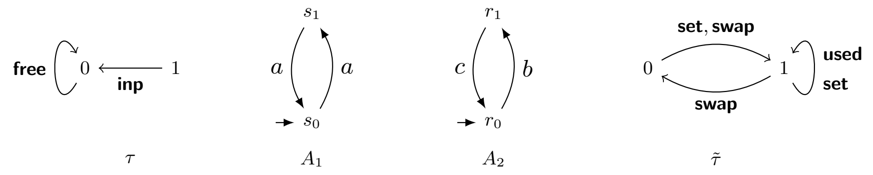

Fig. 2: Left: $\tau = \{\mathsf{inp}, \mathsf{free}\}$. Right: $\tilde{\tau} = \{\mathsf{swap}, \mathsf{used}, \mathsf{set}\}$. The TS A_1 has no ESSP atoms. Hence, it has the τ-ESSP and $\tilde{\tau}$-ESSP. The only SSP atom of A_1 is (s_0, s_1). It is $\tilde{\tau}$-solvable by $R_1 = (sup_1, sig_1)$ with $sup_1(s_0) = 0$, $sup_1(s_1) = 1$, $sig_1(a) = \mathsf{swap}$. Thus, A_1 has the $\tilde{\tau}$-admissible set $\mathcal{R} = \{R_1\}$, and the $\tilde{\tau}$-net $N_A^{\mathcal{R}} = (\{R_1\}, \{a\}, M_0, f)$ with $M_0(R_1) = sup_1(s_0)$ and $f(R_1, a) = sig_1(a)$ solves A_1. The SSP atom (s_0, s_1) is not τ-solvable, thus, neither is A_1. TS A_2 has ESSP atoms (b, r_1) and (c, r_0), which are both $\tilde{\tau}$-unsolvable. 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, but not the $\tilde{\tau}$-ESSP. None of the (E)SSP atoms of A_2 can be solved by any τ-region.

occur, that is $\neg s \xrightarrow{e}$, defines an *event state separation atom* (ESSP atom). A τ-region $R = (sup, sig)$ *solves* (e, s) if $sig(e)$ is not defined on $sup(s)$ in τ, that is, $\neg\delta_\tau(sup(s), sig(e))$. If every ESSP atom of A is τ-solvable then A has the *τ-event state separation property* (τ-ESSP, for short).

A set $\mathcal{R}$ of τ-regions of A is called *τ-admissible* if for each (E)SSP atom there is a τ-region R in $\mathcal{R}$ that solves it. The next lemma establishes the connection between the existence of τ-admissible sets of A and (the solvability of) τ-synthesis:

Lemma 1 ([1]). *A TS A is isomorphic to the reachability graph of a τ-net N if and only if there is a τ-admissible set $\mathcal{R}$ of A such that $N = N_A^{\mathcal{R}}$.*

The synthesis problem for Boolean nets of type τ is formulated as follows.

τ-Synthesis:

Input: an initialized TS A.
Decide: whether there exists a τ-admissible set $\mathcal{R}$ of A.
If so: construct such a set $\mathcal{R}$. *Otherwise:* reject input.

The present paper studies the complexity of τ-synthesis, for the types τ which do not include the interaction nop. The following lemma helps to treat several (isomorphic) types simultaneously:

Lemma 2 (Without Proof). *If $\tau_0 \cong \tau_1$, then an ESSP atom or a SSP atom of A is τ_0-solvable if and only if it is τ_1-solvable.*

3 Polynomial-Time Synthesis for nop-Free Boolean Types of Nets

In this section, we focus on the nop-free Boolean types τ such that $\mathsf{swap} \notin \tau$ and show that τ-synthesis is polynomial for these types of nets:

Theorem 1. *If $\tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$, then τ-solvability is polynomial.*

Notice that Theorem 1 covers exactly 64 types. The roadmap for the proof of Theorem 1 is as follows: Firstly, we show that τ-synthesis is polynomial for all 16 types τ that satisfy $\tau = \{\mathsf{set}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. After that, we do the same for the 16 types τ that satisfy $\tau = \{\mathsf{set}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. By isomorphisms (Lemma 2), this implies also the tractability of τ-synthesis for the 16 types τ with $\tau = \{\mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. Finally, we show that τ-synthesis is polynomial for the remaining 16 types that satisfy $\tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{free}, \mathsf{used}\}$, too. This approach covers all the types of Theorem 1 and thus proves it.

In the context of τ-synthesis, the input TS represents the behavior of the sought-for τ-net. Hence, restrictions of the possible behavioral patterns of the net imply restrictions for structural properties of the admissible synthesis inputs. In particular, if the firing of an arbitrary but fixed transition e of a τ-net N leads always to the same global marking M, then any e-labeled arc in N's state graph A_N has the the same sink s that corresponds to M. We formulate this as a structural property of TS:

Single-Sink Property. We say that TS A has the *single-sink* property if it satisfies $|\{s \in S(A) \mid \xrightarrow{e} s\}| = 1$ for all $e \in E(A)$.

The following lemma establishes that if τ is nop-free and swap-free, then state graphs of τ-nets have the single-sink property. In particular, this makes the single-sink property a necessary condition for the τ-solvability of TS.

Lemma 3. *Let $\tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$. If a TS A is τ-solvable, then it has the single-sink property.*

Proof. Let $e \in E(A)$ be arbitrary but fixed. We have to show that all e-labeled arcs have the same sink. By contraposition, assume there are arcs $\xrightarrow{e} s$ and $\xrightarrow{e} s'$ such that $s \neq s'$. Since A is τ-solvable, there is a τ-region (sup, sig) that solves the SSP atom (s, s'), that is, $sup(s) \neq sup(s')$. This contradicts $sig(e) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{res}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$, cf. Figure 1; hence the claim. $\square$

Clearly, whether a TS A has an event $e \in E(A)$ that has two distinct sinks $s, s' \in S(A)$ is decidable in polynomial time (in the size of A). In case of a positive decision, we simply reject A by Lemma 3. Thus, justified by Lemma 3, in the remainder of this section, unless stated explicitly otherwise, we assume that A is an arbitrary but fixed TS that has the single-sink property.

3.1 Types τ such that $\tau = \{\mathsf{set}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$

In the following, unless stated explicitly otherwise, let τ be a type such that $\tau = \{\mathsf{set}, \mathsf{res}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. Moreover, let τ_0, τ_1, τ_2 and τ_3 be defined in accordance to Figure 3.

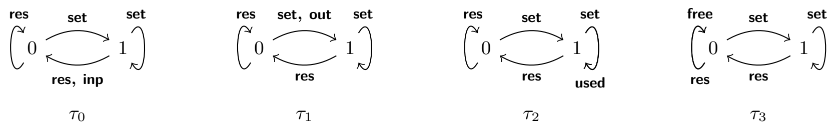

Fig. 3: The isomorphic types $\tau_0 = \{\mathsf{inp}, \mathsf{res}, \mathsf{set}\}$ and $\tau_1 = \{\mathsf{out}, \mathsf{res}, \mathsf{set}\}$ as well as the isomorphic types $\tau_2 = \{\mathsf{res}, \mathsf{set}, \mathsf{used}\}$ and $\tau_3 = \{\mathsf{res}, \mathsf{set}, \mathsf{free}\}$.

The main insights about the structural nature of τ can be summarized as follows: Firstly, every SSP atom (s, s') of A is τ-solvable. Secondly, an ESSP atom of A is τ-solvable if and only if it is τ_0-solvable or τ_2-solvable. Thirdly, if $\alpha = (a, q)$ is an ESSP atom of α, then there are exactly two certain regions for which have to ascertain if they solve α or not. No other regions have to be considered at all.

The following lemma corresponds to the first statement. Its proof is constructive and Figure 4.2 illustrates the region presented in lemma:

Lemma 4. *The TS A has the τ-SSP.*

Proof. If (s, s') is a SSP atom of A, then the following τ-region $R = (sup, sig)$ solves (s, s'): $sup(s') = 0$ and $sup(q) = 1$ for all $q \in S(A) \setminus \{s'\}$; for all $e \in E(A)$, if $\xrightarrow{e} s'$ then $sig(e) = \mathsf{res}$, otherwise $sig(e) = \mathsf{set}$. By definition of R and A's single-sink property, we have, for all $e \in E(A)$ and all $q, q' \in S(A)$, that $sig(e) = \mathsf{res}$ if and only if $q \xrightarrow{e} q'$ implies $q' = s'$. Thus, R is well defined and satisfies $sup(s) \neq sup(s')$. Since (s, s') was arbitrary, this proves the claim. $\square$

The following lemma paves us the way to the comprehension, that it is sufficient for a given ESSP atom $\alpha = (a, q)$ to test if it is τ_0- or τ_2-solvable:

Lemma 5. *If $\alpha = (a, q)$ is an ESSP atom of A, then α is τ-solvable if and only there is $\tau' \in \{\tau_0, \tau_2\}$ and $\tau'' \subseteq \tau$ such that α is τ'-solvable and $\tau' \cong \tau''$.*

Proof. Only-if: Let $R = (sup, sig)$ be a τ-region of A. For a start, we claim that there is a $\{\mathsf{res}, \mathsf{set}, sig(a)\}$-region $R' = (sup, sig')$ of A (with the same support as R) as follows: for all $e \in E(A)$, if $e = a$, then $sig'(e) = sig(e)$; otherwise, if $sig(e) \in \{\mathsf{inp}, \mathsf{res}, \mathsf{free}\}$ then $sig'(e) = \mathsf{res}$, and if $sig(e) \in \{\mathsf{out}, \mathsf{set}, \mathsf{used}\}$, then $sig'(e) = \mathsf{set}$. That R' is well-defined can be seen as follows: R is a τ-region, thus $s \xrightarrow{e} s' \in A$ implies $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$. Moreover, since, for

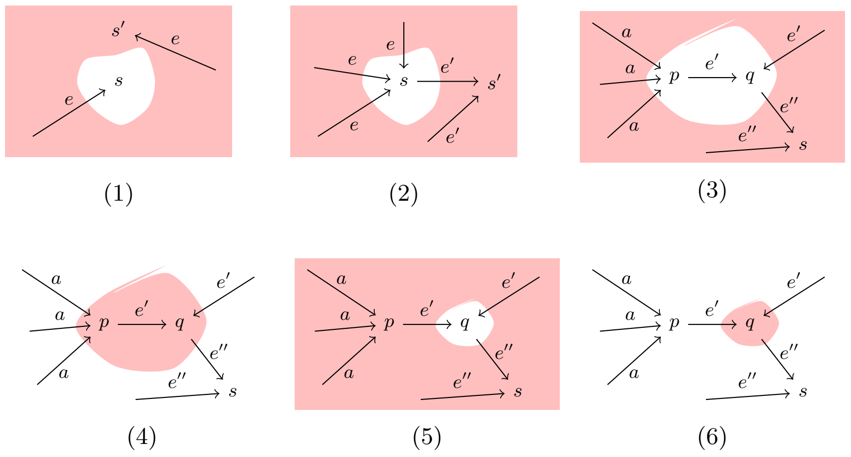

Fig. 4: Sketches of regions, where the red colored area corresponds to the states with a positive support. (1) Sketch of the separation of two states $s \neq s'$ which are both sinks of the same event e, i.e., A does not have the single-sink property. If $\mathsf{nop} \notin \tau$ and $\mathsf{swap} \notin \tau$, then such a separating τ-region is ruled out in advance. (2) Sketch of a region that separates the single-sink s of e from all other states of A according to Lemma 4. (3) Sketch of the $\{\mathsf{inp}, \mathsf{res}, \mathsf{set}\}$-region R that solves (a, q) in accordance to Lemma 6.1 and, sketched by (4), its complementary $\{\mathsf{out}, \mathsf{res}, \mathsf{set}\}$-region. (5) Sketch of the $\{\mathsf{used}, \mathsf{res}, \mathsf{set}\}$-region R that solves (a, q) in accordance to Lemma 6.2 and, depicted by (6), its complementary $\{\mathsf{free}, \mathsf{res}, \mathsf{set}\}$-region.

all $i \in \{\mathsf{inp}, \mathsf{res}, \mathsf{free}\}$ ($i \in \{\mathsf{out}, \mathsf{set}, \mathsf{used}\}$), $b \xrightarrow{i} b'$ implies $b \xrightarrow{\mathsf{res}} b'$ ($b \xrightarrow{\mathsf{set}} b'$) for all $b, b' \in \{0, 1\}$, the definition also implies that $sup(s) \xrightarrow{sig'(e)} sup(s') \in \tau$. Thus, R' is a well-defined region.

Consequently, if $R = (sup, sig)$ solves α, i.e., $\neg sup(q) \xrightarrow{sig(a)}$, which implies $sig(a) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$, then the just introduced $\{\mathsf{res}, \mathsf{set}, sig(a)\}$-region R' solves α. If $sig(a) \in \{\mathsf{inp}, \mathsf{used}\}$, then R' is already a τ_0- or a τ_2-region, which implies that α is τ_0- or τ_2-solvable. Otherwise, if $sig(e) \in \{\mathsf{out}, \mathsf{free}\}$, then either $\{\mathsf{res}, \mathsf{set}, sig(a)\} \cong \tau_0$ or $\{\mathsf{res}, \mathsf{set}, sig(a)\} \cong \tau_2$, and the claim follows by Lemma 2.

If: If $\tau' \subseteq \tau$, then the claim is trivial. Otherwise, $\tau' \cong \tau'' \subseteq \tau$ and, by Lemma 2, α is τ''-solvable. Hence, α is τ-solvable. $\square$

The following lemma tells us exactly for which τ_0- or τ_2-region we have to look for:

Lemma 6. *Let $\tau \in \{\tau_0, \tau_2\}$, $\alpha = (a, q)$ an ESSP atom of A, and p the unique sink of the a-labeled arcs. There is a τ-region $R' = (sup', sig')$ of A that solves α if and only if there is a τ-region $R = (sup, sig)$ as follows:*

1. *If $sig'(a) = \mathsf{inp}$, then $sup = S(A) \setminus \{p, q\}$ and, for all $e \in E(A)$, if $e = a$, then $sig(e) = \mathsf{inp}$; if $e \neq a$ and $\xrightarrow{e} q \in A$ or $\xrightarrow{e} p \in A$, then $sig(e) = \mathsf{res}$; otherwise $sig(e) = \mathsf{set}$.*

2. If $sig'(a) = \textsf{used}$, then $sup = S(A) \setminus \{q\}$ and, for all $e \in E(A)$, if $e = a$, then $sig(e) = \textsf{used}$; if $\xrightarrow{e} q \in A$, then $sig(e) = \textsf{res}$; otherwise $sig(e) = \textsf{set}$.

Proof. The *if*-direction is trivial.

Only-if: Let $R' = (sup', sig')$ be a τ-region, that solves α. By definition of τ and $\neg sup(q) \xrightarrow{sig(a)}$, we have either $sig'(a) = \textsf{inp}$ and $sup'(q) = 0$ or $sig'(a) = \textsf{used}$ and $sup'(q) = 0$. The former case corresponds to $\tau = \tau_0$ and the latter to $\tau = \tau_2$.

Assume, for a start, $sig'(a) = \textsf{inp}$ and $sup'(q) = 0$, which implies $sup'(p) = 0$ and $sup'(s) = 1$ for all $s \in S(A)$ that satisfy $s \xrightarrow{a}$. We argue that R as in (1.) is well-defined. Notice that since A has the single-sink property, the definition of R's signature is consistent. Assume, for a contradiction, that there is an edge $s \xrightarrow{e} s' \in A$ such that $sup(s) \xrightarrow{e} sup(s') \notin \tau$. If $sup(s') = 1$, then $s' \notin \{p, q\}$ and thus $sig(e) = \textsf{set}$, which implies $sup(s) \xrightarrow{e} sup(s') \in \tau$ for all $sup(s) \in \{0, 1\}$. If $sup(s') = 0$, then $s' \in \{p, q\}$ and $sig(e) \in \{\textsf{inp}, \textsf{res}\}$. If $sig(e) = \textsf{res}$, then $sup(s) \xrightarrow{e} sup(s') \in \tau$ for all $sup(s) \in \{0, 1\}$. Thus, $sig(e) = \textsf{inp}$, which implies $e = a$ and $s' = p$. Since $sup(s) \xrightarrow{e} sup(s') \notin \tau$, by $sig(a) = \textsf{inp}$, we get $sup(s) = 0$, which implies $s \in \{p, q\}$. If $s = q$, then (a, q) is not an ESSP atom, a contradiction. Otherwise, if $s = p$, then $p \xrightarrow{e} p \in A$, which contradicts that R' is a region such that $sig'(a) = \textsf{inp}$. Thus $sup(s) \xrightarrow{e} sup(s') \in \tau$.

Similarly, one argues that if $sig'(a) = \textsf{used}$ and $sup'(q) = 0$, the region of (2.) is well defined. This proves the lemma. ☐

Corollary 1. *Let $\tau = \{\textsf{res}, \textsf{set}\} \cup \omega$ such that $\omega \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}, \textsf{free}\}$, A be a TS and $|A| = |S(A)|^2 \cdot |E(A)|$. There is an algorithm that constructs a τ-admissible set $\mathcal{R}$ of A if it exists and rejects A otherwise, which terminates in time $\mathcal{O}(|A|^2)$.*

Proof. If β (α) is an arbitrary but fixed (E)SSP atom, then to define a region in accordance to Lemma 4 (Lemma 5), respectively its "isomorphic" counterpart, and to check its validity is doable in time $\mathcal{O}(|A|)$. We have to check at most three regions per atom and we have at most $|S(A)|^2 \leq |A|$ SSP atoms ($|S(A)| \cdot |E(A)| \leq |A|$ ESSP atoms) to solve. Thus, within the time $\mathcal{O}(|A|^2)$ we have either a τ-admissible set $\mathcal{R}$ that already defines a solving net implicitly be Lemma 1, or A is not τ-solvable and has to be rejected. ☐

3.2 Types τ such that $\tau = \{\textsf{set}\} \cup \omega$ and $\omega \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}, \textsf{free}\}$

In the following, we investigate the types $\tau_0 = \{\textsf{set}\} \cup \omega$ where $\omega \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}\}$ and $\tau_1 = \{\textsf{set}, \textsf{free}\} \cup \omega$ where $\omega \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}\}$ separately. The reason for this distinction is that, for the former types, a solving region for an atom exists if and only if a solving region exists whose support excludes exactly the *necessary* states. More exactly, let $\alpha = (a, q)$ be an ESSP atom of A and p be the unique sink of a-labeled edges in A. By definition, if $R = (sup, sig)$ is a τ-region, that solves α, then $sig(a) = \textsf{inp}$ implies $sup(p) = sup(q) = 0$ (Figure 5.1), and $sig(a) = \textsf{used}$ implies $sup(q) = 0$ (Figure 5.2), and $sig(a) = \textsf{out}$ implies $sup(s) = 0$ for all $s \in \{s \in S(A) \mid s \xrightarrow{a}\}$ (cf. Figure 5.3). In other words, R's support necessarily

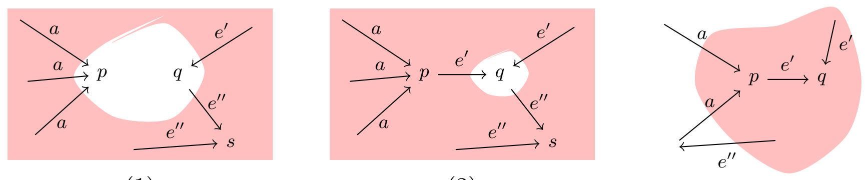

Fig. 5: Sketches of ESSP atom (a, p) solving τ-regions $R = (sup, sig)$ according to Lemma 7, i.e., $\tau = \{\mathsf{set}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$: (1) $sup = S(A) \setminus \{p, q\}$ and $sig(a) = \mathsf{inp}$; (2) $sup = S(A) \setminus \{q\}$ and $sig(a) = \mathsf{used}$; (3) $sup = S(A) \setminus \{s \in S(A) \mid \exists s' \in S(A) : \delta(s, a) = s'\}$ and $sig(a) = \mathsf{out}$.

excludes $\{p, q\}$, $\{s \in S(A) \mid s \xrightarrow{a}\}$ and $\{q\}$ if $sig(a) = \mathsf{inp}$, $sig(a) = \mathsf{out}$ and $sig(a) = \mathsf{used}$, respectively. In the following subsection, we show that α is τ_0-solvable if and only if there is a solving τ_0-region $R = (sup, sig)$ whose support excludes only the necessary states.

Unfortunately, the types covered by τ_1 do not always have this property. For τ_1, we rather work the other way around. In particular, we present an algorithm that starts from a set of necessarily included states and extends it iteratively (by further states) until it either reaches a separating region or a solution does not exists.

Types τ such that $\tau = \{\mathsf{set}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$. In the remainder of this subsection, unless stated explicitly otherwise, let $\tau = \{\mathsf{set}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$.

It turns out, that τ allows to search for separating regions in a very specific way. In particular, the following lemma shows that for a given ESSP atom α of A there are at most three distinct τ-regions that solves α and how they are defined:

Lemma 7. *Let $\alpha = (a, q)$ be an ESSP atom of A, and p the single sink of the a-labeled arcs. There is a τ-region $R = (sup', sig')$ that solves α if and only if there is a τ-region $R = (sup, sig)$ as follows:*

1. *If $sig'(a) = \mathsf{inp}$, then $sup = S(A) \setminus \{p, q\}$ and, for all $e \in E(A)$, if $\xrightarrow{e} p$ or $\xrightarrow{e} q$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{set}$.*
2. *If $sig'(a) = \mathsf{used}$, then $sup = S(A) \setminus \{q\}$ and, for all $e \in E(A)$, if $e = a$, then $sig(e) = \mathsf{used}$; if $\xrightarrow{e} q$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{set}$.*
3. *If $sig'(a) = \mathsf{out}$, then $sup' = S(A) \setminus \{s \in S(A) \mid s \xrightarrow{a}\}$ and, for all $e \in E(A)$, if $e = a$, then $sig(a) = \mathsf{out}$; if $\xrightarrow{e} s$ for some $s \in \{s \in S(A) \mid s \xrightarrow{a}\}$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{set}$.*

Proof. The if direction is trivial.

Only-If: Since R' solves α, it holds $sig'(a) \in \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$ and $\neg sup'(q) \xrightarrow{sig'(a)}$. Let's consider the possible cases:

(1): Assume $sig'(a) = \mathsf{inp}$, which implies $sup'(p) = sup'(q) = 0$. This implies, for all $e \in E(A)$, if there is $s \in \{p, q\}$ such that $\xrightarrow{e} s$, then $sig(e) = \mathsf{inp}$ and thus $sup'(s') = 1$ and $sup'(s'') = 0$ for all e-labeled edges $s' \xrightarrow{e} s''$ of A. In particular, there is no e-labeled edge $s \xrightarrow{e} s'$ such that $s, s' \in \{p, q\}$. We argue that the region R is well defined. Assume, for a contradiction, that there is $s \xrightarrow{e} s' \in A$ such that $sup(s) \xrightarrow{sig(e)} sup(s') \notin \tau$. If $sup(s') = 1$, implying $s' \notin \{p, q\}$, then $sig(e) = \mathsf{set}$ by definition of sig. This would imply $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$ for all $sup(s) \in \{0, 1\}$. Hence, $sig(s') = 0$, which implies $s' \in \{p, q\}$ and thus $sig'(e) = \mathsf{inp}$. Moreover, by $sup(s) \xrightarrow{sig(e)} sup(s') \notin \tau$ and $sup(s') = 0$, we get $sup(s) = 0$, which implies $s \in \{p, q\}$, a contradiction. Hence, R is well-defined and solves α.

(2): If $sig'(a) = \mathsf{used}$, then $sup'(p) = 1$ and $sup'(q) = 0$. This implies, for all $e \in E(A)$, if $s \xrightarrow{e} q$, then $sig(e) = \mathsf{inp}$ and thus $sup'(s) = 1$. This particularly implies $q \xrightarrow{e} q \notin \tau$ for all $e \in E(A)$. Similar to (1), if there is $s \xrightarrow{e} s' \in A$ such that $sup(s) \xrightarrow{sig(e)} sup(s') \notin \tau$, then $sig(s') = 0$, which implies $s' = q$ and thus $sig'(e) = \mathsf{inp}$. Since $sup(s) \xrightarrow{sig(e)} sup(s') \notin \tau$ and $sup(s') = 0$, we get $sup(s) = 0$ and thus $s = q$, a contradiction. Hence, R is well-defined and solves α.

(3): Assume $sig'(a) = \mathsf{out}$, which implies $sup'(p) = sup'(q) = 1$ and $sup(s) = 0$ for all $s \in \{s \in S(A) \mid s \xrightarrow{a}\}$. Moreover, for all $e \in E(A)$, the latter implies that if $s \xrightarrow{e} s' \in E(A)$ such that $s' \in \{s \in S(A) \mid s \xrightarrow{a}\}$, then $sig(e) = \mathsf{inp}$ and $sup(s) = 1$ and thus $s \notin \{s \in S(A) \mid s \xrightarrow{a}\}$. Similar to the arguments of (1) and (2), one get's that this implies that R is well-defined and solves α. □

The following lemma shows that we can also specifically search for regions that solve a given SSP atom:

Lemma 8. *Let $\beta = (p, q)$ be an SSP atom of A. There is a τ-region $R = (sup', sig')$ that solves β if and only if there is a τ-region $R = (sup, sig)$ as follows:*

1. *$sup = S(A) \setminus \{p\}$ and, for all $e \in E(A)$, if $\xrightarrow{e} p$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{set}$.*
2. *$sup = S(A) \setminus \{q\}$ and, for all $e \in E(A)$, if $\xrightarrow{e} p$, then $sig(e) = \mathsf{inp}$, otherwise $sig(e) = \mathsf{set}$.*

Proof. The if-direction is trivial.

(*Only-if*): Since R' solves β, either $sup'(p) = 0$ and $sup'(q) = 1$ or $sup'(p) = 1$ and $sup'(q) = 0$ is true. We consider both cases, where the former corresponds to the first condition of the lemma and the latter to the second.

For a start, let $sup'(p) = 0$ and $sup'(q) = 1$. Since $sup'(p) = 0$, if $s \xrightarrow{e} q \in A$, then $sig'(e) = \mathsf{inp}$ and $sup'(s) = 1$ and thus $s \neq q$. We argue that the definition of R according to (1.) is sound. Assume, for a contradiction, there is an $s \xrightarrow{e} s' \in A$ such that $sup(s) \xrightarrow{sig(e)} sup(s') \notin \tau$. If $sup(s') = 1$, then $sig(e) = \mathsf{set}$ and $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$. Thus, it is $sup(s') = 0$, which implies $s' = q$ and

$sig(e) = \mathsf{inp}$. By $sig'(e) = \mathsf{inp}$, this also implies $s \neq q$ and thus $sup'(s) = 1$. Hence, $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$, a contradiction.

Similarly, if $sup'(p) = 1$ and $sup'(q) = 0$, then the region of (2.) exists. □

Corollary 2. *Let $\tau = \{\mathsf{set}\} \cup \omega$ such that $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$, A be a TS and $|A| = |S(A)|^2 \cdot |E(A)|$. There is an algorithm that constructs a τ-admissible set $\mathcal{R}$ of A if it exists and rejects A otherwise, which terminates in time $\mathcal{O}(|A|^2)$.*

Proof. The proof is similar to the one for Corollary 1. □

The Types τ such that $\tau = \{\mathsf{set}, \mathsf{free}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$. In this subsection, unless stated explicitly otherwise, let τ be a type such that $\tau = \{\mathsf{set}, \mathsf{free}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$. In what follows, we present an algorithm that starts from a set of necessarily included states and extends it iteratively (by further states) until it either reaches a separating region or separation is not possible at all.

Lemma 9. *Let $S \subseteq S(A)$. The set S is a τ-support if and only if the following conditions are true:*

1. *If $s \xrightarrow{e} s' \in A$ such that $S(s) = 1$ and $S(s') = 0$, then $\mathsf{inp} \in \tau$*
2. *For all $e \in E(A)$, if $s \xrightarrow{e} s', q \xrightarrow{e} q' \in A$, then $\{(S(s), S(s')), (S(q), S(q'))\}$ belongs to $\{\{(0,1)\}, \{(1,1)\}, \{(1,0)\}, \{(0,0)\}, \{(0,1), (1,1)\}\}$.*

Proof. Only-if: Assume that S is a τ-support that allows the τ-signature sig, that is, if $s \xrightarrow{e} s' \in A$, then $sup(s) \xrightarrow{sig(e)} sup(s') \in \tau$. Clearly, by the definition of τ, if $s \xrightarrow{e} s' \in A$ such that $S(s) = 1$ and $S(s') = 0$, then $sig(e) = \mathsf{inp}$. Moreover, since A has the single-sink property, for all $e \in E(A)$, if $s \xrightarrow{e} s' \in A$ and $q \xrightarrow{e} q' \in A$, then $sup(s') = sup(q')$. Furthermore, since $\mathsf{res} \notin \tau$, if $sup(s') = sup(q') = 0$, then $\{S(s), S(s'), S(q), S(q')\}$ cannot be $\{(1,0), (0,0)\}$. All the other possibilities are allowed. This proves the Only-if-direction.

If: For all events $e \in E(A)$, we define the following set $S_e = \{(S(s), S(s')) \mid s, s' \in S(A)$ and $s \xrightarrow{e} s'\}$. Notice that if $|S_e| \geq 3$, then there are edges $s \xrightarrow{e} s'$ and $q \xrightarrow{e} q'$ such that $\{(S(s), S(s')), (S(q), S(q'))\} \notin \{\{(0,1)\}, \{(1,0)\}, \{(0,0)\}, \{(0,0)\}, \{(0,1), (1,1)\}\}$. Moreover, if $|S_e| = 2$, then one easily finds out that $S_e = \{\{(0,1), (1,1)\}\}$. Thus, the following definition of sig yields a well-defined τ-region (S, sig) of A: for all $e \in E(A)$, if $S_e = \{(1,0)\}$, then $sig(e) = \mathsf{inp}$; if $S_e = \{(0,1)\}$, then $sig(e) = \mathsf{out}$ if $\mathsf{out} \in \tau$, otherwise $sig(e) = \mathsf{set}$; if $S_e = \{(1,1)\}$, then $sig(e) = \mathsf{used}$ if $\mathsf{used} \in \tau$, otherwise $sig(e) = \mathsf{set}$; if $S_e = \{(0,1), (1,1)\}$, then $sig(e) = \mathsf{set}$; otherwise, $S_e = \{(0,0)\}$ and we then define $sig(e) = \mathsf{free}$. □

Example 1 (Algorithm application). Consider TS A with the set of states $S(A) = \{\iota, p, q, s\}$ and $E(A) = \{a, b, c, d, e\}$ depicted on the left in Figure 6. Let us apply Algorithm 1 to A, with input $\tau = \{\mathsf{set}, \mathsf{free}\}$ and $S = \{\iota, p, s\}$ (the states from S are colored in red in Figure 6). The first *while*-check in line 1 of the algorithm finds the transition $\iota \xrightarrow{a} q$ with $S(\iota) = 1$ and $S(q) = 0$. Since $\mathsf{inp} \notin \tau$, we remove ι from S (line 2). The new set S is depicted in the middle of Figure 6. For $S = \{p, s\}$, we repeat the *while*-check and find $s \xrightarrow{b} \iota$, $q \xrightarrow{b} \iota$ with $S(s) = 1$ and $S(q) = S(\iota) = 0$. Accoridng to the line 2 of the algorithm, we remove s from S, obtaining $S = \{p\}$. The new *while*-check does not report any inconsistency in S, hence $\{p\}$ is a valid τ-suport. The τ-region defined by this support is given on the right of Figure 6.

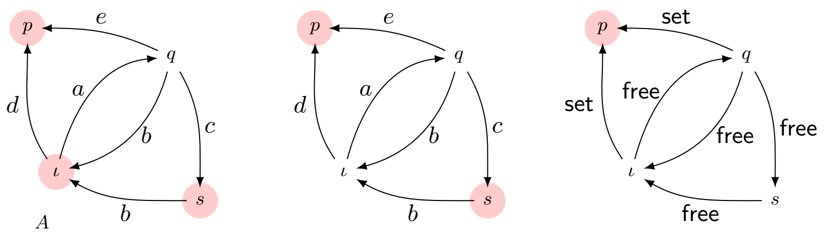

Fig. 6: A $\{\mathsf{set}, \mathsf{free}\}$-region of TS A has been obtained from $\{\iota, p, s\}$ by Algorithm 1.

Lemma 10 (Algorithm 1). *Let $|A| = |S(A)|^2 \cdot |E(A)|$ and $S \subseteq S(A)$. For the input (A, S, τ)*

1. Sup *terminates and its running time is at most $\mathcal{O}(|A|^3)$.*
2. Sup *returns a τ-support.*
3. *If $R = (sup, sig)$ is an S-maximal τ-region such that $sup \subseteq S$, then* Sup *returns sup.*

Proof. (1): If Sup initiates the while loop, then it decreases the cardinality of S. This implies that Sup terminates after at most $|S(A)| \leq |A|$ while-loop iterations. Moreover, the conditions of the while-loop can be tested in time $\mathcal{O}(|A|^2)$. Hence the claim.

(2): If Sup returns S', then the set S' does not satisfy the conditions of the while-loop. Thus, by Lemma 9, S' allows a τ-signature *sig* that is defined in correspondence to Lemma 9.

Algorithm 1: Sup

Input: Type $\{\mathsf{set}, \mathsf{free}\} \subseteq \tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$. TS A with states $S(A)$ and events $E(A)$. $S \subseteq S(A)$.

Result: A τ-support *sup* of A such that $sup \subseteq S$.

```
1 while ($s \xrightarrow{e} s' \in A$ such that $S(s) = 1, S(s') = 0$ and $\mathsf{inp} \notin \tau$) or
    ($s \xrightarrow{e} s', q \xrightarrow{e} q' \in A$ such that $S(s) = 1, S(s') = S(q) = S(q') = 0$) do
2   | $S = S \setminus \{s\}$;
3 end
4 return $S$;
```

(3): By (1), SUP terminates after at most $t \leq |S(A)|$ iterations of the while-loop. Let $S_0 = S$ and, for $i \in \{1, \dots, t\}$, let S_i be the set that originates from S_{i-1} due to the execution of the i-th while-loop iteration. More exactly, after the $i-1$-th execution of the while-loop, we have S_{i-1}; SUP tests (for the i-th time) the while-loop condition; if the condition is not satisfied, then $S_i = S_{i-1}$ and SUP terminates and returns S_i, otherwise, the condition is satisfied and SUP redefines S_{i-1}, that is, $S_i = S_{i-1} \setminus \{s\}$, where s corresponds to the while-loop condition.

In the following, we show by induction on the while-loop iterations that if $sup \subseteq S_{i-1}$, then $sup \subseteq S_i$. By the construction, we have $sup \subseteq S = S_0$. Assume that $i \in \{1, \dots, t\}$ and $sup \subseteq S_{i-1}$. If $S_i = S_{i-1}$, then we have $sup \subseteq S_i$ by the assumption. Otherwise, $S_i = S_{i-1} \setminus \{s\}$, where $s \xrightarrow{e} s' \in A$ such that $S(s) = 1, S(s') = 0$ and $\mathsf{inp} \notin \tau$ or $s \xrightarrow{e} s', q \xrightarrow{e} q' \in A$ such that $S(s) = 1, S(s') = S(q) = S(q') = 0$. If $sup \not\subseteq S_i = S_{i-1} \setminus \{s\}$, then, by $sup \subseteq S_{i-1}$, we have $sup(s) = 1, sup(s') = 0$ and $\mathsf{inp} \notin \tau$, which is a contradiction, or we have $sup(s) = 1, sup(s') = sup(q) = sup(q') = 0$, which contradicts $\mathsf{res} \notin \tau$. This implies $sup \subseteq S_i$ and, in particular, $sup \subseteq S_t$. If $sup \neq S_t$, then there is a state $s \in S_t$ such that $sup(s) = 0$. This implies $|S_t| > |sup|$. Since sup is a S-maximal support, this is a contradiction. Consequently, we have $sup = S_t$. □

Lemma 11. *Let $\{\mathsf{set}, \mathsf{free}\} \subseteq \tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{set}, \mathsf{used}, \mathsf{free}\}$, let A be a TS and let $\alpha = (a, q)$ be an ESSP atom and $\beta = (q, q')$ be an SSP atom of A. If $R = (sup, sig)$ is a (maximal) τ-region that solves α, then Algorithm 1 provides sup. In particular, sup is returned by* SUP$(A, S(A) \setminus \{s \in S(A) \mid \xrightarrow{a} s\}, \tau)$ *or sup is returned by* SUP$(A, S(A) \setminus \{q\}, \tau)$.

Proof. Let $R = (sup, sig)$ be a τ-region that solves α. Let p be the single-sink of a in A. If $sig(a) = \mathsf{inp}$ and $sup(q) = 0$, then we invoke SUP on input $S = S(A) \setminus \{p, q\}$; if $sig(a) = \mathsf{used}$ and $sup(q) = 0$, then we invoke SUP on input $S = S(A) \setminus \{q\}$; if $sig(a) = \mathsf{out}$ and $sup(q) = 1$, then we invoke SUP on input $S = S(A) \setminus \{s \in S(A) \mid s \xrightarrow{a}\}$; if $sig(a) = \mathsf{free}$ and $sup(q) = 1$, then we invoke SUP on input $S = S(A) \setminus \{s, s' \in S(A) \mid s \xrightarrow{a} s'\}$. By Lemma 10, SUP outputs sup. □

Corollary 3. *Let τ be a type of nets such that $\tau = \{\mathsf{set}, \mathsf{free}\} \cup \omega$ and $\omega \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}\}$ and $|A| = |S(A)|^2 \cdot |E(A)|$. There is an algorithm that constructs a τ-admissible set $\mathcal{R}$ of A if it exists and rejects A otherwise, which terminates in time $\mathcal{O}(|A|^4)$.*

Proof. Since we have at most $|S(A)| \leq |A|$ SSP atoms and at most $|S(A)| \cdot |E(A)| \leq |A|$ ESSP atoms to solve, the claim follows by and Lemma 10 and Lemma 11. □

The Types τ such that $\tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$. The following lemma implies that τ-synthesis is solvable in polynomial time if $\tau \subseteq \{\mathsf{inp}, \mathsf{out}, \mathsf{used}, \mathsf{free}\}$:

Lemma 12. *Let $\tau \subseteq \{\textsf{inp}, \textsf{out}, \textsf{used}, \textsf{free}\}$, let A be a TS and let $\alpha = (a, q)$ and $\beta = (q, q')$ be an ESSP and an SSP atom of A, respectively. Deciding if α or β are τ-solvable requires polynomial time.*

(Sketch). First of all, we observe that $S_e \in \{\{(0,0)\}, \{(0,1)\}, \{(1,0)\}, \{(1,1)\}\}$ for all $e \in E(A)$. If $\textsf{free} \in \tau$, then we can test the τ-solvability of α (β) by a procedure that originates as a modification of Algorithm 1: Firstly, we have to check if $s \xrightarrow{e} s' \in A$ such that $S(s) = 0, S(s') = 1$ and $\textsf{out} \not\in \tau$ or $S(s) = 1, S(s') = 1$ and $\textsf{used} \not\in \tau$. In case of a positive decision, we have (at least) to remove s', that is, $S = S \setminus \{s'\}$. Similarly, if $S(s) = 1, S(s') = 0$ and $\textsf{inp} \not\in \tau$, then we have to update $S = S \setminus \{s\}$.

Secondly, we have to check for all $e \in E(A)$ and all pairs $s \xrightarrow{e} s', q \xrightarrow{e} q'$ if there is $e_\tau \in \tau$ such that $S_e \in \{\{(0,0)\}, \{(0,1)\}, \{(1,0)\}, \{(1,1)\}\} \cap \{(x,y) \mid x \xrightarrow{e_\tau} y \in \tau\}$. In case of a negative decision, we have to update S, and one finds out that this is deterministically possible for all S_e under consideration.

Otherwise, if $\textsf{free} \not\in \tau$, then we obtain a similar procedure by starting from a minimal necessary subset $S \subseteq S(A)$ (for example, $\{s \in S(A) \mid s \xrightarrow{a}\}$ for $sig(a) = \textsf{inp}$). Then increase S deterministically, until either we obtain a valid support or we can correctly conclude that such a support does not exist. □

4 NP-Completeness of the Single-Instance ESSP Problem for a nop-Free Type

Let τ be a Boolean type of nets and A be a TS and α be an ESSP atom of A. The *single-instance* ESSP problem consists in deciding whether there is a τ-region R of A that solves α. Notice that the polynomial-time procedures for τ-synthesis which have been presented in Section 3 base on the fact that the single-instance problem is solvable in polynomial-time for the corresponding types, i.e, if $\tau \cap \{\textsf{nop}, \textsf{swap}\} \neq \emptyset$. In this section, we show that the absence of nop does not guarantee that the single-instance ESSP problem for Boolean τ-synthesis is polynomial. In particular, the following theorem exhibits a nop-free Boolean type for which the single-instance ESSP problem is intractable:

Theorem 2. *Let $\tau = \{\textsf{inp}, \textsf{res}, \textsf{set}, \textsf{swap}\}$, A be a TS and α be an ESSP atom of A. Deciding if there is a τ-region R of A that solves α is NP-complete.*

It is noteworthy that Theorem 2 does not imply that $\{\textsf{inp}, \textsf{res}, \textsf{set}, \textsf{swap}\}$-synthesis is NP-complete. Nevertheless, the NP-completeness of the corresponding single-instance ESSP problem gives a good reason to believe that the $\{\textsf{inp}, \textsf{res}, \textsf{set}, \textsf{swap}\}$-synthesis problem might actually be harder than the one for the types which lack both nop and swap.

The proof of Theorem 2 bases on a polynomial-time reduction of the following decision problem, which has been shown to be NP-complete in [5]:
Cubic monotone one-in-three 3Sat. (CM 1-in-3 Sat)

Input: a Boolean formula $\varphi = \{\zeta_0, \dots, \zeta_{m-1}\}$ of negation-free three-clauses $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$, where $i \in \{0, \dots, m-1\}$, with set of variables $V(\varphi) = \bigcup_{i=0}^{m-1} \zeta_i$; every variable $X \in V(\varphi)$ occurs in exactly three clauses.
Decide: whether there exists a *one-in-three model* M of φ, that is, a set $M \subseteq V(\varphi)$ such that $|M \cap \zeta_i| = 1$ for all $i \in \{0, \dots, m-1\}$.

Notice that the fact that every variable occurs in exactly three clauses and φ has exactly m three-clauses (and thus $3m$ unnegated literals) imply that $|V(\varphi)| = m$.

Example 2 (CM 1-in-3 3Sat). The instance $\varphi = \{\zeta_0, \dots, \zeta_5\}$ of CM 1-in-3 3Sat with set of variables $X = \{X_0, \dots, X_5\}$ 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 $M = \{X_0, X_4\}$.

In the remainder of this section, unless stated explicitly otherwise, we let $\varphi = \{\zeta_0, \dots, \zeta_{m-1}\}$ be an arbitrary but fixed input of CM 1-in-3 3Sat, where $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$ for all $i \in \{0, \dots, m-1\}$ such that, moreover, m is even. This is possible without loss of generality, which can be seen as follows: If m is odd and $V(\varphi) = \{X_0, \dots, X_{m-1}\}$, then we get an equivalent instance $\varphi' = \{\zeta_0, \dots, \zeta_{m-1}\} \cup \{\zeta'_0, \dots, \zeta'_{m-1}\}$ with set of variables $V(\varphi') = \{X_0, \dots, X_{m-1}\} \cup \{X'_0, \dots, X'_{m-1}\}$, where any X'_i is new and corresponds one-to-one to X_i. Moreover, ζ'_i contains X'_j if and only if ζ_i contains X_j. Clearly, φ is a *yes*-instance if and only if φ' is a *yes*-instance and, moreover, $|V(\varphi')| = 2m$.

Our Reduction. We reduce a given φ to a pair (A, α) of TS A and ESSP atom α such that φ has a one-in-three model if and only if there is a τ-region of A that solves α.

Notice that our current approach extends earlier methods that we applied to prove NP-completeness of τ-synthesis for many nop-equipped types [8–11]. However, our previous reductions crucially rely on the presence of nop and are thus useless for the current types.

Essentially, the resulting TS A can be seen as a composition of several (independent) TS, which we also call *gadgets*, since any of them is applied for a certain functionality. For a start, the TS A has the following gadgets H (top) and G (bottom) that provide the ESSP-atom $\alpha = (k, g_{24m+6})$ (ignore the gray colored area for now):

$$\begin{tikzcd}
h_0 \arrow[r, "u_0"] & h_1 \arrow[r, "v_0"] & h_2 \arrow[r, "w_0"] & h_3 \cdots h_{24m+3} \arrow[r, "u_{8m+1}"] & h_{24m+4} \arrow[r, "v_{8m+1}"] & h_{24m+5} \arrow[r, "w_{8m+1}"] & h_{24m+6} \arrow[r, "k"] & h
\end{tikzcd}$$

$$\begin{tikzcd}
g \arrow[r, "k"] & g_0 \arrow[r, "u_0"] & g_1 \arrow[r, "v_0"] & g_2 \arrow[r, "w_0"] & g_3 \cdots g_{24m+3} \arrow[r, "u_{8m+1}"] & g_{24m+4} \arrow[r, "v_{8m+1}"] & g_{24m+5} \arrow[r, "w_{8m+1}"] & g_{24m+6}
\end{tikzcd}$$

Notice that the sequence $u_0 v_0 w_0 u_1 v_1 w_1 \dots u_{8m} v_{8m} w_{8m} u_{8m+1} v_{8m+1} w_{8m+1}$ has $24m + 6$ events and precedes and succeeds the single occurrence of k in H and G, respectively.

Secondly, the TS A has, for every $i \in \{0, \dots, m-1\}$, the following four gadgets $T_{i,0}$ (top) and $T_{i,1}, T_{i,2}, T_{i,3}$ (bottom, from left to right) that apply the variables $X_{i_0}, X_{i_1}, X_{i_2}$ of the clause $\zeta_i = \{X_{i_0}, X_{i_1}, X_{i_2}\}$ as events:

$$\begin{tikzcd}
t_{i,0,0} \arrow[r, "k"] & t_{i,0,1} \arrow[r, "v_{8i}"] & t_{i,0,2} \arrow[r, "X_{i_0}"] & t_{i,0,3} \arrow[r, "v_{8i+1}"] & t_{i,0,4} \arrow[r, "X_{i_1}"] & t_{i,0,5} \arrow[r, "v_{8i+2}"] & t_{i,0,6} \arrow[r, "X_{i_2}"] & t_{i,0,7} \arrow[d, "v_{8i+3}"] \\
 & & & & t_{i,0,10} & t_{i,0,9} \arrow[l, "k"'] & & t_{i,0,8} \arrow[ll, "v_{8i+4}"']
\end{tikzcd}$$

$$\begin{tikzcd}
 & t_{i,1,1} \arrow[r, "X_{i_0}"] & t_{i,1,2} \arrow[d, "v_{8i+5}"'] \\
t_{i,1,0} \arrow[ur, "y_{6i}"] \arrow[dr, "y_{6i+1}"'] & & \\
 & t_{i,1,3} \arrow[r, "X_{i_1}"'] & t_{i,1,4}
\end{tikzcd}
\quad
\begin{tikzcd}
 & t_{i,2,1} \arrow[r, "X_{i_0}"] & t_{i,2,2} \arrow[d, "v_{8i+6}"'] \\
t_{i,2,0} \arrow[ur, "y_{6i+2}"] \arrow[dr, "y_{6i+3}"'] & & \\
 & t_{i,2,3} \arrow[r, "X_{i_2}"'] & t_{i,2,4}
\end{tikzcd}
\quad
\begin{tikzcd}
 & t_{i,3,1} \arrow[r, "X_{i_1}"] & t_{i,3,2} \arrow[d, "v_{8i+7}"'] \\
t_{i,3,0} \arrow[ur, "y_{6i+4}"] \arrow[dr, "y_{6i+5}"'] & & \\
 & t_{i,3,3} \arrow[r, "X_{i_2}"'] & t_{i,3,4}
\end{tikzcd}$$

Finally, to connect the introduced gadgets $T_{0,0}, \dots, T_{m-1,3}$, H and G, the TS A has for all $j \in \{0, \dots, 4m\}$, the edge $\bot_j \xrightarrow{\oplus_j} \bot_{j+1}$; for all $i \in \{0, \dots, m-1\}$ and all $j \in \{0, \dots, 3\}$, the edge $\bot_{4i+j} \xrightarrow{\ominus_{4i+j}} t_{i,j,0}$; and the edges $\bot_{4m} \xrightarrow{\ominus_{4m}} h_0$ and $\bot_{4m+1} \xrightarrow{\ominus_{4m+1}} g$. Eventually, A can be sketched as follows:

$$\begin{tikzcd}
\bot_0 \arrow[r, "\oplus_0"] \arrow[d, "\ominus_0"'] & \bot_1 \arrow[r, "\oplus_1"] \arrow[d, "\ominus_1"'] & \bot_2 \arrow[r, "\oplus_2"] \arrow[d, "\ominus_2"'] & \bot_3 \arrow[d, "\ominus_3"'] & \cdots & \bot_{4m-4} \arrow[r, "\oplus_{4m-4}"] \arrow[d, "\ominus_{4m-4}"'] & \bot_{4m-3} \arrow[r, "\oplus_{4m-3}"] \arrow[d, "\ominus_{4m-3}"'] & \bot_{4m-2} \arrow[r, "\oplus_{4m-2}"] \arrow[d, "\ominus_{4m-2}"'] & \bot_{4m-1} \arrow[r, "\oplus_{4m-1}"] \arrow[d, "\ominus_{4m-1}"'] & \bot_{4m} \arrow[r, "\oplus_{4m}"] \arrow[d, "\ominus_{4m}"'] & \bot_{4m+1} \arrow[d, "\ominus_{4m+1}"'] \\
T_{0,0} & T_{0,1} & T_{0,2} & T_{0,3} & \cdots & T_{m-1,0} & T_{m-1,1} & T_{m-1,2} & T_{m-1,3} & H & G
\end{tikzcd}$$

Having just introduced the gadgets of A, we proceed by providing their functionality. To do so, we need the statement of the following lemma:

Lemma 13. *Let τ be a Boolean type of nets such that $\mathsf{nop} \notin \tau$, and let A be a TS and $P_0 = s_0 \xrightarrow{e_1} \dots \xrightarrow{e_m} s_m$ and $P_1 = q_0 \xrightarrow{e_1} \dots \xrightarrow{e_m} q_m$ be two distinct acyclic directed paths of A that both apply the same sequence $e_1 \dots e_m$ of events. If $R = (sup, sig)$ is a τ-region of A such that $sup(s_m) \neq sup(q_m)$, then $sig(e_i) = \mathsf{swap}$ for all $i \in \{1, \dots, m\}$.*

Proof. 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 $sig(e_m) = \mathsf{swap}$. 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. $\square$

Let τ be a type of nets in correspondence to Theorem 2. The gadget G provides the ESSP atom $\alpha = (k, g_{24m+6})$ and the events $v_0, \dots, v_{8m+1}$. While the events k and $v_0, \dots, v_{8m-1}$ occur also outside of H and G, the other events of these gadgets do not. The aim of H and G is to ensure that any τ-region $R = (sup, sig)$ that solves α, which immediately implies $sig(k) = \mathsf{inp}$, satisfies $sig(v_j) = \mathsf{swap}$ for all $j \in \{0, \dots, 8m-1\}$. We prove that H and G fulfill this functionality: Since R solves α, we have $sig(k) = \mathsf{inp}$ and $sup(h_{24m+6}) = 1$ and $sup(g_{24m+6}) = 0$. By Lemma 13, this implies $sig(u_j) = sig(v_j) = sig(w_j) = \mathsf{swap}$ for all $j \in \{0, \dots, 8m+1\}$.

In what follows, we introduce the functionality of the remaining gadgets $T_{0,0}, \dots, T_{m-1,3}$ and particularly argue that R reveals a one-in-three model of φ.

Let $i \in \{0, \dots, m-1\}$ be arbitrary but fixed. The intuition behind $T_{i,0}, \dots, T_{i,3}$ is to ensure that there is *exactly one* variable event $X \in \{X_{i,0}, X_{i,1}, X_{i,2}\}$ such that $sig(X) = \mathsf{set}$. Since i was arbitrary, the set $M = \{X \in V(\varphi) \mid sig(X) = \mathsf{set}\}$ then selects exactly one variable per clause ζ_i for all $i \in \{0, \dots, m-1\}$. Consequently, M defines a one-in-three model of φ.

In what follows, we formally prove the announced functionality of $T_{i,0}, \dots, T_{i,3}$. Let $P_i = t_{i,0,1} \xrightarrow{v_{8i}} \dots \xrightarrow{v_{8i+4}} t_{i,0,9}$ and, let P_i^R be the image of P_i under R in τ, that is, $P_i^R = sup(t_{i,0,1}) \xrightarrow{sig(v_{8i})} \dots \xrightarrow{sig(v_{8i+4})} sup(t_{i,0,9})$. Since $sig(k) = \mathsf{inp}$, we have $sup(t_{i,0,1}) = 0$ and $sup(t_{i,0,9}) = 1$. Thus, P_i^R is a directed path from 0 to 1 in τ and the number of state changes between 0 and 1 on P_i^R is odd. Recall that $sig(v_{8i+j}) = \mathsf{swap}$ for all $j \in \{0, \dots, 4\}$. Thus, $v_{8i}, \dots, v_{8i+4}$ already perform an odd number of state changes on P_i^R. This implies that there is either *exactly one* $X \in \{X_{i,0}, X_{i,1}, X_{i,2}\}$ such that $s \xrightarrow{X} s' \in P_i$ and $sup(s) = sup(s')$, or *for all* $X \in \{X_{i,0}, X_{i,1}, X_{i,2}\}$ holds that if $s \xrightarrow{X} s' \in P_i$, then $sup(s) = sup(s')$. (Otherwise we would have an even number of state changes on P_i^R, which is a contradiction.) In what follows, we argue that the former case is true. Assume, for a contradiction, that $sup(t_{i,0,2}) = sup(t_{i,0,3})$ and $sup(t_{i,0,6}) = sup(t_{i,0,7})$. By $sup(t_{i,0,1}) = 0$ and $sig(v_{8i}) = \mathsf{swap}$, we get $sup(t_{i,0,2}) = sup(t_{i,0,3}) = 1$. This implies $sig(X_{i_0}) = \mathsf{set}$. By $sup(t_{i,0,9}) = 1$ and $sig(v_{8i+3}) = sig(v_{8i+4}) = \mathsf{swap}$, we get $sup(t_{i,0,6}) = sup(t_{i,0,7}) = 1$, which implies $sig(X_{i_2}) = \mathsf{set}$. Moreover, by $sig(X_{i_0}) = \mathsf{set}$ and $sig(v_{8i+6})$, we get $sup(t_{i,2,2}) = 1$ and $sup(t_{i,2,4}) = 0$, respectively. Since $\xrightarrow{X_{i_2}} t_{i,2,4}$, this contradicts $sig(X_{i_2}) = \mathsf{set}$. Altogether, by the former discussion, this implies that there is exactly one event $X \in \{X_{i,0}, X_{i,1}, X_{i,2}\}$ such that its occurrence $s \xrightarrow{X} s' \in P_i$ preserves the support, that is, $sup(s) = sup(s')$. In the following, we argue that this implies $sig(X) = \mathsf{set}$ and $sig(Y) \neq \mathsf{set}$ for all $Y \in \{X_{i,0}, X_{i,1}, X_{i,2}\} \setminus \{X\}$. If $sup(t_{i,0,1}) = sup(t_{i,0,2})$, then, as already discussed, we obtain $sig(X_{i_0}) = \mathsf{set}$. By $sig(X_{i_0}) = \mathsf{set}$ and $sig(v_{8i+5}) = sig(v_{8i+6}) = \mathsf{swap}$, we get $sup(t_{i,1,2}) = sup(t_{i,2,2}) = 1$ and $sup(t_{i,1,4}) = sup(t_{i,2,4}) = 0$, respectively. Clearly, by $\xrightarrow{X_{i_1}} t_{i,1,4}$ and $\xrightarrow{X_{i_2}} t_{i,2,4}$, this yields $sig(X_{i_1}) \neq \mathsf{set}$ and $sig(X_{i_2}) \neq \mathsf{set}$, respectively. Similarly, the assumption $sup(t_{i,0,4}) = sup(t_{i,0,5})$ yields $sig(X_{i_1}) = \mathsf{set}$ and $sig(X_{i_0}), sig(X_{i_2}) \neq \mathsf{set}$, and $sup(t_{i,0,6}) = sup(t_{i,0,7})$ leads to $sig(X_{i_2}) = \mathsf{set}$ and $sig(X_{i_0}), sig(X_{i_1}) \neq \mathsf{set}$.

Thus if $R = (sup, sig)$ is a τ-region that solves α and satisfies $sig(k) = \mathsf{inp}$, then $M = \{X \in V(\varphi) \mid sig(X) = \mathsf{set}\}$ is a one-in-three model of φ.

To complete the proof of Theorem 2, it remains to argue that if M has a one-in-three model, then α is τ-solvable. This is the content of the following paragraph.

A One-in-three Model M of φ Implies the τ-Solvability of A.

Let $\tau = \{\mathsf{inp}, \mathsf{res}, \mathsf{set}, \mathsf{swap}\}$. We argue that α is τ-solvable by presenting a solving region $R = (sup, sig)$. Due to space restrictions, we define R implicitly, that is, we explicitly define $sup(\bot_0)$ and $sig(e)$ for all $e \in E(A)$. By construction, any state $s_n \in S(A)$ is reachable via an unique path $\bot_0 \xrightarrow{e_1} \dots \xrightarrow{e_n} s_n$ of A.

Thus, $sup(s_n)$ can be computed inductively by $sup(s_i) = \delta_\tau(sup(s_{i-1}), e_i)$ for all $i \in \{1, \dots, n\}$.

For abbreviation, we define $Y = \{y_0, \dots, y_{6m-1}\}, \oplus = \{\oplus_0, \dots, \oplus_{4m-1}\}$, $\ominus = \{\ominus_0, \dots, \ominus_{4m+1}\}$, $U = \{u_0, \dots, v_{8m+1}\}$, $V = \{v_0, \dots, v_{8m+1}\}$ and $W = \{w_0, \dots, w_{8m+1}\}$ and $UVW = U \cup V \cup W$.

Let M be a one-in-three model of φ. The following τ-region $R = (sup, sig)$ solves (k, g_{24m+6}): $sup(\bot_0) = 0$; for all $e \in \oplus$, we define $sig(e) = \mathsf{res}$, which yields $sup(\bot_0) = \dots = sup(\bot_{4m+1}) = 0$. Moreover, for all $e \in \{\ominus_{4i} \mid i \in \{0, \dots, m\}\} \cup \{\ominus_{4m+1}\}$, we let $sig(e) = \mathsf{swap}$, which ensures $sup(t_{i,0,0}) = sup(h_0) = sup(g) = 1$ for all $i \in \{0, \dots, m-1\}$. Furthermore, for all $e \in \ominus \setminus \{\ominus_{4i} \mid i \in \{0, \dots, m\}\}$, we define $sig(e) = \mathsf{res}$, which yields $sup(t_{i,1,0}) = \dots = sup(t_{i,3,0}) = 0$ for all $i \in \{0, \dots, m-1\}$. Additionally, for all $e \in UVW \cup (V(\varphi) \setminus M)$, we let $sig(e) = \mathsf{swap}$, and, for all $e \in M$, we define $sig(e) = \mathsf{set}$. Finally, for all $i \in \{0, \dots, m-1\}$ and all $e \in \{y_{6i}, \dots, y_{6i+5}\}$, if $e = y_{6i+5}$ and $X_{i_0} \in M$, then $sig(e) = \mathsf{res}$; if $e = y_{6i+3}$ and $X_{i_1} \in M$, then $sig(e) = \mathsf{res}$; if $e = y_{6i+1}$ and $X_{i_2} \in M$, then $sig(e) = \mathsf{res}$; otherwise, $sig(e) = \mathsf{swap}$. The colored areas of the gadgets H, G and $T_{i,0}, \dots, T_{i,3}$ as introduced in the latter section sketches R where $X_{i_0} \in M$; grey colored states are mapped to 1 and the others to 0. Moreover, the following figures sketch the cases $X_{i_1} \in M$ and $X_{i_2} \in M$ (in the given order):

$$\begin{tikzcd}
\boxed{t_{i,0,0}} \arrow[r, "k"] & t_{i,0,1} \arrow[r, "v_{8i}"] & \boxed{t_{i,0,2}} \arrow[r, "X_{i_0}"] & t_{i,0,3} \arrow[r, "v_{8i+1}"] & \boxed{t_{i,0,4}} \arrow[r, "X_{i_1}"] & \boxed{t_{i,0,5}} \arrow[r, "v_{8i+2}"] & t_{i,0,6} \arrow[r, "X_{i_2}"] & \boxed{t_{i,0,7}} \arrow[d, "v_{8i+3}"] \\
 & & & & & t_{i,0,10} & \boxed{t_{i,0,9}} \arrow[l, "k"'] & t_{i,0,8} \arrow[l, "v_{8i+4}"']
\end{tikzcd}$$

$$\begin{tikzcd}
 & \boxed{t_{i,1,1}} \arrow[r, "X_{i_0}"] & t_{i,1,2} \arrow[d, "v_{8i+5}"] \\
t_{i,1,0} \arrow[ur, "y_{6i}"] \arrow[dr, "y_{6i+1}"'] & & \\
 & \boxed{t_{i,1,3}} \arrow[r, "X_{i_1}"'] & \boxed{t_{i,1,4}}
\end{tikzcd}
\quad
\begin{tikzcd}
 & \boxed{t_{i,2,1}} \arrow[r, "X_{i_0}"] & t_{i,2,2} \arrow[d, "v_{8i+6}"] \\
t_{i,2,0} \arrow[ur, "y_{6i+2}"] \arrow[dr, "y_{6i+3}"'] & & \\
 & t_{i,2,3} \arrow[r, "X_{i_2}"'] & \boxed{t_{i,2,4}}
\end{tikzcd}
\quad
\begin{tikzcd}
 & \boxed{t_{i,3,1}} \arrow[r, "X_{i_1}"] & \boxed{t_{i,3,2}} \arrow[d, "v_{8i+7}"] \\
t_{i,3,0} \arrow[ur, "y_{6i+4}"] \arrow[dr, "y_{6i+5}"'] & & \\
 & \boxed{t_{i,3,3}} \arrow[r, "X_{i_2}"'] & t_{i,3,4}
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{t_{i,0,0}} \arrow[r, "k"] & t_{i,0,1} \arrow[r, "v_{8i}"] & \boxed{t_{i,0,2}} \arrow[r, "X_{i_0}"] & t_{i,0,3} \arrow[r, "v_{8i+1}"] & \boxed{t_{i,0,4}} \arrow[r, "X_{i_1}"] & t_{i,0,5} \arrow[r, "v_{8i+2}"] & \boxed{t_{i,0,6}} \arrow[r, "X_{i_2}"] & \boxed{t_{i,0,7}} \arrow[d, "v_{8i+3}"] \\
 & & & & & t_{i,0,10} & \boxed{t_{i,0,9}} \arrow[l, "k"'] & t_{i,0,8} \arrow[l, "v_{8i+4}"']
\end{tikzcd}$$

$$\begin{tikzcd}
 & \boxed{t_{i,1,1}} \arrow[r, "X_{i_0}"] & t_{i,1,2} \arrow[d, "v_{8i+5}"] \\
t_{i,1,0} \arrow[ur, "y_{6i}"] \arrow[dr, "y_{6i+1}"'] & & \\
 & t_{i,1,3} \arrow[r, "X_{i_1}"'] & \boxed{t_{i,1,4}}
\end{tikzcd}
\quad
\begin{tikzcd}
 & \boxed{t_{i,2,1}} \arrow[r, "X_{i_0}"] & \boxed{t_{i,2,2}} \arrow[d, "v_{8i+6}"] \\
t_{i,2,0} \arrow[ur, "y_{6i+2}"] \arrow[dr, "y_{6i+3}"'] & & \\
 & t_{i,2,3} \arrow[r, "X_{i_2}"'] & \boxed{t_{i,2,4}}
\end{tikzcd}
\quad
\begin{tikzcd}
 & \boxed{t_{i,3,1}} \arrow[r, "X_{i_1}"] & t_{i,3,2} \arrow[d, "v_{8i+7}"] \\
t_{i,3,0} \arrow[ur, "y_{6i+4}"] \arrow[dr, "y_{6i+5}"'] & & \\
 & \boxed{t_{i,3,3}} \arrow[r, "X_{i_2}"'] & \boxed{t_{i,3,4}}
\end{tikzcd}$$

5 Conclusion

In this paper, we completely characterize the computational complexity of Boolean τ-synthesis for types τ that contain neither nop nor swap and show that all these types allow for a polynomial time synthesis procedure. Moreover, we give a first hint that the absence of nop might not guarantee that Boolean synthesis is polynomial. To do so, we show that the single instance ESSP problem is NP-complete if $\tau = \{\mathsf{inp}, \mathsf{res}, \mathsf{set}, \mathsf{swap}\}$. It remains future work to characterize the computational complexity of τ-synthesis for the remaining nop-free Boolean types τ.

References

1. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series, Springer (2015). https://doi.org/10.1007/978-3-662-47967-4
2. Badouel, E., Darondeau, P.: Trace nets and process automata. Acta Inf. **32**(7), 647–679 (1995). https://doi.org/10.1007/BF01186645, https://doi.org/10.1007/BF01186645
3. 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, https://doi.org/10.1007/s00236-012-0170-2
4. Montanari, U., Rossi, F.: Contextual nets. Acta Inf. **32**(6), 545–596 (1995). https://doi.org/10.1007/BF01178907, https://doi.org/10.1007/BF01178907
5. Moore, C., Robson, J.M.: Hard tiling problems with simple tiles. Discrete & Computational Geometry **26**(4), 573–590 (2001). https://doi.org/10.1007/s00454-001-0047-6
6. Pietkiewicz-Koutny, M.: Transition systems of elementary net systems with inhibitor arcs. In: Azéma, P., Balbo, G. (eds.) Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1248, pp. 310–327. Springer (1997). https://doi.org/10.1007/3-540-63139-9_43, https://doi.org/10.1007/3-540-63139-9_43
7. Schmitt, V.: Flip-flop nets. In: STACS. Lecture Notes in Computer Science, vol. 1046, pp. 517–528. Springer (1996). https://doi.org/10.1007/3-540-60922-9_42
8. Tredup, R.: The complexity of synthesizing nop-equipped boolean nets from g-bounded inputs (technical report) (2019), invited to ToPNoC 2020
9. Tredup, R.: Tracking down the bad guys: Reset and set make feasibility for flip-flop net derivatives np-complete. In: ICE. EPTCS, vol. 304, pp. 20–37 (2019). https://doi.org/10.4204/EPTCS.304.2
10. Tredup, R., Rosenke, C.: The complexity of synthesis for 43 boolean petri net types. In: Theory and Applications of Models of Computation. Theoretical Computer Science and General Issues, vol. 11436. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-14812-6
11. Tredup, R., Rosenke, C.: On the hardness of synthesizing boolean nets. In: ATAED@Petri Nets/ACSD. CEUR Workshop Proceedings, vol. 2371, pp. 71–86. CEUR-WS.org (2019)