K-Step Opacity Benchmarks for Discrete Event Systems ★

Jiří Balun, Tomáš Masopust, and Petr Osička

Faculty of Science, Palacky University Olomouc, Czechia

Abstract

Opacity is a generic framework expressing secret in discrete-event systems. Many notions of opacity have been discussed in the literature, including current-state opacity (CSO), k-step opacity (k-SO), and strong k-step opacity (k-SSO). We design new algorithms for the verification of k-SO and k-SSO based on reductions of k-SO to CSO and of k-SSO to k-SO, respectively, and experimentally compare their performance with the existing algorithms. In particular, for CSO, we use the classical algorithm based on the observer construction and an algorithm based on a reduction to the antichain-based language inclusion. For k-SO, we compare our algorithms with the two-way observer of Yin and Lafortune, with the reduction-based algorithm of Wintenberg et al., and with our projected-automata-base algorithm. For k-SSO, we compare our algorithms with the reduction-based algorithm of Wintenberg et al. and with the algorithm of Han et al.. For the comparisons, we use extensive benchmarks with almost twelve thousand instances based on real data.

Key words: Discrete event systems, finite automata, opacity, reductions, complexity, benchmarks.

1 Introduction

Opacity is a property asking whether a system prevents an intruder from revealing system's secret. The intruder is modeled as a passive observer with complete knowledge of the structure of the system, but with only limited observations of its behavior. Based on observations, the intruder estimates system's behavior, and the system is opaque if the intruder never reveals the secret; that is, for any secret behavior, there is a non-secret behavior that looks the same to the intruder. The secret is modeled either as a set of secret states or as a set of secret behaviors. The former results in *state-based opacity* of Bryans et al. (2004, 2008), the latter results in *language-based opacity* of Badouel et al. (2007) and Dubreil et al. (2008); see the overview by Jacob et al. (2016).

Several notions of opacity have been discussed in the literature, including current-state opacity (CSO), initial-state opacity (ISO), language-based opacity (LBO), k-step opacity (k-SO), and strong k-step opacity (k-SSO). Initial-state opacity prevents the intruder from revealing whether the system started in a secret state. Current-state opacity, on the other hand, prevents the intruder from revealing whether the current state of the system is secret. The intruder may, however, realize that the system was in a secret state in the past. This problem led to the introduction of k-SO requiring that the intruder cannot ascertain the secret in the current and k subsequent steps (Bryans et al., 2004; Saboori and Hadjicostis, 2007, 2012). Nonetheless, k-SO still allows the intruder to realize that the system previously visited a secret state, although it keeps the exact time when that happened secret. This issue further motivated Falcone and Marchand (2015) to introduce strong k-SO; a similar property was already mentioned by Saboori and Hadjicostis (2012) under the name of trajectory-based k-step opacity.

Deciding opacity is a PSpace-complete problem, and the existing algorithms are exponential. Balun et al. (2023) showed that, unless the strong exponential time hypothesis of Impagliazzo and Paturi (2001) fails, there is no subexponential-time algorithm deciding opacity in time $O^*(2^{n/(2+\varepsilon)})$, for any $\varepsilon > 0$, where n is the number of states of the input automaton.

In this paper, we design new algorithms to verify k-SO and k-SSO based on reductions of k-SO to CSO and of k-SSO to k-SO. Since the worst-case time complexity is basically the same for all the algorithms, we are interested in the experimental comparison of their performance. To this end, we created benchmarks consisting of 11710 instances of nondeterministic finite automata (NFAs) that are mostly based on real data (see Section 3 for details). In our experiments, every tool was given five minutes to solve each of the instances.

To verify CSO, we compare the classical algorithm based on the observer construction with an algorithm based on a re-

★ Corresponding author: T. Masopust.
Email addresses: jiri.balun@upol.cz (Jiří Balun), tomas.masopust@upol.cz (Tomáš Masopust), petr.osicka@upol.cz (Petr Osička).

duction to the antichain-based language inclusion algorithm of Wulf et al. (2006). The experimental results show that, within the five-minute time limit, both algorithms perform very similarly, see Section 4.

For k-SO, we compare the two-way observer of Yin and Lafortune (2017), the reduction-based algorithm of Wintenberg et al. (2022), and the projected-automata-based algorithm of Balun and Masopust (2023), with two new algorithms based on a reduction of k-SO to CSO. We ran the experiments for $k = 5$, $k = 500$, and $k = 50000$. For $k = 500$, the algorithm of Wintenberg et al. (2022) timed out for more than half of the instances, and therefore we excluded it from further comparisons. For a similar reason, we further excluded one of our new algorithms from comparisons for $k = 50000$. From 11710 instances of the k-step opacity problem, the tools altogether solved more than 11600 instances within the given time limit, for all three choices of k. Our projected-automata-based algorithm was the fastest for the most instances, and used the minimum total time for $k \in \{500, 50000\}$. For $k = 5$, our reduction-based algorithm was the fastest and solved the most instances; in fact, our new reduction-based algorithm was the fastest algorithm for negative instances and all cases of k; see Section 5.4 for more details.

For k-SSO, we first adjust the algorithm of Han et al. (2022) so that it can verify k-SSO of Falcone and Marchand (2015). We then compare this algorithm with an algorithm of Wintenberg et al. (2022) and with two new algorithms based on our reduction of k-SSO to k-SO. We again ran the experiments for $k \in \{5, 500, 50000\}$. For small k, the reduction-based algorithm of Wintenberg et al. (2022) performs the best. However, for larger k, it takes too much time and is outperformed by all the other algorithms. In these cases, the algorithm of Han et al. (2022) and our new reduction-based algorithm perform comparably well from the viewpoint of solved instances, though our algorithm takes for more than six hours less of total time, see Section 6.2 for details.

2 Preliminaries

We assume that the reader is familiar with automata theory and discrete-event systems, see Hopcroft et al. (2006) and Cassandras and Lafortune (2021). For a set S, $|S|$ denotes the cardinality of S, and 2^S denotes the power set of S. An alphabet Σ is a finite nonempty set of events. A string over Σ is a sequence of events from Σ; the empty string is denoted by ε. The set of all finite strings over Σ is denoted by Σ^*. A language L over Σ is a subset of Σ^*. For a string $u \in \Sigma^*$, $|u|$ denotes the length of u.

A *nondeterministic finite automaton* (NFA) over an alphabet Σ is a quintuple $G = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, $I \subseteq Q$ is a nonempty set of initial states, $F \subseteq Q$ is a set of marked states, and $\delta\colon Q \times \Sigma \to 2^Q$ is a transition function that can be extended to the domain $2^Q \times \Sigma^*$ by induction. For any sets $Q_0, Q_F \subseteq Q$, we define the language $L_m(G, Q_0, Q_F) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap Q_F \neq \emptyset\}$ of strings starting in a state of Q_0 and ending in a state of Q_F. Then, the *marked* and *generated* languages of G are $L_m(G) = L_m(G, I, F)$ and $L(G) = L_m(G, I, Q)$, respectively.

An NFA $G = (Q, \Sigma, \delta, I, F)$ is *deterministic* (DFA) if $|I| = 1$ and $|\delta(q, a)| \leq 1$ for every $q \in Q$ and $a \in \Sigma$. In this case, we identify the singleton $I = \{q_0\}$ with its element, and simply write $G = (Q, \Sigma, \delta, q_0, F)$ instead of $G = (Q, \Sigma, \delta, \{q_0\}, F)$.

A *discrete-event system* (DES) G over Σ is an NFA over Σ together with the partition of Σ into Σ_o and Σ_{uo} of *observable* and *unobservable events*, respectively. If G is a DFA, we say that the DES is *deterministic*. If the marked states are irrelevant, we omit them and write $G = (Q, \Sigma, \delta, I)$.

A *run* of a DES $G = (Q, \Sigma, \delta, I)$ from a state q_0 under a string $w = a_1 a_2 \cdots a_n \in \Sigma^*$ is a sequence

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$$

such that $q_k \in \delta(q_{k-1}, a_k)$, for $k = 1, \ldots, n$. We abbreviate the run as $q_0 \xrightarrow{w} q_n$, or simply as $q_0 \xrightarrow{w}$ if q_n is not important. The run is *non-secret* with respect to a set $Q_S \subseteq Q$ of *secret* states if it does not contain any secret state, that is, $q_i \in Q - Q_S$ for $i = 0, 1, \ldots, n$; otherwise, the run is *secret*. The run is *initial* if q_0 is an initial state. A *subrun* of the run $q_0 \xrightarrow{w} q_n$ is the run $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{i-1} \xrightarrow{a_i} q_i$, for some $i \leq n$.

The state estimation of a DES G over Σ is modeled by *projection* $P\colon \Sigma^* \to \Sigma_o^*$ from the alphabet of G to the set of observable events Σ_o, which is a morphism for concatenation defined by $P(a) = \varepsilon$ if $a \in \Sigma_{uo}$, and $P(a) = a$ if $a \in \Sigma_o$. The action of P on a string $a_1 a_2 \cdots a_n$ is to erase all unobservable events, that is, $P(a_1 a_2 \cdots a_n) = P(a_1)P(a_2) \cdots P(a_n)$. The definition can be readily extended to languages.

The *observer* of G is a DFA obtained from G by replacing every transition (p, a, q) by $(p, P(a), q)$, and by the standard subset construction, see Cassandras and Lafortune (2021). The observer of G has exponentially many states compared with G, see Wong (1998), or Jirásková and Masopust (2012) for more details.

3 Benchmarks

For experimental comparisons, we created extensive benchmarks consisting of 11710 instances of nondeterministic finite automata (NFAs) that are suitable for the CSO, k-SO, and k-SSO verification. Most of the automata are based on real data: there are automata comming from abstract regular model-checking, [1] automata based on the elevator example, [2] and automata based on our work on supervisory

[1] https://github.com/ondrik/automata-benchmarks

[2] https://fgdes.tf.fau.de/faudes/luafaudes/elevator_synthesis.html

control of a patient table of an MRI scanner of Theunissen et al. (2014). The dataset also includes hard instances for a tool we use for language inclusion, and some random difficult instances. The instances are selected so that circa half of them is positive and half of them is negative for all of CSO, k-SO, and k-SSO properties.

To focus on the performance of the core of the algorithms, we consider NFAs without unobservable events, which is no restriction because every automaton with unobservable events can be translated to an equivalent NFA without unobservable events (although this needs to be done in a more careful way than the standard elimination of ε transitions). On the other hand, we include the parsing of the input automaton to the measurements. This time is basically the same for all the tools, and hence it does not disqualify any algorithm and, in fact, better corresponds to the time required by the tool to solve the instance.

The instances have only two types of states: secret states, modeled by marked states, and non-secret states, modeled by non-marked states. In other words, the instances do not consider neutral states; indeed, neutral states may be used in reduction-based algorithms, which is, in fact, the case. The number of states of the instances ranges from two to 278372, with 2862,14 states in average, and the number of transitions ranges from four to 95937444, with 82415 transitions in average.

The instances are stored in a `.gen` format used by the C++ library `libFAUDES`,[3] which seems suitable, because the `.gen` format is XML-based, and hence it is readable by humans as well as by computers. We implemented all the algorithms using the C++ libraries `libFAUDES`[3] and `Limi`[4] without additional optimizations. In particular, our implementation of the two-way observer does not use the depth-first search of the observer, but rather the search implemented in `libFAUDES`. We do not know whether it has any impact on the performance of the algorithm. Also, we restricted our implementations to work for our data, and hence they may not work for automata with unobservable events or with states that are neither secret nor non-secret.

The experiments were run on an Ubuntu 22.04.4 LTS virtual machine with 30 Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz processors and 250 GB memory, using the `parallel` tool by Tange (2024). Every tool was given five minutes to solve an instance. The benchmarks and the results of our experiments are available at `https://apollo.inf.upol.cz:81/masopust/k-so-opacity-benchmarks`.

4 Comparison of CSO-verifying algorithms

Current-state opacity (CSO) is the fundamental notion of opacity. It prevents the intruder from revealing whether the current state of the system is secret.

Definition 1 (CSO)**.** A DES $G = (Q, \Sigma, \delta, I)$ is current-state opaque (CSO) with respect to secret states $Q_S \subseteq Q$, non-secret states $Q_{NS} \subseteq Q$, and projection $P\colon \Sigma^* \to \Sigma_o^*$ if for every $s \in L(G)$ with $\delta(I, s) \cap Q_S \neq \emptyset$, there is $s' \in L(G)$ such that $P(s) = P(s')$ and $\delta(I, s') \cap Q_{NS} \neq \emptyset$.

The classical algorithm to verify CSO is based on the construction of the observer, see Algorithm 1. In our implementation, during the construction of the observer, the algorithm verifies whether every state containing a secret state also contains a non-secret state. If this condition is violated, the algorithm returns `false` and terminates.

Algorithm 1. Classical verification of current-state opacity

Require: A DES $G = (Q, \Sigma, \delta, I)$, $Q_S, Q_{NS} \subseteq Q$, $\Sigma_o \subseteq \Sigma$.
Ensure: `true` if and only if G is current-state opaque with respect to Q_S, Q_{NS}, and $P\colon \Sigma^* \to \Sigma_o^*$.

```
Construct the observer G^obs of G step by step.
for every newly generated state X of G^obs do
    if X ∩ Q_S ≠ ∅ and X ∩ Q_NS = ∅ then
        return false
    end if
end for
return true
```

We implemented Algorithm 1 as the CSO_OBS tool and compared it with the algorithm based on the reduction of CSO to language inclusion of Wu and Lafortune (2013). Our language-inclusion verification uses the antichain algorithm of Wulf et al. (2006) implemented in the C++ library `Limi` of Černý et al. (2017); we therefore call the tool CSO_LIMI.

Within five minutes, CSO_OBS solved 11632 instances and CSO_LIMI solved 11604 instances. CSO_OBS was faster for 3168 instances, CSO_LIMI was faster for 8391 instances; 78 instances took both tools the same time. Overall, both tools solved 11637 instances in the given time limit. From the solved instances, 33 were solved only by CSO_OBS and 6 were solved only by CSO_LIMI, whereas the other tool timed out. Furthermore, 5826 of the solved instances were positive and 5811 negative. The results are summarized in Table 1; the total time (rounded up to whole seconds) counts the time to solve the instance or 300 if the tool timed out, while the total time for positive/negative instances counts these times only for instances that were solved at least by one of the tools.

5 K-step opacity

In this section, we design an algorithm to verify k-step opacity (k-SO) that is based on a reduction of k-SO to CSO. Considering the CSO_OBS and CSO_LIMI tools, we obtain two tools, KSO_OBS and KSO_LIMI, which we experimentally compare with the two-way observer of Yin and Lafortune (2017), one of the algorithms of Wintenberg et al. (2022), and the algorithm of Balun and Masopust (2023).

[3] https://github.com/FGDES/libFAUDES
[4] https://github.com/thorstent/Limi

	CSO_OBS	CSO_LIMI
# of instances solved	11632	11604
# of instances solved faster	3168	8391
# of instances solved only by	33	6
# of solved positive instances	5822	5795
# of solved negative instances	5810	5809
total time	29832	65820
total time positive solved	4776	42452
total time negative solved	2856	1168

Table 1
Comparing CSO_OBS and CSO_LIMI within a five-minute time limit.

We first recall the definition of k-SO. To this end, we take the set $\mathbb{N}$ of all non-negative integers, and extend it with its limit to $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$.

Definition 2 (k-SO)**.** Given a DES $G = (Q, \Sigma, \delta, I)$ and a number $k \in \mathbb{N}_\infty$. We say that G is k-step opaque (k-SO) with respect to secret states $Q_S \subseteq Q$, non-secret states $Q_{NS} \subseteq Q$, and projection $P\colon \Sigma^* \to \Sigma_o^*$ if for every string $st \in L(G)$ with $|P(t)| \leq k$ and $\delta(\delta(I, s) \cap Q_S, t) \neq \emptyset$, there exists $s't' \in L(G)$ such that $P(s) = P(s')$, $P(t) = P(t')$, and $\delta(\delta(I, s') \cap Q_{NS}, t') \neq \emptyset$.

Intuitively, for every secret-revealing string s, and its extension t with at most k observable events, there is a string s' and its extension t' such that the intruder cannot distinguish between s and s' in the current and k subsequent steps.

Note that CSO coincides with 0-SO, and that ∞-SO is known as *infinite-step opacity* (INSO) in the literature, see, e.g., Saboori and Hadjicostis (2012). Moreover, Yin and Lafortune (2017) have shown that, for an n-state DES, infinite-step opacity coincides with $(2^n - 2)$-step opacity; therefore, without loss of generality, we may consider k-SO only for $k \leq 2^n - 2$ (however, we do not implement it in the tools).

5.1 Reducing k-SO to CSO: the first step

We now discuss, in two steps, our reduction of k-SO to CSO. In the first step, we reduce INSO to CSO.

Construction 3 (INSO to CSO)**.** For a DES $G = (Q, \Sigma, \delta, I)$ with secret states Q_S, non-secret states Q_{NS}, and observable events Σ_o, we construct the DES

$$G' = (Q \cup Q^+ \cup Q^-, \Sigma \cup \{@\}, \delta', I)$$

by creating two disjoint copies of G, denoted by G^+ and G^-, with the corresponding state sets $Q^+ = \{q^+ \mid q \in Q\}$ and $Q^- = \{q^- \mid q \in Q\}$. The new event @ connects G to G^+ and G^- via transitions

- $(q, @, q^+)$, for every $q \in Q_S$, and
- $(q, @, q^-)$, for every $q \in Q_{NS}$.

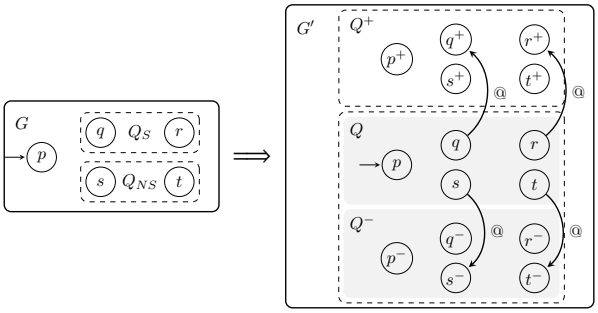

Figure 1. First step of the reduction of k-SO to CSO.

The set of secret states of G' is Q^+, and the set of non-secret states is $Q \cup Q^-$, see Figure 1. The set of observable events of G' is $\Sigma_o \cup \{@\}$. ◇

The following result shows that we can verify INSO of G by checking CSO of G'.

Theorem 4. *A DES G is INSO with respect to Q_S, Q_{NS}, and P if and only if G' obtained from G by Construction 3 is CSO with respect to $Q'_S = Q^+$, $Q'_{NS} = Q \cup Q^-$, and $P'\colon (\Sigma \cup \{@\})^* \to (\Sigma_o \cup \{@\})^*$.*

Proof. Assume that G is INSO. To show that G' is CSO, let $w \in L(G')$ be such that $\delta'(I, w) \cap Q'_S \neq \emptyset$. Since the set of secret states of G' is Q^+, the string w is of the form $w_1@w_2$, and there is a state $q \in \delta(I, w_1) \cap Q_S$ in G such that its copy q^+ belongs to $\delta'(I, w_1@) \cap Q'_S$ in G', and the suffix w_2 is generated from q^+. Then w_2 can be generated from q in G, and hence $\delta(\delta(I, w_1) \cap Q_S, w_2) \neq \emptyset$. The INSO property of G implies that there is $w'_1 w'_2 \in L(G)$ such that $P(w_1) = P(w'_1)$, $P(w_2) = P(w'_2)$, and $\delta(\delta(I, w'_1) \cap Q_{NS}, w'_2) = X \neq \emptyset$. Let $w' = w'_1@w'_2$. Then, $P'(w) = P'(w')$ and $\delta'(\delta'(I, w'_1@) \cap Q'_{NS}, w'_2) = \{x^- \mid x \in X\} \neq \emptyset$, which was to be shown.

On the other hand, if G is not INSO, then there is a string $st \in L(G)$ and a state q such that $q \in \delta(\delta(I, s) \cap Q_S, t)$ and, for every string $s't' \in L(G)$ with $P(s) = P(s')$ and $P(t) = P(t')$, $\delta(\delta(I, s') \cap Q_{NS}, t') = \emptyset$. Then, for $s@t \in L(G')$, the secret state $q^+ \in \delta'(\delta'(I, s@) \cap Q'_S, t) \subseteq \delta'(I, s@t)$ and, for every $s'@t' \in L(G')$ with $P'(s@t) = P'(s'@t')$, we have $\delta'(I, s'@t') \cap Q'_{NS} = \delta'(\delta'(I, s'@) \cap Q'_{NS}, t') = \emptyset$, and hence G' is not CSO. □

To use G' to verify k-SO, we need to extend Construction 3 with a counter of observable events made from a secret state. However, rather than to use a k-state automaton to model the counter, as was done in the literature (Saboori, 2011; Balun and Masopust, 2021; Wintenberg et al., 2022), we use an automaton with $O(\log k)$ states described below.

5.2 Counter automaton

The counter automaton described in this section is a revised and simplified version of the counter automaton presented

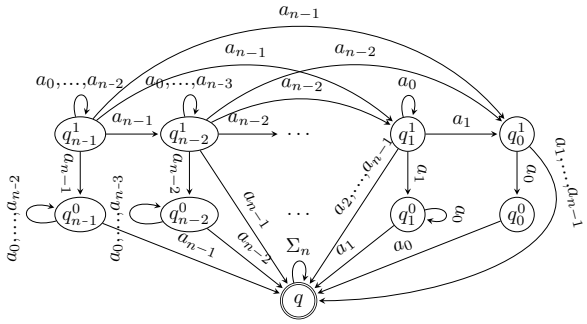

Figure 2. The NFA $\mathcal{A}_k$ of Lemma 5.

in Balun and Masopust (2022).

Lemma 5. *For every integer $k \geq 1$, there is an NFA $\mathcal{A}_k$ with $n = \lceil \log_2(k+1) \rceil$ events and $2n+1$ states, such that $\mathcal{A}_k$ marks all strings except for a unique string W_k of length k and all its prefixes.*

Proof. Let $k \geq 1$, and let $n = \lceil \log_2(k+1) \rceil$. We consider Zimin words Z_n over $\Sigma_n = \{a_0, a_1, \ldots, a_{n-1}\}$ defined by

$$Z_1 = a_0 \quad \text{and} \quad Z_i = Z_{i-1} a_{i-1} Z_{i-1} \text{ for } 1 < i \leq n$$

e.g., $Z_3 = Z_2 a_2 Z_2 = Z_1 a_1 Z_1 a_2 Z_1 a_1 Z_1 = a_0 a_1 a_0 a_2 a_0 a_1 a_0$. The length of Z_n is $2^n - 1$ (Sloan, a), and the event on the ℓth position of Z_n is a_j, where j is the number of trailing zeros in the binary representation of ℓ (Sloan, b). Since Z_n is a palindrome, the same event appears on positions ℓ and $2^n - 1 - \ell$. We denote the suffix of Z_n of length k by W_k, which is the desired string from the statement of the lemma.

Let $b_{n-1} b_{n-2} \cdots b_0$ be the binary representation of k, where the leftmost bit is the most significant; in particular, $b_{n-1} = 1$. We construct the NFA $\mathcal{A}_k = (Q, \Sigma_n, \delta, I, F)$ with the state set $Q = \{q\} \cup \{q_i^1, q_i^0 \mid i = 0, \ldots, n-1\}$ consisting of two states, q_i^1 and q_i^0, for every bit b_i of the binary representation of k, and one additional state, q, which is the only marked state, that is, $F = \{q\}$. The initial states form the set $I = \{q_{n-1}^{b_{n-1}}, q_{n-2}^{b_{n-2}}, \ldots, q_0^{b_0}\}$. The transition function δ is defined as follows, see Figure 2 and Example 7 for an illustration:

(1) For every $a \in \Sigma_n$, $(q, a, q) \in \delta$;
(2) For every state q_i^1,
  (a) $(q_i^1, a_i, q_i^0) \in \delta$;
  (b) $(q_i^1, a_j, q_i^1) \in \delta$, for $0 \leq j \leq i-1$;
  (c) $(q_i^1, a_i, q_j^1) \in \delta$, for $0 \leq j \leq i-1$;
  (d) $(q_i^1, a_j, q) \in \delta$, for $i+1 \leq j \leq n-1$;
(3) For every state q_i^0,
  (a) $(q_i^0, a_i, q) \in \delta$;
  (b) $(q_i^0, a_j, q_i^0) \in \delta$, for $0 \leq j \leq i-1$;
  (c) the other transitions are undefined.

It remains to show that $\mathcal{A}_k$ marks all strings except for prefixes of W_k. To this end, we first show that $\mathcal{A}_k$ does not mark any prefix of W_k, and then we show that $\mathcal{A}_k$ marks all strings that do not form a prefix of W_k.

To show that $\mathcal{A}_k$ does not mark any prefix of W_k, we prove the following claim that we further implicitly use in the rest of the proof.

Claim 6. *After generating the prefix of Z_n of length $\ell \leq 2^n - 1$, the observer of $\mathcal{A}_{2^n-1}$ is in state $\{q_{n-1}^{r_{n-1}}, q_{n-2}^{r_{n-2}}, \ldots, q_0^{r_0}\}$, where $r_{n-1} r_{n-2} \cdots r_0$ is the number $2^n - 1 - \ell$ in binary.*

Proof. By induction on ℓ. For $\ell = 0$, the number $2^n - 1$ is $11 \cdots 1$ in binary, which corresponds to the initial state $\{q_{n-1}^1, q_{n-2}^1, \ldots, q_0^1\}$ of the observer of $\mathcal{A}_{2^n-1}$. Assume that the claim holds for $\ell < 2^n - 1$, that is, the observer of $\mathcal{A}_{2^n-1}$ is in state $\{q_{n-1}^{r_{n-1}}, q_{n-2}^{r_{n-2}}, \ldots, q_0^{r_0}\}$, where $r_{n-1} r_{n-2} \cdots r_0$ is $2^n - 1 - \ell$ in binary. To prove the claim for $\ell + 1$, let r_t be the rightmost non-zero bit of $r_{n-1} r_{n-2} \cdots r_0$. Then, there are t trailing zeros, and hence the event of Z_n at position $\ell + 1$ is a_t. By the definition of $\mathcal{A}_{2^n-1}$, the transition under a_t is undefined in states $q_{t-1}^0, \ldots, q_0^0$, and it is a self-loop in $q_{n-1}^{r_{n-1}}, \ldots, q_{t+1}^{r_{t+1}}$. The transitions from q_t^1 under a_t lead to states $q_{t-1}^1, \ldots, q_0^1$ and to q_t^0. Thus, generating a_t in the next step, the observer of $\mathcal{A}_{2^n-1}$ moves from state $\{q_{n-1}^{r_{n-1}}, q_{n-2}^{r_{n-2}}, \ldots, q_0^{r_0}\}$ to state $\{q_{n-1}^{r_{n-1}}, q_{n-2}^{r_{n-2}}, \ldots, q_{t+1}^{r_{t+1}}, q_t^0, q_{t-1}^1, \ldots, q_0^1\}$, which is a state binary representing the number $2^n - 1 - \ell - 1 = 2^n - 1 - (\ell + 1)$, which proves the claim. □

Claim 6 implies that the automaton $\mathcal{A}_k$ corresponds to the automaton $\mathcal{A}_{2^n-1}$ having generated the prefix of Z_n of length $2^n - 1 - k = |Z_n| - |W_k|$; indeed, in this case, $\mathcal{A}_{2^n-1}$ is in states encoding the number $2^n - 1 - (2^n - 1 - k) = k$ in binary. Consequently, the observer of $\mathcal{A}_k$ generating W_k event by event goes through the respective states representing the numbers $k, k-1, \ldots, 0$ in binary. These states are not marked because they do not contain state q, and hence $\mathcal{A}_k$ does not mark any prefix of W_k.

To show that $\mathcal{A}_k$ marks all strings that do not form a prefix of W_k, assume that the observer of $\mathcal{A}_k$ is in a state of the form $\{q_{n-1}^{r_{n-1}}, q_{n-2}^{r_{n-2}}, \ldots, q_0^{r_0}\}$ reached by a prefix w of W_k. If $w = W_k$, then $r_{n-1} = r_{n-2} = \cdots = r_0 = 0$, and any move of $\mathcal{A}_k$ introduces q to the state of the observer of $\mathcal{A}_k$. Otherwise, let r_t be the rightmost non-zero bit of $r_{n-1} r_{n-2} \cdots r_0$. Then, $W_k = w a_t w'$. Assume that $\mathcal{A}_k$ now generates an event $a_i \neq q_t$. If $i > t$, the transition (q_t^1, a_i, q) is applied, and if $i < t$, the transition (q_i^0, a_i, q) is applied. Since q is never removed from a state of the observer of $\mathcal{A}_k$, all states reachable from now on are marked, which completes the proof. □

Example 7. For an illustration, take $k = 6$. Then $n = 3$ and the binary encoding of 6 is 110. For $Z_3 = a_0 a_1 a_0 a_2 a_0 a_1 a_0$, the suffix of length 6 is $W_6 = a_1 a_0 a_2 a_0 a_1 a_0$. The automaton $\mathcal{A}_6$ is depicted in Figure 3, where the initial states are q_2^1, q_1^1, and q_0^0 corresponding to the bits of 110. The computation

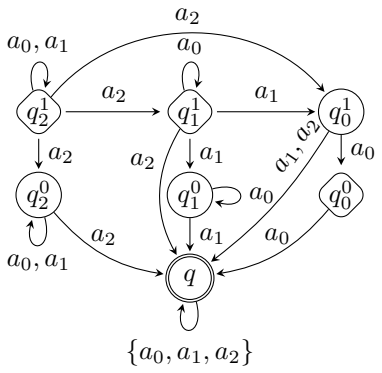

Figure 3. The NFA $\mathcal{A}_6$ with initial states of a diamond shape.

of $\mathcal{A}_6$ on $W_6 = a_1a_0a_2a_0a_1a_0$ is depicted in Figure 6, from which it should be clear that $\mathcal{A}_6$ does not mark any prefix of $W_6 = a_1a_0a_2a_0a_1a_0$, and that it marks all strings different from W_6. ◇

5.3 Reducing k-SO to CSO: the second step

Based on the reduction of INSO to CSO above, we are now ready to provide a reduction of k-SO to CSO by adding a counter. The idea behind the counter is to use the automaton $\mathcal{A}_k$ working along a string W_k that walks the automaton $\mathcal{A}_k$ through non-marked states along a path of length k.

Construction 8. Given a DES $G = (Q, \Sigma, \delta, I)$ and a $k \in \mathbb{N}_\infty$, we consider the automata G^+ and G^- obtained from G by Construction 3, and the counter automaton $\mathcal{A}_\mathrm{k} = (Q_n, \Sigma_n, \delta_n, I_n, \{q\})$ constructed in Section 5.2. We now connect G, G^+, G^-, and $\mathcal{A}_\mathrm{k}$ to a single NFA. However, before doing so, notice that G, G^+, and G^- are over the alphabet Σ, while $\mathcal{A}_k$ is over Σ_n, which is disjoint from Σ. Therefore, we first change the alphabets of the automata to $\Sigma \cup (\Sigma_o \times \Sigma_n)$ as follows.

(1) In G^+ and G^-, we replace every *observable* transition (p, σ, q) by $(p, (\sigma, \gamma), q)$, for every $\gamma \in \Sigma_n$, and we denote the results by $\tilde{G}^+$ and $\tilde{G}^-$.
(2) Similarly, in $\mathcal{A}_\mathrm{k}$, we replace every transition (p, γ, q) by $(p, (\sigma, \gamma), q)$, for every $\sigma \in \Sigma_o$, and we denote the result by $\tilde{\mathcal{A}}_\mathrm{k}$.

We now consider the automata as a single automaton $G' = (Q', \Sigma', \delta', I)$, where the set of states $Q' = Q \cup Q^+ \cup Q^- \cup Q_n$, the alphabet $\Sigma' = \Sigma \cup (\Sigma_o \times \Sigma_n) \cup \{@\}$, where @ is a new observable event, and the transition function $\delta' = \delta \cup \delta^+ \cup \delta^- \cup \delta_n$. In addition, we connect the automata by transitions

- $(q, @, q^+)$ and $(q, @, q_0)$, for every secret state $q \in Q_S$ and every initial state $q_0 \in I_n$ of $\tilde{\mathcal{A}}_k$, and
- $(q, @, q^-)$, for every non-secret state $q \in Q_{NS}$.

The set of secret states of G' is Q^+, the set of non-secret states is $Q^- \cup \{q\}$, and the set of observable events is $\Sigma_o \cup \{@\} \cup \Sigma_o \times \Sigma_n$. ◇

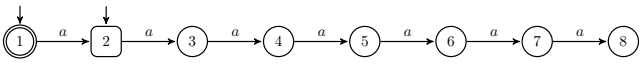

Figure 4. The DES G of the reduction of 6-SO to CSO; the secret state is marked and the non-secret state is squared.

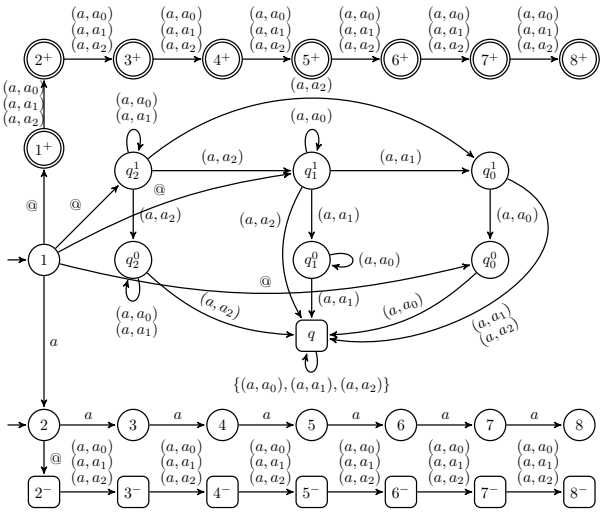

Figure 5. The DES G' of the reduction of 6-SO to CSO; secret states are marked and non-secret states are squared.

To illustrate the construction, we consider the 6-step opacity of $G = (\{1, \ldots, 8\}, \{a\}, \delta, \{1, 2\})$ with $\delta(i, a) = \{i + 1\}$, for $1 \leq i \leq 7$, depicted in Figure 4, where the event a is observable, state 1 is secret, and state 2 is non-secret. To encode $k = 6$, the construction uses the counter automaton $\mathcal{A}_6$ of Figure 3, and results in the automaton G' of Figure 5, where all non-secret states are squared.

In the sequel, we denote strings consisting of events over $\Sigma_o \times \Sigma_n$, e.g., $(a, b)(c, d)$, simply as a pair of corresponding concatenated strings of the components, e.g., (ac, bd).

Theorem 9. *A DES $G = (Q, \Sigma, \delta, I)$ is k-SO with respect to Q_S, Q_{NS}, and $P\colon \Sigma^* \to \Sigma_o^*$ if and only if G' obtained from G by Construction 8 is CSO with respect to $Q'_S = Q^+$, $Q'_{NS} = Q^- \cup \{q\}$, where q is the marked state of the automaton $\tilde{\mathcal{A}}_k$, and $P'\colon (\Sigma' \cup \{@\})^* \to (\Sigma_o \cup \{@\} \cup \Sigma_o \times \Sigma_n)^*$.*

Proof. Assume that G is k-SO. To show that G' is CSO, let $w \in L(G')$ be such that $\delta'(I, w) \cap Q'_S \neq \emptyset$. Since the set of secret states of G' is Q^+, the string w is of the form $w_1@w_2$, and hence $\delta(I, w_1)$ contains a secret state of G, from which w_2 can be generated.

If $|P(w_2)| \leq k$, then the k-SO property of G implies the existence of a string $w'_1w'_2 \in L(G)$ such that $P(w'_1) = P(w_1)$, $P(w'_2) = P(w_2)$, and $\delta(\delta(I, w'_1) \cap Q_{NS}, w'_2) \neq \emptyset$; that is, there is a non-secret state $p \in \delta(I, w'_1)$ from which w'_2 can be generated, reaching a state r. Then, for $w' = w'_1@(w'_2, x)$, where x is the prefix of length $|P(w'_2)|$ of the unique string W_k not marked by $\mathcal{A}_k$, we have $\delta'(I, w') \cap Q'_{NS} \neq \emptyset$, since

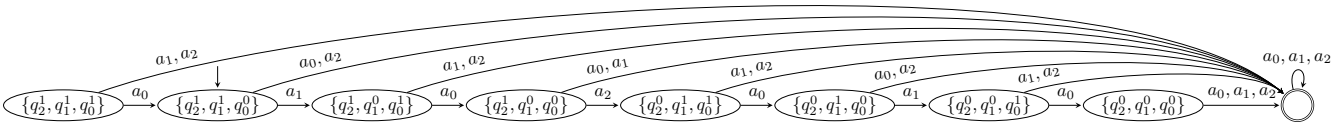

Figure 6. The observer of $\mathcal{A}_6$ showing the behavior of $\mathcal{A}_6$ on Z_3 and W_6. The initial state of $\mathcal{A}_6$ is denoted by the little arrow from above.

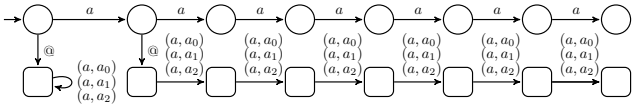

Figure 7. The minimized observer of G' of Figure 5.

the non-secret state $r^- \in Q^-$ is reachable from state p^- in G' by (w'_2, x). Thus, the automaton G' is CSO.

If $|P(w_2)| > k$, then every string $w'_1@(w'_2, y) \in \Sigma'^*$ is such that y is marked by $\mathcal{A}_k$, because $\mathcal{A}_k$ marks all strings longer than k. Therefore, (w'_2, y) is marked by $\tilde{\mathcal{A}}_k$, and hence $q \in \delta'(I, w'_1@(w'_2, y)) \cap Q'_{NS}$, which shows that G' is CSO.

On the other hand, assume that G is not k-SO. Then, there is $st \in L(G)$ such that $|P(t)| \leq k$, $\delta(\delta(I, s) \cap Q_S, t) \neq \emptyset$, and, for every $s' \in P^{-1}P(s)$ and every $t' \in P^{-1}P(t)$, $\delta(\delta(I, s') \cap Q_{NS}, t') = \emptyset$. Then, in G', $\delta'(I, s@) \cap Q'_S \neq \emptyset$. If $\delta(I, s') \cap Q_{NS} = \emptyset$, then $\delta'(I, s'@) \cap Q'_{NS} = \emptyset$. If $\delta(I, s') \cap Q_{NS} = Z \neq \emptyset$, we consider any string $s'@(t', y) \in L(G')$, where y is a prefix of the unique string W_k that is not marked by $\mathcal{A}_k$, which exists because $|y| = |P(t')| \leq k$. Then, (t', y) is not marked by $\tilde{\mathcal{A}}_k$, and hence $\delta'(I, s'@(t', y)) \cap Q'_{NS} = \delta'([\delta'(I, s'@) \cap Q^-], (t', y)) = \delta'(Z^-, (t', y)) = \emptyset$, where $Z^- = \{z^- \mid z \in Z\}$; indeed, (t', y) is not generated in G' from a state of Z^-, because t' is not generated in G from a state of $z \in Z$. Therefore, G' is not CSO. □

For an illustration, the state-minimal observer of G' is depicted in Figure 7, where every state containing a non-secret state of G' is squared. Since every state of the observer, reachable by a string containing @, contains a non-secret state of G', the automaton G' is CSO. Indeed, the automaton G of Figure 4 is 6-SO, since we can make six steps from both states 1 and 2.

If we remove state 8 and the corresponding transitions from G, then G is no longer 6-SO: six steps can be made from the secret state 1, but only five steps is possible from the non-secret state 2. Our reduction results in G' coinciding with the automaton of Figure 5 without states 8, 8^+, 8^-, and the corresponding transitions. The state-minimal observer is depicted in Figure 8, where the unique secret state (marked) denoting the state $\{7^+, q_2^0, q_1^0, q_0^0\}$ is reachable by the string $@(a, a_1)(a, a_0)(a, a_2)(a, a_0)(a, a_1)(a, a_0)$, and hence the automaton G' is not CSO.

5.4 *Algorithms and experimental results*

We implemented five algorithms to verify k-SO: the two-way observer of Yin and Lafortune (2017), called TW_OBS, the

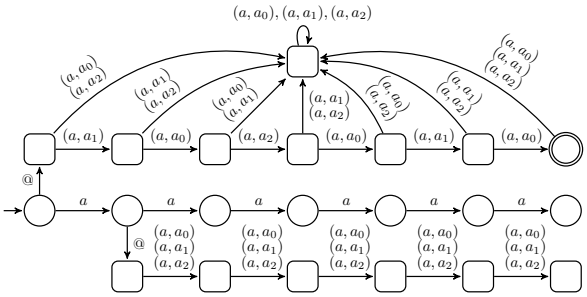

Figure 8. The minimized observer of G' of Figure 5 without states 8, 8^+, 8^-, and corresponding transitions.

algorithm of Balun and Masopust (2023) based on projected automata, called BM_KSO, the reduction of k-SO to the secret observer of Wintenberg et al. (2022), called W-KSO, and our reduction of k-SO to CSO of Construction 8, which results in two algorithms based on the two algorithms verifying CSO; namely, KSO_LIMI and KSO_OBS.

As already pointed out above, we implemented the two-way observer with a slight modification—rather than to use the dept-first search, we used the graph search implemented in `libFAUDES`. We also interrupt the algorithm as soon as a counterexample is found. The impact of the former choice on the performance of the two-way observer algorithm is not known to us.

Notice that Wintenberg et al. (2022) have, in fact, proposed several algorithms for k-SO verification. However, according to their investigation, the fastest was the reduction of k-SO to the secret observer, and therefore we selected this algorithm for comparisons. We have not implemented neither tested the other algorithms.

Combining our reduction of k-SO to CSO with the two algorithms for CSO verification results in two algorithms for k-SO verification: KSO_OBS and KSO_LIMI. We implemented KSO_OBS so that the instance of k-SO is reduced to an instance of CSO, and the observer-based algorithm is adjusted to never investigate a set containing the marked state q of the counter automaton $\mathcal{A}_k$; indeed, every state of the observer reachable from a state containing state q also contains state q, and hence such a path in the observer never violates CSO. This method is formalized as Algorithm 2.

We have run experiments for three choices of k ranging from small to large; namely, for $k = 5$, $k = 500$, and $k = 50000$. For $k = 500$, the algorithm of Wintenberg et al. (2022) timed out for 6433 instances, which is more than half of the instances, and therefore we excluded it from comparisons.

Algorithm 2. KSO_OBS

Require: A DES $G = (Q, \Sigma, \delta, I)$, $Q_S, Q_{NS} \subseteq Q$, $\Sigma_o \subseteq \Sigma$, and $k \in \mathbb{N}_\infty$.
Ensure: `true` if and only if G is k-step opaque with respect to Q_S, Q_{NS}, and $P\colon \Sigma^* \to \Sigma_o^*$.

```
Compute the instance G' of CSO (with the counter automaton A_k) by applying Construction 8 to G
Construct the observer G'^obs of G' step by step.
for every newly generated state X of G'^obs do
    if X contains the marked state q of A_k then
        do not add X to the observer of G', and continue
    end if
    if X ∩ Q_S ≠ ∅ and X ∩ Q_NS = ∅ then
        return false
    end if
end for
return true
```

For $k = 50000$, both the algorithm of Wintenberg et al. (2022) and our inclusion-based algorithm reducing k-SO to CSO timed out for basically all instances, and therefore we excluded them from comparisons.

The results for $k = 5$ are summarized in Table 3. All the tools together solved 11615 instances. Although BM_KSO was the fastest for the most instances, KSO_OBS based on our reduction of k-SO to CSO used the minimum total time and solved the most instances of all the tools.

The total time is the sum of the times to solve an instance, or 300 if the tool timed out when solving the instance, over all 11710 instances. On the other hand, the total time to solve positive/negative instances counts only the times over the instances that were solved by at least one of the tools.

The results for $k = 500$ and $k = 50000$ are summarized in Table 3 and Table 2, respectively. In both cases, the tools together solved 11601 instances. Again, BM_KSO was the fastest for the most instances, and this time it also took the minimum total time. For $k = 500$, KSO_OBS used the second minimum total time, whereas for $k = 50000$, it was the slowest. However, KSO_OBS was extremely fast to solve negative instances, and solved the most negative instances of all the tools. In these cases, the two-way observer solved the most instances within the given time limit.

6 Strong K-step opacity

In this section, we discuss a reduction of k-SSO to k-SO, and use it to design new algorithms to verify k-SSO. From all algorithms resulting from the combinations of our reduction with the algorithms verifying k-SO, we consider two, using our algorithms verifying k-SO, and compare them with the secret-observer-based algorithm of Wintenberg et al. (2022) and with the algorithm of Han et al. (2022).

Before that, we review the notions of k-SSO discussed in the literature. Falcone and Marchand (2015) introduced strong k-SO for deterministic DES in the following form.

Given a projection $P\colon \Sigma^* \to \Sigma_o^*$ and $k \in \mathbb{N}_\infty$, we say that a prefix w' of w is *k-short* if $|P(w)| - |P(w')| \leq k$.

Definition 10 (k-SSO-DFA)**.** For $k \in \mathbb{N}_\infty$, a deterministic DES $G = (Q, \Sigma, \delta, q_0)$ is k-SSO-DFA with respect to the set of secret states $Q_S \subseteq Q$ and projection $P\colon \Sigma^* \to \Sigma_o^*$ if for every $s \in L(G)$, there is $w \in L(G)$ such that $P(s) = P(w)$ and $\delta(q_0, w') \notin Q_S$ for every k-short prefix w' of w .

Recently, Han et al. (2022) tried to generalize it to NFAs.

Definition 11 (k-SSO-HZL)**.** For $k \in \mathbb{N}_\infty$, a DES $G = (Q, \Sigma, \delta, I)$ is k-SSO-HZL with respect to the set of secret states $Q_S \subseteq Q$ and projection $P\colon \Sigma^* \to \Sigma_o^*$ if for every run $q_0 \xrightarrow{s} q_s \xrightarrow{t} q_t$ with $q_0 \in I$, $q_s \in Q_S$, and $|P(t)| \leq k$, there is a run $p_0 \xrightarrow{s_1} p_s \xrightarrow{t_1} p_t$ such that $p_0 \in I$, $P(s_1) = P(s)$, $P(t_1) = P(t)$, and the run $p_s \xrightarrow{t_1} p_t$ is non-secret.

However, we show that k-SSO-HZL suffers from a similar problem as k-SO. Namely, the observer may infer that the system was in a secret state, though it cannot say when. For an example, consider the automaton in Figure 9 with state 1 secret, event a observable, and event u unobservable. For any $k \in \mathbb{N}_\infty$, the system is not k-SSO-DFA, because for $a^k \in L(G)$, any $w \in L(G)$ such that $P(w) = P(a^k) = a^k$ belongs to $\{a^k, ua^k\}$, but for the prefix ε of $w \in \{a^k, ua^k\}$, we have $|P(w)| - |P(\varepsilon)| = k$ and $\delta(1, \varepsilon) \in Q_S$, which contradicts k-SSO-DFA. On the other hand, the system is k-SSO-HZL, because for any run $1 \xrightarrow{s} 1 \xrightarrow{t} r$ with $r \in \{1, 2, 3, 4\}$ such that $|P(t)| \leq k$, we have $s = \varepsilon$ and there are two choices for t: either $P(t) = \varepsilon$, or $P(t) = a^\ell$ with $1 \leq \ell \leq k$. But then the run $1 \xrightarrow{u} 3 \xrightarrow{t'} p$, where $p = 3$ if $P(t) = \varepsilon$, and $p = 4$ if $P(t) = a^\ell = t'$, satisfies $P(u) = P(\varepsilon)$, $P(t') = P(t)$, and the run $3 \xrightarrow{t'} p$ is non-secret.

This problem led us to the following generalization of the definition of k-SSO-DFA.

Definition 12 (k-SSO-NFA)**.** For $k \in \mathbb{N}_\infty$, a DES $G = (Q, \Sigma, \delta, I)$ is k-SSO-NFA with respect to the set of secret

	BM_KSO	KSO_OBS	TW_OBS
Solved	11454	10899	11512
Fastest for	10918	123	438
Solved only by	5	65	81
Solved true inst.	5710	5090	5781
Solved false inst.	5744	5809	5731
Total time	88159	656378	154448
Time true solved	32623	618046	73718
Time false solved	22836	5632	48030

Table 2
Comparing BM_KSO, KSO_OBS, and TW_OBS for $k = 50000$ within a five-minute time limit.

	$k = 5$					$k = 500$			
	BM_KSO	KSO_LIMI	KSO_OBS	TW_OBS	W-KSO	BM_KSO	KSO_LIMI	KSO_OBS	TW_OBS
Solved	11500	11558	11597	11590	11577	11448	11127	11467	11509
Fastest for	5889	1936	3147	340	24	10068	15	817	587
Solved only by	0	3	2	14	0	2	3	25	80
Solved true inst.	5756	5774	5785	5796	5769	5707	5370	5658	5778
Solved false inst.	5744	5784	5812	5794	5808	5741	5757	5809	5731
Total time	74798	77654	43664	112141	64133	89848	314533	104333	164968
Time true solved	22040	37584	11169	58386	26159	32935	231684	66304	79470
Time false solved	24257	11570	3995	25256	9473	24213	50149	5330	52798

Table 3
Comparing BM_KSO, KSO_LIMI, KSO_OBS, TW_OBS, and W-KSO for $k = 5$ within a five-minute time limit, and BM_KSO, KSO_LIMI, KSO_OBS, and TW_OBS for $k = 500$ within a five-minute time limit.

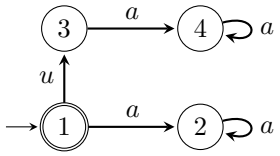

Figure 9. A deterministic DES that is k-SSO-HZL but not k-SSO-DFA, for any $k \in \mathbb{N}_\infty$. The secret state is marked; the event a is observable, and u is unobservable.

states $Q_S \subseteq Q$ and projection $P\colon \Sigma^* \to \Sigma_o^*$ if for every $s \in L(G)$, there is a run $i \xrightarrow{w}$, for some initial state i, such that $P(s) = P(w)$ and for every k-short prefix w' of w the subrun $i \xrightarrow{w'} j$ satisfies $j \notin Q_S$.

The notions of k-SSO-DFA and k-SSO-NFA coincide for deterministic DES. Indeed, if G is a DFA, then $\delta(i, w)$ defines a unique run $i \xrightarrow{w}$, and vice versa, and therefore k-SSO-NFA states that for every $s \in L(G)$, there is a string $w \in L(G)$ whose every k-short prefix w' satisfies $\delta(i, w') \notin Q_S$.

We now clarify the relationship between k-SSO-NFA and k-SSO-HZL by showing that these definitions coincide for systems that do not have unobservable transitions from secret states to non-secret states. We call such systems *normal*.

Theorem 13. *Let G be a normal DES, and let $k \in \mathbb{N}_\infty$. Then, G is k-SSO-HZL with respect to Q_S and P if and only if G is k-SSO-NFA with respect to Q_S and P.*

Proof. Consider an initial run $q_0 \xrightarrow{s} q_s \xrightarrow{t} q_t$ such that $q_s \in Q_s$ and $|P(t)| \le k$. If G is k-SSO-NFA, then there is an initial run $i \xrightarrow{w}$ such that $P(w) = P(st)$, and for every k-short prefix w_1 of w, the subrun $i \xrightarrow{w_1} j$ has $j \notin Q_S$. Let $w = s't'$ with $P(s') = P(s)$ and $P(t') = P(t)$, and observe that for any prefix t'' of t', the string $s't''$ is a k-short prefix of w (recall $|P(t')| = |P(t)| \le k$), and hence the subrun $i \xrightarrow{s't''} j$ has $j \notin Q_S$. Consequently, taking $t'' = \epsilon$ shows that the subrun $j_1 \xrightarrow{t'} j_2$ of $i \xrightarrow{s'} j_1 \xrightarrow{t'} j_2$ is non-secret, verifying that G is k-SSO-HZL.

On the other hand, for $s \in L(G)$, let $q_0 \xrightarrow{s} = q_0 \xrightarrow{t} q_t \xrightarrow{v} q_v$ be an initial run with $q_t \in Q_S$ and v being the longest suffix of s such that $|P(v)| \le k$. If G is k-SSO-HZL, then there is an initial run $p_0 \xrightarrow{t'} p' \xrightarrow{v'} p''$ such that $p' \xrightarrow{v'} p''$ is non-secret, $P(t') = P(t)$, and $P(v') = P(v)$. Let u be the longest suffix of t' consisting of unobservable events, and let $t' = t''u$. Taking $p_0 \xrightarrow{t''} p_1$, the normality of G implies that $p_1 \xrightarrow{u} p' \xrightarrow{v'} p''$ is nonsecret ($p' \xrightarrow{v} p''$ is nonsecret and no unobservable event takes a secret state to a nonsecret one). Moreover, t'' is a prefix of any k-short prefix w of $t''uv'$ (recall $|P(v')| = |P(v)| \le k$), and hence $p_0 \xrightarrow{w} j$ satisfies $j \notin Q_S$. This verifies k-SSO-NFA. □

If a system is not normal, we can normalize it as shown below without affecting the strong k-step opacity property. In particular, the normalization allows us to use the verification algorithms for k-SSO-HZL to verify both k-SSO-DFA and k-SSO-NFA.

Construction 14. Let $G = (Q, \Sigma, \delta, I)$ be a DES, $k \in \mathbb{N}_\infty$, $Q_S \subseteq Q$ be the set of secret states, and $P\colon \Sigma^* \to \Sigma_o^*$ be the projection to observable events. From G, we construct the DES $G_{norm} = (Q, \Sigma, \delta_n, I)$, where δ_n is initialized as δ and further modified in the following two steps:

(1) From δ_n, we remove all transitions $(q_s, u, q_{ns}) \in \delta_n$ where $q_s \in Q_S$ is secret, $q_{ns} \in Q_{NS}$ is non-secret, and $u \in \Sigma_{uo}$ is unobservable.
(2) For every secret state $q_s \in Q_S$ and every observable event $a \in \Sigma_o$, if there is a non-secret state $q_{ns} \in \delta(q_s, P^{-1}(\varepsilon))$ such that $(q_{ns}, a, p) \in \delta$, then we add the transition (q_s, a, p) to δ_n.

The secret states and observable events of G_{norm} coincide with those of G. See Figure 10 for an illustration. ◇

The automaton G_{norm} is the *normalization* of G. Obviously, in G_{norm}, no non-secret state is reachable from a secret state by a sequence of unobservable events. Indeed, G is normal if G and G_{norm} coincide.

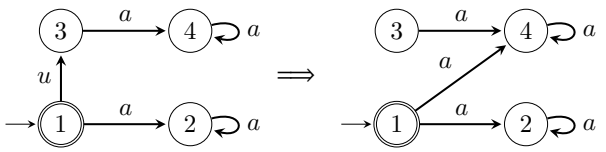

Figure 10. The normalization of a DES.

The following lemma is obvious by Construction 14.

Lemma 15. *Let G_{norm} be the normalization of a DES G by Construction 14. Then, for every run $i \xrightarrow{w}$ in G_{norm}, there is a run $i \xrightarrow{v}$ in G such that $P(w) = P(v)$.* □

The following result is useful.

Lemma 16. *For a DES $G = (Q, \Sigma, \delta, I)$ with secret states Q_S and projection $P\colon \Sigma^* \to \Sigma_o^*$, let G_{norm} be the normalization of G by Construction 14. Then, for every run $i \xrightarrow{w} j$ in G, there is a run $i \xrightarrow{v} j'$ in G_{norm} such that $P(w) = P(v)$, and for every decomposition $i \xrightarrow{w} j = i \xrightarrow{w_1} q \xrightarrow{w_2} j$, there is a decomposition $i \xrightarrow{v} j' = i \xrightarrow{v_1} q' \xrightarrow{v_2} j'$ such that*

(1) $P(w_1) = P(v_1)$,
(2) if $q \in Q_S$, then $q' \in Q_S$, and
(3) if $w_1 = w_3 a$, for some $a \in \Sigma_o$, then $q' = q$.

Proof. By induction on the length of w. For $w = \epsilon$, the claim holds. Thus, assume that $w \neq \varepsilon$, and consider a decomposition of $i \xrightarrow{w} j$ of the form $i \xrightarrow{w_1} q_s \xrightarrow{u} q_{ns} \xrightarrow{w_2} j$ where $q_s \in Q_S$, $q_{ns} \in Q_{NS}$, $u \in \Sigma_{uo}^*$, and w_2 does not start with an unobservable event. If there is no such decomposition, $i \xrightarrow{w} j$ is a run in G_{norm} that satisfies the lemma.

If there is at least one such decomposition, we pick one where the subrun $i \xrightarrow{w_1} q_s$ is not further decomposable in the prescribed way. The lemma holds for $i \xrightarrow{w_1} q_s$ by the previous paragraph, and if $w_2 = \epsilon$, it also holds for $i \xrightarrow{w} j$. If $w_2 = ax$, for some observable event $a \in \Sigma_o$, then the run $i \xrightarrow{w} j = i \xrightarrow{w_1} q_s \xrightarrow{u} q_{ns} \xrightarrow{a} q_1 \xrightarrow{x} j$. By Construction 14, $i \xrightarrow{w_1} q_s \xrightarrow{a} q_1$ is a run in G_{norm} that satisfies the lemma for the run $i \xrightarrow{w_1} q_s \xrightarrow{u} q_{ns} \xrightarrow{a} q_1$ in G. By the induction hypothesis, for the run $q_1 \xrightarrow{x} j$ in G, there is a run $q_1 \xrightarrow{y} j'$ in G_{norm} satisfying the lemma. Therefore, for the run $i \xrightarrow{w} j$ in G, $i \xrightarrow{w_1 a} q_1 \xrightarrow{y} j'$ is a run in G_{norm} that satisfies the lemma. □

We can now show the following result.

Theorem 17. *Let $k \in \mathbb{N}_\infty$. A DES G is k-SSO-NFA with respect to Q_S and P if and only if the normalization G_{norm} of G obtained by Construction 14 is k-SSO-NFA with respect to Q_S and P.*

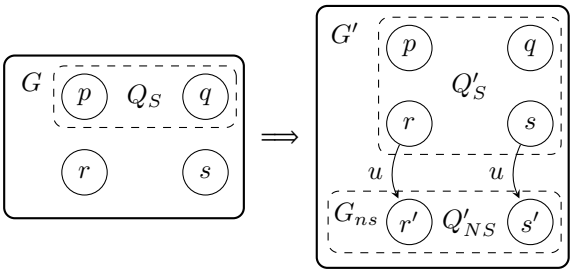

Figure 11. Construction 18 reducing k-SSO to k-SO.

Proof. Assume that G is k-SSO-NFA with respect to Q_S and P and consider any $x \in L(G_{norm})$. By Lemma 15, there is $s \in L(G)$ such that $P(x) = P(s)$, and by k-SSO-NFA of G, there is an initial run $i \xrightarrow{w}$ in G such that $P(s) = P(w)$ and, for every k-short prefix w_1 of w the subrun $i \xrightarrow{w_1} j$ has $j \notin Q_S$. Let $w = w_p z$, where w_p is the shortest k-short prefix of w. If $w_p = \varepsilon$, the run $i \xrightarrow{w}$ is non-secret. If $w_p = va$, for some $a \in \Sigma_o$, then $i \xrightarrow{w} = i \xrightarrow{va} q_{ns,1} \xrightarrow{z} q_{ns,2}$ where $q_{ns,1} \xrightarrow{z} q_{ns,2}$ is non-secret. By Lemma 16, for $i \xrightarrow{va} q_{ns,1}$ in G, there is a run $i \xrightarrow{ya} q_{ns,1}$ in G_{norm} such that $P(ya) = P(va)$. Therefore, $i \xrightarrow{ya} q_{ns,1} \xrightarrow{z} q_{ns,2}$ is a run in G_{norm} such that $P(x) = P(yaz)$ and since every k-short prefix w_1 of yaz has prefix ya, the subrun $i \xrightarrow{w_1} j$ has $j \notin Q_S$. Thus, G_{norm} is k-SSO-NFA with respect to Q_S and P.

Now, assume that G is not k-SSO-NFA with respect to Q_S and P, and let $s \in L(G)$ be a string that violates the property. Consider any initial run $i \xrightarrow{v}$ in G_{norm} such that $P(s) = P(v)$. By Lemma 15, there is a run $i \xrightarrow{w}$ in G with $P(w) = P(v)$. Since $P(w) = P(s)$ and G is not k-SSO-NFA, there is a k-short prefix w_1 of w such that the subrun $i \xrightarrow{w_1} q_1$ has $q_1 \in Q_S$. By Lemma 16, the run $i \xrightarrow{v}$ in G_{norm} can be decomposed as $i \xrightarrow{v} = i \xrightarrow{v_1} q'_1 \xrightarrow{v_2}$ such that $P(v_1) = P(w_1)$ and $q'_1 \in Q_S$. Since v_1 is a k-short prefix of v, G_{norm} is not k-SSO-NFA with respect to Q_S and P. □

6.1 Reducing K-SSO-NFA to K-SO

We now reduce k-SSO-NFA to k-SO. Without loss of generality, we consider only normal DES.

Construction 18. Let $G = (Q, \Sigma, \delta, I)$ be a normal DES, $P\colon \Sigma^* \to \Sigma_o^*$ be the projection, and Q_S be the set of secret states. We construct $G' = (Q', \Sigma \cup \{u\}, \delta', I)$ as a disjoint union of G and G_{ns}, where G_{ns} is obtained from G by removing all secret states and corresponding transitions. We add a new unobservable event u and u-transitions from every state $q \notin Q_S$ to its copy q' in G_{ns}, see Figure 11 for an illustration. The non-secret states Q'_{NS} of G' are the states of G_{ns}, the secret states Q'_S of G' are the states of G, and the observable events of G' coincide with the observable events of G, that is, we define $P'\colon (\Sigma \cup \{u\})^* \to \Sigma_o^*$. ◇

The following theorem describes the relationship between k-SSO-NFA and k-SO.

Theorem 19. *Let G be a normal DES. Then G is k-SSO-NFA with respect to Q_S and P if and only if G' is k-SO with respect to Q'_S, Q'_{NS}, and P', where G',Q'_S, Q'_{NS} and P' are obtained by Construction 18.*

Proof. Assume that G is k-SSO-NFA. Take $st \in L(G')$ such that $|P'(t)| \leq k$ and $\delta'(I, s) \cap Q'_S \neq \emptyset$. Let s_1t_1 be st with u removed. Clearly $P(s_1) = P'(s)$ and $P(t_1) = P'(t)$. It follows that $s_1t_1 \in L(G)$ and since G is k-SSO-NFA there is an initial run $i \xrightarrow{w}$ in G with $P(w) = P(s_1t_1)$ such that for any k-short prefix w_1 of w the subrun $i \xrightarrow{w_1} j$ satisfies $j \notin Q_S$. Let $w = w_pw_r$ where w_p is a k-short prefix with $P(w_p) = P(s_1)$. Then by Construction 18, $i \xrightarrow{w_p} j \xrightarrow{u} q_{ns} \xrightarrow{w_r}$ is a run in G' where $q_{ns} \in Q'_{NS}$.

Assume now that G' is k-SO with respect to Q'_S, Q'_{NS}, and P'. For the sake of contradiction, suppose that G is not k-SSO-NFA with respect to Q_S and P. Then, there is $v \in L(G)$ such that, for every initial run $i_w \xrightarrow{w}$ in G with $P(w) = P(v)$, there is a k-short prefix s_w of w such that the subrun $i_w \xrightarrow{s_w} j_w$ has $j_w \in Q_S$; let S_w denote the set of all such prefixes of w, and let $W = \{w \in L(G) \mid P(w) = P(v)\}$. Pick $w_0 = st \in W$ be such that $P(s)$ is a shortest string from $\bigcup_{w \in W} S_w$. Then, in G', $\delta'(\delta'(I, s) \cap Q'_S, t) \neq \emptyset$, and the k-SO property of G' implies that there is $s't' \in L(G')$ such that $P'(s) = P'(s')$, $P'(t) = P'(t')$, and $\delta'(\delta'(I, s') \cap Q'_{NS}, t') \neq \emptyset$. In particular, $s' = s''us'''$, and there is an initial run $i \xrightarrow{s''} r \xrightarrow{u} r' \xrightarrow{s'''t'} q'$ in G'. Then, $i \xrightarrow{s''} r \xrightarrow{s'''t'} q$ is an initial run in G with $r \xrightarrow{s'''t'} q$ non-secret, and $z = s''s'''t' \in W$. Since G is not k-SSO-NFA, and $P(z) = P(v)$, there is a k-short prefix s_z of z such that the subrun $i \xrightarrow{s_z} j_z$ has $j_z \in Q_S$. However, since the run $r \xrightarrow{s'''t'} q$ is non-secret, s_z is a strict prefix of s''. Since G is normal, $|P(s_z)| < |P(s'')| \leq |P(s)|$, which is a contradiction with the choice of w_0. □

6.2 *Experimental results*

We implemented two algorithms to verify k-SSO from the literature. Namely, the algorithm of Han et al. (2022), called HZL, the usage of which is justified by Theorem 13, because our benchmarks consist only of normal DES, and the reduction of k-SSO to the secret observer of Wintenberg et al. (2022), called W-KSSO. Our reduction of k-SSO to k-SO results in several algorithms, from which we implemented two: the one using the projected-automata-based algorithm of Balun and Masopust (2023), called KSSO-BM, and the one reducing k-SO to CSO using the classical observer-based algorithm, called KSSO-O.

Notice that Wintenberg et al. (2022) proposed several algorithms for the verification of k-SSO. However, according to their study, the reduction of k-SSO to the secret observer is the best. Therefore, we have chosen this algorithm for comparisons, and we have not implemented and tested the other algorithms proposed by Wintenberg et al. (2022).

	HZL	W-KSSO	KSSO-O	KSSO-BM
Solved	11332	11594	11565	11354
Fastest for	8327	2477	165	488
Solved only by	2	21	0	0
True solved	5591	5786	5756	5615
False solved	5741	5808	5809	5739
Total time	241532	54529	73754	186521
Time true solved	184022	14365	32945	125338
Time false solved	23610	6264	6910	27283

Table 4
Comparing HZL, W-KSSO, KSSO-O, and KSSO-BM for $k = 5$ within a five-minute time limit.

	HZL	W-KSSO	KSSO-O	KSSO-BM
Solved	11319	10892	9709	11280
Fastest for	8855	62	58	2501
Solved only by	167	1	12	0
True solved	5578	5150	3903	5541
False solved	5741	5742	5806	5739
Total time	247279	1023909	960910	219473
Time true solved	179976	716691	902364	148316
Time false solved	22903	262816	14145	26756

Table 5
Comparing HZL, W-KSSO, KSSO-O, and KSSO-BM for $k = 500$ within a five-minute time limit.

Similarly to the case of k-SO, we have run the experiments for $k \in \{5, 500, 50000\}$. The results for $k = 5$ are summarized in Table 4. All the tools together solved 11597 instances. Although HZL, implementing the algorithm of Han et al. (2022), was the fastest for the most instances, the minimum total time took the algorithm of Wintenberg et al. (2022), followed by our KSSO-O.

The results for $k = 500$ are summarized in Table 5. All the tools together solved 11562 instances. Even though HZL was the fastest for the most instances, the minimum total time took our reduction-based tool KSSO-BM. Notice that both W-KSSO and KSSO-O took an enormous amount of time.

Finally, the results for $k = 50000$ are summarized in Table 6. The W-KSSO and KSSO-O tools timed out for many instances, and therefore we excluded them from the comparisons. The remaining tools together solved 11496 instances. Although HZL was again fastest for most of the instances, in total it took for more than six hours more than our KSSO-BM.

7 Conclusions

We studied reductions of k-SSO to k-SO, and of k-SO to CSO. These reductions result in sever new algorithms de-

	HZL	KSSO-BM
Solved	11369	11285
Fastest for	8911	2492
Solved only by	211	127
True solved	5628	5545
False solved	5741	5740
Total time	232029	208894
Time true solved	164771	138088
Time false solved	3059	6606

Table 6
Comparing HZL and KSSO-BM for $k = 50000$ within a five-minute time limit.

ciding k-SO and k-SSO. We implemented some of these algorithms and experimentally compared them with the existing algorithms. For the comparisons, we created extensive benchmarks based on almost twelve thousand real-based automata examples with the state spaces ranging from two to 278372 and the number of transitions ranging from four to 95937444. Our results showed that the algorithms based on our reductions perform very well. In addition, by adding our normalization technique to the algorithm of Han et al. (2022), we made it applicable to verify k-SSO-NFA and k-SSO-DFA.

Considering the benchmarks, there are instances that were not solved by any of the considered algorithms within a five-minute time limit. In fact, it turns out that for some of the instances, the considered algorithms run for hours without success. These instances are a challenge for new techniques to be developed that would be able to solve these difficult cases. We should, however, confess that all our implementations are straightforward and unoptimized. It is, therefore, an interesting question whether the implementations can be significantly optimized. We leave it for future research.

References

Badouel, E., Bednarczyk, M., Borzyszkowski, A., Caillaud, B., Darondeau, P., 2007. Concurrent secrets. Discrete Event Dynamic Systems 17, 425–446.

Balun, J., Masopust, T., 2021. Comparing the notions of opacity for discrete-event systems. Discrete Event Dynamic Systems 31, 553–582.

Balun, J., Masopust, T., 2022. On transformations among opacity notions, in: IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3012–3017.

Balun, J., Masopust, T., 2023. Verifying weak and strong k-step opacity in discrete-event systems. Automatica 155, 111153.

Balun, J., Masopust, T., Osicka, P., 2023. Speed Me up If You Can: Conditional Lower Bounds on Opacity Verification, in: International Symposium on Mathematical Foundations of Computer Science, pp. 16:1–16:15.

Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A., 2008. Opacity generalised to transition systems. International Journal of Information Security 7, 421–435.

Bryans, J.W., Koutny, M., Ryan, P.Y.A., 2004. Modelling opacity using petri nets. Electronic Notes in Theoretical Computer Science 121, 101–115.

Cassandras, C.G., Lafortune, S., 2021. Introduction to Discrete Event Systems. 3rd ed., Springer.

Černý, P., Clarke, E.M., Henzinger, T.A., Radhakrishna, A., Ryzhyk, L., Samanta, R., Tarrach, T., 2017. From non-preemptive to preemptive scheduling using synchronization synthesis. Formal Methods in System Design 50, 97–139.

Dubreil, J., Darondeau, P., Marchand, H., 2008. Opacity enforcing control synthesis, in: International Workshop on Discrete Event Systems, pp. 28–35.

Falcone, Y., Marchand, H., 2015. Enforcement and validation (at runtime) of various notions of opacity. Discrete Event Dynamic Systems 25, 531–570.

Han, X., Zhang, K., Li, Z., 2022. Verification of strong k-step opacity for discrete-event systems, in: IEEE Conference on Decision and Control, pp. 4250–4255.

Hopcroft, J.E., Motwani, R., Ullman, J.D., 2006. Introduction to Automata Theory, Languages, and Computation. 3rd ed., Addison-Wesley Longman Publishing Co., Inc., USA.

Impagliazzo, R., Paturi, R., 2001. On the complexity of k-SAT. Journal of Computer and System Sciences 62, 367–375.

Jacob, R., Lesage, J.J., Faure, J.M., 2016. Overview of discrete event systems opacity: Models, validation, and quantification. Annual Reviews in Control 41, 135–146.

Jirásková, G., Masopust, T., 2012. On a structural property in the state complexity of projected regular languages. Theoretical Computer Science 449, 93–105.

Saboori, A., 2011. Verification and enforcement of state-based notions of opacity in discrete event systems. Ph.D. thesis. University of Illinois at Urbana-Champaign.

Saboori, A., Hadjicostis, C.N., 2007. Notions of security and opacity in discrete event systems, in: IEEE Conference on Decision and Control, pp. 5056–5061.

Saboori, A., Hadjicostis, C.N., 2012. Verification of infinite-step opacity and complexity considerations. IEEE Transactions on Automatic Control 57, 1265–1269.

Sloan, N.J.A., a. The on-line encyclopedia of integer sequences (OEIS). URL: https://oeis.org/A123121. A123121.

Sloan, N.J.A., b. The on-line encyclopedia of integer sequences (OEIS). URL: https://oeis.org/A001511. A001511.

Tange, O., 2024. Gnu parallel. doi:10.5281/zenodo.10558745.

Theunissen, R.J.M., Petreczky, M., Schiffelers, R.R.H., van Beek, D.A., Rooda, J.E., 2014. Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner. IEEE Transactions on Automation Science and Engineering 11, 20–32.

Wintenberg, A., Blischke, M., Lafortune, S., Ozay, N., 2022. A general language-based framework for specifying and verifying notions of opacity. Discrete Event Dynamic Systems 32, 253–289.

Wong, K., 1998. On the complexity of projections of discrete-event systems, in: International Workshop on Discrete Event Systems, pp. 201–206.

Wu, Y.C., Lafortune, S., 2013. Comparative analysis of related notions of opacity in centralized and coordinated architectures. Discrete Event Dynamic Systems 23, 307–339.

Wulf, M.D., Doyen, L., Henzinger, T.A., Raskin, J., 2006. Antichains: A new algorithm for checking universality of finite automata, in: International Conference on Computer Aided Verification, pp. 17–30.

Yin, X., Lafortune, S., 2017. A new approach for the verification of infinite-step and K-step opacity using two-way observers. Automatica 80, 162–171.