2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Logical Zonotopes: A Set Representation for the Formal Verification of Boolean Functions

Amr Alanwar[1,2], Frank J. Jiang[3], Samy Amin[2], and Karl H. Johansson[3]

***Abstract*— A logical zonotope, which is a new set representation for binary vectors, is introduced in this paper. A logical zonotope is constructed by XOR-ing a binary vector with a combination of other binary vectors called generators. Such a zonotope can represent up to 2^γ binary vectors using only γ generators. It is shown that logical operations over sets of binary vectors can be performed on the zonotopes' generators and, thus, significantly reduce the computational complexity of various logical operations (e.g., XOR, NAND, AND, OR, and semi-tensor products). Similar to traditional zonotopes' role in the formal verification of dynamical systems over real vector spaces, logical zonotopes can efficiently analyze discrete dynamical systems defined over binary vector spaces. We illustrate the approach and its ability to reduce the computational complexity in two use cases: (1) encryption key discovery of a linear feedback shift register and (2) safety verification of a road traffic intersection protocol.**

I. Introduction

For several decades, logical systems have been used to model complex behaviors in numerous applications. By modeling a system as a collection of logical functions operating in a binary vector space, we can design models that consist of relatively simple dynamics but still capture a complex system's behavior at a sufficient level of abstraction. Some popular approaches to modeling logical systems are finite automatons, Petri nets, or Boolean Networks (BNs).

An important form of analysis for logical systems is reachability analysis. Reachability analysis allows us to formally verify the behavior of logical systems and provide guarantees that, for example, the system will not enter into undesired states. One of the primary challenges of reachability analysis is the need to exhaustively explore the system's state space, which grows exponentially with the number of state variables. To avoid exponential computational complexity, many reachability analysis algorithms are based on a representation called Binary Decision Diagram (BDD). Given a proper variable ordering, BDDs can evaluate Boolean functions with linear complexity in the number of variables [1]. While BDDs play a crucial role in verification, they have well-known drawbacks, such as requiring an externally supplied variable ordering [2], [3]. Outside of BDDs, there are also approaches to reachability analysis for logical systems modeled as BNs, or Boolean Control Networks (BCNs) for systems with control inputs, that rely on the semi-tensor product [4]. However, due to being point-wise and scaling limitations of semi-tensor products, BCN-based approaches become intractable for high-dimensional logical systems [5]. In this work, we propose a novel zonotope representation that significantly reduces the exponential computational complexity of reachability analysis.

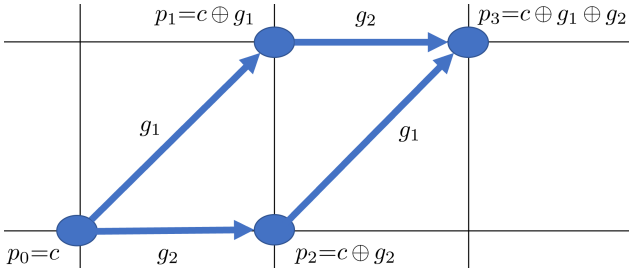


Fig. 1: Representing four points $p_0, \ldots, p_3$ by considering all combinations of turning off and on the two generators g_1 and g_2 of a logical zonotope.

In real vector spaces, zonotopes already play an important role in the reachability analysis of dynamical systems [6], [7]. Classical zonotopes are constructed by taking the Minkowski sum of a real vector center and a combination of real vector generators. Through this construction, a set of infinite real vectors can be represented by a finite number of generators. Then, by leveraging the fact that the Minkowski sum of two classical zonotopes can be computed by combining their respective generators, researchers have formulated computationally efficient approaches to reachability analysis [7]. In this work, we take inspiration from classical zonotopes and formulate logical zonotopes. Similarly, logical zonotopes are constructed by XOR-ing a binary vector center and a combination of binary vector generators. In binary vector spaces, logical zonotopes are able to represent up to 2^γ binary vectors using only γ generators, as illustrated in Figure 1 with $\gamma = 2$. Moreover, we show that any logical operation on the

This work is supported by the Knut and Alice Wallenberg Foundation, the Swedish Strategic Research Foundation, the Swedish Research Council, and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[1]School of Computation, Information and Technology, Technical University of Munich. alanwar@tum.de.

[2]School of Computer Science and Engineering, Constructor University. {aalanwar,samin}@constructor.university.

[3]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. Authors are affiliated with Digital Futures. {frankji, kallej}@kth.se.

generators of the logical zonotopes is either equivalent to or over-approximates the explicit application of the logical operation to each member of the represented set. Based on these results, we formulate our logical zonotope-based reachability analysis for logical systems.

Explicitly, the contributions of this work is summarized by the following: (1) we present our formulation of logical zonotopes, (2) we detail the application of logical operations and forward reachability analysis to logical zonotopes, (3) we illustrate the use of logical zonotopes in two different applications. To recreate our results, readers can use our publicly available logical zonotope library[1].

The remainder of the paper is organized as follows. In Section II, we introduce the notation and preliminary definitions. In Section III, we formulate logical zonotopes and overview the supported operations. Then, in Section IV, we illustrate the applications of logical zonotopes. Finally, in Section V, we conclude the work.

II. Notation and Preliminaries

In this section, we introduce details about the notation used throughout this work and preliminary definitions for logical systems and reachability analysis.

A. Notation

The set of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. We denote the binary set $\{0,1\}$ by $\mathbb{B}$. The XOR, NOT, OR, and AND operations are denoted by $\oplus, \neg, \vee$, and $\wedge$, respectively. Throughout the rest of the work, with a slight abuse of notation, we omit the $\wedge$ from $a \wedge b$ and write $a\,b$ instead. The NAND, NOR, and XNOR are denoted by $\barwedge, \barvee$, and $\odot$, respectively. Later, we use the same notation for both the classical and Minkowski logical operators, as it will be clear when the operation is taken between sets or individual vectors. Like the classical AND operator, we will also omit the Minkowski AND to simplify the presentation. Matrices are denoted by uppercase letters, e.g., $G \in \mathbb{B}^{n\times k}$, and sets by uppercase calligraphic letters, e.g., $\mathcal{Z} \subset \mathbb{B}^n$. Vectors and scalars are denoted by lowercase letters, e.g., $b \in \mathbb{B}^n$ with elements $b_{1:n}$. The identity matrix of size $n \times n$ is denoted I_n. We denote the Kronecker product by $\otimes$. The $x \in \mathbb{B}^n$ is an $n \times 1$ binary vector.

B. Preliminaries

For this work, we consider a system with a logical function $f : \mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \to \mathbb{B}^{n_x}$:

$$x(k+1) = f\big(x(k), u(k)\big) \tag{1}$$

where $x(k) \in \mathbb{B}^{n_x}$ is the state and $u(k) \in \mathbb{B}^{n_u}$ is the control input. The logical function f can consist of any combination of $\oplus, \neg, \vee, \barwedge, \barvee, \odot$, and $\wedge$. We will represent sets of states and inputs for (1) using logical zonotopes. As will be shown, logical zonotopes are constructed using an Minkowski XOR operation, which we define as follows.

Definition 1: **(Minkowski XOR)** Given two sets $\mathcal{L}_1$ and $\mathcal{L}_2$ of binary vectors, the Minkowski XOR is defined between every two points in the two sets as

$$\mathcal{L}_1 \oplus \mathcal{L}_2 = \{z_1 \oplus z_2 | z_1 \in \mathcal{L}_1, z_2 \in \mathcal{L}_2\}. \tag{2}$$

Similarly, we define the Minkowski NOT, OR and AND operations as follows.

$$\neg\mathcal{L}_1 = \{\neg z_1 | z_1 \in \mathcal{L}_1\}, \tag{3}$$

$$\mathcal{L}_1 \vee \mathcal{L}_2 = \{z_1 \vee z_2 | z_1 \in \mathcal{L}_1, z_2 \in \mathcal{L}_2\}, \tag{4}$$

$$\mathcal{L}_1\mathcal{L}_2 = \{z_1 z_2 | z_1 \in \mathcal{L}_1, z_2 \in \mathcal{L}_2\}. \tag{5}$$

We aim to show how logical zonotopes can be used to compute the forward reachable sets of systems defined by (1). We define the reachable sets of system (1) by the following definition.

Definition 2: (**Exact Reachable Set**) Given a set of initial states $\mathcal{X}_0 \subset \mathbb{B}^{n_x}$ and a set of possible inputs $\mathcal{U}_k \subset \mathbb{B}^{n_u}$, the exact reachable set $\mathcal{R}_N$ of (1) after N steps is

$$\mathcal{R}_N = \Big\{x(N) \in \mathbb{B}^{n_x} \;\Big|\; \forall k \in \{0,\dots,N-1\} : \\ x(k+1) = f\big(x(k), u(k)\big),\; x(0) \in \mathcal{X}_0, u(k) \in \mathcal{U}_k\Big\}.$$

Another commonly used operator for BCNs is the semi-tensor product [8]. Since semi-tensor products are useful in many applications, we have extended the classical definition to logical zonotopes. The classical definition for semi-tensor products is as follows.

Definition 3: **(Semi-Tensor Product [9])** Given two matrices $M \in \mathbb{B}^{m\times n}$ and $N \in \mathbb{B}^{p\times q}$, the semi-tensor product, denoted by $\ltimes$, is computed as:

$$M \ltimes N = (M \otimes I_{s_1})(N \otimes I_{s_2}), \tag{6}$$

where $s_1 = s/n$, and $s_2 = s/p$, with s being the least common multiple of n and p.

III. Logical Zonotopes

In this section, we present logical zonotopes and overview several different aspects of their use. We start by defining the set representation of logical zonotopes. Then, we go through the application of Minkowski XOR, NOT, XNOR, AND, NAND, OR, and NOR on logical zonotopes. Using these results, we show that when using logical zonotopes for reachability analysis on (1), we are able to compute reachable sets that over-approximate the exact reachable sets. Finally, we present an algorithm for reducing the number of generators of a logical zonotope.

A. Set Representation

Inspired by the classical zonotopic set representation which is defined in real vector space [10], we propose logical zonotopes as a set representation for binary vectors. We define logical zonotopes as follows.

Definition 4: (**Logical Zonotope**) Given a point $c \in \mathbb{B}^n$ and $\gamma \in \mathbb{N}$ generator vectors in a generator matrix

[1] https://github.com/aalanwar/Logical-Zonotope

$G=\left[g_1,\ldots,g_\gamma\right] \in \mathbb{B}^{n\times\gamma}$, a logical zonotope is defined as

$$\mathcal{L}=\left\{x\in\mathbb{B}^n \;\middle|\; x=c\overset{\gamma}{\underset{i=1}{\oplus}} g_i\beta_i,\ \beta_i\in\{0,1\}\right\}.$$

We use the shorthand notation $\mathcal{L}=\langle c,G\rangle$ for a logical zonotope.

Example 1: Consider a logical zonotope

$$\mathcal{L}=\left\langle \begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1&1\\0&1\end{bmatrix}\right\rangle.$$

With two generators, It represents the following four points:

$$\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}$$

by iterating over all possible binary values of $\beta\in\{00,01,10,11\}$.

Remark 1: Logical zonotopes are defined over $\mathbb{B}^n$ and are different from zonotopes [10], constrained zonotopes [11], and hybrid zonotopes [12] which are all defined over real vector space $\mathbb{R}^n$. Specifically, logical zonotopes are functional sets with Boolean symbols [13].

Logical zonotopes $\mathcal{L}$ can enclose up to 2^γ binary vectors with γ generators. In the following section, we will show that due to their construction, we can apply logical operations to a set of up to 2^γ binary vectors with a reduced computational complexity.

B. Minkowski Logical Operations

Given two sets of binary vectors, we often need to perform logical operations between the members of the two sets. In order to perform these logical operations efficiently, we define new logical operations that only operate on the generators of logical zonotopes instead of the members contained within the zonotopes. We will go through each logical operation and show that when applied to logical zonotopes, they either yield exact solutions or over-approximations.

1) Minkowski XOR ($\oplus$): Given the nature of the logical zonotope construction, we are able to compute the Minkowski XOR exactly and show that logical zonotopes are closed under Minkowski XOR as follows.

Lemma 1: Given two logical zonotopes $\mathcal{L}_1=\langle c_1,G_1\rangle$ and $\mathcal{L}_2=\langle c_2,G_2\rangle$, the Minkowski XOR is computed exactly as:

$$\mathcal{L}_1\oplus\mathcal{L}_2=\left\langle c_1\oplus c_2,\left[G_1,G_2\right]\right\rangle. \tag{7}$$

Proof: Let's denote the right hand side of (7) by $\mathcal{L}_\oplus$. We aim to prove that $\mathcal{L}_1\oplus\mathcal{L}_2\subseteq\mathcal{L}_\oplus$ and $\mathcal{L}_\oplus\subseteq\mathcal{L}_1\oplus\mathcal{L}_2$. Choose any $z_1\in\mathcal{L}_1$ and $z_2\in\mathcal{L}_2$

$$\exists\hat\beta_1: z_1=c_1\overset{\gamma_1}{\underset{i=1}{\oplus}} g_{1,i}\hat\beta_{1,i},$$

$$\exists\hat\beta_2: z_2=c_2\overset{\gamma_2}{\underset{i=1}{\oplus}} g_{2,i}\hat\beta_{2,i}.$$

Let $\hat\beta_{\oplus,1:\gamma_\oplus}=\left[\hat\beta_{1,1:\gamma_1},\hat\beta_{2,1:\gamma_2}\right]$ with $\gamma_\oplus=\gamma_1+\gamma_2$. Given that XOR is an associative and commutative gate, we have the following:

$$\begin{aligned}z_1\oplus z_2&=c_1\overset{\gamma_1}{\underset{i=1}{\oplus}} g_{1,i}\hat\beta_{1,i}\oplus c_2\overset{\gamma_2}{\underset{i=1}{\oplus}} g_{2,i}\hat\beta_{2,i}\\&=c_\oplus\overset{\gamma_1+\gamma_2}{\underset{i=1}{\oplus}} g_{\oplus,i}\hat\beta_{\oplus,i},\end{aligned}$$

where $c_\oplus=c_1\oplus c_2$ and $G_\oplus=\left[G_1,G_2\right]$ with $G_\oplus=\left[g_{\oplus,1},\ldots,g_{\oplus,\gamma_\oplus}\right]$. Thus, $z_1\oplus z_2\in\mathcal{L}_\oplus$ and therefore $\mathcal{L}_1\oplus\mathcal{L}_2\subseteq\mathcal{L}_\oplus$. Conversely, let $z_\oplus\in\mathcal{L}_\oplus$, then

$$\exists\hat\beta_\oplus: z_\oplus=c_\oplus\overset{\gamma_\oplus}{\underset{i=1}{\oplus}} g_{\oplus,i}\hat\beta_{\oplus,i}.$$

Partitioning $\hat\beta_{\oplus,1:\gamma_\oplus}=\left[\hat\beta_{1,1:\gamma_1},\hat\beta_{2,1:\gamma_2}\right]$, it follows that there exist $z_1\in\mathcal{L}_1$ and $z_2\in\mathcal{L}_2$ such that $z_\oplus=z_1\oplus z_2$. Therefore, $z_\oplus\in\mathcal{L}_1\oplus\mathcal{L}_2$ and $\mathcal{L}_\oplus\subseteq\mathcal{L}_1\oplus\mathcal{L}_2$. ■

2) Minkowski NOT ($\neg$), and XNOR ($\odot$): Given that we are able to do Minkowski XOR operation between logical zonotopes, we will able to find the Minkowski NOT with the following:

Corollary 1: Given a logical zonotope $\mathcal{L}=\langle c,G\rangle$, the Minkowski NOT can be computed exactly as:

$$\neg\mathcal{L}=\langle c\oplus 1,G\rangle. \tag{8}$$

Proof: The proof follows directly from truth table of XOR gate and $\neg\mathcal{L}=\mathcal{L}\oplus 1=\{z\oplus 1|z\in\mathcal{L}\}$ which results in inverting each binary vector in $\mathcal{L}$. ■

Similarly, we can perform the Minkowski XNOR exactly as follows.

$$\mathcal{L}_1\odot\mathcal{L}_2=\neg(\mathcal{L}_1\oplus\mathcal{L}_2). \tag{9}$$

3) Minkowski AND: Next, we over-approximate the Minkowski AND between two logical zonotopes as follows.

Lemma 2: Given two logical zonotopes $\mathcal{L}_1=\langle c_1,G_1\rangle$ and $\mathcal{L}_2=\langle c_2,G_2\rangle$, the Minkowski AND can be over-approximated by $\mathcal{L}_\wedge=\langle c_\wedge,G_\wedge\rangle$:

$$\mathcal{L}_1\mathcal{L}_2\subseteq\mathcal{L}_\wedge. \tag{10}$$

where $c_\wedge=c_1c_2$ and

$$\begin{aligned}G_\wedge=\big[&c_1g_{2,1},\ldots,c_1g_{2,\gamma_2},c_2g_{1,1},\ldots,c_2g_{1,\gamma_1},\\&g_{1,1}g_{2,1},g_{1,1}g_{2,2},\ldots,g_{1,\gamma_1}g_{2,\gamma_2}\big].\end{aligned} \tag{11}$$

Proof: Choose $z_1\in\mathcal{L}_1$ and $z_2\in\mathcal{L}_2$. Then, we have

$$\exists\hat\beta_1: z_1=c_1\overset{\gamma_1}{\underset{i=1}{\oplus}} g_{1,i}\hat\beta_{1,i}, \tag{12}$$

$$\exists\hat\beta_2: z_2=c_2\overset{\gamma_2}{\underset{i=1}{\oplus}} g_{2,i}\hat\beta_{2,i}. \tag{13}$$

AND-ing (12) and (13) results in

$$\begin{aligned}z_1z_2=&c_1c_2\overset{\gamma_2}{\underset{i=1}{\oplus}} c_1g_{2,i}\hat\beta_{2,i}\overset{\gamma_1}{\underset{i=1}{\oplus}} c_2g_{1,i}\hat\beta_{1,i}\\&\overset{\gamma_1,\gamma_2}{\underset{i=1,j=1}{\oplus}} g_{1,i}g_{2,j}\hat\beta_{1,i}\hat\beta_{2,j}.\end{aligned} \tag{14}$$

Combining the factors in

$$\hat\beta_\wedge=\left[\hat\beta_{1,1:\gamma_1},\hat\beta_{2,1:\gamma_2},\hat\beta_{1,1}\hat\beta_{2,1},\ldots,\hat\beta_{1,\gamma_1}\hat\beta_{2,\gamma_2}\right]$$

results in having $z_1z_2\in\mathcal{L}_\wedge$ and thus $\mathcal{L}_1\mathcal{L}_2\subseteq\mathcal{L}_\wedge$. ■

Remark 2: The term over-approximation in binary vectors with $\mathcal{L}_1\subseteq\mathcal{L}_2$ means that $\mathcal{L}_2$ contains at least all the binary vectors contained in $\mathcal{L}_1$.

TABLE I: Minkowski Logical Operation Complexity

Operation	Complexity	Type
XOR, NOT, XNOR	$\mathcal{O}(n)$	Exact
AND, NAND, OR, NOR	$\mathcal{O}(n\gamma_1\gamma_2)$	Over-approximation

4) Minkowski NAND ($\barwedge$): Given that we are able to do Minkowski AND and NOT operations, we will be able to do the Minkowski NAND as follows.

Corollary 2: Given two logical zonotopes $\mathcal{L}_1$=$\langle c_1, G_1\rangle$ and $\mathcal{L}_2$=$\langle c_2, G_2\rangle$, the Minkowski NAND can be over-approximated by:

$$\mathcal{L}_1 \barwedge \mathcal{L}_2 = \neg(\mathcal{L}_1\mathcal{L}_2). \tag{15}$$

Proof: The proof follows directly from truth table of NAND function and Lemma 2. ■

5) Minkowski OR ($\vee$), and NOR ($\barvee$): Given that we are able to NAND two sets which is a universal gate operation, we will be able to over-approximate the following logical Minkowski operations as shown next:

$$\mathcal{L}_1 \vee \mathcal{L}_2 = (\neg\mathcal{L}_1) \barwedge (\neg\mathcal{L}_2), \tag{16}$$

$$\mathcal{L}_1 \barvee \mathcal{L}_2 = \neg(\mathcal{L}_1 \vee \mathcal{L}_2). \tag{17}$$

6) Computational Complexity: For analyzing the computational complexity of the Minkowski logical operations, we have two logical zonotopes $\mathcal{L}_1 = \langle c_1, G_1\rangle$ and $\mathcal{L}_2 = \langle c_2, G_2\rangle$, where $c_1, c_2 \in \mathbb{B}^n$, $G_1 \in \mathbb{B}^{n\times\gamma_1}$ and $G_2 \in \mathbb{B}^{n\times\gamma_2}$. In Lemma 1, we see that the Minkowski XOR only consists of n binary operations for XORing the centers c_1 and c_2, resulting in a computational complexity of $\mathcal{O}(n)$. In other words, the complexity of Minkowski XOR scales linearly with the dimension of the binary vector space. Similarly, we see in Corollary 1 that applying the Minkowski NOT to a logical zonotope also has a computational complexity of $\mathcal{O}(n)$. By construction, XNOR also has complexity of $\mathcal{O}(n)$. In Lemma 2, we see that the Minkowski AND operation consists of ANDing the centers and generators with each other. Since each AND operation involves n binary operations, the resulting computational complexity is $\mathcal{O}(n\gamma_1\gamma_2)$. Since the complexity of the Minkowski AND operation dominates the Minkowski NAND, OR, and NOR operations, they also have complexities of $\mathcal{O}(n\gamma_1\gamma_2)$. We list the operations and their corresponding complexities in Table I.

C. Minkowski Semi-Tensor Product

Semi-tensor product has many application in different fields and is often useful in the analysis of logical systems [8]. In order to apply the Minkowski semi-tensor product to logical zonotopes, we first need to generalize logical zonotopes to logical matrix zonotopes.

Definition 5: (**Logical Matrix Zonotope**) Given a matrix $C \in \mathbb{B}^{m\times n}$ and $\gamma\in\mathbb{N}$ generator matrices $G_i \in \mathbb{B}^{m\times n}$ in a generator list $\bar{G}$=$\{G_1, \dots, G_\gamma\}$, a logical matrix zonotope is defined as

$$\mathcal{L} = \left\{ X \in \mathbb{B}^{m\times n} \;\middle|\; X = C \overset{\gamma}{\underset{i=1}{\oplus}} G_i\beta_i,\ \beta_i \in \{0,1\} \right\}.$$

We again use the shorthand notation $\mathcal{L} = \langle C, G\rangle$ for a logical matrix zonotope.

We define the Minkowski semi-tensor product with a slight abuse of the notation.

$$\mathcal{L}_1 \ltimes \mathcal{L}_2 = \{z_1 \ltimes z_2 | z_1 \in \mathcal{L}_1, z_2 \in \mathcal{L}_2\}. \tag{18}$$

We compute the Minkowski semi-tensor product between two logical matrix zonotopes as follows.

Lemma 3: Given two logical matrix zonotopes $\mathcal{L}_1 = \langle C_1, \bar{G}_1\rangle \in \mathbb{B}^{m\times n}$ and $\mathcal{L}_2 = \langle C_2, \bar{G}_2\rangle \in \mathbb{B}^{p\times q}$, the Minkowski semi-tensor product can be over-approximated by $\mathcal{L}_\ltimes = \langle C_\ltimes, \bar{G}_\ltimes\rangle$:

$$\mathcal{L}_1 \ltimes \mathcal{L}_2 \subseteq \mathcal{L}_\ltimes, \tag{19}$$

where

$$C_\ltimes = C_1 \ltimes C_2, \tag{20}$$

$$\bar{G}_\ltimes = \{C_1\ltimes G_{2,1}, \dots, C_1\ltimes G_{2,\gamma_1}, G_{1,1}\ltimes C_2, \dots, G_{1,\gamma_1}\ltimes C_2 \; G_{1,1}\ltimes G_{2,1}, \dots, G_{1,\gamma_1}\ltimes G_{2,\gamma_2}\}. \tag{21}$$

Proof: With s as the least common multiple of n and p, $s_1 = s/n$, and $s_2 = s/p$, choose $z \in \mathcal{L}_1 \ltimes \mathcal{L}_2$. Then, $\exists\hat{\beta}_1, \hat{\beta}_2$ such that

$$\begin{aligned}
z &= \left(C_1 \overset{\gamma_1}{\underset{i=1}{\oplus}} G_{1,i}\hat{\beta}_{1,i}\right) \ltimes \left(C_2 \overset{\gamma_2}{\underset{i=1}{\oplus}} G_{2,i}\hat{\beta}_{2,i}\right) \\
&= \left(\left(C_1 \overset{\gamma_1}{\underset{i=1}{\oplus}} G_{1,i}\hat{\beta}_{1,i}\right) \otimes I_{s_1}\right)\left(\left(C_2 \overset{\gamma_2}{\underset{i=1}{\oplus}} G_{2,i}\hat{\beta}_{2,i}\right) \otimes I_{s_2}\right) \\
&= \left(C_1\otimes I_{s_1} \overset{\gamma_1}{\underset{i=1}{\oplus}} G_{1,i}\hat{\beta}_{1,i}\otimes I_{s_1}\right)\left(C_2\otimes I_{s_2} \overset{\gamma_2}{\underset{i=1}{\oplus}} G_{2,i}\hat{\beta}_{2,i}\otimes I_{s_2}\right) \\
&= \left(C_1\otimes I_{s_1}\right)\left(C_2\otimes I_{s_2}\right) \overset{\gamma_2}{\underset{i=1}{\oplus}} \left(C_1\otimes I_{s_1}\right)\left(G_{2,i}\hat{\beta}_{2,i}\otimes I_{s_2}\right) \\
&\quad \overset{\gamma_1}{\underset{i=1}{\oplus}} \left(G_{1,i}\hat{\beta}_{1,i}\otimes I_{s_1}\right)\left(C_2\otimes I_{s_2}\right) \\
&\quad \overset{\gamma_1,\gamma_2}{\underset{i=1,j=1}{\oplus}} \left(G_{1,i}\hat{\beta}_{1,i}\otimes I_{s_1}\right)\left(G_{2,j}\hat{\beta}_{2,j}\otimes I_{s_2}\right).
\end{aligned}$$

Combining the factors in

$$\hat{\beta}_\ltimes = \left[\hat{\beta}_{1,1:\gamma_1}, \hat{\beta}_{2,1:\gamma_2}, \hat{\beta}_{1,1}\hat{\beta}_{2,1}, \dots, \hat{\beta}_{1,\gamma_1}\hat{\beta}_{2,\gamma_2}\right]$$

results in having $z \in \mathcal{L}_\ltimes$ and thus $\mathcal{L}_1 \ltimes \mathcal{L}_2 \subseteq \mathcal{L}_\ltimes$. ■

D. Logical Zonotope Containment and Generators Reduction

In certain scenarios, we might need to find a logical zonotope that contains at least the given binary vectors. One way to do that is as follows.

Lemma 4: Given a list $\mathcal{S} = \{s_1, \dots, s_p\}$ of p binary vectors in $\mathbb{B}^n$, the logical zonotope $\mathcal{L} = \langle c, G\rangle$ with $s_i \in \mathcal{L}, \forall i = \{1, \dots, p\}$, is given by

$$c = s_1, \tag{22}$$

$$g_{i-1} = s_i \oplus c,\ \forall i = \{2, \dots, p\}. \tag{23}$$

Proof: By considering the truth table of all values of β, we can find that the evaluation of $\mathcal{L}$ results in $c = s_1$ at one point and $g_{i-1} \oplus c = s_i \oplus c \oplus c = s_i$, $\forall i = \{2, \dots, p\}$, at other points. ■

We propose Algorithm 1 for reducing the number of generators while maintaining the same contained individual vectors. We first compute all the different binary vectors contained in the input logical zonotope

Algorithm 1: Function `reduce` to reduce the number of generators of a logical zonotope.

Input: A logical zonotope $\mathcal{L} = \langle c, G \rangle$ with large number γ of generators
Output: A logical zonotope $\mathcal{L}_r = \langle c_r, G_r \rangle$ with $\gamma_r \leq \gamma$ generators

1 $c_r = c$ // Function `reduce` does not change the center
2 $\mathcal{S}$= `evaluate`($\mathcal{L}$) // Compute a list $\mathcal{S}$ of all binary vectors contained in $\mathcal{L}$
3 $G_r = G$ // Start with the same number of generators
4 **for** $i = 1 : \gamma$ **do**
5 $\quad \mathcal{S}_r$= `evaluate`($\mathcal{L}_r \setminus g_i$) // Compute a list $\mathcal{S}_r$ of all binary vectors contained in $\mathcal{L}_r$ without the generator g_i
6 $\quad$ **if** *isequal*($\mathcal{S}, \mathcal{S}_r$) **then**
7 $\quad\quad G_r$ = `removeGenerator`(G_r,g_i) // Remove g_i from G_r
8 $\mathcal{L}_r = \langle c_r, G_r \rangle$

$\mathcal{L}$ in Line 2. Then, the algorithm checks the effect of removing each generator by computing the binary vectors contained in the logical zonotope without the removed generator in Line 5. The chosen generator is deleted if its removal does not remove any binary vector in Lines 6 and 7.

E. Reachability Analysis

We aim to over-approximate the exact reachable region of (1) which is defined in Definition 2 as follows.

Theorem 1: Given a logical function $f : \mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \rightarrow \mathbb{B}^{n_x}$ in (1) and a set of possible inputs $\mathcal{U}_k \subset \mathbb{B}^{n_u}$ and starting from initial set $\hat{\mathcal{R}}_0 \subset \mathbb{B}^{n_x}$ where $x(0) \in \hat{\mathcal{R}}_0$, then the reachable region computed as

$$\hat{\mathcal{R}}_{k+1} = f\left(\hat{\mathcal{R}}_k, \mathcal{U}_k\right) \tag{24}$$

using logical zonotopes operations over-approximates the exact reachable set, i.e., $\hat{\mathcal{R}}_{k+1} \supseteq \mathcal{R}_{k+1}$.

Proof: The logical function consists in general of XOR and NOT operations and any logical operations constructed from the NAND. $\forall x(k) \in \mathcal{R}_k$ and $u(k) \in \mathcal{U}_k$, we are able to compute Minkowski XOR and NOT exactly using Lemma 1 and Corollary 1 and over-approximate Minkowski NAND using Lemma 2 and Corollary 2. Thus, $\hat{\mathcal{R}}_{k+1} \supseteq \mathcal{R}_{k+1}$. ■

In Algorithm 2, we overview an algorithm based on Theorem 1 for N-step reachability analysis using logical zonotopes. First, in Line 1, we use Lemma 4 to convert the initial set of points $\mathcal{S}_0$ to get an initial logical zonotope $\bar{\mathcal{R}}_0$ which is further reduced to $\hat{\mathcal{R}}_0$ using Algorithm 2. Then, we iterate N times to find the Nth-step reachable set as a logical zonotope.

Algorithm 2: Reachability analysis for N-steps

Input: A logical function f, an initial set of points $\mathcal{S}_0$, a set of control input points $\mathcal{S}_{u,k}$, $\forall k = 1, \ldots, N$
Output: A reachable logical zonotope $\hat{\mathcal{R}}_N$ at the N-th step

1 $\bar{\mathcal{R}}_0$ = `enclosePoints`($\mathcal{S}_0$) // Enclose the set of points with a logical zonotope using Lemma 4
2 $\hat{\mathcal{R}}_0$ = `reduce`($\bar{\mathcal{R}}_0$) // Reduce the number of generators using Algorithm 1
3 $\bar{\mathcal{U}}_k$ = `enclosePoints`($\mathcal{S}_{u,k}$), $\forall k = 0, \ldots, N-1$
4 $\mathcal{U}_k$ = `reduce`($\bar{\mathcal{U}}_k$), $\forall k = 0, \ldots, N-1$
5 **for** $k = 0 : N-1$ **do**
6 $\quad \hat{\mathcal{R}}_{k+1} = f(\hat{\mathcal{R}}_k, \mathcal{U}_k)$ // Apply Minkowski logical operations

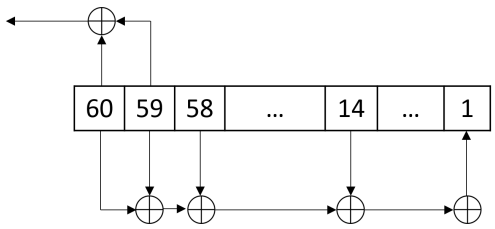


Fig. 2: LFSR A.

IV. Case Studies

To illustrate the use of operating over the generators in logical zonotopes, we present two different use cases. We first show how logical zonotopes can drastically improve the complexity of exhaustively searching for the key of an LFSR. Then, we formulate an intersection crossing problem, where we compare the computational complexity of BDDs, BCN-based semi-tensor products, and logical zonotopes when verifying the safety of four vehicles' intersection crossing protocol. The experiments were done on a processor 11^{th} Generation Intel(R) Core(TM) i7-1185G7 with 16.0 GB RAM.

A. Exhaustive Search for the Key of an LFSR

LFSRs are used intensively in many stream ciphers in order generate pseudo random longer keys from the input key. For simplicity we consider 60-bits LFSR A initialized with the input key K_A with length l_k. The operations on bit level are shown in Figure 2 where

$$A[1] = A[60] \oplus A[59] \oplus A[58] \oplus A[14],$$
$$\text{output} = A[60] \oplus A[59].$$

Each bit i of the output of the LFSR is XOR-ed with the message $m_A[i]$ to obtain one bit of the ciphertext $c_A[i]$.

Now consider that we aim to obtain the input key K_A using exhaustive search by trying out 2^{l_k} key values that can generate the cipher c_A from m_A with worst-case complexity $\mathcal{O}(2^{l_k})$ where $l_k = 60$ is the key length. Instead, we propose to use logical zonotopes in Algorithm 3 to decrease the complexity of the search

TABLE II: Execution Time (seconds) of exhaustive key search.

Key Size	Algorithm 3	Traditional Search
30	1.97	1.18×10^6
60	4.76	1.26×10^{15}
120	7.95	1.46×10^{33}

Algorithm 3: Exhaustive search for LFSR key using logical zonotopes

Input: A sequence of messages m_A and its ciphertexts c_A with length l_m
Output: The used Key $\mathcal{K}_A$ with length l_k in encrypting m_A

1 $\mathcal{L}_B$ =`encklosePoints`([0 1]) // enclose the points 0 and 1 by a logical zonotope
2 comb = $\{00, 01, 10, 11\}$
3 **for** $i = 3 : l_k$ **do**
4 $\mathcal{K}_A[i] = \mathcal{L}_B$ // assign the logical zonotope $\mathcal{L}_B$ to the key bits
5 **for** $i = 1 : 4$ **do**
6 $\mathcal{K}_A[1:2] = \text{comb}[i]$
7 $\mathcal{G}_A = \text{LFSR}(\mathcal{K}_A)$ // generate pseudo random numbers from the key $\mathcal{K}_A$
8 $\mathcal{C}_A = \mathcal{G}_A \oplus m_A$
9 **if** $\neg$*contains*$(\mathcal{C}_A, c_A)$ **then**
10 `continue`; // continue if $c_A \notin \mathcal{C}_A$
11 **for** $j = 3 : l_k$ **do**
12 $\mathcal{K}_A[j] = 0.$
13 $\mathcal{G}_A = \text{LFSR}(\mathcal{K}_A)$ // generate pseudo random numbers from the key $\mathcal{K}_A$
14 $\mathcal{C}_A = \mathcal{G}_A \oplus m_A$
15 **if** $\neg$*contains*$(\mathcal{C}_A, c_A)$ **then**
16 $\mathcal{K}_A[i] = 1$ // assign if $c_A \notin \mathcal{C}_A$
17 **if** *isequal*$(\mathcal{K}_A \oplus m_A, c_A)$ **then**
18 `return` $\mathcal{K}_A$

algorithm. We start by defining a logical zonotope $\mathcal{L}_B$, which contains 0 and 1 in line 1. Initially, we assign a logical zonotope to each bit of LFSR A in line 4 except the first two bits. Then, we set the first two bits of LFSR A to one of the 2^2 options of comb list in line 6. Then, we call the LFSR with the assigned key bits to get a list of logical zonotopes $\mathcal{G}_A$ with misuse of notations. The pseudo-random output of logical zonotopes $\mathcal{G}_A$ is XOR-ed with the message m_A to get a list of ciphertext logical zonotopes $\mathcal{C}_A$. If any cipher of the list c_A is not included in the corresponding logical zonotope $\mathcal{C}_A$, then the assigned two digits in line 6 are wrong, and we do not need to continue finding values for the remaining bits of LFSR A. After finding the correct two bits with $c_A \in \mathcal{C}_A$, we continue by assigning a zero to bit by bit in line 12. Then we generate the pseudo-random numbers $\mathcal{G}_A$ and XOR-ed it with the m_A to get the list of cipher logical zonotopes $\mathcal{C}_A$. The cipher logical zonotopes $\mathcal{C}_A$ are checked to contain the list of ciphers c_A and assign $\mathcal{K}_A$ in line 16, accordingly. We measured the execution time of Algorithm 3 with different key sizes in comparison to the execution time of traditional search in Table II. To compute the execution time of the traditional search, we multiply the number of iterations by the average execution time of a single iteration.

B. Safety Verification of an Intersection Crossing Protocol

In this example, we consider an intersection where four vehicles need to pass through the intersection, while avoiding collision. For comparison, we encode their respective crossing protocols as logical functions and verify the safety of their protocols through reachability analysis using BDDs, a BCN semi-tensor product-based approach, and logical zonotopes. We denote whether vehicle i is passing the intersection or not at time k by $p_i(k)$. Then, we denote whether vehicle i came first or not at time k by $c_i(k)$. We use control inputs $u_i^p(k)$ and $u_i^c(k)$ to denote the decision of vehicle i to pass or to come first at time k, respectively. For each vehicle $i = 1, \ldots, 4$, the intersection passing protocol is represented by the following:

$$p_i(k+1) = u_i^p(k) \neg p_i(k) \neg c_i(k). \tag{25}$$

Then, the logic behind coming first for each vehicle $i = 1, \ldots, 4$ is written as the following:

$$c_i(k+1) = \neg p_i(k+1)(u_i^c(k) \vee (\neg p_i(k) p_i(k+1))). \tag{26}$$

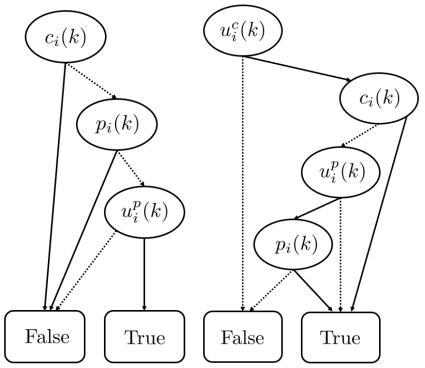
To perform reachability analysis, we initialize the crossing problem with the following conditions: $p_1(0) = 1$, $p_2(0) \in \{0,1\}$, $p_3(0) = 0$, $p_4(0) \in \{0,1\}$, $c_1(0) = 1$, $c_2(0) \in \{0,1\}$, $c_3(0) = 0$, $c_4(0) \in \{0,1\}$. To verify the passing protocol is always safe, under any decision made by each vehicle, we perform reachability analysis under the following uncertain control inputs: $u_1^p(k) \in \{0,1\}$, $u_2^p(k) = 0$, $u_3^p(k) \in \{0,1\}$, $u_4^p(k) = 0$, $u_1^c(k) \in \{0,1\}$, $u_2^c(k) \in \{0,1\}$, $u_3^c(k) \in \{0,1\}$, $u_4^c(k) \in \{0,1\}$, $k = 0, \ldots, N$.

Then, we construct BDDs for each formula and execute the reduced form of the BDDs with uncertainty which is illustrated in Figure 3. For the semi-tensor product-based approach with BCNs, we write state $x(k) = (\ltimes_{i=1}^4 p_i(k)) \ltimes (\ltimes_{i=1}^4 c_i(k))$. We write input $u(k) = (\ltimes_{i=1}^4 u_i^p(k)) \ltimes (\ltimes_{i=1}^4 u_i^c(k))$. The structure matrix L, which encodes (25)-(26), is a $2^8 \times 2^{16}$ matrix where 8 is the number of the states and 16 is the number of states and inputs. We perform reachability analysis for the BCN using $x(k+1) = L \ltimes u(k) \ltimes x(k)$ for all possible combinations. For reachability analysis with logical zonotopes, we represent each uncertain variable in (25)-(26) with a logical zonotope. We first compute the initial zonotope $\hat{\mathcal{R}}_0$ using Lemma 4 which contains the initial and certain states. Then, using Theorem 1, we compute the next reachable sets as logical zonotopes.

The execution time and the size of the reachable sets

TABLE III: Execution Time (seconds) and number of points in each set (size) for verifying an intersection crossing protocol.

	Zonotope		BDD		BCN	
Steps N	Time	Size	Time	Size	Time	Size
10	0.06	16	3.32	14	7.75	14
50	0.15	16	19.87	14	48.40	14
100	0.26	16	39.78	14	104.91	14
1000	1.84	16	406.60	14	1142.10	14



(a) $p_i(k+1)$ (b) $c_i(k+1)$

Fig. 3: Reduced BDDs for the intersection crossing example.

of the three approaches are presented in seconds in Table III. We note that reachability analysis using logical zonotopes provides better execution times when compared with reachability analysis with BDDs and semi-tensor products. Moreover, as the reachability analysis's time horizon increases, the reachability analysis's execution time with logical zonotopes increases slower than the other two methods. The logical zonotopes-based approach adds two extra points due to the over-approximation.

V. Conclusion

This work proposes a novel set representation for binary vectors called logical zonotope. Logical zonotopes can represent up to 2^γ binary vectors using only γ generators. We prove that applying different Minkowski logical operations to logical zonotopes always yields either exact solutions or over-approximations. In general, logical zonotopes allow for a variety of computationally efficient analyses of logical systems. In future work, we are investigating the potential of logical zonotopes for exploring the practical application of logical zonotopes in new use cases.

References

[1] A. J. Hu, *Techniques for efficient formal verification using binary decision diagrams*. stanford university, 1996.

[2] G. Cabodi, P. Camurati, L. Lavagno, and S. Quer, "Disjunctive partitioning and partial iterative squaring: An effective approach for symbolic traversal of large circuits," in *Proceedings of the 34th annual Design Automation Conference*, 1997, pp. 728–733.

[3] M. Byrod, B. Lennartson, A. Vahidi, and K. Akesson, "Efficient reachability analysis on modular discrete-event systems using binary decision diagrams," in *2006 8th International Workshop on Discrete Event Systems*. IEEE, 2006, pp. 288–293.

[4] F. Li and Y. Tang, "Robust reachability of boolean control networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 3, pp. 740–745, 2017.

[5] T. Leifeld, Z. Zhang, and P. Zhang, "Overview and comparison of approaches towards an algebraic description of discrete event systems," *Annual Reviews in Control*, vol. 48, pp. 80–88, 2019.

[6] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, M. Morari and L. Thiele, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 291–305.

[7] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, Technische Universität München, 2010.

[8] D. Cheng, H. Qi, and A. Xue, "A survey on semi-tensor product of matrices," *Journal of Systems Science and Complexity*, vol. 20, no. 2, pp. 304–322, 2007.

[9] H. Qi and D. Cheng, "Analysis and control of boolean networks: A semi-tensor product approach," in *7th Asian Control Conference*. IEEE, 2009, pp. 1352–1356.

[10] W. Kühn, "Rigorously computed orbits of dynamical systems without the wrapping effect," *Computing*, vol. 61, no. 1, pp. 47–67, 1998.

[11] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126–136, 2016.

[12] T. J. Bird, H. C. Pangborn, N. Jain, and J. P. Koeln, "Hybrid zonotopes: A new set representation for reachability analysis of mixed logical dynamical systems," *Automatica*, vol. 154, p. 111107, 2023.

[13] C. Combastel, "Functional sets with typed symbols: Mixed zonotopes and polynotopes for hybrid nonlinear reachability and filtering," *Automatica*, vol. 143, p. 110457, 2022.