Reachability Analysis Using Constrained Polynomial Logical Zonotopes

Ahmad Hafez, Frank J. Jiang, Karl H. Johansson, *Fellow, IEEE*, and Amr Alanwar, *Member, IEEE*

***Abstract*—This letter presents a novel approach for reachability analysis of using constrained polynomial logical zonotopes. We perform reachability analysis to compute the set of reachable states using a recently introduced set representation called polynomial logical zonotopes, enabling computationally efficient and exact reachability analysis on logical systems. Notably, polynomial logical zonotopes address the "curse of dimensionality" when analyzing the reachability of logical systems since the set representation can represent 2^h binary vectors using h generators. After finishing the reachability analysis, the formal verification involves verifying whether the intersection of the calculated reachable set and the unsafe set is empty or not. Polynomial logical zonotopes lack closure under intersections, prompting the formulation of constrained polynomial logical zonotopes, which preserve the computational efficiency and exactness of polynomial logical zonotopes for reachability analysis while enabling exact intersections. Additionally, an extensive empirical study is presented to demonstrate and validate the advantages of constrained polynomial logical zonotopes.**

***Index Terms*—Reachability analysis, logical zonotopes, formal verification.**

I. Introduction

REACHABILITY analysis is vital for logical systems, ensuring the avoidance of undesired states. However, it encounters a significant challenge in exhaustive state space exploration, often exhibiting exponential complexity, particularly as the number of state variables increases exponentially. There are various approaches to perform reachability analysis on logical systems. When logical systems are modeled as Boolean Networks or Boolean Control Networks (BCN), the reachability of the system relies on the semi-tensor product [1].

Manuscript received 8 March 2024; revised 9 May 2024; accepted 29 May 2024. Date of publication 14 June 2024; date of current version 24 September 2024. This work was supported in part by the Swedish Research Council, Swedish Research Council Distinguished Professor under Grant 2017-01078, and in part by the Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant. Recommended by Senior Editor J. Daafouz. *(Corresponding author: Ahmad Hafez.)*

Ahmad Hafez and Amr Alanwar are with the TUM School of Computation, Information and Technology, Department of Computer Engineering, Technical University of Munich, 74076 Heilbronn, Germany (e-mail: a.hafez@tum.de; alanwar@tum.de).

Frank J. Jiang and Karl H. Johansson are with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 100 44 Stockholm, Sweden, and also with Digital Futures, 114 28 Stockholm, Sweden (e-mail: frankji@kth.se; kallej@kth.se).

Digital Object Identifier 10.1109/LCSYS.2024.3414972

Yet, owing to the constraints imposed by point-wise operations and scaling limitations inherent in semi-tensor products, BCN-based approaches become unmanageable for logical systems with high dimensions [2]. Recently introduced for reachability analysis in logical systems, mixed zonotopes [3] and hybrid zonotopes combined with functional decomposition [4], provide an efficient approach to represent boolean functions while maintaining linear memory complexity for reachable sets over time and linear computational complexity relative to the system dimension. Alternatively, some reachability analysis algorithms utilize Binary Decision Diagrams (BDDs), known for their advantages in compact representation and efficient manipulation. However, they face challenges in representing complex systems, often requiring optimal variable ordering, a co-NP-complete problem.

Logical zonotopes, introduced to tackle computational complexity in reachability analysis on logical systems, are sets formed by XORing a binary vector with generators, which are combinations of other binary vectors. They can represent up to 2^h points using h generators. Logical zonotopes can facilitate efficient logical operations within the generator space and, thus, reachability analysis [5]. Despite their computational advantages, logical zonotopes lack support for exact ANDing in the generator space, leading to the exploration of polynomial logical zonotopes [6] as a potential solution. The challenge of achieving exact intersections remains for both polynomial logical zonotopes and logical zonotopes. This letter introduces constrained polynomial logical zonotopes, demonstrating their effectiveness in achieving exact intersections. This contribution strengthens the theoretical basis of set representations for binary vectors, particularly in logical operations, and promotes more robust computational frameworks across various applications. This letter's contributions are threefold:

- We introduce the formulation of constrained polynomial logical zonotopes and detail the application of both Minkowski and Exact XOR, AND, NOT, XNOR, NAND, OR, and NOR logical operations on constrained polynomial logical zonotopes.
- We present and prove the exactness of performing set intersection on two constrained polynomial logical zonotopes.
- We evaluate and compare the utility of constrained polynomial logical zonotopes with other set representations on the sets intersection and reachability analysis on a high-dimensional boolean function.

Readers can reproduce our results by utilizing our openly accessible library.[1]

The subsequent sections of this letter are arranged in the following manner. In Section II, the preliminary and problem statement are introduced. In Section III, constrained polynomial logical zonotopes are formulated, and the various operations they can accommodate are specified. We evaluate the proposed set representation in Section IV. Lastly, in Section V, we discuss the potential of both representations and address future prospects.

II. Problem Statement and Preliminaries

This section introduces the notation, preliminary definitions, and problem statement.

A. Notation

Binary set {0,1} and natural numbers are expressed by $\mathbb{B}$ and $\mathbb{N}$, respectively. The symbols $\barwedge$, $\barvee$, and $\odot$ represent the NAND, NOR, and XNOR operations, respectively. Similarly, the symbols $\oplus$, $\neg$, $\vee$, and $\wedge$ express the XOR, NOT, OR, and AND operations, respectively. To simplify notation, we will henceforth express $x \wedge y$ as $x\,y$ throughout the rest of this letter, acknowledging that this is a slight deviation from standard notation. Matrices are represented by uppercase letters, for example, $G \in \mathbb{B}^{n\times k}$, while sets are indicated by uppercase calligraphic letters, as seen in $\mathcal{Z} \subset \mathbb{B}^n$. Vectors and scalars are expressed using lowercase letters, such as $b \in \mathbb{B}^n$ with elements. The identity matrix of size $n \times n$ is symbolized as I_n. The vector $x \in \mathbb{B}^n$ is a binary vector of size $n \times 1$.

B. Preliminaries

Constrained polynomial logical zonotopes are constructed using the Minkowski XOR operation, which we define as follows.

Definition 1 (Minkowski XOR [5]): 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

$$\mathcal{L}_1 \oplus \mathcal{L}_2 = \{z_1 \oplus z_2 | z_1 \in \mathcal{L}_1, z_2 \in \mathcal{L}_2\}. \tag{1}$$

A constrained polynomial logical zonotope is a generalization of a polynomial logical zonotope, which is defined next.

Definition 2 (Polynomial Logical Zonotope [6]): Given a point $x \in \mathbb{B}^n$ and $h \in \mathbb{N}$ generator vectors in a generator matrix $G = \begin{bmatrix} g_1, \ldots, g_h \end{bmatrix} \in \mathbb{B}^{n\times h}$, dependent factors identifier $id \in \mathbb{N}^{1\times p}$, exponent matrix $E \in \mathbb{B}^{p\times h}$, a polynomial logical zonotope is defined as:

$$\mathcal{P} = \left\{ x \in \mathbb{B}^n \mid x = c \oplus \bigoplus_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) g_i, \alpha \in \{0,1\}^p \right\}. \tag{2}$$

$\mathcal{P} = \langle c, G, E, id \rangle$ is the shorthand notation for a polynomial logical zonotope.

A logical zonotope [5], [6] is a particular case of polynomial logical zonotopes where E is an identity matrix and without id reducing the shorthand definition to $\mathcal{L} = \langle c, G \rangle$.

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

In Minkowski logical operations, we'll use the `uniqueID` operator to generate a vector of unique integer identifiers. Conversely, for exact logical operations, we'll employ the `mergeID` operator to combine the identical identifiers for the constrained polynomial logical zonotopes, as outlined in [6].

C. Problem Statement

For a system that has a logical function $f : \mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \to \mathbb{B}^{n_x}$:

$$x(k+1) = f\big(x(k), u(k)\big) \tag{3}$$

where $x(k) \in \mathbb{B}^{n_x}$ and $u(k) \in \mathbb{B}^{n_u}$ are the state and the control input, respectively. The logical function f can be composed of different arrangements of logical operators: $\neg$, $\oplus$, $\wedge$, $\odot$, $\vee$, $\barvee$, and $\barwedge$. We will use binary sets to represent sets of states and inputs for (3). The reachable set of a system is defined as follows.

Definition 3 (Exact Reachable Set [5]): Given a set of possible inputs $\mathcal{U}_k \subset \mathbb{B}^{n_u}$ and a set of initial states $\mathcal{X}_0 \subset \mathbb{B}^{n_x}$, the exact reachable set $\mathcal{R}_N$ of (3) after N steps is

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

The goal is to calculate the exact forward reachable sets of the system defined in (3), using constrained polynomial logical zonotopes to generalize logical zonotopes and polynomial logical zonotopes.

III. Constrained Polynomial Logical Zonotopes

In this section, the constrained polynomial logical zonotope is introduced along with its set operations.

Example 1: Consider a digital circuit with the following boolean functions: from [6] with $B_i \in \mathbb{B}^{10}$ and $U_i \in \mathbb{B}^{10}$, $i = 1, 2, 3$,

$$B_1(k+1) = U_1(k) \vee (B_2(k) \odot B_1(k)), \tag{4}$$

$$B_2(k+1) = B_2(k) \odot (B_1(k) \wedge U_2(k)), \tag{5}$$

$$B_3(k+1) = B_3(k) \barwedge (U_2(k) \odot U_3(k)). \tag{6}$$

Our aim throughout this letter is to conduct reachability starting from multiple input values (sets of values) for U_i with $i = 1, 2, 3$ and determine the possible intersections of B_i with unsafe sets.

Next, we introduce the constrained polynomial logical zonotope.

Definition 4 (Constrained Polynomial Logical Zonotope): Given a point $c \in \mathbb{B}^n$ and $h \in \mathbb{N}$ generator vectors in a generator matrix $G = \begin{bmatrix} g_1, \ldots, g_h \end{bmatrix} \in \mathbb{B}^{n\times h}$, dependent factors identifier $id \in \mathbb{N}^{1\times p}$, exponent matrix $E \in \mathbb{B}^{p\times h}$, a constraint generator matrix $A \in \mathbb{B}^{m\times q}$, a constraint vector $b \in \mathbb{B}^m$, and a constraint exponent matrix $R \in \mathbb{B}^{p\times q}$, a constrained polynomial logical zonotope is defined as:

$$\mathcal{C} = \left\{ x \in \mathbb{B}^n \mid x = c \oplus \bigoplus_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) g_i \right. \\ \left. \Bigg| \bigoplus_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{k,i}} \right) A_{(.,i)} = b, \alpha \in \{0,1\}^p \right\}. \tag{7}$$

For a constrained polynomial logical zonotope, we use the shorthand notation $\mathcal{C} = \langle c, G, E, A, b, R, id \rangle$.

A. Set Operations

In the context of two sets comprising binary vectors, a recurrent necessity arises to execute logical operations between these sets' elements. We propose to have the logical operations occur at the generator space of constrained polynomial logical zonotope instead of directly interacting with the set elements. This section derives closed-form expressions of intersection and logical operations in the generator space of constrained polynomial logical zonotopes.

1) Sets Intersection: We start by showing the intersection between logical zonotopes, followed by the intersection between constrained polynomial logical zonotopes.

a) Logical Zonotopes intersections: The intersection between binary sets using logical zonotopes can only be overapproximated by the AND operation between the logical zonotopes, as shown in the following lemma.

Lemma 1: Given logical zonotopes $\mathcal{L}_1 = \langle c_1, G_1 \rangle$, and $\mathcal{L}_2 = \langle c_2, G_2 \rangle$ the intersection is overapproximated by $\mathcal{L}_\cap = \langle c_\wedge, G_\wedge \rangle$ as follows.

$$\mathcal{L}_1 \cap \mathcal{L}_2 \subset eq \mathcal{L}_\wedge \tag{8}$$

where $c_\wedge = c_1 c_2$ and

$$G_\wedge = [c_1 g_{2,1}, \ldots, c_1 g_{2,h_2}, c_2 g_{1,1}, \ldots, c_2 g_{1,h_1}, g_{1,1} g_{2,1}, g_{1,1} g_{2,2}, \ldots, g_{1,h_1} g_{2,h_2}]. \tag{9}$$

Following the lines of [7], we apply this to logical zonotopes. For a $z \in \mathcal{L}_1 \cap \mathcal{L}_2$

$$z = c_1 \bigoplus_{i=1}^{h_1} g_{1,i} \beta_{1,i}, \tag{10}$$

$$z = c_2 \bigoplus_{i=1}^{h_2} g_{2,i} \beta_{2,i}. \tag{11}$$

ANDing (10) and (11)

$$z = c_1 c_2 \bigoplus_{i=1}^{h_2} c_1 g_{2,i} \beta_{2,i} \bigoplus_{i=1}^{h_1} c_2 g_{1,i} \beta_{1,i} \bigoplus_{i=1,j=1}^{h_1,h_2} g_{1,i} g_{2,j} \beta_{1,i} \beta_{2,j}. \tag{12}$$

By representing the $\beta_{1,i}\beta_{2,j}$ by a new β we have an overapproximation as logical zonotopes can't represent the β's multiplication. This yields that: $z_1 z_2 \in \mathcal{L}_\wedge$ and thus $\mathcal{L}_1 \mathcal{L}_2 \subseteq \mathcal{L}_\wedge$. As the proved formula in (12) is the same as the ANDing overapproximation for logical zonotopes, then the computational complexity of the logical zonotope overapproximated intersection is $\mathcal{O}(n h_1 h_2)$ [5], [6].

b) Constrained polynomial logical Zonotopes intersection: Exact Intersection remained an unresolved issue with previous logical zonotopes, leading to overapproximations; on the other hand, the intersection between binary sets using constrained polynomial logical zonotopes is exact because of the addition of the constraints, as shown in the following lemma.

Lemma 2: Given constrained polynomial logical zonotopes $\mathcal{C}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1, id_1 \rangle$, and $\mathcal{C}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2, id_2 \rangle$ the intersection is computed as follows.

$$\mathcal{C}_1 \cap \mathcal{C}_2 = \left\langle c_1, G_1, \begin{bmatrix} E_1 \\ 0 \end{bmatrix}, \begin{bmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ 0 & 0 & G_1 & G_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ c_1 \oplus c_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} & E_1 & \mathbf{0} \\ \mathbf{0} & R_2 & \mathbf{0} & E_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle. \tag{13}$$

Following the lines of [8], we limit the factors α_k of $\mathcal{C}_1$ to the values of points that belong to $\mathcal{C}_2$ as follows.

$$c_1 \oplus \bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} = c_2 \oplus \bigoplus_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)}. \tag{14}$$

Using the self-inverse property of XOR, the constraint (14) can be rewritten as follows.

$$\bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{1(k,i)}} \right) G_{1(\cdot,i)} \oplus \bigoplus_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{2(k,i)}} \right) G_{2(\cdot,i)} = c_2 \oplus c_1. \tag{15}$$

This will add the constraint expressed by (15) to the other two constraints defined for the intersecting constraint polynomial logical zonotopes, and the intersection can be described as follows.

$$\begin{aligned} \mathcal{C}_1 \cap \mathcal{C}_2 = \Bigg\{ & c_1 \oplus \bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} p_1 \alpha_k^{E_{(k,i)}} \right) G_{1(\cdot,i)} \Bigg| \bigoplus_{i=1}^{q_1} \left(\prod_{k=1}^{p_1} \alpha_k^{R_{(k,i)}} \right) A_{1(\cdot,i)} = \\ & b_1, \bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_k^{E_{(k,i)}} \right) G_{1(\cdot,i)} \oplus \bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_2} \alpha_{p_1+k}^{E_{(k,i)}} \right) G_{2(\cdot,i)} = \\ & c_2 \oplus c_1, \bigoplus_{i=1}^{q_2} \left(\prod_{k=1}^{p_2} \alpha_{p_1+i}^{R_{(k,i)}} \right) A_{2(\cdot,i)} = b_2, \\ & \alpha_k, \alpha_{p_1+k} \in \{0, 1\} \Bigg\}. \end{aligned} \tag{16}$$

This leads to the same shorthand notation in (13). The computational complexity of this exact intersection operation is $\mathcal{O}(n + p_1 + p_2)$ [6].

2) Minkowski Logical Operations: In the next section, we derive the Minkowski logical operations; we start with Minkowski XOR, AND, NOT, XNOR, NAND, OR, and NOR.

a) Minkowski XOR: The Minkowski XOR over the generator domain of a constrained polynomial logical zonotope is carried out as follows.

Lemma 3: Given two constrained polynomial logical zonotopes $\mathcal{C}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1, id_1 \rangle$, and $\mathcal{C}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2, id_2 \rangle$ the Minkowski XOR is computed as follows.

$$\mathcal{C}_1 \oplus \mathcal{C}_2 = \left\langle c_1 \oplus c_2, [G_1 \; G_2], \begin{bmatrix} E_1 & \mathbf{0} \\ \mathbf{0} & E_2 \end{bmatrix}, \begin{bmatrix} A_1 & \mathbf{0} \\ \mathbf{0} & A_2 \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \begin{bmatrix} R_1 & \mathbf{0} \\ \mathbf{0} & R_2 \end{bmatrix}, \texttt{uniqueID}(p_1 + p_2) \right\rangle. \tag{17}$$

We denote the right-hand side of (17) by $\mathcal{C}_\oplus$. To prove (17), we need to prove that $\mathcal{C}_\oplus \subseteq \mathcal{C}_1 \oplus \mathcal{C}_2$ and $\mathcal{C}_1 \oplus \mathcal{C}_2 \subseteq \mathcal{C}_\oplus$. For $x_1 \in \mathcal{C}_1$ and $x_2 \in \mathcal{C}_2$, we have

$$\begin{aligned} \exists \alpha_1 : x_1 = \Bigg\{ & c_1 \oplus \bigoplus_{i=1}^{h_1} \left(\prod_{k=1}^{p_1} \alpha_{1,k}^{E_{1,(k,i)}} \right) g_{1,i} \\ & \Bigg| \bigoplus_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_{1,k}{}^{R_{1,(k,i)}} \right) A_{1,(\cdot,i)} = b_1 \Bigg\}. \end{aligned} \tag{18}$$

$$\begin{aligned} \exists \alpha_2 : x_2 = \Bigg\{ & c_2 \oplus \bigoplus_{i=1}^{h_2} \left(\prod_{k=1}^{p_2} \alpha_{2,k}^{E_{2,(k,i)}} \right) g_{2,i} \\ & \Bigg| \bigoplus_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_{2,k}{}^{R_{2,(k,i)}} \right) A_{2,(\cdot,i)} = b_2 \Bigg\}. \end{aligned} \tag{19}$$

Let $\alpha_{\oplus,1:p_\oplus} = [\alpha_{1,1:p_1}, \alpha_{2,1:p_2}]$ with $p_\oplus = p_1 + p_2$. As XOR is an associative and commutative gate, we get:

$$x_1 \oplus x_2 = c_\oplus \oplus \left(\bigoplus_{i=1}^{h_\oplus} \left(\prod_{k=1}^{p_\oplus} \alpha_{\oplus,k}^{E_{\oplus,(k,i)}} \right) g_{\oplus,i} \right)$$

with the constraint:

$$\bigoplus_{i=1}^{q_\oplus}\left(\prod_{k=1}^{p_\oplus}\alpha_k^{R_\oplus(k,i)}\right)A_{\oplus(\cdot,i)}=b_\oplus,\alpha_k,\in\{0,1\}$$

where $c_\oplus = c_1 \oplus c_2$, $G_\oplus = [G_1, G_2]$ with $G_\oplus$=[$g_{\oplus,1}$, …, $g_{\oplus,q_\oplus}$], $E_\oplus = \begin{bmatrix}E_1 & 0\\ 0 & E_2\end{bmatrix}$, $A_\oplus = \begin{bmatrix}A_1 & 0\\ 0 & A_2\end{bmatrix}$, $b_\oplus = \begin{bmatrix}b_1\\ b_2\end{bmatrix}$, $R_\oplus = \begin{bmatrix}R_1 & 0\\ 0 & R_2\end{bmatrix}$. Thus, $x_1 \oplus x_2 \in \mathcal{C}_\oplus$ and therefore $\mathcal{C}_1 \oplus \mathcal{C}_2 \subseteq \mathcal{C}_\oplus$. Conversely, let $x_\oplus \in \mathcal{C}_\oplus$, then

$$\exists\alpha_\oplus \;: x_\oplus = c_\oplus \oplus \bigoplus_{i=1}^{h_\oplus}\left(\prod_{k=1}^{p_\oplus}\alpha_{\oplus,k}^{E_{\oplus,(k,i)}}\right)g_{\oplus,i}.$$

Partitioning $\alpha_{\oplus,1:p_\oplus} = [\alpha_{1,1:p_1}, \alpha_{2,1:p_2}]$, it follows that there exist $x_1 \in \mathcal{C}_1$ and $x_2 \in \mathcal{C}_2$ such that $x_\oplus = x_1 \oplus x_2$. Therefore, $x_\oplus \in \mathcal{C}_1 \oplus \mathcal{C}_2$ and $\mathcal{C}_\oplus \subseteq \mathcal{C}_1 \oplus \mathcal{C}_2$.

The Minkowski XOR has a computational complexity of $\mathcal{O}(n + p_1 + p_2)$ [6].

b) Minkowski AND: This section shows the proof of the Minkowski AND for constrained polynomial logical zonotopes.

Lemma 4: Given two constrained polynomial logical zonotopes $\mathcal{C}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1, id_1\rangle$, and $\mathcal{C}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2, id_2\rangle$ the Minkowski AND is computed as follows.

$$\begin{aligned}\mathcal{C}_\wedge &= \mathcal{C}_1 \wedge \mathcal{C}_2\\ &= \left\langle c_1c_2, G_\wedge, E_\wedge, \begin{bmatrix}A_1 & \mathbf{0}\\ \mathbf{0} & A_2\end{bmatrix}, \begin{bmatrix}b_1\\ b_2\end{bmatrix}, \begin{bmatrix}R_1 & \mathbf{0}\\ \mathbf{0} & R_2\end{bmatrix},\right.\\ &\qquad \left.\texttt{uniqueID}(p_1 + p_2 + p_1p_2)\right\rangle, \end{aligned} \tag{20}$$

where $G_\wedge$ and $E_\wedge$ are as follows.

$$\begin{aligned}G_\wedge = \big[&c_1g_{2,1}, \ldots, c_1g_{2,h_2}, c_2g_{1,1}, \ldots, c_2g_{1,h_1},\\ &g_{1,1}g_{2,1}, \ldots, g_{1,h_1}g_{2,h_2}\big],\end{aligned} \tag{21}$$

$$\begin{aligned}E_\wedge = \Bigg[&\begin{bmatrix}0_{p_1\times 1}\\ E_{2,(.,1)}\end{bmatrix}, \ldots, \begin{bmatrix}0_{p_1\times 1}\\ E_{2,(.,h_2)}\end{bmatrix}, \begin{bmatrix}E_{1,(.,1)}\\ 0_{p_2\times 1}\end{bmatrix}, \ldots, \begin{bmatrix}E_{1,(.,h_1)}\\ 0_{p_2\times 1}\end{bmatrix},\\ &\begin{bmatrix}E_{1,(.,1)}\\ E_{2,(.,1)}\end{bmatrix}, \ldots, \begin{bmatrix}E_{1,(.,h_1)}\\ E_{2,(.,h_2)}\end{bmatrix}\Bigg].\end{aligned} \tag{22}$$

The detailed proof is in the extended version of this letter [9].

The Minkowski AND computational complexity is $\mathcal{O}(nh_1h_2 + p_1p_2)$ [6].

c) Other Minkowski Operations: Next, we derive the Minkowski NOT ($\neg$), XNOR ($\odot$), NAND ($\barwedge$), OR ($\vee$), NOR ($\barvee$) operations. In light of the XOR gate's truth table $\neg\mathcal{C} = \mathcal{C} \oplus 1 = \{\mathbf{x} \oplus 1 | x \in \mathcal{C}\}$ which inverts each binary vector in $\mathcal{C}$ [5], the Minkowski NOT can be computed as follows.

$$\neg\mathcal{C} = \langle c \oplus 1_{n\times 1}, G, E, A, b, R, id\rangle \tag{23}$$

The Minkowski NOT computational cost is $\mathcal{O}(n)$ as for polynomial logical zonotopes [6]. The Minkowski XNOR can be can be computed as follows.

$$\mathcal{C}_1 \odot \mathcal{C}_2 = \neg(\mathcal{C}_1 \oplus \mathcal{C}_2) \tag{24}$$

and Minkowski XNOR has a computational complexity of $\mathcal{O}(n + p_1 + p_2)$ [6]. Minkowski NAND can be computed as follows.

$$\mathcal{C}_1 \barwedge \mathcal{C}_2 = \neg(\mathcal{C}_1 \wedge \mathcal{C}_2) \tag{25}$$

The Minkowski NAND has a computational complexity of $\mathcal{O}(nh_1h_2 + p_1p_2)$ [6]. Utilizing the fact that the NAND gate is a universal gate, we can implement the Minkowski OR ($\vee$) and NOR ($\barvee$) using the NAND as follows.

$$\begin{aligned}\mathcal{C}_1 \vee \mathcal{C}_2 &= (\neg\mathcal{C}_1)\barwedge(\neg\mathcal{C}_2)\\ \mathcal{C}_1 \barvee \mathcal{C}_2 &= \neg(\mathcal{C}_1 \vee \mathcal{C}_2)\end{aligned}$$

The computational complexity of Minkowski OR is $\mathcal{O}(nh_1h_2 + p_1p_2)$, while for the Minkowski NOR the computational complexity is $\mathcal{O}(nh_1h_2 + p_1p_2)$ [6].

3) Exact Logical Operations: Building on the exact logical operations introduced in polynomial logical zonotopes in [6], we use the `mergeID` instead of `uniqueID`, as `uniqueID` is used in the case of Minkowski logical operations [6]. It is worth mentioning the dependency problem [3], [10], [11]; in simple words, it is a problem that arises when a binary set shows several times in a calculation, resulting in dealing with the recurring binary set independently. A proposed solution to this issue is using an identifier for each factor [11]. We start with executing `mergeID` on the two constrained polynomial logical zonotopes to combine identical identifiers for the constrained polynomial logical zonotopes as an essential step to perform exact logical operations. Next, we show the exact XOR($\bar{\oplus}$), and exact AND ($\bar{\wedge}$).

a) Exact XOR ($\bar{\oplus}$): The exact XOR is carried out as follows.

Lemma 5: Given two constrained polynomial logical zonotopes with a common id vector as follows. $\mathcal{C}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1, id\rangle$, and $\mathcal{C}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2, id\rangle$, the exact XOR is computed as follows.

$$\begin{aligned}\mathcal{C}_1 \oplus \mathcal{C}_2 = \Bigg\langle &c_1 \oplus c_2, [G_1, G_2], [E_1, E_2], \begin{bmatrix}A_1 & \mathbf{0}\\ \mathbf{0} & A_2\end{bmatrix}, \begin{bmatrix}b_1\\ b_2\end{bmatrix},\\ &\begin{bmatrix}R_1 & \mathbf{0}\\ \mathbf{0} & R_2\end{bmatrix}, id\Bigg\rangle.\end{aligned} \tag{26}$$

Proof: To prove (26) we follow the same steps to prove (17), but taking into consideration that we use here `mergeID` operator and not `uniqueID` operator used in the case of Minkowski logical operations [6], [8]. The exact XOR computational complexity is $\mathcal{O}(n + p_1p_2)$ [6]. ■

b) Exact AND ($\bar{\wedge}$): The exact AND is carried out as follows.

Lemma 6: Given two constrained polynomial logical zonotopes with a common id vector as follows. $\mathcal{C}_1 = \langle c_1, G_1, E_1, A_1, b_1, R_1, id\rangle$, and $\mathcal{C}_2 = \langle c_2, G_2, E_2, A_2, b_2, R_2, id\rangle$, the exact AND is computed as $\mathcal{C}_{\bar{\wedge}} = \langle c_{\bar{\wedge}}, G_{\bar{\wedge}}, E_{\bar{\wedge}}, A_{\bar{\wedge}}, b_{\bar{\wedge}}, R_{\bar{\wedge}}, id\rangle$ where

$$c_{\bar{\wedge}} = c_1c_2. \tag{27}$$

$$\begin{aligned}G_{\bar{\wedge}} = \big[&c_1g_{2,1}, \ldots, c_1g_{2,h_2}, c_2g_{1,1}, \ldots, c_2g_{1,h_1},\\ &g_{1,1}g_{2,1}, \ldots, g_{1,h_1}g_{2,h_2}\big],\end{aligned} \tag{28}$$

TABLE I
EXECUTION TIME (MILLISECONDS) FOR LOGICAL ZONOTOPES INTERSECTION AND NUMBER OF INTERSECTION POINTS

	Log. Zonotope		Poly. log. Zonotope		Con. poly. log Zonotope	
Dimensions	Time	Size	Time	Size	Time	Size
5	0.436	32	0.591	20	0.350	4
7	0.437	128	0.668	50	0.355	2
10	0.451	512	0.727	183	0.374	2

$$E_{\hat{\wedge}} = \Big[E_{2,(.,1)}, \ldots, E_{2,(.,h_2)}, E_{1,(.,1)}, \ldots, E_{1,(.,h_1)}, max(E_{1,(.,1)}, E_{2,(.,1)}), \ldots, max(E_{1,(.,h_1)}, E_{2,(.,h_2)})\Big]. \quad (29)$$

To prove the exact AND, we follow the same steps as the proof of Minkowski AND and use the `mergeID` instead of `uniqueID`. The exact AND computational complexity is $\mathcal{O}(nh_1h_2 + p_1p_2)$ [6]. Similarly, the exact NOT proof will be the same as the Minkowski NOT proof, and the id will remain the same. As we have exact AND and exact NOT, we can get the exact NAND, and as stated, the NAND is a universal gate for all exact logic operations.

B. Reachability Analysis

We use the constrained polynomial logical zonotope to have an exact reachability analysis of (3), defined in Definition 3. This is provided in the following theorem.

Theorem 1: Given a logical function $f{:}\mathbb{B}^{n_x} \times \mathbb{B}^{n_u} \to \mathbb{B}^{n_x}$ in (3) and starting from initial polynomial logical zonotope $\mathcal{R}_0 \subset \mathbb{B}^{n_x}$ where $x(0) \in \mathcal{R}_0$, then the exact reachable region computed as:

$$\mathcal{R}_{k+1} = f\big(\mathcal{R}_k, \mathcal{U}_k\big) \quad (30)$$

The logical function comprises XOR and NOT operations, and any logical operations formed using NAND. $\forall x(k) \in \mathcal{R}_k$ and $u(k) \in \mathcal{U}_k$, Minkowski XOR and NOT can be computed exactly utilizing Lemma 3 and (23), Minkowski NAND using (25). Additionally, we can perform exact XOR using Lemma 5, exact AND using Lemma 6, and exact NAND using exact AND and NOT operations. The universality of the NAND gate indicates that it can serve as the basis for constructing any other logical gates. Constrained polynomial logical zonotopes enable the computation of exact intersections between binary sets and unsafe sets. Afterward, a check is conducted in the point domain to determine whether there is an intersection or if the resulting set remains empty. This involves converting from the generator domain to the point domain by considering all possible combinations of the parameters that satisfy the constraint, which is computationally expensive.

IV. CASE STUDIES

We show various practical use cases to clarify the utilization of operating over the generators' domain of constrained polynomial logical zonotopes. We first illustrate the application of constrained polynomial logical zonotopes for performing intersection and then reachability analysis on a Boolean function with a high-dimensional domain. All of the experiments are executed on a processor Intel Xeon CPU E5-1650 v2 @ 3.50GHz (12 CPUs), with 32.0 GB RAM.

A. Logical Zonotopes Intersection

As previously mentioned, a significant contribution of this letter is the introduction of exact intersections to logical zonotopes. In this experiment, we generated two random logical zonotopes using two random centers and two random generator matrices. Subsequently, we intersected them using overapproximation for both logical zonotopes and polynomial logical zonotopes, while applying an exact intersection for constrained polynomial logical zonotopes.

The results presented in Table I highlight that constrained polynomial logical zonotopes achieve the fewest intersection points. Additionally, the overapproximation intersection for polynomial logical zonotopes yields fewer points compared to logical zonotopes, attributed to the presence of exact logical operations in polynomial logical zonotopes. Furthermore, the execution time for logical zonotope intersection is observed to be lower than that of polynomial logical zonotopes, consistent with the computational complexity of AND operations for both types, as discussed in [6]. The exact intersection is only possible with constrained polynomial logical zonotopes, and its execution time is lower than that of the overapproximation intersection for logical zonotopes and polynomial logical zonotopes. The findings indicate that the execution time is influenced by the dimensions of the inputs, with higher dimensions correlating with longer execution times.

B. Reachability Analysis on a High-Dimensional Boolean Function

Consider the Boolean functions defined in the motivating example in Section III. In the context of reachability analysis, we start by assigning sets comprising two potential values to $B_1(0)$, $B_2(0)$, and $B_3(0)$. Subsequently, we gauge the temporal efficiency of the reachability analysis initiated from this initial condition employing BDDs and logical zonotopes. In this example, we opt not to compare with the semi-tensor product-based approach of BCNs due to the impracticality of handling the structure matrix for high-dimensional systems. The structure matrix in such approaches expands exponentially with the number of states and inputs [6].

The reachability analysis using BDDs for $N > 2$ with the provided variable ordering did not conclude within a reasonable timeframe. Consequently, following [6], we opted to utilize the average execution time for a single iteration, multiplying this time to estimate the total duration for the reachability analysis. The findings are presented in Table II. In high-dimensional systems, logical zonotopes yield a substantial overapproximation. Conversely, constrained polynomial

TABLE II
EXECUTION TIME (SECONDS) FOR REACHABILITY ANALYSIS OF A BOOLEAN FUNCTION (*ESTIMATED TIMES)

	Log. Zonotope		Poly. log. Zonotope		Con. poly. log. Zonotope		BDD	
Steps N	Time	Size	Time	Size	Time	Size	Time	Size
2	0.094	768	0.103	211	0.113	211	0.772	211
3	0.103	896	0.115	580	0.124	580	3.56×10^5*	580
4	0.107	896	0.168	580	0.237	580	4.67×10^6*	-
5	0.184	896	2.054	580	2.090	580	$> 10^7$*	-

logical zonotopes and polynomial logical zonotopes deliver exact reachability analysis with a minimal execution time.

In polynomial scenarios, Minkowski AND operations are feasible with both polynomial logical zonotopes and constrained polynomial logical zonotopes. This results in a saturation of point count (size) at 580 points after three steps. However, in logical zonotopes, where the AND operation is an overapproximation rather than exact, more points are generated compared to polynomial cases. Constrained polynomial logical zonotopes took execution time slightly longer than that of polynomial logical zonotopes.

V. CONCLUSION

In this letter, we advocate the utilization of constrained polynomial logical zonotopes to extend the applicability of polynomial logical zonotopes for reachability analysis in logical systems. Constrained polynomial logical zonotopes are constructed by adding a constraint to a polynomial logical zonotope, which allows for the exact computation of the intersection. In different use cases, it was shown that constrained polynomial logical zonotopes were able to perform computationally efficient logical operations and exact set intersections. More use cases for constrained polynomial logical zonotope-based reachability analysis and search algorithms will be investigated in future work.

REFERENCES

[1] F. Li and Y. Tang, "Robust reachability of boolean control networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 3, pp. 740–745, Jun. 2017.
[2] T. Leifeld, Z. Zhang, and P. Zhang, "Overview and comparison of approaches towards an algebraic description of discrete event systems," *Annu. Rev. Control*, vol. 48, pp. 80–88, Nov. 2019.
[3] C. Combastel, "Functional sets with typed symbols: Mixed zonotopes and polynotopes for hybrid nonlinear reachability and filtering," *Automatica*, vol. 143, Sep. 2022, Art. no. 110457.
[4] J. A. Siefert, T. J. Bird, J. P. Koeln, N. Jain, and H. C. Pangborn, "Reachability analysis of nonlinear systems using hybrid zonotopes and functional decomposition," 2023, *arXiv:2304.06827*.
[5] A. Alanwar, F. J. Jiang, S. Amin, and K. H. Johansson, "Logical zonotopes: A set representation for the formal verification of boolean functions," in *Proc. 62nd IEEE Conf. Decis. Control*, 2023, pp. 60–66.
[6] A. Alanwar, F. J. Jiang, and K. H. Johansson, "Polynomial logical zonotopes: A set representation for reachability analysis of logical systems," 2023, *arXiv:2306.12508*.
[7] A. Alanwar, J. J. Rath, H. Said, K. H. Johansson, and M. Althoff, "Distributed set-based observers using diffusion strategies," *J. Frankl. Inst.*, vol. 360, no. 10, pp. 6976–6993, 2023.
[8] N. Kochdumper and M. Althoff, "Constrained polynomial zonotopes," *Acta Informatica*, vol. 60, pp. 279–316, Sep. 2023.
[9] A. Hafez, F. J. Jiang, K. H. Johansson, and A. Alanwar, "Reachability analysis using constrained polynomial logical zonotopes," 2024, *arXiv:2403.18564*.
[10] C. Combastel and A. Zolghadri, "A distributed Kalman filter with symbolic zonotopes and unique symbols provider for robust state estimation in CPS," *Int. J. Control*, vol. 93, no. 11, pp. 2596–2612, 2020.
[11] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Trans. Autom. Control*, vol. 66, no. 9, pp. 4043–4058, Sep. 2021.