Note: This document is a working document and isn't not fully fleshed out (or as formal as desired)

1 General Sets

1.1 Dimension-awareness

[1] [2] [3]

Definition 1 (Dimensional Aware Set Definition) *Let $\mathcal{D}_V : \nu_i \to i$ be a dictionary that relates the key ν_i to each of the vector fields $\mathbb{V}_i \subset \mathbb{V}$, $\forall_{i \in \mathbb{N}}$.*
Let $X = (X_1, X_2, \dots, X_n) \subseteq (\mathbb{V}_1, \mathbb{V}_2, \dots, \mathbb{V}_n)$ and $\mathcal{D}_X = \{\nu_1 \to 1, \dots, \nu_n \to n\} \subseteq \mathcal{D}_V$. The set and the dictionary $(X, \mathcal{D}_X) \subseteq (V, \mathcal{D}_V)$ is then considered dimensional-aware as the dictionary relates the key ν_i with the associated set $X_i \subseteq \mathbb{V}_i$.
Denote $(\mathbb{V}, \mathcal{D}_V)$ as a space of all dimensionally-aware sets.

do we want to use "dictionary" or some "indexed family" type definition?

Definition 2 (Dimensional Aware Relations) *A dimensionally-aware relation, $\overset{*}{\mathcal{R}}$, between the two dimensionally-aware sets $(X, \mathcal{D}_X), (Y, \mathcal{D}_Y) \subseteq (V, \mathcal{D}_V)$ is defined differently depending on the keys in each associated dictionary:*

1. *For unique keys, ν, within each dictionary $\mathcal{D}_X$ and $\mathcal{D}_Y$, the associated sets, X_i and Y_j are kept and associated within $\mathcal{D}_C$ to the appropriate index in the new collection as C_k.*
2. *For keys that are shared between the two dictionaries $\nu_i \in \mathcal{D}_X = \nu_j \in \mathcal{D}_Y$ the relation is then applied to the corresponding sets $C_k = X_i \mathcal{R} Y_j$ and associated within $\mathcal{D}_C$ to the appropriate index k in the new collection.*

look at the best way to notate this... do we need to make ν within a set? does this make sense as is?

(i.e)

$$\begin{aligned}(X, \mathcal{D}_X) \overset{*}{\mathcal{R}} (Y, \mathcal{D}_Y) &= (C, \mathcal{D}_C) \\ &= ((X_1, \dots, X_{n_a}), \mathcal{D}_X) \overset{*}{\mathcal{R}} ((Y_1, \dots, Y_{n_b}), \mathcal{D}_Y) \\ &= (([X_i]_{\forall_{i \in I_x}}, [Y_j]_{\forall_{j \in J_y}}, [X_i \mathcal{R} Y_j]_{\forall i \in I_{xy}, j \in J_{xy}}), \mathcal{D}_C)\end{aligned} \tag{1}$$

where $I_x = \{i \mid \nu_i^x \notin \mathcal{D}_Y\}$, $J_y = \{j \mid \nu_j^y \notin \mathcal{D}_X\}$, $I_{xy} = \{i \mid \nu_i^x \in \mathcal{D}_Y\}$, $J_{xy} = \{j \mid \nu_j^y \in \mathcal{D}_X\}$, and $\mathcal{D}_C : \nu_k^c \to k$ relates the shared keys to the appropriate sets in collection C.

look at notation...

Remark 1 (Dimension-aware set operations) ***Since set operations are in essence just relations, any set operations between two dimension-aware sets can be implemented as described with the appropriate relation applied upon the appropriate dimension.***

This isn't fully true... need to determine a beter way other then relation as idk if a minkowski-sum specifically is a relation

Remark 2 (Additional Remark on Dimension-aware Set Operations) *Under dimension-aware operations, any dimensions that are unique (i.e. not defined in the other dictionary) then either $\emptyset$ (for cartisian product, union, minkowski sum...) or $\mathbb{V}$ (intersection, ...) may instead be associated with the specific key, ν, and the operation being applied between the appropriate sets would result in the same operation.*

[1] The specific-base variable-type of any set of dimensions is not restricted to any particular $\mathbb{R}$,$\mathbb{Z}$,$\mathbb{C}$ (or in MALTAB class()=double,sym,int,optimvar,etc.) however if the notation of $\mathbb{R}^n$ is used this should still be true...

[2] Should we restrict to just a collection of 1-D dimensions or keep it general to any dimensional spaces being labeled together??? I'm thinking the later as I'm not restricting it to n-dimensional fields anyway...

[3] So I think we could restrict this to a banach space or less to a field, but technically I think any vector space should work... And if we restrict to a space consisting of 1-D (scaler?) for each dimension that could also work...

1.2 Memory-based ("Lazy")-set Definition

[4]

The premise of memory-preservation that this general-notation will be pursuing will consist of having origin sets who are placed through transformation and arbitrary relations (i.e. set-operations) that will be saved as constraints upon the yet to be evaluated preserved-memory set.

Definition 3 (Memory-encoded Sets) *Let $X_1, \dots, X_{n_x} \subseteq \mathbb{V}_X$ and define dictionary $\mathcal{D}_X : \nu_i^X \to i$. The collection of sets $([X_i]_{i=1,\dots,n_x}, \mathcal{D}_X)$ is said to be* <u>memory-encoded</u> *as the dictionary provides an association of between keys and the underlying sets.*

Definition 4 (Memory-preserving Transformation) *A transformation is considered (fully)* **memory-preserving** *if a portion of (all of) the memory-encoded within an origin sets is maintained once transformed.*

Definition 5 *Let $([X_i]_{i=1,\dots,n_x}, \mathcal{D}_X)$ and $([Y_j]_{j=1,\dots,n_y}, \mathcal{D}_Y)$ be memory-encoded sets.*
A transformation $f : V_X \to V_{f(X)}$ is considered <u>fully memory-preserving</u> *if the origin sets can be fully reconstructed from the transformed set. (i.e.) $([X_i]_{i=1,\dots,n_x}, \mathcal{D}_X)$ can be reconstructed from $(f([X_i]_{i=1,\dots,n_x}), \mathcal{D}_{f(X)})$ given knowledge of $f(x)$.*
A transformation, $g_k(x, y) : V_X \times V_Y \to V_{g(x,y)}$, is considered <u>origin memory-preserving</u> *if the relationship between an origin set and the transformed set is maintained.*[5]

1.2.1 Simple (Affine) transformation:

For memory-encoded sets $([X_i]_{i=1,\dots,n_x}, \mathcal{D}_X)$ being placed through the affine transformation $f(x) = Mx + b$, the new sets can be transformed with no adjustment to the dictionary but the individual sets will now be endowed with an affine mapping. (i.e.)

$$([X_i]_{i=1,\dots,n_x}, \mathcal{D}_X) \overset{f(x)}{\to} (f([X_i]_{i=1,\dots,n_x}), \mathcal{D}_{f(X)}) = ([f(X_i)]_{i=1,\dots,n_x}, \mathcal{D}_{f(X)}) = ([MX_i+b]_{i=1,\dots,n_x}, \mathcal{D}_{f(X)})$$

In this instance it is possible for $\mathcal{D}_{f(X)} = \mathcal{D}_X$, or at least the simple transformation ensures that the dictionary would have a one-to-one correspondence with the new dictionary.
In a more general sense, the origin sets could also be retained in a new structure that would directly relate the origin sets with the new set:

$$\left([[X_i]_{i=1,\dots,n_x}], \mathcal{D}_X\right) \overset{f(x)}{\to} \left(\begin{bmatrix} [X_i]_{i=1,\dots,n_x} \\ [MX_i + b]_{i=1,\dots,n_x} \end{bmatrix}, \begin{bmatrix} \mathcal{D}_X \\ \mathcal{D}_{f(x)} \end{bmatrix}\right)$$

Note that the dictionary $\mathcal{D}_{f(X)}$ continues to indicate that the X_i sets in the transformed sets are the same/related/from to the origin set in the same structure.
[6]

1.2.2 General transformation

[4] The previous attempt at "Lazy"-set memory-based operation is not as well-structured or as general as the dimension-aware version, thus I'm going to have to attempt it differently.

[5] Potentially requiring the previous set and explicit knowledge of the transformation

[6] The way this dictionary points and relates the origin-sets to within the other sets is not well-defined... within a zonotope this can just be related to the factors as the collumns, but I'm not sure how this is explicitly described within a generic structure outside of this being a thing

this is only pair-wise... would general transformation on only one set make mores sense?...

eliminate

For memory-encoded sets $([X_i]_{i=1,\dots,m}, \mathcal{D}_X)$ and $([Y_j]_{j=1,\dots,m}, \mathcal{D}_Y)$ $(m < n_x, n_y)$ being placed through the general transformation of $g_k(x,y)$ (of which k is distinct for each pair of i and j), the memory-preservation transformation [7] results in:

$$\left(\begin{bmatrix}[X_i]_{i=1,\dots,m}\\ [Y_j]_{j=1,\dots,m}\end{bmatrix}, \begin{bmatrix}\mathcal{D}_X\\ \mathcal{D}_Y\end{bmatrix}\right) \overset{g_k(x,y)}{\rightarrow} \left(\left[[g_k(X_i,Y_j)]_{\forall_{i=j=k=1,\dots,m}}\right], \mathcal{D}_{g_k(X,Y)}\right)$$

Unlike the simple transformations, the directly inverse of the operation and recreation of the origin sets from the transformed sets cannot be guaranteed in general. If all of the transformations are bijective, then the $\mathcal{D}_{g_k(X,Y)}$ could likely contain all the information needed to recreate the origin sets, but still the structure maintaining the previous sets may be beneficial to describe the complete relationship until a specific projection is needed:

$$\left(\begin{bmatrix}[X_i]_{i=1,\dots,m}\\ [Y_j]_{j=1,\dots,m}\end{bmatrix}, \begin{bmatrix}\mathcal{D}_X\\ \mathcal{D}_Y\end{bmatrix}\right) \overset{g_k(x,y)}{\rightarrow} \left(\begin{bmatrix}[X_i]_{i=1,\dots,m}\\ [Y_j]_{j=1,\dots,m}\\ [g_k(X_i,Y_j)]_{i=j=k=1,\dots,m}\end{bmatrix}, \begin{bmatrix}\mathcal{D}_X\\ \mathcal{D}_Y\\ \mathcal{D}_{g_k(X,Y)}\end{bmatrix}\right)$$

Remark 3 (Usage of structure for input-output mapping) *The structure defined above maintains the relationship between input and output for the generic transformation $g_k(X,Y)$, and thus the set can be described as a memory-encoded transformation $(\Phi(X,Y), \mathcal{D}_\Phi)$:*

$$\Phi(X,Y) = \left\{ \begin{bmatrix}[x_i]_{i=1,\dots,m}\\ [y_i]_{j=1,\dots,m}\\ [g_k(x_i,y_j)]_{i=j=k=1,\dots,m}\end{bmatrix} \mid \begin{matrix}[x_i \in X_i]_{i=1,\dots,m}\\ [y_j \in Y_j]_{j=1,\dots,m}\end{matrix} \right\}$$

where $\mathcal{D}_\Phi$ provides appropriate labeling.

1.2.3 General Operation Implementations

Introduction of set operations to this structure becomes a bit more difficult. Tracking the origin of sets through transformations is simple enough, but retaining all the information may become more difficult under successive operations...
To only maintain the origin-set information, the "lazy"-set approach of not evaluating until the end may retain all the information desired:

using $\mathcal{R}$ as the operation symbol because idk what is better...

Example 1 (relation between all odd-numbered sets) *Let $([X_i]_{i=1,\dots,m}, \mathcal{D}_X)$ be a set of memory-encoded sets.*

$$\left((((X_1 \;\mathcal{R}\; X_3) \;\mathcal{R}\; X_5) \;\mathcal{R}\; \dots), \mathcal{D}_{[X]_{i=1,3,\dots}}\right)$$

Alternatively,

$$\left(\begin{bmatrix}[X_i]_{i=1,\dots,m}\\ X_1 \;\mathcal{R}\; X_3\\ (X_1 \;\mathcal{R}\; X_3) \;\mathcal{R}\; X_5\\ ((X_1 \;\mathcal{R}\; X_3) \;\mathcal{R}\; X_5) \;\mathcal{R}\; X_7\\ \vdots\end{bmatrix}, \begin{bmatrix}\mathcal{D}_X\\ \mathcal{D}_{1\;\mathcal{R}\;3}\\ \mathcal{D}_{1,3\;\mathcal{R}\;5}\\ \mathcal{D}_{1,3,5\;\mathcal{R}\;7}\\ \vdots\end{bmatrix}\right)$$

[7] successor-set structure?

where each of the new sets is in-terms of the previous set?
Or potentially a "constraint" can be implemented to represent each subsequent set operation:

$$\left(\begin{bmatrix} [X_i]_{i=1,\dots,m} \\ X_{1,3} \\ X_{1,3,5} \\ X_{1,3,5,7} \\ \vdots \end{bmatrix}, \begin{bmatrix} \mathcal{D}_X \\ \mathcal{D}_{1,3} \\ \mathcal{D}_{1,3,5} \\ \mathcal{D}_{1,3,5,7} \\ \vdots \end{bmatrix}, \begin{bmatrix} \emptyset, \\ X_{1,3} = X_1 \ \mathcal{R} \ X_3 \\ X_{1,3,5} = X_{1,3} \ \mathcal{R} \ X_5 \\ X_{1,3,5,7} = X_{1,3,5} \ \mathcal{R} \ X_7 \\ \vdots \end{bmatrix}\right)$$

In this case the set-operation constraints would rely on the dictionaries to create said constraints. Note: this is how this information is encoded within the zonotope-based sets but I'm not sure if this is inherently the best way to represent this information in this generic set-based representation.

1.3 Dimensional-awareness and memory-preservation

2 Zonotope-based Set Implementation

Zonotopes are built upon the basic principles of a Minkowski Sum to represent closed and bounded sets within an n-dimensional vector space.

is vector-space too broad? does it need to be restricted to banach space?

Standard zonotopes are represented strictly as the minkowski sum of line-segments, often described in an affine form constructed of a center-vector ($c \in \mathbb{V}^n$) and a generator matrix ($G \in \mathbb{V}^{n \times n_g}$) with factors ($\xi_i \in [0,1]$) acting upon individual generators$g_i \in \mathbb{V}^n$:

$$Z = \{G, c\} = \left\{c + G\xi \mid \forall_{\xi \in \mathbb{R}^{n_g}} \|\xi\|_\infty \leq 1\right\}$$

In order to represent non-symetric convex polytopes, additional constraints can be introduced upon the factors using constraint matrix $X \in \mathbb{R}^{n_c \times n_g}$ and vector $b \in \mathbb{R}^{n_c}$, ($X\xi = b$), to break symmetry and represent more complicated shapes:

$$Z = \{G, c, A, b\} = \left\{c + G\xi \mid \forall_{\xi \in \mathbb{R}^{n_g}} \|\xi\|_\infty \leq 1 \wedge A\xi = b\right\}$$

Another extension involves the introduction of a restriction upon certain factors to become discrete factors (i.e. $\xi_i \in \{-1, 1\}$). This "hybrid-zonotope" can be represented as two separate generator and constraint matrices (as is common in other literature) [8] or alternatively an additional constraint can be placed upon certain generators that requires them to be discrete. This can be easily described with an additional vector $\kappa \in \{0,1\}^{n_g}$, in which the boolean true ($\kappa_i = 1$) indicates that the generator is discrete. This restriction can either be ensured with distinct constraints of $|\xi_i| = 1 \forall_{i \mid \kappa_i = 1}$ or through an element-wise 1-norm constraint of $\|\kappa \odot \xi\|_1 = \|\kappa\|_1$.

is this an okay marker? or is there a better approach for notation w/out doing sucky split matrix versions?

$$\begin{aligned} Z &= \{G, c, A, b, \kappa\} \\ &= \left\{c + G\xi, \forall_{\xi \in \mathbb{R}^{n_g}} \|\xi\|_\infty \leq 1 \wedge \|\kappa \odot \xi\|_1 = \|\kappa\|_1 \wedge A\xi = b\right\} \end{aligned}$$

[8] I don't want to use that notation... it's more confusing to me when doing the full zonotope-based framework... plus it gets converted into this for the optimization problem anyway...

2.1 Dimension-awareness in Zonotope-based Sets

Given the structure of the affine and generator-based construction of a zonotope within an n-dimensional space, it makes sense to label each of the dimensions of the center and generator matrices instead of maintaining the ordered collection (family?) of vector-spaces for each set. Thus, for zonotope-based set $Z = \{G_Z, c_Z, A_Z, b_Z, \kappa_Z\}$, each of the keys, ${}_d\nu_i^Z \in {}_d\mathcal{D}_Z$, will be associated with a row index i of generator matrix G_Z and center vector c_Z.

2.2 Memory-preservation within Zonotope-based sets

Under all the defined memory-perserving set-operations (as defined above/below?), the information of volume origin is solely contained within the generators and any associated origin constraints, while the relationships between sets imparted by set-operations appear as additional constraints. This allows for memory of origin to be maintained through labeling and appropriate bookkeeping of factors as the columns of the generator and constraint matrices (and κ indices) with ${}_g\nu_i^Z \in {}_g\mathcal{D}_Z$, while the memory of both origin and relationship constraints can be maintained by keeping track of the rows of the constraint matrix and vector with ${}_c\nu_i^Z \in {}_c\mathcal{D}_Z$.

2.3 Dimension-awareness and Memory-preservation within zonotope-based sets

To preserve both dimension-awareness and memory-preservation, the dictionaries associated with dimensions, factors, and constraints are maintained. Since each of the keys has a one-to-one correspondence[9], each dictionary can represented as ordered sets [10] where the index within the set[11] directly corresponds with the index of the dimensions, factors, or constraints.

Definition 6 (Dimension-aware and Memory-preserving Zonotope-based Set Definition) *Let $G \in \mathbb{V}^{n \times n_g}$, $c \in \mathbb{V}^n$, $A \in \mathbb{R}^{n_c \times n_g}$, $b \in \mathbb{R}^{n_c}$, and $\kappa \in \{0,1\}^{n_g}$.*
A zonotope-based set Z is defined as

$$\begin{aligned} Z &= \{G, c, A, b, \kappa\} \\ &= \left\{c + G\xi, \forall_{\xi \in \mathbb{R}^{n_g}} \|\xi\|_\infty \leq 1 \wedge \|\kappa \odot \xi\|_1 = \|\kappa\|_1 \wedge A\xi = b\right\} \end{aligned} \tag{2}$$

Let $\mathcal{D}_Z = \{{}_d\mathcal{D}_Z, {}_g\mathcal{D}_Z, {}_c\mathcal{D}_Z\}$ be the dictionaries for the dimensions, factors, and constraints respectively.
$(Z, \mathcal{D}_Z)$ is defined to be dimension-aware and memory-encoded.

should we do $(Z, \mathcal{D}_Z)$ or $Z = \{G, c, A, b, \kappa$

Remark 4 (Exclusion of some memory-components) *The primary element of set memory that is eliminated is the origin of components of the zonotopes center. Under a minkowski sum operation (what the zonotope is literally based upon) the center of the two zonotopes is summed in the shared dimensions. Unless the origin set is maintained in an augmented structure (or a matrix of center-vectors w/ collumns of origin sets labeled) then that specific contribution to the updated set is eliminated.*
Additionally, note that both the memory of origin set constraints and operation constraints are stored within the same constraint matrices and labeled in a shared constraint dictionary. The same is true for generators created by constraints (such as during unions) and the storing of binary generator

[9](bijective?)
[10]string/cell-array in matlab
[11](cell array)

restrictions. This is less of a problem for dimensions as all n-dimensional spaces are split into n independently-labeled dimensions. Inherently this does not eliminate the information being stored but encodes it further into the underlying structure of the set requiring more complicated dictionary management.

The following are explicit definitions for the opperations but need not be fully written out (or in seperate definitions) for each to provide all the info

Definition 7 (Demension-aware and Memory-preserving Minkowski Sum) *Let $(X, \mathcal{D}_X)$ and $(Y, \mathcal{D}_Y)$ be demension aware and memory-encoded sets. The dimension-aware and memory-preserving minkowski-sum* [12] $(X, \mathcal{D}_X) \overset{*}{\oplus} (Y, \mathcal{D}_Y)$ *results in the following:*

$$(X, \mathcal{D}_X) \overset{*}{\oplus} (Y, \mathcal{D}_Y) = \left(\begin{array}{c} \left\{ \begin{array}{c} \begin{bmatrix} G_X^{x,x} & G_X^{x,xy} & \mathbf{0} \\ \mathbf{0} & G_Y^{y,xy} & G_Y^{y,y} \\ G_X^{xy,x} & G_X^{xy,xy} + G_Y^{xy,xy} & G_Y^{y,y} \end{bmatrix}, \begin{bmatrix} c_X^x \\ c_Y^y \\ c_X^{xy} + c_Y^{xy} \end{bmatrix}, \\ \begin{bmatrix} A_X^{x,x} & A_X^{x,xy} & \mathbf{0} \\ \mathbf{0} & A_{XY}^{xy,xy} & \mathbf{0} \\ \mathbf{0} & A_Y^{y,xy} & A_Y^{y,y} \end{bmatrix}, \begin{bmatrix} b_X^x \\ b_{XY}^{xy} \\ b_Y^y \end{bmatrix}, \begin{bmatrix} \kappa_X^x \\ \kappa_{XY}^{xy} \\ \kappa_Y^y \end{bmatrix} \end{array} \right\}, \\ \mathcal{D}_{X\cap Y} = \left\{ \begin{array}{l} {}_d\mathcal{D}_{X\cap Y} = \left\{ {}_d\nu_X^x, {}_d\nu_Y^y, {}_d\nu_{XY}^{xy} \right\} \\ {}_g\mathcal{D}_{X\cap Y} = \left\{ {}_g\nu_X^x, {}_g\nu_{XY}^{xy}, {}_g\nu_Y^y \right\} \\ {}_c\mathcal{D}_{X\cap Y} = \left\{ {}_c\nu_X^x, {}_c\nu_{XY}^{xy}, {}_c\nu_Y^y \right\} \end{array} \right\} \end{array} \right) \tag{3}$$

Definition 8 (Demension-aware and Memory-preserving Intersection) *Let $(X, \mathcal{D}_X)$ and $(Y, \mathcal{D}_Y)$ be demension-aware and memory-encoded sets. The dimension-aware and memory-preserving intersection* [13] $(X, \mathcal{D}_X) \overset{*}{\cap} (Y, \mathcal{D}_Y)$ *results in the following:*

$$(X, \mathcal{D}_X) \overset{*}{\cap} (Y, \mathcal{D}_Y) = \left(\begin{array}{c} \left\{ \begin{array}{c} \begin{bmatrix} G_X^{x,x} & G_X^{x,xy} & \mathbf{0} \\ \mathbf{0} & G_Y^{y,xy} & G_Y^{y,y} \\ G_X^{xy,x} & G_X^{xy,xy} + G_Y^{xy,xy} & G_Y^{xy,y} \end{bmatrix}, \begin{bmatrix} c_X^x \\ c_Y^y \\ c_X^{xy} \end{bmatrix}, \\ \begin{bmatrix} A_X^{x,x} & A_X^{x,xy} & \mathbf{0} \\ \mathbf{0} & A_{XY}^{xy,xy} & \mathbf{0} \\ \mathbf{0} & A_Y^{y,xy} & A_Y^{y,y} \\ G_X^{xy,x} & G_X^{xy,xy} - G_Y^{xy,xy} & -G_Y^{xy,y} \end{bmatrix}, \begin{bmatrix} b_X^x \\ b_{XY}^{xy} \\ b_Y^y \\ c_Y^{xy} - c_X^{xy} \end{bmatrix}, \begin{bmatrix} \kappa_X^x \\ \kappa_{XY}^{xy} \\ \kappa_Y^y \end{bmatrix} \end{array} \right\}, \\ \mathcal{D}_{X\cap Y} = \left\{ \begin{array}{l} {}_d\mathcal{D}_{X\cap Y} = \left\{ {}_d\nu_X^x, {}_d\nu_Y^y, {}_d\nu_{XY}^{xy} \right\} \\ {}_g\mathcal{D}_{X\cap Y} = \left\{ {}_g\nu_X^x, {}_g\nu_{XY}^{xy}, {}_g\nu_Y^y \right\} \\ {}_c\mathcal{D}_{X\cap Y} = \left\{ {}_c\nu_X^x, {}_c\nu_{XY}^{xy}, {}_c\nu_Y^y, {}_c\nu_{X\cap Y}^{new} \right\} \end{array} \right\} \end{array} \right) \tag{4}$$

[12] we could likely just call this intersection and define a specific algebra instead of doing the $\overset{*}{\oplus}$ and fancy name

[13] we could likely just call this intersection and define a specific algebra instead of doing the $\overset{*}{\cap}$ and fancy name

Would a better definition of dictionaries and the convertion from general to zonotope (like origin vector spaces to individual dims labels or the origin sets generator/constrai vs set-operation generators/constra be useful? (well potentially, but is it simple or complicated to generalize?) Is this related to how tags may work/apply?

how do we best notate this... this relates to the function combine(X,Y)

how do

The union (and compliment) operations are super interesting and a memory-preserving and dim-aware implimentation from the hybrid-zonotope paper appears to be a natural extension... here's that hyb-zono union/compliment paper: `https://ieeexplore.ieee.org/document/9638970`

Definition 9 (Dimension-aware and Memory-preserving Affine Transformation) *Let $(X, \mathcal{D}_X)$ be a dimension-aware and memory-encoded zonotope-based set. For any standard affine transformation, $f(x) = MX + b$, memory will be easily preserved. Which the affine transformation is also endowed with specifications of appropriate input and output dimensions, ${}_d\mathcal{D}_{f(x)} : {}_d\nu_i^X \rightarrow {}_d\nu_j^{f(X)}$, then a memory-preservation transformation can easily occur.*

Put the definition here

Definition 10 (Dimension-aware and memory-preserving transformations) *Let $(X, \mathcal{D}_X)$ and $(Y, \mathcal{D}_Y)$ be dimension-aware and memory-encoded zonotope-based sets. For a function $g : \mathbb{V}_X \times \mathbb{V}_Y \rightarrow \mathbb{V}_{g(X,Y)}$,*

convex?... do we limit X,Y to convex too? (eliminates hyb-zono...)

Definition 11 (Zonotope-based set transformation mapping) *(definition of essentially the open-loop state-update equation... but a generic mapping)*

2.4 Application to reachability

Definition 12 (Dimension-aware and Memory-preserving Successor Set Definition) *Look at the hybrid-zonotope paper for the specifics of the reachability example... eq 5/ & 6 could become much simpler using the dimensional-awareness definition...* ***`https://arxiv.org/pdf/2304.06827.pdf`***
The memory components are what I'm currently finding tricky and how it'll propagate through the successor-set mapping... is it just like subsubsection 1.2.3 and successive constraints being added like in Example 1? Or is it completely different if we restrict the transformation into a zonotope-based dim-aware set?