

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 $\mathcal{D}_V : \nu_i \rightarrow i$ 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 $\mathcal{D}_X = \{\nu_1 \rightarrow 1, \dots, \nu_n \rightarrow n\} \subseteq \mathcal{D}_V$. 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.

Definition 2 (Dimensional Aware Relations) A dimensionally-aware relation, \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.

(i.e)

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

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 $\mathcal{D}_C : \nu_k^c \rightarrow k$ relates the shared keys to the appropriate sets in collection C .

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.

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 cartesian 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.

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

²Should 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...

³So 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...

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

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

look at notation...

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

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 $\mathcal{D}_X : \nu_i^X \rightarrow i$. The collection of sets $([X_i]_{i=1, \dots, n_x}, \mathcal{D}_X)$ is said to be memory-encoded 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 $f : V_X \rightarrow V_{f(X)}$ is considered fully memory-preserving 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, $g_k(x, y) : V_X \times V_Y \rightarrow V_{g(x, y)}$, is considered origin memory-preserving 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.)

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

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:

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

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

⁴The 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.

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

⁶The 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 columns, but I’m not sure how this is explicitly described within a generic structure outside of this being a thing

this is only pairwise... would general transformation on only one set make mores sense?

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:

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

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 until a specific projection is needed:

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

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:

“Lazy”-set Successive Set Operations

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.

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

Alternatively,

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

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.

⁷successor-set structure?

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

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.

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 $g_i \in \mathbb{V}^n$:

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

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:

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

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 $\|\kappa \odot \xi\|_1 = \|\kappa\|_1$.

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

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 ${}_c\nu_i^Z \in {}_c\mathcal{D}_Z$.

⁸I don't want to use that notation... it's too confusing to me when comparing to the full zonotope framework

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

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 correspondence⁹, each dictionary can be represented as ordered sets¹⁰ where the index within the set¹¹ 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

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

Let $\mathcal{D}_Z = \{\mathcal{D}_Z, \mathcal{D}_Z, \mathcal{D}_Z\}$ be the dictionaries for the dimensions, factors, and constraints respectively.

(Z, \mathcal{D}_Z) is defined to be dimension-aware and memory-encoded.

Remark 3 (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/ columns 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 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 operations but need not be fully written out (or in separate definitions) for each to provide all the info

Definition 7 (Dimension-aware and Memory-preserving Minkowski Sum) Let (X, \mathcal{D}_X) and (Y, \mathcal{D}_Y) be dimension aware and memory preserving sets. The dimension-aware and memory-

⁹(bijective?)

¹⁰string/cell-array in matlab

¹¹(cell array)

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

preserving minkowski-sum¹² $(X, \mathcal{D}_X) \oplus^* (Y, \mathcal{D}_Y)$ results in the following:

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

Definition 8 (Demension-aware and Memory-preserving Intersection) Let (X, \mathcal{D}_X) and (Y, \mathcal{D}_Y) be demension aware and memory preserving sets. The demension-aware and memory-preserving intersection¹³ $(X, \mathcal{D}_X) \cap^* (Y, \mathcal{D}_Y)$ results in the following:

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

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>

¹²we could likely just call this intersection and define a specific algebra instead of doing the \oplus^* and fancy name

¹³we could likely just call this intersection and define a specific algebra instead of doing the \cap^* and fancy name

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

how do we best notate this... this relates to the function merge(X,Y) also, is there a generalized intersection version? I'm not sure... outside of an affine transform going into