A Verified Algorithm for Geometric Zonotope/Hyperplane Intersection

Fabian Immler
Technische Universität München
immler@in.tum.de

Abstract

To perform rigorous numerical computations, one can use a generalization of interval arithmetic, namely affine arithmetic (AA), which works with zonotopes instead of intervals. Zonotopes are also widely used for reachability analysis of continuous or hybrid systems, where an important operation is the geometric intersection of zonotopes with hyperplanes.

We have implemented a functional algorithm to compute the zonotope/hyperplane intersection and verified it in Isabelle/HOL. The algorithm is similar to convex hull computations, our verification is therefore inspired by Knuth's axioms for an orientation predicate of points in the plane, which have been successfully used to verify convex hull algorithms. The interesting fact is that we combine a mixture of different fields: a discrete geometrical algorithm to perform operations on the continuous sets represented by zonotopes.

Categories and Subject Descriptors D.2.4 [*Software/Program Verification*]: Correctness proofs; F.3.1 [*Specifying and Verifying and Reasoning about Programs*]: Mechanical verification

Keywords Interactive Theorem Proving, Geometric Algorithms

1. Introduction

In order to perform rigorous numerical computations, sets of continuous values need to be represented with appropriate data structures. *Zonotopes* are the geometric objects that arise when computing with *affine arithmetic (AA)* [5]. They improve over interval arithmetic by their ability to track linear dependencies and have therefore become widely used, for example as numeric domain in static analysis of programs [8] or for the analysis of continuous or hybrid systems [2, 3, 6]. Then, zonotopes often occur in combination with hyperplanes, e.g. when hyperplanes are used to model the discrete jump conditions of a hybrid system. This is one situation where zonotope/hyperplane intersections need to be computed. Moreover, computing the intersection can also be an important optimization for the analysis of continuous systems. The intended application of such analyses is often a safety-critical system, which is why formal verification of the utilized algorithms is highly desirable.

CPP '15, January 12–14, 2015, Mumbai, India.

ACM 978-1-4503-3296-5/15/01…$15.00.
http://dx.doi.org/10.1145/2676724.2693164

Our main result is a functional implementation and mechanically checked verification of an algorithm to compute zonotope/hyperplane intersection, carried out in Isabelle/HOL. The intersection is performed geometrically, as described by Girard and Le Guernic [7] and is similar to convex hull algorithms. Such algorithms have been successfully verified with Knuth's [13] theory of counterclockwise systems for discrete sets of points. As Zonotopes represent continuous sets, we needed to extend Knuth's theory to continuous vector spaces. In summary, this work bridges a gap between verification of discrete geometrical algorithms and the formalization of continuous mathematics.

We build on our formalization of zonotopes in the context of AA, which has already been used to analyze ordinary differential equations [10], and therefore the continuous part of hybrid systems. With the ability to compute intersections, this work will enable us to include discrete jumps into the analysis.

1.1 Outline

We start by giving a short introduction into the mathematical background theory and its formalization in Isabelle/HOL in section 2, then explain affine arithmetic and zonotopes in section 3. We give a high-level overview of the geometric intersection algorithm in section 4. This will motivate the theory for geometric reasoning in section 5. A central part of the algorithm consists of computing the border (or hull) of two-dimensional zonotopes, the verification thereof is explained in section 6. We bring together the whole algorithm and its verification in section 7. Finally, we present the practical usability of our verified algorithm on some small examples in section 8 and discuss related work in section 9.

1.2 Notation

We work with the interactive theorem prover Isabelle, using the logic Isabelle/HOL [16]. For the sake of clearer presentation, we take the liberty to present our formalization from a relatively abstract, high-level point of view and prefer standard mathematical notation, rather than sticking to the formalization which is closer to functional programming.

We will not distinguish between sets, lists or sequences: we denote sets with uppercase letters $A, B, \ldots, Z$, as well as lists and indexed sequences, where we notate the corresponding elements in lowercase letters. For example, A can denote a set $\{a_1, \ldots, a_n\}$, a list $[a_1, \ldots, a_n]$, or a sequence of elements $i \mapsto a_i$. Despite the lax presentation, every theorem of the following has a strictly formal, mechanically checked counterpart. The theory sources can be browsed in the archive of formal proofs [11, 12].

2. Background from Analytic Geometry

We build our formalization on Isabelle/HOL's library for real and multivariate analysis, i.e., we can use real numbers $\mathbb{R}$ and vectors

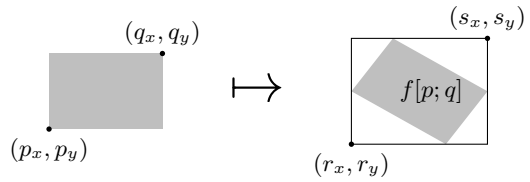

Figure 1. Wrapping effect

$\mathbb{R}^n$. We will need basic concepts from analytic geometry, which are just basic tools from linear algebra. Nevertheless we introduce them, together with the geometric meaning and with the hope to ease understanding.

We write vectors $p \in \mathbb{R}^2$ as tuples $p = (a, b)$ and use subscripts to refer to components, i.e., $p_x = a$ and $p_y = b$. We write all operations between vectors $p, q \in \mathbb{R}^n$ and real numbers $r, s \in \mathbb{R}$ explicitly: multiplication is written $r * s$ and scaling p by a factor r is written $r \cdot p$. We refrain from omitting $\cdot$ and $*$, as is sometimes customary, because we will use two subsequent symbols pq to denote line segments between the points p and q and three subsequent symbols pqr for a predicate of geometric orientation, which we introduce in section 5.

Another important operation in analytical geometry is the inner product (or dot product) of vectors $p, q \in \mathbb{R}^n$, for which we write $\langle p, q\rangle = \sum_{i=1}^{n} p_i * q_i$. In particular for $p, q \in \mathbb{R}^2$, we have $\langle p, q\rangle = p_x * q_x + p_y * q_y$. As geometric intuition, the inner product represents the cosine of the angle between two vectors multiplied with the product of their length. It is zero if the vectors are perpendicular.

Recall that (hyper-)planes can be characterized in normal vector form, i.e. $X = \{x \in \mathbb{R}^m.\ \langle n, x\rangle = c\}$ defines a plane orthogonal to the vector n and whose position in space is determined by $c \in \mathbb{R}$. Similarly, $\{x \in \mathbb{R}^m.\ \langle n, x\rangle \leq c\}$ defines a half-space.

3. Affine Arithmetic (AA)

If one aims at doing rigorous numerical computations, the first and most naive approach is to use interval arithmetic (IA). IA, however, suffers from well-known drawbacks like the dependency problem and the wrapping effect.

The dependency problem occurs when expressions with repeated occurrences of the same quantity are evaluated. As a simple example, consider $f(x) = x - x$. Evaluating $f([-1; 1])$ naively results in the gross overestimation $[-2; 2]$ of the real quantity 0.

When computing with multidimensional intervals – which are Cartesian products of intervals – the wrapping effect can occur. It can be visualized already in the two-dimensional case. Assume a rectangle $[p; q] = [p_x; q_x] \times [p_y; q_y]$ as in figure 1 and a function f rotating the rectangle. When using IA, the rotated rectangle needs to be enclosed in a Cartesian product $[r_x; s_x] \times [r_y; s_y]$ of intervals again, leading to large overapproximations of the real quantities.

With AA, Figueiredo and Stolfi [5] describe a generalization of IA in which sets are not represented by the bounds of intervals, but by affine forms. An n-dimensional affine form is a formal expression,

$$a_0 + \sum_{0<i\in\mathbb{N}} \varepsilon_i \cdot a_i$$

where $a_0 \in \mathbb{R}^n$ is called the *center* and the $a_i \in \mathbb{R}^n$ are called *generators*. The *noise symbols* ε_i are just formal variables. We assume implicitly from now on that there are only finitely many nonzero generators. In the formalization, we work with a suitable subtype of functions $\mathbb{N} \to \mathbb{R}^n$ for the generators.

The set $\gamma(a_0 + \sum_i \varepsilon_i \cdot a_i)$ represented by an affine form, or its *range*, is the set of all elements given by the form when the ε_i range over $[-1; 1]$.

$$\gamma\left(a_0 + \sum_i \varepsilon_i \cdot a_i\right) = \left\{a_0 + \sum_i \varepsilon_i \cdot a_i.\ -1 \leq \varepsilon_i \leq 1\right\}$$

Affine forms track linear dependencies, because the formal variables are treated symbolically. To return to the example demonstrating the dependency problem from before, if we have the affine form $1 \cdot \varepsilon_1$ representing the interval $[-1; 1] = \gamma(1 \cdot \varepsilon_1)$, then evaluating f using affine forms yields $f(1 \cdot \varepsilon_1) = 1 \cdot \varepsilon_1 - 1 \cdot \varepsilon_1 = 0 \cdot \varepsilon_1$, where the result represents the exact quantity $\gamma(0 \cdot \varepsilon_1) = \{0\}$. It is therefore important to consider the dependencies between formal variables when looking at several affine forms, because the joint range $\left\{\left(a_0 + \sum_i \varepsilon_i \cdot a_i, b_0 + \sum_i \varepsilon_i \cdot b_i\right).\ \forall i.\ -1 \leq \varepsilon_i \leq 1\right\}$ of two affine forms may be a proper subset of the Cartesian product of their individual ranges.

The guarantees one gets from computing with affine forms therefore always include dependencies between the involved "real" quantities and are expressed via the joint range of affine forms, what we illustrate here with the example of addition:

THEOREM 3.1. *If $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ are in the joint range, i.e.,*

$$(x, y) \in \left\{\left(a_0 + \sum_i \varepsilon_i \cdot a_i, b_0 + \sum_i \varepsilon_i \cdot b_i\right).\ -1 \leq \varepsilon_i \leq 1\right\}$$

then $x + y$ is in the range of the resulting affine form:

$$x + y \in \left\{(a_0 + b_0) + \sum_i \varepsilon_i \cdot (a_i + b_i).\ -1 \leq \varepsilon_i \leq 1\right\}$$

Other operations, like scaling, and in general every linear operation φ can be represented exactly, i.e., without losing correlations between input and output quantities, because linear operations distribute over the formal sum:

$$\varphi\left(\gamma\left(a_0 + \sum_i \varepsilon_i \cdot a_i\right)\right) = \gamma\left(\varphi(a_0) + \sum_i \varepsilon_i \cdot \varphi(a_i)\right)$$

Since rotation is a linear operation, AA does not suffer from the wrapping effect like IA does. Moreover, affine forms allow to represent high-dimensional sets much more compactly than e.g., general polytopes. We elaborate on the geometry of the sets represented by affine forms in the following.

Zonotopes Apart from looking at affine forms as a formal sum, studying the sets they represent geometrically gives some valuable insights. The represented sets are called zonotopes, and they are particular centrally symmetric, convex sets. A zonotope can be visualized as the Minkowski sum of line segments defined by the generators. The Minkowski sum $X \oplus Y$ operates on two sets and returns the set containing all possible sums between elements of the first and second set:

$$X \oplus Y = \{x + y.\ x \in X \wedge y \in Y\}$$

For a generator a_i, the corresponding line segment is $l_i = \{\varepsilon \cdot a_i.\ -1 \leq \varepsilon \leq 1\}$. Figure 2 illustrates how a zonotope is built by incrementally taking the Minkowski sum of the three line segments l_1, l_2, l_3 corresponding to generators a_1, a_2, a_3. In the two-dimensional case, we can speak of corners $(c_0, c_1, \dots)$ and edges $(c_0c_1, c_1c_2, \dots)$ of the zonotope (compare figure 3). Corners are of the form $a_0 + \sum_i \varepsilon_i \cdot a_i$ for $\varepsilon_i \in \{-1; 1\}$, edges are of the form $c_i(c_i + 2 \cdot a_j)$ for corners c_i and generators a_j. We call the set of all edges the hull of the zonotope.

4. Intersection

Computing the intersection of zonotopes with hyperplanes is an important operation and can be done geometrically. Unfortunately,

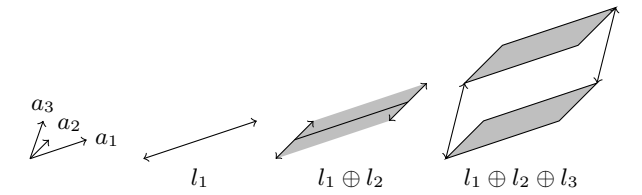

Figure 2. Construction of a zonotope

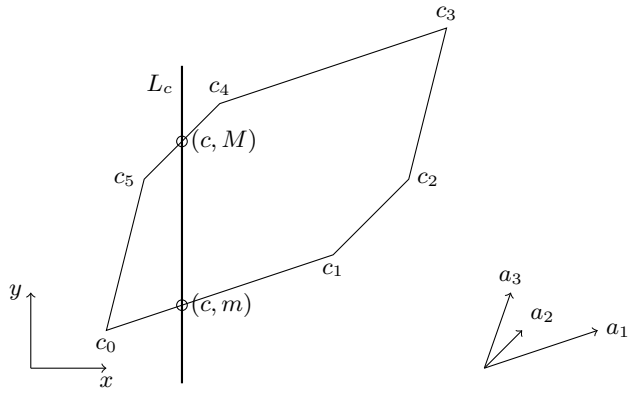

Figure 3. Corners and edges of a zonotope, intersecting line L_g

the complexity for computing the exact intersection grows exponentially with the dimension. An overapproximation of the zonotope before computing the intersection is possible but often leads to too coarse overapproximations. Therefore Girard and Le Guernic [7] proposed a way to directly compute overapproximations to the intersection by reducing it two a two-dimensional problem.

In this section we give a rough overview over the algorithm and the challenges it poses for verification.

4.1 Reduction to a Two-Dimensional Problem

The first idea is to overapproximate a given set X tightly from a set D of directions, which can be chosen arbitrarily. For every direction $d \in D \subseteq \mathbb{R}^n$, the infimum m_d and supremum M_d of the sets $\{\langle x, d\rangle.\ x \in X\}$ needs to be determined. Geometrically speaking, m_d and M_d give the position of two hyperplanes with normal vector d. The two hyperplanes bound X from below and above, respectively. An overapproximation P is then given by the points between all of these hyperplanes:

$$X \subseteq P = \{x \in \mathbb{R}^n.\ \forall d \in D.\ m_d \leq \langle x, d\rangle \leq M_d\}$$

The second observation of Girard and Le Guernic is that when the set X is the intersection of some set Z with a hyperplane $G = \{x.\ \langle x, g\rangle = c\}$, then the computation of the overapproximation P can be reduced to a two-dimensional problem with the linear transformation $\Pi_{g,d} : \mathbb{R}^n \to \mathbb{R}^2$, $\Pi_{g,d}(x) = (\langle x, g\rangle, \langle x, d\rangle)$.

THEOREM 4.1 (Two-Dimensional Reduction).

$$\{\langle x, d\rangle.\ x \in Z \cap G\} = \{y.\ (c, y) \in \Pi_{g,d}(Z)\}$$

The theorem is an easy consequence of the definitions of G and $\Pi_{g,d}$. For every direction d, the theorem allows to reduce the computation of the intersection $Z \cap G$ on the left-hand side to the intersection of the projected two-dimensional zonotope $\Pi_{g,d}(Z)$ with the vertical line $L_c = \{(x, y).\ x = c\}$.

We therefore need an algorithm *bound-intersect-2D* that computes the intersection of a two-dimensional zonotope S with a vertical line L_c, returning the minimal m and maximal M second coordinate of the intersection, as illustrated in figure 3. Correctness of the algorithm is specified as follows:

$$\textit{bound-intersect-2D}(S, L_c) = (m, M) \implies S \cap L_c \subseteq \{(x, y).\ x = c \wedge m \leq y \leq M\}$$

With this specification and theorem 4.1 in mind, we can easily define the algorithm *bound-intersect*(Z, g, c, d), which returns the lower and upper position (m, M) of the hyperplanes with normal vector d that bound the intersection $Z \cap \{x.\ \langle x, g\rangle = c\}$ of a zonotope Z with the hyperplane normal to g at c.

$$\textit{bound-intersect}(Z, g, c, d) := \textit{bound-intersect-2D}(\Pi_{g,d}(Z), L_c)$$

The idea to implement the algorithm *bound-intersect-2D* for two-dimensional zonotopes is to first compute the edges of the zonotope with an algorithm *hull-of-zonotope* S. We compute bounds on the intersection of the vertical line L_c with every edge ab as follows: for the line segment ab we assume $a_x \leq g \leq b_x$, otherwise we just flip the corners. If the segment is vertical, i.e., $a_x = b_x$, we return $m = \min(a_y, b_y)$ and $M = \max(a_y, b_y)$ as bounds on the intersection. If $a_x < b_x$, we use approximate operations with fixed precision to compute bounds on the exact point of intersection $m \leq \frac{b_y - a_y}{b_x - a_x}(c - a_x) + a_y \leq M$. The zonotope then intersects the line between the minimum and maximum bounds m, M of all edges. The only part missing is then how to compute *hull-of-zonotope*, which we sketch in the following.

4.2 Computation of Two-Dimensional Hulls

To formally reason about the computed intersection, some guarantees concerning the edges computed by *hull-of-zonotope* are required: in particular that they actually enclose the zonotope. To see how this can be ensured, we first take a look at how they are actually computed. Intuitively, assuming that all generators point upwards, one starts at the lowest corner c_0 in figure 3 and appends to it the "rightmost" generator a_1 (twice) to reach c_1. One then continues with the "rightmost" of the remaining generators, a_2 and is in the process essentially "wrapping up" the hull of the zonotope.

We therefore need a way to reason about "rightmost" vectors. Similar ideas of "wrapping up" a set of points also occur for convex hull algorithms, they have been studied extensively in the literature and we can build on a nice abstraction by Knuth [13], namely the theory of counterclockwise (ccw) systems.

The algorithm *hull-of-zonotope* is easier to implement than convex hull algorithms, because it basically only needs to sort the generators. The verification, however, poses additional challenges as we do not deal with discrete set of points, but rather with the continuous set given by the zonotope which needs to be enclosed by the computed segments.

In the following section, we will present our formalization of Knuth's ccw system and how we extended it from discrete to continuous sets.

5. Geometry

In order to verify geometric algorithms, one needs a formal notion of the geometric concepts involved. For convex hull algorithms, which are similar to *hull-of-zonotope*, Knuth [13] has given a small theory that axiomatizes the notion of orientation of points. The intuition is that for three points p, q, r in the plane, visiting them in order requires either a counterclockwise (ccw) turn (written pqr) or clockwise ($\neg pqr$) turn. Knuth observed that already few of the properties fulfilled by the ccw predicate pqr suffice to define a theory rich enough to formalize many concepts in algorithmic geometry.

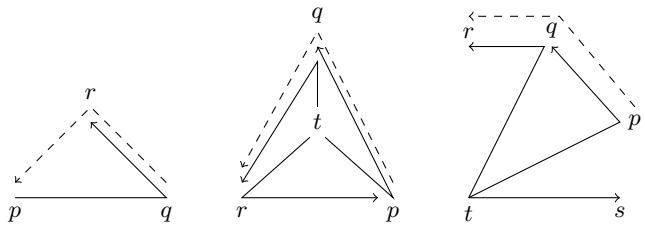

Figure 4. Cyclic symmetry (left), interiority (middle), transitivity (right); dashed predicates are implied by solid ones

We start this section by presenting Knuth's system of axioms and how they abstractly induce a total order on particular sets of points. Then we present the standard instantiation for points in the plane and which additional constraints are needed to talk about ccw systems on vector spaces instead of discrete sets.

5.1 CCW System

Knuth introduces the notion of a *ccw system* as a set of points together with a ccw predicate written pqr for points p, q, r, which satisfies the following properties, inspired by the relations satisfied by points in the plane. For all axioms in the following, there is the additional implicit assumption that the involved points are pairwise distinct. For three points, only simple axioms need to be fulfilled:

AXIOM 5.1 (Cyclic Symmetry). $pqr \implies qrp$

AXIOM 5.2 (Antisymmetry). $pqr \implies \neg prq$

AXIOM 5.3 (Nondegeneracy). $pqr \vee prq$

Cyclic symmetry and the more interesting case of interiority, which involves four points, are illustrated in figure 4. Interiority states that if one point t is left of three lines pq qr rp, then the three other points are oriented in a triangle according to pqr.

AXIOM 5.4 (Interiority). $tpq \wedge tqr \wedge trp \implies pqr$

The most important tool for reasoning is transitivity, which involves five points and works if three points p, q, r lie in the half-plane left of the line ts, i.e., $tsp \wedge tsq \wedge tsr$. Then, fixing t as first element for the ccw relation, we have transitivity in the second and third element: $tpq \wedge tqr \implies tpr$ (see figure 4).

AXIOM 5.5 (Transitivity).

$$tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$$

The same intuition also holds for the other side of the half-plane:

AXIOM 5.6 (Dual Transitivity).

$$stp \wedge stq \wedge str \wedge tpq \wedge tqr \implies tpr$$

As it is done by Knuth, we can show that axioms 5.1, 5.2, 5.3 and 5.5 together imply axiom 5.6 and vice versa. As this reasoning is carried out abstractly in a small first order theory, sledgehammer (Isabelle's interface to various automatic theorem provers) finds a proof consisting of one single invocation of an automated prover, which is a nice contrast to more than half of a page of low level reasoning in Knuth's presentation.

5.2 Total Order from CCW

As sketched in section 4.2, we need to be able to select a "rightmost" element of a set of vectors. With the transitivity relation presented before, we can obtain a total order on vectors which allows to do just this: Given a center t and another point s, the orientation predicate tpq can be used to define a total order on points p, q in the half-plane left of ts, i.e., $p < q$ iff tpq. Axioms 5.2 and 5.3 directly provide antisymmetry and totality. Transitivity of the order follows from axiom 5.5 and the assumption that all points are in the half-plane left of ts.

This ordering is then used to specify a *ccw-sorted* list of points R, with respect to a center p:

$$\textit{ccw-sorted}\ p\ R := (\forall i\ \forall j.\ i < j \implies pr_i r_j)$$

A list of points can only be sorted if all points are in one half-plane through the center, because the first element r_0 of a *ccw-sorted* list restricts all subsequent points to the half-plane left of pr_0.

5.3 Instantiation for Points in the Plane

Up to now, our reasoning was based abstractly on ccw systems, but of course we also want to use the results for a concrete ccw predicate. Well known from analytic geometry is the fact that ccw orientation is given by the sign of the following determinant $|pqr|$:

$$|pqr| := \begin{vmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{vmatrix} = \begin{matrix} p_x * q_y + p_y * r_x + q_x * r_y - \\ (r_x * q_y + r_y * p_x + q_x * p_y) \end{matrix}$$

Points are collinear iff $|pqr| = 0$. Under the assumption that one works with a finite set of points where no three points are collinear, the following predicate $pqr^>$ satisfies the axioms of a ccw system.

$$pqr^> := |pqr| > 0$$

Most axioms can easily be proved using Isabelle/HOL's rewriting for algebraic structures. Transitivity is slightly more complicated, but can also be solved automatically after a proper instantiation of Cramer's rule, which is easily proved automatically:

$$|tpr| = \frac{|tqr||stp| + |tpq||str|}{|stq|}\ \text{, if } |stq| \neq 0$$

5.4 CCW on a Vector Space

Knuth presented his axioms with a finite set of discrete points in mind, in our case we need to talk about orientation of arbitrary points in a continuous set. We therefore require consistency of the orientation predicate when vector space operations are involved.

We stick to the the predicate $pqr^>$ because we can rule out degenerate cases in pre- and postprocessing phases (sections 6.5 and 6.4). As vector space, we consider points $p, q, r \in \mathbb{R}^2$.

One obvious requirement is that orientation is invariant under translation (figure 5, left):

THEOREM 5.7 (Translation).

$$(p + s)(q + s)(r + s)^> = pqr^>$$

With translation invariance, we can reduce every ccw triple to a triple with 0 as origin, and from there it is easy to state consistency with respect to scaling: If at q, there is a ccw turn to r, then every point on the ray from 0 through q will induce a ccw turn to r (figure 5, right).

THEOREM 5.8 (Scaling).

$$\alpha > 0 \implies 0(\alpha \cdot q)r^> = 0qr^>$$

Negative scalars can be treated by requiring that reflecting one point at the origin inverts the ccw predicate:

THEOREM 5.9 (Reflection).

$$0(-p)q = 0qp$$

Furthermore, the addition of vectors q and r, which are both ccw of a line p needs to be ccw of p as well:

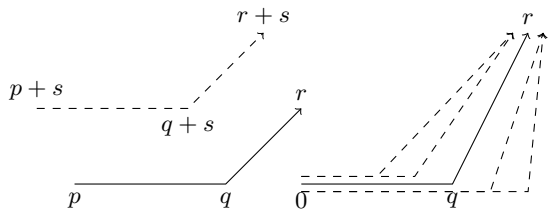

Figure 5. Invariance under translation (left), invariance under scaling (right)

THEOREM 5.10 (Addition).

$$0pq \implies 0pr \implies 0p(q+r)$$

Equipped with these theorems, we can simplify many of the ccw predicates that can occur. For example, one can get rid of all parts of the third component which are collinear with the second:

$$\gamma > 0 \implies 0a(\gamma \cdot a + b)^{>} = 0ab^{>}$$

Some ccw predicates involving a sum can be reduced to showing the ccw predicate for every summand:

$$\forall i \leq k.\, pq(r_i)^{>} \implies pq\left(\sum_{i\leq k} r_i\right)^{>}$$

5.5 The Interior of a Polygon

Looking at how $pqr^{>}$ is defined, one can see that the ccw predicate can also be used to describe half-planes: $X_{pq} = \{x.\ pqx^{>}\}$ is the half-plane left of the line pq. The interior of any given convex polygon can therefore be described as the intersection of half-planes defined by consecutive edges: for corners $c_0, \ldots, c_n$ of a polygon, we have that the interior of the polygon is the set $P = \bigcap_{i=0}^{n} X_{c_j c_{j+1}}$.

6. Verification of Two-Dimensional Hulls

Equipped with the formalisms to reason about orientation in the plane, we now detail on the algorithm *hull-of-zonotope* to compute the hull of a two-dimensional zonotope $a_0 + \sum_i \varepsilon_i \cdot a_i$ and how we verified it.

The verification of the algorithm is simpler when we assume that the generators a_i are not collinear and that all of them point upwards, i.e., $(a_i)_y > 0$. In fact, the instantiation of the ccw predicate $pqr^{>}$ requires this. We present a suitable preprocessing in section 6.5, which ensures that these conditions are always met when computing the hull. We can also assume that the zonotope is centered around the origin, i.e., $a_0 = 0$.

The aim is to compute a list of corners c_i of the zonotope generated by the a_i. We first compute the corners on the right side (c_1, c_2, c_3 in figure 2) by appending the generators in sorted order. Then we reflect the obtained corners according to the central symmetry of zonotopes. In a bit more detail, the algorithm can be described as in figure 6

Implementing this algorithm in a functional language is straightforward. The specification that we aim to verify is that the returned edges enclose the interior of the zonotope, i.e., every point x in the zonotope is left of all the edges $c_i c_{i+1}$ (as described in section 5.5). The verification can be outlined as follows: First, appending sorted vectors in order keeps certain linear combinations ccw oriented with respect to the line segments. Second, we establish that these linear combinations represent the interior of the zonotope, therefore the interior of the zonotope is ccw of the line segments after step 5. Then, because of symmetries, the same holds for the reflected

1. input: a zonotope $a_0 + \sum_i \varepsilon_i \cdot a_i$, given by the list of its generators $a_0, \ldots, a_n$, all pointing upwards
2. find the lowest corner c_0 of the zonotope, i.e., $c_0 = -\sum_i a_i$
3. sort the generators, i.e., assume $i < j \implies 0(a_i)(a_j)^{>}$
4. double the generators, i.e., $b_i := 2 \cdot a_i$
5. append generators in order, i.e., $c_{i+1} = c_i + b_{i+1}$
6. reflect the corners $c_0, \ldots, c_n$, i.e., $c_{n+i+1} = -c_{i+1}$
7. output: a list of edges $c_0c_1, c_1c_2 \ldots, c_nc_{n+1}, \ldots, c_{2n-1}c_{2n}$

Figure 6. Algorithm *hull-of-zonotope*

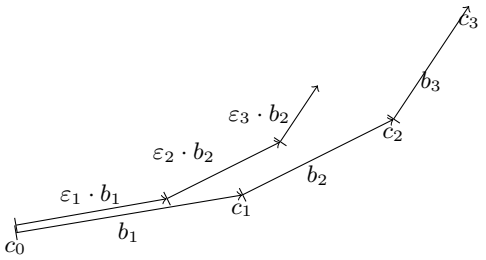

Figure 7. $c_0 + \mathit{polychain}([b_1, b_2, b_3])$ and illustration of lemma 6.1

corners on the left, i.e., the ones after step 6. Finally, we give a relaxed notion of ccw that allows to include not only the interior but also the edges of the zonotope.

6.1 Appending Sorted Vectors

We write *polychain* B for the list of points obtained by appending the vectors $b_i \in B$ in order, i.e, $(\mathit{polychain}\ B)_{i+1} = (\mathit{polychain}\ B)_i + b_{i+1}$. A crucial property is that whenever a list of vectors B is sorted, then certain linear combinations of the elements b_i of B are ccw of *polychain* B. Compare also figure 7.

LEMMA 6.1. *Assume ccw-sorted* B *and* $\forall i.\ 0 < \varepsilon_i < 1$ *and* $C = \mathit{polychain}\ B$. *Then:*

$$c_j c_{j+1} \left(\sum_i \varepsilon_i \cdot b_i\right)^{>}$$

The proof goes by induction the length of B resp. C and makes use of ccw vector space theorems like the ones given in section 5.4.

6.2 The Interior of the Zonotope

The interior of the Zonotope, i.e., the points constructed as linear combinations $a_0 + \sum_i \varepsilon_i \cdot a_i$ for $-1 < \varepsilon_i < 1$, can also be represented as linear combinations $c_0 + \sum_i \varepsilon_i \cdot b_i$ for a different set of $0 < \varepsilon_i < 1$: the linear combinations are just translated to the lowest point c_0, and doubling the vectors in step 3 makes up for the smaller range for the ε_i.

Now assume that after step 4, we have computed n corners $c_0, \ldots, c_n$. Since the c_j are sorted, we have from lemma 6.1 that all linear combinations $x = c_0 + \sum_i \varepsilon_i \cdot b_i$ are left of the line segments $c_j c_{j+1}$, but according to the previous considerations, this means that the interior of the Zonotope is left of the computed line segments after step 4:

LEMMA 6.2. *After step 4,*
$c_j c_{j+1}(\sum_i \varepsilon_i \cdot a_i)^{>}$ *holds for* $-1 < \varepsilon_i < 1$

Note that at this step it is important to consider only the strict interior (i.e., $-1 < \varepsilon_i < 1$) of the zonotope, because points on the

edges do not satisfy the (strict) ccw predictate $pqr^{>}$ and can only be reached with $-1 \leq \varepsilon \leq 1$.

6.3 Reflected Corners

Step 5 of the algorithm simply reflects the already computed corners at the origin, an operation under which the orientation predicate remains invariant. Because in addition to that, every zonotope is centrally symmetric, we can deduce from lemma 6.2 that the reflected line segments enclose the interior of the zonotope as well. We also have $c_n = -c_0$.

LEMMA 6.3. *After step 5,*
$c_j c_{j+1}(\sum_i \varepsilon_i \cdot a_i)^{>}$ *holds for* $-1 < \varepsilon_i < 1$

6.4 "Postprocessing": Continuously Relaxing CCW

In order to also include the edges into our reasoning, we define the slightly relaxed ccw predicate $pqr^{\geq}$, which holds for all points on the line through pq and for all points on the half-plane left of pq.

$$pqr^{\geq} := |pqr| \geq 0$$

For every segment $c_j c_{j+1}$, the half-plane $X_j = \{x.\ c_j c_{j+1} x^{\geq}\}$ is topologically closed. We know that all points from the interior are contained in this half-plane and show that points from the edges of the zonotope are also contained: Assume some $x = \sum_i \varepsilon_i \cdot a_i$ with $-1 \leq \varepsilon_i \leq 1$, i.e., x may be on the edges or in the interior. Then define $x_m = \sum_i (\varepsilon_i \cdot (1 - \frac{1}{m})) \cdot a_i$ for $m = 1, 2, \ldots$, we therefore have strict inequalities $-1 < \varepsilon_i \cdot (1 - \frac{1}{m}) < 1$, which imply according to lemma 6.3 that $x_m \in X_j$. Moreover, as m goes to infinity, x_m tends to x, and since X_j is closed, we can conclude $x \in X_j$.

In summary, the line segments output by *hull-of-zonotope* define a polygon (as intersection of half-planes) that encloses the zonotope:

THEOREM 6.4. *After step 5,*
$c_j c_{j+1}(\sum_i \varepsilon_i \cdot a_i)^{\geq}$ *holds for* $-1 \leq \varepsilon_i \leq 1$

Theorem 6.4 proves that *hull-of-zonotope* encloses all points of the zonotope, but not that all enclosed points actually belong to the zonotope. We have also proved that theorem, but do not make use of it, as we are only interested in an overapproximation of the intersection.

6.5 Preprocessing for Generators

Recall that at the beginning of this section, we assumed the generators a_i of the zonotope to be nonaligned and pointing upwards. Given a zonotope with arbitrary generators a'_i, it is easy to compute a new set of generators a_i that meet the above conditions and represent the same zonotope.

Consider an element $x = \sum_i \varepsilon_i \cdot a'_i$ with generators a'_i which point downwards, i.e., $(a'_i)_x < 0$. Set $a_i = -a'_i$ and also negate ε_i, then $x = \sum_i \varepsilon_i \cdot a_i$ and all generators a_i point upwards.

Concerning collinear generators, one can start with the first generator a'_1 and find the indices C of collinear generators, i.e., $|0 a'_1 a'_i| = 0$ for $1 < i \in C$. Then we can set $a_1 = a'_1 + \sum_{i \in C} a'_i$ and find an appropriate ε_1 such that $x = \sum_i \varepsilon_i \cdot a_i$. Then recurse on the list of remaining generators a_i with $i \notin C$ and finally obtain a list of generators which are pairwise not collinear, i.e., for $i \neq j$, we have $|0 a_i a_j| \neq 0$.

7. The Final Intersection Algorithm

Recall that the algorithm *bound-intersect* returns hyperplanes that bound the intersection from above and below. The final result of our verification can then be summarized by a short formal statement: The intersection of a zonotope (the range of the affine form given by generators $A = (a_i)_i$) with a hyperplane is bounded by the computed half-planes:

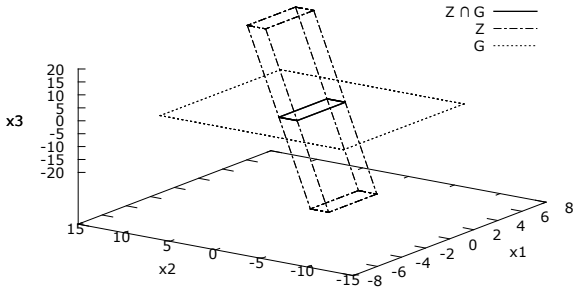

Figure 8. Intersection as it might occur in reachability analysis

$$\textit{bound-intersect}(A, g, c, d) = (m, M) \implies$$
$$\{a_0 + \sum_i \varepsilon_i \cdot a_i.\ -1 \leq a_i \leq 1\} \cap \{x.\ \langle x, g \rangle = c\} \subseteq$$
$$\{x.\ m \leq \langle x, d \rangle \leq M\}$$

The intersection is represented by half-planes, if however one wants to continue calculating with zonotopes, it is necessary, to choose several directions d_i in a way that the polytope resulting from the intersection of all the half-planes can be represented as a zonotope. For this purpose, one can choose for example hyperrectangles or parallelotopes.

8. Experiments

All of the algorithms we presented are given as functional programs in Isabelle/HOL. We can therefore make use of Isabelle's code generator [9] to obtain highly trusted code in Standard ML, which we can use to conduct some experiments and demonstrate that our formalization can actually be used to compute intersections of zonotopes with hyperplanes, we represent numbers by idealized floating point numbers $m \cdot 2^e$ for unbounded integers m, e.

We just give a short artificial example, but it is one that could occur e.g., when computing flowpipes of differential equations and intersecting them with hyperplanes, to perform a discrete jump in the hybrid system. Depending on the step size, the set might be relatively large in one direction, but small transversal to the hyperplane. A naive approach of projecting the set to the hyperplane G would result in a very large overestimation, the intersection (in this case from Basis-parallel directions) is tight, i.e. it touches the original set in every direction. The zonotope Z in the example is given by $[(0, 0, 0); (2, 1, 0)] \oplus (5, 10, 20)$. The hyperplane is given by $G = \{x.\ \langle (0, 0, 1), x \rangle = 3\}$. Computing the intersection in this case takes negligible time. For a more complex scenario, we created random ten-dimensional zonotopes with fifty generators (the zonotopes we encountered in our work on ODEs [10] are two or three dimensional and consist of around ten generators), computing the intersection with a random hyperplane takes 6-7 milliseconds on a 2.9 GHz laptop. Twice the number of generators requires twice as much time, doubling the dimension also increases the amount of time needed by a factor of two. We can therefore say that for our purposes, the code exhibits reasonable performance and scaling behavior.

9. Conclusion

We needed to extend the classical ccw systems for vector spaces. A crucial trick in the formalization was to exclude degenerate cases by looking only at the strict interior of the zonotope. Extending the reasoning from the interior to the whole zonotope could be accomplished in an elegant way using continuity. The code we extracted from our formalization exhibits reasonable performance.

9.1 Related Work

Noting that the main contribution of our work is the verification of geometric algorithm, we can compare this work with several other formalizations, especially with verifications of convex hull algorithms: They all have in common that they base their reasoning on Knuth's notion of ccw system.

Pichardie and Bertot [17] were the first to formalize Knuth's ccw system and verify a functional convex hull algorithm in Coq. Meikle and Fleuriot [15] formalized an imperative algorithm and verified it using Hoare logic in Isabelle/HOL. Brun *et al.* [4] verify an algorithm based on hypermaps to compute the convex hull.

The basic notion of a ccw system is straightforward to formalize, however it excludes in its pure form "degenerate" configurations of points, that is, three points lying on the same line. To cope with these special cases, different approaches have been used: Pichardie and Bertot give two possible solutions, one is to extend the ccw theory with an additional predicate pqr', which is true whenever q lies on the line between p and r. This requires nine additional axioms and therefore makes the abstract theory more cumbersome to use. The second approach they formalized (and which has been elaborated by Knuth) perturbs the points of the system in a continuous manner to get rid of degenerate configurations, continuity carries the results over to the degenerate case. Our approach from section 6.4 is similar in the sense that we extend results from the nondegenerate interior of a zonotope to the frontier, which contains degenerate points. Meikle and Fleuriot take a more pragmatic approach and modify their algorithms to explicitly check for collinear points.

In section 10, we digress into yet another possibility (also already described by Knuth) to refine the ccw predicate $pqr^>$ in a consistent way for arbitrary points in the plane, and see how we can follow Knuth's reasoning with our formalization of ccw systems in Isabelle/HOL. Unfortunately this approach does not directly work for ccw systems on vector spaces.

It is worth mentioning that we restrict our attention to zonotopes and not the more general approach of using support functions as done by Le Guernic and Girard [14], because many algorithms related to reachability analysis get more involved and require to solve e.g. linear programming and other optimization problems that are not formalized in Isabelle/HOL. Althoff and Krogh [1] propose a way to avoid geometric computations and directly compute an overapproximation to a zonotope/hyperplane intersection, this approach is restricted to reachability analysis of hybrid systems and cannot be applied to zonotope/hyperplane intersections in other applications.

9.2 Future Work

We work at incorporating the intersection algorithm into our formally verified tool for numerically solving ODEs [10]. Being able to perform intersections allows us to calculate e.g., Poincaré maps precisely and would also allow us to deal with discrete jumps given by hyperplanes and therefore open the door to formally verified hybrid systems reachability analysis.

10. A Consistent CCW Predicate for Degenerate Cases

This section is not directly related to the verification of zonotope/hyperplane intersection, but it is interesting as an example of formalizing geometry.

Recall that the instantiation of a ccw system with $pqr^>$ only worked with the additional assumption that all involved points are not collinear. In some cases it is possible to get rid of "degenerate" situation with some sort of preprocessing, which is what we did in section 6.5. Here we give an alternative instantiation of ccw systems for arbitrary points in the plane, as demonstrated in chapter 14 of Knuth's monograph.

There Knuth proposes to refine the ccw predicate $pqr^>$ in degenerate cases by including the lexicographic order $\prec$ on points.

$$p \prec q := p_x < q_x \vee (p_x = q_x \wedge p_y < q_y)$$

The refined ccw predicate pqr^* can then be defined as follows:

$$pqr^* := pqr^> \vee (|pqr| = 0 \wedge p \prec q \prec r \vee q \prec r \prec p \vee r \prec p \prec q)$$

And this predicate can be shown to form a ccw system for arbitrary points in the plane. As elaborated by Knuth as well, an important part of the reasoning is that the convex combination of two points lies lexicographically between them:

$$p \prec q \wedge 0 \leq \alpha \leq 1 \implies p \prec \alpha \cdot p + (1-\alpha) \cdot q \prec q$$

Knuth does not explicitly mention it, but a similar rule for triangles is needed as well.

$$p \prec q \prec r \wedge 0 \leq \alpha \wedge 0 \leq \beta \wedge 0 \leq \gamma \wedge \alpha + \beta + \gamma = 1 \implies p \prec \alpha \cdot p + \beta \cdot q + \gamma \cdot r \prec r$$

This rule is necessary to establish the following rules, all of which establish some sort of consistency between the lexicographic order and the orientation predicate. Like Knuth, we abbreviate $s \in \Delta pqr = spq^* \wedge sqr^* \wedge srp^*$ to express s lying inside the triangle given by p, q, r and $\square pqrs = pqr^* \wedge qrs^* \wedge rsp^* \wedge spq^*$ to describe an oriented tetragon p, q, r, s. Then each of the following configurations is impossible (for pairwise distinct points): first (or second), the rightmost (or leftmost) point s lies in the triangle given by the points p, q, r on the left (or right). Third, if p and q are left of r and s, then the points cannot form an oriented tetragon p, r, q, s, because two of the lines would have to cross.

$$p \prec q \prec r \prec s \wedge s \in \Delta pqr$$

$$s \prec p \prec q \prec r \wedge s \in \Delta spqr$$

$$p \prec r \wedge p \prec s \wedge q \prec r \wedge q \prec s \wedge \square prqs$$

As in the previous sections, transitivity of pqr^* can be established relatively easily algebraically, using Cramer's rule. However, when collinear points are involved, case distinctions are needed, some need to derive new collinearities, e.g., if q lies on a line with tp and on a line with tr, then r and p are on the same line: $|tpq| = |tqr| = 0 \implies |trp|$.

In addition it is necessary to rule out configurations which are according to Knuth's presentation without further elaboration just "impossible", but require careful formal proof. Consider e.g. t, s, r, p aligned and q not aligned according to $tpq^>$ and $tqr^>$. This configuration can be realized, and is only impossible by investigating subtle contradictions given by the possible lexicographic orderings of t, s, r, p in the context of the proof.

Other important lemmas that are not mentioned by Knuth are needed for translating the corner of a ccw turn, which of course only works when translating in one direction, which is why the lexicographic order is needed:

$$|trs| = 0 \wedge t \prec r \prec s \wedge trp^> \implies tsp^>$$

Acknowledgments

This work has been supported by DFG RTG 1480 (PUMA). I would like to thank Lars Hupel, Johannes Hölzl, and the anonymous reviewers for providing helpful feedback on earlier versions of this paper.

References

[1] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pages 45–54, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1220-2. .

[2] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233 – 249, 2010. ISSN 1751-570X. . {IFAC} World Congress 2008.

[3] O. Bouissou, A. Chapoutot, and A. Djoudi. Enclosing temporal evolution of dynamical systems using numerical methods. In G. Brat, N. Rungta, and A. Venet, editors, *NASA Formal Methods*, volume 7871 of *Lecture Notes in Computer Science*, pages 108–123. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38087-7. .

[4] C. Brun, J.-F. Dufourd, and N. Magaud. Designing and proving correct a convex hull algorithm with hypermaps in Coq. *Computational Geometry*, 45(8):436 – 457, 2012. ISSN 0925-7721. . Geometric Constraints and Reasoning.

[5] L. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004. ISSN 1017-1398. .

[6] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25108-8. .

[7] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control*, volume 4981 of *Lecture Notes in Computer Science*, pages 215–228. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78928-4. .

[8] E. Goubault and S. Putot. Static analysis of numerical algorithms. In K. Yi, editor, *Static Analysis*, volume 4134 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-37756-6. .

[9] F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In M. Blume, N. Kobayashi, and G. Vidal, editors, *Functional and Logic Programming*, volume 6009 of *Lecture Notes in Computer Science*, pages 103–117. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12250-7. .

[10] F. Immler. Formally verified computation of enclosures of solutions of ordinary differential equations. In J. Badger and K. Rozier, editors, *NASA Formal Methods*, volume 8430 of *Lecture Notes in Computer Science*, pages 113–127. Springer International Publishing, 2014. ISBN 978-3-319-06199-3. .

[11] F. Immler. Affine arithmetic. *Archive of Formal Proofs*, 2015. ISSN 2150-914x. `http://afp.sf.net/entries/Affine_Arithmetic.shtml`, Formal proof development.

[12] F. Immler. Affine arithmetic. *Archive of Formal Proofs*, Jan. 2015. `http://afp.sf.net/devel-entries/Affine_Arithmetic.shtml`, Formal proof development.

[13] D. Knuth. *Axioms and Hulls*. Springer, Berlin New York, 1992. Number 606 in Lecture Notes in Computer Science.

[14] C. Le Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02657-7. .

[15] L. Meikle and J. Fleuriot. Mechanical theorem proving in computational geometry. In H. Hong and D. Wang, editors, *Automated Deduction in Geometry*, volume 3763 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-31332-8. .

[16] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A proof assistant for Higher-Order Logic*. Number 2283 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43376-7.

[17] D. Pichardie and Y. Bertot. Formalizing convex hull algorithms. In R. Boulton and P. Jackson, editors, *Theorem Proving in Higher Order Logics*, volume 2152 of *Lecture Notes in Computer Science*, pages 346–361. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42525-0. .