SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Verification of Convex Hull Algorithms in Isabelle/HOL

Author

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Verification of Convex Hull Algorithms in Isabelle/HOL

Titel der Abschlussarbeit

Author:	Author
Supervisor:	Prof. Nipkow
Advisor:	Lukas Stevens
Submission Date:	Submission date

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Section

Citation test [Lam94].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}, \ac{TUM}` ⇒ Technical University of Munich (TUM), TUM

For more details, see the documentation of the `acronym` package[1].

1.1.1 Subsection

See Table 3.1, Figure 3.1, Figure 3.2, **??**.

Table 1.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

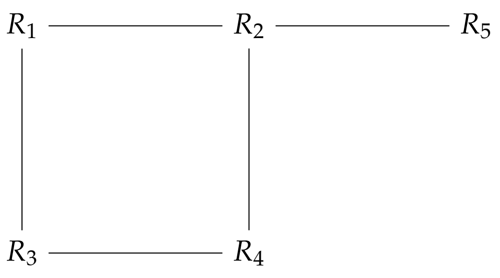

Figure 1.1: An example for a simple drawing.

!TeX root = ../main.tex

[1] `https://ctan.org/pkg/acronym`

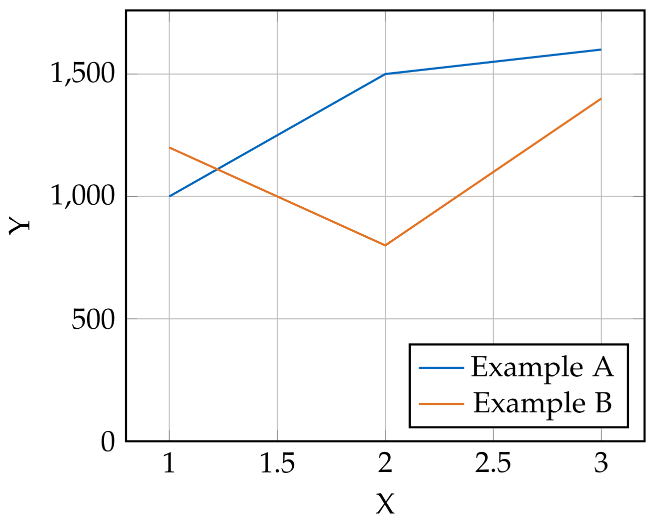

Figure 1.2: An example for a simple plot.

2 Definitions and Algorithms

2.1 Convex Hull

2.1.1 Basics

First the Convex Hull will be defined. A set $s \subseteq \mathbb{R}^2$ is convex if for every two points p and q in s it holds that all points on the line segment connecting p and q are in s again. This can be expressed, as the fact that the any convex combination of p and q has to be in s again. In Isabelle the convex predicate is defined exactly this way:

```
definition convex :: 'a real_vector set ⇒ bool where
convex s ⟷ (∀x∈s. ∀y∈s. ∀u≥0. ∀v≥s. u + v = 1 ⟶ u *R x + v *R y ∈ s)
```

The convex hull of a set s is the smallest convex set in which s is contained. There are several alternative ways in which the convex hull can be defined. One possible way is to define it as the intersection of all convex sets containing s, which is also the definition used in Isabelle/HOL. We have already seen the convex predicate, the hull predicate is defined as the intersection of all sets t that contain s and fulfill the predicate S.

```
definition hull :: (a' set ⇒ bool) ⇒ a' set ⇒ a' set where
S hull s = ⋂{t. S t ∧ s ⊆ t}
```

Consequently `convex hull s` refers to the intersection of all convex sets that contain s and therefore the convex hull of the set s. In the two dimensional case for a finite $s \subset \mathbb{R}^2$, the convex hull CH of s is a convex polygon and all the corners of this convex polygon are points from S (see figure 1). [De 00] As this thesis will focus on the two dimensional case and only give an outlook on the the three dimensional case, we will deal with computing the convex hull of $s \in \mathbb{R}^2$ in the following and therefore computing a convex polygon as representation of the convex hull of s. Assuming no three points in s are colinear, then the edges $E \subseteq s^2$ of the polygon can be described as exactly those $(p, q) \in s^2$ for which all points in s lie on the left of the vector $\vec{pq}$. Notice that the direction of the vector i.e. from p to q is relevant for expressing that a point lies on the left of the vector $\vec{pq}$. Of course the symmetric definition of E as those $(p, q) \in s^2$ for which all points in s lie on the right of the line $\vec{pq}$ works as well. The only difference is that in the set of directed edges we get, every edge now points into the opposite direction. Both definitions make sense, but because there is already infrastructure in place for first definition i.e. (p,q) is an edge if and only if all points in s are left of $\vec{pq}$, we will use this definition. But first we need to state the concept of a point q being left of the vector $\vec{pq}$ more precicsely, especially when there can be three colinear points in s.

2.1.2 Orientation

Figure x shows the convex hull of the points $s = \{p_0, p_1, p_2, p_3\}$ in the form of a convex polygon. When using the previous definition, (p_1, p_2), (p_2, p_3) and (p_1, p_3) would be edges of the convex polygon, because it holds that all points in s are left of $\vec{p_1p_2}$, left of $\vec{p_2p_3}$ and left of $\vec{p_1p_3}$. This is an unintuitive definition which should be avoided. Therefore we define the condition for (p,q) to be an edge of the convex hull polygon more precicsely. $(p,q) \in s^2$ is an edge of the convex hull polygon if and only if all points $r \in s$ are either strictly left of the vector $\vec{pq}$ (p, q and r are not colinear) or r is contained in the closed segment between p and q. The second part can be written as `r ∈ closed_segment p q` in Isabelle where `closed_segment` is defined as:

```
definition closed_segment :: 'a::real_vector ⇒ 'a ⇒ 'a set
where closed_segment a b = {(1 - u) *R a + u *R b | u::real. 0 ≤ u ∧ u ≤ 1 }
```

The fact that r lies strictly left of $\vec{pq}$ can be expressed differently by stating that (p, q , r) are making a strictly counterclockwise turn. The three points are written as a tuple as it is again necessary to state the order of p, q and r when talking about a counterclockwise turn. In the following a counterclockwise turn will always refer to a strict counterclockwise turn. Checking if a point r lies strictly left of a vector is an operation that is essential for almost all convex hull algorithms. To check if the points $((x_1, y_1)\ , (x_2, y_2), (x_3, y_3))$ make a counterclockwise turn, we can look at the sign of the determinant of the following matrix.

$$\det \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

If the determinant is positive, we know that the sequence $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ makes a counterclockwise turn, if the determinant is zero we know that the three points are colinear and if the determinant is negative, we know the the sequence $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ makes a clockwise turn. In Isabelle the function that calculates the above determinant for three points is called `det3`.

```
fun det3:: point ⇒ point ⇒ point ⇒ real where
det3 (x1, y1) (x2, y2) (x3, y3) =
x1 * y2 + y1 * x3 + x2 * y3 - y2 * x3 - y1 * x2 - x1 * y3"
```

Based on `det3` the `ccw'` predicate is defined, which expresses that three points (p,q,r) make a counterclockwise turn.

```
definition ccw' p q r ⟷ 0 < det3 p q r
```

Lastly the predicate `ccw'_seg p q r` holds if and only if r either lies counterclockwise of $\vec{pq}$ or r is contained in the closed segment between p and q.

```
definition ccw'_seg p q r = ccw' p q r ∨ r ∈ closed_segment p q
```

2.1.3 Order

In both algorithms we need to do the following operation. Given an corner p of the convex polygon, find another corner by searching for a point q such that for all other points $r \in s$ either `ccw' p q r` or `r ∈ closed_segment p q` holds. In short, we search for a q that fulfills `∀ r ∈ s. ccw'_seg p q r`. Then all other points are "left" of $\vec{pq}$ and we know q is going to be a corner of the polygon again. Intuitively it makes sense that given a finite $s \subseteq \mathbb{R}^2$ and a corner of the convex hull polygon, we can find a unique next corner. Figuratively speaking, we rotate a line that starts in p counterclockwise until we hit a point q, which is going to be the next corner. If we hit several points at the same time, we are just going to take the point further away from p. Now to translate this into a formal framework, we want to find a q that fulfills $\forall r \in s.$ `(ccw'_seg p)` $q\ r$. If `(ccw'_seg p)`

2.2 Jarvis-March Algorithm

The Jarvis March or Gift-Wrapping Algorithm is a simple output-sensitive way of calculating the convex hull of a given finite set $S \subseteq \mathbb{R}^2$ of points. It calculates the convex hull by calculating the corresponding convex polygon and returning an ordered list of the corners of the polygon. The algorithm has runtime O(n * h), where n is the number of points in S and h is the number of points that lie on the convex hull or the number of corners on the calculated polygon to be more precicse. The algorithm starts by choosing a point that is guaranteed to lie on the convex hull, for example a $p_0 = min_y min_x S$. Then the next corner of the convex polygon is found by searching a p_1 such that every point $r \in s$ lies counterclockwise of $p_0\vec{p}_1$ or is contained in the closed segment between p_0 and p_1. As explained in 3.1 we then know that (p_0, p_1) is an edge of the wanted convex polygon and we know that p_1 is once again a point on the conex hull and even a corner of the polygon as we used the rigorous definition for what an edge of the polygon is. Therefore we can repeat the previous step and search for a p_2 such that every point in S lies counterclockwise of $p_1\vec{p}_2$ or is in the closed segment between p_0 and p_1. In Isabelle terms we want a p_2 such that for all $r \in s$, it holds that `ccw' p₁ p₂ r ∨ r ∈ closed_segment p₁ p₂` . Again p_2 has to be a corner of the convex polygon and (p_1, p_2) an edge on of the polygon. The algorithm continues until a $p_h = p_0$ is found to be the next point and stops, because the first corner of the polygon is encountered again. The ordered sequence of points $p_0, p_q, ..., p_{h-1}$ are the corners of the convex polygon and $(p_0, p_1), (p_1, p_2)..., (p_{h-2}, p_{h-1}), (p_{h-1}, p_0)$ are the edges of the polygon.

The algorithm is simpler than the Graham Scan or the Chan's algorithm and has a worse runtime than both unless h is small. Graham Scan achieves a $O(nlog(n))$ runtime and Chan's algorithm a $O(nlog(h))$ runtime. If h is small Jarvis March can be faster than Graham Scan.

```
lemma turns_only_right st ⟹
turns_only_right (grahamsmarch qs st)
```

2.3 Graham Scan

2.4 Chans Algorithm

Citation test [Lam94].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}, \ac{TUM}` ⇒ TUM, TUM

For more details, see the documentation of the `acronym` package[1].

2.4.1 Subsection

See Table 3.1, Figure 3.1, Figure 3.2, **??**.

Table 2.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

!TeX root = ../main.tex

[1] `https://ctan.org/pkg/acronym`

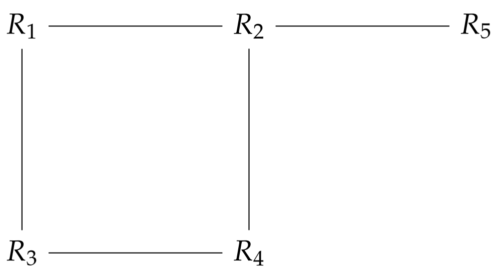

Figure 2.1: An example for a simple drawing.

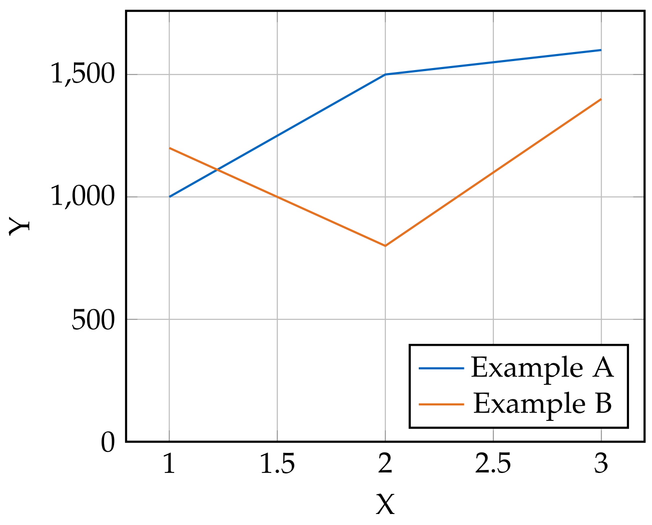

Figure 2.2: An example for a simple plot.

3 Definitions and Algorithms

3.1 Convex Hull

First the Convex Hull will be defined. A set $S \subseteq \mathbb{R}^2$ is convex if for every two points p and q in S it holds that all points on the line segment connecting p and q are in S again. This can be expressed, as the fact that the any convex combination of p and q has to be in S again, i.e. $\{x | \exists u, v \geq 0. p * u + q * v = x\} \subseteq S$ has to hold. The convex hull of a set S is the smallest convex set in which S is contained. There are several ways in which the convex hull can be defined. The convex hull CH of S is the intersection of all convex sets containing S, which is also the definition Isabelle/HOL is going to use. But the convex hull can also be defined as the set of all convex combinations of points in S, which can be proven equivalent to the previous definition. In the two dimensional case for a finite $S \subset \mathbb{R}^2$, the convex hull CH of S is a convex polygon, where the corners of this convex polygon are points from S (see figure 1). [De 00] An edge connecting two points $(p, q) \in S^2$ is an edge of the convex polygon iff. all points lie to the left of the line $\overline{pq}$ connecting p and q. Similarly the set of all edges of the convex polygon can be defined as all $(p, q) \in S^2$ for which all points in S lie to the right of $\overline{pq}$ As this thesis will focus on the two dimensional case and only give an outlook on the the three dimensional case, the examined algorithms compute a convex polygon for a given $S \subset \mathbb{R}^2$.

3.2 Jarvis-March Algorithm

The Jarvis March or Gift-Wrapping Algorithm is a simple output-sensitive way of calculating the convex hull of a given finite set $S \subseteq \mathbb{R}^2$ of points. It calculates the convex hull by calculating the corresponding convex polygon and returning an ordered list of the corners of the polygon. The algorithm has runtime O(n * h), where n is the number of points in S and h is the number of points that lie on the convex hull or the number of corners on the calculated polygon to be more precicse. First we will assume that no three points in S are colinear. The algorithm starts by choosing a point that is guaranteed to lie on the convex hull, for example a $p_0 = min_y min_x S$. Then the next corner of the convex polygon is found by searching a p_1 such that all points in S lie to the left of the line $\overline{p_0 p_1}$. As explained in 3.1 we know that (p_0, p_1) is an edge of the wanted convex polygon and we know that q is once again a point on the conex hull, i.e. a corner of the polygon. Therefore we can repeat the previous step and search for a p_2 such that all points in S lie left to the line $\overline{p_1 p_2}$. Again p_2 has to be a corner of the convex polygon and (p_1, p_2) an edge on of the polygon. The algorithm continues until a $p_h = p_0$ is found to be the next point and stops, because the first corner of the polygon is encountered again. The ordered sequence of points $p_0, p_q, ..., p_{h-1}$ are the corners of the convex polygon and $(p_0, p_1), (p_1, p_2)..., (p_{h-2}, p_{h-1}), (p_{h-1}, p_0)$ are the edges of the polygon. Now without the assumption that no three points are colinear, we require more rigorous definitions. Given a p_i that is a corner of the convex polygon the next corner p_{i+1} has to fulfill the following condition for all $q \in S$. Either q lies strictly left of $\overline{p_i p_{i+1}}$ (p_i , p_{i+1} and q are not colinear) or q is contained in the closed segment between p_i and p_{i+1}. In the following a point q lying strictly left of a line $\overline{p_i p_{i+1}}$ will be expressed as q lying counterclockwise of the line $\overline{p_i p_{i+1}}$. This clarification avoids, that points which are not a corner but still lie on the convex hull are ignored (see figure 2). The algorithm is simpler than the Graham Scan or the Chan's algorithm and has a worse runtime than both unless h is small. Graham Scan achieves a $O(nlog(n))$ runtime and Chan's algorithm a $O(nlog(h))$ runtime. If h is small Jarvis March can be faster than Graham Scan.

```
lemma turns_only_right st ⟹
turns_only_right (grahamsmarch qs st)
```

3.3 Graham Scan

3.4 Chans Algorithm

Citation test [Lam94].

Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ac{TUM}, \ac{TUM}` ⇒ TUM, TUM

For more details, see the documentation of the `acronym` package[1].

3.4.1 Subsection

See Table 3.1, Figure 3.1, Figure 3.2, **??**.

[1] `https://ctan.org/pkg/acronym`

Table 3.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

$$\begin{tikzcd}
R_1 \arrow[r, dash] \arrow[d, dash] & R_2 \arrow[r, dash] \arrow[d, dash] & R_5 \\
R_3 \arrow[r, dash] & R_4
\end{tikzcd}$$

Figure 3.1: An example for a simple drawing.

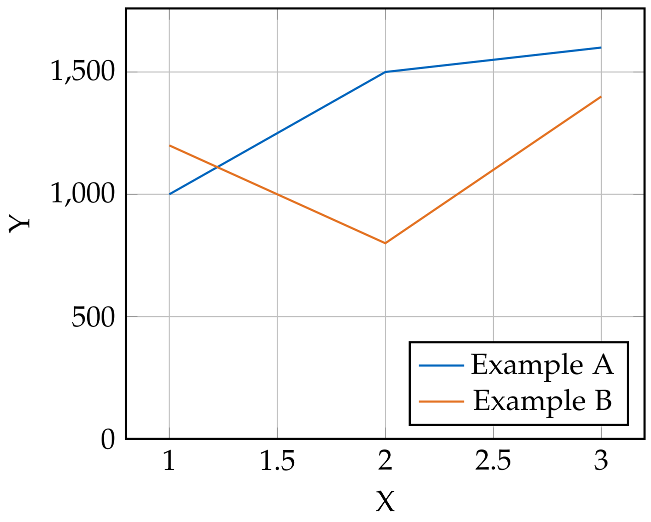

Figure 3.2: An example for a simple plot.

Abbreviations

TUM	Technical University of Munich

List of Figures

List of Tables

Bibliography

[De 00] M. De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[Lam94] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.