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

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

Figure 1.1: An example for a simple drawing.

!TeX root = ../main.tex

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

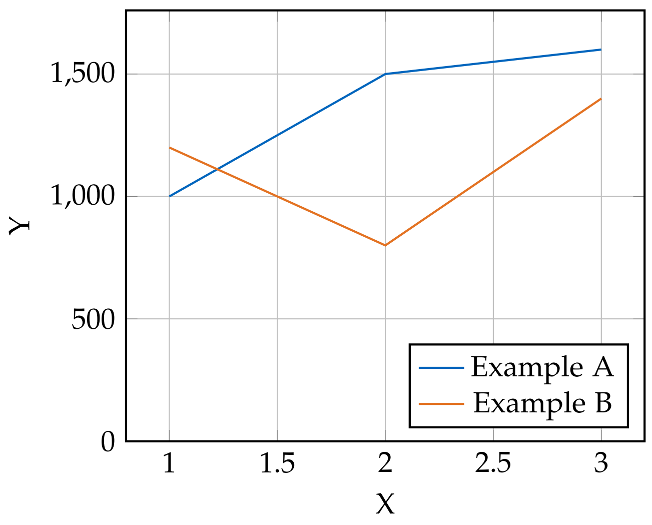

Figure 1.2: An example for a simple plot.

2 Definitions and Algorithms

2.1 Convex Hull

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 $s \subseteq \mathbb{R}^2$. 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] The edges $E \subseteq s^2$ of the polygon are exactly those $\{p, q\} \in s^2$ for which either all points in s lie left of the line $\overline{pq}$ or all points in s lie left of the line $\overline{qp}$ 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 $S \subset \mathbb{R}^2$.

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. 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)
```

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

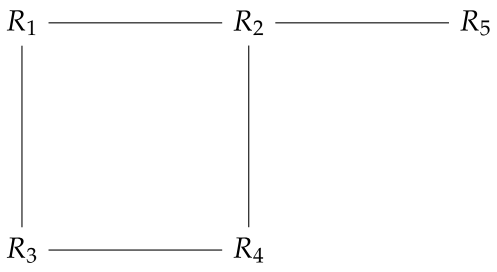

Figure 2.1: An example for a simple drawing.

!TeX root = ../main.tex

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

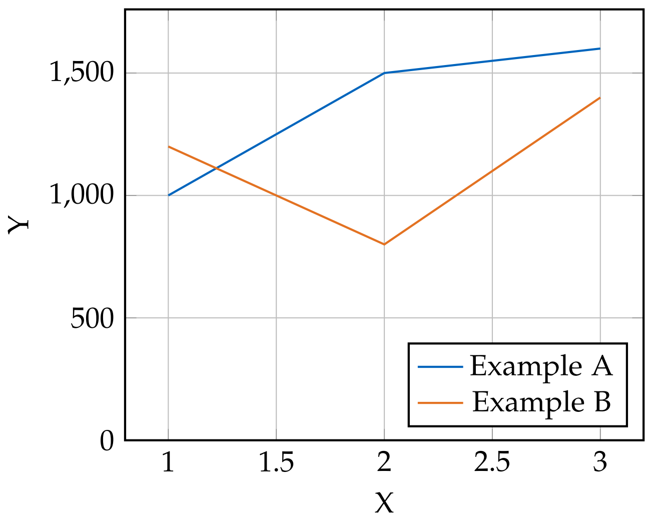

Figure 2.2: An example for a simple plot.

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.

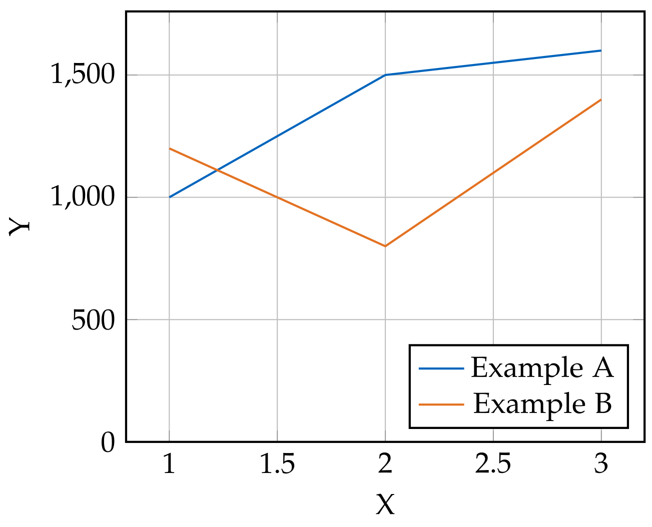

Figure 3.2: An example for a simple plot.

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.