Hoon Hong
Dongming Wang (Eds.)

LNAI 3763

Automated Deduction in Geometry

5th International Workshop, ADG 2004
Gainesville, FL, USA, September 2004
Revised Papers

Springer

Hoon Hong Dongming Wang (Eds.)

Automated Deduction in Geometry

5th International Workshop, ADG 2004
Gainesville, FL, USA, September 16-18, 2004
Revised Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Hoon Hong
North Carolina State University, Department of Mathematics
Box 8205, Raleigh, NC 27695, USA
E-mail: hong@math.ncsu.edu

Dongming Wang
Beihang University, School of Science
37 Xueyuan Road, Beijing 100083, China
E-mail: Dongming.Wang@lip6.fr

Library of Congress Control Number: 2005938552

CR Subject Classification (1998): I.2.3, I.3.5, F.4.1, I.5, G.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-31332-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-31332-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11615798 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 3763

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

Automated Deduction in Geometry (ADG) is a series of international workshops where active researchers exchange ideas and views, present research results and progress, and demonstrate software tools on the *intersection* between geometry and automated deduction. This volume contains several excellent papers (selected via peer review) based on the talks given at the ADG 2004 meeting hosted by the University of Florida, USA, during September 16–18, 2004 . The previous four meetings were held in Linz (2002), Zurich (2000), Beijing (1998), and Toulouse (1996).

This volume consists of 12 papers. The paper by *Laura I. Meikle* and *Jacques D. Fleuriot* shows how to prove the correctness of an algorithm for computing convex hulls, by using Hoare logic and Isabelle. The paper by *Judit Robu*, *Tetsuo Ida*, *Dorin Ţepeneu*, *Hidekazu Takahashi*, and *Bruno Buchberger* shows how to prove the correctness of an origami construction (heptagon), by using the Theorema system and Gröbner bases. The paper by *Xuefeng Chen*, *Peng Li*, *Long Lin*, and *Dingkang Wang* shows how to treat degenerate cases in geometric theorems rigorously, by introducing partitioned-parametric Gröbner bases. The paper by *Pavel Pech* shows how to derive formulas for the area and radius of cyclic polygons, by using Gröbner bases. The paper by *Lu Yang* and *Zhenbing Zeng* shows how to solve certain piano movers' problems, by using a specialized real quantifier elimination method (discriminant chains). The paper by *Daniel Lichtblau* shows how to compute curves bounding trigonometric planar maps, by using Gröbner bases and some numerical methods. The paper by *Francisco Botana* and *Tomás Recio* tackles several non-trivial problems (continuity, locus generation, proving, and discovering) arising in dynamic geometry, by using Gröbner bases and other symbolic ideas and methods. The paper by *Britta Denner-Broser* tackles other non-trivial problems (tracing and reachability) arising in dynamic geometry, by introducing an alternative method (to the standard purely algebraic method). The paper by *Tielin Liang* and *Dongming Wang* describes the design and a prototype for an object-oriented language suitable for (parametrically) computing, reasoning about, and visualizing geometric objects. The paper by *Dmytro Chibisov*, *Ernst W. Mayr*, and *Sergey Pankratov* shows how to solve the motion planning problem, by using real quantifier elimination and R-functions. The paper by *Hongbo Li* shows how to reconstruct an nD polyhedral scene from a single 2D line drawing, by using Grassmann–Cayley algebra and various other tools along with carefully chosen heuristics. The paper by *Gui-Fang Zhang* and *Xiao-Shan Gao* introduces planar generalized Stewart platforms and provides a complete characterization.

We, the editors, on behalf of the organizers, thank the speakers and the authors for their excellent talks and papers. On behalf of the speakers and the authors, we would like to thank Neil White, the General Chair of ADG 2004, for organizing the wonderful meeting, and Manfred Minimair, the Publicity Chair,

for making this emerging field known to wider communities. We would also like to thank the Program Committee members (listed on the next page) for lending all their time and expertise in ensuring the high quality of the talks and the papers. Due to all their tireless effort, the meeting was highly successful, fostering lively and insightful discussions, which certainly inspired the papers published in this volume. We eagerly look forward to meeting again in 2006 to share all the new exciting progress being made!

November 2005

Hoon Hong
Dongming Wang

Organization

Invited Speakers

Doron Zeilberger (Rutgers University, USA)
Ileana Streinu (Smith College, USA)

General Organization

Neil White (Gainesville, USA), Chair
Manfred Minimair (South Orange, USA), Publicity

Program Committee

Hoon Hong (Raleigh, USA), Chair
Andreas Dress (Bielefeld, Germany)
Christopher Brown (Annapolis, USA)
Deepak Kapur (Albuquerque, USA)
Dongming Wang (Beijing, China/Paris, France)
Franz Winkler (Linz, Austria)
Giuseppa Carrà Ferro (Catania, Italy)
Hongbo Li (Beijing, China)
Jacques D. Fleuriot (Edinburgh, UK)
Jürgen Richter-Gebert (Munich, Germany)
Laureano González-Vega (Santander, Spain)
Lu Yang (Chengdu, China)
Luis Fariñas del Cerro (Toulouse, France)
Meera Sitharam (Gainesville, USA)
Neil White (Gainesville, USA)
Quoc-Nam Tran (Beaumont, USA)
Rafael Sendra (Madrid, Spain)
Shang-Ching Chou (Wichita, USA)
Thierry Boy de la Tour (Grenoble, France)
Thomas Sturm (Passau, Germany)
Tomás Recio (Santander, Spain)
Volker Weispfenning (Passau, Germany)
Xiao-Shan Gao (Beijing, China)

Table of Contents

Mechanical Theorem Proving in Computational Geometry

Laura I. Meikle and Jacques D. Fleuriot

School of Informatics, University of Edinburgh, Appleton Tower, Crichton Street, Edinburgh, EH8 9LE, UK
{lauram, jdf}@dai.ed.ac.uk

Abstract. Algorithms for solving geometric problems are widely used in many scientific disciplines. Applications range from computer vision and robotics to molecular biology and astrophysics. Proving the correctness of these algorithms is vital in order to boost confidence in them. By specifying the algorithms formally in a theorem prover such as Isabelle, it is hoped that rigorous proofs showing their correctness will be obtained. This paper outlines our current framework for reasoning about geometric algorithms in Isabelle. It focuses on our case study of the convex hull problem and shows how Hoare logic can be used to prove the correctness of such algorithms.

1 Introduction

Computational geometry is the branch of computer science that studies algorithms for solving geometric problems [14]. It has applications in, among other fields, computer graphics, robotics, molecular biology, astrophysics and statistics. Verifying that these algorithms do indeed produce the correct output is important, particularly where they are used in mission-critical instances. Formal verification by computer would boost confidence in the algorithms and would also provide a valuable insight into how they work. However, little has been achieved in this field to date. Our aim is to build a framework for reasoning about geometric algorithms in the theorem prover Isabelle.

As convex hulls are used widely in computational geometry, we look at an algorithm for computing them in two dimensions, known as Graham's Scan [5]. We chose to carry out our formal verification in Hoare logic [7], as it provides a mechanisable formal system on which one can reason about imperative programs.

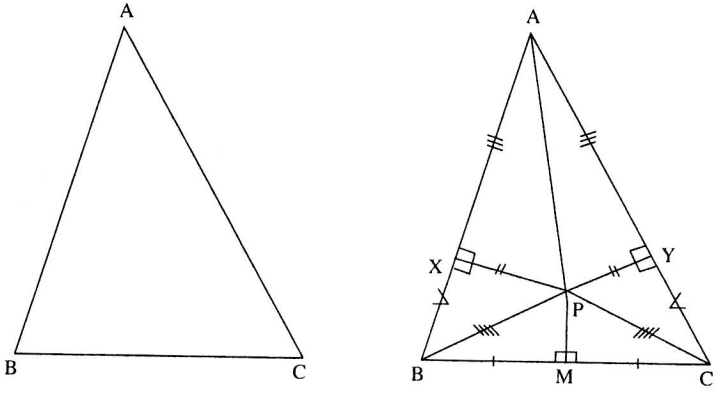
The next section outlines a few of the issues with geometric reasoning. After that we introduce the theorem prover Isabelle and describe the logic and calculus we use. Next we focus on our case study, Graham's Scan, and in particular the formal specification and verification. Finally, we conclude by discussing some related work and our goals for the future.

2 All Triangles Are Equilateral

Are all triangles equilateral? The following diagram and steps of reasoning give quite a convincing argument in favour of this statement: take any arbitrary

H. Hong and D. Wang (Eds.): ADG 2004, LNAI 3763, pp. 1–18, 2006.

triangle $\triangle ABC$ and let the perpendicular bisector of BC and the internal angle bisector of $\angle A$ meet at some point P. If we then construct points X and Y such that $PX \perp AB$ and $PY \perp AC$, then $AX = AY$ (by the angle-side-angle congruence test). Using the hypotenuse-side congruence test it can be shown that $\triangle PXB \cong \triangle PYC$, since $PX = PY$, $PB = PC$, and $\angle PXB$ is a right angle. Thus, $XB = YC$. Therefore, $AX + XB = AY + YC$. This means $AB = AC$. If we now rotate the triangle and apply the same argument on sides AC and BC, we get the conclusion that $AC = BC$. Therefore, all 3 sides of the triangle must be equal and we have proved that all triangles are equilateral.



What is the flaw in this proof? Our intuition tells us the proof is wrong, but if the result were not so obviously false we might be convinced by this spurious proof. Intuition is an important sanity check for proofs that appear logically sound.

On the flip side, if our intuition agrees with a result it may allow us to overlook missing steps or even flaws in proofs. Even Hilbert's rigorous axiomatisation of Euclidean geometry suffered from this human tendency. He argued that his proofs were free of intuition and only required the rules of logic and formal reasoning [6]. However, after formalising his work in Isabelle, we noted that Hilbert's proofs do in fact rely on intuition, through the use of diagrams and the exclusion of certain case splits [9]. We believe that if the commonly accepted "formalisation" of Euclidean geometry relies so unwittingly on intuition, then our confidence in geometric algorithms is suspect.

Diagrams in particular appeal to our intuition. They give a quasi-formal reassurance for geometric reasoning and geometric algorithms, but they can be misleading. Even if our hypotheses are tested through the use of diagrams, we still do not have complete validation of our results. In fact diagrams can be a minefield for making mistakes with undue confidence. The diagrammatic proof that all triangles are equilateral illustrates this. Although the design and verification of geometric algorithms can be aided through diagrams, clearly more rigorous reasoning is required.

A common approach to verification is simulating the algorithm on certain examples. Typically these examples are either human-selected or randomly generated. Although this method is useful, it is often subject to developer bias and can rarely verify every possible case. We believe formal verification is the surest way to have confidence in theorems, proofs, and algorithms. Furthermore, our understanding of a geometric algorithm is increased by formal verification, as it reveals inconsistencies, ambiguities and incompleteness that might otherwise go undetected.

3 The Theorem Prover Isabelle

Isabelle is a generic theorem prover, written in ML, which can be used as a specification and verification system [12]. There are a number of logics in which Isabelle allows the user to encode particular problems. Of specific interest to this work is the capacity for proofs in higher order logic (HOL). This provides a framework powerful enough to reason about algorithms and sophisticated mathematical notions. Isabelle/HOL is influenced by Gordon's HOL theorem prover [4] which itself originates from a classic paper by Church [2]. It provides an extensive library of theories and some automatic proof methods which combine simplification and classical reasoning. These tools greatly help mechanisation.

In particular, the development of Floyd-Hoare logic within Isabelle/HOL is highly relevant [10]. With this logic, the formal specifications of geometric algorithms can closely resemble their implementations.

4 Floyd-Hoare Logic

Floyd-Hoare logic is widely viewed as a way of reasoning mathematically about imperative programs [7]. The logic can not only allow verification of programs but can also aid their constructions from their specifications. Hence, it should be beneficial to reasoning about geometric algorithms in a sound and rigorous manner.

Hoare introduced a notation, called a *partial correctness specification*, for specifying what a program does: $\{P\}\ C\ \{Q\}$ is said to be true if whenever C is executed in a state satisfying P and C terminates, then the state in which C terminates satisfies Q. Total correctness is what ultimately needs to be proved when verifying a program. Informally *total correctness* = *termination* + *partial correctness*.

Although Hoare developed this logic as a means of reasoning about imperative programs, he never fully mechanised it. This was first done by Gordon [3] in the theorem prover HOL using an embedding of an annotated WHILE language in higher order logic and a verification conditions generator. In Gordon's approach, a program can be annotated to show the relationships between the variables by inserting statements called assertions. These assertions express conditions that are meant to hold at various intermediate points.

In particular, in order to formally verify programs involving loops, the partial correctness specification is annotated with mathematical statements known as invariants. An invariant R must satisfy the following conditions: it must hold initially; it must establish the result with the negated test; and the body of the program must leave it unchanged. In Gordon's system, a set of purely mathematical statements called *verification conditions* (or VCs) are then generated. A program is partially correct if the following VCs can be proven:

1. $P \rightarrow R$
2. $R \wedge$ Loop Test $\rightarrow$ body of loop preserves R
3. $R \wedge$ Negated Loop Test $\rightarrow Q$

In Isabelle, work on Hoare logic has been formalised by Nipkow [10] and will be of interest to us. In this work, the notation for representing a theorem about an imperative program with a while loop is of the general form:

```
theorem
 "|- .{ P }.
     variable assignments;;
     WHILE Loop test
     INV .{ Loop invariant }.
     DO
        program
     OD
     .{ Q }."
```

5 Geometric Preliminaries

Our mathematical framework for reasoning formally about geometric algorithms builds upon a 2D real vector theory developed in Isabelle. In this theory a real vector is a pair of real numbers, represented by (a, b). This can be interpreted in two ways; either as the point with coordinates (a, b) or as the position vector $\overrightarrow{(0,0)(a,b)}$. The vector theory defines the notions of vector addition, subtraction, dot product and scalar product. It also formalises the outer product ($\times$), defined in terms of the coordinates of the points A and B:

$$A \times B \equiv A_x * B_y \text{ - } A_y * B_x$$

Here, A_x and A_y denote the x and y-coordinates of point A respectively. Using the outer product definition, it is possible to capture the notion of the signed area of three points A, B and C:

$$\texttt{area}\ A\ B\ C \equiv (B \text{ - } A) \times (C \text{ - } A)$$

Our theory uses signed areas to identify where three points lie in relation to each other. The property of collinearity can be captured as follows:

$$\texttt{coll}\ A\ B\ C \equiv \texttt{area}\ A\ B\ C = 0$$

Another property which can be formalised is the notion of a *left turn*. If a point C lies to the left of the directed line from A to B, we write:

$$\texttt{Left\_turn}\ A\ B\ C \equiv \texttt{area}\ A\ B\ C > 0$$

As shall be seen in the following sections, testing for left turns is very useful in formalising and constructing convex hulls. Another useful property is whether a point lies between two others. We say that B lies between A and C if the following holds:

$$\begin{aligned} B\ \texttt{isBetween}\ A\ C \equiv \texttt{coll}\ A\ B\ C \ \wedge\ A \neq C \ \wedge \\ (\forall D.\ \ \texttt{area}\ A\ C\ D \neq 0 \ \longrightarrow \\ (0 < (\texttt{area}\ A\ B\ D\ /\ \texttt{area}\ A\ C\ D) < 1)\ ) \end{aligned}$$

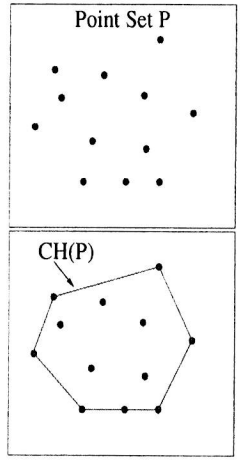
Note that we do not need to state explicitly that all the points are distinct because the *area* tests would not hold simultaneously if the points were identical.

6 Defining Convex Hulls

In this section we describe what the convex hull of a set of points is and show how we formally define it in the theorem prover Isabelle.

6.1 What Is a Convex Hull?

Many definitions exist for *convex hulls*. Intuitively, one may think of a set of points P in 2D as being nails sticking upwards from a board. Now imagine stretching a rubber band around them and letting go so its length is minimised. The region enclosed by the rubber band is known as the convex hull of P. The convex hull of P can also be described as the smallest convex set containing P. Despite the simplicity of this definition, it is not conducive to algorithm development as it is not constructive. For our formal specification we were inspired by a definition Knuth gives in his book *Axioms and Hulls* [8]. This is described in the next section.



6.2 Formal Specification of Convex Hulls

In his book, Knuth describes an orientation predicate which is equivalent to our previously defined `Left_turn`. Using this, he defines a counter-clockwise system (CC) as one which satisfies five axioms that capture the minimal properties of the orientation predicate:

1 (cyclic symmetry). $\texttt{Left\_turn}\ p\ q\ r \Longrightarrow \texttt{Left\_turn}\ q\ r\ p$

2 (antisymmetry). $\texttt{Left\_turn}\ p\ q\ r \Longrightarrow \neg\ \texttt{Left\_turn}\ p\ r\ q$

3 (nondegeneracy). $\texttt{Left\_turn}\ p\ q\ r \vee \texttt{Left\_turn}\ p\ r\ q$

4 (interiority). $\texttt{Left\_turn}\ t\ q\ r \wedge \texttt{Left\_turn}\ p\ t\ r \wedge \texttt{Left\_turn}\ p\ q\ t \Longrightarrow \texttt{Left\_turn}\ p\ q\ r$

5 (transitivity). $\texttt{Left\_turn}\ s\ t\ p \wedge \texttt{Left\_turn}\ s\ t\ q \wedge \texttt{Left\_turn}\ s\ t\ r \wedge \texttt{Left\_turn}\ s\ p\ q \wedge \texttt{Left\_turn}\ s\ q\ r \Longrightarrow \texttt{Left\_turn}\ s\ p\ r$

Although our system bears much resemblance to Knuth's, we have not adopted his axiomatic approach. We have followed Isabelle's methodology of maintaining consistency by developing new theories on top of old ones through conservative extensions only; in our case we have built upon our theory of vectors.

One drawback of Knuth's CC system for our purposes is that it disallows collinear points. This leads to many elegant results in his framework, but for real-world applications this restriction is not practical. In our formalisation, which permits collinear points, four of Knuth's axioms remain true and have been proven from first principles in Isabelle. The remaining axiom, Knuth's third, is altered and proven in our system with the obvious modification as the following theorem:

$$p{\neq}q \wedge p{\neq}r \wedge q{\neq}r \Longrightarrow \texttt{Left\_turn}\ p\ q\ r \vee \texttt{Left\_turn}\ p\ r\ q \vee \texttt{coll}\ p\ q\ r$$

Knuth presents an alternate version of Axiom 5:

5b. $\texttt{Left\_turn}\ s\ t\ p \wedge \texttt{Left\_turn}\ s\ t\ q \wedge \texttt{Left\_turn}\ s\ t\ r \wedge \texttt{Left\_turn}\ t\ p\ q \wedge \texttt{Left\_turn}\ t\ q\ r \Longrightarrow \texttt{Left\_turn}\ t\ p\ r$

Axiom 5b is illustrated in Figure 1 and has also been proven from first principles in Isabelle. Our mechanical proof of Axiom 5b follows the outline Knuth sketches using three applications of Axiom 5. However, allowing collinear points in our system dramatically increases the number of cases which needed to be considered. In fact we had 13 additional configurations to reason about.

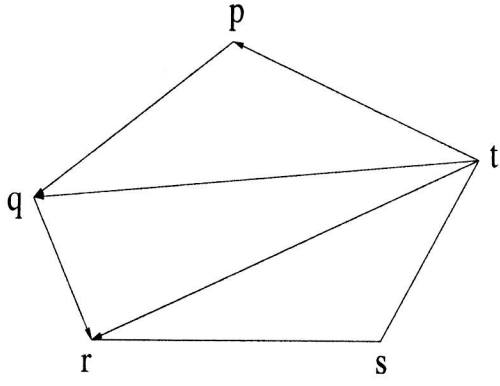


Fig. 1. Knuth's Axiom 5b

Knuth then describes the convex hull C of a set of points P, satisfying his CC system, as the set of all consecutive points ts such that `Left_turn` $t\ s\ p$ holds for all $p \notin \{s, t\}$ (see Figure 2). Clearly this definition only holds when we are traversing the hull in a counter-clockwise direction.

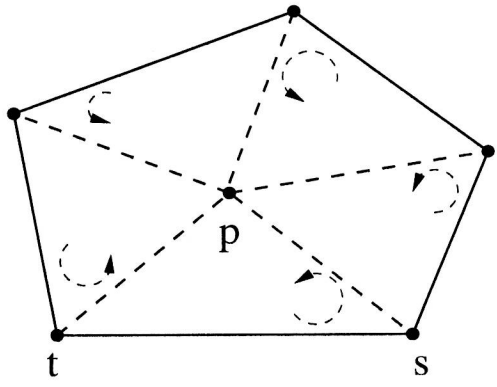


Fig. 2. Formal description of convex hull

Our formalisation of a convex hull also has to allow the possibility that any point in P could lie between two consecutive vertices. In Isabelle, we assume that the points in C are ordered clockwise and check that C is the convex hull of the points P using the infix predicate `isConvexHull`.

$$
\begin{aligned}
C\ \texttt{isConvexHull}\ P \equiv\ & \neg\texttt{all\_collinear}\ P \wedge \texttt{distinct}\ C \wedge \texttt{set}\ C \subseteq \texttt{set}\ P \wedge \\
& (\forall\ n < \texttt{length}\ P.\ \forall\ i < (\texttt{length}\ C\ -\ 1).\\
& \ (\texttt{Left\_turn}\ C_{i+1}\ C_i\ P_n \vee \\
& \ \ P_n\ \texttt{mem}\ [C_{i+1},\ C_i] \vee \\
& \ \ P_n\ \texttt{isBetween}\ C_{i+1}\ C_i) \wedge \\
& \ (\texttt{Left\_turn}\ (\texttt{hd}\ C)\ (\texttt{last}\ C)\ P_n \vee \\
& \ \ P_n\ \texttt{mem}\ [\texttt{hd}\ C,\ \texttt{last}\ C] \vee \\
& \ \ P_n\ \texttt{isBetween}\ (\texttt{hd}\ C)\ (\texttt{last}\ C)))
\end{aligned}
$$

Note that we have represented the point sets C and P using lists and the nth member of a list C is denoted by C_n. The predicate `isConvexHull` ensures that every point in C is distinct and belongs to the original, non-collinear point set P. We then check that every point in P either makes a left turn with respect to consecutive vertices in C, or is a vertex of the hull, or lies between consecutive vertices in C. The last case in the definition simply closes the convex polygon and ensures that the first and last vertices in C satisfy the same tests carried on consecutive vertices.

In the next section we describe Graham's Scan algorithm for computing convex hulls, which was chosen for formalisation in Isabelle.

7 Graham's Scan Algorithm

Computing the convex hull of a set of points is a problem which has been greatly studied. As a result there exists an abundance of algorithms. Graham's Scan is just one which computes 2D convex hulls. We chose to formally verify this algorithm as it is familiar to researchers in the field and easily understood. It is described in the following section, followed by its formalisation in Isabelle's Hoare logic.

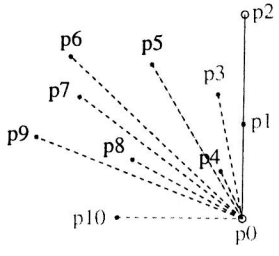
7.1 How It Works

Graham's Scan uses a method known as rotational sweep and solves the problem by maintaining a stack C of candidate points. Each point of the input set P is pushed once onto the stack, and the points that are not vertices of the convex hull are eventually popped. When the algorithm terminates, the stack C contains exactly the vertices of the hull, in counterclockwise order of their appearance on the boundary (see pseudo-code below, from [11]).

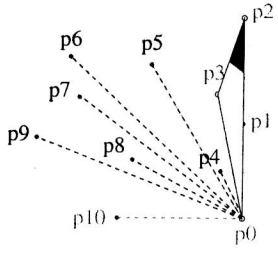
GRAHAM'S SCAN ALGORITHM

```
Find rightmost lowest point; label it P_0.
Sort all other points angularly about P_0,
     break ties in favour of closeness to P_0;
     label P_1,...,P_{n-1}.
Stack C=(P_{n-1},P_0)=(P_{t-1},P_t); t indexes top.
i = 1
while i < n do
    if P_i is strictly left of (P_{t-1},P_t)
          then Push(C,i) and set i ← i + 1
          else Pop(C)
```

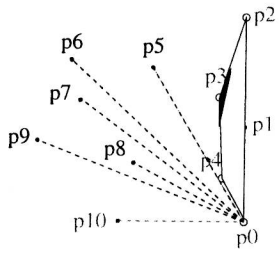
Notice that point P_{n-1} ends up twice on C, so a final pop is required in the algorithm. The following diagrams illustrate some of the behaviour of Graham's Scan. The hollow points are the vertices of the hull (shaded region) C at each stage.



Step 3: i =2



Step 4: i =3



Step 5: i =4