การใช้คอมพิวเตอร์กับปัญหาเรขาคณิตแบบยุคลิด

นายปริญญา ศิริคติธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559

USING COMPUTER FOR EUCLIDEAN GEOMETRY PROBLEMS

Mr. Parinya Sirikatitum

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Mathematics

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2016

Thesis Title	USING COMPUTER FOR EUCLIDEAN GEOMETRY PROBLEMS
By	Mr. Parinya Sirikatitum
Field of Study	Mathematics
Thesis Advisor	Associate Professor Wacharin Wichiramala, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in

Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Science

(Associate Professor Polkit Sangvani, Ph.D.)

THESIS COMMITTEE

.....Chairman

(Associate Professor Songkiat Sumetkijakan, Ph.D.)

.....Thesis Advisor

(Associate Professor Wacharin Wichiramala, Ph.D.)

.....Examiner

(Associate Professor Nataphan Kitisin, Ph.D.)

.....Examiner

(Assistant Professor Khamron Mekchay, Ph.D.)

.....External Examiner

(Sittisede Polwiang, Ph.D.)

ปริญญา ศิริคติธรรม : การใช้คอมพิวเตอร์กับปัญญาเรขาคณิตแบบยุคลิด. (USING COMPUTER FOR EUCLIDEAN GEOMETRY PROBLEMS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.วัชรินทร์ วิชิรมาลา, 32หน้า.

ในการพิจารณาปัญหาเรขาคณิตในสองมิตินั้น เราจะใช้คุณสมบัติต่างๆของรูปเรขาคณิตแต่ละรูป สมบัติเกี่ยวกับมุมและด้าน และสมบัติอื่นๆที่เกี่ยวข้อง ซึ่งเปรียบสมบัติแต่ละอย่างนั้นเสมือนสมการ จากนั้นทำการแก้สมการเพื่อหาคำตอบของความสัมพันธ์ระหว่างมุมหรือด้าน ซึ่งจะเกิดการลองผิดลองถูก อาจทำให้มองข้ามวิธีการที่ง่าย จนอาจใช้เครื่องมือที่เกินความจำเป็น ในวิทยานิพนธ์นี้ จึงมีจุดประสงค์ที่จะสร้างโปรแกรมทางคณิตศาสตร์เพื่อแก้ไขปัญหาเรขาคณิตแบบยูคลิด โดยมีแนวคิดที่จะให้คอมพิวเตอร์เปรียบเสมือนแรงงานที่จะทำงานผูกสมการเพิ่ม จนมากเพียงพอและแก้สมการทั้งหมด ทั้งนี้โปรแกรมนี้สามารถช่วยตรวจสอบความจำเป็นและเพียงพอของเครื่องมือในการแก้ไขปัญหาของโจทย์เรขาคณิตในแต่ละโจทย์ได้อีกด้วย

ภาควิชา ...คณิตศาสตร์และวิทยาการ..... ลายมือชื่อนิสิต .....

...คอมพิวเตอร์..... ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

สาขาวิชา ...คณิตศาสตร์.....

ปีการศึกษา ...2559.....

5772144623 : MAJOR MATHEMATICS

KEYWORDS : EUCLIDEAN GEOMETRY PROBLEM, COMPUTATOINAL APPROACH, PROGRAMMING ALGORITHM

PARINYA SIRIKATITUM : USING COMPUTER FOR EUCLIDEAN GEOMETERY PROBLEMS. ADVISOR : ASSOC. PROF. WACHARIN WICHIRAMALA, Ph.D. 32pp.

To solve a Euclidean geometry on the plane using only ruler and compass problem, we use properties of angles, sides and some related geometry properties which are equivalent to solving equations with multiple variables. Practically in order to solve a problem, we usually do trial and error which is a waste of time and might use unnecessary tools. By using a mathematical program, we can prevent human errors and unnecessary task on a problem. The idea of this work is to use a computer as a labour to help us solve the equations after teaching some necessary theorems to it. We create algorithms for a program to solve the problems.

Department : ...Mathematics and..... Student's Signature : .....

...Computer Science... Advisor's Signature : .....

Field of Study : .....Mathematics.....

Academic Year : .....2016.....

ACKNOWLEDGEMENTS

In the completion of my Master Thesis, I am deeply indebted to my thesis advisor, Associate Professor Dr.Wacharin Wichiramala, not only for coaching my research but also for broadening my academic vision. I would like to express my special thanks to my thesis committee: Associate Professor Dr.Songkiat Sumetkijakan, Associate Professor Dr.Nataphan Kitisin, Assistant Professor Dr.Khamron Mekchay and Dr.Sittisede Polwiang. Their suggestions and comments are my sincere appreciation. Moreover, I feel very thankful to all of my teachers who have taught me for my knowledge and skills. Also, I wish to express my thankfulness to my family and my friends for their encouragement throughout my study.

Finally, I would like to thank Chulalongkorn University graduate scholarship to commemorate the 72^{nd} Anniversary of his Majesty King Bhumibol Adulyadej for financial support throughout my graduate study.

CONTENTS

CHAPTER I

INTRODUCTION

In this chapter, the background and signification and the scope of the research will be presented.

1.1 Background and Signification

Consider the following problem. Let ABC be a triangle and let D be a point on BC. Prove that

$$\angle ABD + \angle DAB = \angle ADC. \tag{1.1.1}$$

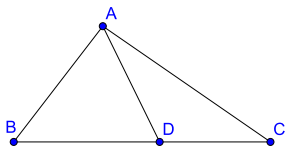


Figure 1.1: The triangle ABC

To solve this problem, we use the fact that the sum of three angles of any triangle is equal to π radian or $(180°)$. Then we have

$$\angle ABD + \angle BDA + \angle DAB = \pi. \tag{1.1.2}$$

And from the fact that

$$\angle BDA + \angle ADC = \pi. \tag{1.1.3}$$

We combine $(1.1.2)$ and $(1.1.3)$ to obtain $(1.1.1)$.

From the previous problem, we can see that solving this euclidean geometry problem we use properties of angles, sides and some related geometry properties which are equivalent to solving equations with multiple variables. In some problems, the solving part has lots of routine task and easy to make mistake. The part of solving equations with multiple variables sparked this work. The process can be done by computers if we teach them to solve the problem. This idea can also prevent the human error that usually occur in most of the problems. There are attempts to do this trial and error work by computers [1, 2, 3]. One explicit software may be found at MathWay [4]. But its performance is still very limited as users may input only a small information of objects and may ask only some specific kinds of questions. In this work, we construct an algorithm to help solving most of Euclidean geometry problems using a computational approach.

1.2 Scope of the Research

1.2.1 Algorithm

We spot that the process of reaching the solution of the Euclidean geometry problems is obtained by solving the equations with multiple variables. At present, the computers can manage to solve this problem in fewer time. So, in this research we create an algorithm to help solving Euclidean geometry problems using a computer software. That means the software for the algorithm may depends on the potential that suit for the algorithm.

1.2.2 The limitation of a computer

In the equations generating, it is impossible to generate the equations without stop since the memory of the computer is limited. For example, if we construct an algorithm that add a middle point between any two points, the added point can be act as another point to repeats on adding another middle point. This example leads to a big problem that we cannot afford to add more points into the system. For some problems, it is very helpful to add some specific points, as the proof would be

much shorter. Some problems cannot be proved without adding some crucial points. But it is a big misunderstanding to think that adding these useful points is simple. Without any experience, one may not know intuitively where to locate such points. Hence we may not expect a practical step to add points to the initial situation. Then, no point will be generated by this algorithm.

CHAPTER II

GENERALIZE THE PROBLEMS

In this chapter, we start to generalize the problems into steps and create an efficiently way to input the information of the Euclidean geometry problems into the computer. The structure of the input will be presented in this chapter.

2.1 Basic Step

A computer can solve multiple variables problems. For example, if we have initial conditions $a + b = c$ and $d = c - b$, the computer can conclude that whether $a = d$ or not. So, we can use this ability to help us solve the Euclidean geometry problems by considering the geometric elements as variables.

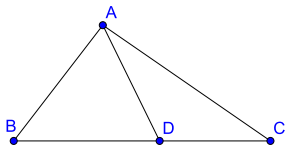


Figure 2.1: The triangle ABC

In a situation in Figure 2.1, a triangle ABC with a line AD where D is on the side BC. To conclude that $\angle ABD + \angle DAB = \angle ADC$, we can assign variables for each angle as in Figure 2.2 and then input the conditions $a + b + c = \pi$ and $c + e = \pi$. Now the computer can easily conclude that $a + b = e$ which is the desired equation. In order to use this process to solve the problem using computer, the user needs to manually input all related equations and variables which

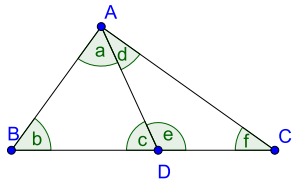


Figure 2.2: The triangle ABC

takes lots of task to do. In reality, a situation frequently has lots of variables and contains tons of equations which is hard to input manually. Therefore, not only we let the computer helps us solve the equations but also needs to help us generate the variables and equations.

Definition 2.1.1. A ***set-up*** is the information about lines, points and their relation. In particular, the order of points must be given and the orientation of each triangle is known.

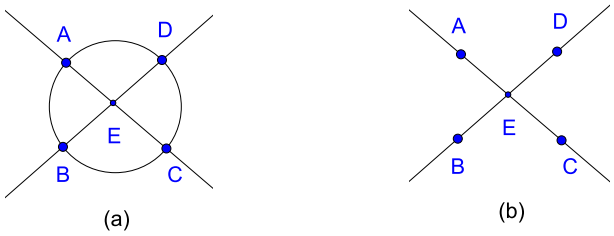


Figure 2.3: Two set-ups

For example, two set-ups in Figure 2.3, the set-up (a) has more information than (b) since the points that lie on the same circle were given.

The first step is to figure out a way to input the set-up into the computer. As mentioned, one of the way is to manually input all variables and equations but we need an efficiently way to input the

set-up using the least amount of task. The goal is that the computer can generate the variables and equations from the input. Then, we need to know what the input looks like and the computer needs to understand the input for the further equation generating. So, we begin to construct the structure of the input.

2.2 Structure

There are many ways to assign the variables for the geometric elements in a set-up. For example, in Figure 2.2 we defined the variables of angles in the set-up as $a, b, ..., f$. This assignment is easy to do but hard for the programming to identify the angle, for which variable is. It might be more useful if we assign the variables related to the name of the angles. The name of angles usually come from the points in the set-up. So, there might be some way to use the points in a set-up to generate the variables of all angles.

Furthermore the variables needs to contain the properties of the relations from the set-up. For example, in Figure 2.2, a triangle ABC, both triangles ABD and ADC shared the same side BC. So, the variable for the side BC in the triangle ABD and the variable for the side BC in the triangle ADC must be the same.

So, the input that we need must have the information of how the points and lines are positioning. It is observed that two crossed lines form angles. If the input is based on the lines in a set-up, any angles might be reached by the search since each angle formed by two lines. So, we aim to use the structure of the input using the lines. Since each line has points in it, we may use the order of points of the line to describe it. The input that describes the set-up might be just the order of the points of each line. So, we need to know that it is enough to describe the set-up by the order of points of each line.

Taking care of placement of lines and points is the most difficult laborious part in describing all detail of the set-up of a problem. For each problem, we must first start from drawing all possible

set-up described by the problem. For example, considering the following problem, called the power of point. For a given circle O, a line L passing through O at A at B, and a line K passing through O at C and D, suppose L and K meet at E. Then, $EA \cdot EB = EC \cdot ED$. For this problem, we have 2 different possible set ups as in Figure 2.4.

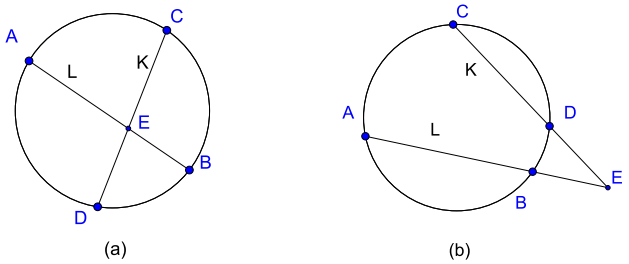


Figure 2.4: The 2 situations for the power point problem

We will find later that when A and B are swapped or C and D are swapped, the process and the outcome are the same. Then for each set up, we have to describe all necessary information about lines and points and relations between them.

2.3 Describing lines

In this section, we mention how to describe lines and their relations. We find a convenient way to describe relations between lines, including angles and triangles and their orientations.

The first important information about lines are just points and their order on each line. For examples, in Figure 2.4 (a) the lines are AEB and CED. While in the situation in Figure 2.4 (b), the lines are ABE and CDE. For any problem, when these lines are "well interlaced", we may have only 2 ways to draw them regardless rotation. Consider the following situation where the lines are not well interlaced ABC, DEF and BGE. Even though they are connected, we may draw 4 possible situations.

From the previous example, if we consider the points A, B and G, the triangle formed by

(a) (b) (c) (d)

Figure 2.5: The 4 possible situations

these points has two different orientations AGB in (a) and ABG in (b) reading counterclockwise. There are 6 ways to arrange 3 letters and 2 ways for arranging 3 letters in a circle. The different of the orientations of a triangle can be considered as arranging 3 letters in a circle. So, any three points can form only two different triangles regardless orientations. In some set-up, we can change the orientations of some triangles if the set-up does not has enough information of how lines are intersected. For example, in Figure 2.5, we can flip the line ABC to make changes of the orientation of a triangle ABG while the orientation of the triangle GDE does not change. In some set-up, we cannot change the orientations of some triangles since the lines are well interlaced and if we change any orientation of a triangle, the orientation of every triangle will be changed. We then define the concept on how lines are interlaced.

Definition 2.3.1. Any 2 triangles are called ***deeply adjacent*** if they share one common side and there is a line through one of other remaining side for each triangle.

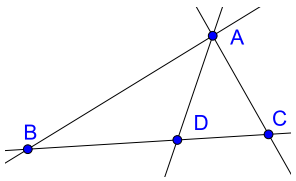


Figure 2.6: The two deeply adjacent triangles ABD and ADC

For example, in Figure 2.6, the triangles ABD and ADC are deeply adjacent since AD is the

common side and the line BDC passing through one of other remaining side. There are two pairs of triangles ABD, ABC and ADC and ABC are also deeply adjacent. Obviously, any triangle is deeply adjacent to itself.

Furthermore, we can find the orientation of a triangle that is deeply adjacent to a known orientation triangle. For example, given two triangles ABC and ADC that are deeply adjacent and the line BDC passing through one side of both triangles. Suppose we know that the orientation of the triangle ABC is ABC. Since the point D lies between B and C and both triangle shared the same side AC, the character D follows A in the orientation ADC. Similarly, if the line is BCD the orientation of the triangle ADC will be ACD

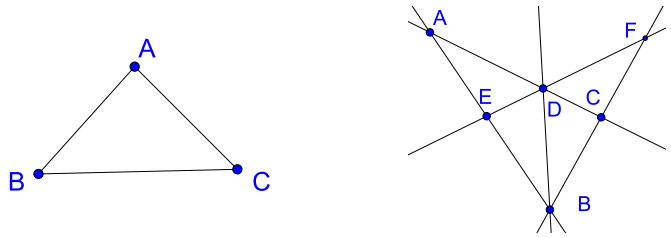
Definition 2.3.2. We say that two triangles t and t' are ***deeply connected*** if there are triangles $t = t_1, t_2, t_3, ..., t_n = t'$ where t_i and t_{i+1} are deeply adjacent.

Lemma 2.3.3. *Given a set-up where 2 triangles ABC and XYZ are deeply connected. If we know the orientation of one triangle then we may find the orientation of another triangle.*

Proof. Suppose that ABC is the triangle that we know its orientation. Since the triangles ABC and XYZ are deeply connected, there are triangles $ABC = t_1, t_2, t_3, ..., t_n = XYZ$ where t_i and t_{i+1} are deeply adjacent. Inductively, we may find the orientation of XYZ. To show that this process is independent of choice of path t_i, we suppose the orientation that we previously obtained is XYZ and suppose there are another triangles $ABC = s_1, s_2, s_3, ..., s_k = XYZ$ where s_i and s_{i+1} are deeply adjacent and the orientation of s_k is XZY. Since $XYZ = t_n, t_{n-1}, ..., t_1 = ABC$ where t_i and t_{i-1} are deeply adjacent. By reversing the inductive process on $t_1, t_2, t_3, ..., t_n$, we obtain that the orientation of ABC is ACB. This contradicts to the original orientation of ABC. □

Definition 2.3.4. We say that a set-up is ***well interlaced*** if every two triangles

in the set-up are deeply connected.



(a) (b)

Figure 2.7: The well interlaced set-ups

For example, the lines in Figure 2.7 (a) and Figure 2.7 (b) are examples of well interlaced set-up while the lines in Figure 2.8 is not, since the triangles ACB and BEG are not deeply adjacent.

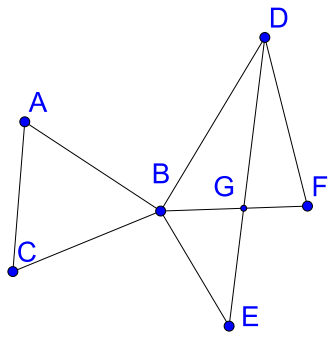


Figure 2.8: The non well interlaced set-up

If we know the orientation of a triangle t, we can find the orientation of the triangle that deeply adjacent to t. Further more, we can find the orientation of any triangles that deeply connected to t by the following theorem.

Theorem 2.3.5. *For any well interlaced set-up, if we know an orientation of a triangle in the set-up, we may find the orientations of all triangles.*

Proof. The proof is followed by lemma 2.3.3. □

In the next chapter, we will use Theorem 2.3.5 as follows. As we wish to use computer to help solving Euclidean geometry problems. For each set up, we have to describe the orientation of every triangles. The theorem will help us by reducing amount of work to describe all orientations in a well interlaced set-up as we need only one orientation of any triangle.

CHAPTER III

EQUATION GENERATING

This chapter shows the algorithm in steps. The code that input in the computers can be written in various ways depending on the language and program using.

The main idea of solving a Euclidean geometry problem by using the algorithm is to command the computer to generate multiple variable equations and let the computer solve the desired expression from those equations. The elements in a set-up such as points, lines and angles will be considered as variables in the equations. The variables will be created in the form of set with order preserved. The computer will be commanded to generate equations from the set-up using some Euclidean geometric properties and keep those equations in the set of conditions $\mathcal{C}$. The computer will use the set $\mathcal{C}$ to conclude an expression.

To begin the method, let $\mathcal{S}$ be a well interlaced set-up and then we command the computer to perform the following steps on $\mathcal{S}$.

Step 1. Input the information of each line and initial conditions

In this step, the order of points in each lines of $\mathcal{S}$ will be required to manually input by the user. This is a very simple and practical way to input a set-up without requiring any advance geometry skills. The data required in this step is the orders of points in each lines. For example, in a line ABC, we can input the points in order $[A, B, C]$ or $[C, B, A]$. The inputs in this step will be consider as variables with information of order.

The initial conditions are input to the set $\mathcal{C}$. If there is no initial conditions the set $\mathcal{C}$ will remains empty. Since in this step does not generate any equation, the $\mathcal{C}$ remains unchanged.

Step 2. Input the information of points that lie on the same circle

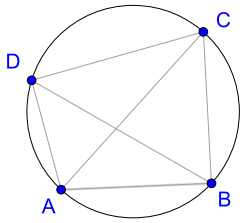


Figure 3.1: Points $A,\ B,\ C$ and D lie on the same circle

This step is the generating step for the information of points that lie on the same circle. The equations that related to circles will be generated in this step. The user needs to input the information in $\mathcal{S}$ to let the computer knows which points lie on the same circle.

Like Step 1, we require the user to manually input the points that lie on the same circle where those points must be ordered in counterclockwise.

The information leads to equations as illustrator by Figure 3.1. Where we get $\angle BCA = \angle BDA$, $\angle DAC = \angle DBC$, $\angle CDB = \angle CAB$ and $\angle ABD = \angle ACD$. Further more, we obtain $\angle ABC + \angle CDA = \pi$ and $\angle BCD + \angle DAB = \pi$. For more than 4 points on the same circle, the equations will be generated in similar way.

Thus for any collection of points that lie on the same circle, we can generate the previous equations for $\mathcal{C}$ and if the center of each circle has given, we can generate the equality of the distance of each radius. In addition, we obtain some of exist angles from the order of points that lie in the same circle which can be used in the next step.

Step 3. Generates all angles

In this step, the variables for all angles will be generated. The variables needed to keep some information like their angle's orientation. So, each variable needs to has relation to its orientation information for the further check. Some program can use a set or function as variable to keep some information when checking on its condition. For example, some program can use a function $f[A, B, C]$ for the variable of an angle ABC.

Every three points form a triangle. The angles will be considered as variables with combination of three points. The name of variables for angles can be named in any different way. In this case, we use the variable $a[A, B, C]$ for the angle ABC with the vertex B read counterclockwise. In order to generate all angles, we need to know the orientation of all triangles. The question is that how can we obtain the orientation of any angle in the set-up from the information in Step 1. From the Theorem 2.3.5, if the orientation of an angle in $\mathcal{S}$ is known, the orientations of every angle is known for the well interlaced set-up $\mathcal{S}$. So, there is a search to find all orientations of the angles if the orientation of an angle is known.

From Step 2, an orientation of some angles can be obtained from the order of circles. So, Theorem 2.3.5 can be applied if there is a group of points that lies on the same circle. If there is no group of points that lies on the same circle, an orientation of an angle in the set-up will be required to be input. Now the computer can use the transitivity of orientation in $\mathcal{S}$ to generate all angles form a known orientation of an angle.

Step 4. Triangles

In this step, the equation of the sum of measures of three angles of each triangle will be generated. The concept behind this step is to generate the relations between three angles in each triangle.

From Step 2, variables of all angles are generated. So, the computer is commanded to generate the equations of the sum of the three angles of any triangle is equal to π radian. The generated equations are kept in the set $\mathcal{C}$.

From now on, it can be observed that the properties the set-up which used angles can be input in the form of equations using the variables of the angles.

Step 5. Combining sides and angles

In this step, the equations of the sum of adjacent angles and the sum of adjacent sides will be generated. This is a trivial step but essential. The computer never know which two sides or angles

can be combined into one another. So this step is needed to measure the combining of any two adjacent angles and sides.

For points A, B and C on the same line with such order, the computer commanded to generate the measure of the distance equations $AB + BC = AC$ and for any tree orientation known angles $\angle ABD$, $\angle DBC$ and $\angle ABC$ the computer commanded to generate $\angle ABD + \angle DBC = \angle ABC$.

Step 6. Law of Sine and Cosine

In this step, the equations of law of sine and cosine of each triangle will be generated and kept in $\mathcal{C}$. The law of sine and cosine is the meaning of measuring the area of the triangles in the set-up. The expression that needs the area of triangles to reach the goal can now be solved. This is a big leap to measure the area instead of measuring just the distance of the lines or the size of the angles.

For any triangle ABC with a, b and c the corresponding sides, the computer commanded to generate the equations $\dfrac{\sin A}{a} = \dfrac{\sin B}{b} = \dfrac{\sin C}{c}$, $a^2 = b^2 + c^2 - 2bc\cos A$, $b^2 = a^2 + c^2 - 2ac\cos B$ and $c^2 = a^2 + b^2 - 2ab\cos C$.

Step 7. Similar triangles

In this step, the equations of the similar triangles will be generated. If 2 of 3 of the corresponding angles of two triangles have the same sizes then generate the equations of the proportion lengths of corresponding sides. This will generate the relations of the sides between any two triangles that are similar.

This step has the checking part before the generating. This means, the computer needs to solve the equations from $\mathcal{C}$ that which two triangles have 2 of 3 of the corresponding angles the same sizes before generate the proportion lengths. Moreover, if an equation of a pair of triangles is generated, the generated equation can be affect to the similarity of another pair of the triangles. So, the checking

Step 8. Congruent triangles

In this step, the equation of equality of the two triangles that are congruent will be generated. If the conditions of the congruent triangles SSS, SAS, AAS of ASA are satisfied on any 2 triangles, then the computer commanded to generate the equalities of the corresponding angles and sides. This explained the computer to understand how the two triangle are equal using the information of 3 elements in a triangle.

From information in Step 2 and the initial information, together with Step 1 and Theorem 2.3.5, we may know all orientations of triangles and thus angles if the lines are well interlace. Hence any mathematical assisting software may use these equations and inequalities in $\mathcal{C}$ to verify a given statement.

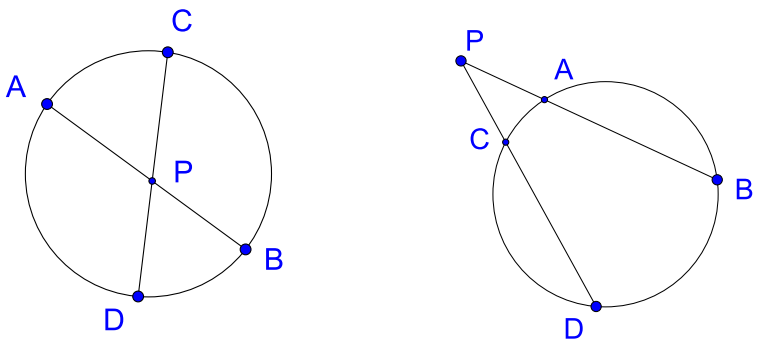
CHAPTER IV

EXAMPLE

This chapter shows the examples of the algorithm on some problem. Note that, the variables for the angles in this chapter will be in the form $a[A, B, C]$ where $\angle ABC$ is an angle with the vertex B read counterclockwise and the variables for the distance of the lines is $d[P, Q]$ where P and Q are points and $d[P, Q] = d[Q, P]$.

4.1 Power of a point theorem

Given a point P and a circle, pass two lines through P that intersect the circle in points A and B and, respectively, C and D as in Figure 4.1. Then $AP \cdot BP = CP \cdot DP$.



(a) (b)

Figure 4.1: The 2 situations for power a point theorem

This problem can be drawn into two set-up. So, in order to use the program, we have to split into two cases.

Case (a) There are two lines APB and CPD and the points on the same circle are A, D, B and C. In Step 1, the lines APB and CPD will be input. In Step 2, the points on the same circle A, D, B and C will be input with respected order. Then the equations

(i) $a[B, A, D] = a[B, C, D]$

(ii) $a[C, A, B] = a[C, D, B]$

(iii) $a[A, D, C] = a[A, B, C]$

(iv) $a[D, C, A] = a[D, B, A]$

(v) $a[A, D, B] + a[B, C, A] = \pi$

(vi) $a[C, A, D] + a[D, B, C] = \pi$

are generated. The variables of the angles $\angle BAD$, $\angle BCD$, $\angle CAB$, $\angle CDB$, $\angle ADC$, $\angle ABC$, $\angle DCA$, $\angle DBA$, $\angle ADB$, $\angle BCA$, $\angle CAD$ and $\angle DBC$ are also generated in Step 2. In Step 3, an angle from Step 2 will be chosen to determine the angles' orientation and then generate the variables of the angles. The next part is an example of the search for the angles' orientation which can be done by various ways.

In order to generate all angles, we first choose one of the angle we've already know its orientation, in this case we choose $\angle BAD$. Since any 3 points form a triangle and also form three angles, then we can get the orientation of all angles if we can find the orientation of all triangles. In this set-up, we have 8 triangles to reach. The search starts from considering the two lines that the angle $\angle BAD$ uses which is APB and AD. So the the next triangle that we can get its orientation is PAD, since the point P is lies between A and B. Now we know the orientation of angles $\angle PAD$, $\angle ADP$ and $\angle DPA$. Next, consider the angle $\angle ADP$ and the line DPC. Since the point C is lies next to D and P, the triangle DCA is reached. Follow by the angle $\angle DCA$ and the line DPC, the triangle PCA is reached. Continue the process to reach the triangles BCA, BCP, BCD and

BPD. Now we can generate the variables of all angles using the orientations of the triangles. The equalities of the angles that use the same two lines such as $a[B, A, D] = a[P, A, D]$ and $a[B, C, D] = a[B, C, P]$ are also generated into $\mathcal{C}$ in this part.

Moving to Step 4, we can generate the sum of tree angles of each triangle since we already know the orientation of all triangles. The equations

(i) $a[B, A, D] + a[D, B, A] + a[A, D, B] = \pi$

(ii) $a[P, A, D] + a[D, P, A] + a[A, D, P] = \pi$

(iii) $a[C, A, D] + a[D, C, A] + a[A, D, C] = \pi$

(iv) $a[C, A, P] + a[P, C, A] + a[A, P, C] = \pi$

(v) $a[C, A, B] + a[B, C, A] + a[A, B, C] = \pi$

(vi) $a[C, P, B] + a[B, C, P] + a[P, B, C] = \pi$

(vii) $a[C, D, B] + a[B, C, D] + a[D, B, C] = \pi$

(viii) $a[P, D, B] + a[B, P, D] + a[D, B, P] = \pi$

are generated. The Step 5, Step 6, Step 7 and Step 8 can be similarly perform as Step 4 since they are the routine generating steps. In Step 7, the similarity equations for the triangles ADP and BCP is generated since we have $a[P, A, D] = a[B, C, P]$ and $a[A, D, P] = a[P, B, C]$ from Step 2 and Step 3. So, the equation $\frac{d[A,P]}{d[C,P]} = \frac{d[D,P]}{d[B,P]}$ is generated into $\mathcal{C}$. Finally, if we command the computer to simplify the expression $d[A, P]\ d[B, P] = d[C, P]\ d[D, P]$ form $\mathcal{C}$, the computer can decides that the expression is true.

Case (b) Similarly to Case(a), the lines PCD and PAB and the points on the same circle C, D, B and A are input. The process can also be perform similarly to Case(a). The triangles PDA and PCB are similar since $a[A, P, D] = a[B, P, C]$ and $a[C, B, P] = a[P, D, A]$ from Step

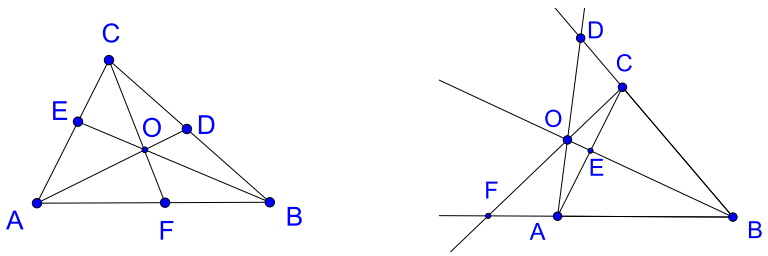
2 and Step 3. So, the equation $\frac{d[A,P]}{d[C,P]} = \frac{d[D,P]}{d[B,P]}$ is also generated and the computer able to conclude the expression $d[A,P] \cdot d[B,P] = d[C,P] \cdot d[D,P]$.

It is observed that in Example 4.1 the Step 5 Step 6 and Step 8 are not necessary to be performed to conclude the expression. We can improve the performance speed by cutting off some of the unnecessary steps that we already know.

4.2 Ceva's theorem

Given a triangle ABC, let the lines AO, BO and CO be drawn from the vertices to a common point O (not on one of the sides of ABC), to meet opposite sides at D, E and F respectively as in Figure 4.2. Then

$$\frac{AF}{FB} \cdot \frac{BD}{DC} \cdot \frac{CE}{EA} = 1. \tag{4.2.1}$$



(a) (b)

Figure 4.2: The two situations for the Ceva's theorem

In this example, we will skip the input and generating part and discuss about the conclusion part. Suppose that we already done all 8 steps for the Case (a). The Ceva's theorem can be proved by using multiple variables equations. To do that, we need to use the area of a triangle's equation. Note that, the area of a triangle of a given height is proportional to its base. So

$$\frac{|\triangle FOA|}{|\triangle BOF|} = \frac{AF}{FB} = \frac{|\triangle FCA|}{|\triangle BCF|}. \tag{4.2.2}$$

Therefore,

$$\frac{AF}{FB} = \frac{|\triangle FCA| - |\triangle FOA|}{|\triangle BCF| - |\triangle BOF|} = \frac{|\triangle OCA|}{|\triangle BCO|}. \tag{4.2.3}$$

Similarly,

$$\frac{CE}{EA} = \frac{|\triangle BCO|}{|\triangle ABO|} \tag{4.2.4}$$

and

$$\frac{BD}{DC} = \frac{|\triangle ABO|}{|\triangle CAO|}. \tag{4.2.5}$$

We combine the equations $(4.2.3)$, $(4.2.4)$ and $(4.2.5)$ for $\frac{AF}{FB} \cdot \frac{BD}{DC} \cdot \frac{CE}{EA} = 1$ as desired.

To measure the area of the triangle FOA, we can consider as

$$\frac{1}{2} \cdot d[A, F] \cdot d[O, F] \cdot Sin(a[A, F, O]). \tag{4.2.6}$$

Since $a[A, F, O] + a[O, F, B] = \pi$, then $Sin(a[A, F, O]) = Sin(\pi - a[O, F, B]) = Sin(a[O, F, B])$. Then, to conclude the equation $(4.2.3)$, we use the fact that

$$\frac{\frac{1}{2} \cdot d[A, F] \cdot d[C, F] \cdot Sin(a[A, F, O]) - \frac{1}{2} \cdot d[A, F] \cdot d[O, F] \cdot Sin(a[A, F, C])}{\frac{1}{2} \cdot d[B, F] \cdot d[C, F] \cdot Sin(a[O, F, B]) - \frac{1}{2} \cdot d[B, F] \cdot d[O, F] \cdot Sin(a[C, F, B])} = \frac{d[A, F]}{d[F, B]}. \tag{4.2.7}$$

The equation $(4.2.7)$ is true by factoring $d[A, F]$ and $d[B, F]$ and use the equalities of the sine functions. To complete the equation $(4.2.3)$, we also need the fact

$$\frac{\frac{1}{2} \cdot d[A, F] \cdot d[C, F] \cdot Sin(a[A, F, O]) - \frac{1}{2} \cdot d[A, F] \cdot d[O, F] \cdot Sin(a[A, F, C])}{\frac{1}{2} \cdot d[B, F] \cdot d[C, F] \cdot Sin(a[O, F, B]) - \frac{1}{2} \cdot d[B, F] \cdot d[O, F] \cdot Sin(a[C, F, B])} = \frac{\frac{1}{2} \cdot d[O, A] \cdot d[O, C] \cdot Sin(a[A, O, C])}{\frac{1}{2} \cdot d[O, B] \cdot d[O, C] \cdot Sin(a[C, O, B])} \tag{4.2.8}$$

The equation $(4.2.8)$ is true from the law of sine equations

$$Sin(a[A, F, O]) \cdot d[A, F] = Sin(a[F, C, A]) \cdot d[A, C] = Sin(a[A, O, C]) \cdot d[O, A] \tag{4.2.9}$$

and

$$Sin(a[O,F,B]) \cdot d[B,F] = Sin(a[B,C,F]) \cdot d[B,C] = Sin(a[C,O,B]) \cdot d[O,B] \tag{4.2.10}$$

which are generated in Step 6 and the equation $d[C,F] - d[O,F] = d[O,C]$ which is generated in Step 5. Combining $(4.2.8)$ and $(4.2.7)$ we have

$$\frac{d[A,F]}{d[F,B]} = \frac{\frac{1}{2} \cdot d[O,A] \cdot d[O,C] \cdot Sin(a[A,O,C])}{\frac{1}{2} \cdot d[O,B] \cdot d[O,C] \cdot Sin(a[C,O,B])} \tag{4.2.11}$$

which is the representation of $(4.2.3)$. The equations represented for $(4.2.4)$ and $(4.2.5)$ are true in similar way. The computer can use the fact of such equations to conclude the expression $\frac{d[A,F]}{d[F,B]} \cdot \frac{d[B,D]}{d[D,C]} \cdot \frac{d[C,E]}{d[E,A]} = 1$. The Case (b) can be done in similar way by replacing the minus in some triangles combining equations to plus.

This example shows the ability to conclude the expressions that require the area equations.

CHAPTER V

CONCLUSION AND SUGGESTION

5.1 Conclusion

We can use the algorithm to help solving the Euclidean geometry problems. Most of the problems can be solved by using our algorithm. If the problem cannot be conclude by this algorithm, such problem might use more additional equations to solve. This is very helpful to identify the difficulty level of the problems.

5.2 Suggestion

In some problem, one or two steps may be skipped as the generated equations are not needed to verify the given statement. For example, some problems don't need the congruent triangles equation. So we can cut off the Step 8 which is not needed for the conclusion to improve the efficiency of the algorithm.

We have tried to use Mathematica 9.0 and it works pretty well but there still some sort of easy command that the Mathematica still does not work correctly. So, there is no program suggestion on what suit to the algorithm. In addition, it will be useful if the software that we use can shows the solution of the conclusion.

To enhance the performance of this algorithm, some additional steps can be added to generate more equations. Presently, the performance of the computer is still not well enough to runs all 8 steps in a big set-up. We hope for the future that our algorithm will be used in a better performance computer.

APPENDIX

APPENDIX

The code for Mathematica

This part shows an example code and explanation of a computer program used the algorithm in Chapter 3. The computer software that we use is Mathematica 9.0.

The following code is the code for Step 1. The initial condition and the input of the set-up will be required the user to input. The order of the points in the same line will be input into the set "lines" and the points on the same circle will be input into the set "circles". The initial conditions can be put in the set "conditions". Some of the sets that we use for the checking remain empty for the further generating.

```
lines = {}; points = {}; circles = {}; angles = {}; conditions = {}; similar = {}; sas = {}; sss = {}; isosceles = {}; Ver = Join[Union @@ lines, points]; triangles = {}; disttri = {}; anglesbwlines = {};
```

The following code is the code for Step 2.

```
oncircles[P_] := Do[oncircle @@ xy, {xy, Subsets[P, {4}]}];
oncircle[x_, y_, z_, w_] := (oncircle1[x, y, z, w];
oncircle1[y, z, w, x]; oncircle1[z, w, x, y]; oncircle1[w, x, y, z];
regangle[x, y, z]; regangle[y, z, w]; regangle[z, w, x]; regangle[w, x, y];
AppendTo[conditions, a[x, y, z] + a[z, w, x] == Pi];
AppendTo[conditions, a[y, z, w] + a[w, x, y] == Pi];);
oncircle1[x_, y_, z_, w_] := (regangle[x, y, w]; regangle[x, z, w];
AppendTo[conditions, a[x, y, w] == a[x, z, w]];);
```

The following code is the code for Step 3.

```
addangle[{x_, y_, z_}] := (AppendTo[angles, {x, y, z}];a[z, y, x] = a[x, y, z]);
```

```
regangle[x_, y_, z_] :=
If[! MemberQ[angles, {x, y, z}],
la = lineof[{x, y}]; lb = lineof[{y, z}];
px = Position[la, x][[1]][[1]]; pya = Position[la, y][[1]][[1]];
pz = Position[lb, z][[1]][[1]]; pyb = Position[lb, y][[1]][[1]];
anglesbwlines = Join[anglesbwlines, {al[la, lb], al[lb, la]}];
AppendTo[conditions, al[la, lb] + al[lb, la] == Pi];
{ix, ixc} =
If[px < pya, {Range[pya - 1],
Range[pya + 1, Length[la]]}, {Range[pya + 1, Length[la]], Range[pya - 1]}];
{jz, jzc} =
If[pz < pyb, {Range[pyb - 1],
Range[pyb + 1, Length[lb]]}, {Range[pyb + 1, Length[lb]], Range[pyb - 1]}];
Do[a[la[[i]], y, lb[[j]]] = al[la, lb];
addangle[{la[[i]], y, lb[[j]]}], {i, ix}, {j, jz}];
Do[a[la[[i]], y, lb[[j]]] = al[la, lb];
addangle[{la[[i]], y, lb[[j]]}], {i, ixc}, {j, jzc}];
Do[a[lb[[j]], y, la[[i]]] = al[lb, la];
addangle[{lb[[j]], y, la[[i]]}], {i, ix}, {j, jzc}];
Do[a[lb[[j]], y, la[[i]]] = al[lb, la];
addangle[{lb[[j]], y, la[[i]]}], {i, ixc}, {j, jz}];];
lineof[s_] : =
(Do[If[subset[s, line], ll = line; Goto[exitlineof]], {line, lines}];
Return[s]; Label[exitlineof]; Return[ll]);
angleof[x_, y_, z_] : = {lineof[{x, y}], lineof[{y, z}]};
subset[S_, T_] : = (Do[If[! MemberQ[T, s], Goto[exitsubset]], {s, S}];
```

```
Return[True]; Label[exitsubset]; Return[False]);

Do[oncircles[circle],{circle, circles}];
```

The following code is the code for Step 4.

```
triangles3pack = {};

changed = True;

While[changed,

changed = False;

Do[ {x, y, z} = xyz; If[! MemberQ[triangles, xyz],

triangles = Join[triangles, {{x, y, z}, {y, z, x}, {z, x, y}}];

AppendTo[triangles3pack, {{x, y, z}, {y, z, x}, {z, x, y}}];

regangle[y, z, x]; regangle[z, x, y];

AppendTo[conditions, a[x, y, z] + a[y, z, x] + a[z, x, y] == Pi];

changed = True],{xyz, angles}]];
```

The following code is the code for Step 5.

```
conditions = Join[conditions,

Complement[Flatten[Table[If[Length[Union[{x, y, z, w}]] == 4 &&

MemberQ[angles, {x, y, w}] && MemberQ[angles, {w, y, z}] && MemberQ[angles, {x, y, z}],

a[x, y, w] + a[w, y, z] == a[x, y, z]], {x, Ver}, {y, Ver}, {z, Ver}, {w, Ver}]], {Null}]];

Do[If[Length[Union[{x, y, z}]] == 3 &&

MemberQ[Subsets[line], {x, y, z}],

conditions = Join[conditions, {d[x, y] + d[y, z] == d[x, z]}]],

{x, Ver}, {y, Ver}, {z, Ver}, {line, lines}]
```

The following code is the code for Step 6.

```
sincheck = {}

While[changed, changed = False; Do[{x, y, z} = xyz;

If[! MemberQ[sincheck, xyz],
```

```
triangles = Join[sincheck, {{x, y, z}, {y, z, x}, {z, x, y}}];
AppendTo[conditions, Sin[a[x, y, z]]/d[x, z] == Sin[a[y, z, x]]/d[y, x] == Sin[a[z, x, y]]/d[z, y]];
changed = True] , {xyz, angles}]];
coscheck = {};
While[changed, changed = False; Do[{x, y, z} = xyz;
If[! MemberQ[sincheck, xyz],
triangles = Join[sincheck, {{x, y, z}, {y, z, x}, {z, x, y}}];
AppendTo[conditions, d[x, z]^2 == d[y, z]^2 + d[x, y]^2 - 2 d[y, z] d[x, y] Cos[a[x, y, z]]];
AppendTo[conditions, d[x, y]^2 == d[x, z]^2 + d[y, z]^2 - 2 d[x, z] d[y, z] Cos[a[y, z, x]]];
AppendTo[conditions, d[y, z]^2 == d[x, z]^2 + d[x, y]^2 - 2 d[x, z] d[x, y] Cos[a[z, x, y]]];
changed = True],{xyz, angles}]];
```

The following code is the code for Step 7.

```
Do[flag[tri3] = 1, {tri3, triangles3pack}];
changed = True;round = 1;
Print[Length[triangles3pack]];
While[changed, Print["round ： ", round];
changed = False; Do[ pack1 = triangles3pack[[i1]]; pack2 = triangles3pack[[i2]];
{x1, y1, z1} = pack1[[1]]; {x0, y0, z0} = pack2[[1]];
Print[i1, " : ", i2]; If[(flag[pack1] == round || flag[pack2] == round)
&& ! MemberQ[similar, {pack1, pack2}],
Do[{x2, y2, z2} = xyz2; Print[xyz2];
If[Simplify[ a[x1, y1, z1] == a[x2, y2, z2] && a[y1, z1, x1] == a[y2, z2, x2], conditions],
Print[x1, y1, z1, " ", x2, y2, z2];
AppendTo[similar, {pack1, pack2}];
conditions = Join[conditions, {d[x1, y1]/d[x2, y2] == d[y1, z1]/d[y2, z2] == d[z1, x1]/d[z2, x2]}];
changed = True;
```

```
flag[pack1] = round + 1; flag[pack2] = round + 1;
Goto[dosas]]
, {xyz2, {{x0, y0, z0}, {y0, z0, x0}, {z0, x0, y0}, {x0, z0,
y0}, {z0, y0, x0}, {y0, x0, z0}}(}]];
If[!MemberQ[sss,{xyz1,xyz0}], Do[{xyz2}= If[Simplify[d[x1,y1]==d[x2,y2]&&
d[x1,z1]==d[x2,z2]&& d[y1,z1]==d[y2, z2],conditions],conditions=Join[conditions,
{a[x1,y1,z1]==a[x2,y2, z2],a[y1,z1,x1]==a[y2,z2,x2]}];
AppendTo[sss,{xyz1,xyz0}];
changed=True;
Goto[endsss]]
,{xyz2,{{x0,y0,z0},{y0,z0,x0},{z0,x0,y0},{x0,z0,y0},{z0,y0,x0},{y0, x0,z0}}}]];
Label[endsss];
Label[dosas];
If[!MemberQ[sas,{xyz1,xyz0}],
If[Simplify[a[z1,x1,y1]==a[z2,x2,y2]&& d[x1,y1]==d[x2,y2]&& d[x1,z1]==
d[x2,z2],conditions],conditions=Join[conditions,{a[x1,y1,z1]==a[x2,
y2,z2](*,a[b,c,a]==,d[b,c]==*)}];
Join[sas,allpairs[xyz1,xyz2]];
changed=True]];
If[!MemberQ[sss,{xyz1,xyz0}],
If[Simplify[d[x1,y1]==d[x2,y2]&& d[x1,z1]==d[x2,z2]&& d[y1,z1]==d[y2, z2],conditions],
conditions=Join[conditions,{a[x1,y1,z1]==a[x2,y2, z2],a[y1,z1,x1]==a[y2,z2,x2]}];
Join[sss,allpairs[xyz1,xyz2]];
changed=True] ];
If[!MemberQ[sas,{xyz1,xyz2000}],
If[Simplify[a[z1,x1,y1]==a[z2,x2,y2]&& d[x1,y1]==d[x2,y2]&& d[x1,z1]== d[x2,z2],conditions],
```

```
conditions=Join[conditions,{a[x1,y1,z1]==a[x2, y2,z2](*,a[b,c,a]==,d[b,c]==*)}];

Join[sas,allpairs[xyz1,xyz2]];

changed=True]

]; If[!MemberQ[sss,{xyz1,xyz2000}],

If[Simplify[d[x1,y1]==d[x2,y2]&& d[x1,z1]==d[x2,z2]&& d[y1,z1]==d[y2, z2],conditions],

conditions=Join[conditions,{a[x1,y1,z1]==a[x2,y2, z2],a[y1,z1,x1]==a[y2,z2,x2]}];

Join[sss,allpairs[xyz1,xyz2]];

changed=True] ]

, {i1, Length[triangles3pack]}, {i2, i1 + 1, Length[triangles3pack]}]; round++]
```

The following code is the code for Step 8.

```
changed = True;

While[changed, changed = False;

Do[ If[! MemberQ[isosceles, xyz1], {x1, y1, z1} = xyz1;

If[Simplify[d[x1, y1] == d[x1, z1], conditions], Print[xyz1];

conditions = Join[conditions, {a[x1, y1, z1] == a[y1, z1, x1]}];

AppendTo[isosceles, xyz1]; changed = True];]

If[! MemberQ[isosceles, xyz1], {x1, y1, z1} = xyz1;

If[Simplify[a[x1, y1, z1] == a[y1, z1, x1], conditions],

conditions = Join[conditions, {d[x1, y1] == d[x1, z1]}];

changed = True] ] , {xyz1, triangles}]; ]
```

REFERENCES

[1] S.C. Chou, X.S. Gao, and J.Z. Zhang, Automated production of traditional proofs for constructive geometry theorems, *Proc. of Eighth IEEE Symposium on Logic in Computer Science*, 1993, 48-56.

[2] H. Gelernter, J.R. Hanson and D.W. Loveland, Empirical explorations of the geometry-theorem proving machine, *Proc. West. Joint Computer Conf.*, 1960, 143–147.

[3] H. Gelernter, Realization of a geometry-theorem proving machine, *Computers and Thought*, eds. E.A. Feigenbaum, J.Feldman, 1963, 134–152.

[4] Mathway software. [Online] Available: `https://mathway.com/`, 2016.

VITA

Name Mr. Parinya Sirikatitum

Date of Birth 15 April 1992

Place of Birth Songkla, Thailand

Education B.Sc. (Mathematics) (First-Class Honors),
Chulalongkorn University, 2013

Scholarship Chulalongkorn University graduate scholarship to commemorate the 72^{nd} Anniversary of his Majesty King Bhumibol Adulyadej

Conference **Speaker**

• *Computerization of Euclidean Geometry Problems*
at the 21^{th} Annual Meeting in Mathematics Annual Pure and Applied Mathematics Conference, 23-25 May 2016
at Chulalongkorn University