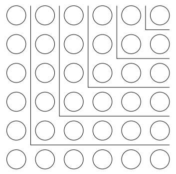
# Automation of Diagrammatic Proofs in Mathematics *

Mateja Jamnik, Alan Bundy, Ian Green †

**Abstract**

Theorems in automated theorem proving are usually proved by logical formal proofs. However, there is a subset of problems which can also be proved in a more informal way by the use of geometric operations on diagrams, so called diagrammatic proofs. Insight is more clearly perceived in these than in the corresponding logical proofs: they capture an intuitive notion of truthfulness that humans find easy to see and understand. The proposed research project is to identify and ultimately automate this diagrammatic reasoning on mathematical theorems. The system that we are in the process of implementing will be given a theorem and will (initially) interactively prove it by the use of geometric manipulations on the diagram that the user chooses to be the appropriate ones. These operations will be the inference steps of the proof. The constructive $\omega$-rule will be used as a tool to capture the generality of diagrammatic proofs. In this way, we hope to verify and to show that the diagrammatic proof of a given theorem can be formalised.

## 1 Introduction



$$1 + 3 + 5 + \ldots + (2n - 1) = n^2$$

It requires only basic secondary school knowledge of mathematics to realise that the diagram above represents a visual (diagrammatic) proof of the theorem about the sum of odd integers.

It is an interesting property of diagrams that allows us to "see" and understand so much just by looking at a simple diagram. Not only that we know what theorem the diagram represents, but we also understand the proof of the theorem represented by the diagram and believe it is correct.

Is it possible to simulate this sort of diagrammatic reasoning on machines? Or is it a kind of intuitive reasoning particular to humans that mere machines are incapable of? Roger Penrose claims that it is not possible to automate such diagrammatic proofs.[1] We are taking his claim as a challenge for this project and will try to capture the kind of diagrammatic reasoning that Penrose is talking about so that we will be able to simulate it on a computer.

The importance of diagrams in many domains of reasoning has been extensively discussed by Larkin and Simon [7], who claim that "a diagram is (sometimes) worth *ten* thousand words". The advantage of a diagram is that it concisely stores information, explicitly represents the relations among the elements of the diagram, and it supports a lot of perceptual inferences that are very easy for humans. For example, the reader should just think of the number of thoughts and information that went through his/her mind when he/she first looked at the diagram above.

It is exactly these characteristics of diagrams that we wish to exploit in our project. In this paper we present a system (which is currently being developed) called DIAMOND (DIAgraMmatic

*The research reported in this paper was supported by Artificial Intelligence Dept. grant, University of Edinburgh, and the Slovenska Znanstvena Fundacija (Slovenian Scientific Fundation) supplementary grant for M. Jamnik, and by EPSRC grant GR/J/80702 for A. Bundy and I. Green.

†Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK. E-mail: matejaj@dai.ed.ac.uk, A.Bundy@ed.ac.uk, I.Green@ed.ac.uk.

[1] Roger Penrose made this claim in the lecture at International Centre for Mathematical Sciences in Edinburgh, in celebration of the 50th anniversary of UNESCO on 8 November, 1995.

reasONing and Deduction), which reasons with diagrams. With this system, the user inputs a theorem to be proved, instructs the system what diagram to start the search for the proof from, and decides what geometric operations to perform during the proof search. It is not the aim of our system to discover proofs, but to formalise and check diagrammatic proofs. Usually, theorems have been *formally* proved with the use of propositional inference steps which often do not convey the intuitive notion of truthfulness to humans. The inference steps are just propositional statements that follow the rules of some logic. The reason we trust they are correct is that the logic has been previously proved to be correct. Following and applying the rules of such a logic guarantees us that there is no mistake in the proof. We might not have such a guarantee in DIAMOND, but will gain a more informal insight of the proof.

The domain of problems that is to be considered in this project is restricted to mathematics. This is still a vast area, which will be further restricted to those theorems which can be represented diagrammatically. Other domain restrictions are still a subject to our research (see Section 2.3).

In summary, the output of DIAMOND is a proof which uses a diagram for diagrammatic inference steps and for guiding the proof. The user supplies the system with the diagram, and interactively instructs it as to what geometric operations to apply in the proof. Ultimately, the entire process will illuminate the issues of formality, rigour, truthfulness and power of diagrammatic proofs.

In Section 2 we list some of the theorems that we aim to prove. Section 3 clarifies DIAMOND's proposed architecture, some operations that will be required and the possible representations to formalise these. Section 4 lists the remaining tasks to successfully implement DIAMOND. Section 5 discusses some of the related diagrammatic reasoning systems. Finally, we conclude by summarising the main points of this paper.

# 2 'Diagrammatic' Theorems

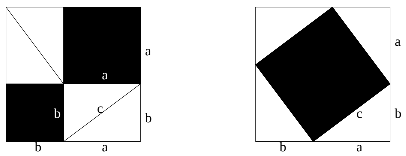
We are interested in mathematical theorems that admit diagrammatic proofs. In order to clarify what we mean by diagrammatic proofs in the scope of our project, we first list some example theorems. Then, we introduce the taxonomy for categorising these examples in order to be able to define the domain of problems that we are going to be concentrating on.

## 2.1 Examples

### Pythagoras' Theorem

The Pythagoras' Theorem states that the square of the hypotenuse of a right angle triangle equals the sum of the squares of its other two sides. Here is one example of many different diagrammatic proofs of this theorem, taken from [8, page 3]:

$$a^2 + b^2 = c^2$$

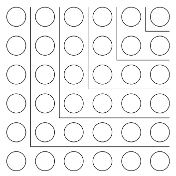



Take any right angle triangle. Along the hypotenuse, join this triangle to another identical right angle triangle, to make a rectangle. Join to this a square on the longer side of one triangle, and a square on the shorter side of the other triangle. Join to both squares, along their adjacent sides another two identical original right angle triangles. This completes the bigger square.

Now re-arrange the triangles into the bigger square so that each side of the square is formed from one side of one and the other side of another triangle. Thus, the size of the bigger square is preserved and the square in the middle is the square on the hypotenuse. Thus, when we subtract the areas of the four triangles from the original bigger and the new square, it is clear that the sum of the squares on the sides of the right angle triangle (in the original bigger square) equals to the square on the hypotenuse of this triangle (in the new square).

**Sum of Odd Integers**

This example is also taken from [8, page 71]. The theorem about the sum of odd integers states the following:

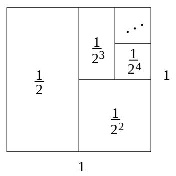


$$1 + 3 + 5 + \ldots + (2n - 1) = n^2$$

Note the use of parameter $n$. If we take a square we can cut it into as many L's (which are made up of two adjacent sides of the square) as the size of the side of the square. Note that one L is made out of two sides, i.e. $(2n)$, but the joining vertex of the two sides has been counted twice. Therefore, one L has a size of $(2n - 1)$, where $n$ is the size of the square.

**Geometric Sum**

This example is also taken from [8, page 118]. The formal definition of a geometric sum is given as follows:




$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1$$

Note the use of ellipsis in the diagram. Take a square of unit size. Cut it down the middle. Now, cut one half of the previous cut square into halves again. This will create two identical squares making up a half of the original square. Take one of these two squares and continue doing this procedure indefinitely.

## 2.2 Classification

From the analysis of the examples that we presented above, and many other that can be found in [4] amongst others, three categories of proofs can be distinguished:

**Category 1:** Proofs that are not schematic: there is no need for induction to prove the general case. Simple geometric manipulations of a diagram prove the individual case. At the end, generalisation is required to show that this proof will hold $\forall a, b$. Example theorem: *Pythagoras' Theorem*.

**Category 2:** Proofs that are schematic: they require no inductive step to prove the theorem for each individual concrete case of a diagram, but require induction for the $n^{th}$ case (a concrete diagram cannot be drawn for this case unless using abstraction). The constructive $\omega$-rule (explained in more detail in Section 2.4) is used to generate a proof for the $n^{th}$ case from the individual proof instances. Example theorem: *Sum of Odd Integers*.

**Category 3:** Proofs that are inherently inductive: for each individual concrete case of the diagram they need an inductive step to prove the theorem. Every particular instance of a theorem, when represented as a diagram requires the use of abstractions to represent infinity. Thus, the constructive $\omega$-rule, which generalises from individual proof instances, is not applicable here. Example theorem: *Geometric Sum.*

## 2.3 Problem Domain

Having introduced the examples and their categorisation, which is by no means exhaustive, we are now able to further restrict our domain of mathematical theorems.

First, we narrow down the domain to a subset of theorems that can be represented as diagrams without the necessity to use abstractions (e.g. the use of ellipsis, as in the above example theorem for *geometric sum*). The use of abstractions in diagrams is quite common. However, it is problematic in diagrammatic proofs, since it is very difficult to keep track of these abstractions while manipulating the diagram during the proof procedure.

Second, we will concentrate on diagrammatic proofs that require induction to prove the general case (i.e. Category 2 above). Namely, diagrams can be drawn only for concrete situations and objects. We cannot draw, for example, an $n \times n$ square. Our challenge is to find a way around it, so that it will be no longer necessary to use abstractions in diagrams. The generality of the proof will be captured in an alternative way (by using the constructive $\omega$-rule – see Section 2.4).

It is clear that we will need a stronger problem domain definition. However, this remains a subject of our research. One possibility is to consider theorems of arithmetic or number theory only. To date, DIAMOND is targeted to prove examples of Category 2, but we may implement diagrammatic theorem proving of examples for Category 1 as well.

## 2.4 Constructive $\omega$-Rule

As mentioned above we could use the constructive $\omega$-rule to prove theorems of Category 2. The work on constructive $\omega$-rule has been done by Siani Baker in [1]. She generated a program for constructing schematic proofs for arithmetic theorems. Here, we explain the idea behind schematic proofs and how it can be applied to diagrammatic proofs.

### 2.4.1 Schematic Proof

Schematic proofs use the $\omega$-rule which is an alternative for induction. The $\omega$-rule is that rule of inference which allows inference of the sentence $\forall x P(x)$ from an infinite sequence $P(n)$ $n \in \omega$ of sentences.

$$\frac{P(0), P(1), P(2), \ldots}{\forall n. P(n)}$$

Unfortunately, the $\omega$-rule, as it stands, is unsuitable for implementation, since an infinite number of premises need to be proved. Baker used the constructive version of the $\omega$-rule which states that "if each P(n) can be proved *in a uniform way* (from parameter n), then conclude ∀n P(n)." The criterion for uniformity of the procedure of proof using the constructive $\omega$-rule is taken to be the provision of a general schematic proof, namely the proof of $P(n)$ in terms of $n$, where some rules $R$ are applied some function of $n$ (i.e. $f_R(n)$) times (a rule can also be applied a constant number of times). Now, $proof(n)$ is schematic in $n$, since we applied some rule $R$ $n$ times. The following procedure summarises the essence of using the constructive $\omega$-rule in schematic proofs:

1. Prove a few special cases (e.g. $P(0)$, $P(16)$, ...).
2. Generalise (guess) $proof(n)$ (e.g. from $proof(0)$, $proof(16)$, ...).
3. Prove that $proof(n)$ proves $P(n)$ by meta-induction on $n$.

The general pattern is extracted (guessed, to be exact) from the individual proof instances by (learning type) inductive inference. By meta (mathematical) induction we mean that we introduce system $PA_\omega$ (i.e. Peano Arithmetic with $\omega$-rule) such that:

$$proof(n) : P(n) \vdash_{PA_\omega} proof(s(n)) : P(s(n))$$

where ":" stands for "is a proof of", and $s(n)$ is a notation for successor of $n$. This essentially says that by using the rules on $P(s(n))$ we can reduce it to $P(n)$. For more information, see [1].

#### 2.4.2 Diagrams and Schematic Proofs

We claim that we can apply Baker's work on schematic proofs in our diagrammatic proofs in that the generality of the diagrammatic proof is embedded in the schematic proof. Thus, we can overcome the problem of diagrammatic reasoning in which only concrete diagrams can be manipulated.

The diagrammatic schematic proof starts with a few particular concrete cases of the theorem represented by the diagram. The diagrammatic procedures (i.e. operations) on the diagram are performed next, capturing the inference steps of the diagrammatic proof. In DIAMOND, this step (also referred to as the proof checking step) is done *interactively* with the user, and corresponds to the first step of the schematic proof procedure given in the previous section.

The second step in the schematic proof procedure is generalisation, i.e. we need to generalise the operations involved in the schematic proof for the $n^{th}$ case. Note that the generality is represented as a sequence of diagrammatic procedures (operations) and not as a general representation of a diagram. In DIAMOND, this step will be done *automatically*, but to date has not been implemented.

The last step in the schematic proof procedure is to prove by meta-induction that the generalised diagrammatic schematic proof is indeed correct. It remains a subject of this research project to determine whether this will be implemented at all or not. An alternative at this point could be to translate the diagrammatic proof to an algebraic proof.

### 2.5 Schematic Diagrammatic Proof for the Sum of Odd Integers

Now we can attempt to structure the diagrammatic proofs in a more formal way. Here we list the proof for the theorem about the sum of odd integers as a sequence of steps that need to be performed on the diagram:

1. Cut a square into $n$ L's, where an L consists of 2 adjacent sides of the square.
2. Cut each L into two segments.
3. For each L, join these two segments one on top of the other length-wise (note that one of the two segments is always one unit longer than the other, thus an L always consists of an odd number of units).

Identifying the operations (i.e. geometric manipulations) that were required to prove the theorem will help us define a large repertoire of such operations which will be used in the diagrammatic proofs. The operations that were performed under special constraints on the diagram in this example are *cut* and *join*. The generality of the proof is captured by the use of the constructive $\omega$-rule, by which we take a few special cases of the diagram (say a square of size 15 and 16), and find the general pattern of the proof that will hold for each case (i.e. the schematic proof given above).

## 3 Proposed System

Here, we shall list different representations that we can use for the diagrams. Then, we discuss the architecture for DIAMOND. Finally, we will discuss the geometric operations required for conducting diagrammatic proofs.

## 3.1 Representations

During the development of DIAMOND, an important aim is to devise the best representation for its *internal* representation of objects and manipulations. The hope is that these capture the intuitiveness, rigour and simplicity of human reasoning with diagrams.

Humans with our vision perception can observe and inspect diagrams directly and see (depending on how accustomed we are to spatial mental manipulations) the inference that needs to be made to prove a theorem in a diagrammatic way. A computer does not possess such visual abilities. Thus, we need to choose a way for the theorem prover to understand in a similar manner the diagrams and the manipulations by non-visual means. There are several representations available to us. Some of them are the following:

- Cartesian representation: diagrams are represented in terms of the coordinate system, typically 2 or 3 dimensional. On the computer, they are implemented as lists of sets of coordinates.
- Projective geometry
- Diagrams on a raster
- Vector representation
- Topological (relational) representation: it expresses the relation between the elements of a diagram.

So far, we have used a mixture of Cartesian and topological representation in the current prototype of DIAMOND. However, in the future more work has to be done on this issue.

## 3.2 Architecture

There are three most obvious functions of the system which we want to implement. These include dealing with a diagram, dealing with the diagrammatic inference steps, and mediating the communication between these two functions. This suggests that DIAMOND should consist of the following three parts:

**1) Inference engine:** This is the knowledge base component of the system, and hence the main component of the system. It generates and processes strings of goals and geometric operations (instructions) that are passed to the diagram. It accepts and inspects the diagram. The important submodule of the inference engine guesses a generalised schematic proof (i.e. implements the constructive $\omega$-rule).

**2) Diagrammatic component:** The Cartesian representation of a diagram is formed in this component. It puts into effect the geometric operation, and modifies the current diagram which is then passed back to the inference engine.

**3) Translation component:** it mediates the communication between the inference engine and the diagrammatic component.

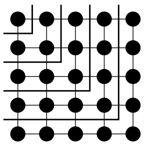
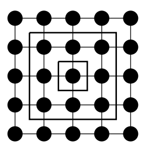
In DIAMOND we want to reverse the role that the symbolic inferencing and the diagram typically have by designing a system that will use diagrammatic inference steps instead of symbolic ones. Thus, the high level reasoning will be entirely dependent on the diagram and the diagrammatic methods. The propositional statement of the theorem will help guiding our proof search in the sense that it will suggest to us what diagram to start the proof from, and what possible diagrammatic inference steps might be next in the proof. For example, if we are dealing with the theorem, the conclusion of which is $n^2$, then this is a clear suggestion that the initial diagram for a proof should be a square. It remains a matter of our research to determine how many of these heuristic methods will be performed by users, how many will be automated, and to what extent these methods can be automated.

### 3.3 Operations

In Diamond, geometric operations (also referred to as manipulations or procedures) capture the inference steps of the proof. Thus, we need to identify a large number of such operations which will be available to us in our search for the proof. These are then formalised, so that Diamond can be instructed to apply them in the proof. Since we are not generating, i.e. discovering diagrammatic proofs, but rather we are trying to understand them, we can allow the user to input these operations, i.e. instruct Diamond to apply them. Before these operations are identified and formalised, their properties (variant, invariant, created) need to be analysed. To date, only a small number of such operations is implemented and available to the user. We distinguish between two types of operations:

**Atomic operations:** they are basic one-step operations that will be combined into more complex operations. Examples of such operations are: rotate, translate, cut, join, project from 3D to 2D, remove, insert a segment.

**Composite operations:** they are more complex, typically recursive operations, composed from simple atomic ones. These have not been implemented yet. Perhaps, we can think of them as tactics or tacticals in automated reasoning. In the future we need to look into the issues of recursive structures of shapes and objects. Namely, we can identify several different recursive structures of the same diagram (see the two diagrams below). Depending on the theorem that we are proving, we use a different recursive composite operation. Ideally, the internal representation of the diagram should be pertinent to the composite operation that we are performing on it.

## 4 Remaining Tasks

Throughout this paper, we have indicated which features of Diamond have already been implemented. These include representations of a small number of shapes and operations on those shapes. we have also identified a number of issues that need to be considered more closely in the future. They give us an indication as to what are the main tasks that will need to be accomplished in the course of the next two years to complete the implementation of Diamond. These are:

1. Implement an instance of one complete proof.
2. Implement a tool for generalisation of schematic proofs.
3. Implement representations for other diagrams.
4. Implement more geometric operations.
5. Address the issue of user interface and improve it.
6. Testing: especially to verify that the general schematic diagrammatic proof is indeed correct. In the last stage of testing of Diamond we hope to be able to prove a large number of theorems of sufficient depth and range.

## 5 Related Work in Diagrammatic Reasoning

Diagrammatic reasoning is a relatively new research area, dealing with the issues of visual representations and the role of diagrams in human thinking. To date, several diagrammatic reasoning systems have been implemented. One of the first ones to use diagrams in proving theorems was Gelernter's Geometry Machine [3]. The Geometry Machine uses a diagram to prune the search space by rejecting hypotheses (subgoals) that are not true in the diagram, and to shorten the inference paths by assuming various facts that are obvious in the diagram as true, i.e. it verifies the correctness of simple goals by checking them in the diagram.

Koedinger and Andreson [6] implemented a geometry problem solver called the Diagram Configuration model. The key feature of the system is that it organises its data in perceptual chunks, called diagram configurations. These are analogous to key features of diagrams that humans recognise when they inspect a diagram. Therefore, during the process of generating a solution path, DC infers the key steps first, and ignores along the way the less important features of the diagram.

&/GROVER, developed by Barker and Bailin [2] is an automated reasoning system which uses information from a diagram to guide proof search. It consists of the & automated theorem prover, and GROVER which is the diagram interpreting component of the system, which passes the crucial information to prove the theorem from the inspected diagram to the & theorem prover. How GROVER works is that the information is extracted from the diagram and translated into logical formulae in the language of & which are then proved by &. Then they are used as additional hypotheses (lemmas) to the main proof of the conjecture.

Several other systems such as McDougal's Polya, Barwise's Hyperproof and Lindsay's Archimedes are also of relevance to our system. For more information and additional references, refer to [5] which is a literature survey on existing diagrammatic systems in different problem domains.

## 6 Conclusion

In this paper we discussed a diagrammatic reasoning system, DIAMOND, currently being developed, which interactively proves theorems of mathematics by the use of geometric operations on a diagram. These geometric operations capture the inference steps of the proof. The hypothesis of this research project could be summarised as the following:

> **Hypothesis:** *It is possible to formalise and automate the kind of 'informal' diagrammatic reasoning that mathematicians employ when solving problems.*

The entire process of research, i.e. emulating the 'informal' diagrammatic reasoning of humans on machines, will shed some light on the issues of formality, informality, rigour and 'intuitive' understanding of the correctness of diagrammatic proofs.

This paper is an extended abstract of the original thesis proposal. For a fuller discussion of the points raised here, refer to [4].

## References

[1] S. BAKER, A. IRELAND, AND A. SMAILL. On the use of the constructive omega rule within automated deduction. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning — LPAR 92, St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 214–225. Springer-Verlag, 1992.

[2] D. BARKER-PLUMMER AND S.C. BAILIN. Proofs and pictures: Proving the diamond lemma with the GROVER theorem proving system. In *Working Notes of the AAAI Symposium on Reasoning with Diagrammatic Representations*, Stanford, USA, March 1992.

[3] H. GELERNTER. Realization of a geometry theorem-proving machine. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 134–52. McGraw Hill, 1963.

[4] M. JAMNIK. Automation of diagrammatic proofs in mathematics. Discussion Paper 173, Dept. of Artificial Intelligence, University of Edinburgh, September 1996.

[5] M. Jamnik. Diagrammatic reasoning systems. Working Paper 260, Dept. of Artificial Intelligence, University of Edinburgh, September 1996.

[6] K.R. Koedinger and J.R. Anderson. Abstract planning and perceptual chunks. *Cognitive Science*, 14:511–550, 1990.

[7] J.H. Larkin and H.A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[8] R.B. Nelsen. *Proofs Without Words: Exercises in Visual Thinking*. The Mathematical Association of America, 1993.