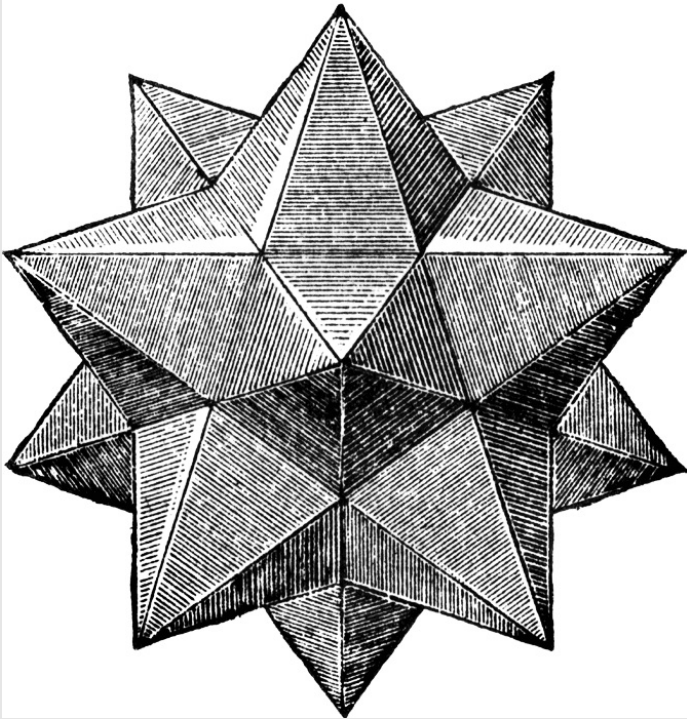


# Euler's polyhedron formula in mizar

Jesse Alama  
Center for Artificial Intelligence  
New University of Lisbon  
Portugal

September 14, 2010 /  
ICMS 2010 / Kobe, Japan



Philosophical motivation

---

Details

---

Future work





*Is formal logic an appropriate  
tool for understanding  
mathematical knowledge?*

---

Despite the evidently successful application of formal logic to mathematics, Lakatos emphasizes the **growth** of mathematical knowledge.





To illustrate his argument, Lakatos discusses the history of **Euler's polyhedron formula**.

---

He shows how the theorem, its proofs, and the concepts involved therein **developed** and **grew** (and how they were at times **rejected**).

---

The overall challenge is:  
**How can formal logic account for these aspects of mathematical knowledge?**





What can one *discover* in a formalised theory? *First*, one can discover the solution to problems which a suitably programmed Turing machine could solve in a finite time (such as: is a certain alleged proof a proof or not?).

No mathematician is interested in following out the dreary mechanical 'method' prescribed by such decision procedures. *Secondly*, one can discover the solutions to problems (such as: is a certain formula in a non-decidable theory a theorem or not?), where one can be guided only by the 'method' of 'unregimented insight and good fortune'.





I propose to retain the time-honoured technical term 'proof' for a *thought-experiment—or 'quasi-experiment'—which suggests a decomposition of the original conjecture into subconjectures or lemmas, thus embedding it in a possibly quite distant body of knowledge*



Lakatos's views might have been plausible in the 1950s and 1960s when he was working in philosophy of mathematics.

---



Are they as plausible now, in the face of significant progress on interactive proof assistants, with which many formalizations of significant mathematical theorems have been carried out?

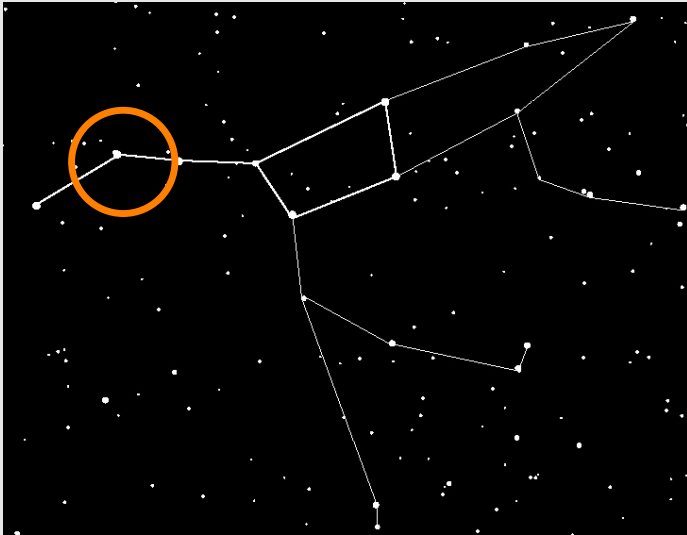
---

To take on Lakatos's challenge, let's formalize Euler's polyhedron formula in some modern interactive proof assistant.



mizar is a proof checking system based on classical first-order logic and (first-order) set theory.

---



its library of formalized mathematical knowledge is quite large:

- nearly 1100 'articles',
  - about 50000 definitions,
  - about 10000 theorems.
- 

its proof formalism  
natural deduction





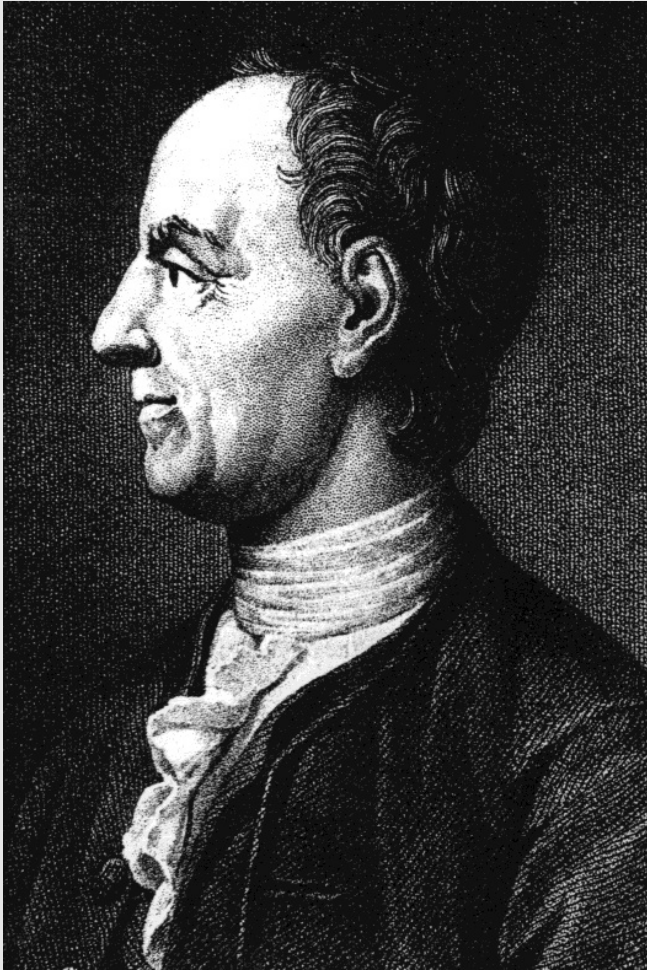
## “Why mizar and not system X?”

---

1. I needed to learn something quickly
  2. mizar’s foundations (first-order classical logic, set theory, declarative natural deduction proofs) are familiar
  3. its library was rich enough for my purposes
- 

Plausible alternatives (Isabelle, Coq, HOL light, etc., etc.) were not seriously considered, even when they satisfied all these conditions.





## Euler's polyhedron formula

---

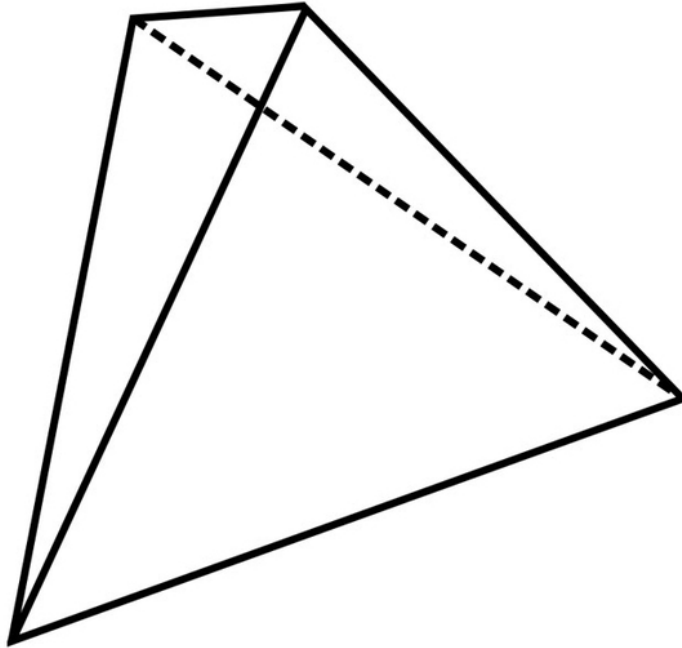
For a polyhedron  $p$ , we have

$$V - E + F = 2,$$

where  $V$ ,  $E$ , and  $F$  are, respectively, the numbers of vertices, edges, and faces of  $p$ .



# What representation of polyhedra?



0	1	1	0	1	0
1	0	0	0	1	1
1	1	0	1	0	0
0	0	1	1	0	1

1	0	0	1
0	1	0	1
0	0	1	0
1	1	0	0
0	1	1	1
1	0	1	0



# Generalization of Euler's formula

- For a polyhedron  $p$  of dimension  $d$ , we have

$$\sum_{k=0}^{d-1} (-1)^k N_k = 1 - (-1)^{d+1},$$

where  $N_k$  is the number of polytopes of  $p$  of dimension  $k$ .

- Note that when  $d = 3$ , we recover the classical form of Euler's formula.
- When  $d = 2$ , we get the familiar relation that  $V = E$ , for a polygon.
- With  $d = 1$ , we get the even more familiar relation that  $V = 2$  for a line segment.





## Poincaré's algebraic topological proof of Euler's formula

---

Introduces basic concepts of algebraic topology, such as chain and cycle spaces.

---

A streamlined presentation of Poincaré's proof is given by Lakatos himself; that one was chosen for formalization.



# Some mathematical details

- $k$ -chain: set of  $k$ -dimensional polytopes
- $k$ -chain space  $C_k$ : vector space over the two-element field  $F_2$  whose elements are chains (0 is the empty chain, addition is disjoint union, scalar multiplication is as expected); since the singletons  $\{a\}$ , where  $a$  is a  $k$ -dimensional polytope, are linearly independent and span  $C_k$ , we have that  $\dim C_k$  is the number of  $k$ -polytopes.
- boundary operator(s)  $\partial_k(c)$ : yields the  $(k-1)$ -dimensional polytopes that are on the 'boundary' of the  $k$ -chain  $c$
- $\partial_k$  is a linear transformation from  $C_k$  to  $C_{k-1}$
- $k$ -cycle space  $Z_k$ : kernel of  $\partial_k$
- bounding  $k$ -chain space  $B_k$ : image of  $\partial_{k+1}$



# Definitions in mizar

definition

let  $p$  be polyhedron,  $k$  be Integer;

func

$k$ -chain-space  $p$   $\rightarrow$  finite-dimensional VectSp of  $Z_2$

equals

bspace  $k$  polytopes  $p$ ;

end;



# A little theorem

theorem

num-polytopes  $p$  ,  $k = \dim$   $k$ -chain-space  $p$

proof

set  $n = \dim$   $k$ -chain-space  $p$ ;

singletons  $k$ -polytopes  $p$  is Basis of  $k$ -chain-space  $p$

by [BSPACE:41](#);

then  $n = \text{card}$  (singletons  $k$ -polytopes  $p$ )

by [VECTSP\\_9:def 2](#);

hence thesis by [BSPACE:42](#);

end;



# Boundary operator

## definition

let  $p$  be polyhedron;

let  $k$  be Integer;

let  $v$  be Element of ( $k$ -chain-space  $p$ );

func

Boundary  $v \rightarrow$  Element of ( $(k-1)$ -chain-space  $p$ )

means

(( $(k-1)$ -polytopes  $p$  is empty  
implies it = 0. ( $(k-1)$ -chain-space  $p$ ))

&

(not ( $k-1$ )-polytopes  $p$  is empty  
implies

for  $x$  being Element of ( $k-1$ )-polytopes  $p$

holds

( $x$  in it iff Sum (incidence-sequence  $x, v$ ) = 1.Z\_2));

existence;

uniqueness;



# Boundary as linear transformation

theorem

for  $c, d$  being Element of  $k$ -chain-space( $p$ ) holds

$$\text{Boundary}(c+d) = \text{Boundary}(c) + \text{Boundary}(d)$$

theorem

for  $a$  being Element of  $Z_2$ ,

$c$  being Element of  $k$ -chain-space( $p$ ) holds

$$\text{Boundary}(a*c) = a*(\text{Boundary}(c))$$

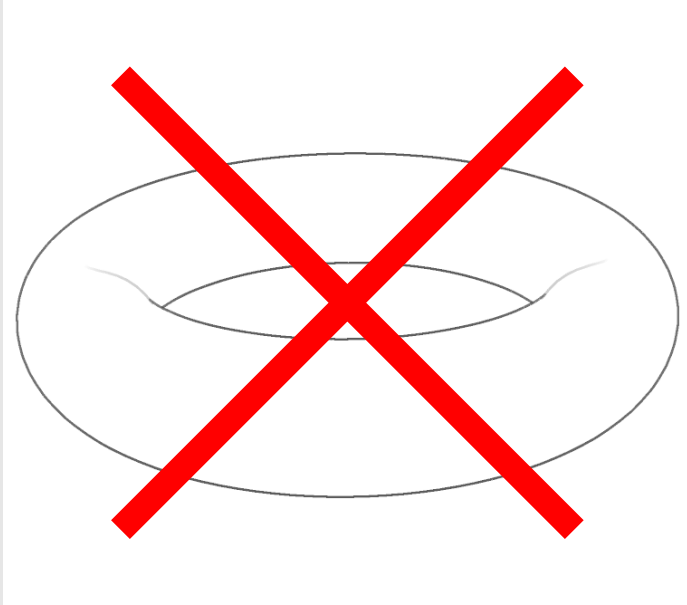


# Definition of polyhedron in mizar

```
definition
  mode
  polyhedron
  is
  polyhedron_1 polyhedron_2 polyhedron_3 PolyhedronStr;
end;
```

(Taken together, the mizar attrutes polyhedron\_1, polyhedron\_2, and polyhedron\_3 say that a PolyedronStructure is a polyhedron if it consists of finite polytope sets and incidence matrices for these sets.)





## Essential missing condition

---

Simple connectedness:  
the polyhedron should be  
homologous to a sphere  
(and not to a torus, etc.).



# 'No holes' in mizar

definition

let  $p$  be polyhedron;

attr

$p$  is simply-connected

means

for  $k$  being Integer

holds

$k$ -circuits  $p$  =  $k$ -bounding-chains  $p$ ;



# Main Theorem

theorem

$p$  is simply-connected implies  $p$  is eulerian

theorem

$p$  is simply-connected &  $\dim p = 3$

implies

num-vertices  $p$

- num-edges  $p$

+ num-faces  $p$

= 2;



# Results

## ■ Three mizar articles:

- RANKNULL: the rank+nullity theorem: if  $T$  is a linear transformation from a finite-dimensional vector space  $V$  to a finite-dimensional vector space  $W$ , then

$$\dim V = \dim \operatorname{im} T + \dim \ker T.$$

- BSPACE: the powerset of a set forms a vector space over the two-element field  $Z_2$
- POLYFORM: Euler's formula proper.

## ■ Total: about 6000 lines of text (about 60 definitions and 180 theorems)



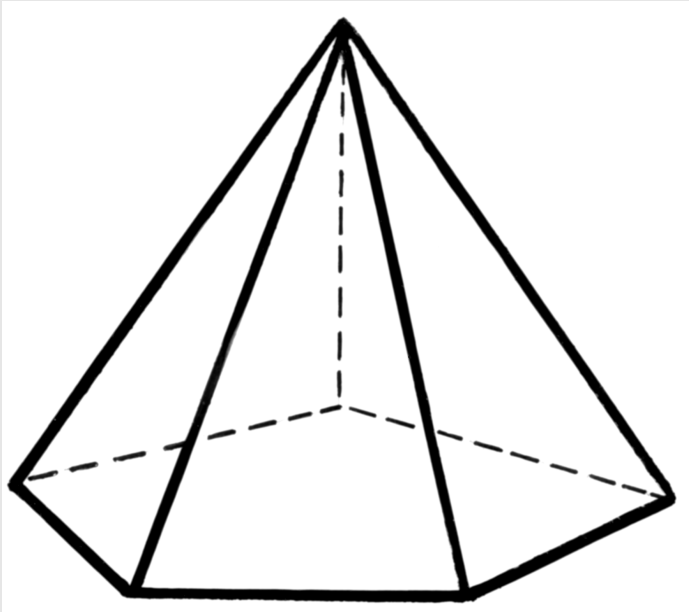
# Future work

- But Lakatos's treatment of Poincaré's proof is too abstract/combinatorial. Some contact with real algebraic topology would be an improvement.
- Such an improved formalization of Euler's polyhedron formula is underway (based primarily on Pontryagin's *Foundations of Combinatorial Topology*).
- It lifts the restriction of working with homology spheres (polyhedra on surfaces of characteristic zero).
- The main focus is on the Euler-Poincaré formula: for all polyhedra  $p$  we have

$$\sum_{k \geq 0}^n (-1)^k N_k = \sum_{k=0}^n b_k,$$

where  $N_k$  is the number of  $k$ -dimensional polytopes of  $p$  and  $b_k$  is the  $k$ -dimensional Betti number of  $p$ .





Thanks

