DEMUESTRA - Complete Installation and Usage Guide

The complete **DEMUESTRA** project can be downloaded from GitHub or consulted there in full under user **quark-cha**, repository **DEMUESTRA**: https://github.com/quark-cha/DEMUESTRA.

1. What DEMUESTRA is

DEMUESTRA is an environment for submitting a mathematical proof to a process of structural analysis and formal verification through **Lean 4**.

The researcher starts from their own proof written in a Markdown `.md` file.

During the process, a **free AI may be used as an auxiliary tool** for tasks such as:

- writing formulas correctly in LaTeX;
- transforming the proof into JSON;
- transforming mathematical propositions into Lean;
- locating errors;
- linking an error to the corresponding point in the proof;
- helping to correct syntax or formalization problems.

The AI **does not take part in the creative process of the proof**.

The AI does not provide the mathematical idea and does not replace the researcher's reasoning: it only acts as a tool to adjust the proof formally, organize it, translate formats, and help express precisely a proof that already belongs to the author.

The proposal, its hypotheses, definitions, reasoning, and conclusions belong to the author.

DEMUESTRA also does not aim to decide, by means of a simple score, whether a scientific theory is true or false.

Its objective is much more specific:

to make the deductive chain explicit and to formally verify that the mathematical steps that have been brought into Lean are correctly built from their premises.

---

2. Authorship, License, and Ethical Use

This project is part of the development work of **Victor Estrada Diaz**.

You are free to use these utilities in your own project as long as you respect my authorship and do not use them for commercial purposes.

You are also free to distribute and share them as long as you clearly state the part for which I am the author.

The applicable license is:

```
CC BY-SA-NC
```

The use of any part of this development is prohibited in unethical environments, including:

- wars;
- uses that do not comply with Human Rights;
- exploitation or abuse of people or animals;
- uses intended to humiliate or undervalue other people.

Any such use violates my license.

---

3. Updated Repository

The **DEMUESTRA** project has a public repository from which you can download the updated version and access all its files.

GitHub user:

```
quark-cha
```

Repository:

```
DEMUESTRA
```

PART I - INSTALLATION

4. What you need to install

Before using DEMUESTRA, you need to prepare the computer.

Required

Install:

1. **Python**
2. **Lean 4**
3. the **DEMUESTRA** repository

Recommended

We also recommend:

4. **Anaconda**
5. **Visual Studio Code**

Visual Studio Code is not essential, but it makes editing `.md`, `.json`, and `.lean` files much easier.

---

5. Download DEMUESTRA

Download or clone the **DEMUESTRA** repository into your usual development folder.

For example, you may have a general structure similar to:

```
MIS_PROYECTOS/
│
├── PROYECTO_A/
├── PROYECTO_B/
├── PUBLICAR/
└── DEMUESTRA/
```

Your projects, **PUBLICAR** if you use it, and **DEMUESTRA** are therefore at the same level.

DEMUESTRA does not need to directly modify your original projects.

PUBLICAR is optional. You do not need to install PUBLICAR to use DEMUESTRA. PUBLICAR is only useful if you also want to use that other PDF generation and publication system.

PART II - PREPARING A PROOF

6. The original document

Suppose you have a project called:

```
MI_TEORIA/
```

and inside it you have written a proof:

```
MI_TEORIA/
└── demostracion.md
```

This `.md` is your original document.

We are not going to use that file directly during the tests.

First, we make a copy so that we can work with it inside DEMUESTRA.

---

7. Copy the Markdown file into DEMUESTRA

Copy the `.md` file that you want to prove into DEMUESTRA's project area.

Conceptually:

```
MI_TEORIA/
└── demostracion.md
          |
          | COPY
          v
DEMUESTRA/
└── proyectos/
    └── MI_TEORIA/
        └── demostracion.md
```

From this moment on, two documents must be clearly distinguished:

```
MI_TEORIA/demostracion.md
```

is the document that remains in your original project.

Whereas:

```
DEMUESTRA/proyectos/MI_TEORIA/demostracion.md
```

is the **working copy**.

Important rule

Throughout the entire verification process:

work and make corrections on the `.md` file located inside DEMUESTRA.

You do not need to continuously modify the original project.

Once the whole process has been completed satisfactorily, you can copy the corrected `.md` file from DEMUESTRA back to your original project.

PART III - HOW TO WRITE THE MARKDOWN

8. The `.md` file

Markdown is simply a plain text format.

You can write it with any text editor, although Visual Studio Code makes the work easier.

Mathematical formulas are written using LaTeX.

Inline formulas

In your actual Markdown file, an inline formula is delimited with the $ character at the beginning and another $ character at the end.

It must be written like this, as text:

$formula$

For example:

Energy is given by $E=mc^2$.

Display formulas

In your actual Markdown file, a display formula in its own paragraph is delimited with two $$ characters at the beginning and two $$ characters at the end.

It must be written like this, as text:

$$formula$$

For example:

$$E=mc^2$$

You can ask an AI to review the document and place the formulas and their delimiters correctly.

PART IV - THE PDF

9. Do I need to generate a PDF?

No.

The PDF is not necessary to carry out the proof.

The fundamental working document is the `.md` file.

To start working, you only need that `.md` file.

With the `.md` file and the help of an AI, you can extract the arithmetic and deductive derivations that will later be brought into the Lean file.

You can generate a PDF if you want a version intended for reading, distribution, or publication.

You can do it with an AI or with any specific conversion tool.

In my case, I use another project called **PUBLICAR**.

PUBLICAR converts my projects to PDF and automates their publication in Zenodo and on my website.

However, PUBLICAR uses my own project structure.

Therefore:

PUBLICAR is optional and is not necessary to use DEMUESTRA.

You should not install PUBLICAR if you only want to verify a proof with DEMUESTRA.

PART V - STRUCTURAL ANALYSIS

10. Convert the Markdown to JSON

A proof may seem evident to the person who developed it and yet contain concepts that have not been explicitly defined or relationships that have been assumed.

That is why we first perform a structural analysis.

You can ask a free AI to transform your Markdown proof into the JSON structure required by **Gianluca R. Pisano**'s analyzer.

The process is:

```
demostracion.md
      |
      v
     AI
      |
      v
demostracion.json
```

Here, the AI is performing a **translation of format and structure**.

It must not invent new premises or modify your proof in order to make it pass the analysis.

---

11. Run the JSON through the analyzer

Enter the obtained JSON into Gianluca R. Pisano's analyzer.

The analyzer may detect problems such as:

- symbols that appear without having been defined;
- insufficiently specified concepts;
- dependencies that are not clear;
- propositions that use previous results without identifying them;
- gaps in the deductive chain;
- hypotheses that need to be made explicit;
- problems of formal structure.

---

12. An analyzer error does not necessarily mean that you are wrong

This point is fundamental.

If the analyzer detects a problem, **it does not automatically mean that the mathematical claim is false**.

It may mean, for example, that you know perfectly well why a result follows from the previous one, but you have not written it explicitly.

In that case, the problem is in the formal exposition of the proof.

But we must not ignore the error either.

We must find out exactly what it is pointing to.

PART VI - CORRECTING THE PROOF

13. Always return to the Markdown

The document that must be corrected is:

```
DEMUESTRA/proyectos/MI_TEORIA/demostracion.md
```

We must not turn the JSON into our main document.

The origin of the proof always remains the Markdown file located inside DEMUESTRA.

You can use an AI to ask:

> Which part of my Markdown file corresponds to this error detected by the analyzer?

Then you study the problem and decide what needs to be corrected.

---

14. Generate the JSON again

After modifying the Markdown:

```
Corrected Markdown
        |
        v
new JSON
```

```
        |
        v
analyzer
```

If new problems appear:

```
 analyzer
     |
     v
locate problem
     |
     v
correct .md
     |
     v
generate JSON
     |
     v
analyze again
```

This cycle may be repeated as many times as necessary.

The principle is:

do not modify the proof simply to satisfy the analyzer; first understand what it is pointing to and correct what genuinely needs to be made explicit or corrected.

PART VII - ENTERING DEMUESTRA

15. When to use Lean

When the mathematical chain is sufficiently defined and structured, we move on to formal verification through Lean 4.

Now we need to create a file:

```
.lean
```

corresponding to our proof.

---

16. Create the Lean file

You can use a free AI to produce a first translation of the mathematical propositions from the Markdown into the Lean language.

You should ask it to bring into the Lean file:

- definitions;
- variables;
- domains;
- hypotheses;
- propositions;
- lemmas;
- theorems;

- dependencies between results.

The objective is not for the AI to invent a different proof.

The objective is **to formalize the proof that already exists in the Markdown**.

For the final mathematical proof of the deductions, DEMUESTRA only needs the `.lean` file.

The relationship must be traceable:

```
Markdown
  |
  ├── definition
  ├── proposition 1
  ├── proposition 2
  ├── lemma
  └── theorem
        |
        v
      Lean
        |
  ├── definition
  ├── proposition 1
  ├── proposition 2
  ├── lemma
  └── theorem
```

In this way, if Lean finds a problem, we can return to the corresponding point in the Markdown.

PART VIII - RUNNING DEMUESTRA

17. Run the program

Open a console or terminal.

Move to the corresponding root folder and run:

```
python PckDemuestra\demuestra.py
```

DEMUESTRA will search for the projects placed in its working folder:

```
DEMUESTRA/proyectos/
```

and will process the projects it finds there.

That is why it does not work directly on your original projects.

It works on the copies that you have decided to place inside `DEMUESTRA/proyectos` .

---

18. What DEMUESTRA does

DEMUESTRA uses Lean 4 to verify the Lean files associated with the projects located in its working area.

The program will report which ones compile correctly and which ones contain errors.

A successful compilation means that Lean has been able to verify the provided formal construction.

An unsuccessful compilation means that there is something to study.

PART IX - WHEN LEAN FINDS AN ERROR

19. Do not correct blindly

If Lean produces an error, we should not simply ask an AI:

> Make it compile.

That could hide precisely the problem we are trying to discover.

We must locate:

```
 Lean error
     |
     v
affected Lean proposition
     |
     v
corresponding Markdown proposition
     |
```

```
    v
mathematical reasoning
```

You can use an AI to help you locate that correspondence.

---

20. Two fundamental kinds of problems

An error may be due simply to the formalization.

For example:

- incorrect Lean syntax;
- undeclared variable;
- incorrect type;
- a hypothesis that has not been transferred;
- incomplete definition.

In that case, we correct the `.lean` file.

But Lean may also reveal that a certain result **cannot be obtained from the premises that we have formalized**.

Then we must return to the `.md` file and study that step.

A justification may be missing.

A hypothesis may be missing.

Or there may truly be a mathematical error.

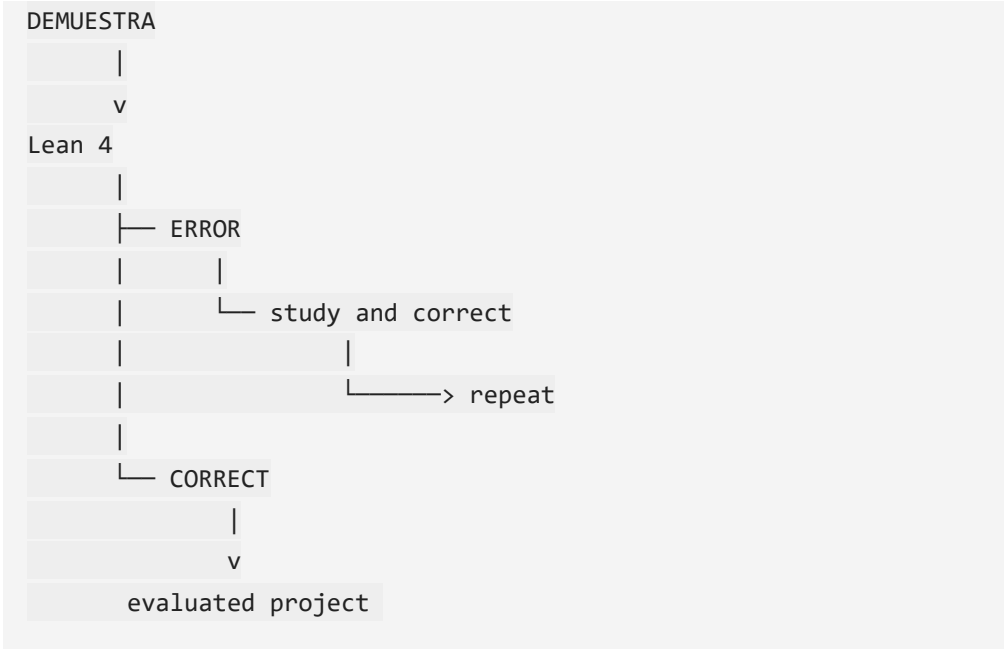
DEMUESTRA is useful precisely because it forces us to distinguish these situations.

PART X - REPEATING THE PROCESS

21. Verification cycle

The complete procedure can be represented as follows:

```
 ORIGINAL PROJECT
       |
       | copy .md
       v
DEMUESTRA/proyectos
       |
       v
Working Markdown
       |
       v
JSON
       |
       v
Gianluca R. Pisano's analyzer
       |
       v
Corrections
       |
       v
Corrected Markdown
       |
       v
Lean
       |
       v
```



PART XI - ALREADY EVALUATED PROJECTS

22. Avoid reprocessing a project

Once the process has finished, DEMUESTRA can move the project, **if you wish**, from the pending projects area to the area intended for already evaluated projects.

This prevents it from being processed unnecessarily again on the next run.

The logic is:

```
DEMUESTRA/
|
├── proyectos/
|   ├── pending project
|   └── project under evaluation
|
└── evaluados/
    └── already evaluated project
```

Thus:

```
proyectos
```

contains pending or active work.

And:

```
evaluados
```

contains the projects that have already completed the process.

PART XII - RETURNING THE RESULT TO THE ORIGINAL PROJECT

23. The final Markdown

Throughout this whole process, we have worked on the copy that exists inside DEMUESTRA.

Our original project remains separate.

When we are satisfied with the result, we recover the corrected Markdown:

```
DEMUESTRA/proyectos/MI_TEORIA/demostracion.md
```

or the corresponding file kept by DEMUESTRA after the evaluation.

We copy it back to:

```
MI_TEORIA/demostracion.md
```

Now our original project includes all the corrections that arose during the analysis and formalization process.

PART XIII - WHAT LEAN PROVES AND WHAT IT DOES NOT PROVE

24. What it means for Lean to compile

This point must be understood correctly.

If Lean accepts the formalization, we have a mechanical verification that the formalized results are correctly derived from the definitions, axioms, hypotheses, and theorems used inside that formalization.

This is much stronger than a simple style review or an AI opinion.

Lean acts as a **formal verifier**.

---

25. What it does not mean

The fact that a Lean file compiles does not automatically prove that all the premises used correctly describe nature.

For example, a physical theory may have an internally correct mathematical chain and contain a physical hypothesis that must later be tested experimentally.

Nor does it mean that an incomplete formalization validates what has not been formalized.

That is why it is essential for the Lean file to truly represent all the necessary propositions of the proof.

PART XIV - DEMUESTRA'S PHILOSOPHY

26. The objective

DEMUESTRA does not aim to replace the researcher.

Nor does it aim to replace scientific review or experimental verification when that is necessary.

Its purpose is to provide something different:

an explicit, reproducible mathematical chain that can be formally verified.

The AI can help us translate.

The analyzer can help us discover deficiencies in the exposition.

Lean can verify the formalized mathematical relationships.

But the idea, the hypotheses, and the responsibility for the proof continue to belong to the researcher.

The working principle can be summarized as follows:

```
 HUMAN IDEA
      v
HUMAN PROOF
      v
MARKDOWN
      v
STRUCTURAL ANALYSIS
      v
CORRECTION
      v
LEAN FORMALIZATION
      v
LEAN 4 VERIFICATION
      v
REPRODUCIBLE RESULT
```

DEMUESTRA does not ask for a proof to be believed.

It allows it to be examined step by step.