**Logical Proof Tool – User Manual**

**table of contents:**

1. [**General information**](#GeneralInfo) **2**
2. [**Interface use**](#InterfaceUse) **3-5**
3. **Rules 6-31**
   1. Assumption **6**
   2. Data 7
   3. Proven I 8
   4. Proven e 9
   5. Law of Excluded Middle (“LEM”) 10
   6. Proof By Contradiction (“PBC”) 11
   7. Modus Ponens (“MP”) 12
   8. Modus Tollens (“MT”) 13
   9. Copy 14
   10. And Introduction (“∧i“ ) 15
   11. And Elimination (“∧e1”, “∧e2”) 16
   12. Or Introduction (“∨i1”, “∨i2”) 17
   13. Or Elimination (“∨e”) 18
   14. Double Negative Introduction (“¬¬i") 19
   15. Double Negative Elimination (“¬¬e”) 20
   16. → introduction (“→i”) 21
   17. Contradiction Elimination (“⊥e”) 22
   18. Negation Introduction (“¬i”) 23
   19. Negation Elimination (“¬e”) 24

**Predicates**

* 1. Variable Introduction (“X0/Y0 i”) 25
  2. Equal Introduction (“=i”) 26
  3. Equal Elimination (“=e”) 27
  4. All Introduction (“∀x i”, “∀y i”) 28
  5. All Elimination (“∀x e”, “∀y e”) 29
  6. Exist Introduction (“∃x i”, “∃y i”) 30
  7. Exist Elimination (“∃x e”,” ∃y e”) 31

1. **General information:**

**About the tool:**

This is a tool that was made in order to help people to write logical proofs. It **does not** solve logical expressions.

This tool consists of two parts: Logical Proof, and Text Editor. Many usage examples are present in the example folder. The tool also allows you to, save and load to/from word documents.

**Logical Proof Usage:**

Using the Logical Proof requires entering a valid logical expression at the text box at the top of the page, which will be referred as the main logical expression. It must contain exactly one ⊢ symbol between two valid logical expressions, the one to the right is the expression you want to prove. Some examples for a valid main expressions:

p^q ⊢ q

P(t), ∀x(P(x)→¬Q(x)) ⊢ ¬Q(t)

⊢ (q→r) → ((¬q→¬p) → (p→r))

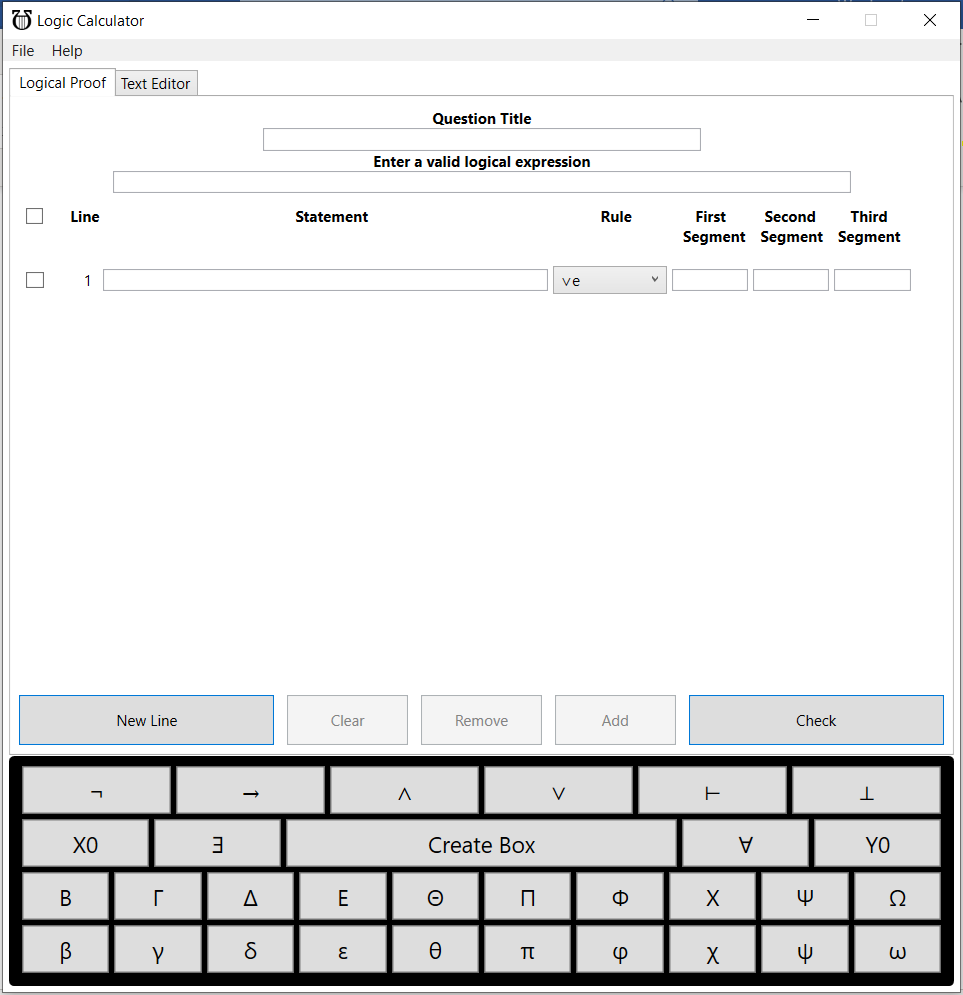
**Text Editor Usage:**

The Text Editor is designed mainly in order for writing logical statements. The tool’s keyboard has Greek letters and logical symbols that can help writing proofs more easily than any other text editor.

Saving from text editor will copy all the textbox content that has been written into it, to a word document.

The Text Editor is more suited for **English** use.

1. **Interface use**



\*Note: Ctrl + Click on every hyper link over picture will jump to correct page.

* **Keyboard**

Clicking any button but the "Create Box" button will put the clicked symbol wherever the cursor is currently placed, in both the Text Editor and the Logical Proofs.

* + “Create Box” – available **only** in the Logical proofs tab, requires checking 2 lines that the box should surround. The upper closer of the box will be placed **above the upper** checked line, and the bottom closer will be placed **under the lower** checked line
* **New Line**

Adds new line at bottom of the table.

* **Clear**

Remove all text inputs in the selected lines, must check at least one line.

* **Remove**

Delete all selected lines, must check at least one line.

* **Add**

Add **one** new line **above** each selected line, need to check at least one line.

* **Check**

Checks if the proof is valid, and if the thing that was needed to be proven has been achieved.

User will get notification if something is wrong in his proof and the tool will point to the place the problem takes place and the reason.

* **Master Checkbox**

Check/Uncheck all lines in proof, if checked, tool will open buttons for “check mode” - “Clear”, “Remove” and “Add”, if unchecked tool will go back to “insert mode”.

* **Checkbox**

Check specific line, when used, tool will open buttons for “check mode” - “Clear”, “Remove” and “Add”

* **Rule Picker**

Drop down all available rules

* **Main expression**

The logical proof expression should be written inside that text box, must be filled in order to use the Logic Proofs tool. Must be valid logical expression

* **Question Title**

Will be the written as title in word file when proof saved, **optional.**

* **Tab Picker**

Switch tabs between “Logical Proof” and “Text Editor” by clicking on the corresponding tab. When saving, the currently picked tab's content will be saved.

* **Menu bar**
  + **File:**
    - **New**

Clear all current input in both tabs.

* + - **Open**

Loads the content of a word document to the tool. If the file a Logical Proof made by this tool it will load to “Logical Proof”. Otherwise, it will load to the “Text Editor”.

* + - **Save**

Opens the save to word document dialog.

* + - **Exit**

Closes the software (same as clicking the X in the top right of the screen).

* + **Help:**
    - **User Manual**

Opens the user manual (which you are reading).

* + - **About**

Show copyrights.

* **Minimize**

Minimize program (no data will be lost).

1. **Rules**

**Assumption**

* **General info:**

Assumption can be any logical expression or variable.

* **Limitation:**

Assumption must start a box, tool will verify for box opener above any line with “Assumption” in Rule Picker.

* **Tool Example:**



**Data**

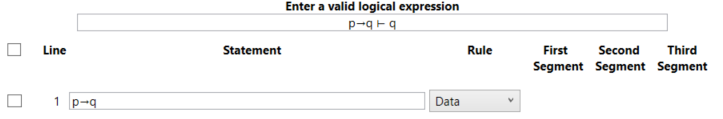
* **General info:**

Use for data that was declared in the main expression.

* **Limitation:**

Data declaration is valid only if the expression is written in main expression as data.

* **Tool Example:**



**Proven I**

* **General info:**

Use to declare Proof already been proofed before that known as fact.

Tool will **not verify** the input in statement (Only syntax check, not logical) and will refer to it as known fact.

This one

* **Limitation:**

Same syntax verified like in main expression, must be valid logical expression.

This rule **not supported over predicates proofs.**

* **Tool Example:**

**Proven e**

* **General info:**

Used when all the data needed in “Proven i” rule achieved during proof progress,

This rule is generic which means the data letters used in “Proven i” can be different than letters used in Proof.

**Example** (see photo below):

in this case “Proven i” had 2 data expressions: “δ∧β”, “γ” and we already proofed that if we have those data it provides “γ→β”.

In our current Proof we have “r”, “(p∨r)∧q”.

In careful look we can see we have the same data as in “Proven i”,

“(p∨r)∧q” == “δ∧β” 🡺 “p∨r” == “δ”, “q” == “β”;

“r” == “γ”

And we already know from “Proven i” that mean we have “γ→β” in our case:

“r→q”, we get it by using “Proven e”.

* **Segments use:**

First segment – “Proven i” line number.

Second segment – lines contain the needed data declared in “Proven i” line **separated by “,”**

Order of referred lines in second segment **must be in the same order** as “Proven i” data.

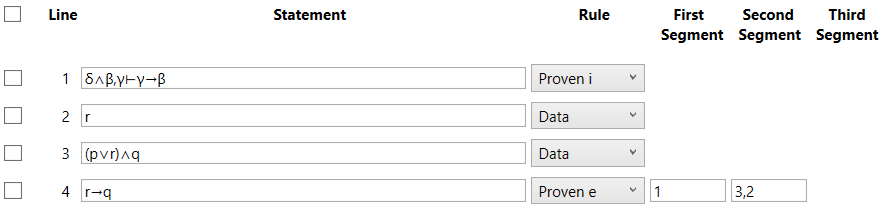
In this example line 3 🡺 (p∨r)∧q == δ∧β

And then line 2 🡺 r == γ.

* **Limitation:**

**Not supported in predicates.**

* **Tool Example:**



**Law of Excluded Middle (“LEM”)**

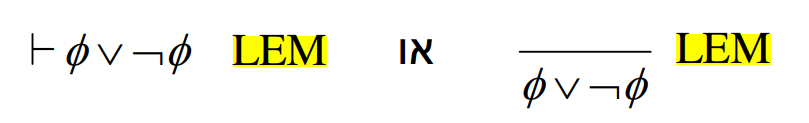
* **General info:**

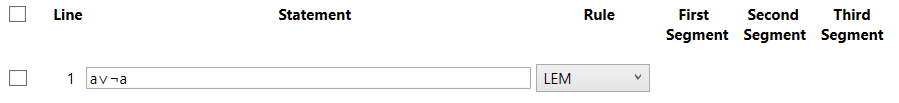
Argument must be true or false, there is no third option.

* **Limitation:**

None

* **Presentation example:**



* **Tool Example:**

**Proof By Contradiction (“PBC”)**

* **General info:**

Using when we had Assumption and got contradiction.

If we got contradiction from negative assumption it means the non-negative is true.

* **Segments use:**

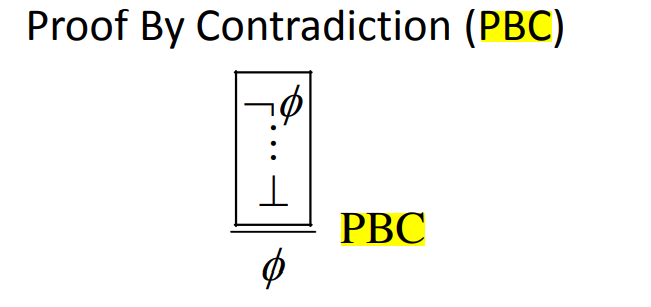
First segment – negative assumption line number (3) till contradiction line (7) separate with **“-”**

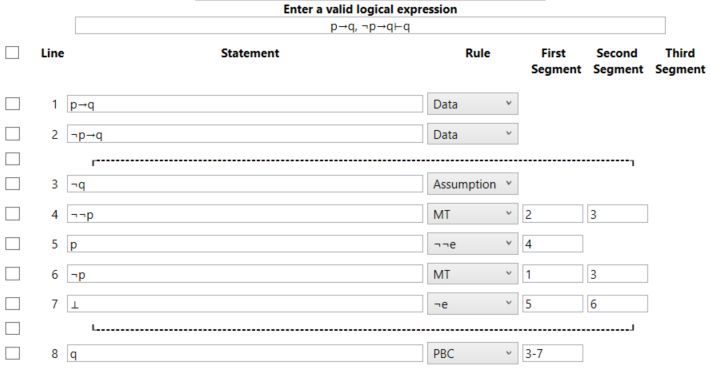
* **Limitation**:

First row rule picker given in First segment must be “Assumption”.

Last row statement given in First segment must be “⊥”.

* **Presentation example:**



* **Tool Example:**

**Modus Ponens (“MP”)**

* **General info:**

If we know “φ→ψ” and we have “φ” we can conclude “ψ”

* **Segments use:**

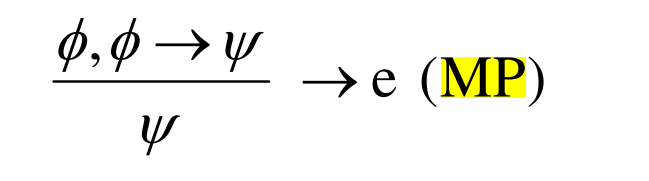
First segment – line number of “φ”

Second segment – line number of “φ→ψ”

* **Limitation:**

None

* Presentation example:



* **Tool Example:**

**Modus Tollens (“MT”)**

* **General info:**

If we know “φ→ψ” and we don’t have “ψ” we can conclude we don’t have “φ”

* **Segments use:**

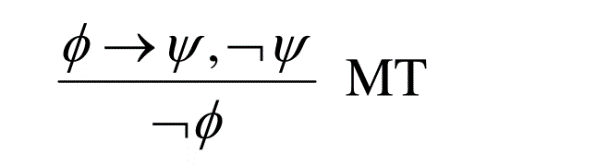
First segment – line number of “φ→ψ”

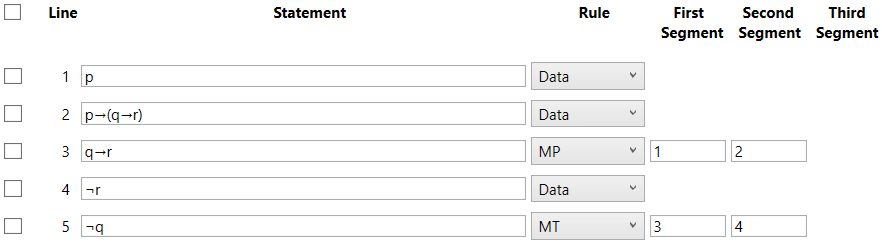
Second segment – line number of “¬ ψ”

* **Limitation:**

None

* **Presentation example:**



* **Tool Example:**

**Copy**

* General info:

Copy statement to current line.

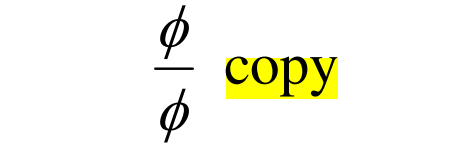
* Segments use:

First segment – line number of statement first instance

* Limitation:

Can use copy only for lines that are not in temporary assumption that already closed.

* Presentation example:



* ****Tool Example:

**And Introduction (“∧i“)**

* **General info:**

Combine two already known variables.

* **Segments use:**

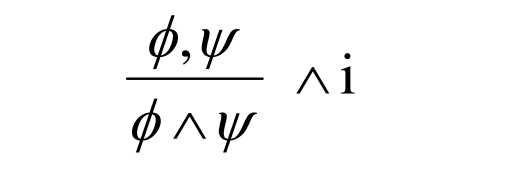
First segment – line number of first variable

Second segment – line number of second variable

* **Limitation:**

None

* **Presentation example:**



* **Tool example:**

****

**And Elimination (“∧e1”, “∧e2”)**

* **General info:**

Use to separate to variable connected with “∧”.

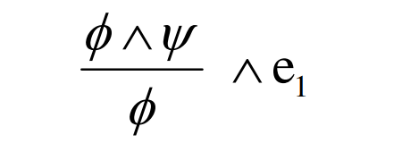
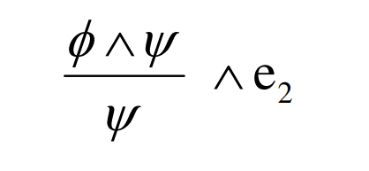
* **Segments use:**

First segment – number of line statement contains “∧” between two variables or expressions.

* **Limitation:**

None

* **Presentation example:**

****

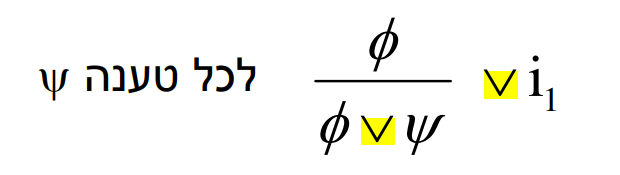
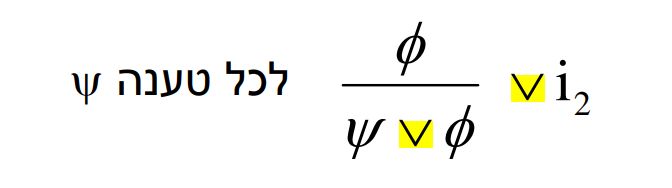
* **Tool Example:**

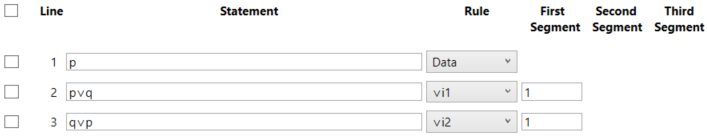
**Or Introduction (“∨i1”, “∨i2”)**

* **General info:**

if “”

* **Segments use:**
* **Limitation:**
* **Presentation example:**

****

* **Tool Example:**

Or Elimination (“∨e”) 18

1. Double Negative Introduction (“¬¬i") 19
2. Double Negative Elimination (“¬¬e”) 20
3. → introduction (“→i”) 21
4. Contradiction Elimination (“⊥e”) 22
5. Negation Introduction (“¬i”) 23
6. Negation Elimination (“¬e”) 24

**Predicates**

1. Variable Introduction (“X0/Y0 i”) 25
2. Equal Introduction (“=i”) 26
3. Equal Elimination (“=e”) 27
4. All Introduction (“∀x i”, “∀y i”) 28
5. All Elimination (“∀x e”, “∀y e”) 29
6. Exist Introduction (“∃x i”, “∃y i”) 30
7. Exist Elimination (“∃x e”,” ∃y e”)

* **General info:**
* **Segments use:**
* **Limitation:**
* **Presentation example:**
* **Tool Example:**