

# txt2tex Reference

Write math like you're at a whiteboard. Get L<sup>A</sup>T<sub>E</sub>X you can hand in.

<https://github.com/jmf-pobox/txt2tex>

## Document Structure

=== Title ===	<code>\section *{Title}</code>
* Solution N **	Bold heading with spacing
(a) Text	Part label
TEXT: paragraph	Prose block (smart formula detection)
LATEX: <code>\cmd</code>	Raw L <sup>A</sup> T <sub>E</sub> X passthrough
PAGEBREAK:	Page break
TITLE: ...	Document title
AUTHOR: ...	Author name
DATE: ...	Date string
BIBLIOGRAPHY: f.bib	BibTeX file
[cite key]	<code>\citep {key}</code>

## Logic

<code>p land q</code>	$p \wedge q$
<code>p lor q</code>	$p \vee q$
<code>lnot p</code>	$\neg p$
<code>p =&gt; q</code>	$p \Rightarrow q$
<code>p &lt;=&gt; q</code>	$p \Leftrightarrow q$
<code>forall x : S   P</code>	$\forall x : S \bullet P$
<code>exists x : S   P</code>	$\exists x : S \bullet P$
<code>exists_1 x : S   P</code>	$\exists_1 x : S \bullet P$
<code>lambda x : S   E</code>	$\lambda x : S \bullet E$
<code>mu x : S   P</code>	$\mu x : S \bullet P$

**Note:** Use `land`, `lor`, `lnot`. English and/or/not are not supported.

## Sets & Types

<code>x elem S</code>	$x \in S$
<code>x notin S</code>	$x \notin S$
<code>A subset B</code>	$A \subseteq B$
<code>A union B</code>	$A \cup B$
<code>A inter B</code>	$A \cap B$
<code>A setminus B</code>	$A \setminus B$
<code>power S</code>	$\mathcal{P} S$
<code>F S</code>	$\mathcal{F} S$
<code>{x : S   P}</code>	Set comprehension
<code>A cross B</code>	$A \times B$
<code>N / Z / N1</code>	$\mathbb{N} / \mathbb{Z} / \mathbb{N}_1$
<code>n..m</code>	$n \dots m$

## Relations

<code>A &lt;-&gt; B</code>	$A \leftrightarrow B$
<code>x  -&gt; y</code>	$(x \mapsto y)$
<code>dom R / ran R</code>	$\text{dom } R / \text{ran } R$
<code>A dres R</code>	$A \triangleleft R$
<code>A dsub R</code>	$A \trianglelefteq R$
<code>R rres B</code>	$R \triangleright B$
<code>R rsub B</code>	$R \trianglerighteq B$
<code>R oplus S</code>	$R \oplus S$ (override)
<code>R o9 S</code>	$R \circ S$ (forward comp.)
<code>R comp S</code>	$R \circ S$ (backward comp.)

## Functions

<code>A pfun B</code>	$A \mapsto B$
<code>A fun B</code>	$A \rightarrow B$ (total)
<code>A pinj B</code>	$A \mapsto\!\!\!\rightarrow B$
<code>A inj B</code>	$A \hookrightarrow B$
<code>A surj B</code>	$A \twoheadrightarrow B$
<code>A bij B</code>	$A \xrightarrow{\sim} B$
<code>A ffun B</code>	$A \rightrightarrows B$ (finite)
<code>f(x)</code>	Function application

## Sequences

<code>&lt;&gt; / &lt;a, b&gt;</code>	$\langle \rangle / \langle a, b \rangle$
<code>s ^ t</code>	$s \hat{\ } t$ (concatenation)
<code>seq S / seq1 S</code>	$\text{seq } S / \text{seq}_1 S$
<code># s</code>	$\#s$ (length)

## Z Notation Blocks

<code>given A, B</code>	Given types: $[A, B]$
<code>T ::= a   b</code>	Free type
<code>name == expr</code>	Abbreviation
<code>axdef / schema Name / gendef [X]</code>	
<code>declarations</code>	Variable declarations
<code>where</code>	Separator
<code>predicates</code>	Constraining predicates
<code>end</code>	Close the block

## Proof Tree Syntax

<code>indent</code>	Premises indented under conclusion
<code>[rule]</code>	Justification label
<code>[rule [n]]</code>	Discharge assumption $n$
<code>[n] expr</code>	Boxed assumption labelled $n$
<code>::</code>	Sibling premises

Rules: `land intro`, `land elim 1/2`, `lor intro 1/2`, `lor elim`, `=> intro/elim`, `<=> intro/elim`, `lnot intro/elim`, `forall intro/elim`, `exists intro/elim`.

## Usage

<code>txt2tex f.txt</code>	$\rightarrow$ f.tex + f.pdf
<code>txt2tex f.txt --tex-only</code>	$\rightarrow$ f.tex only
<code>txt2tex -i</code>	Interactive REPL
<code>txt2tex --check-env</code>	Verify dependencies

# txt2tex Proofs — By Example

Each example shows what you type and what txt2tex renders.

## How Proof Trees Work

In txt2tex, proof trees are written **bottom-up**: the conclusion is the first line, and its premises are indented below it. Think of it as reading the inference rule from conclusion to justification:

conclusion	First line (least indented)
premise	Indented = supports the line above
[rule]	Justification in brackets
[n] expr	Boxed assumption (labelled $n$ )
[rule [n]]	Discharge assumption $n$
::	Marks sibling premises

Deeper indentation = deeper in the tree. Two premises at the same indent level are siblings that jointly support their parent.

## Truth Table Prop. Logic

**You write:**

```
TRUTH TABLE:
p | q | p => q
T | T | T
T | T | T
T | F | F
F | T | T
F | F | T
```

**You get:**

$p$	$q$	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

## Equivalence Chain Prop. Logic

**You write:**

```
EQUIV:
lnot (p land q)
lnot p lor lnot q [De Morgan]
```

**You get:**

$$\neg (p \wedge q) \\ \Leftrightarrow \neg p \vee \neg q \quad [\text{De Morgan}]$$

**EQUIV:** joins steps with  $\Leftrightarrow$ . Use for *predicate* equivalences.

## Modus Ponens Proof Trees

**You write:**

```
PROOF:
  q [=> elim]
  p => q
  p
```

**You get:**

$$\frac{p \Rightarrow q \quad p}{q} \Rightarrow \text{-elim}$$

## $\wedge$ -intro Proof Trees

**You write:**

```
PROOF:
  p land q [land intro]
  p
  q
```

**You get:**

$$\frac{p \quad q}{p \wedge q} \wedge \text{-intro}$$

## $\Rightarrow$ -intro with discharge Proof Trees

**You write:**

```
PROOF:
  p => p [=> intro [1]]
  [1] p
```

**You get:**

$$\frac{\lceil p \rceil^{[1]}}{p \Rightarrow p} \Rightarrow \text{-intro}^{[1]}$$

[1] p = boxed assumption. [=> intro [1]] discharges it.

## $\vee$ -elim / case analysis Proof Trees

**You write:**

```
PROOF:
  r [lor elim]
  p lor q
  :: p => r
    [1] p
    r [from p]
  :: q => r
    [2] q
    r [from q]
```

**You get:**

$$\frac{p \vee q \quad \frac{\lceil p \rceil^{[1]}}{r} \quad \frac{\lceil q \rceil^{[2]}}{r}}{r} \vee \text{-elim}$$

:: = sibling premises (case arms).

## Equality Chain Induction

**You write:**

```
EQUAL:
0 + 0
0 [0+0 = 0]
length(<>)
[length(<>) = 0]
```

**You get:**

$$0 + 0 \\ = 0 \quad [0+0 = 0] \\ = \text{length}(\langle \rangle) \quad [\text{length}(\langle \rangle) = 0]$$

**EQUAL:** = expression chains (=). Use for N/sets/sequences.