

# The propositions package\*

Cian Dorr (with help from Claude Code)  
ciandorr@gmail.com

2026/02/13

## Abstract

The **propositions** package provides a key-value driven system for labelling propositions, theses, and premises in academic papers. Items may be given names like ‘(P)’ or ‘Physicalism’, or auto-numbered using different counters; all carry robust cross-references with configurable formatting. The package integrates with **amsmath**, **hyperref**, and **cleveref**.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>History</b>	<b>2</b>
<b>3</b>	<b>Basic usage</b>	<b>2</b>
<b>4</b>	<b>Advanced usage</b>	<b>3</b>
<b>5</b>	<b>The prop environment and \pitem</b>	<b>4</b>
<b>6</b>	<b>Item types</b>	<b>6</b>
6.1	Built-in types . . . . .	7
<b>7</b>	<b>Cross-referencing</b>	<b>7</b>
7.1	How it works: \propapply . . . . .	9
<b>8</b>	<b>Package options</b>	<b>10</b>
8.1	List dimensions . . . . .	10
8.2	Default types . . . . .	11
8.3	Formatting and referencing equation numbers . . . . .	11
<b>9</b>	<b>Compatibility</b>	<b>12</b>
<b>10</b>	<b>Known issues</b>	<b>12</b>

---

\*This document describes version v0.2, dated 2026/02/13.

## 1 Introduction

In some academic disciplines (such as philosophy), it is common to have displayed propositions (examples, theses, premises, . . . ) with various kinds of labels. A thesis might be referred to as ‘(P)’ or ‘Physicalism’; the premises of an argument might be numbered as ‘P1’, ‘P2’, ‘P3’, . . . ; or examples might be numbered consecutively over the course of a whole article. Standard L<sup>A</sup>T<sub>E</sub>X environments like `enumerate` can handle some of these cases, but cross-referencing is awkward: `\ref` produces a bare number or letter, and the author must manually add parentheses or other formatting at every point of reference. The standard `description` environment does not allow cross-referencing at all.

The `propositions` package solves this by attaching formatting information to each label. A short item like `\pitem[P]` is displayed as “(P)” and `\ref` automatically produces “(P)” as well—complete with parentheses, hyperlinks, and `cleveref` support. The full key-value interface supports named items, numbered items, custom counters, glosses, shorthands, and per-item format overrides.

The `\ptag` command (which requires `amsmath`) extends this to displayed math environments: an equation can be tagged with a proposition label instead of (or using) its equation number.

## 2 History

I wrote the ancestor to this package in the 90s while finishing my Ph.D. thesis, but never documented it or shared it with the world. This new version is a thorough re-implementation in L<sup>A</sup>T<sub>E</sub>X3, written in 2026 with extensive help from Claude Code. I hope others will find it as useful as I have.

## 3 Basic usage

Load the package with `\usepackage{propositions}` or `\usepackage[options]{propositions}` (see [section 8](#) below for valid package options).

The `prop` environment generates a list of propositions, each introduced by a `\pitem`. `\pitem` with an optional argument gives a `description`-like label:

<code>\begin{prop}</code>	
<code>\pitem[Physicalism] Everything is</code>	<b>Physicalism</b> Everything is physi-
<code>physical. \label{phys}</code>	cal.
<code>\pitem[Idealism] Everything is</code>	<b>Idealism</b> Everything is mental.
<code>mental. \label{ideal}</code>	
<code>\end{prop}</code>	

Unlike the standard `description` environment, one can refer back to these propositions using the standard `\ref` command:

<code>\ref{phys}</code> is more plausible	<b>Physicalism</b> is more plausible than
than <code>\ref{ideal}</code> .	<b>Idealism</b> .

With no optional argument, `\pitem` will by default generate numbered items similar to `enumerate`, but with numbering that persists across the document:

<pre> \begin{prop}   \pitem Every atom is physical.   \label{atoms} \end{prop} \ref{phys} follows from the conjunction of \ref{atoms} and \begin{prop}   \pitem Everything is an atom.   \label{atomism} \end{prop&gt; </pre>	<p>(1) Every atom is physical.</p> <p><b>Physicalism</b> follows from the conjunction of (1) and</p> <p>(2) Everything is an atom.</p>
---	--

As with `enumerate`, the counter and formatting depend on the nesting level:

<pre> \begin{prop}   \pitem \label{dual}   \begin{prop}     \pitem Some things are physical.     \label{some}     \pitem Some things are not     physical. \label{notall}   \end{prop} \end{prop} Without \ref{some}, \ref{dual} would be consistent with \ref{ideal}. </pre>	<p>(3) a. Some things are physical.</p> <p>b. Some things are not physical.</p> <p>Without (3a), (3) would be consistent with <b>Idealism</b>.</p>
---	--

## 4 Advanced usage

The format of the proposition labels, and of subsequent references, are both configurable using a key=value syntax (see section 5 for the possible keys):

<pre> \begin{prop}   \pitem[No Overlap,     align=flush,     display format=\textbf{#1},     ref format=\textit{#1}]     Nothing mental is     physical. \label{incomp} \end{prop} Is \ref{dual} consistent with \ref{phys}? </pre>	<p><b>No Overlap</b> Nothing mental is physical.</p> <p>Is (3) consistent with <b>Physicalism</b>?</p>
---	--

Preset *item types* can be declared and used instead of configuring the `\pitems` one at a time:

<pre> \begin{prop}   \pitem[Nihilism,     type=long]     There is nothing.     \label{nihilism} \end{prop} Does \ref{nihilism} imply \ref{phys}, \ref{dual}, or both? Discuss. </pre>	<p><b>Nihilism</b> There is nothing.</p> <p>Does <b>Nihilism</b> imply <b>Physicalism</b>, (3), or both? Discuss.</p>
---	---

Shortcut commands—e.g., `\litem[⟨keys⟩]` for `\pitem[type=long, ⟨keys⟩]`—can also be defined. The package loads with a range of predefined types.

When the package is loaded with `\usepackage[equations]{propositions}`, first-level `\pitems` use the same counter as equations. (This looks better with the `leqno` option to `\documentclass`.)

<pre>\begin{equation}   \exists x (\text{Mental}(x)   \wedge \text{Physical}(x)) \end{equation} \begin{prop}   \pitem   There is overlap between   the mental and the physical. \end{prop}</pre>	$(4) \quad \exists x(\text{Mental}(x) \wedge \text{Physical}(x))$ <p>(5) There is overlap between the mental and the physical.</p>
--	--

The `\ptag` command (requires `amsmath`) is an analogue of `\pitem` that works inside displayed math environments.

<pre>\begin{equation}   \ptag[Monism] \label{mon}   \exists x \forall y (y = x) \end{equation} Is \ref{mon} compatible with \ref{dual}?</pre>	$\textbf{Monism} \quad \exists x \forall y (y = x)$ <p>Is <b>Monism</b> compatible with (3)?</p>
---	--

## 5 The prop environment and \pitem

```
\begin{prop}[<keys>]
  <environment content>
\end{prop}
```

Creates a displayed list of propositions. It is a standard L<sup>A</sup>T<sub>E</sub>X list, so by default its formatting will depend on the standard length parameters like `\itemsep` and `\topsep`, although these can be overridden by setting package keys.

Within `prop`, `\pitem` (see below) creates labelled items. The ordinary `\item` command is still available for unlabelled items.

The optional `<keys>` argument accepts the same keys as `\proppoptions`<sup>P. 10</sup> (section 8), with effects local to this environment.

```
\begin{inlineprop}[<keys>]
  <environment content>
\end{inlineprop}
```

Like `prop`, but does not create a list. Allows `\pitem` to be used outside list environments, e.g. for generating numbers at the beginning of paragraphs. Steps the `prop` counter and increments the nesting level. Accepts the same optional `<keys>` as `prop`.

```
\pitem[<keys>]
```

Inside the `prop` and `inlineprop` environments, introduces a labelled proposition. The optional argument is a comma-separated list of `<key>=<value>` pairs.

When used without an optional argument (or without setting `name`, `counter`, or `type`), it behaves as `\pitem[type=<type>]`, where `<type>` depends on

the nesting depth. The defaults are `numbered`, `leveltwo`, `levelthree`, `levelfour`, `levelfive`; these can be changed with the `level`  $\langle n \rangle$  keys (section 8).

The following keys can be used in the optional argument of `\pitem`:

`name`= $\langle text \rangle$  (no default)

The proposition's name. A bare string (without =) in the key list is equivalent to `name`= $\langle text \rangle$ .

`type`= $\langle type \rangle$  (no default)

An item type, equivalent to a preset collection of keys. Types can be declared using `\SetItemType`<sup>P.6</sup> or `\DeclareNumberedType`<sup>P.6</sup>, and several come predefined (section 6).

`counter`= $\langle name \rangle$  (no default)

Counter to use. The counter is automatically stepped, and the item's `name` is set to `\the` $\langle name \rangle$ , though this can be overridden by explicitly setting `name`.

`align`= $\langle type \rangle$  (no default)

How the label should be positioned. Possible values: `default` (offset controlled by `\labelwidth` and `\labelsep`), `right` (right-aligned, like `enumerate`), `left` (aligned with left margin of surrounding text), `flush` (aligned with left margin of item text), `nextline` (label on its own line), `left-nextline`, and `flush-nextline`. Has no effect inside `inlineprop` or `\ptag`.

`format`= $\langle template \rangle$  (no default)

Formatting applied to the `name`: use `#1` for the argument, e.g. `format`=`\textbf{(#1)}`. Shorthand for setting both `display format` and `ref format`.

`display format`= $\langle template \rangle$  (no default)

Format for displaying the name in the proposition's label. Does not affect cross-references.

`ref format`= $\langle template \rangle$  (no default)

Format for subsequent cross-references to this proposition. Does not affect the display.

`shorthand`= $\langle text \rangle$  (no default)

An abbreviation displayed after the name. If present, the shorthand becomes the reference text: `\ref` produces the shorthand (formatted with `ref format`) rather than the full name.

`shorthand format`= $\langle template \rangle$  (initially `~[#1]`)

Format for displaying the shorthand in the label.

`gloss`= $\langle text \rangle$  (no default)

A parenthetical gloss displayed after the name. Does not affect cross-references.

`gloss format`= $\langle template \rangle$  (initially `~(#1)`)

Format for displaying the gloss in the label.

`ref=<text>` (no default)

Explicitly set the reference text, overriding what would be derived from `name`, `counter`, or `shorthand`.

`label=<label>` (no default)

Equivalent to a trailing `\label{<label>}`.

`crefname=<type>` (no default)

When `cleveref` is loaded, assigns an arbitrary reference type to this proposition. For example, `crefname=lemma` causes `\cref` to use the names defined by `\crefname{lemma}{...}{...}` instead of the default `proposition` type. The `<type>` must be known to `cleveref`; new types can be declared with `\crefname`.

`\ptag[<keys>]`

Available only when `amsmath` is loaded. Works inside displayed math environments like `equation` and `align`. Accepts the same keys as `\pitem`<sup>P.4</sup>, except that `align` has no effect (positioning is controlled by the tag placement system).

## 6 Item types

`\SetItemType{<name>}{<keys>}`

Defines or modifies an item type for use with the `type` key. All `\pitem`<sup>P.4</sup> keys are accepted, plus the following:

`macro=<command>` (no default)

A new user macro, equivalent to `\pitem[type=<name>]`. Any further keys given to the macro are passed to `\pitem`.

If the type `<name>` already exists, `\SetItemType` modifies or adds keys. For example, `\SetItemType{short}{align=flush}` changes the alignment of the built-in `short` type while preserving its other settings.

<pre> \SetItemType{angle}{   align = left,   display format =     \textbf{\$\angle\$#1\$\rangle\$},   ref format =     \$\angle\$#1\$\rangle\$,   macro = \angitem } \begin{prop}   \angitem[Angle thesis]   Everything is angular. \end{prop} No further discussion of \Lastref{} is needed.</pre>	<p><b>⟨Angle thesis⟩</b> Everything is angular.</p> <p>No further discussion of ⟨Angle thesis⟩ is needed.</p>
---	---

`\DeclareNumberedType{<name>}[<keys>]`

Creates a new L<sup>A</sup>T<sub>E</sub>X counter named `<name>` and a matching item type with `counter=<name>`. All `\SetItemType` keys are accepted, plus:

**parent**= $\langle counter \rangle$  (no default)

A parent counter; the new counter resets when the parent steps (same mechanism as `\numberwithin`). A dedicated **prop** counter (stepped by each **prop** and **inlineprop**) is available for non-persistent numbering.

**counter format**= $\langle format \rangle$  (default  $\langle name \rangle \backslash \text{arabic} \{ \langle name \rangle \}$ )

The representation of the new counter ( $\backslash \text{the} \langle name \rangle$ ).

<code>\DeclareNumberedType{P}</code>	<b>P1</b> First premise.
<code>\begin{prop}</code>	<b>P2</b> Second premise.
<code>\pitem[counter=P] First premise.</code>	From <b>P1</b> and <b>P2</b> ...
<code>\label{p1}</code>	
<code>\pitem[counter=P] Second premise.</code>	
<code>\label{p2}</code>	
<code>\end{prop}</code>	
<code>From \ref{p1} and \ref{p2}\ldots</code>	

## 6.1 Built-in types

The following item types are predefined.  $\backslash \text{SetItemType}^{\rightarrow \text{P.6}}$  can modify their behaviour.

Type	Shortcut	Counter	Display	Ref	Align
short	none*	none	<b>Name</b>	<b>Name</b>	left
long	<code>\litem</code>	none	<b>Name</b>	Name	flush
bullet	<code>\bitem</code>	none	•	•	default
roman	<code>\ritem</code>	roman	(i)	(i)	left
alph	<code>\aitem</code>	alph	(a)	(a)	left
numbered	none <sup>†</sup>	numbered <sup>‡</sup>	(1)	(1)	left
leveltwo	none <sup>†</sup>	leveltwo	a.	(1a)	left
levelthree	none <sup>†</sup>	levelthree	(i)	(1a.i)	left
levelfour	none <sup>†</sup>	levelfour	•	•	default
levelfive	none <sup>†</sup>	levelfive	–	–	default

\* The **short** type is auto-selected when `\pitem` or `\ptag` has an optional argument but no **type** key.

† The **numbered**–**levelfive** types are auto-selected when `\pitem` or `\ptag` has no optional argument, depending on nesting level.

‡ The **equations** package option changes **numbered**’s counter to **equation**. The **leveltwo**–**levelfive** counters reset automatically when a `\pitem` at the next lower level is processed.

## 7 Cross-referencing

Labels placed after `\pitem` items work with the standard `\label/\ref` mechanism. The key difference from ordinary L<sup>A</sup>T<sub>E</sub>X references is that `\ref` produces *formatted* output: for example, `\textbf` might be applied to the name, or the number might be wrapped in parentheses. The formatting is controlled by the **format** key (or separately by **display format** and **ref format**).

`\nref{<label>}`  
`\nref*{<label>}`

“Naked ref.” Outputs the bare reference content with all formatting stripped. If `\ref{premise}` produces ‘(P1)’, then `\nref{premise}` produces ‘P1’. The starred form suppresses the hyperlink.

`\nref` is often useful in the argument of `\pitem`, when the the name of one proposition should depend on that of another:

<code>\SetItemType{short}{format=#1}</code>	(Phys) Everything is physical.
<code>\begin{prop}</code>	(7*) Almost everything is physical.
<code>\pitem[Phys]</code>	
Everything is physical.	
<code>\label{premise}</code>	(7*) The version (7*), which uses
<code>\pitem[\nref{premise}*]</code>	<code>\nref</code> , looks better than the one
Almost everything is physical.	with <code>\ref</code> , unless for some reason
<code>\label{newpremise}</code>	one wants two lots of parentheses.
<code>\pitem[\ref{premise}*]</code>	
The version <code>\ref{newpremise}</code> , which	
uses <code> \nref </code> , looks better	
than the one with <code> \ref </code> , unless	
for some reason one wants two	
lots of parentheses.	
<code>\end{prop}</code>	

*Warning:* documents where the name of one item includes a reference to that of another, and there are further references to that item, will require multiple L<sup>A</sup>T<sub>E</sub>X runs to resolve all references. To save time, it is better to avoid long chains of dependencies of this sort.

`\oref[<prefix>][<suffix>]{<label>}`  
`\oref*[<prefix>][<suffix>]{<label>}`

“Ref with options.” Extends `\ref` by injecting a prefix and/or suffix *inside* the formatting. With one optional argument, `<suffix>` is appended; with two, `<prefix>` is prepended and `<suffix>` appended. For instance, if `\ref{premise}` produces ‘(P1)’, then `\oref[*]{premise}` produces ‘(P1\*)’ and `\oref[cf.~]{*}{premise}` produces ‘(cf.~P1\*)’. The starred form suppresses the hyperlink.

`\oref` can also be useful in the name of `\pitems`, if one wants the display format for the modified item to depend on that originally used

<code>\begin{prop}</code>	7 This will use boldface and
<code>\label{premise}</code>	parentheses because the original
<code>\pitem[name=\oref[*]{premise},</code>	referenced item did.
<code>format=#1]</code>	
This will use boldface and	
parentheses because the original	
referenced item did.	
<code>\end{prop}</code>	

Another handy use for `\oref` is in combination with `\nref` to refer to ranges:

The first two numbered examples in	The first two numbered examples in
this document were	this document were were (1–2).
were <code>\oref[--\nref{atomism}]{atoms}</code> .	



`\Lastref[⟨prefix⟩]{⟨suffix⟩}`

Formatted reference to the most recently processed `\pitem` or `\ptag`, even without a `\label`. Useful for back-references in running text. With one argument, `⟨suffix⟩` is appended; with two, `⟨prefix⟩` is also prepended. Use `\Lastref{}` for a plain reference.

`\nLastref`

Like `\Lastref{}` but returns the bare content without formatting. Takes no arguments; simply output any desired suffix directly afterwards.

`\Parentref[⟨prefix⟩]{⟨suffix⟩}`

Inside a nested `prop` (or `inlineprop`), produces a formatted reference to the most recent item of the enclosing level. Same argument convention as `\Lastref`.

`\nParentref`

Like `\Parentref{}` but returns the bare content without formatting. Takes no arguments; simply output any desired suffix directly afterwards.

`\Parentref` and `\nParentref` are useful for making subitems whose names derive from their parent's:

<pre>\DeclareNumberedType{inner}[   counter format=\alph{inner},   display format=#1.] \begin{prop}   \pitem[OI] Outer item.   \label{outer2}   \begin{prop}     \pitem[type=inner,       format=\Parentref{.#1}]     \label{dsub1}     Ref: \ref{dsub1},     naked: \nref{dsub1}.     \pitem[type=inner,       display format=#1.,       ref format=\Parentref{.#1}]     \label{dsub2}     Ref: \ref{dsub2},     naked: \nref{dsub2}.   \end{prop} \end{prop}</pre>	<p>(OI) Outer item. (OI.a) Ref: (OI.a), naked: a. b. Ref: (OI.b), naked: b.</p>
--	---

The built-in `leveltwo` and `levelthree` types have `ref format=\Parentref{#1}` and `ref format=\Parentref{.#1}`, respectively, so that if the parent references as ‘(P1)’, a `leveltwo` sub-item references as ‘(P1a)’ and `\nref` returns just ‘a’.

## 7.1 How it works: `\propapply`

`\propapply{⟨template⟩}{⟨content⟩}`

Internally, each reference is stored in the `.aux` file as `\propapply{⟨template⟩}{⟨content⟩}`. The `⟨template⟩` contains formatting with the placeholder `\propfmtarg`<sup>→ P. 10</sup>

where `content` appears. At reference time, `\propapply` evaluates the template with `\propfmtarg` bound to  $\langle content \rangle$ . The `\oref`<sup>P.8</sup> and `\nref`<sup>P.8</sup> commands work by locally redefining `\propapply`.

In normal use, you need not interact with `\propapply` directly.

### `\propfmtarg`

Placeholder used inside templates; expands to the content argument of the enclosing `\propapply`<sup>P.9</sup>.

## 8 Package options

### `\propoptions{\keys}`

Sets package-level keys. These can also be set:

- In the optional argument of `prop` and `inlineprop` (local to that environment).
- In the preamble with `\usepackage[\options]{propositions}` (global).

*Exception 1:* keys containing # (such as `equation format`) cannot be set in the optional argument of `\usepackage`, due to how L<sup>A</sup>T<sub>E</sub>X handles # in option values.

*Exception 2:* `equations` is a global key; it cannot be used in the optional argument of `prop` or `inlineprop`.

### 8.1 List dimensions

`topsep`= $\langle length / length list \rangle$   
`partopsep`= $\langle length / length list \rangle$   
`itemsep`= $\langle length / length list \rangle$   
`parsep`= $\langle length / length list \rangle$   
`leftmargin`= $\langle length / length list \rangle$   
`rightmargin`= $\langle length / length list \rangle$   
`labelwidth`= $\langle length / length list \rangle$   
`labelsep`= $\langle length / length list \rangle$   
`itemindent`= $\langle length / length list \rangle$   
`listparindent`= $\langle length / length list \rangle$

Override the standard L<sup>A</sup>T<sub>E</sub>X list dimensions. Accept the same values as `\setlength`, including rubber lengths (e.g. `itemsep=4pt plus 2pt`). Each key may also take a comma-separated list of per-level values: e.g. `leftmargin={2.5em, 0em}`. The first value applies at level 1, the second at level 2, etc. Gaps in the list (e.g. `leftmargin={, 0em}`) cause the class default to be used for that level.

`labelindent`= $\langle length / length list \rangle$  (no default)

Positions label left edges at  $\langle length \rangle$  from the enclosing margin, adjusting `\labelsep` or `\itemindent` as needed. Positive values move rightward, negative leftward.

`tightspacing` (no value)

Sets all vertical spacing to the compact defaults that the standard document classes use for level-three lists (`topsep` and `itemsep` to 2pt with stretch/shrink, `parsep` to 0pt, `partopsep` to 1pt). Individual dimension keys set afterward override.

**nosep** (no value)  
Sets `\topsep`, `\itemsep`, and `\parsep` all to zero.

## 8.2 Default types

**default type**= $\langle type \rangle$  (initially **short**)  
The type used when `\pitem` or `\ptag` is given a name or counter but no explicit type.

**default ptag type**= $\langle type \rangle$  (initially empty)  
If set, overrides **default type** for `\ptag` only.

**level n**= $\langle type \rangle$  (see below)  
Default item type at nesting level  $n$  (1–5) when `\pitem` has no optional argument.

Level	Default type
1	<b>numbered</b>
2	<b>leveltwo</b>
3	<b>levelthree</b>
4	<b>levelfour</b>
5	<b>levelfive</b>

The **leveltwo**–**levelfive** types use special counters **leveltwo**–**levelfive**, which reset automatically when `pitem` is used at lesser nesting levels. Other types can also use these counters.

## 8.3 Formatting and referencing equation numbers

**equations** (no value, **global only**)  
Shares the **equation** counter between `\pitem[type=numbered]` and standard displayed equations. Also installs the equation format hooks (see below), so that `\oref` and `\nref` work with equation labels.

**equation format**= $\langle template \rangle$  (initially (#1))  
Shorthand: sets both **equation display format** and **equation ref format**. Also installs the equation format hooks independently of the **equations** option.

**equation display format**= $\langle template \rangle$  (initially (#1))  
Controls how equation tags appear in the PDF (via `\tagform@`). Use #1 for the number. Does not affect `\ref` output. Also installs the equation format hooks independently of the **equations** option. Locally scoped.

**equation ref format**= $\langle template \rangle$  (initially (#1))  
Controls how `\ref` (and `\oref`, `\nref`) render equation labels. Use #1 for the number. Does not affect the displayed tag. Also installs the equation format hooks independently of the **equations** option. Locally scoped.

## 9 Compatibility

The `propositions` package is designed to work with `hyperref`, `cleveref`, and `amsmath`. `amsmath` is required for `\ptag` and the `equations` option. With `cleveref` loaded, all `\pit` items are assigned to a default `proposition` reference type; the `crefname` key can override this.

Recommended load order:

```
\usepackage{hyperref}  
\usepackage{amsmath} % if using \ptag  
\usepackage{propositions}  
\usepackage{cleveref} % if used
```

## 10 Known issues

When using `\ptag` with a named counter (e.g. `\ptag[counter=P]`) inside an `amsmath` equation environment, `hyperref` may emit warnings of the form:

```
pdfTeX warning: destination with the same  
identifier (name{equation.N}) has been already  
used, duplicate ignored
```

These warnings are harmless and do not affect the correctness of cross-references.