

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

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

*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^AT_EX 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^AT_EX3, 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:

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

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

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

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

```
\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}
```

(1) Every atom is physical.

Physicalism follows from the conjunction of (1) and

(2) Everything is an atom.

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

```
\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}.
```

(3) a. Some things are physical.

b. Some things are not physical.

Without (3a), (3) would be consistent with **Idealism**.

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):

```
\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}?
```

No Overlap Nothing mental is physical.

Is (3) consistent with **Physicalism**?

```
\begin{prop}
  \pitem[Nihilism,
    type=long]
    There is nothing.
  \label{nihilism}
\end{prop}
Does \ref{nihilism} imply
\ref{phys}, \ref{dual}, or
both? Discuss.
```

Nihilism There is nothing.

Does **Nihilism** imply **Physicalism**, (3), or both? Discuss.

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) $\exists x(\text{Mental}(x) \wedge \text{Physical}(x))$ (5) There is overlap between the mental and the physical.
--	---

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>	Monism $\exists x \forall y (y = x)$ Is Monism compatible with (3)?
---	--

5 The prop environment and `\pitem`

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

Creates a displayed list of propositions. It is a standard L^AT_EX 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 `\propositions`^{P. 10} (section 8), with effects local to this environment.

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

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 `inlinetext` 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=⟨text⟩` (no default)

The proposition's name. A bare string (without `=`) in the key list is equivalent to `name=⟨text⟩`.

`type=⟨type⟩` (no default)

An item type, equivalent to a preset collection of keys. Types can be declared using `\SetItemTypeP. 6` or `\DeclareNumberedTypeP. 6`, and several come predefined (section 6).

`counter=⟨name⟩` (no default)

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

`align=⟨type⟩` (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=⟨template⟩` (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=⟨template⟩` (no default)

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

`ref format=⟨template⟩` (no default)

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

`shorthand=⟨text⟩` (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=⟨template⟩` (initially `~[#1]`)

Format for displaying the shorthand in the label.

`gloss=⟨text⟩` (no default)

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

`gloss format=⟨template⟩` (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 `\pitemP.4`, 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 `\pitemP.4` 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.

```
\SetItemType{angle}{
    align = left,                                <Angle thesis> Everything is angular.
    display format =
        \textbf{\$\langle\$#1$\rangle\$},
    ref format =
        \$\langle\$#1$\rangle\$,
    macro = \angitem
}
\begin{prop}
    \angitem[Angle thesis]
    Everything is angular.
\end{prop}
No further discussion of <Angle thesis> is needed.
```

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

Creates a new L^AT_EX counter named `<name>` and a matching item type with `counter=<name>`. All `\SetItemType` keys are accepted, plus:

`parent=<counter>` (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=<format>` (default `<name>\arabic{<name>}`)

The representation of the new counter (`\the<name>`).

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

6.1 Built-in types

The following item types are predefined. `\SetItemType→ P.6` can modify their behaviour.

Type	Shortcut	Counter	Display	Ref	Align
short	none*	none	Name	Name	left
long	<code>\litem</code>	none	Name	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†	numbered‡	(1)	(1)	left
leveltwo	none†	leveltwo	a.	(1a)	left
levelthree	none†	levelthree	(i)	(1a.i)	left
levelfour	none†	levelfour	•	•	default
levelfive	none†	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^AT_EX 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 name of one proposition should depend on that of another:

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

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^AT_EX 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

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

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

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

\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→ P. 10`

where content appears. At reference time, `\propapply` evaluates the template with `\propfmtarg` bound to `(content)`. The `\orefP.8` and `\nrefP.8` 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 `\propapplyP.9`.

8 Package options

`\proptions{<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^AT_EX 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=<length / length list>
partopsep=<length / length list>
itemsep=<length / length list>
parsep=<length / length list>
leftmargin=<length / length list>
rightmargin=<length / length list>
labelwidth=<length / length list>
labelsep=<length / length list>
itemindent=<length / length list>
listparindent=<length / length list>
```

Override the standard L^AT_EX 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=<length / length list>` (no default)

Positions label left edges at `<length>` 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=*<type>* (initially `short`)

The type used when `\pitem` or `\ptag` is given a name or counter but no explicit type.

default ptag type=*<type>* (initially empty)

If set, overrides `default type` for `\ptag` only.

level n=*<type>* (see below)

Default item type at nesting level *n* (1–5) when `\pitem` has no optional argument.

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

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=*<template>* (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=*<template>* (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=*<template>* (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 `\pitems` 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.