# Pergamon

BibLaTeX-inspired bibliography management for Typst https://github.com/alexanderkoller/pergamon

## v0.4.0, 13 October 2025 Alexander Koller

## **Contents**

1 Introduction	
2 Example	
References	3
3 Pergamon styles	4
3.1 Builtin reference style	4
References	4
3.2 Builtin citation styles	5
References	6
References	6
3.3 Implementing custom styles	7
4 Advanced usage	8
4.1 Multiple refsections	8
4.2 Styling the bibliography	9
4.3 Styling individual references	9
4.4 Sorting the bibliography	10
4.5 Styling the citations	10
4.6 Showing the entire bibliography	11
4.7 Continuous numbering	12
4.8 Highlighting references	12
5 Data model	13
5.1 Dates	13
6 Detailed documentation	13
6.1 Reference dictionaries	13
6.2 Main functions	15
6.3 The builtin styles	24
6.4 Utility functions	38
7 Changelog	41

## 1 Introduction

Pergamon is a package for typesetting bibliographies in Typst. It is inspired by BibLaTeX, in that the way in which it typesets bibliographies can be easily customized through Typst code. Like Typst's regular bibliography management model, Pergamon can be configured to use

different styles for typesetting references and citations; unlike it, these styles are all defined through Typst code, rather than CSL.

Pergamon has a number of advantages over the builtin Typst bibliographies:

- Pergamon styles are simply pieces of Typst code and can be easily configured or modified.
- The document can be easily split into different refsections, each of which can have its own bibliography (similar to Alexandria). Unlike in Alexandria, you do not have to manually add bibliography prefixes to your citations.
- Paper titles can be automatically made into hyperlinks as in blinky, but much more flexibly and correctly.
- Bibliographies can be filtered, and bibliography entries programmatically highlighted, which is useful e.g. for CVs.
- References retain nonstandard BibTeX fields (unlike in Hayagriva), making it e.g. possible to split bibliographies based on keywords.

At the same time, Pergamon is very new and has a number of important limitations compared to the builtin system.

- Pergamon currently supports only bibliographies in BibTeX format, not the Hayagriva YAML format.
- Only a handful of styles are supported at this point, in contrast to the large number of available CSL styles. Pergamon comes with implementations of the BibLaTeX styles numeric, alphabetic, and authoryear.
- Pergamon still requires a lot of testing and tweaking.

Pergamon was an ancient Greek city state in Asia Minor. Its library was second only to the Library of Alexandria around 200 BC.

## 2 Example

The following piece of code typesets a bibliography using Pergamon. (You can try out a more complex example yourself: download example.typ from Github; see also the generated PDF.)

```
1 #import "@preview/pergamon:0.4.0": *
3 #let style = format-citation-numeric()
4 #add-bib-resource(read("bibliography.bib"))
6 #refsection(format-citation: style.format-citation)[
7
     ... some text here ...
8
    #cite("bender20: climb nlu")
9
10
     #print-bibliography(
11
          format-reference:
                format-reference(reference-label: style.reference-label),
12
          label-generator: style.label-generator,
13
          sorting: "nyt")
14
15 ]
```

... some text here ... [1]

## References

[1] Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL). 2020.

Figure 1: Example bibliography typeset with Pergamon (numeric citation style).

This will generate the output shown in Figure 1. Let's go through the different parts of this code one by one.

The first relevant function that is called here is add-bib-resource. It parses a BibTeX file and adds all the BibTeX entries to Pergamon's internal list of references. Notice that you have to read the BibTeX file yourself and pass its contents to add-bib-resource as a string. This is because Typst packages can't access files in your working directory for security reasons.

We then create a refsection. A refsection is a section of Typst content that shares a bibliography. Pergamon tracks the citations within each refsection separately and prints only those references that were cited within the current refsection when you call print-bibliography.

The actual citation is generated by the call <code>cite("bender20:\_climb\_nlu")</code>. Pergamon currently does not use the regular Typst citation syntax <code>@bender20:\_climb\_nlu</code> for a number of reasons (see <code>issue #40</code>). Instead you call Pergamon's <code>cite</code> function, with the key of the BibTeX entry as a string (not a Typst label). Note that this is not the same as Typst's builtin <code>cite</code> function, which is overwritten by Pergamon.

Notice that refsection has a parameter format-citation to which we passed style.format-citation in the example. This tells the refsection how to typeset citations – in the example, that #cite("bender20:\_climb\_nlu") should be rendered as "[1]".

The format-citation function is typically defined in a *Pergamon style*, along with a companion function format-reference that specifies how the individual references in the bibliography are rendered. In the example, the style is obtained through a call to format-citation-numeric() in line 4. Observe that it has an opening and closing bracket after the function name. This is because citation and reference formatters can be configured by passing arguments to this function. In the example, we just use the default configuration for the numeric style.

Finally, the example calls print-bibliography to typeset the bibliography itself. This is where you pass the format-reference function that renders the individual references. You can furthermore specify how the references should be ordered in the bibliography through the sorting parameter. In the example, the references are ordered by ascending author name; then ascending publication year; then ascending title. Note also that style.label-generator is passed as an argument to print-bibliography. This function generates internal style-specific information that is used to typeset both references and citations.

All of these functions can take additional parameters that you can use to customize the appearance of the bibliography. See Section 6 for details.

... some text here ... [1]

## References

[1] Emily M. Bender and Alexander Koller (2020). Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Figure 2: Bibliography with the configuration of Section 3.1.

## 3 Pergamon styles

Pergamon is highly configurable with respect to the way that references and citations are rendered. Its defining feature is that the *styles* that control the rendering process are defined as ordinary Typst functions, rather than in CSL.

There are two different types of styles in Pergamon:

- *Reference styles* define how the individual references are typeset in the bibliography.
- *Citation styles* define how citations are typeset in the text.

Obviously, the reference and citation style that is used in a refsection should fit together to avoid confusing the reader.

Pergamon comes with one predefined reference style and three predefined citation styles. We will explain these below, and then we will sketch how to define your own custom styles.

## 3.1 Builtin reference style

The builtin reference style is defined by the format-reference function. This reference style aims to replicate the builtin reference style of BibLaTeX, but note that not all features of BibLaTeX are implemented at this point.

The builtin reference style can currently render the following BibLaTeX entry types:

- @article
- @book
- @incollection
- @inproceedings
- @misc
- @thesis

These are explained in more detail in Section 2.1.1 of the BibLaTeX documentation. BibTeX entries of a different type are typeset as a references in a dummy format which displays the entry type and BibTeX key. The aim is to eventually support all BibLaTeX styles; see issue #1 to track the progress, and feel free to submit pull requests implementing them.

The builtin format-reference function can be customized by passing arguments to it. These arguments are explained in detail in Section 6.3. As one example, the following arguments will change the output of the example above to look like in Figure 2:

```
1 #print-bibliography(
2 format-reference: format-reference()
3    reference-label: style.reference-label,
4    print-date-after-authors: true,
5    format-quotes: it => it
6    ),
7    label-generator: style.label-generator,
8    sorting: "nyt")
```

We passed true for the argument print-date-after-authors. This moved the year from the end of the reference to just after the authors and put it in brackets.

We also passed the function it => it as the format-quotes argument. The builtin reference style surrounds all papers that are contained in bigger collections (in this case, a volume of conference proceedings) in quotes by applying the format-quotes function. By replacing the default format-quotes function with the identity function, we can make the quotes disappear in the output.

Pergamon exploits Typst's ability to pass functions as arguments quite heavily. This makes it cleaner in some ways than BibLaTeX, which is built on top of LaTeX, whose macros are much less flexible.

## 3.2 Builtin citation styles

Pergamon comes with three builtin citation styles: *alphabetic*, *numeric*, and *authoryear*. These replicate the BibLaTeX styles of the same names (see e.g. the examples on Overleaf).

Unlike in Typst's regular bibliography mechanism, you write <code>#cite("key")</code> to insert a citation into your document when you use Pergamon. The exact string that is inserted depends on the citation style you use.

The difference between the three builtin citation styles is illustrated in Figure 1 (numeric), Figure 3 (alphabetic), and Figure 4 (authoryear). *Numeric* and *alphabetic* both create a label for each bibliography entry; in the case of *numeric*, the label is the position in the bibliography, and in the case of *alphabetic*, it is a unique string consisting of the first characters of the author names and the year. In both cases, these labels are displayed next to the references and also used as the string to which #cite(key) expands. By contrast, *authoryear* does not display any labels next to the references; it expands #cite(key) to a string consisting of the last names of the authors and the year.

Some of the citation styles have options that will let you control the appearance of the citations in detail. These are documented in Section 6.3. For instance, to enclose the year in the *authoryear* style in square brackets rather than round ones, you can replace line 4 in the above example with

```
1 #let style = format-citation-authoryear(format-parens: nn(it => [[#it]]))
```

```
... some text here ... [BK20]
```

## References

[BK20] Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

Figure 3: Bibliography with the alphabetic citation style.

```
... some text here ... (Bender and Koller 2020)
```

## References

Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

Figure 4: Bibliography with the authoryear citation style.

#### Citation forms

Each citation style offers you different *citation forms* for presenting your citation. These are documented in Table 1. Citation forms are selected using the optional form argument to the cite function:

```
1 #cite("bender20:_climb_nlu", form: "t")
```

If you do not specify a citation form, the auto citation forms will be used. For your convenience, BibLaTeX defines the functions citet, citep, citen, and citeg, which all just call cite with the respective citation form.

#### Citation options

You can pass extra named arguments to the Pergamon citation commands cite, citet, etc. These arguments will be passed as *options* to the citation formatter. For instance, the following citation will render as "(see Bender et al. 2020, p. 3)":

```
1 #cite("bender20:_climb_nlu", prefix: "see ", suffix: ", p. 3")
```

Currently, the only builtin citation style that supports such arguments is the *authoryear* style. It accepts the prefix and suffix options, as in the example, and will print them just inside the opening and closing parentheses in the p and auto citation styles. You can still pass options to the other citation styles; they will simply ignore them.

## 3.3 Implementing custom styles

Instead of using the builtin styles, you can also define your own Pergamon style – either a reference style or a citation style or both. It is recommended to look at the default styles in bibtypst-styles.typ to get a clearer picture.

Implementing a reference style amounts to defining a Typst function that can be passed as the format-reference argument to print-bibliography. Such a function receives arguments (index, reference) containing the zero-based position of the reference in the bibliography and a reference dictionary. A call to your function should return some content, which will be displayed in the bibliography.

Reference dictionaries are a central data structure of Pergamon. They represent the information contained in a single bibliography entry and are explained in detail in Section 6.1.

Implementing a citation style is a little more involved, because a citation style consists of three different functions:

- The *label generator* is passed as an argument to print-bibliography. It is a function that receives a reference dictionary and the bibliography position as arguments and is expected to return an array of length two. Its first element is the bib entry's *label*; it is stored under the label field of the reference dictionary and can contain any information that your style finds useful. The second element is a string summarizing the contents of the label. It is used to recognize when two bib entries have the same label and therefore need an "extradate" to make it unique, e.g. the letter "a" in a citation string like "Smith et al. (2025a)". It is up to your style to ensure that entries with the same label also have the same string summary.
- The *reference labeler* is passed as an argument to format-reference. It receives a reference dictionary and the bibliography position as arguments and returns content. If this content is not none, it will be typeset as the first column in the bibliography, next to the reference itself. Of the builtin citation styles, the reference labeler of authoryear always returns none (indicating a one-column bibliography layout), and the other two return their respective labels in square brackets. Pergamon assumes that the reference labeler of a citation style either always returns none or never returns none, making for a consistent number of columns.
- The *citation formatter* is passed as an argument to refsection. It is responsible for formatting the actual citations into content that is inserted into the document text. The citation formatter receives three arguments: an array of citation specifications, a citation form (cf. "Citation forms" in Section 3.2), and an options dictionary (cf. "Citation options" in Section 3.2). See the format-citation argument of the refsection function in Section 6.2 for details on the citation specifications.

Note that the label information that the label generator produces will be stored in the label field of the reference dictionary. When the reference labeler and the citation formatter are called, the label information will still be available, allowing you to precompute any information you find useful.

	authoryear	alphabetic	numeric
auto	(Bender and Koller 2020)	[BK20]	[1]
p	(Bender and Koller 2020)	_	_
t	Bender and Koller (2020)	-	-
g	Bender and Koller's (2020)	-	-
name	Bender and Koller	-	-
year	2020	-	_
n	Bender and Koller 2020	BK20	1

Table 1: Citation forms.

## 4 Advanced usage

## 4.1 Multiple refsections

A Pergamon document consists of one or more refsections. Each refsection is a segment of the document that shares a bibliography: Pergamon collects all citations within each refsection and prints them in the refsection's bibliography. This allows you to have multiple bibliographies per document, as in the well-known Alexandria package.

Every refsection in a document has a unique identifier that distinguishes it from the other refsections. These identifiers are prepended to the keys of the bibliography entries in every citation. You still write #cite("key") in your document, with the same key that you also use in your BibTeX file; in a refsection with identifier id, Pergamon automatically converts this into a reference to id-key. The only situation where you will notice that the keys were modified is when a citation is undefined; in this case, Typst will warn you that the reference id-key couldn't be resolved, rather than key.

You can specify the refsection identifier yourself by passing it as the id argument to the refsection function. But typical usage will be to not specify an explicit id argument and let Pergamon assign a unique identifier automatically. In this case, the first refsection in the document will have the identifier none, and the subsequent ones will be names "ref1", "ref2", and so on. When the identifier is none, Pergamon simply uses the key itself as the label, rather than prepending it with an identifier string. For the frequent use case where the document has only one refsection, this will make error messages easier to read.

You may use print-bibliography only in the context of a refsection. If your document has only a single refsection, you can configure it through a document show rule, like this:

```
1 #let style = format-citation-numeric()
2 #show: doc => refsection(format-citation: style.format-citation, doc)
3
4 #add-bib-resource(read("bibs/bibliography.bib"))
5 #cite("bender20:_climb_nlu")
```

Emily M. Bender and **Alexander Koller**. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

Figure 5: Reference with highlighted author.

## 4.2 Styling the bibliography

A Pergamon bibliography is displayed as a grid, with one row per bibliography entry. The grid has one or two columns, depending on whether a first column is needed to display a label or not.

You can style this grid by passing a dictionary in the grid-style argument of print-bibliography. The values in this dictionary will be used to overwrite the default values.

In addition, the individual entries in the bibliography are typeset as paragraphs, one per bib entry. You can style these paragraphs through set par rules, e.g. to give the entries a hanging indentation.

## 4.3 Styling individual references

The default reference style of Pergamon gives you fine-grained control over the way the individual fields of a reference are rendered. A *field formatter* is a function with parameters (value, reference, field, options, style), where field is the name and value is the value of a field in the reference; reference is the entire reference dictionary; options are the options that were passed to the reference style; and style is an optional style specification for the field. The field formatter is expected to return content representing the field's value.

You can override the formatters for specific fields by using the format-fields parameter of format-reference. The argument should be a dictionary that maps field names to field formatters. One use of this mechanism is to highlight specific authors in a reference. For instance, to highlight my name in a reference, I could use the following call:

```
1 #format-reference(
2
     format-fields: (
3
       "author": (dffmt, value, reference, field, options, style) => {
         let formatted-names = value.map(d => {
4
5
           let highlighted = (d.family == "Koller")
           let name = format-name(d, format: "{given} {family}")
6
7
           if highlighted { strong(name) } else { name }
8
         })
9
10
         concatenate-names(formatted-names, maxnames: 999)
11
       }
12
     )
13 )
```

This will produce output as in Figure 5.

Another effect that can be achieved by overriding field formatters is to change the presentation of the volume and number of the journal in which an article appears. Here's how the default presentation "VOL.NUM" can be replaced with "vol. VOL, no. NUM":

```
1 #format-reference(
     volume-number-separator: ", ",
 3
     format-fields: (
 4
       "volume": (dffmt, value, reference, field, options, style) => {
 5
         if reference.entry_type == "article" {
 6
           [vol. #value]
 7
         } else {
           dffmt(value, reference, field, options, style)
 8
 9
         }
       },
10
11
       "number": (dffmt, value, reference, field, options, style) => {
12
         if reference.entry type == "article" {
13
14
           [no. #value]
15
         } else {
           dffmt(value, reference, field, options, style)
16
17
         }
18
       },
     )
19
20 )
```

Note that this implementation makes use of the dffmt argument, which receives the default implementation of the field formatters for volume and number, respectively, so that we can delegate the formatting of the field for references that are not journal articles.

One special case that is not covered by field formatters arises in the case of subtitles. In BibLaTeX, the titles of journals, books, special issues, and multi-volume books can have optional subtitles. If both are specified, Pergamon concatenates the title and subtitle with the subtitlepunct argument, e.g. to "Title: Subtitle". It then applies a formatting function to the entire concatenated title and subtitle; for instance, format-journaltitle defaults to setting the title and subtitle in italics. You can override the default behavior by passing your own functions in these arguments.

## 4.4 Sorting the bibliography

You can furthermore control the order in which references are presented in the bibliography. To this end, you can pass a sorting string in the sorting argument of print-bibliography. See the documentation of this argument in Section 6 for details.

## 4.5 Styling the citations

Citations in Pergamon are link elements, and can be styled using show rules. However, it is not entirely trivial to distinguish a link element that represents a Pergamon citation from any other link element (referring e.g. to a website). Pergamon therefore provides a function if-

citation which will make this distinction for you. The following piece of code typesets all Pergamon citations in blue:

```
1 #show link: it => if-citation(it, value => {
2   set text(fill: blue)
3   it
4 }})
```

The value argument contains metadata about the citation; value.reference is the reference dictionary (see Section 6.1). You can use this information to style citations conditionally. For instance, in order to typeset all citations to my own papers in green and all other citations in blue, I could write:

```
1 #show link: it => if-citation(it, value => {
2   if "Koller" in family-names(value.reference.fields.parsed-author) {
3     set text(fill: green)
4     it
5   } else {
6     set text(fill: blue)
7     it
8 }})
```

## 4.6 Showing the entire bibliography

It is sometimes convenient to display the entire bibliography, and not just those references that were actually cited in the current refsection. You can instruct print-bibliography to do this using the show-all argument:

```
1 #print-bibliography(
2 format-reference: format-reference(),
3 show-all: true
4 )
```

To obtain finer control over the bibliography entries that are shown, you can use the filter argument. This function receives a reference dictionary as its argument and returns true if this reference should be included in the bibliography and false otherwise. For instance, the following call shows all journal articles and nothing else:

```
1 #print-bibliography(
2  format-reference: format-reference(),
3  show-all: true,
4  filter: reference => reference.entry_type == "article"
5 )
```

Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

Figure 6: Highlighting a reference.

## 4.7 Continuous numbering

When you typeset a CV, it is sometimes useful to have separate bibliographies for journal articles, papers in conference proceedings, and so on. In this case, you might want to use the *numeric* citation style to number all your references, and you might want to continue counting the papers across the different bibliographies; if you have seven journal papers, you want the first conference paper to be "[8]".

You can achieve this using the resume-after parameter of print-bibliography. If you pass the number 7 for this parameter, the first entry in the bibliography will be labeled "[8]".

It would be desirable to automatically keep a running count of the bibliography entries, so the arguments for resume-after can be calculated automatically. This should be doable using counters or states, but I have not managed to figure out how to do it in a stable way. The function count-bib-entries is intended for this use. If you find out how to do it, please let me know!

## 4.8 Highlighting references

The builtin format-reference function accepts a parameter highlight, which you can use to highlight individual references in the bibliography. The highlight function takes a rendered-reference as argument; this is a piece of content representing the entire rendered bibliography entry, just before it is printed. It also receives the corresponding reference dictionary and the position in the bibliography.

Let's say that you use the keywords field in your BibTeX entries to contain the keyword highlight if you want to highlight the paper. You can then selectively highlight references like this:

```
1 #let f-r = format-reference(
2 highlight: (rendered, reference, index) => {
3   if "highlight" in reference.fields.at("keywords", default: ()) {
4      [#text(size: 8pt)[#emoji.star.box] #rendered]
5   } else {
6      rendered
7 }})
```

This will place a marker before each reference with the "highlight" keyword, and will leave all other references unchanged.

## 5 Data model

Pergamon makes a number of assumptions on the contents of the BibTeX entries. These are typically consistent with those that BibLaTeX makes (see Chapter 2 "Database Guide" in the BibLaTeX documentation), but some of them are worth discussing.

#### 5.1 Dates

Dates can occur in a number of places in the BibTeX entry. The most important one is the publication date of the reference. It can be specified in one of two ways:

- In the date field, using the ISO 8601-2 format: "YYYY-MM-DD" or "YYYY-MM" or "YYYY". In this format, years, months, and days must all be positive integers.
- In the year and month fields. In this format, the value of year should be a positive integer. The month field should contain a positive integer (1–12), full English month names like january, or three-letter abbreviations of English month names like feb. Dates specified in this way will be potentially localized using bibstring (see Section 6.4). As a fallback option, you can also specify some other string for the month, which will be printed in the reference verbatim.

All other BibTeX fields whose name ends in date (e.g. urldate) will also be parsed as in the date option described above.

The parsed dates will be stored in the reference dictionary reference under reference.fields.parsed-X, where X is the name of the date field. (If the publication date is specified with a year field, its value will still be stored in reference.fields.parsed-date.) This makes them available for other functions and styles. A parsed date is represented as a dictionary with keys year, month, and day, all of which may be missing. The values under these fields are all positive integers. If a date specification in the BibTeX entry cannot be parsed, it will be represented as an empty dictionary.

In addition to these basic date specifications, BibLaTeX allows for date ranges (issue #55, approximate dates (issue #56), and years before the Common Era (issue #57). These features are not yet supported in Pergamon.

## 6 Detailed documentation

Let's now go through the detailed documentation of all the functions and data structures that Pergamon exposes to you.

## 6.1 Reference dictionaries

The central data structure that Pergamon manages is the *reference dictionary*. This is a dictionary as shown in Figure 7; its purpose is to represent all information about a single bib entry. It is obtained by parsing a BibTeX file using citegeist and then enriching it with some extra information from within Pergamon.

```
1 (
 2
     entry type: "inproceedings",
     entry key: "bender20: climb nlu",
 3
 4
     fields: (
 5
       author: "Emily M. Bender and Alexander Koller",
 6
       award: "Best theme paper",
 7
       booktitle: "Proceedings of the 58th Annual Meeting of the Association
 8
                    for Computational Linguistics (ACL)",
 9
       doi: "10.18653/v1/2020.acl-main.463",
10
       keywords: "highlight",
       title: "Climbing towards NLU: On Meaning, Form, and Understanding
11
               in the Age of Data",
12
13
       year: "2020",
14
       parsed-author: (
         (given: "Emily M.", family: "Bender"),
15
         (given: "Alexander", family: "Koller"),
16
17
       ),
       sortstr-author: "Bender, Emily M. Koller, Alexander",
18
19
       parsed-editor: none,
20
       parsed-translator: none,
21
       parsed-date: (year: 2020)
22
     label: ("Bender and Koller", "2020"),
23
     label-repr: "Bender and Koller 2020",
24
25 )
```

Figure 7: Example of a reference dictionary.

From the perspective of a style developer, the most important part of the reference dictionary is the fields part, which contains the fields of the BibTeX entry. In the example, the BibTeX entry defined a number of standard BibTeX fields, such as author and title; it also defines some lesser-known fields that are standard in BibTeX, such as keywords; and it also has extra fields, such as award. The reference dictionary makes all of these BibTeX fields available to you.

The keys entry\_type and entry\_key at the top of Figure 7 also come from the BibTeX file; they represent the key and entry type of the BibTeX entry. In addition, Pergamon preprocesses the reference dictionary and adds some fields of its own.

- The label and label-repr fields store the output of the label-generator call.
- The parsed-X fields contain parsed name information. For instance, parsed-author represents the outcome of parsing the names in the author field. It consists of an array of name-part dictionaries, which map the keys given and family (aka "first" and "last" names) to the parts of that author's name.
- The sortstr-author field concatenates the author names, family-name first. It is used when sorting references in a bibliography using the n identifier.

The preprocessing happens relatively early, so the code in a reference or citation style can rely on the presence of these fields in the reference dictionary.

## 6.2 Main functions

These are functions implementing the base functionality of Pergamon, such as cite and print-bibliography.

#### >> add-bib-resource

Parses BibTeX references and makes them available to Pergamon. Due to architectural limitations in Typst, Pergamon cannot read BibTeX from a file. You will therefore typically call read yourself, like this:

```
1 #add-bib-resource(read("bibliography.bib"))
```

You can call add-bib-resource multiple times, and this will add the contents of multiple bib files. However, all bibliography entries must have different keys, even if they are in different source files.

#### **Parameters**

```
add-bib-resource(
  bibtex-string: str,
  source-id: str none
) -> none
```

## bibtex-string str

A BibTeX string to be parsed.

```
source-id str or none
```

If source-id is not none, it is added to all references loaded from this BibTeX source under the source-id field. This can e.g. be used to filter bibliographies by source id.

For instance, this value of filter for print-bibliography() will only show the references that were assigned the source id other.bib:

```
filter: reference => reference.fields.at("source-id", default: none) ==
"other.bib"

Default: none
```

## >> cite

Typesets a citation to the bibliography entry with the given keys. The cite function keeps track of what refsection we are in and uses that refsection's citation formatter to typeset the citation.

You can pass a single key, cite(key), to typeset a citation of a single reference. Alternatively, you can pass multiple keys, cite(key1, key2, key3), to generate a sequence of citations. Depending on the citation style, this may give you a compact and neat citation, such as "[1, 5]" or "(Author 2020; Other 2021)".

Note that bib keys are always given as strings in Pergamon, e.g. cite("paper1").

You can pass a form for finer control over the citation string, depending on what your citation style supports (see Section 3.2). If you do not specify the form, its default value of auto will generate a default form that depends on the citation style.

#### **Parameters**

```
cite(
    ..keys: arguments,
    form: str auto
) -> content
```

```
..keys arguments
```

The keys of the BibTeX entries you want to cite.

```
form str or auto

The citation form.

Default: auto
```

## >> citeg

Typesets a citation with the form "g", e.g. "Smith et al.'s (2020)". See cite() for details.

### **Parameters**

```
citeg(..keys: arguments)
```

```
..keys arguments
```

The keys of the BibTeX entries you want to cite.

#### >> citen

Typesets a citation with the form "n", e.g. "Smith et al. 2020". See cite() for details.

#### **Parameters**

```
citen(..keys: arguments)
```

## ..keys arguments

The keys of the BibTeX entries you want to cite.

#### >> citename

Typesets a citation with the form "name", e.g. "Smith et al.". See cite() for details.

## **Parameters**

```
citename(..keys: arguments)
```

```
..keys arguments
```

The keys of the BibTeX entries you want to cite.

## >> citep

Typesets a citation with the form "p", e.g. "(Smith et al. 2020)". See cite() for details.

## **Parameters**

```
citep(..keys: arguments)
```

```
..keys arguments
```

The keys of the BibTeX entries you want to cite.

#### >> citet

Typesets a citation with the form "t", e.g. "Smith et al. (2020)". See cite() for details.

## **Parameters**

```
citet(..keys: arguments)
```

```
..keys arguments
```

The keys of the BibTeX entries you want to cite.

#### >> citeyear

Typesets a citation with the form "year", e.g. "2020a". See cite() for details.

#### **Parameters**

```
citeyear(..keys: arguments)

..keys arguments
The keys of the BibTeX entries you want to cite.
```

#### >> count-bib-entries

Counts the entries that would be rendered by print-bibliography. This may (eventually?) be useful to implement automatic continuous counting using print-bibliography's resume-after parameter; see Section 4.7.

This function uses context, but does not acquire it internally. This is so you can access the int value returned by count-bib-entries in your own code, rather than getting it back wrapped in content. Calls to the function must therefore be enclosed in the context keyword, e.g.

```
1 #context { count-bib-entries() }
```

See print-bibliography() for an explanation of the parameters.

#### **Parameters**

```
count-bib-entries(
  show-all,
  filter
) -> int
```

#### >> if-citation

Helper function for rendering the links to a bibliography entry. The first argument is assumed to be a Typst link element, obtained e.g. as the argument of a show rule. If this link is a citation pointing to a bibliography entry managed by Pergamon, e.g. generated by Pergamon's cite function, the function passes the metadata of this bib entry to the citation-content function and returns the content this function generated. Otherwise, the link is passed to other-content for further processing.

The primary purpose of if-citation is to facilitate the definition of show rules. A typical example is the following show rule, which colors references to my own publications green and all others blue.

```
1 #show link: it => if-citation(it, value => {
```

```
if "Koller" in family-names(value.reference.fields.parsed-author) {
    set text(fill: green)
    it
    } else {
    set text(fill: blue)
    it
}
```

#### **Parameters**

```
if-citation(
  it: link,
  citation-content: function,
  other-content: function
) -> content
```

### it link

A Typst link element.

### citation-content function

A function that maps the metadata associated with a Pergamon reference to a piece of content. The metadata is a dictionary with keys reference, index, and key. reference is a reference dictionary (see Section 6.1), key is the key of the bib entry, and index is the position in the bibliography.

#### other-content function

A function that maps the link to a piece of content. The default argument simply leaves the link untouched, permitting other show rules to trigger and render it appropriately.

```
Default: x \Rightarrow x
```

## >> print-bibliography

Prints the bibliography for the refsection() in which it is contained. This function cannot be used outside of a refsection.

#### **Parameters**

```
print-bibliography(
  format-reference: function,
  label-generator: function,
  sorting: function str none,
  show-all: bool,
  filter: function,
```

```
grid-style: dictionary,
title: str none,
name-fields: array,
resume-after: int
) -> none
```

#### format-reference function

A function that renders the reference into Typst content, which will then be included in the printed bibliography. This function will typically be defined in a Pergamon style, to be compatible with the format-citation function that is passed to refsection().

format-reference is passed the position of the reference in the bibliography as a zero-based int in the first argument. It is passed the reference dictionary for the reference in the second argument.

It returns an array of contents. The elements of this array will be laid out as the columns of a grid, in the same row, permitting e.g. bibliography layouts with one column for the reference label and one with the reference itself. If only one column is needed (e.g. in the authoryear citation style), format-reference should return an array of length one. All calls to format-reference should return arrays of the same length.

```
Default: (index, reference) => ([REFERENCE],)
```

### label-generator function

Generates label information for the given reference. The function takes the reference dictionary and the reference's index in the sorted bibliography as input and returns an array (label, label-repr), where label can be anything the style finds useful for generating the citations and label-repr is a string representation of the label. These string representations are used to detect label collisions, which cause the generation of extradates.

The default implementation simply returns a number that is guaranteed to be unique to each reference. Styles that want to work with extradate will almost certainly want to pass a different function here.

The function passed as label-generator does not control whether labels are printed in the bibliography in their own separate column; it only computes information for internal use. A style can decide whether it wants to print labels through its format-reference function.

```
Note that label-repr must be a str.

Default: (index, reference) => (index + 1, str(index + 1))
```

```
sorting function or str or none
```

A function that defines the order in which references are shown in the bibliography. This function takes a reference dictionary as input and returns a value that can be sorted, e.g. a number, a string, or an array of sortable values.

Alternatively, you can specify a BibLaTeX-style sorting string. The following strings are supported:

- n: author name (lastname firstname)
- t: paper title
- y: the year in which the paper was published; write yd for descending order
- d: the date on which the paper was published; write dd for descending order
- v: volume, if defined
- a: the contents of the label field (if defined); for the alphabetic style, this amounts to the alphabetic paper key

For instance, "nydt" sorts the references first by author name, then by descending year, then by title.

See Section 5.1 for details on how dates are parsed in the BibTeX entries. If a field of the date (year, month, day) is missing, it is treated as zero for the purposes of sorting. Months that are specified as strings (e.g. "July" rather than 7 or jul) are also treated as zero.

If none or the string "none" is passed as the sorting argument, the references are sorted in an arbitrary order. There is currently no reliable support for sorting the references in the order in which they were cited in the document.

Default: none

#### show-all bool

Determines whether the printed bibliography should contains all references from the loaded bibliographies (true) or only those that were cited in the current refsection (false).

Default: false

#### filter function

Filters which references should be included in the printed bibliography. This makes sense only if show-all is true, otherwise not all your citations will be resolved to bibliography entries. The parameter should be a function that takes a reference dictionary as argument and returns a boolean value. The printed bibliography will contain exactly those references for which the function returned true.

Default: reference => true

## grid-style dictionary

A dictionary for styling the grid in which the bibliography is laid out. By default, the grid is laid out with row-gutter: 1.2em and column-gutter: 0.5em. You can overwrite these values and specify new ones with this argument; the revised style specification will be passed to the grid function.

```
Default: (:)
```

```
title str or none
```

The title that will be typeset above the bibliography in the document. The string given here will be rendered as a first-level heading without numbering. Pass none to suppress the bibliography title.

Default: "References"

#### name-fields array

BibTeX fields that contain names and should be parsed as such. For each X in this array, Pergamon will enrich the reference dictionary with a field parsed-X that contains an array of name-part dictionaries, such as ("family": "Smith", "given": "John"). See Section 6.1 for an example.

If the field X is not defined in the BibTeX entry, Pergamon will still insert a field parsed-X; in this case, it will have the value none.

#### resume-after int

Starts the numbering of entries in this bibliography after the number specified in this argument. Let's say you typeset two bibliographies in your document, and the first one

has 15 entries. You can pass 15 in the resume-after argument to make the numbering of entries in the second bibliography start at 16.

The index parameters of functions like format-reference and format-citation will receive the sum of resume-after and the actual position in this particular bibliography as an argument. In the example above, the first reference in the second bibliography will be called with index=15 (because the count in the second bibliography is zero-based). The only default citation style that cares about indices is *numeric*.

Default: 0

#### >> refsection

Defines a section of the document with its own bibliography. You need to load a bibliography with the add-bib-resource() function in a place that is earlier than the refsection in rendering order.

#### **Parameters**

```
refsection(
  format-citation: function auto,
  id: str auto,
  doc: content
) -> none
```

## format-citation function or auto

A function that generates the citation string for a list of references. The function receives an array of *citation specifications* as its first argument, a form string as its second argument, and an options dictionary as its third argument. It returns the content that is displayed in place of a cite() call.

A citation specification is an array (lbl, reference), where lbl is a citation label and reference is a reference dictionary. The citation formatter is expected to use the information in the reference dictionary to generate the citation and then embed it in a link to the given label (which is anchored by the reference in the bibliography). This might look like this:

```
#link(label(lbl), format(reference))
```

In addition to (lbl, reference) pairs, the citation specification array can also contain elements that are strings (i.e. Typst objects of type str). This happens in cases where the user cites a paper that does not exist in the bibliography. In this case, the string is the key of the cited paper, and the citation formatter is expected to render an appropriate error message. The builtin styles render the key as "?key?".

The form string specifies the exact form in which the citation is rendered; see Section 3.2 for details. This makes the difference e.g. between "Smith et al. (2025)" and "(Smith et al. 2025)".

The options dictionary specifies options that control the rendering of the citation in detail. For instance, the *authoryear* style accepts prefix and suffix arguments. Not every citation style is required to interpret the same options; see the documentation of the citation style for details.

The function you pass here will typically be defined in a Pergamon citation style, to be compatible with the format-reference function that is passed to print-bibliography().

You can pass auto in this argument to indicate that you want to use the same citation formatter as in the previous refsection. If you pass auto to the first refsection in the document, Pergamon will use the dummy citation formatter (references, form) => [CITATION].

Default: auto

#### id str or auto

A unique identifier for this refsection. Each refsection needs its own unique id, which distinguishes it from all the other refsections. You can either specify an explicit identifier here, or you can pass auto to indicate that Pergamon should assign an identifier automatically. In this case, the first refsection in the document receives the identifier none, and the subsequent refsections will be named ref1, ref2, and so on.

All references and citations within a refsection with identifier X will be prefixed by X-; so for instance, the citation cite("knuth1990") in the refsection ref1 will silently introduce and reference a label ref1-knuth1990. If the refsection identifier is none, the original label knuth1990 will be used instead.

Default: auto

#### doc content

The section of the document that is to be wrapped in this refsection.

## 6.3 The builtin styles

Here we explain the builtin reference and citation styles.

## >> format-citation-alphabetic

The *alphabetic* citation style renders citations in a form like "[BK20]". The citation string consists of a sequence of the first letters of the authors' family names. See Figure 3 for an example.

If there are too many authors, the symbol "+" is appended to the citation string to indicate "et al.", e.g. "[YDZ+25]". What constitutes "too many" is controlled by the maxalphanames parameter.

If there is only one author, the first few characters of the author name are displayed instead, as in "[Knu90]". The number of characters is controlled by the labelalpha parameter.

If more than one reference would receive the same citation string under this policy, the style appends an "extradate" character. For example, if two papers would receive the label "[BDF+20]", then the first one (in the sorting order of the bibliography) will be replaced by "[BDF+20a]" and the second one by "[BDF+20b]".

The function format-citation-alphabetic returns a dictionary with keys format-citation, label-generator, and reference-label. You can use the values under these keys as arguments to refsection, print-bibliography, and format-reference, respectively.

#### **Parameters**

```
format-citation-alphabetic(
  maxalphanames: int,
  labelalpha: int,
  labelalphaothers: str,
  citation-separator: str,
  format-brackets: function
)
```

## maxalphanames int

The maximum number of authors that will be printed in a citation string. If the actual number of authors exceeds this value, the symbol specified under labelalphaothers below will be appended to indicate "et al".

Default: 3

## labelalpha int

The maximum number of characters that will be printed for single-authored papers.

Default: 3

## labelalphaothers str

The "et al" character that is appended if the number of authors exceeds the value of the maxalphanames parameter.

Default: "+"

```
citation-separator str
```

The string that separates the author and year in the p citation form.

```
Default: ", "
```

#### format-brackets function

Wraps text in square brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

```
Default: nn(it => [[#it]])
```

## >> format-citation-authoryear

The *authoryear* citation style renders citations in a form like "(Bender and Koller 2020)". The citation string consists of a sequence of the authors' family names. See Figure 4 for an example.

If a paper has more than two authors, only the first author will be printed, together with the symbol "et al.".

If more than one reference would receive the same citation string under this policy, the style appends an "extradate" character. For example, if two papers would receive the label "(Yao et al. 2025)", then the first one (in the sorting order of the bibliography) will be replaced by "(Yao et al. 2025a)" and the second one by "(Yao et al. 2025b)".

The *authoryear* citation style supports a particularly rich selection of citation forms (see Table 1).

The function format-citation-authoryear returns a dictionary with keys format-citation, label-generator, and reference-label. You can use the values under these keys as arguments to refsection, print-bibliography, and format-reference, respectively.

#### **Parameters**

```
format-citation-authoryear(
  format-parens: function,
  format-brackets: function,
  author-year-separator: str,
  citation-separator: str,
  list-middle-delim: str,
  list-end-delim-two: str,
  list-end-delim-many: str,
  bibstring: dictionary,
  maxnames: int,
  minnames: int)
```

## format-parens function

Wraps text in round brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

```
Default: nn(it => [(#it)])
```

#### format-brackets function

Wraps text in square brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

```
Default: nn(it => [[#it]])
```

## author-year-separator str

The string that separates the author and year in the p citation form.

Default: " "

## citation-separator str

Separator symbol to connect the citations for the different keys.

```
Default: "; "
```

#### list-middle-delim str

When typesetting lists (e.g. author names), Pergamon will use this delimiter to combine list items before the last one.

Will typically match the list-middle-delim argument of format-reference().

```
Default: ", "
```

#### list-end-delim-two str

When typesetting lists (e.g. author names), Pergamon will use this delimiter to combine the items of lists of length two.

Will typically match the list-end-delim-two argument of format-reference().

```
Default: " and "
```

## list-end-delim-many str

When typesetting lists (e.g. author names), Pergamon will use this delimiter in lists of length three of more to combine the final item in the list with the rest.

Will typically match the list-end-delim-many argument of format-reference().

Default: ", and "

## bibstring dictionary

Overrides entries in the bibstring table. The bibstring table is a dictionary that maps language-independent IDs of bibliographic constants (e.g. "in") to their language-dependent surface forms (such as "In: " or "edited by"). The ID-form pairs you specify in the bibstring argument will overwrite the default entries.

Will typically match the bibstring argument of format-reference().

See the documentation for default-bibstring in Section 6.4 for more information on the bibstring table.

Default: (:)

#### maxnames int

Maximum number of names that are displayed in name lists (author, editor, etc.). If the actual number of names exceeds maxnames, only the first maxnames names are shown and bibstring.andothers ("et al.") is appended.

This parameter is modeled after the maxnames/maxcitenames option in BibLaTeX. Just like maxbibnames is not necessarily the same as maxcitenames, the value of maxnames here need not match the value of maxnames in format-reference().

Default: 2

#### minnames int

Minimum number of names that are displayed in name lists (author, editor, etc.). This can be used in conjunction with maxnames to create name lists like "Jones, Smith et al." (minnames = 2, maxnames = 2).

minnames trumps maxnames: That is, if the name list is at least as long as minnames, the reference will show minnames names, even if this exceeds maxnames. In typical use cases, minnames will be less or equal than maxnames, so this situation will usually not occur.

This parameter is modeled after the minnames/mincitenames option in BibLaTeX. Just like minbibnames is not necessarily the same as mincitenames, the value of minnames here need not match the value of names in format-reference().

#### >> format-citation-numeric

The *numeric* citation style renders citations in a form like "[1]". The citation string is the position of the reference in the bibliography. See Figure 1 for an example.

The function format-citation-numeric returns a dictionary with keys format-citation, label-generator, and reference-label. You can use the values under these keys as arguments to refsection, print-bibliography, and format-reference, respectively.

#### **Parameters**

```
format-citation-numeric(
  citation-separator: str,
  format-brackets: function
)
```

## citation-separator str

The string that separates the author and year in the p citation form.

Default: ", "

## format-brackets function

Wraps text in square brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

```
Default: nn(it => [[#it]])
```

## >> format-reference

The standard reference style. It is modeled after the standard bibliography style of BibLaTeX.

A call to format-reference takes a number of options as argument and returns a function that will take arguments index and reference and return a rendered reference. This function is suitable as an argument to the format-reference parameter of print-bibliography, and will control how the references in this bibliography are rendered. See the documentation of print-bibliography for a more detailed specification of the format-reference function in general.

Most of the options of format-reference have sensible default values. The one exception is the mandatory named argument reference-label, which you obtain from your citation style.

#### **Parameters**

```
format-reference(
  reference-label: function,
  highlight: function,
  link-titles: bool,
  print-identifiers: array,
  print-url: bool,
  print-doi: bool,
  print-eprint: bool,
  use-author: bool,
  use-translator: bool,
  use-editor: bool,
  print-date-after-authors: bool,
  eval-mode: str,
  eval-scope: dictionary,
  list-middle-delim: str,
  list-end-delim-two: str,
  list-end-delim-many: str,
  author-type-delim: str,
  subtitlepunct: str,
  format-journaltitle: function,
  format-issuetitle: function,
  format-maintitle: function,
  format-booktitle: function,
  format-fields: dictionary,
  format-parens: function,
  format-brackets: function,
  format-quotes: function,
  name-format: str dictionary,
  volume-number-separator: str,
  bibeidpunct: str,
  bibpagespunct: str,
  print-isbn: bool,
  bibstring: dictionary,
  additional-fields: array none,
  suppress-fields: array dictionary none,
  period: str array,
  comma: str array,
  maxnames: int,
  minnames: int
)
```

The reference labeler that should be used for this bibliography; see Section 3.3 for a detailed explanation.

The reference labeler typically comes from a citation style (e.g. *authoryear*, *numeric*, *alphabetic*).

Unlike the other parameters of format-reference, you *must* pass a meaningful argument for this parameter. If you leave it at the default value of none, Typst will not even show you a proper error message; it will just say "warning: layout did not converge within 5 attempts".

Default: none

## highlight function

Selectively highlights certain bibliography entries. The parameter is a function that is applied at the final stage of the rendering process, where the whole rest of the entry has already been rendered. This is an opportunity to e.g. mark certain entries in the bibliography by boldfacing them or prepending them with a marker symbol. See Section 4.8 for an example.

The highlighting function accepts arguments rendered-reference (str or content representing the reference as it is printed), index (position of the reference in the bibliography), and reference (the reference dictionary). It returns content. The default implementation simply returns the rendered-reference unmodified.

Default: (rendered-reference, index, reference) => rendered-reference

#### link-titles bool

If true, titles are rendered as hyperlinks pointing to the reference's DOI or URL. When both are defined, the DOI takes precedence.

Default: true

#### print-identifiers array

Array of reference identifiers that should be printed at the end of the bibliography entry. The array contains a list of strings; possible values are "doi", "url", and "eprint". For each bib entry, these values are considered in the order in which they appear in the array, and the first value that is defined as a key in the bib entry is printed.

print-identifiers acts as a flexible version of the print-url, print-doi, and print-eprint parameters. For the first X in the array that is defined in the bib entry, it effectively sets print-X to true. The values of the other print-X parameters are left unchanged; thus, you could e.g. still force the rendering of eprint by setting print-eprint to true, even if print-identifiers matches on the DOI instead.

#### Default: ()

## print-url bool

If true, prints the reference's URL at the end of the bibliography entry.

Default: false

## print-doi bool

If true, prints the reference's DOI at the end of the bibliograph entry.

Default: false

## print-eprint bool

If true, prints the reference's eprint information at the end of the bibliography entry. This could be a reference to arXiv or JSTOR.

Default: true

#### use-author bool

If true, prints the reference's author if it is defined.

Default: true

## use-translator bool

If true, prints the reference's translator if it is defined. Note that support for "authors" that are not the author is currently weak. See issue 28 to track progress on this.

Default: true

#### use-editor bool

If true, prints the reference's editor if it is defined. Note that support for "authors" that are not the author is currently weak. See issue 28 to track progress on this.

Default: true

## print-date-after-authors bool

If true, Pergamon will print the date right after the authors, e.g. 'Smith (2020). "A cool paper".' If false, Pergamon will follow the normal behavior of BibLaTeX and place the date towards the end of the reference.

Default: false

### eval-mode str

When Pergamon renders a reference, the title is processed by Typst's eval function. The eval-mode argument you specify here is passed as the mode argument to eval.

The default value of "markup" renders the title as if it were ordinary Typst content, typesetting e.g. mathematical expressions correctly.

Default: "markup"

## eval-scope dictionary

The scope argument that is passed to the eval call (see eval-mode). This allows you to call Typst functions from within the BibTeX entries.

Default: (:)

### list-middle-delim str

When typesetting lists (e.g. author names), Pergamon will use this delimiter to combine list items before the last one.

Default: ", "

### list-end-delim-two str

When typesetting lists (e.g. author names), Pergamon will use this delimiter to combine the items of lists of length two.

Default: " and "

## list-end-delim-many str

When typesetting lists (e.g. author names), Pergamon will use this delimiter in lists of length three of more to combine the final item in the list with the rest.

Default: ", and "

## author-type-delim str

String that is used to combine the name of an author with the author type, e.g. "Smith, editor".

Default: ","

## subtitlepunct str

String that is used to combine a title with a subtitle.

Default: "."

## format-journaltitle function

Renders the title and subtitle of a journal as content. The default argument typesets it in italics.

Certain titles (journaltitle, issuetitle, maintitle, booktitle) can be combined with subtitles (e.g. journalsubtitle). The title and subtitle are joined by subtitlepunct to obtain e.g. "Journal title: subtitle". The subtitlepunct needs to be formatted the same as the title and subtitle, but its formatting cannot be controlled by format-fields. This is why Pergamon offers parameters such as format-journaltitle to format the entire concatenated title and subtitle.

Default: it => emph(it)

### format-issuetitle function

Renders the title of a special issue as content. The default argument typesets it in italics.

See format-journaltitle for further explanation.

Default: it => emph(it)

#### format-maintitle function

Renders the main title of a multi-volume work as content. The default argument typesets it in italics.

See format-journaltitle for further explanation.

Default: it => emph(it)

#### format-booktitle function

Renders the title of a book as content. The default argument typesets it in italics.

See format-journaltitle for further explanation.

Default: it => emph(it)

## format-fields dictionary

Overrides the way individual fields in the reference are rendered. The argument is a dictionary that maps the names of fields in a BibTeX entry to *extended field formatters*, i.e. functions that compute Typst content and take the following positional parameters:

- dffmt: the *default* field formatter, which is applied if no more specific field formatter is specified through format-fields.
- value: the value of the field in the BibTeX entry.
- reference: the reference dictionary, cf. Section 6.1.
- field: the name of the field.
- options: a dictionary containing all the options that were passed to format-reference as arguments.
- style: an optional style specification for the field.

If you do not override the rendering of a field, Pergamon uses a default field formatter, i.e. a function that computes content from the value, reference, field, options, and style parameters explained above. This default field formatter is passed as the first argument (dffmt) to the extended field formatter explained above.

Some examples of how format-fields can be used are shown in Section 4.3.

Default: ()

## format-parens function

Wraps text in round brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

Default: nn(it => [(#it)])

## format-brackets function

Wraps text in square brackets. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

Default: nn(it => [[#it]])

## format-quotes function

Wraps text in double quotes. The argument needs to be a function that takes one argument (str or content) and returns content.

It is essential that if the argument is none, the function must also return none. This can be achieved conveniently with the nn function wrapper, see Section 6.4.

```
Default: nn(it => ["#it"])
```

## name-format str or dictionary

The format in which names (of authors, editors, etc.) are printed. This is an arbitrary string which may contain the placeholders {given} and {family}; these will be replaced by the person's actual name parts. You can use {g} and {f} for the first letters of the given and family name, respectively.

Instead of a string, you can also pass a dictionary in this argument. The keys are name types ("author", "editor", etc.), and the values are name format strings as explained above. The style will use a default format of "{given} {family}" for name types that you did not specify.

Default: "{given} {family}"

## volume-number-separator str

Separator symbol for "volume" and "number" fields, e.g. in @articles.

Default: "."

#### bibeidpunct str

Separator symbol that connects the EID (Scopus Electronic Identifier) from other journal information.

Default: ","

## bibpagespunct str

Separator symbol that connects the "pages" field with related information.

Default: ","

## print-isbn bool

If true, prints the ISBN or ISSN of the reference if it is defined.

Default: false

## bibstring dictionary

Overrides entries in the bibstring table. The bibstring table is a dictionary that maps language-independent IDs of bibliographic constants (e.g. "in") to their language-dependent

dent surface forms (such as "In: " or "edited by"). The ID-form pairs you specify in the bibstring argument will overwrite the default entries.

See the documentation for default-bibstring in Section 6.4 for more information on the bibstring table.

Default: (:)

## additional-fields array or none

An array of additional fields which will be printed at the end of each bibliography entry. Fields can be specified either as a string, in which case the field with that name is printed using the reference style's normal rules. Alternatively, they can be specified as a function (reference, options) -> content, in which case the returned content will be printed directly. Instead of an array, you can also pass none to indicate that no additional fields need to be printed.

For example, both of these will work:

```
additional-fields: ("award",)
additional-fields: ((reference, options) =>
   ifdef(reference, "award", (:), award => [*#award*]),)
Default: none
```

## suppress-fields array or dictionary or none

A specification of field names that should not be printed. References are treated as if they do not contain values for these fields, even if the BibTeX file defines them.

You can pass an array of strings here. These field names will be suppressed in all references.

Alternatively, you can pass a dictionary that maps entry types to arrays of strings. ("inproceedings": ("editor", "location")) means that the editor and location fields will not be printed in inproceedings references, but may be printed in other entry types. You can use the special key "\*" to suppress fields in *all* entry types.

Finally, you can also pass none to indicate that no fields should be suppressed.

Default: none

```
period str or array
```

Specifies how string or content elements are joined using period symbols. This corresponds roughly (but not precisely) to blocks in BibLaTeX.

If you specify a string here, this string will be used to join the elements. Pergamon will avoid duplicating connector symbols, i.e. it will not print a period if the preceding symbol was already a period.

Alternatively, you can specify an array (connector, skip-chars), where connector is the period symbol. The connector is skipped if the preceding character occurs in the string skip-chars.

Default: (".", ".,?!;:")

## comma str or array

Function that joins an arbitrary number of strings or contents with a comma symbol. This corresponds roughly (but not precisely) to units in BibLaTeX.

See the documentation for period for details.

Default: ","

#### maxnames int

Maximum number of names that are displayed in name lists (author, editor, etc.). If the actual number of names exceeds maxnames, only the first maxnames names are shown and bibstring.andothers ("et al.") is appended.

This parameter is modeled after the maxnames/maxbibnames option in BibLaTeX.

Default: 9999

#### minnames int

Minimum number of names that are displayed in name lists (author, editor, etc.). This can be used in conjunction with maxnames to create name lists like "Jones, Smith et al." (minnames = 2, maxnames = 2).

minnames trumps maxnames: That is, if the name list is at least as long as minnames, the reference will show minnames names, even if this exceeds maxnames. In typical use cases, minnames will be less or equal than maxnames, so this situation will usually not occur.

This parameter is modeled after the minnames/minbibnames option in BibLaTeX.

Default: 9999

## **6.4 Utility functions**

The following functions may be helpful in the advanced usage and customization of Pergamon.

#### >> concatenate-names

Concatenates an array of names. If there is only one name, it is returned unmodified. If there are two names, they are concatenated with the value options.list-end-delim-two ("and"). If there are three names, they are concatenated with the value options.list-end-delim-two (", "), except the last name is joined with options.list-end-delim-many (", and").

#### **Parameters**

```
concatenate-names(
  names: array,
  options: dictionary,
  maxnames,
  minnames
) -> str content
```

#### names array

An array of names. Each name can a string or content. If the names are strings, the function will return a string; if at least one name is content, the function will return content.

## options dictionary

Options that control the concatenation. concatenate-names defines reasonable default options for list-end-delim-two, list-end-delim-two, list-end-delim-many, and bibstring. You can override these options by passing them in a dictionary here.

Default: (:)

#### maxnames

Maximum number of names that is displayed before the name list is truncated with "et al." See the maxnames parameter in format-reference() for details.

Default: 2

#### minnames

Minimum number of names that is guaranteed to be displayed. See the minnames parameter in format-reference() for details.

Default: 1

#### >> nn

Wraps a function in none-handling code. nn(func) is a function that behaves like func on arguments that are not none, but if the argument is none, it simply returns none. Only works for functions func that have a single argument.

#### **Parameters**

```
nn(func) -> function
```

## >> family-names

Extracts the list of family names from the list of name-part dictionaries. For instance, the parsed-author entry of the example in Figure 7 will be mapped to the array ("Bender", "Koller").

If parsed-names is none, the function returns none.

#### **Parameters**

```
family-names(parsed-names) -> array none
```

#### >> format-name

Spells out a name-part dictionary into a string. See the documentation of the name-format argument of format-reference() for details on the format string.

#### **Parameters**

```
format-name(
  name-parts-dict: dictionary,
  name-type: str,
  format: str dictionary
) -> str
```

```
name-parts-dict dictionary
```

A name-part dictionary

```
name-type str
```

The type of name as which the name-part dictionary should be formatted. If format is a dictionary, format-name will look up the format string under this key in format.

```
Default: "author"
```

## format str or dictionary

A format string or dictionary which specifies how the name should be formatted. You can either pass a string, or you can pass a dictionary that maps name types to strings.

```
Default: "{family}, {given}"
```

## >> default-bibstring

The default bibstring table. A bibstring table maps a number of language-independent identifiers to the strings that will be printed in the bibliography (see the bibstring parameter of format-reference). Modifying the bibstring table can be useful if you want to localize your bibliography to a language other than English, or if your bibliographic conventions differ from those of the standard BibLaTeX style. For instance, to keep the reference style from printing "In:" with a trailing colon, you could use a bibstring table in which the "in" key maps to "In" rather than "In:".

This default bibstring table is a verbatim copy of the standard English bibstring table in BibLaTeX. It contains only the "long" versions of these entries; so references will be typeset to print e.g. "Jones, editor", rather than "Jones, ed.". You can find this table in the file bibstrings.typ and modify it to your needs.

## 7 Changelog

## Changes in 0.4.0 (unreleased)

- Introduced the format-field argument, which allows flexible control over how Pergamon formats individual BibTeX fields in the reference.
- Added the minnames and maxnames arguments to format-reference and format-citation-authoryear, replicating the BibLaTeX options.
- More flexible control over the bibstring table.
- Cleaned up the formatting of titles that permit subtitles.

#### Changes in v0.3.2 (2025-10-02)

- Fixed a number of bugs.
- Added the print-identifiers parameter to format-reference.

## Changes in v0.3.1 (2025-09-22)

- Fixed a number of bugs.
- Can now specify different name formats for authors, editors, etc.
- Can now specify prefixes and suffixes in authoryear citations.
- Citing a nonexistent reference key now displays a warning.

## Changes in v0.3.0

• Author names are now parsed as in BibLaTeX, using the parser of the biblatex crate.

- Added an eval-scope argument to format-reference and dropped the eval-mode parameter from print-bibliography. It is now specified directly on format-reference.
- Added name and year citation forms.
- print-bibliography now has a resume-after parameter.
- More correct date handling: months can now be suppressed in the bibliography, and the d sorting specifier works correctly.

## Changes in v0.2.0

- Aggregated citation commands: #cite("key1", "key2", ...).
- Support for date and month fields.
- Added source-id parameter to add-bib-resource.
- The suppress-fields option in format-reference can now be defined per entry type.
- Default style avoids printing two punctuation symbols in a row.
- Defining multiple references with the same key is now an error.