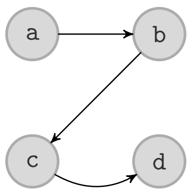
The argumentation Package

Lars Bengel[*]

lars.bengel@fernuni-hagen.de

Version 1.4 [2024/11/03]



```
\begin{af}[argumentstyle=gray,namestyle=monospace]
    \argument{a}
    \argument[right=of a1]{b}
    \argument[below=of a1]{c}
    \argument[right=of a3]{d}

    \attack{a1}{a2}
    \attack{a2}{a3}
    \attack[bend right]{a3}{a4}
    \label{af:example}
\end{af}
```

Contents

1 Quick Guide

To create an argumentation framework in your LaTeX-document, you first have to import the `argumentation` package in the preamble:

```
\usepackage{argumentation}
```

You can then create a new `af` environment in which the argumentation framework can then be built:

```
\begin{af}
    ⟨environment contents⟩
\end{af}
```

You may want to wrap the `af` environment in a `figure` environment in order to add a caption and reference label. You can also add a label inside the `af` environment via `\label{⟨label⟩}`. Anywhere in your document, you can then reference the af with `\ref{⟨label⟩}`.

Inside the `af` environment, you can then add an argument as follows:

```
\argument{⟨name⟩}
```

Here, ⟨*name*⟩ is the name of the argument displayed in the graph and the argument is automatically assigned an *identifier* of the form: $a1$, $a2$, ...

To properly add further arguments, you also need to specify a position. The `argumentation` package offers two easy ways of doing that:

```
\argument[⟨dir⟩=of ⟨argId⟩]{⟨name⟩}

\argument{⟨name⟩} at (⟨posX⟩,⟨posY⟩)
```

The first instance is *relative positioning* where ⟨*dir*⟩ is the direction of placement relative to the argument with the identifier ⟨*argId*⟩, with ⟨*dir*⟩ typically being one of: right, left, above, below.

The second instance is *absolute positioning* where (⟨*posX*⟩, ⟨*posY*⟩) is a set of coordinates, for example something like `(2, 0)`, `(0, -2)` or `(-1, 3.5)`.

The next step is adding attacks. For that you can simply use the following command:

```
\attack{⟨a1⟩}{⟨a2⟩}
```

Substitute ⟨*a1*⟩ and ⟨*a2*⟩ with the identifier of the two arguments. Alternatively, you can also directly create bidirectional attacks and self-attacks with the following two commands:

```
\dualattack{⟨a1⟩}{⟨a2⟩}
\selfattack{⟨a1⟩}
```

To customize the look of the arguments and attacks and for a detailed overview over all options and commands provided by this package, please refer to the following example or to the full documentation in Section 3.

2 Example Usage

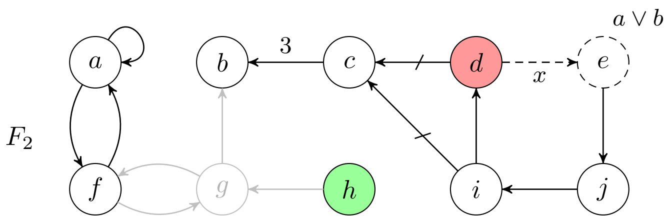


Figure 1: The AF F_2 created with the `argumentation` package.

```
\usepackage[namestyle=math]{argumentation}
...
\begin{document}
...
\begin{figure}[ht]
    \centering
    \begin{af}
        \argument{a}
        \argument[right=of a1]{b}
        \argument[right=of a2]{c}
        \argument[rejected,right=of a3]{d}
        \argument[right=of a4,incomplete]{e}
        \argument[below=of a1]{f}
        \argument[inactive,right=of a6]{g}
        \argument[accepted,right=of a7]{h}
        \argument[right=of a8]{i}
        \argument[right=of a9]{j}

        \annotation[right,yshift=-0.4cm]{a5}{$a\lor b$}
        \afname{$F_{\ref{af:ex2}}$} at (-1,-1)

        \selfattack{a1}
        \dualattack{a1}{a6}
        \dualattack[inactive]{a6}{a7}

        \attack[inactive]{a8}{a7}
        \attack[inactive]{a7}{a2}
        \attack{a5}{a10}
        \attack{a10}{a9}
        \attack{a9}{a4}

        \annotatedattack[above]{a3}{a2}{$3$}
        \annotatedattack[below,incomplete]{a4}{a5}{$x$}
        \support{a4}{a3}
        \support{a9}{a3}
        \label{af:ex2}
    \end{af}
    \caption{The AF $F_{\ref{af:ex2}}$ created with the \argumentation package.}
    \label{fig:example}
\end{figure}
...
\end{document}
```

3 Documentation for Version 1.4 [2024/11/03]

The `argumentation` package provides an easy way for creating argumentation frameworks[1] in LaTeX-documents. It builds on the TikZ package for drawing the argumentation graphs. The `argumentation` package provides simplified syntax while keeping the same customisation options and keeping full compatibility with all TikZ features. In addition to that, the `argumentation` package provides the ability to label and reference the created argumentation frameworks as well as some other additional features.

The `argumentation` package can be imported via the command

```
\usepackage[⟨options⟩]{argumentation}
```

In the following, we give an overview over the functionality of the `argumentation` package. Most importantly, that includes the `af` environment to encapsulate the created argumentation frameworks, the command `\argument{ }` to create argument nodes and the `attack{ }{ }` command to create attack edges. Options to customise the appearance of arguments and attacks are described in Section 4.

3.1 The af Environment

The `argumentation` package provides an environment for creating argumentation frameworks in LaTeX-documents.

```
\begin{af} [⟨options⟩]
    ⟨environment contents⟩
\end{af}
```

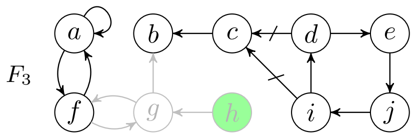
The `af` environment supports referencing. For that add the command `\label{⟨label⟩}` anywhere inside an `af` environment. The AFs are automatically numbered in ascending order of occurrence. The *⟨label⟩* allows you to reference the corresponding AF via `\ref{⟨label⟩}` anywhere in the document.

If you want to create an AF that is excluded from the automatic numbering, the `argumentation` package provides the `af*` version of the environment, which has the same functionality otherwise:

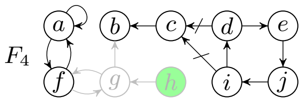
```
\begin{af*} [⟨options⟩]
    ⟨environment contents⟩
\end{af*}
```

The `af` (and `af*`) environment also accepts the package style options (see Section 4). Locally set style options override defaults and the values set globally with the package import.

In general, the `af` environment extends the `tikzpicture` environment, meaning all TikZ commands and parameters can be used for the `af` environment. The `argumentation` package also provides the options small or tiny for the `af` environment to create smaller AFs. This is especially useful for two-column layout documents.



(a) An AF created with the small option set.



(b) An AF created with the tiny option set.

Figure 2: Argumentation frameworks using the small and tiny option of the `af` environment.

[1]Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artificial intelligence.

3.2 Creating Arguments

Inside an `af` (or `af*`) environment, you can create argument nodes for the argumentation framework with the following command

`\argument [⟨options⟩] (⟨id⟩) {⟨name⟩} at (⟨posX⟩,⟨posY⟩)`

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters and/or relative positioning information.

⟨*id*⟩ (optional) the identifier of the new argument. Per default, when omitted, arguments will automatically be assigned an identifier of the form: $a1, a2, a3, \ldots$.

⟨*name*⟩ the displayed name of the argument.

⟨*posX*⟩,⟨*posY*⟩ (optional) the coordinates where the argument is placed. Must be omitted if relative positioning is used.

3.2.1 Positioning

The `argumentation` package also provides the ability to use *relative positioning* instead of absolute positioning via coordinates. For that, it relies on relative placement via the Ti*k*Z-library positioning. The relative positioning information is provided as an optional parameter via `[⟨options⟩]` as follows

`\argument[⟨dir⟩=of ⟨argId⟩]{⟨name⟩}`

⟨*dir*⟩ The direction of placement relative to the argument ⟨*argId*⟩. Typically one of: above, right, left or below.

⟨*argId*⟩ The identifier of another argument.

⟨*name*⟩ The displayed name of the argument.

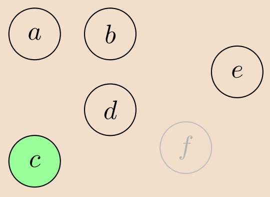
Additionally, you can adjust the horizontal/vertical position of an argument by adding `xshift=`⟨*v*⟩ or `yshift=`⟨*v*⟩ inside the `[...]`. The value ⟨*v*⟩ is hereby the horizontal/vertical offset, e. g., `-6.6ex` or `1cm`.

In the following, we list some useful style options for the `\argument` command, provided by the`argumentation` package:

inactive	The argument is displayed with grey outline and text.
incomplete	The argument is displayed with a dotted outline.
invisible	The argument node is completely transparent.
accepted	The argument is displayed with green background color.
rejected	The argument is displayed with red background color.
undecided	The argument is displayed with cyan background color.
highlight	The argument is displayed with yellow background color.

Table 1: Some style options for the `\argument` command (for exact definition see Section 5).

Example 1



```
\begin{af}[namestyle=math]
    \argument{a}
    \argument{b} at (1,0)
    \argument[below=of a1,accepted](c){c}
    \argument(x){d} at (1,-1)
    \argument[right=of x,yshift=0.5cm]{e}
    \argument[inactive]{f} at (2,-1.5)
\end{af}
```

3.3 Creating Attacks

To create an directed attack between two argument nodes, you can use the following command. The parameters ⟨*arg1*⟩ and ⟨*arg2*⟩ are the identifiers of the two arguments.

`\attack [⟨options⟩] {⟨argId1⟩} {⟨argId2⟩}`

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters.

⟨*arg1*⟩ Identifier of the attacking argument.

⟨*arg2*⟩ Identifier of the attacked argument.

To simplify creating special types of attacks, like bidirectional attacks or self-attacks, the following two commands are provided.

`\dualattack [⟨options⟩] {⟨argId1⟩} {⟨argId2⟩}`

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters.

⟨*arg1*⟩ Identifier of the first argument.

⟨*arg2*⟩ Identifier of the second argument.

`\selfattack [⟨options⟩] {⟨argId⟩}`

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters.

⟨*arg1*⟩ Identifier of the self-attacking argument.

For `\selfattack` you might want to specify the position of the attack loop. For that, you should provide the start and end point of the attack-edge (as a degree from `0` to `360`) via the optional Ti*k*Z-parameters `in=`⟨*degree1*⟩ and `out=`⟨*degree2*⟩. The default values are `in=0` and `out=60`.

If you want to attach a value to an attack edge, you may use the following command.

`\annotatedattack [⟨options⟩] {⟨argId1⟩} {⟨argId2⟩} {⟨value⟩}`

⟨*options*⟩ Optional Ti*k*Z parameters. Must also include one of the following parameters to specify placement of the annotation relative to the attack arrow: above, below, left, right.

⟨*arg1*⟩ Identifier of the attacking argument.

⟨*arg2*⟩ Identifier of the attacked argument.

⟨*value*⟩ The text that is annotated.

As an alternative to the standard attack arrow, you can use the following command.

`\support [⟨options⟩] {⟨argId1⟩} {⟨argId2⟩}`

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters.

⟨*arg1*⟩ Identifier of the supporting argument.

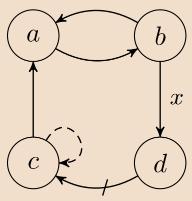
⟨*arg2*⟩ Identifier of the supported argument.

Some useful style options for the attacks (and other edges) are listed below:

inactive	The attack is displayed in grey.
incomplete	The attack is displayed with a dotted line.
invisible	The attack is completely transparent.
selfattack	Use if source and target of the attack are the same node.
bend right	The attack arrow is bent to the right. Can additionally provide the angle, e. g., bend right=40.
bend left	The attack arrow is bent to the left. Can also provide an angle.

Table 2: Some useful style options for the `\attack` (and related) commands. For the exact definition see Section 5.

Example 2



```
\begin{af}[namestyle=math]
    ...

    \attack{a3}{a1}
    \selfattack[incomplete]{a3}
    \dualattack{a1}{a2}
    \annotatedattack[right]{a2}{a4}{$x$}
    \support[bend left]{a4}{a3}
\end{af}
```

3.4 Beamer

If you want to reuse (parts of) previously created argumentation frameworks in some form, the `argumentation` package provides some useful commands that can be enabled via the package option beamer=true. While primarily intended for the use inside the beamer document class when creating presentations, the commands also work in any other document class. Each command required the label of some argumentation framework and a list of argument IDs of that framework and then creates a copy of that framework, with some changes depending on the command.

`\aflabeling` {⟨*af-label*⟩} {⟨*argument list*⟩}

Applies the style parameter accepted to all arguments in ⟨*argument list*⟩, the parameter rejected to those attacked by arguments in ⟨*argument list*⟩ and undecided to all other arguments.

`\afextension` {⟨*af-label*⟩} {⟨*argument list*⟩}

Applies the style parameter accepted to all arguments in ⟨*argument list*⟩.

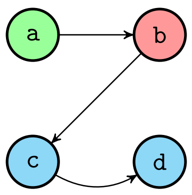
`\afreduct` {⟨*af-label*⟩} {⟨*argument list*⟩}

Applies the style parameter inactive to all arguments in ⟨*argument list*⟩ and those attacked by them. All attacks involving at least one such argument also receive the parameter inactive.

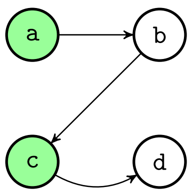
`\afrestriction` {⟨*af-label*⟩} {⟨*argument list*⟩}

Applies the style parameter invisible to all arguments *not* in ⟨*argument list*⟩. All attacks involving at least one such argument also receive the parameter invisible.

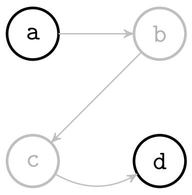
See Figure 3 for some examples.



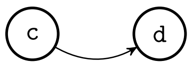
(a) Result of `\aflabeling{af:example}{a1}`.



(b) Result of `\afextension{af:example}{a1,a3}`.



(c) Result of `\afreduct{af:example}{a2}`.



(d) Result of `\afrestriction{af:example}{a3,a4}`.

Figure 3: Example usage of the four commands provided by the beamer package option.

3.5 Other Commands

The `argumentation` package also provides some additional features. The following command can be used to create a text annotation next to an argument node in the argumentation framework. The annotation we be placed above the argument node, to adjust its position you should use `xshift` and `yshift`.

```
\annotation [⟨options⟩] {⟨argId⟩} {⟨value⟩}
```

⟨*options*⟩ Optional Ti*k*Z style parameters.

⟨*argId*⟩ Identifier of the argument.

⟨*value*⟩ The annotation text.

The command `\afname` can be used to create a simple text node inside the AF. Mainly intended to add the name of the AF into the picture, you can generally put any text there. The command behaves essentially exactly like the `\argument` command.

```
\afname [⟨options⟩] (⟨id⟩) {⟨name⟩} at (⟨posX⟩, ⟨posY⟩)
```

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters and/or relative positioning parameters.

⟨*id*⟩ (optional) Identifier of the text node. If omitted the identifier will be caption.

⟨*text*⟩ Text to be displayed.

⟨*posX*⟩,⟨*posY*⟩ (optional) the coordinates for placement. Omit if using relative positioning.

If you want to define your own style for arguments, attacks or supports, you may use one of the following commands to override the package-wide settings to your liking. For that you may also reuse some of the pre-defined parameters of the `argumentation` package (see Section 5).

```
\setargumentstyle {⟨style parameters⟩}
\setatackstyle {⟨style parameters⟩}
\setsupportstyle {⟨style parameters⟩}
```

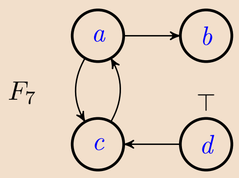
Similarly, you can also use the following command to override the default settings of the `af` environment, e. g., the `node distance`.

```
\setafstyle {⟨style parameters⟩}
```

Finally, when using the argumentstyle=*colored* package option, you may use the following command to set the color scheme.

```
\setargumentcolorscheme {⟨outer color⟩} {⟨inner color⟩}
```

Example 3



```
\setargumentstyle{argument thick,text=blue}
\setafstyle{node distance=0.75cm}
\begin{af}\label{af:8}
    ...

    \annotation[yshift=-0.2cm]{a4}{$\top$}
    \afname{$F_{\ref{af:8}}$} at (-1,-0.75)
\end{af}
```

3.5.1 Argumentation Macros

To facilitate referencing argumentation frameworks and working with them in general, the `argumentation` package provides some additional macros that can be enabled with the package option macros=true. Most importantly, there is the macro `\afref{⟨label⟩}` which works like the `ref` command but adds the reference number directly into the index of the `\AF` symbol. You may redefine any of the first four commands if you prefer a different naming scheme for AFs.

`\AF`	F
`\arguments`	A
`\attacks`	R
`\AFcomplete`	$F = (A, R)$
`\afref{af:example}`	F_1
`\fullafref{af:example}`	$F_1 = (A_1, R_1)$

Table 3: Macros provided by the package option macros=true and their respective output.

4 Package Options

The `argumentation` package comes with some package options to customize the appearance of the created argumentation frameworks as well as some additional features. All style package options can both be set globally when importing the package and also locally for each `af` environment. To import the `argumentation` package, use the following command in the preamble of your LaTeX-document:

```
\usepackage[⟨options⟩]{argumentation}
```

The following package options are currently available:

argumentstyle (default `standard`) Globally sets the appearance of the argument nodes. The `argumentation` package provides five options: `standard`, `large`, `thick`, `gray` and `colored`. Detailed descriptions of these options can be found below.

attackstyle (default `standard`) Globally sets the appearance of the attack edges. The package comes with three available options: `standard`, `large` and `modern`. Detailed descriptions of these options can be found below.

supportstyle (default `standard`) Globally sets the appearance of the support edges. The package comes with three available options: `standard`, `dashed` and `double`. Detailed descriptions of these options can be found below.

namestyle (default `none`) Sets the text formatting applied to the argument names in the document. The package comes with five available options: `none`, `math`, `bold`, `monospace` and `monoemph`. Detailed descriptions of these options can be found below.

indexing (default `numeric`) Enables or disables automatic generation of Ti*k*Z node-IDs for the created arguments. The available options are: `none`, `numeric` and `alphabetic`. Under the default numeric indexing the generated argument IDs are of the form $a1, a2, \ldots$. With alphabetic indexing the IDs will simply be letters: $a, b, \ldots$. If `none` is selected, no IDs will be generated and you are required to provide them for each argument via the parameter `(`⟨*id*⟩`)` of the `\argument` command.

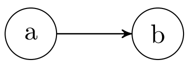
macros Boolean (default `false`) When enabled provides additional macros for naming and referencing argumentation frameworks (see Table 3).

beamer Boolean (default `false`) When enabled, provides the commands for recreating argumentations frameworks described in Section 3.4.

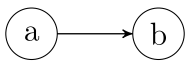
In the following we give an overview of the different options for the style parameters that can be used to customise the created argumentation frameworks. For the exact definitions of these parameters, refer to Section 5.

argumentstyle=⟨*option*⟩

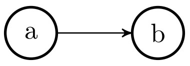
standard	Circular argument node with normal size argument name.
large	Larger font of the argument name.
thick	Thick black outline and normal size argument name.
gray	Thick gray outline, light gray background.
colored	Thick blue outline, light blue background.



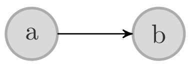
(a) argumentstyle=*standard*



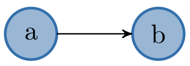
(b) argumentstyle=*large*



(c) argumentstyle=*thick*



(d) argumentstyle=*gray*

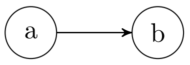


(e) argumentstyle=*colored*

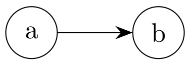
Figure 4: Available options for argumentstyle.

attackstyle=⟨*option*⟩

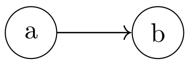
standard	Standard 'stealth' Ti*k*Z arrow tip.
large	Arrow tip is larger and sharper.
modern	Ti*k*Z ModernCS arrow tip.



(a) attackstyle=*standard*



(b) attackstyle=*large*

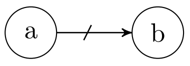


(c) attackstyle=*modern*

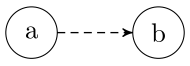
Figure 5: Available options for attackstyle.

supportstyle=⟨*option*⟩

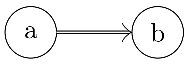
standard	Same tip as attack arrow, perpendicular mark on arrow line.
dashed	Dashed arrow line and same tip as attack arrow.
double	Double arrow line and large flat tip.



(a) supportstyle=*standard*



(b) supportstyle=*dashed*

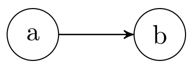


(c) supportstyle=*double*

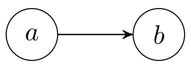
Figure 6: Available options for supportstyle. Note that for *standard* and *dashed* the arrow tip of the selected attackstyle will be used.

namestyle=⟨*option*⟩

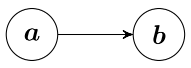
none	No effect applied to argument name.
math	The argument name is rendered as *math* text. (name must be given without mathmode).
bold	The argument name is rendered in ***bold***. (name must be given without mathmode).
monospace	The argument name is rendered in `monospace` font. (name must be given without mathmode).
monoemph	The argument name is rendered as ***`name`***.



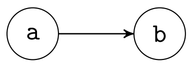
(a) namestyle=*none*



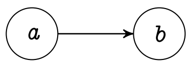
(b) namestyle=*math*



(c) namestyle=*bold*



(d) namestyle=*monospace*



(e) namestyle=*monoemph*

Figure 7: Available options for namestyle. You can of course apply any formatting yourself when using the default namestyle=*none*.

5 Style Parameter Reference

For reference, the style parameters provided by this package are listed below. You may use or redefine them at your own discretion.

`TikZ-keyword`	`style parameters`
argument size	*`contains the currently selected argument size`*
argument	*`contains the currently selected argument style and size`*
argument standard	`circle,inner sep=0,outer sep=0,draw=black`
argument large	`circle,inner sep=0,outer sep=0,draw=black,font=\large`
argument thick	`circle,inner sep=0,outer sep=0,draw=black,line width=0.1em`
argument gray	`argument thick,fill=gray!30,draw=gray!65,text=black!80`
argument colored	`argument thick,fill=aigblue!40,draw=aigblue!80,text=black!80`
attack	*`contains the currently selected attack style`*
attack standard	`-{stealth'}`
attack large	`-{Stealth[scale=1.25]}`
attack modern	`-{To[sharp,length=0.65ex,line width=0.05em]}`
selfattack	`loop,min distance=0.4em,in=0,out=60,looseness=4.5`
support	*`contains the currently selected support style`*
support standard	`attack,postaction={decorate,decoration={...}}`
support dashed	`attack,densely dashed`
support double	`-{Classical TikZ Rightarrow},double`
inactive	`fill=none,draw=gray!50,text=gray!60`
incomplete	`densely dashed`
accepted	`fill=green!40`
rejected	`fill=red!40`
undecided	`fill=cyan!40`
highlight	`fill=aigyellow!60`
invisible	`draw=none,fill=none,opacity=0.0`
standard	`node distance=6.6ex,argument size/.style=minimum size=4.5ex, attack width/.style=line width=0.05em`
small	`node distance=3.5ex,argument size/.style=minimum size=3.4ex, attack width/.style=line width=0.045em`
tiny	`node distance=2.3ex,argument size/.style=minimum size=2.6ex, attack width/.style=line width=0.03em,font=\small`

Table 4: Reference list of TikZ-style parameters provided by the `argumentation` package.

6 Version History

[v1.4 2024/10/31]

- Added functions `\aflabeling`, `\afextension`, `\afreduct` and `\afrestriction` that recreate (parts of) previously created argumentation frameworks. Can be enabled via the package option beamer=true.
- Added internal storage of arguments and attacks of an argumentation framework to enable further computations.
- Added environment `af*` for argumentation frameworks that are unlabeled/uncounted.
- Added command `\setargumentcolorscheme{ }{ }` to change color scheme of the colored argument style.
- Added command `\setafstyle{ }` to set global style options for the AFs.
- Added optional parameter (⟨*value*⟩) to `\attack` command to add a label to the attack edge (undocumented for now).
- Major revision of the documentation.
- Various minor changes to internal functions, naming scheme and comments.

[v1.3 2024/09/25]

- Added support for `\label{ }` and `\ref{ }` to `af` environment.
- Added commands `\AF`, `\arguments`, `\attacks` and `\AFcomplete` to facilitate consistent naming of AFs. Have to be loaded with the package option macros=true.
- Added commands `\afref{ }` and `\fullafref{ }` to reference AFs.
- adjusted scaling of nodes and arrows for larger page sizes.
- added new style options for arguments.
- Various minor fixes and changes regarding the namestyle package option.

[v1.2 2024/06/07]

- Changed Syntax of `\argument` command. The *id* parameter is now given inside parenthesis instead of curly braces and is optional.
- Added absolute positioning to `\argument` command, like for Ti*k*Z nodes.
- Added package option indexing to toggle automatic generation of identifiers for created argument nodes. Can be set to *none*, or selected between *alphabetic* and *numeric* (default).
- All package style options can now also be set locally in the `af` environment.
- Adjusted `\annotatedattack` to require position parameter.
- Various minor bugfixes regarding the namestyle package option.
- Added new argumentstyle large.

[v1.1 2023/12/03]

- Adjusted standard styles.
- Added command for creating annotated attacks.
- Now only provides one environment, which can be parameterised.
- Changed option management to pgfkeys.
- Updated and improved documentation.

[v1.0 2023/11/05]

- First Version.