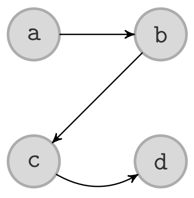
The `argumentation` Package

Lars Bengel[*]

lars.bengel@fernuni-hagen.de

Version 1.3 [2024/09/22]



```
\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}
\end{af}
```

Contents

[*]Please report any issues at `https://github.com/aig-hagen/tikz_argumentation`

1 Quick Guide

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

```
\usepackage{argumentation}
```

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

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

You may want to wrap the `af` environment in a `figure` environment in order to add a caption and reference label.

The easiest way to create an argument in the argumentation framework is:

```
\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 ⟨arg_id⟩]{⟨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 ⟨*arg_id*⟩, 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 $(2, 4)$.

To create an attack between two arguments, you simply use the 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

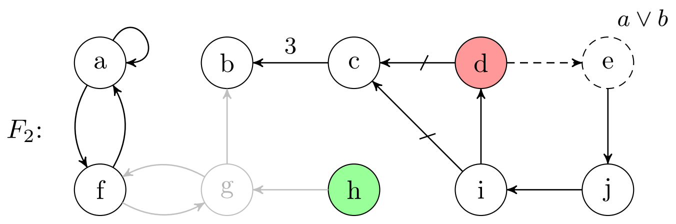


Figure 1: The AF F_2 created with the `argumentation` package.

```
\usepackage{argumentation}
\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{\afref{af:ex1}:} at (-1,-0.9)

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

        \attack[inactive]{a8}{a7}
        \attack[inactive]{a7}{a2}
        \annotatedattack[above]{a3}{a2}{$3$}
        \attack[incomplete]{a4}{a5}
        \attack{a5}{a10}
        \attack{a10}{a9}
        \attack{a9}{a4}

        \support{a4}{a3}
        \support{a9}{a3}
        \label{af:ex1}
    \end{af}
    \caption{The AF \afref{af:ex1} created with the \argumentation package.}
    \label{fig:example}
\end{figure}
```

3 Documentation for Version 1.3 [2024/09/22]

The `argumentation` package provides an easy way for creating argumentation frameworks[1] in LaTeX-documents. It builds on the Ti*k*Z package for drawing the graphs and provides simplified syntax while keeping the same customisation options and keeping full compatibility with all Ti*k*Z commands. The package comes with multiple predefined style options for arguments, attacks and supports. In the following, we give an overview over the functionality of the `argumentation` package. The `argumentation` package can be imported via the command

```
\usepackage{argumentation}
```

Alternatively, one can also adjust the appearance by providing some package options via

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

3.1 Style Options

The `argumentation` package provides the following options to customize the look of the argumentation frameworks. They can both be set globally (as an option for the *usepackage* command) and also locally for each `af` environment (see Section 3.2).

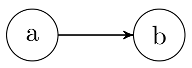
argumentstyle	Customizes the appearance of argument nodes.
attackstyle	Customizes the appearance of attack edges.
supportstyle	Customizes the appearance of support edges.
namestyle	Customizes the font style of the argument names.

In the following, we list the available options for each of the style parameters. Detailed definitions of each of these style parameters can be found in Section 3.8.

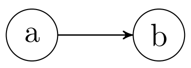
argumentstyle=*⟨option⟩*

The argumentstyle parameter accepts five options

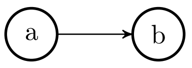
standard	(default) Standard style for the argument nodes.
large	Larger font.
thick	Thick outline.
gray	Thick gray outline, light gray background.
colored	Thick blue outline, light blue background.



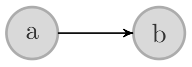
(a) argumentstyle=*standard*



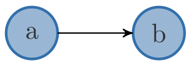
(b) argumentstyle=*large*



(c) argumentstyle=*thick*



(d) argumentstyle=*gray*



(e) argumentstyle=*colored*

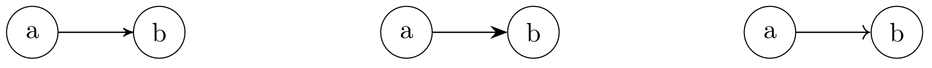
Figure 2: Available options for argumentstyle.

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

attackstyle=⟨*option*⟩

The attackstyle parameter accepts three options

standard	(default) Standard style for the attack arrow tips.
large	Arrow tip is larger and sharper.
modern	Ti*k*Z ModernCS arrow tip.



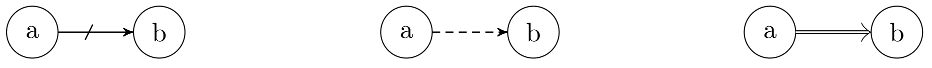
(a) attackstyle=*standard* (b) attackstyle=*large* (c) attackstyle=*modern*

Figure 3: Available options for attackstyle.

supportstyle=⟨*option*⟩

The supportstyle parameter accepts three options

standard	(default) Standard style for the attack arrow tips.
dashed	Dashed arrow line, same tip.
double	Double arrow line and large flat tip.



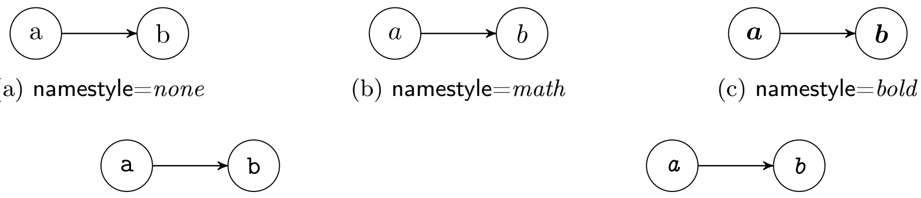
(a) supportstyle=*standard* (b) supportstyle=*dashed* (c) supportstyle=*double*

Figure 4: Available options for supportstyle. Note that for *standard* and *dashed* the arrow tip of the selected attackstyle will be used.

namestyle=⟨*option*⟩

The namestyle parameter offers four different options

none	(default) 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 5: Available options for namestyle. You can of course apply any formatting yourself when using the default namestyle=*none*.

3.2 The `af` Environment

The `argumentation` package provides an environment for creating abstract argumentation frameworks and many of its extensions in LaTeX-documents.

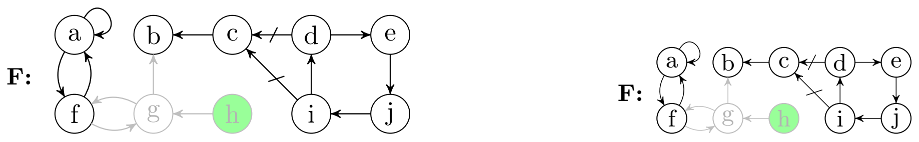
```
\begin{af}[⟨options⟩]
  ⟨environment content⟩
\end{af}
```

The above described style options can be set locally for each instance of the `af` environment by setting the respective parameters in the options of the environment. Local settings override the defaults and globally set values (See Section 3.1 for examples).

The `af` environment extends the tikzpicture environment, meaning all TikZ commands can be used inside the `af` environment as well. Furthermore, all options for the tikzpicture environment can be used for the `af` environment as well, e. g the option `node distance`, which is set to `1cm` (`6.6ex`) per default.

If you want to create an argumentation framework with limited space available, you can use one of the following predefined options for the `af` environment. This is especially useful for two-column layout documents.

tiny	node distance is set to $0.35cm$ and nodes are smaller.
small	node distance is set to $0.55cm$ and nodes are smaller.



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

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

Figure 6: Two AFs with smaller nodes, created by using the small and tiny options of the `af` environment.

3.2.1 Referencing

The package provides to ability to label the created argumentation frameworks and reference them via the `\ref` command. For that, place the command `\label{⟨name⟩}` anywhere in an `af`-environment and you can reference to it via `\ref{⟨name⟩}` anywhere in the document.

Additionally, the following commands are provided to facilitate referencing argumentation frameworks. Most importantly the command `\afref{⟨name⟩}` 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 argumentation frameworks.

`\AF`	F
`\arguments`	A
`\attacks`	R
`\AFcomplete`	$F = (A, R)$
`\afref{af:ex1}`	F_2
`\fullafref{af:ex1}`	$F_2 = (A_2, R_2)$

Table 1: Commands and their respective output.

3.3 Creating Arguments

Arguments can be created with the `\argument` command. The full command is defined as follows

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

⟨*options*⟩ (optional) a list of Ti*k*Z style parameters and/or relative positioning information (see Section 3.3.1).

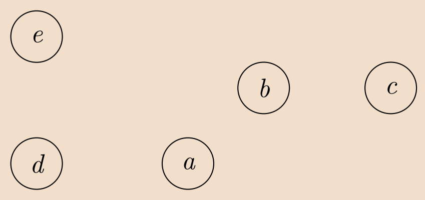
⟨*id*⟩ (optional) the identifier of the new argument. When omitted, arguments will automatically be assigned an identifier of the form: $a1, a2, a3, ...$

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

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

When creating an argument you only have to provide the ⟨*name*⟩, all other parameters are optional. The ⟨*id*⟩ of an argument is used for referencing, e. g., when creating attacks or for the relative positioning of other arguments.

Example 1



```
\begin{af}
    \argument{a}
    \argument{b} at (1, 1)
    \argument[right=of a2]{c}
    \argument(argD){d} at (-2, 0)
    \argument[above=of argD]{e}
\end{af}
```

3.3.1 Relative Positioning

Placement of argument nodes with the `argumentation` package relies on relative placement via the Ti*k*Z-library positioning. The relative positioning information is provided as an optional parameter as follows

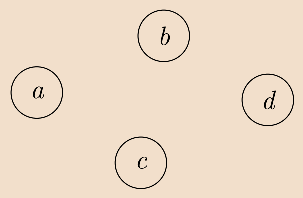
`\argument[`⟨*dir*⟩`=of `⟨*arg_id*⟩`](`⟨*id*⟩`){`⟨*name*⟩`}`

⟨*dir*⟩ has to be one of: *right*, *left*, *below* and *above*

⟨*arg_id*⟩ is the identifier of another argument

Additionally, you can adjust the horizontal/vertical position of an argument via the options `xshift=`⟨*v*⟩ and `yshift=`⟨*v*⟩. The value ⟨*v*⟩ is the horizontal/vertical offset, e. g., `-6.6ex` or `1cm`.

Example 2



```
\begin{af}
    \argument{a}
    \argument[right=of a1,yshift=0.75cm]{b}
    \argument[below=of a2,xshift=-2ex]{c}
    \argument[right=of a3,yshift=5.5ex]{d}
\end{af}
```

3.3.2 Argument Identifiers

When creating an argument, it is automatically assigned an identifier for the sake of simplicity. You can override this identifier by giving another identifier in parenthesis to the `\argument` command. Per default, the argument identifiers will be numerical of the form $a1, a2, a3, \ldots$ based on their order of creation inside the `af` environment. If you prefer alphabetical identifiers, you can set the package option indexing=*alphabetic* and the argument identifiers will instead be $a, b, c, \ldots$.

3.3.3 Additional Argument Styling

Furthermore, you can provide optional parameters to adjust the style of the argument node. For that you can use all Ti*k*Z-style options and additionally the following predefined style parameters (Refer to Section 3.8 for a detailed description of these parameters):

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.

3.4 Creating Attacks

Attacks between two arguments can be created with the command

`\attack{`$\langle arg1\rangle$`}{`$\langle arg2\rangle$`}`

where $\langle arg1\rangle$ and $\langle arg2\rangle$ are the identifiers of two previously defined arguments.

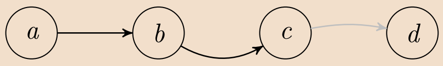
3.4.1 Additional Attack Styles

To customize an attack you can provide additional optional parameters:

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.

Of course, all Ti*k*Z style parameters can be used here as well.

Example 3



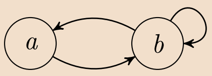
```
\begin{af}
    \argument{a}
    \argument[right=of a1]{b}
    \argument[right=of a2]{c}
    \argument[right=of a3]{d}

    \attack{a1}{a2}
    \attack[bend right]{a2}{a3}
    \attack[bend left=10,inactive]{a3}{a4}
\end{af}
```

Additionally, there is a shortcut for creating a symmetric attack between two arguments with
`\dualattack{⟨arg1⟩}{⟨arg2⟩}`

and a shortcut for a self-attack for an argument with
`\selfattack{⟨arg1⟩}`

For both commands, you can use the same optional parameters as for the `\attack` command.

Example 4



```
\begin{af}
    \argument{a}
    \argument[right=of a1]{b}

    \selfattack{a2}
    \dualattack{a1}{a2}
\end{af}
```

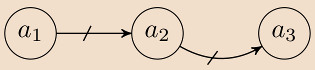
3.5 Creating Support Relations

You can create a support relation between two arguments with the command

`\support{⟨arg1⟩}{⟨arg2⟩}`

where ⟨*arg1*⟩ and ⟨*arg2*⟩ are the identifiers of two previously defined arguments. The support arrow use the same tip as the attack arrows, but have a perpendicular mark to distinguish them from attacks. Supports can be customized in the same way as attacks.

Example 5



```
\begin{af}
    \argument{$a_1$}
    \argument[right=of a1]{$a_2$}
    \argument[right=of a2]{$a_3$}

    \support{a1}{a2}
    \support[bend right]{a2}{a3}
\end{af}
```

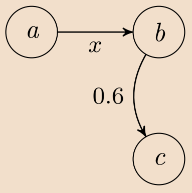
3.6 Annotated Attacks

Many extensions of the original abstract argumentation framework rely on attacks with an associated value. This may, for instance, be probabilities in the case of probabilistic argumentation frameworks or numerical weights in the case of weighted argumentation frameworks. These annotations can be added manually via Ti*k*Z or via the following command

```
\annotatedattack[⟨position⟩]{⟨arg1⟩}{⟨arg2⟩}{⟨value⟩}
```

where ⟨*arg1*⟩ and ⟨*arg2*⟩ are the identifiers of two previously defined arguments and ⟨*value*⟩ is the text or number that should be annotated to the attack. ⟨*position*⟩ specifies where the annotation should be placed relative to the attack arrow and should be one of: above, below, left, right.

Example 6



```
\begin{af}
    \argument{a}
    \argument[right=of a1]{b}
    \argument[below=of a2]{c}

    \annotatedattack[below]{a1}{a2}{$x$}
    \annotatedattack[left,bend right]{a2}{a3}{$0.6$}
\end{af}
```

3.7 Further Commands

If you want to display an name (or some other text) for your argumentation framework in the picture, you can use the `\afname` command. Just like the `\argument` command it can include an optional identifier and supports both relative and absolute positioning. Some example usages:

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

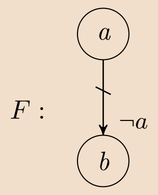
where ⟨*id*⟩ is an (optional) identifier for the created node (if omitted, the default identifier will be *cap*) and ⟨*name*⟩ is the text displayed in the picture. Additional positioning information, via xshift or yshift, can be provided via the optional parameters.

To create an annotation, e. g., an acceptance condition or a weight, next to an argument, the following command can be used.

```
\annotation[⟨optional⟩]{⟨arg_id⟩}{⟨text⟩}
```

where ⟨*arg_id*⟩ is the identifier of some argument and ⟨*text*⟩ is the text to be displayed. Additionally, positioning information, via xshift or yshift, can be provided via the optional parameters.

Example 7



```
\begin{af}
    \argument{a}
    \argument[below=of a1]{b}
    \afname{$F:$} at (-1,-1)
    \annotation[yshift=-0.5cm,xshift=0.4cm]{a2}{$\neg a$}

    \support{a1}{a2}
\end{af}
```

3.8 Style Definitions

You can manually override the argumentstyle, attackstyle and supportstyle parameters and set a custom style globally via the following commands respectively.

```
\setargumentstyle{⟨style⟩}
\setattackstyle{⟨style⟩}
\setsupportstyle{⟨style⟩}
```

where *⟨style⟩* is a list of Ti*k*Z style parameters. For reference, the style parameters provided by this package are listed below. You may use or override them at your own discretion. For instance, one could use them to combine or adapt styles, e. g.,

```
\setargumentstyle{argument large,argument gray,text=white}
```

Ti*k*Z-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,text=black!0,fill=none`

Table 2: Reference list of Ti*k*Z-style parameters provided by the `argumentation` package.

4 Version History

[v1.3 2024/09/22]

- Added support for `\label{ }` and `\ref{ }` to `af` environment.
- Added commands `\AF`, `\arguments`, `\attacks` and `\AFcomplete` to facilitate consistent naming of AFs.
- 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.