The `argumentation` Package

Lars Bengel*
lars.bengel@fernuni-hagen.de

Version 1.1 [2023/12/03]

Contents

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

1 Example

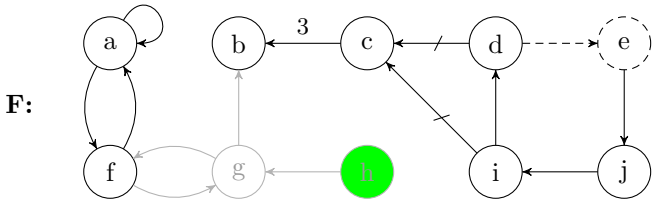


Figure 1: An exemplary AF created with the `argumentation` package.

```
\usepackage{argumentation}
\begin{figure}[ht]
    \centering
    \begin{af}
        \argument{args1}{a}
        \argument[right=of args1]{args2}{b}
        \argument[right=of args2]{args3}{c}
        \argument[right=of args3]{args4}{d}
        \argument[right=of args4,incomplete]{args5}{e}
        \argument[below=of args1]{args6}{f}
        \argument[inactive,right=of args6]{args7}{g}
        \argument[inactive,argin,right=of args7]{args8}{h}
        \argument[right=of args8]{args9}{i}
        \argument[right=of args9]{args10}{j}

        \afname[left of=args1,yshift=-0.8cm,xshift=-0.2cm]{cap}{\textbf{F:}}

        \selfattack{args1}
        \dualattack{args1}{args6}
        \dualattack[inactive]{args6}{args7}

        \attack[inactive]{args8}{args7}
        \attack[inactive]{args7}{args2}
        \annotatedattack{args3}{args2}{$3$}
        \attack[incomplete]{args4}{args5}
        \attack{args5}{args10}
        \attack{args10}{args9}
        \attack{args9}{args4}

        \support{args4}{args3}
        \support{args9}{args3}
    \end{af}
    \caption{An exemplary AF created with the \argumentation package.}
    \label{fig:example}
\end{figure}
```

2 Documentation for Version 1.1 [2023/12/03]

In the following, we give an overview over the functionality of the `argumentation` package. In general, the functionality provided by this package is fully compatible with Ti*k*Z. Meaning every command from this package can be used inside the tikzpicture environment and every Ti*k*Z command or option can be used inside the `af` environment or in context of the argument nodes and attack edges.

2.1 Package Options

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}`

The package provides the following *optional* options to customize the look of the argumentation frameworks.

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

In the following, we list the available options for each of the style parameters.

`namestyle=`⟨*option*⟩

The namestyle parameter accepts three different options

normal	(default) The argument name is rendered normally.
italics	The argument name is rendered in *italics*.
bold	The argument name is rendered in **bold**.
bolditalics	The argument name is rendered with ***both***.
monospace	The argument name is rendered in `monospace` font.

`argumentstyle=`⟨*option*⟩

The argumentstyle parameter accepts two options

standard	(default) Standard style for the argument nodes.

`attackstyle=`⟨*option*⟩
The attackstyle parameter accepts two options

standard	(default) Standard style for the attack arrow tips.
large	Alternative style, arrow tip is larger and sharper.

`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.

You can override the argumentstyle, attackstyle and supportstyle parameters and set a custom style via the following commands respectively.
`\setargumentstyle{`⟨*style*⟩`}`
`\setattackstyle{`⟨*style*⟩`}`
`\setsupportstyle{`⟨*style*⟩`}`

where ⟨*style*⟩ is a list of Ti*k*Z style parameters.

2.2 The `af` Environment

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

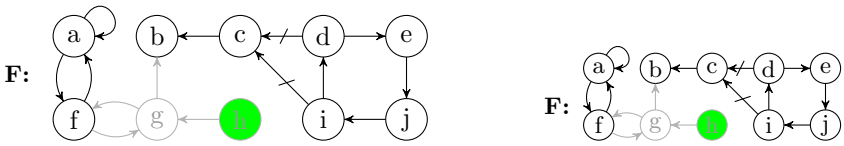
`\begin{af}[`⟨*options*⟩`]`
⟨*environment content*⟩
`\end{af}`

The `argumentation` package provides the `af` environment for creating abstract argumentation framework. The `af` environment extends the tikzpicture environment, meaning all Ti*k*Z 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. For instance, the option parameter `node distance`, which is set to `1cm` per default.

If you want to create an argumentation framework with limited space available, you can provide one of the following predefined options for the 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.

Example 1 *Consider the two AFs in Figure 2 created with the* small *and* tiny *option respectively.*



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

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

Figure 2: Two AFs with smaller nodes, created by using the small and tiny options of the `af` environment.

While the following commands are intended to be used inside the `af` environment, they can also be used inside the tikzpicture environment.

2.3 Arguments

Arguments can be created with the following command

`\argument{`⟨*id*⟩`}{`⟨*name*⟩`}`

⟨*id*⟩ is the identifier of the new argument

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

To create an argument, you must provide a unique identifier and the name to be displayed in the picture. The ⟨*id*⟩ of an argument is then referred to when creating attacks as well as for the relative positioning of other arguments.

2.3.1 Relative Positioning

This package supports relative placement of the arguments via the Ti*k*Z-library positioning. The relative positioning information is provided as an optional parameter via

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

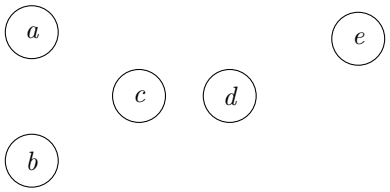
⟨*dir*⟩ has to be one of: *right*, *left*, *below* and *above*

⟨*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., `5pt`, `-3ex` or `0.2cm`.

Example 2

```
\begin{af}
    \argument{arg1}{a}
    \argument[below=of arg1]{arg2}{b}
    \argument[right=of arg1, yshift=-24pt,xshift=-8pt]{arg3}{c}
    \argument[right=of arg3, xshift=-0.5cm]{arg4}{d}
    \argument[right=of arg4, yshift=5ex]{arg4}{e}
\end{af}
```



2.3.2 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:

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.
argin	The argument is displayed with green background color.
argout	The argument is displayed with red background color.
argundec	The argument is displayed with cyan background color.

2.4 Attacks

Attacks between two arguments can be created with the command

`\attack{`⟨*arg1*⟩`}{`⟨*arg2*⟩`}`

where ⟨*arg1*⟩ and ⟨*arg2*⟩ are the identifiers of two previously defined arguments.

2.4.1 Attack Styling

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{arg1}{a}
    \argument[right=of arg1]{arg2}{b}
    \argument[right=of arg2]{arg3}{c}
    \argument[right=of arg3]{arg4}{d}

    \attack{arg1}{arg2}
    \attack[bend right]{arg2}{arg3}
    \attack[bend left=10,inactive]{arg3}{arg4}
\end{af}
```

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

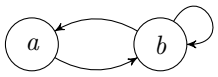
and a shortcut for a self-attack for an argument with
`\selfattack{`⟨*arg1*⟩`}`

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

Example 4

```
\begin{af}
    \argument{arg1}{a}
    \argument[right=of arg1]{arg2}{b}
```

```
    \selfattack{arg1}
    \dualattack{arg1}{arg2}
\end{af}
```



2.5 Supports

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

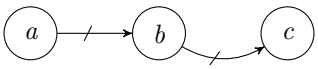
`\support{`⟨*arg1*⟩`}{`⟨*arg2*⟩`}`

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{arg1}{a}
    \argument[right=of arg1]{arg2}{b}
    \argument[right=of arg2]{arg3}{c}

    \support{arg1}{arg2}
    \support[bend right]{arg2}{arg3}
\end{af}
```



2.6 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

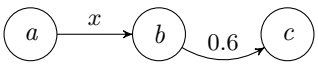
`\annotatedattack{`⟨*arg1*⟩`}{`⟨*arg2*⟩`}{`⟨*value*⟩`}`

where ⟨*arg*1⟩ and ⟨*arg*2⟩ are the identifiers of two previously defined arguments and ⟨*value*⟩ is the number or string that should be annotated to the attack. With this command, the annotation is placed above the attack arrow.

Example 6

```
\begin{af}
    \argument{arg1}{a}
    \argument[right=of arg1]{arg2}{b}
    \argument[right=of arg2]{arg3}{c}

    \annotatedattack{arg1}{arg2}{$x$}
    \annotatedattack[bend right]{arg2}{arg3}{$0.6$}
\end{af}
```



2.7 Further Commands

If you want to display an identifier for your argumentation framework in the picture, you can use the command

`\afname{`⟨*id*⟩`}{`⟨*name*⟩`}`

where ⟨*id*⟩ is an identifier for the created node and ⟨*name*⟩ is the text displayed in the picture. Additionally, positioning information 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.

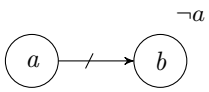
`\annotation{`⟨*arg_id*⟩`}{`⟨*text*⟩`}`

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{arg1}{a}
    \argument[right=of arg1]{arg2}{b}
    \afname[left=of arg1]{caption}{$F:$}
    \annotation[yshift=-0.4cm,xshift=0.4cm]{arg2}{$\neg a$}
```

```
    \attack{arg1}{arg2}
\end{af}
```

F : 

3 Version History

[v1.0 2023/11/05]

- First Version.

[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.