The `cora-macs` Package

Tobias Ladner, Lukas Koller
TUM - Cyber-Physical Systems Group
`tobias.ladner@tum.de, lukas.koller@tum.de`

2024-12-11

Abstract

The `cora-macs` package provides specialized LaTeX tools for researchers in cyber-physical systems. It offers comprehensive commands for mathematical set notation, operations, and other definitions. Designed to accompany the CORA toolbox, the package also facilitates beautiful visualizations through TikZ figures exported from Matlab and a custom color palette.

Contents

1 Introduction

The `cora-macs` package is written to accompany our toolbox CORA[1] for continuous reachability analysis. It is designed to assist with the notation of various set representation and operations often encountered in cyber-physical systems analysis. It also includes predefined color schemes and and an option to export your figures from CORA to LaTeX.

2 Installation

To install the `cora-macs` package, follow these steps:

1. Place the `cora.sty` file in your working directory or in a directory where LaTeX can find it (e.g., in the local `texmf` tree).
2. Include the package in your LaTeX document preamble with the command:

```
\usepackage[options]{cora-macs}
```

The `cora-macs` package provides several options that can be passed when loading the package. These options control the inclusion of different sets of commands.

- `sets`: Enables macros for working with continuous sets (Sec. 3.1).
- `operations`: Enables macros for common operations in set-based computing (Sec. 3.2).
- `nn`: Enables macros related to neural networks (Sec. 3.3).
- `colors`: Defines a set of colors specific to the `cora-macs` package (Sec. 3.4).
- `tikz`: Include your Matlab figures in LaTeX for beautiful visualizations (Sec. 3.5).

Example usage:

```
\usepackage[sets, operations]{cora-macs}
```

---

[1] CORA: `https://cora.in.tum.de/`

3 Commands and Environments

3.1 Continuous Sets (`sets` option)

When the `sets` option is enabled, the package provides several commands for defining and manipulating continuous sets.

- `\contSet{name}`: Defines a set in calligraphic font, e.g., $\mathcal{S}$.
- `\shortI{a}{b}`: Defines a closed interval $[a, b]$. For example, the command `$\contSet{I}=\shortI{1}{2}$` results in $\mathcal{I} = [1, 2]$.
- `\defZ`: States the definition of a zonotope. For example,

  ```
  \begin{equation}
      \contSet{Z}=\shortZ{c}{G}=\defZ
  \end{equation}
  ```

  results in

$$\mathcal{Z} = \langle c, G \rangle_Z = \left\{ c + \sum_{i=1}^{p} \beta_i G_{(\cdot,i)} \;\middle|\; \beta_i \in [-1, 1] \right\} \tag{1}$$

Similar commands also exist for other set representations, which are accessible with its respective abbreviation. Please use the abbreviation `I` for intervals, `Z` for zonotopes, and `PZ` for polynomial zonotopes.

3.2 Operations (`operations` option)

Enabling the `operations` option introduces a variety of operations that are useful in mathematical and cyber-physical systems contexts.

- `\operator{name}{args}`: Defines a custom operator with the given name and arguments.

Some operations are already defined within the CORA package for reference. For example,

- `\opEnclose{S1}{S2}`: Computes the enclosure of two sets.
- `\opIntervalEnclosure{S}`: Specifies an interval enclosure.
- `\opProject{S}`: Projects a set onto a given dimension.
- `\diag{matrix}`: Results the diagonal matrix of the given input.

The commands ensure that the surrounding parenthesis are large enough to capture the content. As this can look weird when used inline within text, one can use the option `inline` to disable the automatic scaling of the parenthesis. For example,

```
Given two sets $\contSet{S}$ and $\widetilde{\contSet{S}}$,
the enclosure is computed using $\opEnclose{\contSet{S}}{\widetilde{\contSet{S}}}$.
As this does not look very nice,
let us try the \texttt{inline} option instead: $\opEnclose[inline]{\contSet{S}}{\widetilde{\contSet{S}}}$.
```

results in

> Given two sets $\mathcal{S}$ and $\widetilde{\mathcal{S}}$, the enclosure is computed using $\texttt{enclose}\left(\mathcal{S}, \widetilde{\mathcal{S}}\right)$. As this does not look very nice, let us try the `inline` option instead: $\texttt{enclose}(\mathcal{S}, \widetilde{\mathcal{S}})$.

3.3 Neural Networks (`nn` option)

The `nn` option introduces commands related to neural networks, including notation for layers, inputs, outputs, and propagations.

- `\NN`: Represents the symbol for a neural network Φ.
- `\nnLayer[name]{index}{input}`: Represents a neural network layer with a specified input. Thus `$\nnLayer[LIN]{k}{x}$` results in $L_k^{\mathrm{LIN}}(x)$. Use `\nnLayerName[name]{index}` to display the layer without explicit input.

One also often wants to state the propagation of an input vector or a set through the network. For example,

```
Given an input $\nnInput\in\R^{\numNeurons_0}$ and a neural network $\NN$,
the output $\nnOutput=\NN(\nnInput)\in\R^{\numNeurons_\numLayers}$ is computed by:
\begin{align}
    \begin{split}
        \nnHidden_0 &= \nnInput, \\
        \nnHidden_k &= \nnLayer{k}{\nnHidden_{k-1}}, \quad k\in[\numLayers], \\
        \nnOutput &= \nnHidden_\numLayers.
    \end{split}
\end{align}
```

results in

Given an input $x \in \mathbb{R}^{n_0}$ and a neural network Φ, the output $y = \Phi(x) \in \mathbb{R}^{n_\kappa}$ is computed by:

$$\begin{aligned} h_0 &= x, \\ h_k &= L_k\left(h_{k-1}\right), \quad k \in [\kappa], \\ y &= h_\kappa. \end{aligned} \tag{2}$$

Similarly, the corresponding set commands `\nnInputSet`, `\nnHiddenSet`, and `\nnOutputSet` can be used to describe the set propagation of the input:

Given an input set $\mathcal{X} \subset \mathbb{R}^{n_0}$ and a neural network Φ, the output $\mathcal{Y} = \Phi(\mathcal{X}) \subset \mathbb{R}^{n_\kappa}$ is computed by:

$$\begin{aligned} \mathcal{H}_0 &= \mathcal{X}, \\ \mathcal{H}_k &= L_k\left(\mathcal{H}_{k-1}\right), \quad k \in [\kappa], \\ \mathcal{Y} &= \mathcal{H}_\kappa. \end{aligned} \tag{3}$$

3.4 Colors (`colors` option)

The option `colors` define a variety of colors that can be used in your documents and is especially useful in combination with the `tikz` option (Sec. 3.5). For example, these colors are available:

- `CORAcolor1`-`CORAcolor7`: Cycles through the default colors of CORA (🟦🟧🟨🟪🟩🟦🟥).
- `CORAcolorBlue`, `CORAcolorRed`, ...: Specifies default color values used in CORA (🟦🟧🟨🟪🟩🟦🟥).
- `CORAcolorReachSet`: A predefined color for reachable sets (🟦). If two reachable sets are compared in one plot, one case use `CORAcolorReachSet2` (🟦). The initial set and simulations can be plotted with `CORAcolorInitialSet` (⬜) and `CORAcolorSimulations` (⬛), respectively.
- `CORAcolorSafe`: A predefined color for safe sets (🟩).
- `CORAcolorUnsafe`: A predefined color for unsafe sets (🟥). As this color can be intimidating when used at large scale, there is also the `CORAcolorUnsafeLight` color (🟥).
- `TUMblue`: TUM corporate blue color (🟦).

3.5 Tikz Figures (`tikz` option)

Visualizations are a crucial part in communicating your scientific findings. We found that the best option to do so is using tikz[2][3] figures. The `tikz` option provides you with a wide variety of useful functionality.

3.5.1 Introduction

Let us create a small TikZ graphic in a file `./figures/mytikzfigure.tikz`:

```
\begin{tikzpicture}
    \draw[CORAcolorBlue] (-6,0) rectangle (-4,2);
    \draw[CORAcolorRed] (-2,1) circle (1cm);
    \draw[CORAcolorYellow] (0,0) -- (0,1) -- (1,2) -- (2,2) -- (2,1) -- (1,0) -- (0,0);
\end{tikzpicture}
```

To include the figure in the main document, please use the following commands:

```
\begin{figure}
    \centering
    \includetikz{./figures/mytikzfigure}
    \caption{My first TikZ figure.}
    \label{fig:mytikzfigure}
\end{figure}
```

To compile the document, you need to enable shell-escape[4]. Usually, this is achieved by adding the compiler argument `--shell-escape` when compiling the document. For example, run `pdflatex --shell-escape main.tex`. The code above then results in Fig. 1 within your document.

Shell-escape allows LaTeX to *externalize* the tikz figures, i.e., creating a stand-alone PDF file which it can use in future compilations to speed up the compilation. The PDF file is located at `./figures/externalize/figures/mytikzfigure.pdf`, which you can also use outside of your LaTeX document. In case something is wrong with your tikz figures, you can also find a log file within the same directory.

If you do not want to externalize your tikz image, you can use the `noexport` option:

```
\begin{figure}
    \centering
```

---

[2] Manual: `http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`

[3] Example: `https://www.overleaf.com/learn/latex/TikZ_package`

[4] Shell-escape: `https://tex.stackexchange.com/questions/598818/how-can-i-enable-shell-escape`

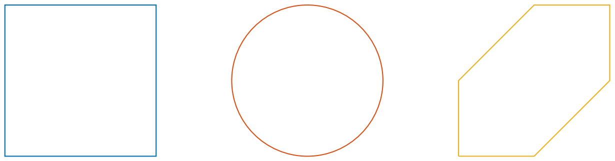

Figure 1: My first TikZ figure.

```
    \includetikz[noexport]{./figures/mytikzfigure}
    \caption{My first TikZ figure.}
\end{figure}
```

If you cannot enable shell-escape but have access to the externalized PDF files, you can use `\CORAexternalizeusepdf` to directly use the PDF files instead of the TikZ files.

3.5.2 Exporting Figures from Matlab/CORA

As this package is designed to assist you with our toolbox CORA[5], we also provide an option to export your Matlab figures to LaTeX named `matlab2tikz`[6]. The resulting figures are then in TikZ/PGF[7] format. Please ask for access if you want to use this option.

After clone the repository and adding it to the Matlab path, you can convert your current open figure with

```
path = 'mymatlabfigure.fig';
savefig(path);
autoConvertToTikz(path);
```

which makes use of the underlying `matlab2tikz` function by setting some optimization options. Type

```
open autoConvertToTikz
```

for details. After successful conversion, you should find a file `mymatlabfigure.tikz` in the current directory. This figure can then be moved to your LaTeX project and added to your document as above:

```
\begin{figure}
```

---

[5] CORA: `https://cora.in.tum.de/`

[6] matlab2tikz: `https://gitlab.lrz.de/cora/matlab2tikz`, fork from: `https://github.com/matlab2tikz/matlab2tikz`

[7] TikZ/PGF Manual: `http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`

```
    \centering
    \includetikz{./figures/mymatlabfigure}
    \caption{My first Matlab figure.}
    \label{fig:mymatlabfigure}
\end{figure}
```

which results in Fig. 2.

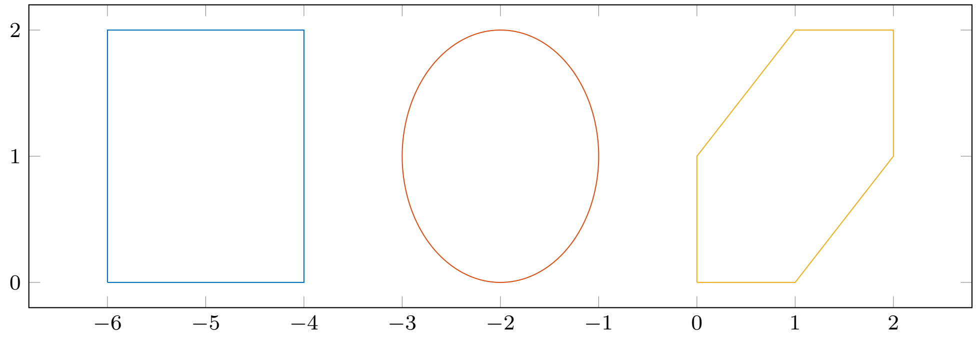

Figure 2: My first Matlab figure.

If you have a figure with multiple subplots, `matlab2tikz` creates multiple TikZ files: The main TikZ file contains all settings and plotting options of your figure, the content of each subplot is exported in a separate file, and an additional file for the legend of your figure is generated. Such a figure can be added as above but you have to specify the path to the TikZ files within the main TikZ file, e.g.,

```
\def\basepath{./figures/}
```

An example figure is visible in Fig. 3.

Your figures can then be customized as required, e.g., adapting the color (also see Sec. 3.4), adding labels of the x- and y-axis, and adapt the legend entries. Please visit the TikZ/PGF manual[8] for details. We have also found that large language models like ChatGPT work surprisingly well with TikZ, so you might want to consult them about specific requests.

---

[8] TikZ/PGF Manual: `http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`

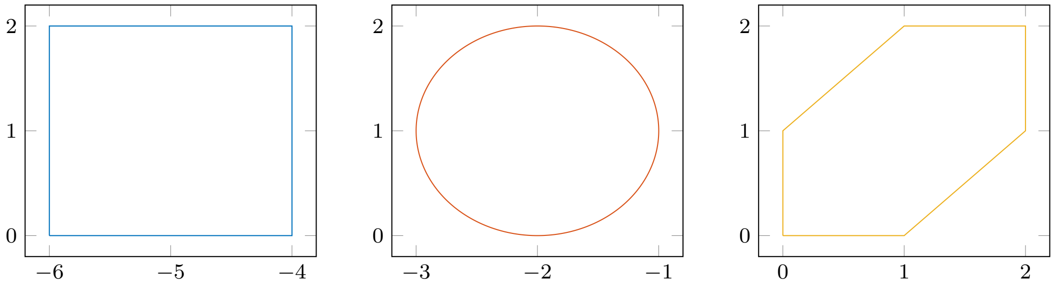

Figure 3: My first Matlab figure with multiple subplots.

3.5.3 TikZ, Git, and Overleaf

If you synchronize your LaTeX document with git, we recommend you to also synchronize the externalized PDF files. You can add `*.log`, `*.dpth`, and `*.md5` to your `.gitignore` and only need to synchronize the PDF itself.

If you want to use this TikZ figures in Overleaf[9], we recommend you to build the figures locally and upload the PDF files into the correct folder. While you can also externalize the figures directly in Overleaf[10], this has several disadvantages, e.g., having to rebuild all figures when re-opening the document after some time. Note: TUM students can also use the TUM instance of Overleaf, which is called ShareLaTeX[11]. There, they can create a git repository out of the project (Menu > Git) and clone the repository to interact with the project locally.

3.5.4 Animated Tikz Figures in Presentations

Generally, you can use the same commands as above to include your TikZ figures in presentations[12]. However, this package also provides some nice commands to animate your figures. We provide an example presentation showcasing these options along with the respective animated TikZ figures in `./examples/presentation.tex`.

To enable animations, please specify the number of overlays of the current frame and add the `animate` option when including a TikZ figure:

```
\begin{frame}<1-4>{Introduction to \CORA}
    \CORA provides you with a wide range of plotting capabilities:
```

---

[9] Overleaf: `https://overleaf.com/`

[10] Overleaf and TikZ: `https://tex.stackexchange.com/questions/596557/overleaf-cache-not-working-properly-with-tikzexternalize-and-include`

[11] ShareLaTeX: `https://sharelatex.tum.de/`

[12] Document class `beamer`: `https://www.overleaf.com/learn/latex/Beamer`

```
    \begin{figure}
        \includetikz[animate]{./figures/mymatlabfigure}
    \end{figure}
\end{frame}
```

You can then animate a plot using the `visible on`/`invisible on` options. For example,

```
\addplot [color=CORAcolor1, visible on=<2->]
```

only shows the respective plot from the second overlay onwards. You can also use `\only<>{}` to display certain text only on certain slides. For example

```
\only<1|handout:0>{My secret message.}
```

only shows this text on the first overlay of the current frame but not if the document is set to `handout` mode[13]. This option can also be used inside TikZ figures, e.g., to specify node labels depending on the current overlay.

Mode advances animations can be achieved using the `alt` option, which allows to alter the plot style depending on the current overlay (if-then-else). For example,

```
\nextgroupplot[..., plotstyle1/.append style={alt=<1>{fill=CORAcolor1}{fill=none}}]
```

fills all objects with `plotstyle1` in the first overlay but not in subsequent overlays. This changes the style for each subplot. Alternatively, the style can also be animated across all subplots.

```
  \pgfplotsset{
    plotstyle1/.style={...,alt=<2->{}{draw=none},alt=<2|handout:0>{fill=CORAcolor1!20}{}}
  }
```

3.5.5 Drawing Neural Networks

This package also provides an option to draw neural networks:

```
\begin{tikzpicture}
    % draw network
    \pic[] at (0,0) { nn={3}{2,5,5,2}{0.15cm}{0.02cm} };
```

---

[13] Beamer `handout` mode: `https://tex.stackexchange.com/questions/473485/latex-beamer-handout-beamer-mode-with-overlays`

```
    % label the third node in the second layer with a '#'
    \node also [neuron label={\#}] (l2-3);
\end{tikzpicture}
```

which results in Fig. 4.

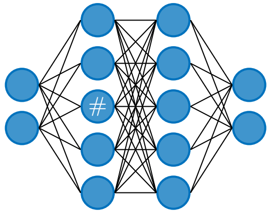

Figure 4: Example neural network with a neuron label.

4 Examples

Here are some examples demonstrating how to use the commands from the `cora-macs` package. We recommend to define LaTeX commands for each variable you use in your document.

4.1 Set Notation

For example, to denote the initial set, you can define

```
\newcommand{\initialSet}{\contSet{X}_0}
```

within your preamble and then use it within your document as follows:

```
\begin{equation}
    \initialSet = \shortZ{c}{G}.
\end{equation}
```

which results in

$$\mathcal{X}_0 = \langle c, G \rangle_Z \,. \tag{4}$$

4.2 Operations

To define a new operation, simply add

```
\newcommand{\opLinComb}[3][]{\operatortt[#1]{linComb}{#2,#3}}
```

to your preamble and then use it within your document as follows:

```
Given two sets $\contSet{S}_1,\contSet{S}_2\subset\R^n$, their linear combination is defined as
\begin{equation}
    \opLinComb{\contSet{S}_1}{\contSet{S}_2} \coloneqq
        \left\{ \lambda s_1 + (1-\lambda s_2)\ \middle|\
            s_1\in\contSet{S}_1,\,s_2\in\contSet{S}_2,\,\lambda \in[0,1] \right\}.
\end{equation}
```

which results in

> Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, the linear combination is defined as
>
> $$\texttt{linComb}\left(\mathcal{S}_1, \mathcal{S}_2\right) \coloneqq \left\{ \lambda s_1 + (1-\lambda s_2) \,\middle|\, s_1 \in \mathcal{S}_1,\, s_2 \in \mathcal{S}_2,\, \lambda \in [0,1] \right\}. \tag{5}$$

4.3 Neural Networks

To specify a specific network layer, one can write

```
A linear layer is defined as
\begin{equation}
    \nnLayer[LIN]{k}{\nnHidden_{k-1}} \coloneqq W\nnHidden_{k-1} + b,
        \quad W\in\R^{\numNeurons_{k}\times\numNeurons_{k-1}},\, b\in\R^{\numNeurons_k}.
\end{equation}
```

which results in

> A linear layer is defined as
>
> $$L_k^{\mathrm{LIN}}\left(h_{k-1}\right) \coloneqq W h_{k-1} + b, \quad W \in \mathbb{R}^{n_k \times n_{k-1}},\, b \in \mathbb{R}^{n_k}. \tag{6}$$

5 Concluding Remarks

The `cora-macs` package offers a comprehensive set of tools for researchers working in cyber-physical systems. By providing a consistent notation and color scheme, this package simplifies the process of documenting complex mathematical objects and operations. We very much encourage you to extend the functionality of this package and define your own commands for everything you need to improve the consistency throughout your LaTeX document. If you have suggestions to include certain commands in this package, please contact us and we are happy to discuss them! For more information, visit the TUM CPS Group website.