UiT The Arctic University of Norway

<faculty>
<department>

<title>

<author>

<degree> thesis in <major> — December 2024

Supervisors

Main Supervisor:	Navn Navnesen	UiT The Arctic University of Norway, Faculty of Science and Technology, Department of Computer Science
External Supervisor:	Kari Nordmann	External Company A/S

This document was typeset with Typst using the *modern UiT thesis* template
© 2024 – Moritz Jörg and Ole Tytlandsvik
https://github.com/mrtz-j/typst-thesis-template

“The problem with object-oriented languages is they’ve got all this implicit environment that they carry around with them. You wanted a banana but what you got was a gorilla holding the banana and the entire jungle.”
— Joe Armstrong

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

CPU Central Processing Unit

UiT University of Tromsø – The Arctic University of Norway

1

Introduction

This typst template is intended for use by UiT students to write their Bachelor's or Master's thesis, or any other document requiring a similar format. Although it meets the official requirements set by the UiT, this template is **unofficial**, and students should verify with their supervisor whether it can be used to typeset the thesis or not. The styling and layout for this thesis template has been largely inspired by Erlend Graff's LaTeX template.[1]

In addition to demonstrating how this template will typeset your content, this document is meant to serve as a helpful reference for how to use its features. The remaining chapters will present a number of useful examples and should be useful for users regardless of their experience with typst.

[1] see https://github.com/egraff/uit-thesis

2

Basic Usage

This chapter will go over the template structure and its basic usage. Users should note that the file structure discussed here is merely a recommended starting point and not required for using the template package.

2.1 Template structure

As opposed to lightweight and uncomplicated report templates you may be familiar with if you have used typst or LaTeX before, this template has a slightly more involved *file structure*. Instead of writing all content in one large `thesis.typ` file, each chapter is written into its own file and imported in `thesis.typ` instead. These chapters are placed in their own directory.

Listing 2.1 shows a tree view of the default file structure of this template. In addition to the `chapters` directory, there is also one for figures. Here you can neatly store all your `.svg`, `.png` or `.jpg` files and reference them in the chapters. Alternatively, some students might prefer to organize further with a directory for each chapter for both typst content and figures, when their thesis grows in size.

Another important file to note is `refs.bib`. This is where you put your BibTeX entries that will produce your bibliography, just like you are used to when working with LaTeX.

```
template
├── chapters
│   ├── basic-usage.typ
│   ├── figures.typ
│   ├── global.typ
│   ├── introduction.typ
│   ├── typst-basics.typ
│   └── utilities.typ
├── figures
│   ├── dining_philosophers.png
│   ├── philosophers.png
│   ├── plot_serial.svg
│   └── uit_aurora.jpg
├── refs.bib
├── thesis.pdf
├── thesis.typ
└── utils
    ├── caption.typ
    ├── feedback.typ
    ├── form.typ
    ├── subfigure.typ
    ├── symbols.typ
    └── todo.typ
```

Listing 2.1 – File structure tree view

2.2 Getting Started

In order to get started using this template document class for your thesis, you can start off with the template you are reading right now right from the typst webapp[2]. Very similar to Overleaf, it is an online editor which conveniently compiles and displays your document as you write, and allows for easy online access for your supervisor. You can also edit the document simultaneously with your co-author if you have one. The typst webapp lets you browse templates and will initialize this template for you when you select it.

If you want to work with typst locally in your favorite text editor instead, make sure you have `typst` installed and run

[2]see https://typst.app°

`typst init @preview/modern-uit-thesis:0.1.1 my-thesis` and the template will be initialized into `my-thesis`. Now you can compile the document using `typst compile`, or `typst watch` to automatically reload as you make changes to it.

Starting in `thesis.typ`, we can see a function call to a function `thesis`. This is how the thesis template style is applied to the document. There are a number of parameters that can be sent into this invocation both to provide special content like title, abstract or list of abbreviations as well as additional customization details. The default arguments demonstrated in `thesis.typ` should give you an idea of the usage.

We recommend you follow along in the typst code for each chapter as you read them in order to discover how you can leverage the useful features demonstrated there yourself.

3

Figures

This chapter will demonstrate how to insert, manipulate and reference figures of various types. The functionality offered by typst to work with figures is powerful and relatively intuitive, especially if you're coming from LaTeX. However, this template also features a few additional lightweight packages to further simplify working with figures.

3.1 Images

Typst allows us to render images in formats such as `png`, `jpg` and `svg` out of the box. Using the vector graphic format `svg` is recommended when applicable, for instance when inserting graphs and plots, as it ensures good quality and readability when printed or viewed on a large screen. Be aware that `svg` images may render differently when rendered on different PDF viewers in some cases.

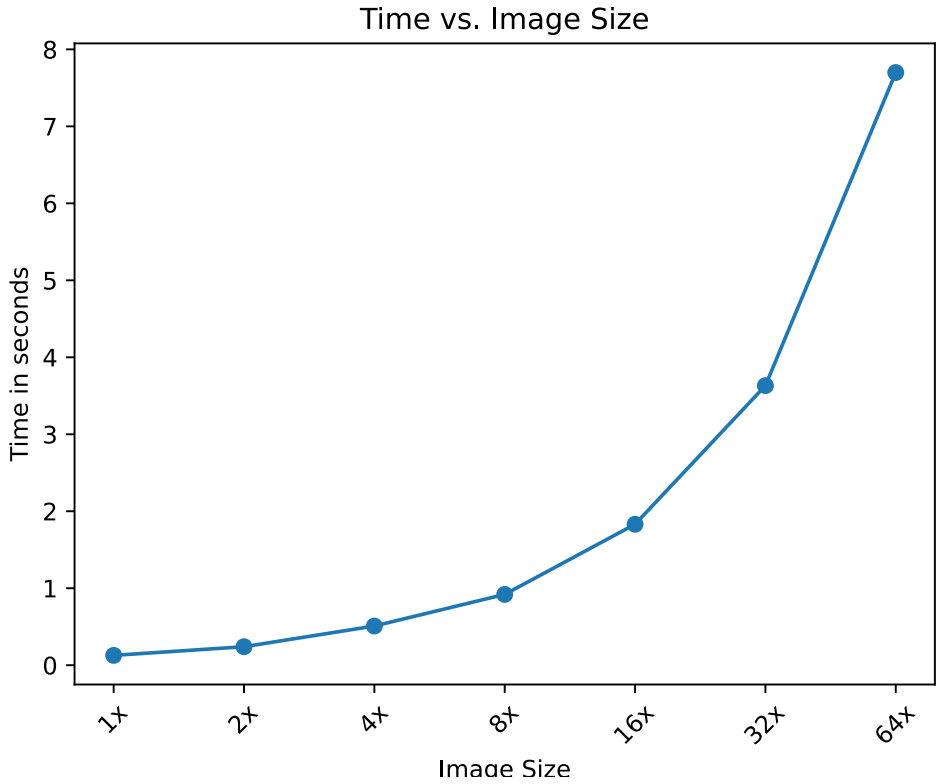


Figure 3.1 – A plot from python's matplotlib, exported to svg

Inserting a figure in typst is very simple, and we can now easily refer to Figure 3.1 anywhere in the document. We can also easily customize the image, for instance by adjusting the width of it so that it doesn't take up as much space, like Figure 3.2. The typst documentation[3] covers images more in depth.

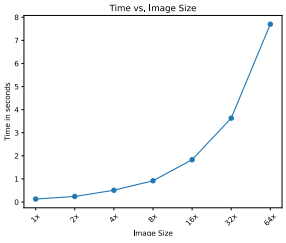


Figure 3.2 – A smaller version of the figure

[3] see https://typst.app/docs/reference/visualize/image/

3.2 Tables

Creating a basic table with typst is quite simple, yet we can also customize them a great deal if we would like to. This thesis template also has some custom default styling for tables, to make the stroke gray and headers distinct.

store	location	open sundays
Coop Extra	Breivika	No
Joker	Dramsvegen	Yes
Rema 1000	Ki	No
Coop Obs	Jekta	No

Table 3.1 – Table with default styling

While Table 3.1 is a very simple table with no extra styling, Table 3.2 is more advanced, using bold for the headers and letting them span multiple rows/columns. Note that we also set the alignment for the text inside the table cells.

Classifier	**Precision**					
	1	2	3	1&2	1&3	All
Perceptron	0.78	0.82	0.24	0.81	0.77	0.83
Decision Tree	0.65	0.79	0.56	0.75	0.65	0.73
One-Class SVM	0.74	0.72	0.50	0.80	0.73	0.85
Isolation Forest	0.54	0.51	0.52	0.53	0.54	0.53

Table 3.2 – A slightly more elaborate table

On a page break, a table can also break and continue on the subsequent page. If a table header and/or footer is set, like in Table 3.3, these will also repeat on both pages by default.

Week	Distance (km)	Time (hh:mm:ss)
1	5	00:30:00
2	7	00:45:00
Goal	*42.195*	*02:45:00*

Week	Distance (km)	Time (hh:mm:ss)
3	10	01:00:00
4	12	01:10:00
5	15	01:25:00
6	18	01:40:00
7	20	01:50:00
8	22	02:00:00
…	…	…
Goal	*42.195*	*02:45:00*

Table 3.3 – A table that breaks with the page

We can also override the default styling to customize tables. Table 3.3 sets a custom fill color for the header and Table 3.4 uses `table.hline()` to enable the border stroke on certain lines only. The second column in Table 3.4 is also set to fill all space available to it.

09:00	Badge pick up
09:45	Opening Keynote
10:30	Talk: Typst's Future
11:15	Session: Good PRs
Noon	*Lunch break*
14:00	Talk: Tracked Layout
15:00	Talk: Automations
16:00	Workshop: Tables
19:00	Day 1 Attendee Mixer

Table 3.4 – A table with no border stroke

There is a lot more customization to be done with tables. Read the official table guide[4] to discover how to create a table by reading a `csv` file with typst, achieving zebra highlighting and much more.

[4] see https://typst.app/docs/guides/table-guide/°

3.3 Listings

For code listings, this template uses a third party package called **codly**[5] in order to provide some out of the box styling and proper syntax highlighting. Unless you want to customize the appearance you don't need to touch codly at all, simply create a normal code block like you would in markdown.

```rust
pub fn main() {
  println!("Hello, world!");
}
```

Listing 3.1 – Hello world! in rust

By default, code listings are configured with zebra lines, line numbering and a label displaying the programming language, like the rust snippet in Listing 3.1. If we want to, we can disable or customize these features locally using the codly `#local()` function, demonstrated with Listing 3.2. Note that too many calls to `#local()` may cause issues, so always use `#codly()` where possible.

```fsharp
[<EntryPoint>]
let main () =
  "Hello, world!"
  |> printfn
```

Listing 3.2 – F# snippet with no zebras and label

We can also skip lines in the code snippet. Note that it doesn't actually skip lines in your snippet, but rather changes the line numbers to represent skipped code. This is demonstrated in Listing 3.3 below.

```c
int main() {
  ...
  printf("Hello, world!");
  return(0);
}
```

Listing 3.3 – C snippet with skipped lines

Codly also allows us to highlight code using line and column positions. Listing 3.4 demonstrates highlighting a line and giving it a tag "assignment".

[5] see https://typst.app/universe/package/codly

```python
if __name__ == "__main__": assignment
  d = {'a': 1}
  print("Hello, world!")
```

Listing 3.4 – Python snippet with highlights

Code snippets can also be imported from files using the `code-block` macro. Listing 3.5 shows a F# snippet imported from a file.

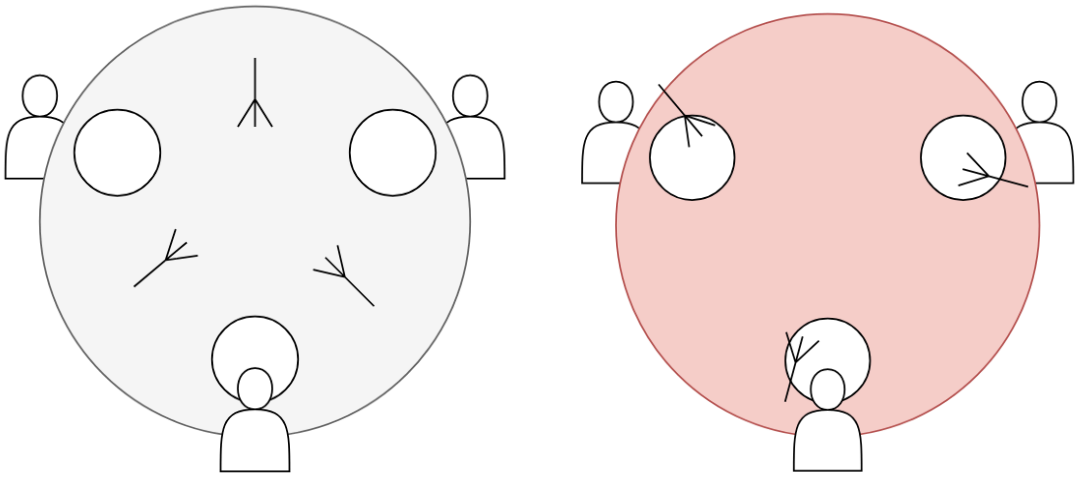
```fsx
open System

let cowsay (message: string) =
    let messageLength = message.Length
    let border = String.replicate (messageLength + 2) "-"
    let cow = @"
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
    "
    printfn $" {border}"
    printfn $"< {message} >"
    printfn $" {border}"
    printfn $"{cow}"

[<EntryPoint>]
let main argv =
    printf "Enter a message: "
    let input = Console.ReadLine()
    cowsay input
    0
```

Listing 3.5 – F# snippet imported from file

3.4 Subfigures

A lot of times we want to display figures side by side and be able to reference them separately as well as together. To make this process easy, this thesis template includes

the **subpar**[6] package. It lets us easily lay out figures in a *grid* while making all labels available for reference.



(a) Subfigure with dining philosophers (b) Subfigure with a deadlock

Figure 3.3 – A figure composed of two subfigures

Now we can refer to Figure 3.3a, Figure 3.3b and the parent Figure 3.3 separately. To access subpar, we use a custom function `#subfigure()` which is included in this template. It's a simple wrapper that sets up the numbering for us to match the rules of the template.

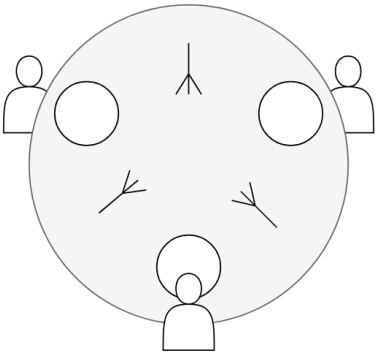
[6] see https://typst.app/universe/package/subpar°

```fsharp
[<EntryPoint>]
let main () =
  "Hello, world!"
  |> printfn
```

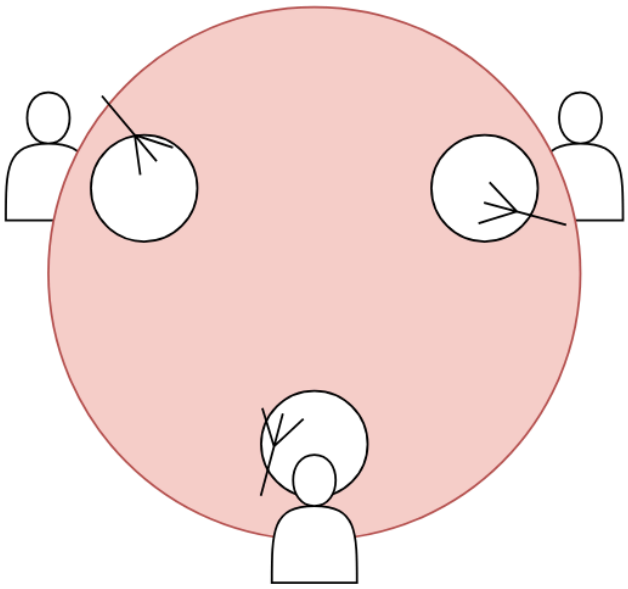
(a) F# snippet in a subfigure

STORE	LOCATION	OPEN SUNDAYS
Coop Extra	Breivika	No
Joker	Dramsvegen	Yes
Rema 1000	Ki	No
Coop Obs	Jekta	No

(b) Table in a subfigure



(c) Subfigure with philosophers



(d) Subfigure with dining philosophers

Figure 3.4 – A figure with multiple subfigure kinds

We can include as many figures as we want in the grid, and even mix and match figure types. Figure 3.4 also has the first column set to a width of `150pt` while the second column is set to take up the remaining space. Note that by default, subfigures do not appear in the List of Figures, and the supplement of referring to for instance Figure 3.4a is *not* "Listing" like we might expect.

Figure 3.5 – A nice picture of UiT, the Arctic University of Norway, under the northern lights. The picture is taken from https://www.wur.nl/en/°.

Another handy function available in this template is the `#dynamic-caption()`, which takes two arguments: a short and a long version of a caption. The long version is displayed under the figure, like in Figure 3.5, however the short version is used in the List of Figures at the start of the thesis.

Using the custom macro `csv-table` it is possible to include data dynamically for csv files. Table 3.5 demonstrates this.

Column 1	Column 2	Column 3
Value 1A	Value 1B	Value 1C
Value 2A	Value 2B	Value 2C
Value 3A	Value 3B	Value 3C

Table 3.5 – A table with data from a csv file

3.5 Equations

Typst has great built-in support for mathematical equations and this template applies numbering to them by default, so that we can refer to Equation (3.1) just like we would a figure.

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \tag{3.1}$$

By default, we can use powerful symbols and functions inside equation blocks (`$ ... $`) to typeset quite advanced equations. For instance, `#attach()` grants us fine control over symbol placement, like in Equation (3.2).

$$\prod_{4+5}^{1} {}_{\beta}^{\alpha}\!\prod_{\lambda}^{2+3} \tag{3.2}$$

Many of the functions have additional parameters to further customize their behavior. For instance, the matrix function allows us to specify the delimiter, see Equation (3.3).

$$\begin{bmatrix} 1 & 2 & \dots & 10 \\ 2 & 2 & \dots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 10 & \dots & 10 \end{bmatrix} \tag{3.3}$$

We can also define our own classes to use within equation blocks, and much more. Refer to the typst reference[7] to see all capabilities.

$$\sqrt[3]{5♠} \in \mathbb{R} \tag{3.4}$$

3.6 Physica

To expand on the already considerable built-in support for math symbols, we've also included the physica[8] package. It makes a far greater range of functions available, allowing us to quickly typeset common symbol sequences without having to build them with the vanilla library. For example, big O notation is easily available:

$$\mathcal{O}(n \log(n)) \tag{3.5}$$

This section covers only a fraction of the available symbols and handy shorthands like expectation value in Equation (3.6), and digital timing diagrams in Equation (3.7). Refer to the full user manual on github[9] to see the full usage.

[7] see https://typst.app/docs/reference/math/

[8] see https://typst.app/universe/package/physica

[9] see https://github.com/Leedehai/typst-physics/

$$\langle \psi | p | \psi \rangle \tag{3.6}$$

clk:  (3.7)

4

Typst Basics

This chapter will go into some basic features of the typst typesetting system, and how to use them in conjunction with this thesis template. Note that you should supplement this with the official guides on the typst website[10].

4.1 Headings

We've already used headings a number of times in this template, and they are intuitive and easy to write. The syntax is markdown-like with `=` instead of `#` serving as its symbol. Chapters are the lowest depth headings, and this template styles them distinctly with a large graphic and new page, like we see here in Chapter 4.

4.1.1 An even deeper heading

We can also create sections of various depths.

4.1.1.1 Super deep heading

Chapter 4.1.1.1 is an example of heading of depth 4. Note that we can also easily give headings labels and reference them anywhere in the document: We can for instance refer to Chapter 1 from here. The same applies to any kind of figure. We should also mention that the prefix in for instance `<subsec:headings>` is only a suggestion to keep your labels organized, you can call it whatever you want.

[10] see https://typst.app/docs/tutorial/°

4.2 Lists

As with headings, typst has syntactic sugar for lists so that we can use simple markdown-like syntax to create them. This goes for both ordered lists:

- One list item
- Another list item
  - A sub-item
  - Another sub-item

…as well as ordered lists:

1. A numbered list item
2. Another one
   - Sub-items can go here too

4.3 Other Nifty Features

Creating **bold**, *italic* and ***bold italic*** text segments is also easily available with markdown-like syntax along with `inline code blocks`. In addition to this, a number of functions are available to achieve most of the textual styling. We can add ~~strikethrough to our text~~, easily add color to it. We get subscripts$_{\text{like this}}$ and superscripts$^{\text{like this}}$. Typst also makes it easy to move characters around, for example to create the LaTeX symbol (Take a look in the function in utils to see how it's achieved).

Footnotes[11] are another useful feature natively supported in typst. Note that any function that takes content (anything inside `[brackets]`) allows us to use other functions in the content we give it. Look for example at this footnote[12]

For further reference, the typst official library reference, tutorial and guides along with the unofficial *Typst Examples Book* found here[13] are great resources.

[11] These are useful for clickable links: https://typst.app/docs/reference/model/footnote/

[12] *We can do **all sorts*** of stuff in here

[13] see https://sitandr.github.io/typst-examples-book/book/about.html

4.4 Citing References

We can cite references defined in our bibliography file easily [1]. Once a reference is cited, it will appear in the bibliography at the end of the thesis.

5

Utilities

Now we will take a look at some useful custom utilities included with this template under `utilities/`. If you find yourself needing some other function for your thesis, it's a great idea to implement it here and include it in your chapters in the same manner.

5.1 Figures

We've already seen the utilities we have implemented for use with figures earlier, refer to Chapter 3.4.

5.2 TODOs and Feedback

Two functions are available for inserting temporary comments into the document to help you while writing your thesis.

TODO: The `#TODO()` function is handy for inserting comments to your future self about your thesis. It has a default yellow color to make it easily visible and prevent you from overseeing it when reviewing the document.

FIXME: Alternatively, you can also pass in another color and/or title to distinguish different types of notes from one another

1. Feedback: Another function is `#feedback()`. It lets your advisor easily insert feedback comments into your document…

Response: …and also lets you add your own response to the note so that you can discuss it in your next meeting. Also note that https://typst.app° has a comment functionality, however it requires a paid subscription.

5.3 Forms

In some cases, you might want to print out your document and leave spots for yourself or others to add data such as signatures to it by hand. This can be achieved using the `#form()` utility function. Leaving the second argument empty, we can leave space for the signature:

Leslie Lamport
Main supervisor

We can also fill in the content with typst by supplying some content:

1980-01-01 – Tromsø, Norway
Date and location

5.4 Abbreviations

In order to easily deal with abbreviations in an automatic and consistent manner, the glossarium[14] package is used in this template. As an input to the thesis function in `thesis.typ` we can supply a dictionary of glossary entries, giving a short version and a long one. These entries will be displayed at the start of the thesis in the List of Abbreviations.

When we refer to an abbreviation the first time in the document, the long version will be printed: University of Tromsø – The Arctic University of Norway (UiT)°. All subsequent references will instead use the short one: UiT°. After we have referred to Central Processing Unit (CPU)° the first time, we might also want to refer to it in plural when writing about multiple CPUs°.

[14] see https://typst.app/universe/package/glossarium/°

Bibliography

[1] O. Qayyum and W. Yu, "Toward Replicated and Asynchronous Data Streams for Edge-Cloud Applications," in *37th ACM/SIGAPP Symposium on Applied Computing (SAC)*, ACM, 2022, pp. 339–346. doi: 10.1145/3477314.3507687°.