Here is a presentation with a very long title aiming at filling several lines in the title page

It is also possible to define a relatively long subtitle such as this one

FirstnameA LastnameA[1] FirstnameB LastnameB[2]
FirstnameC LastnameC[1,2]

[1] Computer Science Lab, Avignon University – LIA EA 4128
{firstname.lastname}@univ-avignon.fr

[2] Institute of Disruptive Innovation, University of Excellence
{firstname.lastname}@univ-excell.fr

November 4, 2019

LABORATOIRE INFORMATIQUE D'AVIGNON

AVIGNON UNIVERSITÉ

Presentation

This is an unofficial adaptation of the official template of Avignon Université, originally proposed only in MS PowerPoint and LO Impress formats at the following address (authentication required):
https://e-doc.univ-avignon.fr/maison-de-la-communication/charte-graphique-de-luniversite/

This presentation file also illustrates how to use basic Beamer features, assuming the reader already knows how to use LaTeX. A more complete (and general) LaTeX tutorial (in French) can be found here:
https://www.overleaf.com/latex/templates/modele-rapport-uapv/pdbgdpzsgwrt
It is also a report template for the same institution.

The LaTeX source code contains additional details under the form of comments.

A French version of these slides can be produced by compiling `main_FR.tex` instead of `main_EN.tex`.

Outline

Section 1

Basics Features

AVIGNON
UNIVERSITÉ

Beamer Class & AU Theme

The original Beamer class takes a number of options, among which:

- **`handout`**: document rendered for printing (disables overlays, dark backgrounds, etc.);
- regular font size: default is **`11pt`**;

The AU Beamer theme proposed here, **`beamerthemeAU`**, also has an option **`light`**, which changes the background of the title and section pages.

The *aspect ratio* of the slides can be controled through parameter **`aspectratio`**. This theme supports two options: **`43`** (4:3, the default value), and **`169`** (16:9).

Pages : It is possible to have long titles spanning two lines, like here

In Beamer, one page of the presentation is called a *frame*, and can be defined through the `frame` environment.

The title and subtitle of a page are set using the `\title` and `\subtitle` commands, respectively. The subtitle can be omitted, as in this page. See next page for an example of subtitle.

The `frame` environment has a number of options, including:

- `plain`: to remove all decorations;
- `allowframebreaks`: to split automatically long content over several pages (eg. bibliography, Section 5);
- `fragile`: when using verbatim environments (eg. `listings`, Page 23);
- Vertical alignment: `c` (center), `t` (top) or `b` (bottom).

Sections
It is possible to add a subtitle below the title, which can also span two lines if needed

Sections and subsections are handled like in regular LaTeX documents, i.e. using the `\section` and `\subsection` commands, respectively.

Using the command `\tableofcontents`, it is possible to automatically fill a page with the table of contents. The option `hideallsubsections` allows including only sections (and not subsections).

The command `\sectionframe` can be used to insert a page containing only the title of the current section, eg. Section 1.

Text Formatting
Standard Commands

Like in regular LaTeX documents, it is possible to define *italic* (`\textit`) or **bold** (`\textbf`) text, or ***both*** (by combining these commands).

One can also use a `monospaced` font (`\texttt`).

It is also possible to use the `\alert` command, which is specific to Beamer, in order to add some color, like here: Alert!

Text Formatting
Theme-Specific Commands

Command `\textsc`, which allows formatting the text using smaller upper-case letters (a.k.a. *small caps*) has no effect, because the font used by the theme does not support small caps. However, the theme proposes command `\fauxsc` in order to simulate this effect: THIS IS HOW THE TEXT LOOKS LIKE WHEN USING THIS COMMAND.

The middle dot used in French for gender-inclusive writting can be obtained thanks to command `\textperiodcentered`, or the shorter `\tpc`. For instance: chercheur·se·s. The user can alternatively directly insert the character by copying and pasting this instance: "·".

The theme contains commands for matching *check* and *cross marks*: ✓ (`\cmark{}`) and ✗ (`\xmark{}`).

Quotations

One can quote text using the **verse** environment, e.g.

> *Quotation, n: The act of repeating erroneously the words of another.*
>
> *–A. Bierce*

Or the **quote** environment:

> *Build a man a fire, and he'll be warm for a day. Set a man on fire, and he'll be warm for the rest of his life.*
>
> *–T. Pratchett*

Mathematical Expressions

In-line equations are defined with `$`, e.g. `$f(x) = ax + b$` is rendered as $f(x) = ax + b$.

Off-line equations, which are numbered, are inserted using the `equation` environment, e.g.

$$f(x) = ax + b. \tag{1}$$

Command `\nonumber` prevents the numbering of the off-line equations, e.g.

$$r(\mathbf{x}, \mathbf{M}) = \frac{\mathbf{x}^\top \mathbf{M} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}}.$$

Columns

Using the `columns` and `column` environments (cf. this page's source code), it is possible to split a page in several independent columns.
This can ease defining the structure of the page.

For `columns`, options `t`, `b`, `c`, `T` control the vertical alignment, and `totalwidth` the global width (see this page for more details).
The width of each column can be specified through the parameter of `column`.

It is possible to put text, but also images in columns.

Figure 1: Logo of Avignon Université

Blocks

Blocks are specific to Beamer, and allow highlighting points of interest. There are three types of blocks, each one withi its own colors:

Regular

`block` environment: used most of the time.

Alert

`alertblock` environment: to highlight particularly important points.

Example

`exampleblock` environment: to show some concrete illustrations of the presented concepts.

One can reduce block size by using commands changing the font size, e.g. `\smaller[2]`, as above with the example block (see source code).
in order to center a formula in a block

Lists

Lists are defined as in LaTeX documents, using the `\item` command and one of the three existing environments:

`itemize`:

- Rock;
  - Paper;
    - Scissors.

`enumerate`:

1. First;
   1. Second;
      1. Third.

`description`:

Yin Yang.

Hot Cold.

Note the use of the `columns` and `column` environments in this page.

Side Notes
Defining Notes

It is possible to associate textual notes to each slide, using the `\note` command and specifying a comment as parameter. Notes can be defined:

- *Inside* a `frame` environment: all notes contained in the same frame are gathered on the same text slide.
- *Outside* frames: each note constitutes a single text slide.

The `\note` command has two options `item` (inside frames) and `itemize` (outside), both allowing to automatically organize comments as a list in the produced text slide. See the notes associated to this page.

Side Notes
Displaying Notes

A number of Beamer options allow controlling how notes are shown:

- `hide notes` (default): notes are not shown in the slides.
- `show notes`: shows notes in additional text slides inserted between regular slides.
- `show only notes`: shows only these additional text slides, and not the regular ones.
- `show notes on second screen=right` (or `left`, `top`, `bottom`): take advantage of package `pgfpages` to enlarge the slides so that notes can be shown on a second screen (and therefore hidden to the audience). An appropriate software is then required to display the presentation, such as Impress!ve or Dual-Screen PDF Viewer.

Note: You can show the notes associated to the previous slide by setting the option accordingly at the begining of this document.

Section 2

Floats & Cross-References

AVIGNON
UNIVERSITÉ

Figures

Figures can be inserted using the **`figure`** environment, as in regular LaTeX documents.

Figure 2: Logo of the Computer Science Lab of Avignon Université

For an image stored in a *file* (JPEG, BMP, PDF, PNG, etc.), as above, one uses the **`\includegraphics`** command.

Diagrams

Besides image files, it is also possible to insert *diagrams* into figures, thanks to the `TikZ` package:

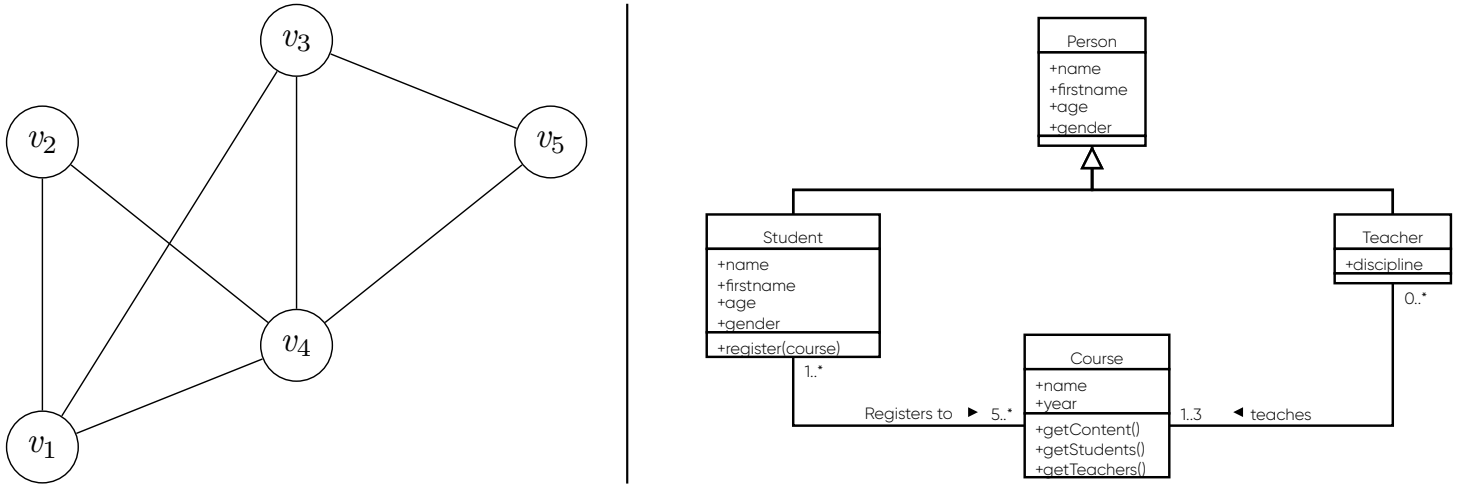

Figure 3: Examples of *graph* (left) and *class diagram* (right).

The left diagram was designed manually, whereas the right one was produced using Dia, an open-source WYSIWYG software able to export diagrams as **`TikZ`** code.

Tables

Similarly to figures, tables can be inserted like in regular LaTeX documents, using the `table` environment.

City	Population
Avignon, FR	92,130
Avignon, QC	15,246
Avignonet-Lauragais, FR	1,443
Avignon-lès-Saint-Claude, FR	390
Avignonet, FR	194

Table 1: Population of a selection of cities.

If the table is too wide, it can be put in a `\resizebox` command, e.g. `\resizebox{\textwidth}{!}{<mytable>}`.

Note: Some argue that one can directly use the `tabular` environment, as captions are not really necessary in such presentations.

CSV-based Tables

Instead of manually defining the content of a table, it is possible to automatically and dynamically retrieve it from a CSV file, thanks to the `csvsimple` package:

Given Name	Patronymic	Family Name	Sex	Age
Fyodor	Pavlovich	Karamazov	M	54
Dmitri	Fyodorovich	Karamazov	M	28
Ivan	Fyodorovich	Karamazov	M	24
Alexei	Fyodorovich	Karamazov	M	19
Pavel	Fyodorovich	Smerdiakov	M	N/A
Agrafena	Alexandrovna	Svietlova	F	22
Katerina	Ivanovna	Verkhovtseva	F	N/A

Table 2: Table retrieved from CSV file **`data/karamazov.csv`**.

Algorithms

It is possible to present algorithms as pseudo-code, thanks to the `algorithm2e` environment based on the `algorithm2e` package:

Data: ℓ: list
Result: m: maximum

```
// Initialization
1 m ← −∞

// Main loop
2 for i ← 1 to length(ℓ) do
3 |   if ℓ[i] > m then
4 |   |   m ← ℓ[i]
5 |   end if
6 end for
```

Algorithm 1: Compute the maximum of a list.

Source Code

One can insert properly formatted source code, thanks to the `lstlisting` environment proposed in the `listings` package:

```
1  // Hello.java
2  import javax.swing.JApplet;
3  import java.awt.Graphics;
4
5  public class Hello extends JApplet
6  {  public void paintComponent(Graphics g)
7     {  g.drawString("Hello: world!", 65, 95);
8     }
9  }
```

Listing 1: Hello World Java applet.

Note: frames containing this environment must be defined using the `fragile` option.

File Content & Console

This theme includes two additional environments based on the `framed` package.

The first one, **`filetext`**, is meant to display the content of some text file:

```
First line in the text file;
second line, with some highlighting here and there;
and here's a third line.
```

File 1: Content of some text file.

The second one, **`consoletext`**, aims at displaying text shown in some terminal or console:

```
invite/>mycommand myparameter &
mycommand:  command not found
invite/>
```

Console 1: Example of command lines.

Note: like for the source code (Page 23), one must select the **`fragile`** option when defining a frame that uses one of these environments.

Mathematical Floats

Beamer contains a number of predefined environments to handle mathematical concepts: `theorem`, `corollary`, `definition`, `fact`, `example`, and `proof`. They are rendered as blocks (cf. Page 13):

Theorem 2.1 (Jörg Neunhäuserer)

In any subset of $M = \{1, 2, ..., 2m\}$ with at least $m + 1$ elements, there are numbers a, b such that a divides b [Neu13].

The option `envcountsect` of the Beamer class allows including the section number in the theorem (or other similar objects) numbers, as above.

Proof.

Let $\{a_1, ..., a_{m+1}\} \in M$ and decompose $a_i = 2^{r_i} q_i$, where the q_i are odd numbers. There are only m odd numbers in M hence one q_i appears in the decomposition of two different numbers a_i and a_j. Now a_i divides a_j if $a_i < a_j$. ∎

General Cross-References

Cross-references work as in regular LaTeX documents, i.e. using commands **\label** and **\ref**. For instance: Figure 2, Table 1, Eq.(1), Theorem 2.1, Page 2, Algorithm 1, Line 4 (in an algorithm), Listing 1, File 1, Console 1.

Hyperlinks are defined as in regular LaTeX documents too, using package hyperref, and in particular its commands **\href** and **\url** (e.g. Page 2).

Footnotes can be inserted using the traditional **\footnote** command, like here[1].

[1] Some footnote.

Bibliographical References

This Beamer theme is set to use `BibLaTeX` and Biber (instead of the traditional BibTeX), which solves all sorts of diacritic-related problems (e.g. presence of accents, cedillas and such in author names or article titles).

The BibTeX file itself is defined as usual, see **`biblio.bib`** for an example. One can use the classic categories of bibliographic entries:

Nature of the reference	BibTeX type	Examples
Journal article	`@Article`	[For10; CLD16]
Conference article	`@InProceedings`	[WC89; MLF08]
Monograph	`@Book`	[Wol98; ML16]
Monograph chapter	`@InBook`	[Mai07; Rei09]
Edited book	`@Collection`	[PRD03; BE05]
Book chapter	`@InCollection`	[Dan+07; LB12]
Technical report	`@TechReport`	[RB09; Par+13]
PhD thesis	`@PhDThesis`	[Won78; Ger10]

Table 3: Main categories of bibliographical entries.

Short bibliographical citations

There are various ways to cite bibliographic references with BibLaTeX.
The main ones are:

- The classic `\cite`, or equivalently (in this specific case) `\parencite`, just inserts the bibliographic code: [CLD16];
- Command `\textcite` additionally mentions the authors' names: Cossu et al. [CLD16];
- Command `\citeauthor` cites only the authors' names: Cossu et al.

Long bibliographical citations

It is also possible to insert full references right here:

- Command **\fullcite** inserts the full reference right away: J.-V. Cossu, V. Labatut, and N. Dugué. "A review of features for the discrimination of Twitter users: application to the prediction of offline influence". In: *Social Network Analysis and Mining* 6 (2016), pp. 1–23. DOI: 10.1007/s13278-016-0329-x. ⟨hal-01203171⟩ ;
- Command **\footfullcite** does the same, but as a footnote[2].

Note that this class supports the **hal** field in BibLaTeX files, which allows referring to HAL-hosted preprints (see the above example).

[2] J.-V. Cossu, V. Labatut, and N. Dugué. "A review of features for the discrimination of Twitter users: application to the prediction of offline influence". In: *Social Network Analysis and Mining* 6 (2016), pp. 1–23. DOI: 10.1007/s13278-016-0329-x. ⟨hal-01203171⟩ .

Section 3

Overlays & Animations

AVIGNON
UNIVERSITÉ

Notion of Overlay

Overlays allow progressively uncovering various parts of a page, instead of showing everything at once. They are quite similar (albeit more limited) to the animations proposed by MS Powerpoint or LO Impress.

Beamer offers two ways of hiding parts of a page:

- *Covered* text appears as some sort of watermark;
- *Invisible* text is completely hidden.

Each step defined to uncover the `frame` results in the generation of a specific slide in the final presentation. Put differently, with overlays, one `frame` is shown as a sequence of several slides.

Pauses

The simplest way to use overlays is to place the `\pause` command wherever one wants to stop displaying the page content. The rest of the content is then covered.

For instance, here is a pause: `\pause` . And here is the end of the paragraph. To be precise, this additional text is on a *different slide*, but on the *same page*: the page number in the footer has not changed.

Pauses

The simplest way to use overlays is to place the **\pause** command wherever one wants to stop displaying the page content. The rest of the content is then covered.

For instance, here is a pause: **\pause** . And here is the end of the paragraph. To be precise, this additional text is on a *different slide*, but on the *same page*: the page number in the footer has not changed.

Pauses

The simplest way to use overlays is to place the **\pause** command wherever one wants to stop displaying the page content. The rest of the content is then covered.

For instance, here is a pause: **\pause** . And here is the end of the paragraph. To be precise, this additional text is on a *different slide*, but on the *same page*: the page number in the footer has not changed.

Note that, when covered, the above text was rendered as some sort of watermark. This can be helpful, for instance, if the speaker does not remember what comes after a **\pause**.

Absolute Overlay Specifications
Description

Beamer redefines certain standard LaTeX commands and environments in order for them to accept additional parameters called *overlay specifications*. These allow assigning numbers to various elements constituting the page. After rendering, these elements appear one after the other, in the specified order –which, again, results in the generation of several slides for this page.

Formally, these parameters are mentioned between angle brackets, e.g. **<3>** means the element appears at the third slide (of the considered page).

Absolute Overlay Specifications
Description

Beamer redefines certain standard LaTeX commands and environments in order for them to accept additional parameters called *overlay specifications*. These allow assigning numbers to various elements constituting the page. After rendering, these elements appear one after the other, in the specified order –which, again, results in the generation of several slides for this page.

Formally, these parameters are mentioned between angle brackets, e.g. `<3>` means the element appears at the third slide (of the considered page). In order to keep the element visible at the subsequent slides, it is necessary to add a hyphen: `<3->`.

Absolute Overlay Specifications
Description

Beamer redefines certain standard LaTeX commands and environments in order for them to accept additional parameters called *overlay specifications*. These allow assigning numbers to various elements constituting the page. After rendering, these elements appear one after the other, in the specified order –which, again, results in the generation of several slides for this page.

Formally, these parameters are mentioned between angle brackets, e.g. `<3>` means the element appears at the third slide (of the considered page). In order to keep the element visible at the subsequent slides, it is necessary to add a hyphen: `<3->`. It is also possible to make the element disappear at some later slide, e.g. `<3-5>`.

Absolute Overlay Specifications
Description

Beamer redefines certain standard LaTeX commands and environments in order for them to accept additional parameters called *overlay specifications*. These allow assigning numbers to various elements constituting the page. After rendering, these elements appear one after the other, in the specified order –which, again, results in the generation of several slides for this page.

Formally, these parameters are mentioned between angle brackets, e.g. `<3>` means the element appears at the third slide (of the considered page). In order to keep the element visible at the subsequent slides, it is necessary to add a hyphen: `<3->`. It is also possible to make the element disappear at some later slide, e.g. `<3-5>`. Finally, one can combine such times ranges using commas, e.g. `<3-5,7-10,15>`.

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Notice how the position of the items is fixed, and independent from whether the other items are shown or not.

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Notice how the position of the items is fixed, and independent from whether the other items are shown or not.

Overlay specifications do not necessarily affect visibility, e.g. `\textbf<7>` or `\textcolor<8>`.

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- `\item<3->`
- `\item<2,4>`

Notice how the position of the items is fixed, and independent from whether the other items are shown or not.

Overlay specifications do not necessarily affect visibility, **e.g.** `\textbf<7>` or `\textcolor<8>`.

Absolute Overlay Specifications
Examples

The `\item` command accepts overlay specifications, for instance:

- `\item<1>`
- `\item<2-3>`
- **`\item<3->`**
- `\item<2,4>`

Notice how the position of the items is fixed, and independent from whether the other items are shown or not.

Overlay specifications do not necessarily affect visibility, e.g. **`\textbf<7>`** or `\textcolor<8>`.

Relative Overlay Specifications
Basic Operators

Instead of explicitly giving slide numbers in overlay specifications, it is possible to proceed *incrementally* using operator `+`. This symbol is replaced by the current value of the slide counter, *then* this counter is incremented by 1. For instance, let C and C' be the current and updated counter values, and S be the slide in which the element appears:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+->`: $C = 2,$ $S = 2,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$

When using `.` instead of `+`, this symbol is replaced by the current counter value *minus one*, and this counter is *not* modified. Basically, this allows displaying simultaneously several elements:

- `\item<+->`: $C = 4,$ $S = 4,$ $C' = 4 + 1 = 5.$
- `\item<.->`: $C = 5,$ $S = 5 - 1 = 4,$ $C' = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<.->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6.$

Relative Overlay Specifications
Basic Operators

Instead of explicitly giving slide numbers in overlay specifications, it is possible to proceed *incrementally* using operator `+`. This symbol is replaced by the current value of the slide counter, *then* this counter is incremented by 1. For instance, let C and C' be the current and updated counter values, and S be the slide in which the element appears:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+->`: $C = 2,$ $S = 2,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$

When using `.` instead of `+`, this symbol is replaced by the current counter value *minus one*, and this counter is *not* modified. Basically, this allows displaying simultaneously several elements:

- `\item<+->`: $C = 4,$ $S = 4,$ $C' = 4 + 1 = 5.$
- `\item<.->`: $C = 5,$ $S = 5 - 1 = 4,$ $C' = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<.->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6.$

Relative Overlay Specifications
Basic Operators

Instead of explicitly giving slide numbers in overlay specifications, it is possible to proceed *incrementally* using operator `+`. This symbol is replaced by the current value of the slide counter, *then* this counter is incremented by 1. For instance, let C and C' be the current and updated counter values, and S be the slide in which the element appears:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+->`: $C = 2,$ $S = 2,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$

When using `.` instead of `+`, this symbol is replaced by the current counter value *minus one*, and this counter is *not* modified. Basically, this allows displaying simultaneously several elements:

- `\item<+->`: $C = 4,$ $S = 4,$ $C' = 4 + 1 = 5.$
- `\item<.->`: $C = 5,$ $S = 5 - 1 = 4,$ $C' = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<.->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6.$

Relative Overlay Specifications
Basic Operators

Instead of explicitly giving slide numbers in overlay specifications, it is possible to proceed *incrementally* using operator `+`. This symbol is replaced by the current value of the slide counter, *then* this counter is incremented by 1. For instance, let C and C' be the current and updated counter values, and S be the slide in which the element appears:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+->`: $C = 2,$ $S = 2,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$

When using `.` instead of `+`, this symbol is replaced by the current counter value *minus one*, and this counter is *not* modified. Basically, this allows displaying simultaneously several elements:

- `\item<+->`: $C = 4,$ $S = 4,$ $C' = 4 + 1 = 5.$
- `\item<.->`: $C = 5,$ $S = 5 - 1 = 4,$ $C' = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<.->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6.$

Relative Overlay Specifications
Basic Operators

Instead of explicitly giving slide numbers in overlay specifications, it is possible to proceed *incrementally* using operator `+`. This symbol is replaced by the current value of the slide counter, *then* this counter is incremented by 1. For instance, let C and C' be the current and updated counter values, and S be the slide in which the element appears:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+->`: $C = 2,$ $S = 2,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$

When using `.` instead of `+`, this symbol is replaced by the current counter value *minus one*, and this counter is *not* modified. Basically, this allows displaying simultaneously several elements:

- `\item<+->`: $C = 4,$ $S = 4,$ $C' = 4 + 1 = 5.$
- `\item<.->`: $C = 5,$ $S = 5 - 1 = 4,$ $C' = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<.->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Relative Overlay Specifications
Offsets

One can also specify an *offset* after `+` or `.`, between parentheses, which will be used to modify the slide counter. Note that this offset can be positive, but also negative.

Here are some examples:

- `\item<+->`: $C = 1,$ $S = 1,$ $C' = 1 + 1 = 2.$
- `\item<+(1)->`: $C = 2,$ $S = 2 + 1 = 3,$ $C' = 2 + 1 = 3.$
- `\item<+->`: $C = 3,$ $S = 3,$ $C' = 3 + 1 = 4.$
- `\item<+(2)->`: $C = 4,$ $S = 4 + 2 = 6,$ $C' = 4 + 1 = 5.$
- `\item<+->`: $C = 5,$ $S = 5,$ $C' = 5 + 1 = 6.$
- `\item<+(-1)->`: $C = 6,$ $S = 6 - 1 = 5,$ $C' = 6 + 1 = 7.$
- `\item<.(1)->`: $C = 7,$ $S = 7 - 1 + 1 = 7,$ $C' = 7.$
- `\item<+->`: $C = 7,$ $S = 7,$ $C' = 7 + 1 = 8.$

Main Overlay Commands
Description

There are also several Beamer-specific commands to define overlays. They take the form `\xxxxx<y>{zzzz}`, where `xxxxx` is the command, `y` is an overlay specification, and `zzzz` is the targeted content.

The main three such commands aim at showing the specified content on the specified slides, and hiding it on the other slides. They differ in the way they hide this content:

- `\visible`: the hidden content is completely *invisible*.
- `\uncover`: the hidden content is *covered*.
- `\only`: the hidden content is outright *removed*.

Main Overlay Commands
Examples

Here is an illustration:

This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

Main Overlay Commands
Examples

Here is an illustration:
This text appears on the 2nd slide using `\only`.
This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

Main Overlay Commands
Examples

Here is an illustration:
This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

Main Overlay Commands
Examples

Here is an illustration:

This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

This text appears on the 4th slide using `\only`. It is located after the space corresponding to the previous text, which was inserted with `\uncover`.

Main Overlay Commands
Examples

Here is an illustration:

This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

This text appears on the 5th slide also using `\only`, in place of the previous text.

Main Overlay Commands
Examples

Here is an illustration:

This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

This text appears on the 6th slide using `\visible`, in place of the previous text.

Main Overlay Commands
Examples

Here is an illustration:

This text appears on the 3rd slide using `\uncover`. It replaces the previous text, since it was inserted using `\only`).

Finally, this text appears on the 7th slide also using `\visible`, and only after the space left by the previous text.

Other Overlay Commands

The `\invisible` command is simply the opposite of `\visible`: it makes the specified content completely invisible on the specified slides, and shows it on the other slides.

The `\onslide` command is quite generic. It takes an additional optional parameter called *modifier*, which can be `+` or `*`. Basically:

- `\onslide` behaves like `\uncover`;
- `\onslide+` behaves like `\visible`;
- `\onslide*` behaves like `\only`.

The `\alt` command allows showing two different contents in the specified slides vs. the other slides. For instance, `\alt<1>{first}{second}` results in: first.

The `\temporal` command works similarly, but distinguishes before/during/after the overlay specification. For instance, `\temporal<2>{first}{second}{third}` results in: first.

Other Overlay Commands

The `\invisible` command is simply the opposite of `\visible`: it makes the specified content completely invisible on the specified slides, and shows it on the other slides.

The `\onslide` command is quite generic. It takes an additional optional parameter called *modifier*, which can be `+` or `*`. Basically:

- `\onslide` behaves like `\uncover`;
- `\onslide+` behaves like `\visible`;
- `\onslide*` behaves like `\only`.

The `\alt` command allows showing two different contents in the specified slides vs. the other slides. For instance, `\alt<1>{first}{second}` results in: second.

The `\temporal` command works similarly, but distinguishes before/during/after the overlay specification. For instance, `\temporal<2>{first}{second}{third}` results in: second.

Other Overlay Commands

The `\invisible` command is simply the opposite of `\visible`: it makes the specified content completely invisible on the specified slides, and shows it on the other slides.

The `\onslide` command is quite generic. It takes an additional optional parameter called *modifier*, which can be `+` or `*`. Basically:

- `\onslide` behaves like `\uncover`;
- `\onslide+` behaves like `\visible`;
- `\onslide*` behaves like `\only`.

The `\alt` command allows showing two different contents in the specified slides vs. the other slides. For instance, `\alt<1>{first}{second}` results in: second.

The `\temporal` command works similarly, but distinguishes before/during/after the overlay specification. For instance, `\temporal<2>{first}{second}{third}` results in: third.

Fixed-Size Changing Area

It is possible to define a fixed-size region in the page, whose content changes over slides. The interest is that the rest of the page does not move as this content changes.

Environment `\overlayarea` allows specifying the width and height of such region, whereas environment `\overprint` requires only to define a width: the height is automatically adjusted to the largest content in the environment.

Here is an example using `overlayarea`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Here is an example using `overprint`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Fixed-Size Changing Area

It is possible to define a fixed-size region in the page, whose content changes over slides. The interest is that the rest of the page does not move as this content changes.

Environment `\overlayarea` allows specifying the width and height of such region, whereas environment `\overprint` requires only to define a width: the height is automatically adjusted to the largest content in the environment.

Here is an example using `overlayarea`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Here is an example using `overprint`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Fixed-Size Changing Area

It is possible to define a fixed-size region in the page, whose content changes over slides. The interest is that the rest of the page does not move as this content changes.

Environment `\overlayarea` allows specifying the width and height of such region, whereas environment `\overprint` requires only to define a width: the height is automatically adjusted to the largest content in the environment.

Here is an example using `overlayarea`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Here is an example using `overprint`:

Text on the second slide. Longer text located on the third slide.

Here is some text located out of the region.

Overlays in Tables

It is possible to use the main overlay commands to control the visibility of table elements. Here is an example:

Body Parts	Centaur	Chimera	Minotaur	Sphinx	Tarasque
Dragon	✗	✓	✗	✗	✓
Bear	✗	✗	✗	✗	✓
Bull	✗	✗	✓	✗	✓
Goat	✗	✓	✗	✗	✗
Horse	✓	✗	✗	✗	✓
Human	✓	✗	✓	✓	✓
Lion	✗	✓	✗	✓	✓
Scorpion	✗	✗	✗	✗	✓
Snake	✗	✓	✗	✗	✗
Turtle	✗	✗	✗	✗	✓

Table 4: Composition of various legendary creatures.

Overlays in Tables

It is possible to use the main overlay commands to control the visibility of table elements. Here is an example:

Body Parts	Centaur	Chimera	Minotaur	Sphinx	Tarasque
Dragon	✗	✓	✗	✗	✓
Bear	✗	✗	✗	✗	✓
Bull	✗	✗	✓	✗	✓
Goat	✗	✓	✗	✗	✗
Horse	✓	✗	✗	✗	✓
Human	✓	✗	✓	✓	✓
Lion	✗	✓	✗	✓	✓
Scorpion	✗	✗	✗	✗	✓
Snake	✗	✓	✗	✗	✗
Turtle	✗	✗	✗	✗	✓

Table 4: Composition of various legendary creatures.

Overlays in Tables

It is possible to use the main overlay commands to control the visibility of table elements. Here is an example:

Body Parts	Centaur	Chimera	Minotaur	Sphinx	Tarasque
Dragon	✗	✓	✗	✗	✓
Bear	✗	✗	✗	✗	✓
Bull	✗	✗	✓	✗	✓
Goat	✗	✓	✗	✗	✗
Horse	✓	✗	✗	✗	✓
Human	✓	✗	✓	✓	✓
Lion	✗	✓	✗	✓	✓
Scorpion	✗	✗	✗	✗	✓
Snake	✗	✓	✗	✗	✗
Turtle	✗	✗	✗	✗	✓

Table 4: Composition of various legendary creatures.

Overlays in Diagrams

This theme allows using overlays in **TikZ** diagrams, so that one can progressively uncover parts of the figure, or modify it in other ways. For this purpose, one must use the additional **visible on** parameter in the options of the concerned **TikZ** command. Its value must be an overlay specification. Here is an example:

In the figure, vertex v_1 is drawn using the following command:

```
\node[shape=circle,
draw=red, visible on=<2->]
(v1) at (0,0) {$v_1$};
```

And its edges with:

```
\draw[visible on=<2->,
draw=red] (v1) -- (v2);
```

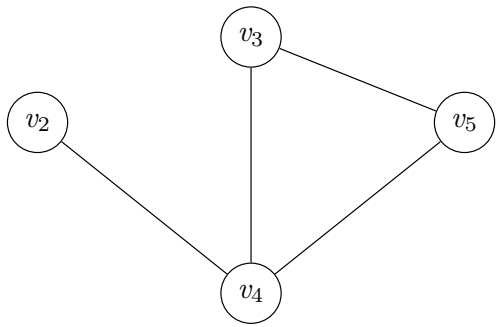

Figure 4: Overlays in a **TikZ** figure.

Overlays in Diagrams

This theme allows using overlays in **TikZ** diagrams, so that one can progressively uncover parts of the figure, or modify it in other ways. For this purpose, one must use the additional **visible on** parameter in the options of the concerned **TikZ** command. Its value must be an overlay specification. Here is an example:

In the figure, vertex v_1 is drawn using the following command:

```
\node[shape=circle,
draw=red, visible on=<2->]
(v1) at (0,0) {$v_1$};
```

And its edges with:

```
\draw[visible on=<2->,
draw=red] (v1) -- (v2);
```

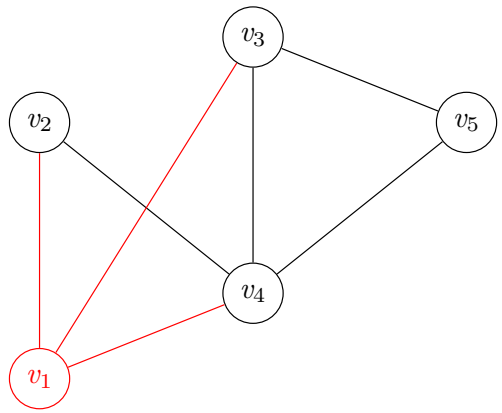

Figure 4: Overlays in a **TikZ** figure.

Questions?

AVIGNON
UNIVERSITÉ

Section 4

Appendix

AVIGNON
UNIVERSITÉ

Additional Material

It is possible to use the command `\appendix` to define additional pages, for instance containing additional material possibly useful to answer the audience's questions.

Thanks to package `appendixnumberbeamer`, these extra pages are not counted in the numbering displayed in the bottom right corner of the footer.

The bibliographical pages (see next) are not counted either.

Section 5

References

AVIGNON
UNIVERSITÉ

References I

[BE05] U. Brandes and T. Erlebach, eds. *Network Analysis: Methodological Foundations*. Vol. 3418. Lecture Notes in Computer Science. Springer, 2005. DOI: 10.1007/b106453.

[CLD16] J.-V. Cossu, V. Labatut, and N. Dugué. "A review of features for the discrimination of Twitter users: application to the prediction of offline influence". In: *Social Network Analysis and Mining* 6 (2016), pp. 1–23. DOI: 10.1007/s13278-016-0329-x. ⟨hal-01203171⟩.

[Dan+07] L. Danon, J. Duch, A. Arenas, and A. Díaz-Guilera. "Community structure identification". In: *Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science*. World Scientific, 2007. Chap. 5, pp. 93–113. DOI: 10.1142/9789812771681_0006.

[For10] S. Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3-5 (2010), pp. 75–174. DOI: 10.1016/j.physrep.2009.11.002.

[Ger10] A. Gerbaud. "Modélisation de réseaux d'interaction par des graphes aléatoires". PhD Thesis. Grenoble, FR: Université de Grenoble, 2010. ⟨tel-00460708⟩. URL: http://www.theses.fr/s123781.

References II

[LB12] V. Labatut and J.-M. Balasque. "Detection and Interpretation of Communities in Complex Networks: Methods and Practical Application". In: *Computational Social Networks: Tools, Perspectives and Applications*. Ed. by A. Abraham and A.-E. Hassanien. Springer, 2012. Chap. 4, pp. 81–113. DOI: 10.1007/978-1-4471-4048-1_4. ⟨hal-00633653⟩.

[Mai07] K. Mainzer. "Complex Systems and the Evolution of Computability". In: *Thinking in Complexity: The Computional Dynamics of Matter, Mind and Mankind*. 2007. Chap. 5, pp. 179–226. DOI: 10.1007/978-3-540-72228-1_5.

[ML16] N. Masuda and R. Lambiotte. *A Guide to Temporal Networks*. World Scientific, 2016. DOI: 10.1142/9781786341150.

[MLF08] A. Mauttone, M. Labbé, and R. Figueiredo. "A Tabu Search approach to solve a network design problem with user-optimal flows". In: *VALIO/EURO Conference on Combinatorial Optimization*. Buenos Aires, 2008, pp. 1–6. ⟨hal-01882382⟩.

[Neu13] J. Neunhäuserer. 12^2 *beautiful mathematical theorems with short proofs*. Tech. rep. 2013.

References III

[Par+13] D. C. Paraskevopoulos, T. Bektas, T. Gabriel Crainic, and C. N. Potts. *A Cycle-Based Evolutionary Algorithm for the Fixed-Charge Capacitated Multi-Commodity Network Design Problem.* Tech. rep. Montréal: Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, 2013, pp. 1–31.

[PRD03] R. Pastor-Satorras, M. Rubi, and A. Diaz-Guilera, eds. *Statistical Mechanics of Complex Networks.* Vol. 625. Lecture Notes in Physics. Springer, 2003. DOI: 10.1007/b12331.

[Rei09] J. Reichardt. "Modularity of Dense Random Graphs". In: *Structure in Complex Networks*. Vol. 766. Lecture Notes in Physics. 2009. Chap. 5, pp. 69–86. DOI: 10.1007/978-3-540-87833-9_5.

[RB09] M. Rosvall and C. T. Bergstrom. *Fast stochastic and recursive search algorithm.* Tech. rep. Umeå, SE: Department of Physics, Umeå University, 2009.

[WC89] Y.-C. Wei and C.-K. Cheng. "Towards efficient hierarchical designs by ratio cut partitioning". In: *IEEE International Conference on Computer Aided Design*. 1989, pp. 298–301. DOI: 10.1109/ICCAD.1989.76957.

[Wol98] L. A. Wolsey. *Integer programming.* New York, USA: Wiley-Interscience, 1998.

References IV

[Won78] R. T. Wong. "Accelerating Benders decomposition for network design". PhD Thesis. Massachusetts Institute of Technology, 1978.