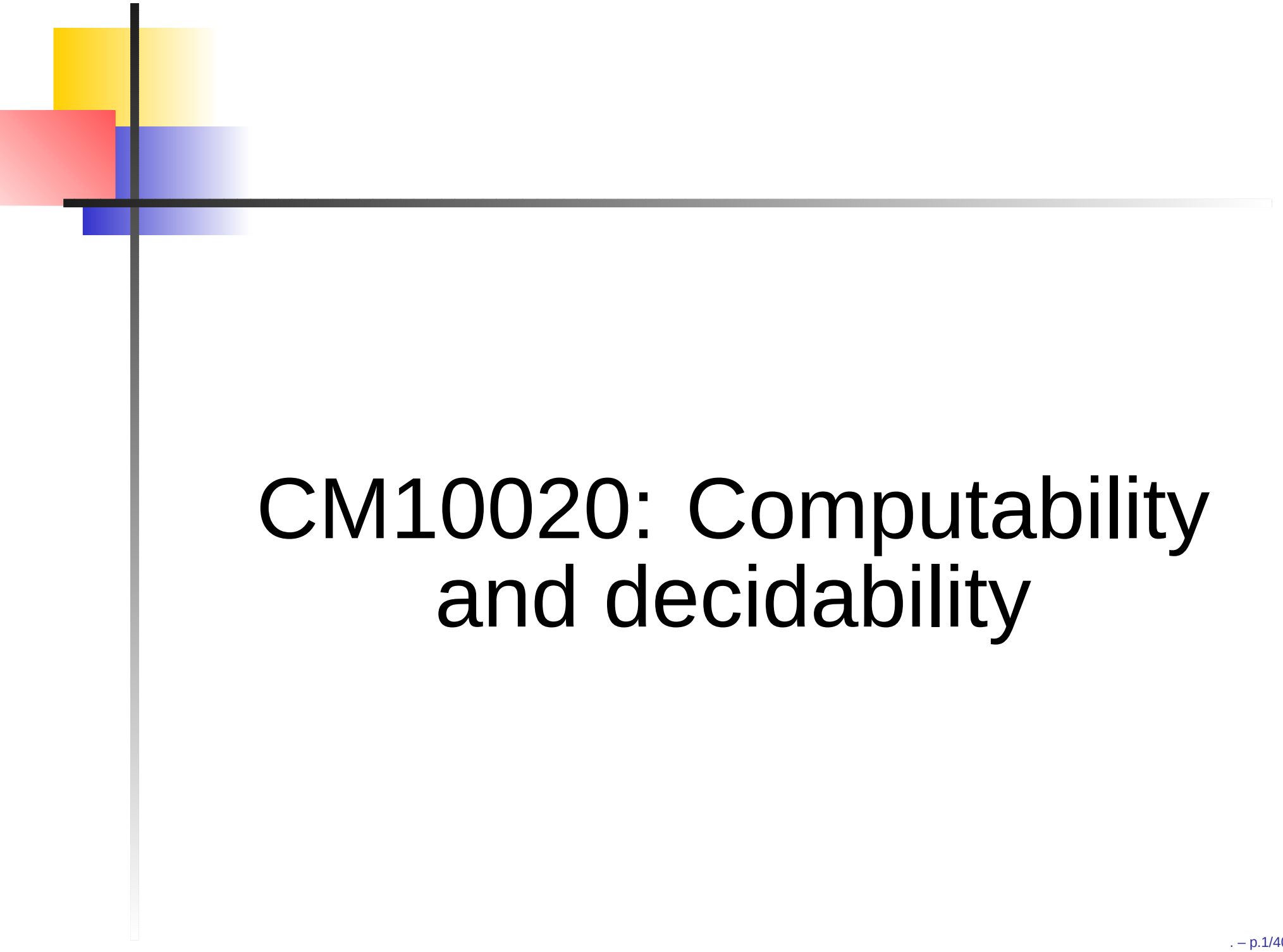
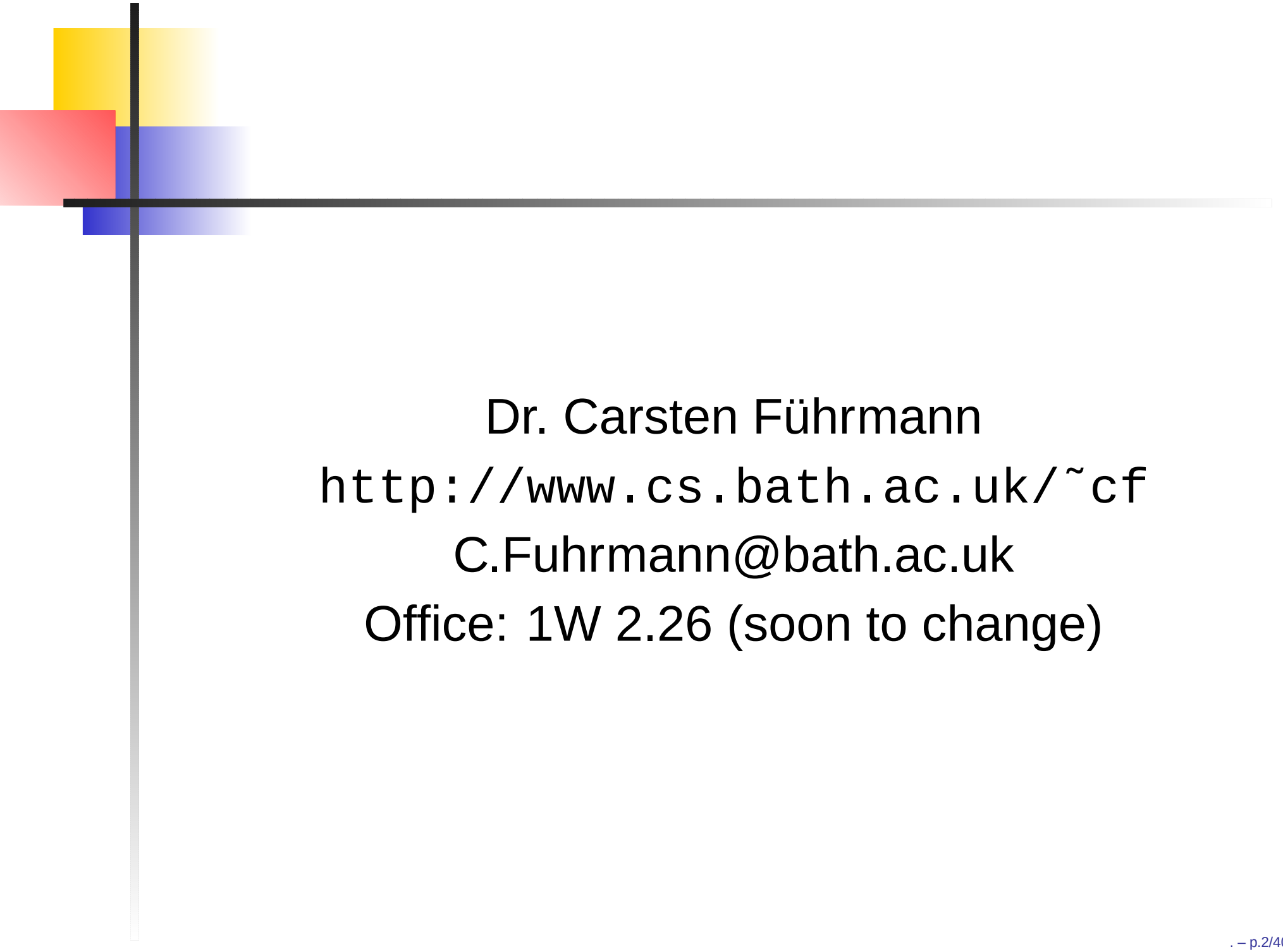


CM10020: Computability and decidability



Dr. Carsten Führmann

http://www.cs.bath.ac.uk/~cf

C.Fuhrmann@bath.ac.uk

Office: 1W 2.26 (soon to change)

Why this course?

The aims of this course are to

1. present some celebrated results about the limits of what can be computed (greater part of this course), and
2. introduce automata and formal languages, which have numerous applications in practical computer science (smaller part of this course).

Book for computability

Computability and Logic (4th ed.), Boolos/Burgess/Jeffrey, Cambridge University Press.

Stocked by Waterstone's; also available in the library. Recommended purchase. (Comparatively cheap; also contains valuable material about logic, which may be useful in later years.)

Book for automata & languages

Introduction to Automata Theory, Languages, and Computation (2nd ed.), Hopcroft/Motwani/Ullman, Addison Wesley.

Contains optional background reading (and many other interesting topics); very expensive, and therefore not stocked by Waterstone's; if interested, see library, Amazon. . .

Handouts

- There will be handouts at the beginning of most lectures.
- These handouts are identical with my slides.
- I will also make them available (as ps and pdf files) on my homepage (`http://www.cs.bath.ac.uk/~cf`).
- My slides will cover all topics, but background reading might help understanding.

Assessment

- Assessed by a 2-hour written exam.
- My handouts will contain **exam-relevant** exercises, which will be discussed in the weekly tutorials. These exercises will **not** be assessed.

Tutorials

- Five tutorials per week, each student assigned to **one** of them (see first-year notice board).
- Starting Monday 16.
- Two on Monday at 15:15, three on Friday at 14:15.
- The tutor Dan Wiley won’t be there during the week of the 16th; please attend other tutorials.

Purpose of the tutorials

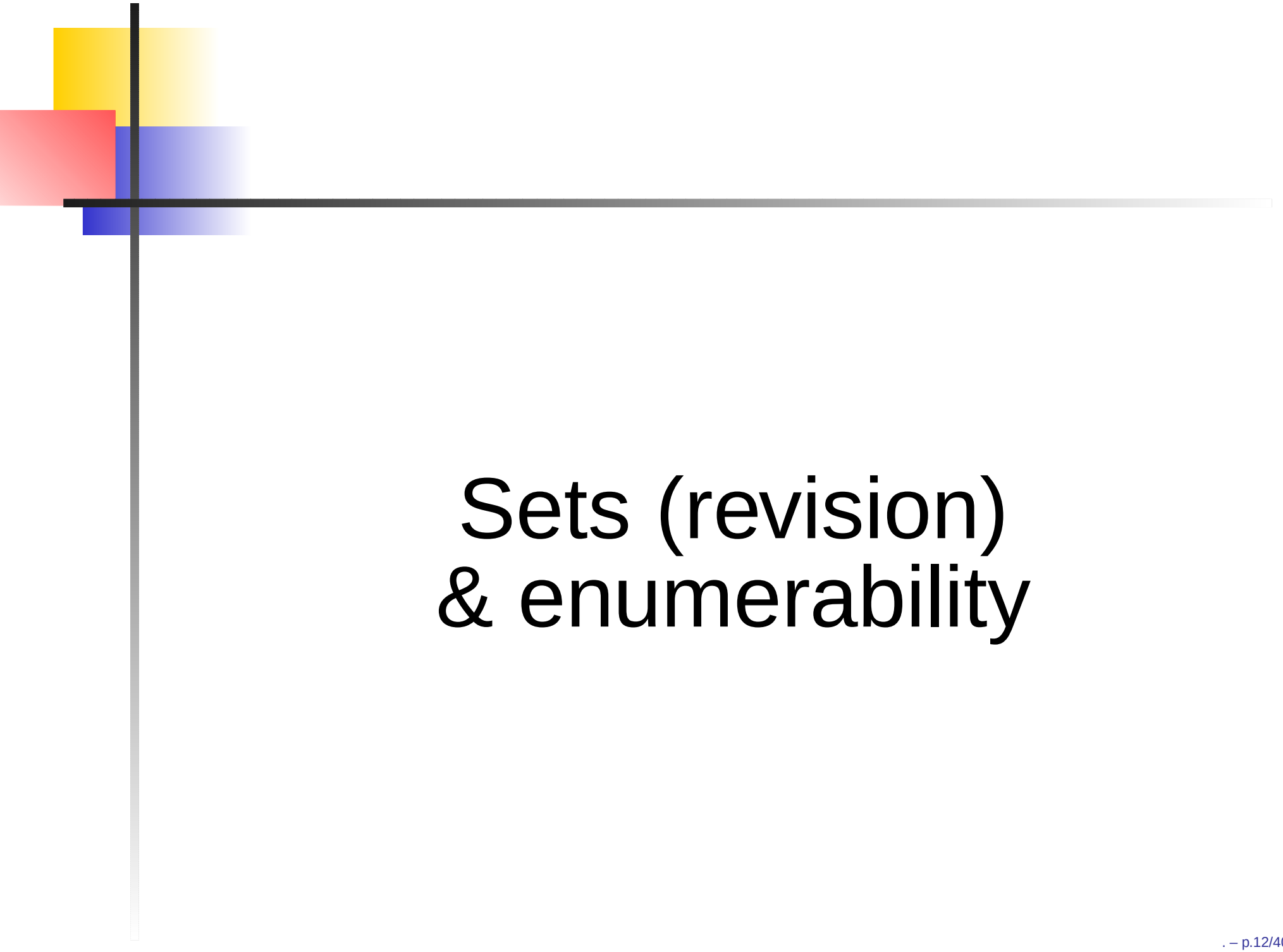
- The tutorials are there to help you understand and prepare for the exam.
- The exercises in my handouts will be discussed in the tutorials.
- Those exercises are similar, but not identical, to some exam questions.
- Tutors can give solutions, but only if they think it's appropriate. Try exercises before tutorials.
- There will also be exam questions about the **story** of the lecture (e.g. "explain what a Turing machine is").

Outline

1. Introduction
2. Sets and functions (revision)
3. Enumerability
4. Finite automata and regular languages
5. The Chomsky-hierarchy—in particular, context-free grammars
6. Turing machines, Turing's thesis
7. Uncomputability—in particular, the halting problem
8. Abacus machines
9. Primitive recursive functions, mu-recursive functions, Church's thesis
10. Explanations why Turing machines, abacus machines, and mu-recursive expressions describe the same notion of computation
11. Recursive and semirecursive sets
12. Lambda-calculus

Timeline

Weeks 1–7	lectures
Easter Break	
Weeks 8–11	lectures
Week 12	office hours for questions



Sets (revision) & enumerability

Enumerability: overview

- Goal of the course: present some celebrated theorems about the limits of what can be computed.
- Computations involve integers (1, 2, 3, . . .), strings of text, . . .
- It is important to understand the difference between two kinds of infinite sets: **enumerable sets** and **non-enumerable sets**.
- Before discussing enumerability, we'll go through a reminder of set theory.

Sets

A set is a collection of objects. Examples:

- Days of the week: $D = \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}\}$. Different notation: $D = \{x | x \text{ is a weekday}\}$.
- **Positive integers** (also called "natural numbers"): $N = \{1, 2, 3, 4 \ldots\}$.
- Prime numbers: $P = \{2, 3, 5, 7, \ldots\}$, or $P = \{x | x \text{ is a prime number}\}$.
- **Empty set**: $\emptyset = \{\}$.
- Real numbers.

Sets

- We write $x \in S$ if x is in S. We say "x is an element of S".
- We write $x \notin S$ if x is not in S.
- A set A is called a **subset** of a set B if every element of A is an element of B. We write $A \subseteq B$.
- Two sets A and B are considered equal if they have the same elements; we write $A = B$.

Russell's paradox (1901)

- Things are not as simple as they seem: suppose that R is the set of sets that are not members of themselves, i.e.

$$R = \{x | x \notin x\}.$$

- Is $R \in R$?
- If the answer is "yes", it follows that not$(R \in R)$, which is a contradiction.
- If the answer is "no", it follows that $R \in R$, which is also a contradiction!

Escape from Russell's Paradox

- Strictly speaking, we cannot write

$$\{x | x \text{ has a certain property}\}.$$

We can only write

$$\{x \in U | x \text{ has a certain property}\},$$

where U is another set.

- However, paradoxes like Russell's are rarely a problem in everyday mathematics, and the issue is usually ignored.

New sets from old

- **Union** of A and B:

$$A \cup B = \{x \,|\, x \in A \text{ or } x \in B\}$$

- **Intersection** of A and B:

$$A \cap B = \{x \,|\, x \in A \text{ and } x \in B\}$$

- A **less** B:

$$A - B = \{x \in A \,|\, x \notin B\}$$

Product of sets

- **Product** of A and B. This is the set of **ordered pairs** whose first component is in A and whose second component is in B:

$$A \times B = \{(x, y) \mid x \in A \text{ and } y \in B\}$$

- For a non-negative integer k, we write A^k for the k-fold product of A with itself, i.e. the lists of length k whose components are in A:

$$A^k = \{(x_1, x_2, \ldots, x_k) \mid x_1 \in A, x_2 \in A, \ldots, x_n \in A\}$$

Lists over a set

- The set A^* of **lists** over a set A is

$$A^* = \{(a_1, a_2, \ldots, a_k) : k \geq 0 \text{ and } a_1, \ldots, a_k \in A\}.$$

In other words, A^* is the word of k-tuples of arbitrary length over A. Note that this includes the **empty tuple** $()$.

Example. Text strings can be considered as lists. For example, the set of strings over the Latin alphabet is $\{a, b, c, \ldots, z\}^*$.

Powersets

- **Powerset** of A. This is the set of all subsets of A:

$$P(A) = \{S \mid S \subseteq A\}$$

Example: the powerset of $\{red, green, blue\}$ is

$$\{\{\},$$
$$\{red\}, \{green\}, \{blue\},$$
$$\{green, blue\}, \{red, blue\}, \{red, green\},$$
$$\{red, green, blue\}\}.$$

Total functions

Definition. A **total function** f from a set A to a set B is an assignment that sends each element a of A to a unique element $f(a)$ of B.

Example. Doubling a number is a function d from N to N:

$$d(1) = 2$$
$$d(2) = 4$$
$$d(3) = 6$$
$$\vdots$$

Functions

Definition. A **function** f from a set A to a set B is an assignment that sends each element a of A to a unique element $f(a)$ of B **or is undefined on** a.

Example. The assignment that halves even numbers and is undefined on non-even numbers is a partial function h from the set integers to the set of integers:

$$h(1) = undefined \qquad h(2) = 1$$
$$h(3) = undefined \qquad h(4) = 2 \qquad \ldots$$

Functions: terminology

- Warning: the terminology in most areas of mathematics differs from the one in this course. In mathematics, functions are often assumed to be total by default. If they are not total, they are called “partial functions”.
- Our terminology differs because for computable functions, being partial is the normal (because programs can go into infinite loops), and being total is special.

Composition of functions

Definition. Let $f : A \to B$ and $g : B \to C$ be functions. We define

$$g(f(a)) = \begin{cases} undefined & \text{if } f(a) \text{ is } undefined \\ g(b) & \text{if } f(a) = b \end{cases}.$$

The resulting function $A \to C$ is called the **composition** of g and f; it is denoted by $g \circ f$.

Domain of a function

Definition. The **domain** of a function $f : A \to B$ is defined to be the set of all a in A such that $f(a)$ is defined.

So a function $f : A \to B$ is total if and only if its domain is the whole of A.

Example. The domain of the halving function h from the earlier slide is the set of even positive integers.

Range of a function

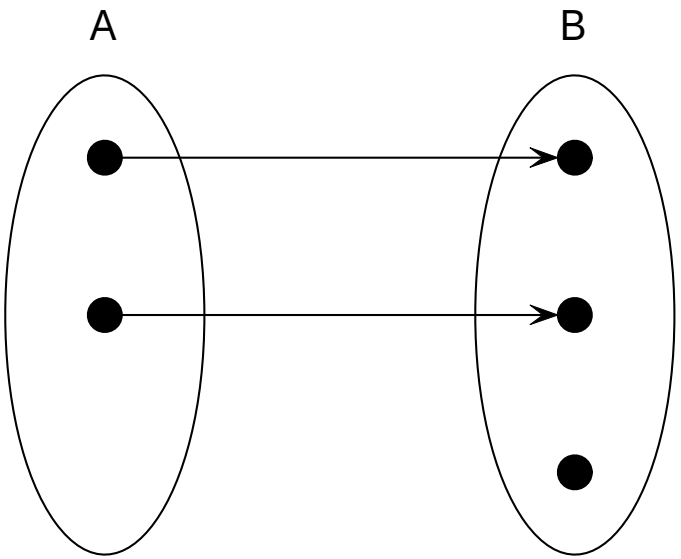
Definition. The **range** of f is defined to be the set of b in B that are of the form $f(a)$ for some a in A.

Example. The range of the doubling function d from the earlier slide is the set of even positive integers.

Surjective functions

Definition. A function $f : A \to B$ is called **surjective** if its range is the whole of B.

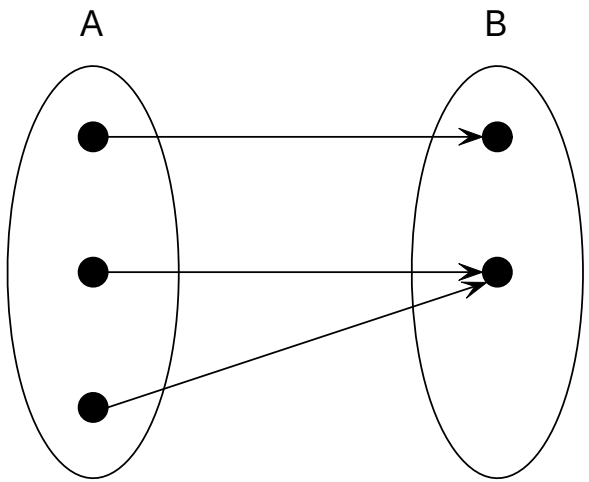
Example. A function of the kind below is **not** surjective.



Injective functions

Definition. A function $f : A \to B$ is called **injective** (or "one-to-one") if for every element b of B there is at most one a such that $b = f(a)$.

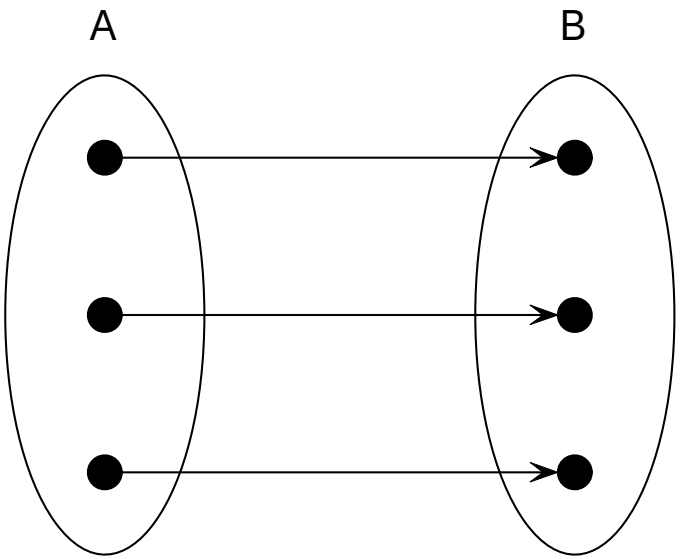
Example. A function of the kind below is **not** injective.



Bijective functions

Definition. A **total** function is called **bijective** if it is both injective and surjective.

Example. Total functions of the kind below are bijective.



Exercise

Let $f : A \to B$ and $g : B \to C$ be functions. Show that

1. If f and g are total, then so is $g \circ f$;
2. If f and g are injective, then so is $g \circ f$;
3. If f and g are surjective, then so is $g \circ f$;
4. If f and g are bijective, then so is $g \circ f$.

Exercise

Show the following statements:

1. If there is a bijection $A \to B$, there is also a bijection $B \to A$. (In this case, A and B are called **equinumerous**, and we write $A \simeq B$.)
2. Every set is equinumerous with itself.
3. If $A \simeq B$ and $B \simeq C$, then $A \simeq C$.

Exercise

Show that

1. The set of real numbers x with $0 < x < 1$ is equinumerous with the set R^+ of positive real numbers.
2. The set of real numbers x with $0 < x < 1$ is equinumerous with the set R of **all** real numbers.
3. If $A \simeq C$ and $B \simeq D$, then $A \times B \simeq C \times D$.
4. If $A \simeq C$ are equinumerous, and $B \simeq D$, and the intersections $A \cap B$ and $C \cap D$ are empty, then $A \cup B \simeq C \cup D$.

Exercise

1. Let $s : A \to B$ be a surjective function. Show that there exists an injective total function $i : B \to A$ such that $s(i(b)) = b$ for all $b \in B$.
2. Let $i : B \to A$ be an injective total function. Show that there exists a unique surjective function $s : A \to B$ such that $s(i(b)) = b$ for all $b \in B$ and $i(s(a))$ is either undefined or equal to a for all $a \in A$.

Remark: this exercise foreshadows the connection between enumerations and encodings, which we shall see later

Arity

- The argument of a function

$$f : A_1 \times A_2 \times \cdots \times A_k \to B$$

  is a list $(x_1, x_2, \ldots, x_k)$, where $x_i \in A_i$.

- We say that "f takes k arguments", or the **arity** of f is k.

Enumerability: informal description

- A set is called **enumerable** (or "countable") if it is either finite or its elements can be written as a list.
- For example, the set of odd numbers is enumerable. The list is $1, 3, 5, 7, 9, \ldots$.
- "$1, 3, 7, 9, \ldots, 2, 4, 6, 8, \ldots$" is **not** a valid list, and neither is "$\ldots, -2, -1, 0, 1, 2, \ldots$".
- By "list" we mean that every element's position must be given by a positive integer.

Enumeration by a list

- Consider the list of even natural numbers $2, 4, 6, 8, \ldots$
- For a natural number i, let $f(i)$ be the i-th element of that list, i.e.

$$f(1) = 2, f(2) = 4, f(3) = 6, f(4) = 8, \ldots$$

- The set of even natural numbers is the range of the function $f : N \to N$
- So the function f is an **enumeration** of the set of even natural numbers.

Enumeration by a list with holes

- Let $g : N \to N$ be the function

$$g(n) = \begin{cases} n & \text{if } n \text{ is even} \\ \text{undefined} & \text{otherwise} \end{cases} .$$

- The range of g is the set of even numbers.
- The function g is another enumeration of the set of even numbers.
- "List with holes": $2, \ldots, 4, \ldots, 6, \ldots$

Enumerability: formal definition

Definition. A set A is called **enumerable** if it is the range of a function $f : N \to A$ from the positive integers to A (in other words, if there is a surjective function $N \to A$). We call f the **enumeration function** of A.

Remark: the "list with holes" f is allowed to have double occurrences, e.g. $2, 2, 4, 4, 6, 6, \ldots$ is an enumeration function for the set of even numbers.

Enumerability of the integers

The set of integers. A simple enumeration is

$$1, -1, 2, -2, 3, -3, \ldots$$

That is, the enumerating function is

$$f(1) = 1, f(2) = -1, f(3) = 2,$$

$$f(4) = -2, f(5) = 3, f(6) = -3, \ldots$$