Languages and Machines

Introduction

Jorge A. Pérez

Bernoulli Institute for Mathematics, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

April 14, 2025

L&M, In a Nutshell

- A course on **automata theory and formal languages**
- Lecturers: Dr. Dan Frumin and Dr. Jorge Pérez
- Lectures (at least one per week) and tutorials (one per week)
- We assume you have passed (and still remember!)
  - Introduction to Logic
  - Discrete Structures (in particular: the induction principle)
- Assessment:
  - Three individual homeworks (mandatory)
  - A final exam (2h)
- Self-study is important!
- Helpdesk email: `lm25.cs.rug@gmail.com`

The Foundations of Computation

Basic questions:

- What does it mean for a function to be computable?
- Are there any non computable functions?
- Computational power $\leftrightarrow$ Programming constructs?

The Foundations of Computation

Basic questions:

- What does it mean for a function to be computable?
- Are there any non computable functions?
- Computational power $\leftrightarrow$ Programming constructs?

Looking for answers $\rightarrow$ Fundamental concepts

- State
- Transition
- Non-determinism
- Undecidability
- $\dots$

Persistent concepts, despite many (and frequent) technology changes

Models of Computation

In order of increasing power:

(a) Finite Memory:
Finite automata; regular expressions

(b) Finite Memory with stack:
Pushdown automata

(c) Unrestricted:
Turing machines (terminating and non-terminating)

Grammars and Languages

The **Chomsky hierarchy** - in order of increasing complexity:

(i) Right-linear grammars

(ii) Context-free grammars

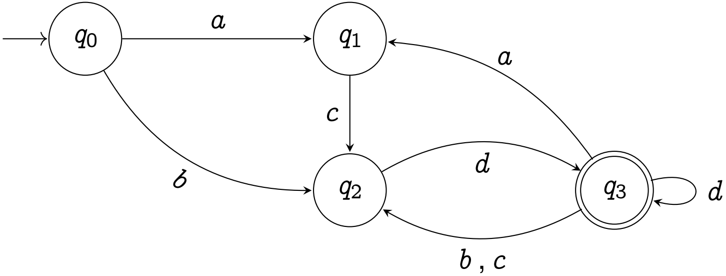
(iii) Unrestricted grammars

Grammars and Machines

- Superficially very different
- Parsing a sentence in a **language** (a set of strings) is quite similar to computation
- Grammar types (i)-(iii) are **equivalent** to machines (a)-(c)!

State-based systems are everywhere!

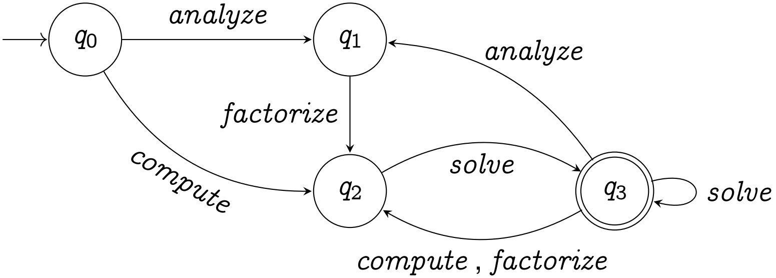
A finite-state machine



- Above, q_0, q_1, q_2, q_3 are states of the machine.
  While q_0 is the initial state, q_3 is the final state.
  Symbols a, b, c, d are recognized by moving between states.
- The machine recognizes a certain language: a set of strings.
  Such a set includes strings like 'acd' and '$bddd$' but not 'acb' nor 'bca'.

State-based systems are everywhere!

A finite-state machine... can also be a specification for object-oriented programs!

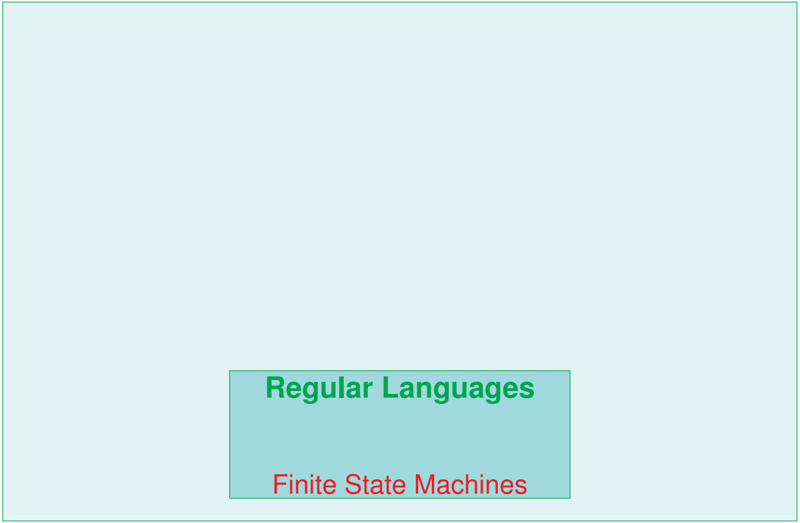


- Above, q_0, q_1, q_2, q_3 are states of the machine.
  While q_0 is the initial state, q_3 is the final state.
  Symbols a, b, c, d are recognized by moving between states.
- By interpreting symbols a, b, c, d as class methods, we can specify the sequences of allowed invocations. (This is called a *typestate*.)

Many applications

- Programming language design and implementation
  (Compiler construction, domain specific languages, etc)
- Software and hardware verification
  (Model checking, run-time verification, etc)
- Learning and AI
- Bioinformatics
- Security
- . . .

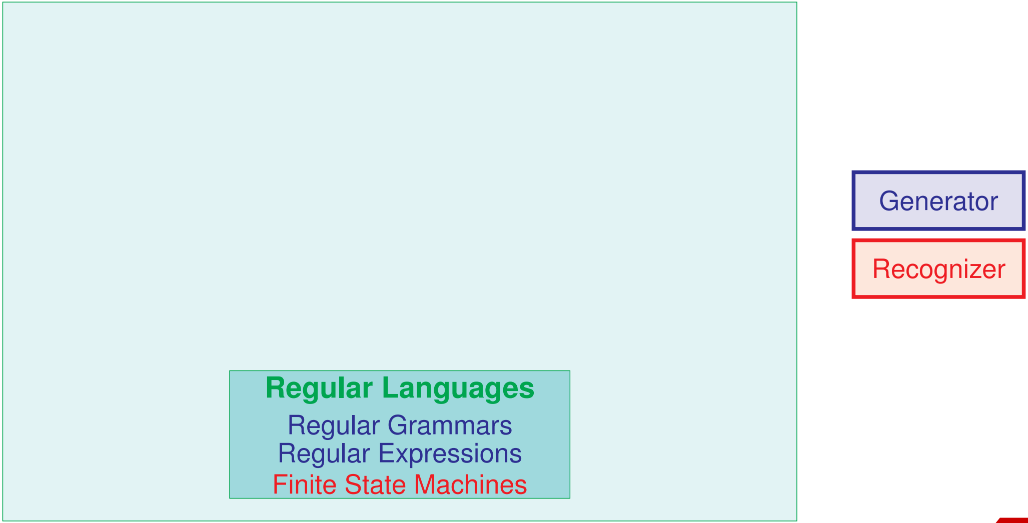
The Course, In a Nutshell



Regular Languages

Finite State Machines

The Course, In a Nutshell



Regular Languages

Regular Grammars

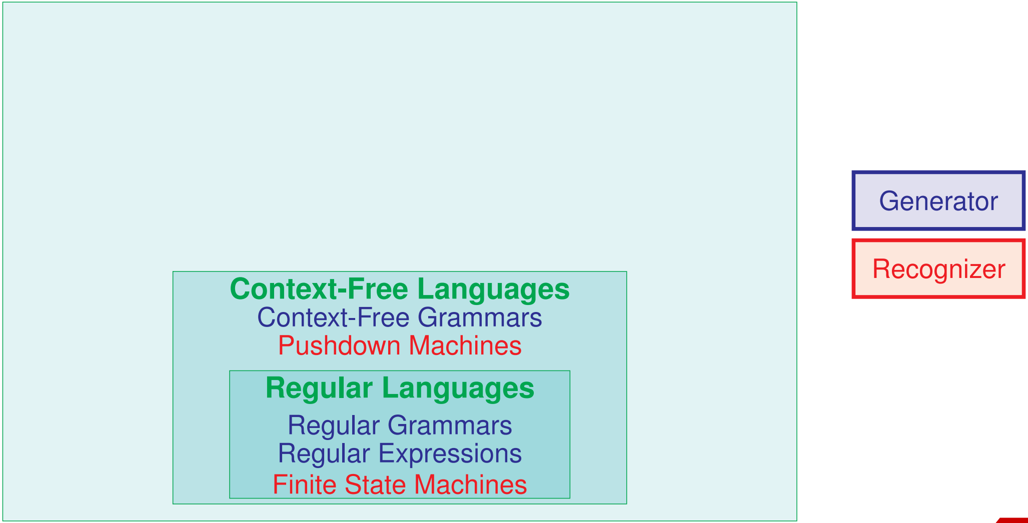
Regular Expressions

Finite State Machines

Generator

Recognizer

The Course, In a Nutshell



Context-Free Languages

Context-Free Grammars

Pushdown Machines

Regular Languages

Regular Grammars

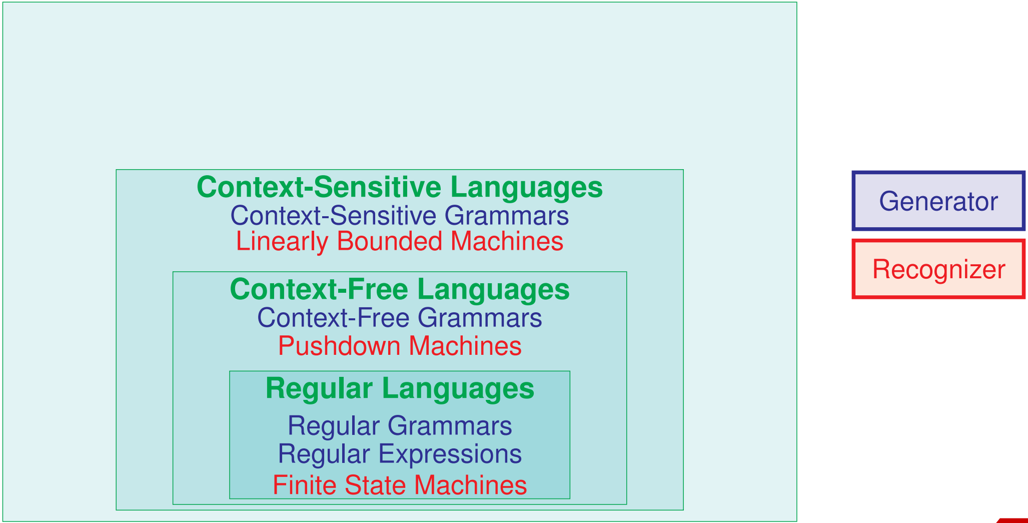
Regular Expressions

Finite State Machines

Generator

Recognizer

The Course, In a Nutshell



Context-Sensitive Languages
Context-Sensitive Grammars
Linearly Bounded Machines

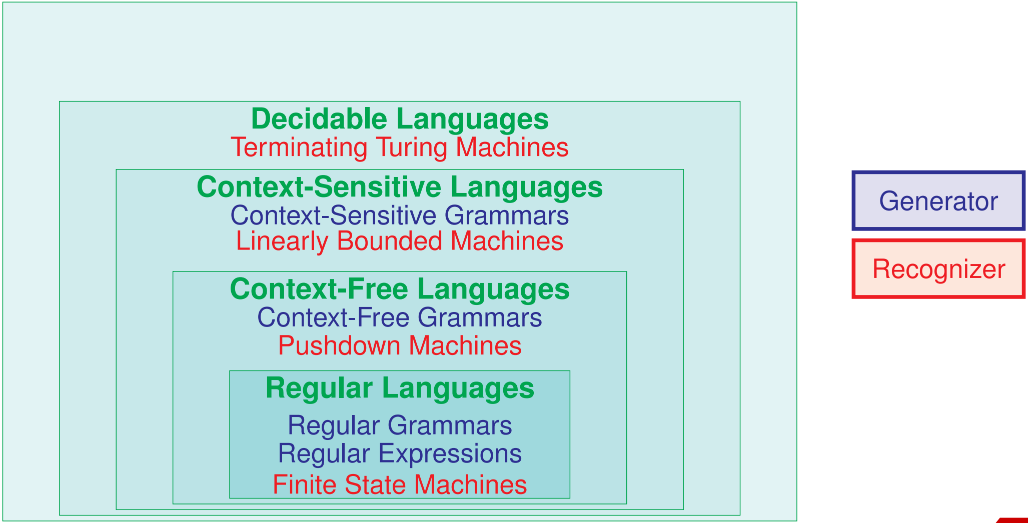
Context-Free Languages
Context-Free Grammars
Pushdown Machines

Regular Languages
Regular Grammars
Regular Expressions
Finite State Machines

Generator

Recognizer

The Course, In a Nutshell



Decidable Languages
Terminating Turing Machines

Context-Sensitive Languages
Context-Sensitive Grammars
Linearly Bounded Machines

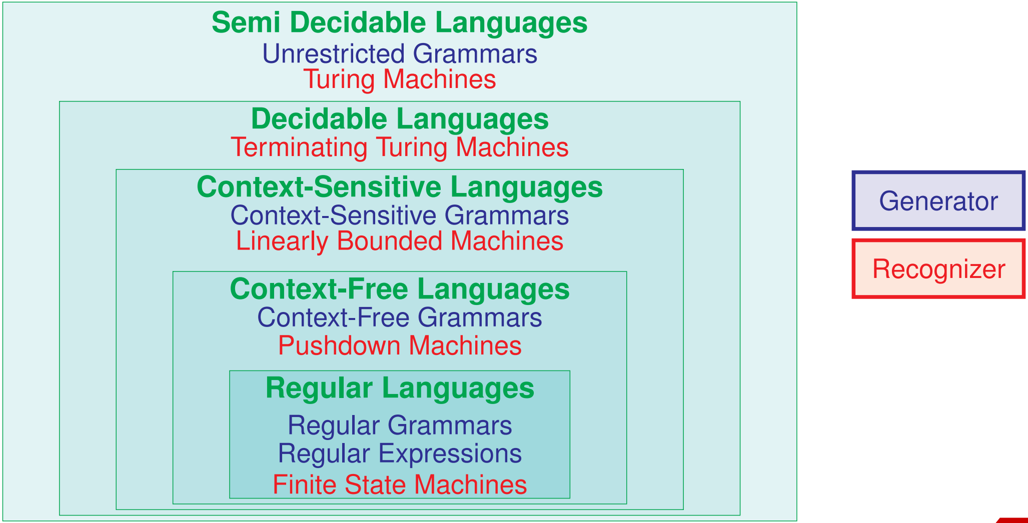
Context-Free Languages
Context-Free Grammars
Pushdown Machines

Regular Languages
Regular Grammars
Regular Expressions
Finite State Machines

Generator

Recognizer

The Course, In a Nutshell



Semi Decidable Languages

Unrestricted Grammars

Turing Machines

Decidable Languages

Terminating Turing Machines

Context-Sensitive Languages

Context-Sensitive Grammars

Linearly Bounded Machines

Context-Free Languages

Context-Free Grammars

Pushdown Machines

Regular Languages

Regular Grammars

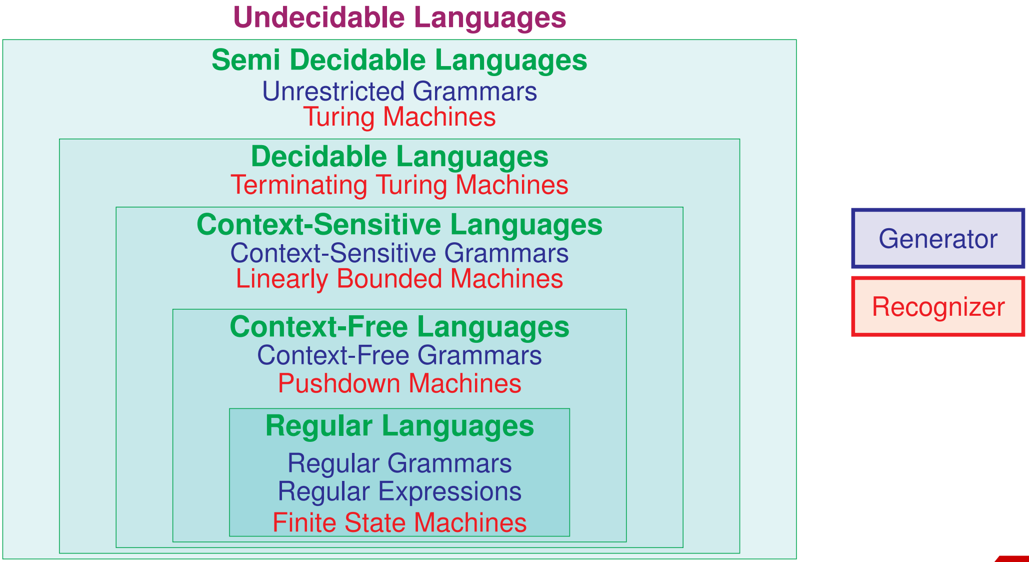
Regular Expressions

Finite State Machines

Generator

Recognizer

The Course, In a Nutshell



Undecidable Languages

Semi Decidable Languages

Unrestricted Grammars

Turing Machines

Decidable Languages

Terminating Turing Machines

Context-Sensitive Languages

Context-Sensitive Grammars

Linearly Bounded Machines

Context-Free Languages

Context-Free Grammars

Pushdown Machines

Regular Languages

Regular Grammars

Regular Expressions

Finite State Machines

Generator

Recognizer

The Course, In a Nutshell

Expressiveness

Simplicity / Efficiency

Regular

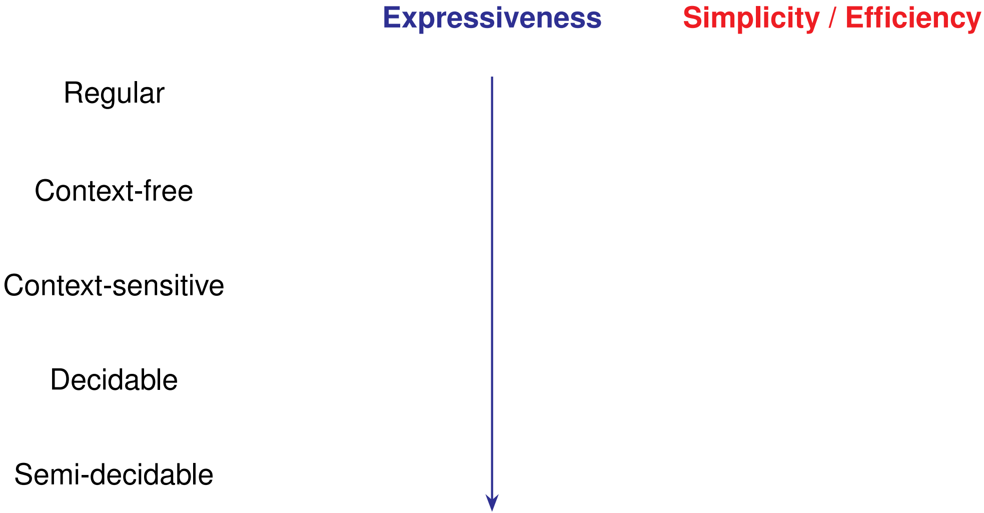
Context-free

Context-sensitive

Decidable

Semi-decidable

The Course, In a Nutshell



Expressiveness

Simplicity / Efficiency

Regular

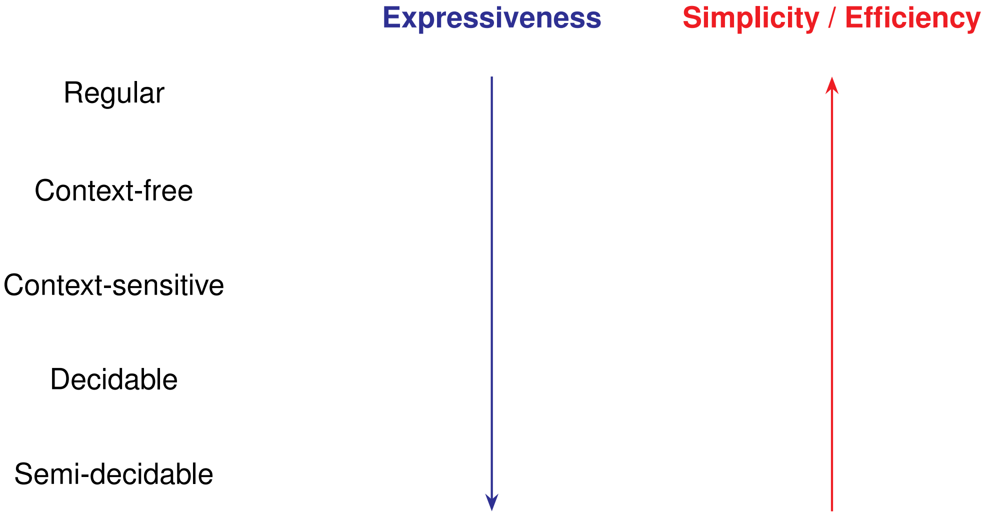
Context-free

Context-sensitive

Decidable

Semi-decidable

The Course, In a Nutshell



Expressiveness

Simplicity / Efficiency

Regular

Context-free

Context-sensitive

Decidable

Semi-decidable

Overview

In this course, we will describe, analyse, and classify the languages that can be read by machines and the machines that can read them.

The course does not concern the interpretation of such languages.

Roughly, seven parts:

1. Regular and context-free languages
2. Finite state machines
3. Properties of regular languages
4. Pushdown machines
5. Properties of context-free languages
6. Turing machines
7. Decidability issues

Learning Outcomes (Ocasys)

1. The student learns to understand and apply:
   (a) The basic theory of finite state, pushdown, and Turing machines, and of the regular, context-free, and decidable and semi-decidable languages.
   (b) The relationships between machines and languages, and the translation algorithms between the various representations (e.g. regular expressions, normal forms of grammars).
2. The student obtains an elementary understanding of decidability, undecidability, semi-decidability, computability, time complexity, the classes P and NP, and the Chomsky hierarchy.

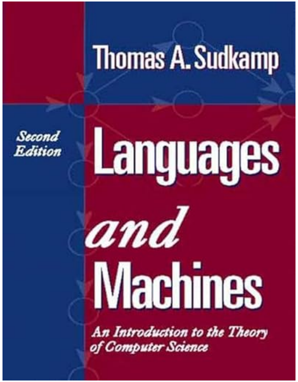
Material

“If it ain’t broke, don’t fix it”

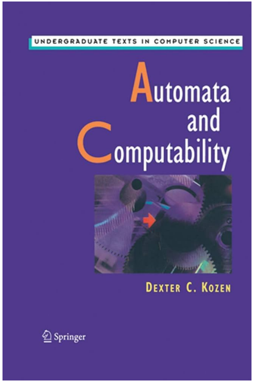
- Lecture Notes (reader) “Languages and Machines” by Wim Hesselink.
  Many exercises, some of which are discussed at the tutorials.
  PDF available in Brightspace: you comments are welcome!
- New this year: the reader is continuously revised: make sure to get the latest version!

The reader is our main reference, but it is not a textbook. Many good textbooks around!

Two Recommended Textbooks



Languages and Machines: An Introduction to the Theory of Computer Science
by Thomas A. Sudkamp


UNDERGRADUATE TEXTS IN COMPUTER SCIENCE
Automata and Computability
DEXTER C. KOZEN
Springer

Automata and Computability
by Dexter C. Kozen

Teaching Method

On our side:

- In-person lectures (usually twice per week).
- Tutorials (once per week).
  Teaching Assistants:
  Sarah Baksteen, Miguel Bartelsman, Aron Hardeman, Barnabás Tarcali

Schedules in Brightspace (subject to changes).

Teaching Method

On our side:

- In-person lectures (usually twice per week).
- Tutorials (once per week).
  Teaching Assistants:
  Sarah Baksteen, Miguel Bartelsman, Aron Hardeman, Barnabás Tarcali

Schedules in Brightspace (subject to changes).

On **your side**: self-study!

Self-Study

Before **each lecture**:

- Study the reader and consult textbooks as needed
- Identify potential questions

Before **each tutorial**:

- Work on the suggested exercises — it is good to “get stuck”!
- Contact TAs about potential questions or specific exercises to discuss

Self-Study

Before **each lecture**:

- Study the reader and consult textbooks as needed
- Identify potential questions

Before **each tutorial**:

- Work on the suggested exercises — it is good to “get stuck”!
- Contact TAs about potential questions or specific exercises to discuss

Self-study also means:

- Attending lectures and tutorials
- Handing homeworks on time
- Look into topics not covered (or partially covered) in the course
- Providing constructive feedback to the lecturer and TAs

Grading (Ocasys)

Components

1. H: Three individual homeworks
2. E: Exam

Your Final Grade

$$F = (0.6 \times E) + (0.4 \times H)$$

There is also a resit.
Note: H does not count at the resit, nor can it be transferred to future academic years.

Homeworks and Exam

Two different (and complementary) ways of assessing your learning process

- Homeworks are intended to cover selected portions of the content of the course
  In contrast, the exam is meant to cover the entire course content
- By design, homeworks allow you to reflect about the topics and your understanding of them, and to discuss with TAs and fellow students (more on this later)

Homeworks and Exam

Two different (and complementary) ways of assessing your learning process

- Homeworks are intended to cover selected portions of the content of the course
  In contrast, the exam is meant to cover the entire course content
- By design, homeworks allow you to reflect about the topics and your understanding of them, and to discuss with TAs and fellow students (more on this later)
- In contrast, the exam comes with specific requirements in time and space

Homeworks and Exam

Two different (and complementary) ways of assessing your learning process

- Homeworks are intended to cover selected portions of the content of the course
  In contrast, the exam is meant to cover the entire course content
- By design, homeworks allow you to reflect about the topics and your understanding of them, and to discuss with TAs and fellow students (more on this later)
- In contrast, the exam comes with specific requirements in time and space
- Accordingly, exercises in homeworks and exam are necessarily different in nature, goal, and scope
- Exam questions are designed to assess essential and representative concepts, while enabling fair, efficient grading

Homeworks and Exam

Two different (and complementary) ways of assessing your learning process

- Homeworks are intended to cover selected portions of the content of the course
  In contrast, the exam is meant to cover the entire course content
- By design, homeworks allow you to reflect about the topics and your understanding of them, and to discuss with TAs and fellow students (more on this later)
- In contrast, the exam comes with specific requirements in time and space
- Accordingly, exercises in homeworks and exam are necessarily different in nature, goal, and scope
- Exam questions are designed to assess essential and representative concepts, while enabling fair, efficient grading
- Although the exam may include topics/questions you have encountered in the homeworks, this is not necessarily the case!

Important Dates

Three individual homeworks:

- Deadline on Fridays, 10h:
  April 25, May 9, May 30.
- See Brightspace for instructions / updates.

Exam and Resit

- See the rooster for dates / times.

Contact Us

1. General questions: send us an email

   lm25.cs.rug@gmail.com

2. Specific questions, feedback, requests for (online) meetings

   j.a.perez[at]rug.nl, d.frumin[at]rug.nl

You can always reach out to the academic advisor (Korrie Bonnema):

- Email: academicadvisor.cs[at]rug.nl

Plagiarism and Scientific Integrity

Study Guide Computer Science, section on “Fraud Prevention & Scientific Integrity” — `https://student.portal.rug.nl/infonet/studenten/fse/programmes/bsc-cs/`:

> *Plagiarism is not accepted at this university nor elsewhere in the scientific community.*
> *In all cases in which plagiarism is found or suspected, the examiner will inform the Board of Examiners.*
>
> *Possible consequences:*
>
> - *Warning*
> - *Exclusion from exams for the relevant course for 1 academic year*
> - *Exclusion from exams for several courses for 1 academic year*
> - *Exclusion from programme*

Plagiarism and Scientific Integrity

- Certain forms of collaboration among students are allowed, plagiarism is not.

Plagiarism and Scientific Integrity

- Certain forms of collaboration among students are allowed, plagiarism is not.
- Homeworks are strictly individual. This doesn't exclude the possibility of discussing homework exercises with your fellow students.
- Discussions are encouraged: they can be useful to clarify concepts and rule out misconceptions. No need to refer to specific exercises.

Plagiarism and Scientific Integrity

- Certain forms of collaboration among students are allowed, plagiarism is not.
- Homeworks are strictly individual. This doesn't exclude the possibility of discussing homework exercises with your fellow students.
- Discussions are encouraged: they can be useful to clarify concepts and rule out misconceptions. No need to refer to specific exercises.
- When in doubt, contact the lecturers in advance (**before** handing in your solutions). Declaring collaborations (and their nature) is also possible.
- Once again, plagiarism suspicions detected **after** receiving homework solutions will always be forwarded to the BoE.
- Last but not least: The use of AI tools (such as ChatGPT) for attempting homework exercises is discouraged and disallowed.

The End