

Regular Lecture 1

Practicalities and Computer Aids

Prof. Dr. Jasmin Blanchette

Chair of Theoretical Computer Science and
Theorem Proving

Version of May 13, 2026



Practicalities

Teaching Staff

Lecturer

- ▶ Prof. Dr. Jasmin Blanchette
jasmin.blanchette@ifi.lmu.de

Assistants

- ▶ Dr. Martin Desharnais-Schäfer
desharnais-schaefer@ifi.lmu.de
- ▶ Dr. Michael Kirsten
michael.kirsten@ifi.lmu.de

Module

The seminar is recognized as “**Einführung persönliche und soziale Kompetenz**” (3 ECTS) in various computer science bachelor study programs at the LMU.

Make sure to register in LSF for the seminar and the examination.

Goals

The seminar will help you

- ▶ learn the basics of **scientific** and **technical writing** for **computer science**, such as the proper use of citations, quotations, mathematical notations, and code excerpts;
- ▶ acquire techniques for **structuring the writing process**, to help avoid the blank page syndrome;
- ▶ learn the fundamentals of **writing in plain formal English**, including grammar, punctuation, and style;
- ▶ learn how to develop **various types of documents**, such as papers, theses, slides, posters, and software documentation.

Why Writing Matters

Extracts from reviews of research paper submissions:

- ▶ The paper is well-written, the design choices are well explained and well-motivated.
- ▶ This is a solid and well written paper.
- ▶ Overall, this is a very well-written paper; the structure is clear, the method is understandable.
- ▶ This is a well-written report on an excellent bit of research.
- ▶ Overall, the paper is well written and I really like the idea.
- ▶ The paper is also very well written.
- ▶ The article is well written and clear.
- ▶ The article is well written and I find the contribution interesting.
- ▶ The submission is well-structured and well-written.
- ▶ Overall, the paper is well-written and easy to follow.

Do you think these papers were **accepted** or **rejected**?

Two Schools of Writing

The “**academic**” **school** favors

- ▶ passive voice;
- ▶ long sentences;
- ▶ impenetrableness;
- ▶ abstractness;
- ▶ verbosity.

The **plain English school** favors

- ▶ active voice;
- ▶ short to medium sentences;
- ▶ clarity;
- ▶ concreteness;
- ▶ conciseness.

To which school does this seminar belong?

Format

The seminar consists of

- ▶ 13 in-person **regular lectures** of 90 minutes each;
- ▶ 2 in-person **mystery guest lectures** of 90 minutes each;
- ▶ 11 ungraded peer-reviewed **homework** sets;
- ▶ a regular and a retake **examination**.

Each of the regular lectures consists of a traditional lecture immediately followed by an exercise session. **So bring your laptop, tablet, or pen and paper.**

Lectures

1. Practicalities and Computer Aids
2. Structure
3. Style
4. Mystery Guest 1
5. Words and Phrases
6. Sentences and Paragraphs
7. Punctuation, Part 1
8. Punctuation, Part 2
9. Mystery Guest 2
10. Typesetting
11. Citations and Quotations
12. Revisions
13. Theses and Papers
14. Slides and Posters
15. Software Documentation

Online Presence

LSF lsf.verwaltung.uni-muenchen.de
for course and examination registration

Web www.tcs.ifi.lmu.de/teaching/courses-ws-2026-27/scientific-and-technical-english
for slides, exercise sheets, practice examinations, and solutions

Moodle moodle.lmu.de/course/view.php?id=TODO
for announcements, extra materials, and homework submission and peer review

Zulip chat chat.ifi.lmu.de, stream [TCS-26W27-STE](https://tcs-26w27-ste.zulipchat.com)
for questions and answers about practicalities and content

LMUcast cast.itunes.uni-muenchen.de/vod/playlists/FV9dOHYHpY.html
for videos of the regular lectures from the winter semester 2025/26

Exercise Sheets

The 13 regular lectures (including today) are each complemented by an exercise sheet with in-class exercises and homework.

- ▶ You do the **in-class exercises** in class.
Solutions are published immediately afterward.
- ▶ The first 11 exercise sheets include peer-reviewed **homework** assignments.
The submission deadline is at 14:15 on the day of the following lecture.
Solutions are published once the deadline has passed.

You can ask questions about the homework on Zulip.

Peer Review

If you upload your homework on time, it will be anonymously peer-reviewed by **two fellow students**.

In exchange, you will be expected to review two other students' homework. The peer-review deadline is one week after the submission deadline.

Three requests:

- ▶ Please apply the **feedback sandwich** strategy: Surround your negative comments with praise.
- ▶ Please apply **this seminar's principles** to improve the quality of your own review writing.
- ▶ Please do not use **generative artificial intelligence assistants** to produce the reviews.

Example of Peer Review

I found your introduction generally well written and well structured. Nevertheless, I have identified a few points that can probably be improved.

- 1. The second and fifth paragraphs are very long. Faced with such long paragraphs, I tend to lose track of their main idea. Maybe break them down?*
- 2. I have noticed a few typos: “occurence” (should be “occurrence”), “red” (should be “read”), and “principal” (should be “principle”). At least the first typo would have been caught by a spellchecker. You should really use one, as a basic courtesy to your readers.*
- 3. Verb tenses seem to be used inconsistently. In the first paragraph, you write “The program takes 45 s to find a solution.” Yet in the fourth paragraph, you use the past tense and write “A solution was found almost immediately.”*

These issues should be easy to address, and you will end up with a very pleasant introduction.

Examinations

The regular and retake examinations are **on paper** and **closed book**. You will be given **90 minutes**. The outcome is **pass** or **fail**.

Registration via LSF is necessary to take the examination. It will open in the coming weeks.

The examination dates will be announced later via Moodle.

Three **practice examinations** are available on the seminar's web site.

Expectations

This will be an **interactive** and **hands-on** seminar.

We expect you to

- ▶ attend class regularly;
- ▶ participate in discussions and small-group exercises;
- ▶ be prepared to be called on at random;
- ▶ do the exercises and homework;
- ▶ do your share of peer reviewing;
- ▶ pass the examination with flying colors.

Selected Literature

- ▶ Paul R. Halmos, “**How to Write Mathematics**,” *L’Enseignement mathématique* 16(2), pp. 123–152, 1970.
This essay gives useful advice about and beyond writing mathematics.
- ▶ Lawrence C. Paulson, “**Researcher, Don’t Make Your Readers Scream!**” www.cl.cam.ac.uk/~lp15/Pages/Scream.html, accessed April 22, 2026.
This brief essay gives basic advice on research paper writing.
- ▶ Helen Sword, **Stylish Academic Writing**, Harvard University Press, 2012.
If you plan to buy only one book this semester, buy this one.
- ▶ Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts, **Mathematical Writing**, Mathematics Association of America, 1989.
These are lecture notes for a course similar to the present seminar.
Computer science writing is covered as well.

Selected Literature

- ▶ Mary-Claire van Leunen, **A Handbook for Scholars**, Revised Edition, Oxford University Press, 1992.

This guide is slightly outdated but still informative, with lots of bad-vs.-good examples and humor.

- ▶ Carol Fisher Saller, **The Subversive Copy Editor: Advice from Chicago**, University of Chicago Press, 2016.

This guide tells you how to polish your manuscript, as a copy editor would.

- ▶ Joseph M. Williams and Joseph Bizup, **Style: Lessons in Clarity and Grace** (12th edition), Pearson, 2016.

If you plan to buy only two books this semester, buy this one too.

- ▶ William Zinsser, **On Writing Well: The Classic Guide to Writing Nonfiction** (30th anniversary edition), Harper Perennial, 2006.

This is another classic. Its focus is more journalistic than scientific or technical.

Selected Literature

- ▶ William Strunk, Jr., **The Elements of Style**, Harcourt, Brace and Howe, 1920. This “little book” is probably the best known style guide. The second edition is posthumously coauthored by E. B. White.
- ▶ Steven Pinker, **The Sense of Style: The Thinking Person’s Guide to Writing in the 21st Century**, Viking, 2014. This style guide is thought-provoking and well worth a read, despite the subtitle.
- ▶ H. W. Fowler, revised and edited by Sir Ernest Gowers, **A Dictionary of Modern English Usage** (second edition), Oxford University Press, 1965. This is a classic style guide, organized alphabetically. It is dated but still enlightening and entertaining. Avoid the third and fourth editions.
- ▶ Bryan A. Garner, **Garner’s Modern American Usage** (fifth edition), Oxford University Press, 2023. Garner is the 21st-century Fowler. His guide keeps the alphabetical order.

Acknowledgments

Thanks to Martin Desharnais-Schäfer, Michael Kirsten, Elisabeth Lempa, Jannis Limperg, and Mark Summerfield for helping me develop the material.

Thanks to Benjamin C. Pierce and Rajeev Alur for allowing instructors to reuse material from their course *Writing and Speaking with Style*.

Several of my slides are based on theirs.

Thanks to Matthias Budde for permitting me to adapt some of his material on paper writing.

Thanks to Arnt Gulbrandsen for introducing me to the pyramid principle.

Thanks to an anonymous reader for wording suggestions.

Computer Aids to Writing

Dictionary Software

Dictionary applications such as Dictionary on Windows and the similarly named application on macOS as well as web sites such as Dictionary.com provide general dictionaries, thesauruses, and bilingual dictionaries.

- ▶ **General dictionaries** are monolingual.
They provide definitions and examples of English in English.
- ▶ **Thesauruses** list synonyms and antonyms of English words.
- ▶ **Bilingual dictionaries** can be used to map words or phrases from your native language to valid English.

Spell-Checkers and Grammar Checkers

As a minimum, run a spellchecker on your document before you share it with other people (e.g., your supervisor) or upload the final version.

Examples of spellcheckers:

- ▶ Linux and macOS include `aspell`.
- ▶ Most text editors, including Microsoft Word, Google Docs, and Overleaf, include spellcheckers. Some of these also check the **grammar** and even the **style**.
- ▶ The Grammarly and LanguageTool add-ons provide more ample **stylistic advice**. Both tools are available for free as well as in Pro and Premium editions.

Used properly, Grammarly and similar tools can be very useful. **But be careful:**

- ▶ Their suggestions are sometimes unsuitable.

Therefore, check every word critically.

Translation Tools

Tools such as DeepL Translate and Google Translate can be used to **translate** longer texts from your native language to English.

Used properly, translation tools can be very useful. **But be careful:**

- ▶ The output might be unidiomatic or even invalid English.
- ▶ The output might have a different meaning than the original.

Therefore, check every word critically.

Generative Artificial Intelligence Assistants

Chatbots such as ChatGPT and Copilot can be used to

- ▶ check **spelling** and **grammar**;
- ▶ ensure your text is written in the right **register** (e.g., formal English);
- ▶ **translate** between languages;
- ▶ **expand** a bullet list into continuous text;
- ▶ **create** new text.

Generative Artificial Intelligence Assistants

Used properly, chatbots can be very useful tools. **But be careful:**

- ▶ Check every word critically.
- ▶ Since you have no control over what chatbots do with your input, you should avoid providing them with sensitive data.
- ▶ **Using chatbots for generating new text is strongly discouraged.**
The produced text will usually be substandard (even if the English is smooth), and you might be plagiarizing without noticing, perhaps infringing copyright.

Also be aware of any organizational policies concerning chatbots (e.g., the plagiarism policy¹ of the Institute for Informatics of the LMU).

¹<https://www.medien.ifi.lmu.de/lehre/Plagiate-IfI.pdf>

Search Engines

Search engines such as Google and Bing can be used to determine **how frequently** certain phrases are used.

For example, using Google:

- ▶ *colour* yields about 3 360 000 000 results, whereas *color* yields about 14 400 000 000 results;
- ▶ *translate German into English* yields 26 results, whereas *translate German to English* yields about 123 000 results;
- ▶ *early binding in Java* yields about 5380 results, whereas *static binding in Java* yields about 6240 results.

Remember to put the phrases you search for in **double quotes** ("") and to **provide some context** (e.g., *in Java*).

Word Frequency Viewers

Tools such as Google Books Ngram Viewer are based on **vast corpora** of English.

You can use them to

- ▶ compare the **frequencies** of two or more words or short phrases;
- ▶ plot the frequency of words or phrases **over centuries**.

You can tell Ngram Viewer to consider only American or British English books.

A Sample Flawed Text

The authors of a Java book presented a small puzzling program and suggested that the readers compile and run it before reading the following.

*It seems obvious that the program, should print 1cafebabe. After all, that is the sum of the hex numbers 100000000_{16} and $cafebabe_{16}$. The program peruses `long` arithmetic which permits 16 hex digits, so arithmetic overflow is not an issue. Yet, if you ran the programm, you found that it prints `cafebabe`, with no leading 1 digit. This output represents the low-order 32 bits of the correct sum, but somehow the thirty third bit gets lost. It is as if the program was doing `int` arithmetic instead of `long`, or forget to add the first operator. What's going on here?*¹

¹Adapted from Joshua Bloch and Neal Gafter, *Java Puzzlers: Traps, Pitfalls, and Corner Cases*, Addison-Wesley Professional, 2005.

A Sample Flawed Text (with the Flaws Underlined)

The authors of a Java book presented a small puzzling program and suggested that the readers compile and run it before reading the following.

It seems obvious that the program, should print 1cafebabe. After all, that is the sum of the hex numbers 100000000_{16} and $cafebabe_{16}$. The program peruses long arithmetic which permits 16 hex digits, so arithemtic overflow is not an issue. Yet, if you ran the programm, you found that it prints `cafebabe`, with no leading 1 digit. This output represents the low-order 32 bits of the correct sum, but somehow the thirty_third bit gets lost. It is as if the program was doing `int` arithmetic instead of `long`, or forget to add the first operator. What's going on here?¹

¹Adapted from Joshua Bloch and Neal Gafter, *Java Puzzlers: Traps, Pitfalls, and Corner Cases*, Addison-Wesley Professional, 2005.

Flaws in the Text

- ▶ **Wrong comma:** *the program, should print*;
- ▶ **informality:** *print* vs. *output* or *display*;
- ▶ **bad taste:** *cafebabe*;
- ▶ **informality:** *hex* vs. *hexadecimal*;
- ▶ **“malapropism”:** *peruses* vs. *uses*;
- ▶ **missing comma:** *long arithmetic which permits*;
- ▶ **spelling mistake:** *arithemtic* vs. *arithmetic*;
- ▶ **spelling mistake:** *programm* vs. *program*;
- ▶ **missing hyphen:** *thirty third* vs. *thirty-third*;
- ▶ **wrong verb form:** *as if the program was doing* vs. *... were doing*;
- ▶ **wrong verb form:** *forget* vs. *forgetting*;
- ▶ **“malapropism”:** *operator* vs. *operand*;
- ▶ **informality:** *What's going on here?*

Microsoft Word

- ▶ Word wrongly suggests replacing *cafebabe* with *cafe babe*.
- ▶ Word correctly identifies *arithentic* and *programm* as spelling mistakes.
- ▶ Word wrongly suggests replacing *int* with words such as *into*.
- ▶ Word wrongly suggests removing the comma in the sentence *It is as if the program was doing int arithmetic instead of long, or forget to add the first operator.*

Grammarly

- ▶ Grammarly identifies the wrong comma in *the program, should print*, but it misses the missing comma in *long arithmetic which permits*.
- ▶ Grammarly wrongly suggests replacing *cafebabe* with *cafe babe*.
- ▶ Grammarly correctly identifies *arithemtic* and *programm* as spelling mistakes, and finds the missing hyphen in *thirty three*.
- ▶ Grammarly helpfully suggests replacing *forget* with *forgot* (although *forgetting* would have been better).
- ▶ Grammarly Pro also suggests removing *It seems obvious that* with the justification “Want to sound more confident?”
But this would damage the paragraph.

Copilot

Given the prompt “Can you flag the grammar and style mistakes in the following paragraph?” Copilot produced this output.

Sure, I'd be happy to help! Here are the flagged grammar and style mistakes in your paragraph:

1. **Comma usage:** *“It seems obvious that the program, should print 1cafebabe.” - The comma after “program” is unnecessary.*
2. **Word choice:** *“peruses” should be “uses” or “employs”.*
3. **Spelling:** *“arithemtic” should be “arithmetic”.*
4. **Spelling:** *“programm” should be “program”.*
5. **Word choice:** *“permits” should be “allows”.*

6. **Consistency:** *"cafebabe_16" should be consistent with "100000000_16" (either both with or without the underscore).*
7. **Hyphenation:** *"low-order" should be "low order".*
8. **Word choice:** *"thirty third" should be "thirty-third".*
9. **Verb tense:** *"was doing" should be "is doing" for consistency with the present tense.*
10. **Verb form:** *"forget" should be "forgot".*

Here's a revised version of your paragraph:

It seems obvious that the program should print 1cafebabe. After all, that is the sum of the hex numbers 100000000_16 and cafebabe_16. The program uses long arithmetic, which allows 16 hex digits, so arithmetic overflow is not an issue. Yet, if you ran the program, you found that it prints cafebabe, with no leading 1 digit. This output represents the low order 32 bits of the correct sum, but somehow the thirty-third bit gets lost. It is as if the program is doing int arithmetic instead of long, or forgot to add the first operator. What's going on here?

Does this help? Let me know if you need any further assistance!

Copilot

- ▶ Points 1, 2, 3, 4, and 8 are valid.
- ▶ Point 5 is also valid: *Allows* is slightly better than *permits*.
- ▶ Points 9 and 10 correctly identify mistakes but not the ideal remedies.
- ▶ Points 6 and 7 are wrong.
- ▶ The missing comma was silently inserted.
- ▶ There were no warnings for bad taste or informality, and one of the “malapropisms” (*operator*) slipped through.

Let us try a different prompt.

Copilot

Given the prompt “Can you make the revised paragraph more formal? Not solemn or hyperformal, just formal.” Copilot produced this output.

Certainly! Here's a more formal version of the revised paragraph:

It appears evident that the program should output 1cafebabe. This result is the sum of the hexadecimal numbers 100000000_16 and cafebabe_16. The program employs long arithmetic, which accommodates 16 hexadecimal digits, thereby preventing arithmetic overflow. However, if you executed the program, you observed that it outputs cafebabe, without the leading 1 digit. This output represents the lower 32 bits of the correct sum, but the thirty-third bit is somehow omitted. It seems as though the program is performing integer arithmetic instead of long arithmetic, or it failed to include the initial operator. What could be the cause of this issue?

Does this meet your needs? Let me know if there's anything else you'd like to adjust!

Copilot

- ▶ The new paragraph is more formal indeed:
hex vs. *hexadecimal*, *doing* vs. *performing*,
What's going on here? vs. *What could be the cause of this issue?*
- ▶ The new paragraph is slightly more formal than necessary:
seems obvious vs. *appears evident*, *use* vs. *employ*.
- ▶ *int* was confusingly changed to *integer*, a wider concept that encompasses *long*.
- ▶ Partly due to the needless changes, verbosity went up:
The paragraph went from 560 to 649 characters, a 16% increase.

If we **pick and choose**, we can construct a paragraph that combines Copilot's and our best ideas.

Final Revised Text

It seems obvious that the program should output 1deadc0de. After all, that is the sum of the hexadecimal numbers 100000000_{16} and $deadc0de_{16}$. The program uses `long` arithmetic, which supports 16 hexadecimal digits, so arithmetic overflow is not an issue. Yet, if you ran the program, you would have found that it displays `deadc0de`, with no leading 1 digit. This output represents the low-order 32 bits of the correct sum, but somehow the thirty-third bit gets lost. It is as if the program were performing `int` arithmetic instead of `long`, or forgetting to add the first operand. What could be the cause of this issue?