Lecture 10
Revisions

Prof. Dr. Jasmin Blanchette

Chair of Theoretical Computer Science and Theorem Proving

Version of October 21, 2025

Drafting

Drafting

You have collected ideas as bullet points, turned them into a sketch, and turned the sketch into text. The result is your **first draft**.

Starting from this rough first draft, you transform it into an acceptable **second draft** and an excellent **third draft.**

> If you remember only one thing from this book, please let it be this:
> revise the hell out of everything you write.
> —Kirsten Ghodsee, *From Notes to Narrative: Writing Ethnographies That Everyone Can Read* (2016)

First Draft

To write clearly, you must first **think** clearly,
but to think clearly, you might need to **write something** first.

Your first draft can be **very rough**. If writing a document is like building a sand castle, then the first draft corresponds to shoveling sand into the sandbox.

You can write the first draft **sequentially** or **in any order**.
If your sketch is precise enough, the order of writing is unimportant.

You would not usually circulate this draft. Let it be your little **secret**.

Recycling vs. Rewriting

If you have written about a similar topic before, it may be tempting to reuse old text.
Do not do this. The old text will likely not fit in the new context.
It will be too short, too long, or written for a different audience.
It might also be copyrighted by a publisher.

Use old text as inspiration, but **write new text**.

Second Draft

Armed with a first draft, you can **reread** and **improve** it.

It often helps to read the text aloud to check its **flow** and **rhythm**.
Try to put yourself in the shoes of readers from your target audience.

- You can make **large changes**, such as adding or removing an idea, expanding or compressing a passage, splitting a section, moving a section, moving content between sections, or even redrafting a section entirely.
- You can also start making **small changes**, such as altering the wording, splitting or joining sentences or paragraphs, or rewriting a sentence or paragraph.

Once your second draft is ready, circulate it among colleagues and friends to get their **feedback**.

Third Draft

Your third draft is a **polished** version of your second draft.
It incorporates both feedback from early readers and your own improvements.

At this point, usually only **small, local changes** are made.
There should be no need to rearrange the sections or to introduce new ones.

You can share the third draft with a **wider public**.

Incorporating Feedback

Readers of your drafts, or reviewers of a conference or journal submission, will give you **feedback**.

Sometimes you need to add a new paragraph or **refactor** some text. This might break the text's flow, and you should reread the new text and its surroundings with fresh eyes.

Some feedback can be addressed **locally**—e.g., adding or removing a comma, correcting a typo, adding a sentence.

Resist the temptation to incorporate feedback by **adding footnotes**.

Often, your draft can be improved simply by removing material; this is called **via negativa**.

Avoid **major changes** close to the final deadline.

Final Version

At some point, you prepare the final version of your document.
In the spirit of **wabi-sabi**, you must accept that it will not be perfect.

A work is never truly completed, but **abandoned**.

Make sure you **thank** anyone who helped you.
In a peer-reviewed paper, you can even acknowledge the anonymous reviewers.

Compressing

Amputation vs. Haircut

To meet a page limit or make your text sharper **shorten** it.

There are two basic approaches, which can be combined:

- With the **amputation**, you remove entire sections, paragraphs, or sentences.
- With the **haircut**, you reword individual sentences or paragraphs.

Amputation

The amputation is technically easier to do but requires **sacrificing content**.

You need to identify the least essential, or least interesting, parts of your text. Feedback from early readers can help.

You can often omit the largely pointless outline at the end of the introduction of a paper (*This paper is structured as follows: . . .*).

Haircut

The haircut is more difficult to do but ultimately more rewarding, because you can **avoid sacrificing content**.

- Look for **needless words** and **metadiscourse**.
- Look for consecutive sentences that can be **merged**.
- Look for paragraphs ending with a **word alone** on their last line.
- Look for item lists and displayed equations that can be **inlined**.
- Look for mathematical proofs that can be shortened or given in **outline**.

The abstract, introduction, background, conclusion, and any other **noncore** sections of a paper are prime candidates for a haircut.

Example of Haircut

Before (172 words):

> *In thread-based object-oriented languages, synchronous method calls usually provide the mechanism to transfer control from caller to callee, blocking the caller until the call is completed. This model of control flow is well-suited for sequential and tightly coupled systems but may be criticized in the concurrent and distributed setting, not only for unnecessary delays but also for the reasoning complexity of multithreaded programs. Concurrent objects propose an alternative to multithread concurrency for object-oriented languages, in which each object encapsulates a thread of control and communication between objects is asynchronous. Creol is a formally defined modeling language for concurrent objects which clearly separates intra-object scheduling from inter-object communication by means of interface encapsulation, asynchronous method calls, and internal processor release points. This separation of concerns provides a very clean model of concurrency which significantly simplifies reasoning for highly parallel and distributed object-oriented systems. This paper gives an example-driven introduction to these basic features of Creol and discusses how this separation of concerns influences analysis of Creol models.*[1]

[1] Einar Broch Johnsen, Jasmin Christian Blanchette, Marcel Kyas, and Olaf Owe, "Intra-Object versus Inter-Object: Concurrency and Reasoning in Creol," *Electronic Notes in Theoretical Computer Science* 243, pp. 89–103, 2009.

Example of Haircut

After haircut (131 words):

In thread-based object-oriented languages, control is usually transferred from caller to callee via synchronous method calls, which block the caller. This model is well suited for sequential and tightly coupled systems, but in concurrent and distributed settings, it causes needless delays and complicates reasoning about programs.

Concurrent objects are an alternative in which each object encapsulates a thread of control and objects communicate asynchronously. Creol is a formally defined modeling language for concurrent objects that cleanly separates intra-object scheduling from inter-object communication through interface encapsulation, asynchronous method calls, and internal processor release points. This separation of concerns greatly simplifies reasoning for highly parallel and distributed object-oriented systems. This paper introduces these Creol features with examples and discusses how the separation of concerns influences the analysis of Creol models.

Example of Haircut

In thread-based object-oriented languages, synchronous method calls usually provide the mechanism to transfer control from caller to callee, blocking the caller until the call is completed.

VS.

In thread-based object-oriented languages, control is usually transferred from caller to callee via synchronous method calls, which block the caller.

Example of Haircut

This model of control flow is well-suited for sequential and tightly coupled systems but may be criticized in the concurrent and distributed setting, not only for unnecessary delays but also for the reasoning complexity of multithreaded programs.

VS.

This model is well suited for sequential and tightly coupled systems, but in concurrent and distributed settings, it causes needless delays and complicates reasoning about programs.

Example of Haircut

Concurrent objects propose an alternative to multithread concurrency for object-oriented languages, in which each object encapsulates a thread of control and communication between objects is asynchronous.

VS.

Concurrent objects are an alternative in which each object encapsulates a thread of control and objects communicate asynchronously.

Example of Haircut

Creol is a formally defined modeling language for concurrent objects which clearly separates intra-object scheduling from inter-object communication by means of interface encapsulation, asynchronous method calls, and internal processor release points.

VS.

Creol is a formally defined modeling language for concurrent objects that cleanly separates intra-object scheduling from inter-object communication through interface encapsulation, asynchronous method calls, and internal processor release points.

Example of Haircut

This separation of concerns provides a very clean model of concurrency which significantly simplifies reasoning for highly parallel and distributed object-oriented systems.

vs.

This separation of concerns greatly simplifies reasoning for highly parallel and distributed object-oriented systems.

Example of Haircut

This paper gives an example-driven introduction to these basic features of Creol and discusses how this separation of concerns influences the analysis of Creol models.

vs.

This paper introduces these Creol features with examples and discusses how the separation of concerns influences the analysis of Creol models.

. . . and of Amputation

After haircut and amputation (110 words):

> *In thread-based object-oriented languages, control is usually transferred from caller to callee via synchronous method calls, which block the caller. This model is well suited for sequential and tightly coupled systems, but in concurrent and distributed settings, it causes needless delays and complicates reasoning about programs.*
>
> *Concurrent objects are an alternative in which each object encapsulates a thread of control and objects communicate asynchronously. Creol is a formally defined modeling language for concurrent objects that cleanly separates intra-object scheduling from inter-object communication through interface encapsulation, asynchronous method calls, and internal processor release points. This separation of concerns greatly simplifies reasoning for highly parallel and distributed object-oriented systems.*

Appendices and Technical Reports

Amputated text can be provided in **appendices** or in a **technical report**:

- With appendices, the material is **not in its natural location**.
  This is problematic for lemmas and theorems, which should precede their invocations.
- A technical report, providing the **long version** of your document, is often preferable.

Using a mechanism called conditional compilation, LaTeX can be used to generate multiple variants of a document from a **single source**. You can prepare a paper and its technical report **together**.

Conditional Compilation in Action

Let us analyze two research papers and their technical reports:

- Ahmed Bhayat and Giles Reger, "**A Combinator-Based Superposition Calculus for Higher-Order Logic**," *IJCAR 2020*, pp. 278–296, Springer, 2020.
- Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann, "**Superposition with Lambdas**," *CADE-27*, pp. 55–73, Springer, 2019.