**# Syllabus**

**## General Information**

- Class: CPSC 354 Programming Languages

- Instructor: [Alexander Kurz](https://alexhkurz.github.io/)

- Lectures in Leatherby Libraries Lab B13 (Aug 30 - Dec 18):

- 354-01: TuTh 8:30AM - 9:45AM

- 354-02: TuTh 10:00AM - 11:15PM

- Office Hours: TuTh after the second lecture 11:30 - 13:00 via [zoom](https://chapman.zoom.us/j/91616790270) ... or contact me via [email](mailto:akurz@chapman.edu?subject=CPSC-354).

- Course material: https://github.com/alexhkurz/programming-languages-2021

**## Course Description**

**\*\*The aim of the course\*\*** is to have a look under the hood of programming languages. How do programming languages work? Could you design your own programming language? Instead of looking at particular examples of programming languages, we will build our own. It will be a small language and the implementation will not be efficient, but we will be able to understand in all detail how it works.

Next semester, in Compiler Construction, we will demonstrate that the principles we learn this semester also apply to industrial scale programming languages such as C++.

**\*\*Prerequisites:\*\*** (MATH 250, CPSC 350) I assume that you know at least one, ideally a few more, programming languages. It would also be good to have learned something about computer architecture. One theme of the course is how to bridge the gap between a programming language and the actual machine, so some awareness of how actual machines work is needed to fully appreciate the material. Finally, while I will introduce the mathematics that we need to engineer our programming languages, some ability in manipulating formal mathematical models as typically acquired in a discrete mathematics or introductory logic course will be needed.

Note that this has been (slightly) modified from the current [course catalogue](https://catalog.chapman.edu/content.php?catoid=11&navoid=556).

**## Course Learning Outcomes**

See also the [Fowler School of Engineering Program Learning Outcomes](https://docs.google.com/document/d/1OESCtPUolnWFV\_qRFuRzNrzxmUtYr5H-dFaYVmPUKY0/edit?usp=sharing).

After completing this course, students will be able to

- explain how an interpreter for a small functional and imperative programming languages work

- use a context-free grammar and a parser generator such as BNFC

- add features of their own to the languages and interpreters

- explain syntax and semantics of various calculi and programming languages

- understand lambda-calculus as a foundation of programming languages and the concepts of operational and denotational semantics

- explain various features of programming languages based on the computational model of term-rewriting

- define and prove basic properties of term-rewriting systems

- understand the significance of abstract and algebraic data types and their relationship to universal algebra

- explain the basic principles of program verification

- use well-founded orders in termination proofs of simple programs

- use invariants in partial correctness proof of simple programs

- write Haskell programs that use recursion over algebraic data types

Moreover, students will be able to acquire the basic ideas of advanced topics such as theorem proving, dependent types and mathematics as a specification and programming language.

Finally, students will learn to appreciate that mathematics is not only important for developers who create the tools used in everyday engineering practice, but also that the knowledge of the mathematical concepts and results underpinning these engineering tools impact everyday engineering practice (and increase your chances to pass a coding interview). I discuss some aspects of this in a these notes on [mathematics as a programming language](https://hackmd.io/s/ByGLTvFDE).

**## Overview**

Together with Compiler Construction this course is the first one of a pair of courses in which we are going to learn the basic theory and practice of designing and implementing a programming language.

**#### Summary**

The course will have a practical and a theoretical component.

- *\*The theoretical component\** will teach some of the mathematics underpinning the design of programming languages. These will include some of the basics of logic, rewriting, ordered structures, universal algebra, and category theory (type theory will be deferred to next semester's course on compiler construction). Just enough theory to help the writing of interpreters and to gain an outlook on some of the questions guiding programming languages research. I made a [list of theoretical concepts](theory.md) that appear regularly in programming languages blogs and which we will learn about something.

- *\*The practical component\** will be about building interpreters for small programming languages. We will start with a calculator, that is, an interpreter for the language of high-school arithmetic, then go on to the smallest proper programming language known as lambda calculus. Lambda calculus provides variables and functions and other programming languages can be seen as extensions of lambda calculus. Thus, once we have an interpreter for lambda caclulus, we will approximate more realistic programming languages by extending lambda calculus: conditionals, recursion, data types, assignment, while loops, addresses (pointers), memory allocation, dereferencing, arrays. There will be room for invention and adventure if you feel like it. But there will also be many guided exercises.

**#### Detailed Description**

The course is divided in a practical and theoretical component.

On the practical side, students will learn

- the basics of Haskell

- how to build interpreters in Haskell for small programming languages of increasing complexity:

- numbers, addition, multipliciation, fractions, ...

- calculator

- lambda-calculus

- a small functional programming language

- a small imperative programming language

This will include the basics of parsing and the use of a parser generator. The aim is to treat programming languages that only have a small number of features. We will see that these methods scale in next semester's course on compiler construction.

On the theoretical side, students will learn the fundamental ideas of

- lambda calculus

- operational and denotational semantics

- term rewriting

- invariants and well-founded orders

- program verification and Hoare logic

- structural induction, universal algebra, and category theory

On the way, students will encounter different programming languages such as Dafny and Haskell and , if time permits, Lisp (Scheme), theorem provers such as Isabelle, dependently typed programming languages such as Idris as well as programming concepts such as algebraic data types, recursion, variable binding, polymorphism, and more.

During the course we will pay special attention to compositionality. For example, from a software engineering point of view, we see compositionality in the division between syntax and semantics, or, parsing and interpretation; the programming technique of recursion over abstract syntax trees is another incarnation; in program verification, we will separate termination from partial correctness, which will allow us to give a compositional calculus for program verification known as Hoare logic.

**## Required Text**

The technical content of the course will be distributed via github. For general background I recommend to read the book Goedel-Escher-Bach.

**## Course Materials**

All course materials will be made available via github.

**## Assessment**

Assessment will be divided in a total of 200 points.

- 3 programming [assignments](assignments.md) each worth 30 points (90 points total)

- Participation worth 10 points.

- A report worth 100 points.

**## Course Grade Breakdown**

Grading scale used for the course:

| Percentage | Letter |

|---|---|

| 90 | A |

| 80-89 | B |

| 70-79 | C |

| 60-69 | D |

| < 60 | F |

You must score a 70 or above to receive a P when taking the course P/NP.

**## Late Policy**

If you need more time for an assignment

- convince me that you already have done substantial work (for example by showing me the code in your GitHub repository);

- explain the special circumstances that would allow me to justify giving you more time.

**## Participation**

- It is expected that students attend every lecture and participate in the Discussion Forum.

- I also appreciate if students make use of the office hours, which gives me valuable feedback on how the class is going.

**## Policies required to be listed via University guidelines**

**#### Chapman University’s Academic Integrity Policy**

Chapman University is a community of scholars that emphasizes the mutual responsibility of all members to seek knowledge honestly and in good faith. Students are responsible for doing their own work and academic dishonesty of any kind will be subject to sanction by the instructor/administrator and referral to the university Academic Integrity Committee, which may impose additional sanctions including expulsion. Please see the full description of Chapman University's policy on Academic Integrity.

**#### Chapman University’s Students with Disabilities Policy**

In compliance with ADA guidelines, students who have any condition, either permanent or temporary, that might affect their ability to perform in this class are encouraged to contact the Office of Disability Services. If you will need to utilize your approved accommodations in this class, please follow the proper notification procedure for informing your professor(s). This notification process must occur more than a week before any accommodation can be utilized. Please contact Disability Services at (714) 516–4520 if you have questions regarding this procedure or for information or to make an appointment to discuss and/or request potential accommodations based on documentation of your disability. Once formal approval of your need for an accommodation has been granted, you are encouraged to talk with your professor(s) about your accommodation options. The granting of any accommodation will not be retroactive and cannot jeopardize the academic standards or integrity of the course.

**#### Chapman University’s Equity and Diversity Policy**

Chapman University is committed to ensuring equality and valuing diversity. Students and professors are reminded to show respect at all times as outlined in Chapman’s Harassment and Discrimination Policy. Please review the full description of Harassment and Discrimination Policy. Any violations of this policy should be discussed with the professor, the Dean of Students and/or otherwise reported in accordance with this policy.”

**#### Student Support at Chapman University**

Over the course of the semester, you may experience a range of challenges that interfere with your learning, such as problems with friend, family, and or significant other relationships; substance use; concerns about personal adequacy; feeling overwhelmed; or feeling sad or anxious without knowing why. These mental health concerns or stressful events may diminish your academic performance and/or reduce your ability to participate in daily activities. You can learn more about the resources available through Chapman University’s Student Psychological Counseling Services.

Fostering a community of care that supports the success of students is essential to the values of Chapman University. Occasionally, you may come across a student whose personal behavior concerns or worries you, either for the student’s well-being or yours. In these instances, you are encouraged to contact the Chapman University Student Concern Intervention Team who can respond to these concerns and offer assistance. While it is preferred that you include your contact information so this team can follow up with you, you can submit a report anonymously. 24-hour emergency help is also available through Public Safety at 714-997-6763.

**#### Religious Accommodation**

Religious Accommodation at Chapman University Consistent with our commitment of creating an academic community that is respectful of and welcoming to persons of differing backgrounds, we believe that every reasonable effort should be made to allow members of the university community to fulfill their obligations to the university without jeopardizing the fulfillment of their sincerely held religious obligations. Please review the syllabus early in the semester and consult with your faculty member promptly regarding any possible conflicts with major religious holidays, being as specific as possible regarding when those holidays are scheduled in advance and where those holidays constitute the fulfillment of your sincerely held religious beliefs.

**#### Changes**

This syllabus is subject to change. Updates will be posted on the course website.