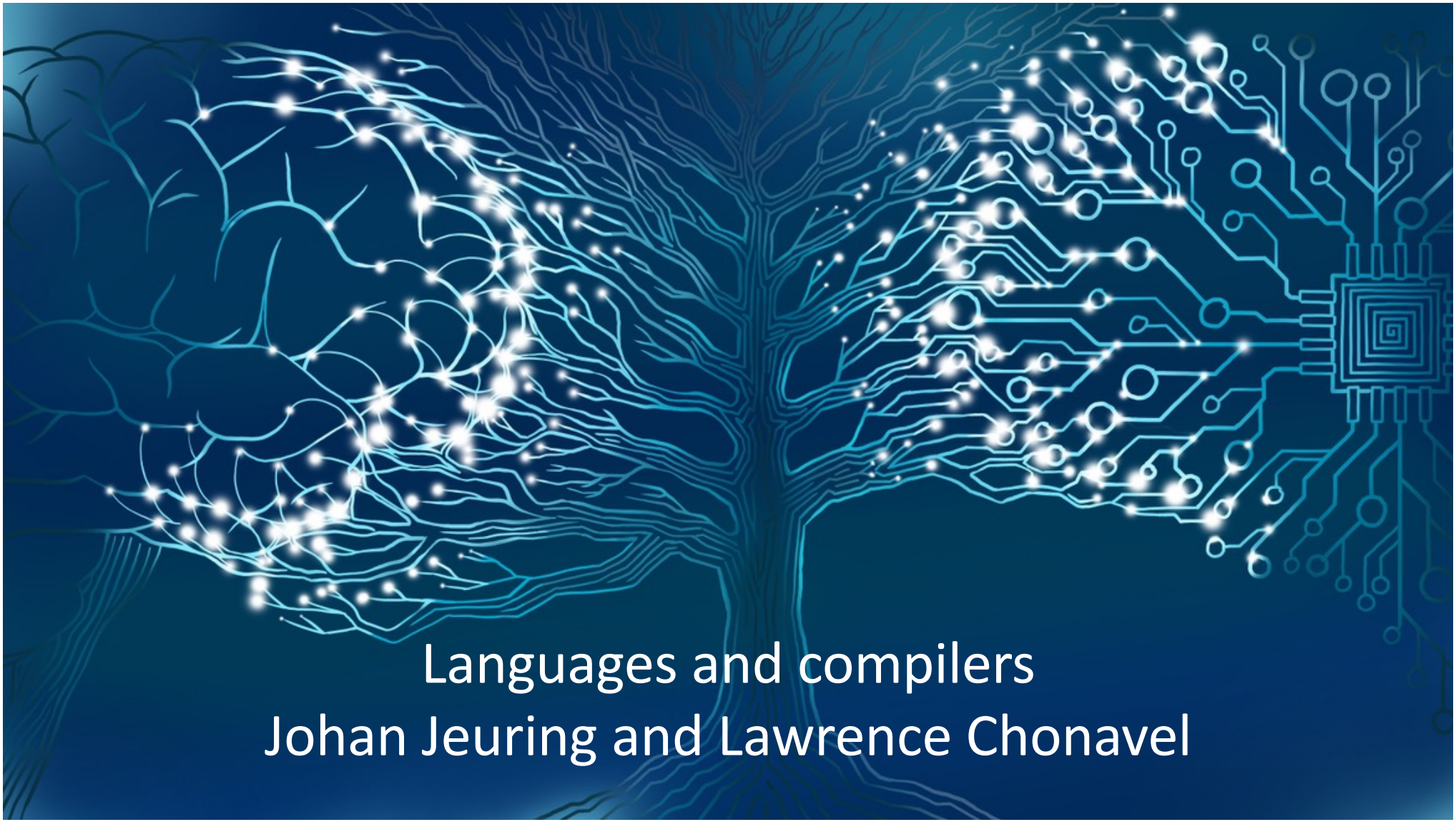

Languages and compilers
Johan Jeuring and Lawrence Chonavel

Who am I?

- Professor Software Technology for Learning and Teaching
- Information and Computing Sciences Departement and Freudenthal Institute
- Taught this course until 2017, after that head of department
- Now again teaching and research

Who are you?

- Go to wooclap.com
- Event code: ZFNFAM

Utrecht University

What is this course about?

Lecture notes: 1

Utrecht University

Information processing

We describe and transfer **information** by means of **language**

Information is obtained by assigning **meaning** to **sentences**

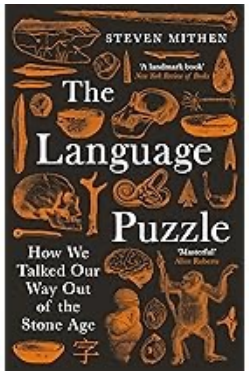
The **meaning** of a sentence is inferred from its **structure**

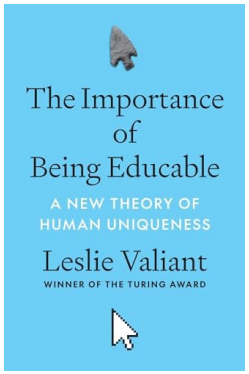
Utrecht University

Languages…

Language is the composition of words into meaningful utterances by using rules to modify and arrange them in a particular order.

Languages are used to educate or instruct.





Languages are essential for many human activities: verbal and written communication, musical scores, DNA, and also programming

Languages…

A language is a set of “correct” sentences.

What does that mean?

Are natural and formal languages different?

Are all languages equally expressive?

How can we decide if a sentence is correct?

How can we represent a correct sentence?

… and Compilers

A compiler translates one language into another (possibly the same). How?

Get hold of the structure of the input program

Attach semantics to a sequence of symbols

Check whether a program makes sense

Optimize

Generate good machine code

Learning goals

To **describe** language structures using **grammars**

To **parse**, i.e., to recognise (build) such structures in (from) a sequence of symbols

To **analyse** grammars to determine whether or not specific properties hold

To **compose** components such as parsers, analysers, and code generators

To **apply these techniques** in the construction of all kinds of programs

To **explain** and **prove** why certain problems can or cannot be described by means of formalisms such as context-free grammars or finite-state automata.

Utrecht University

Learning how?

- Lectures (Tuesdays, Thursdays)
- Labs
- Lab/exercise sessions (Tuesdays, Thursdays)
- Lecture notes

Learning requires effort and time

Can be both frustrating and (very!) satisfying

Utrecht University

Lectures

Johan Jeuring (Tuesdays)
Lawrence Chonavel (Thursdays)

We'll discuss most topics you need to learn in the course
But some topics might only appear in the lecture notes

Try to actively participate

- Asking questions is fine
- Participating in the online activities helps learning (and gives some bonus points)
- Don't disturb your fellow students

Utrecht University

Labs

- P0: Haskell refresh, does not contribute to your grade (but bonus)
- P1 – P3: practical and theoretical aspects of the contents of the course
- Work and submit in pairs (why?)
  Build teams of 2. No lab-partner? Mail j.t.jeuring@uu.nl
  Do not divide the work, but discuss it together

- Do not use GenAI (why?)
- AI-index level 1
- We will report fraud and plagiarism to the exam committee

Utrecht University

Oral exam

- For each pair, one lab will be the subject of an oral exam, to verify authorship
- If you fail, you won’t get a grade for the lab

Utrecht University

Exams

Midterm and a final exam in Remindo

Utrecht University

Exercises

- We will recommend exercises to solve to practice with the contents, and to prepare for the exam
- Help and feedback during the lab/exercise sessions. Sometimes a bonus
- Go to your assigned labroom

Bonus

- We will organize at most 20 (but probably fewer) activities at which you can earn a bonus point by participating
- Each bonuspoint adds 5 centipoints to your grade (0.05)
- Your final grade cannot be higher than 10

First three bonus activities:

- Wooclap in this lecture
- Showing that you work on P0 in the lab session
- Reporting a new error in the lecture notes (except exercises) to me. Report as often as you want (please!), but bonus just once

Utrecht University

Grade

Activity	Max points
P1	1
P2	1.5
P3	2.5
T1	2
T2	3
Bonus	<=1

You have to participate in all assessment activities

You can resit at most one assessment activity

Your grade for T1+T2 needs to be at least 2.5

You can participate in a resit assessment if your grade is at least 4

Haskell

Many of the learning goal components need to be described in a programming language. We will use Haskell for this.

Overview of FP concepts in Visual Studio, and Wooclap

Haskell and formal languages

Haskell	Formal languages
Datatype	Alphabet
List	Sequence
A concrete list	Sentence or word
Datatype	Abstract syntax
Parser	Grammar
Parser transformation	Grammar transformation
Value of an abstract syntax type	Parse tree
Fold function, algebra	Semantics

Utrecht University

Formal languages

Lecture notes: 2.1

What is a language?

A language is a set of sentences (or words)

Which sentences belong to a language, and why?

In natural languages, this is often informally defined and subject to discussion

For a formal language, we want a precise definition

Utrecht University

Sets

A **set** is a collection of elements

- No duplicates
- No order
- The empty set: Ø
- A nonempty set: {a,b,c,d,e}

Utrecht University

Alphabet

An **alphabet** is a (finite) set of symbols that can be used to form sentences.

- The set of all Latin letters
- {a,b,c,d,e}
- {0,1}
- The set of all ASCII characters
- The set of all Unicode code points
- {A,C,G,T}
- {if, then, else, do}
- {♫, ♪♪♪, ♭}

Utrecht University

Sequence

Given a set, we can consider (finite) sequences of elements of that set.

Let A= {a, b, c}. Examples of sequences over A:

- abc
- a
- acccabcabcabbaca
- bbbbbbbbb
- ε

The empty sequence is difficult to visualize.

ε is a placeholder to denote the empty sequence.

Utrecht University

Sequences, inductively

Given an arbitrary sequence over elements of a set A, we can make one of the two following observations:

- it is the empty sequence ε
- the sequence has a first element a ∈ A, and if we split off that element, the tail is still a (possibly empty) sequence z

We can use this observation to define sequences

Utrecht University

Sequences, inductively

Given a set A. The set of sequences over A, written A^*, is defined as follows

- The empty sequence ε is in A^*
- If $a \in A$ and $z \in A^*$, then az is in A^*

In such an inductive definition, it is implicitly understood that

- Nothing else is in A^*
- We can only apply the construction steps a finite number of times, i.e., only finite sequences are in A^*

Utrecht University

Sets and sequences

How many elements do the following sets have?

- $\emptyset$
- $\emptyset^*$
- {a, b, c}
- {if, then, else}*

Utrecht University

Language

Given an alphabet A, a **language** is a subset of A^*

Note that we consider any set X to be a subset of itself: $X \subseteq X$

So A^* is a valid language with alphabet A

Defining a language

A language is a set of sentence

How do we define such a set?

- By enumerating all elements?
- By using a predicate?
- By giving an inductive definition?

All of these are possible, and more

Utrecht University

An example

Consider the alphabet Latin = {a,b,c,d,…,z}

How do we describe the language L defined by

L = {thumb, index finger, middle finger, ring finger, little finger}

The language L is the language over the Latin alphabet consisting of all finger names

Languages by enumeration

Enumerating all elements of a language is impossible if the language is infinite

Most interesting languages are infinite: C#, Haskell, natural languages

Defining a language using a predicate seems more promising

Utrecht University

Languages by predicate: example

Let DNA be the alphabet {A,C,T,G}

Then PAL = $\{ s \in DNA^* \mid s=s^R \}$

where s^R is the reverse of s, is the language of **palindromes** over DNA

Utrecht University

Languages by induction: example

DNA palindromes can also be defined inductively:

- ε is a DNA palindrome
- A, C, T, G are DNA palindromes
- If P is a DNA palindrome, then so are APA, CPC, TPT, GPG

Utrecht University

By predicate versus by induction

Definition by predicate is (in this case) shorter

How can we check whether a given sequence is in PAL?

How can we generate all the words in PAL?

An inductive definition gives us more structure, and is self-contained, making it easier to explain **why** a sentence is in a language

Utrecht University

Summary

Concept	Definition
Alphabet	Finite set of symbols
Language	A set of words/sentences, i.e., sequences of symbols from the alphabet
Grammar	Next lecture: A way to define a language inductively by means of rewrite rules

This werkcollege: Haskell setup and P0