Principles of Programming Languages
Summer 2000
Dr. J.P.E. Hodgosn
Introduction

• Rationale

• Overview

• Some Fundamental Notions
Rationale

- Programming languages are the means whereby we express to a computer a problem which we wish to solve.

- The goal is to narrow the “cognitive gap” between the way in which we think about a problem and the way in which we express it for the machine.

- Processes need not be described procedurally.

- You program into a language. Not in a language.
Overview 1.

- Three paradigms: Functional, Logical, Imperative.

- Each paradigm is introduced through a specific language.

- For the functional and logical languages a computational model will be described.
Overview 2.

• The concept of the semantics of a language will be introduced in order to be able to indicate how the languages can be implemented.

• The imperative model will be used as an extended illustration of “operational semantics”.

• We will close with the Church-Turing thesis and the equipotence of the languages.
Overview 3.

- Because it is important to “get inside” each language there will be many (short) programming assignments.

- Other languages that illustrate important aspect of language theory will be discussed as necessary, but in less detail.

- A certain amount of fundamental notation is required to say things right. This will be introduced as and where appropriate.
Fundamentals

We need to have the following fundamental notions:

- A review of some basic language design goals.

- Grammars: More particularly Backus-Naur Form for specifying language syntax.

- Automata or Machines in General for specifying operational semantics.
Design Goals

- Data Abstraction
- Modularization
- Typing
Grammars 1.

- Backus Naur Form (BNF) invented to describe the syntax of Algol

- Uses simple Production Rules.

\[
decimal\_char ::= \langle 0 \rangle | \langle 1 \rangle | \langle 2 \rangle | \langle 3 \rangle | \langle 4 \rangle | \langle 5 \rangle | \langle 6 \rangle | \langle 7 \rangle | \langle 8 \rangle \\
\]

\[
list ::= \text{nil} | \\
\text{'}(\text{'}, \text{items}, \text{')'} \\
\]

\[
items ::= \text{item, white, items} | \text{item} \\
\]
Grammars 2.

- Context Free grammars

- \( \langle \text{nonterminal} \rangle \rightarrow \text{body} \)

- A \textit{body} is a sequence of one or more items separated by commas. Each item is either a nonterminal sequence of terminal symbols.
Machines 1.

A *machine* consists of:

- A set $S$ of *states*,

- A set $\Lambda$ of *inputs*

- A set $\Omega$ of *outputs*

- A *transition mapping* $\Lambda \times S \rightarrow \Omega \times S$
Machines 2.

- Machines can be *acceptors* that output only when a certain input string is recognized.

- Machines can be translators, that turn input strings from \( \Lambda \) into output strings on \( \Omega \)
Machines 3.