Scheme

Adam Boult (www.bou.lt)

November 2, 2023

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Simple Lisp without functions

Chapter 1

Functions and recursion

1.1 Functions

1.1.1 Intro

Also called subroutines

1.1.2 Functions

We can call a function: $y = f(x)$

Here x are the local variables to be used.

However the global state may also affect the outcome, so we have:

$y = f(x, z)$

As a result, calling the same function twice with the same inputs can have different outputs.

Side effects. If a function modifies the state outside of its local variables it has side effects.

1.2 Recursion

1.2.1 Stack overflows

Write too many instructions to stack. can be caused by infinite recursion. eg fun x() return x()).

1.3 Other

1.3.1 Global and local variables

Chapter 2

The Church-Turing thesis

2.1 Storing knowledge

2.1.1 Introduction

2.2 The Church-Turing thesis

2.2.1 Introduction

Chapter 3

Imperative and functional programming

3.1 Introduction

3.1.1 Imperative programming

In imperative programming, we say exactly what we want to happen. Each step changes the state.

3.1.2 Functional programming

With functional programming, we write functions to be called. These functions should not depend on the state, outside of local variables.

3.1.3 Side effects

If we remove side effects from all functions then functional programming has no global variables which could affect the output.

Chapter 4

Lisp

Part II

Functions in Lisp

Part III

Other Lisp