Concepts in Prog Langs - Part 1

/

From a mathematical perspective, what is a program?

-

A definition of a function.

/

What determines the output of a program?

-

The inputs and the state prior of the machine prior to program start.

/

Why might an expression not have a value?

-

Error termination or nonterminations.

/

What is a partial function?

-

A function with value some arguments but not all.

/

What is the domain of a function?

-

All possible inputs.

/

When is a function f: A -> B computable.

-

When an algorithm exists that given any x contained in A as input (the domain), halts with y = f(x) as output, (the range).

/

What is the class of partial recursive functions?

-

The class of functions on natural numbers that are computable in principle.

/

What did Alonzo Church do?

-

proposed Church’s thesis.

/

What is Church’s thesis?

-

A widely held belief about the relationship between mathematical definitions and the real world of computing, stating that the same class of functions on the integers can be computed by any general computing device. This is the class of partial recursive functions, sometimes called the class of computable functions.

/

What are the three equivalent definitions of the class of computable functions.

-

A mathematical one, a second based on turing machines, and the lambda calculus.

/

What does it mean if a programming language is Turing complete mean?

-

That it can express the class of partially recursive functions. (pretty much all standard programming languages.)

/

What is an example of an undecidable program?

-

The halting problem.

/

What is the halting problem?

-

Try to write a program that will tell you another program will stop. You can’t. You need to halt to report if the program is un-halting, just after which the program could halt. Proof uses a chain of programs to develop a contradiction. Program P with input x, then program Q(P, x), then D(P) such that it runs forever if Q(P, x) halts, otherwise if Q(P, x) runs forever it halts.

/

Who had a lisp and liked parenthesis?

-

John McCarthy

/

What are the three most basic parts of speech for a programming language?

-

Expressions, Statements, and Declarations

/

What is an expression?

-

A syntactic entity that may be evaluated to determine its value.

(may also change the state of the machine in addition to producing a value.)

/

What is a statement?

-

A command that alters the state of the machine an explicit way.

/

What is a declaration?

-

A syntactic entity that introduces (declares) a new identifier.

/

Which of the basic parts of speech is Lisp based on?

What about Assembly?

-

Expression

Statements

(Are declarative languages based on declarations?)

/

What type of expression did Lisp introduct?

-

Conditional expressions that produce a value instead of causing a jump.

/

What are the four components of the Lisp abstract machine?

-

A Lisp expression

A continuation – function representing part of program left to evaluate.

Association (A) list – run time stack

A heap – memory reference by pointers in the A-list. (set of cons cells)

/

Five basic ops on a cons cell.

-

atom – special bit pattern where function returns true if atom false otherwise.

eq – equality

cons – construct x and y return pointer z (combine two lists into a new list)

car – return address register (head of list)

cdr – return decrement register (tail of list – everything but head)

/

What distinct runtime feature did lisp introduce?

-

The ability to construct and execute a program.

/

What are Lisp computations based upon instead of assignment and iterative loops?

-

functions and recursive calls.

/

What is a high order function?

-

A function that either takes other functions as input or produces them as output.

Different from first order functions (input and results are not functions).

Or, the set of all non first order functions (2nd order, 3rd order, …)

2nd order takes a 1st order and so on.

/

What is garbage?

-

Parts of memory not accessible to a program. The automatic collections of which was introduced by Lisp.

/

What do pure lisp expression not have, that many other programming language expressions do?

-

Side Effects

/

What do you call the text of a program?

-

Syntax

/

What do you call the thing the program does?

-

Semantics

/

“Programming languages that are convenient for people to use are built around concepts and abstractions that may not correspond directly to features of the underlying machine.”

-

/

What does a compiler do?

-

Translates a programming language into machine code.

/

What does an interpreter do?

-

Combines translation and program execution.

/

What are the six typical phases of a compiler, from source to target?

-

Lexical Analysis – input symbols scanned and turned into tokens

Syntax Analysis – tokens grouped into syntactic units: expressions, statements and declarations. Also called parsing.

Semantic Analysis – Apply rules and procedure depending upon the context of expressions, such as type checking. (different from the more common use of the word: “meaning”)

Intermediate code generator – not necessarily target machines code, i.e. java byte code.

Code optimizer

Code generator – choose memory location, register or both for each variable.

/

What are five standard optimizations?

-

Common Subexpression Elimination

Copy Propagation (eliminate pointer chains on assignment: z=y, y=x to z=x)

Dead-Code Elimination (places never reached)

Loop Optimization (remove expressions)

In-Lining Function Calls

/

What is a grammar?

-

A method for defining an infinite set of expressions.

/

Does BNF have anything to do with BFFs?

-

No, BNF is Backus Normal Form.

Not sure on the relation with <a href="http://www.w3schools.com" target="\_blank">Bacchus</a>

/

Example BNF

e ::= n | e + e | e - d

n :: = d | nd

d ::= 0 | 1 | 2| 3 | 4 | 5 | 6 | 7

What is e?

What are terminals and non-terminals?

-

e is the start symbol

0 – 7, +, - are terminals

e, n, d or non-terminals

/

What is a derivation?

-

A sequence of steps converting an expression into a string of terminals.

/

What is a parse tree?

-

A representation of the derivation in graph form (a tree).

/

What is an ambiguous grammar?

-

A grammar where an expression has more than one parse tree.

/

What two parsing conventions help to eliminate ambiguous grammars?

-

1) Precedence (i.e. \* before +)

2) Associativity (i.e. 1 + 3 + 7, evaluate from left to right or vice versa)

/

What is the lambda calculus

-

Mathematical system that can illustrate programming concepts.

/

What are the three main parts of lambda calculus?

-

1) Notation for defining functions

2) Proof system for proving equations

3) Set of calculations called reductions

/

Is lambda calculus a type of calculus (analysis)?

-

No, its calculation is a form of symbolic evaluation of expressions.

/

What two concepts, that are common to almost all programming languages, does the lambda calculus provide fundamental forms of?

-

1) parameterization (through function expression)

2) binding (by declarations)

/

What the two components of the pure form of lambda calculus?

-

1) Function definition (lambda abstraction)

2) Function application

There are no domain-specific operators such as addition.

i.e. h(x) = f(g(x)) but not h(x) = f(x + x)

It is possible to add domain-specific operators

/

What is the identity function?

-

&lambda;x.x

more familiar: I(x) = x

/

Pure lambda calculus plus additional data types equals what?

-

A programming language (or applied lambda calculus)

/

What is the BNF grammar for lambda calculus?

-

M ::= x | MM | &lambda;x.M

/

What is the form of (&lambda;.x.x)5?

What is it called?

-

The form is MM where M1 = &lambda;.x.x and M2 = 5

This is called an application

/

What is the form of &lambda;x.(f(gx))?

What is it called?

-

The form is &lambda;x.M.

This is called a lambda abstraction

/

What are the two possible states of a variable?

-

Either the free fighter or the engaged fighter (free or bound).

/

What makes a variable free?

-

If it isn’t declared, i.e. x + 3

/

What is &lambda; called?

-

The binding operator. It binds a variable with a specific scope.

/

What is scope?

-

Part of an expression.

/

Is x in &lambda;x.M bound or free?

-

It is bound.

/

What does it mean if a variable is bound?

-

That it is just a placeholder. The meaning of x does not depend on x in &lambda:x.M.

i.e. x in integral &int; f(x) or in the logical formula &forall; x.P(x)

So &lambda:x.x = &lambda:y.y just like &int; f(x) = &int; f(y)

/

What are alpha equivalents?

-

Expressions that differ only in bound variable names.

/

In &lambda;x.M, what is M called?

-

The scope of the binding &lambda;x

A variable appearing in an expression M is bound if it appears in the scope of some &lambda;x and is free otherwise.

(good set theoretic description in book)

/

In &lambda;x.(&lambda;y.xy)y what are the different x and y terms called?

-

Fist x is binding occurrence

Second x is a bound occurrence (variable)

First y is a binding occurrence

Second y is a bound occurrence

Third y is a free occurrence

/

What is the axiom of beta equivalence?

-

The central equational axiom. (three others along with four inference rules)

(&lambda;x.M)N = [N/x]M

Or substituting occurrence of x in M with N

Parallels exist with macroexpansion and inline substitution of funcions.

/

What involves find fixed points?

-

Recursive declarations

/

What is a fixed point of a function?

-

A value such that the function produces the value, i.e. G(f) = f

/

In lambda calculus how are fixed points defined?

-

Defined with a fixed-point operator:

Y = &lambda;f.( &lambda;x.f(xx))( &lambda;x.f(xx))

Where for any f, Yf is a fixed point of f: Yf = f(Yf)

(the Y combinator)

/

What does denotational semantics refer to?

-

A style of mathematical semantics for imperative programs, where the meaning of a simple program is a mathematical function from states to states, in which a state is a mathematical function representing the values in memory.

/

What are natural languages?

-

The languages humans speak and write.

/

What are the four main kinds of sentences?

-

Imperative (command to implicit subject: Pick up the fish.)

Declarative (expresses a fact: Claude likes bananas.)

Interrogatives (questions)

Exclamatory (interjection: Ugh!)

/

What are two ways of defining the phrase functional language?

-

A language where most computation is done by evaluation of expressions containing functions.

Or

A language that does not have side effects.

/

What is the declarative language test?

-

Within the scope of of specific definitions x1, …, xn, all occurrences of an expression e containing only variables x1, …, xn have the same value.

Pure functional languages pass this test. Called referentially transparent.

/

What’s appealing about functional languages for optimization and execution?

-

Optimization – if an expression appears multiple times in a scope, it only needs to be computed once since there are no side effects.

Execution – expressions can be computed in parallel since no state needs to be maintained between different expressions.