Formulas: single variable, F(defined), \( \lambda \). Application

Variable directly after \( \lambda \) is bound, any occurrence after is bound to \( \lambda \) else free

\eta\text{ reduction: } \lambda x.f(x) \rightarrow f\text{ where } f(x)\text{ is a function}

Reducing: renaming and substituting until it is put into normal form.

Extension: set of definitions which augment a language with a new facility that can be used in the same way that preexisting facilities are used.

Lambda calculus defines a minimal semantic basis for computation. Using variables, lambda expressions, and applications, lambda calculus can represent program computations.

Symbols: \( x \) (variable), \( ?? \) (expression), and \( \ast \) (application) are formulas

A language's semantics are extendable if they may be augmented with new actions, data types, or control structures.

### Syntax

The syntax of a language is its grammatical rules. These are usually defined through EBNF (Extended Backus-Naur Form) and/or syntax diagrams. The meaning of a program is represented by program code or by a computation tree. The language syntax defines the computation tree that corresponds to each legal source program.

#### Chapter 1 - The Nature of Language

4. Syntax touches on the grammar/symbols - the set of rules, and principles that govern the structure of sentences in a given language. The semantics specifies the meaning attached to each placement of a word in a sentence, the meaning of a sentence as the whole, the meaning of omitted a sentence element, and the meaning of each individual word. Semantics.

English, in English, very redundant, meaning can be ascertained despite many of the potential errors. Whereas programming languages are partly redundant, and the required redundancy serves as a way to identify errors: (i.e., overloading functions, blocks, explicit/duplicate type declarations). Implicit communication: English - read between the lines. However, when learning a new language, a programmer must learn its implicit assumptions, more commonly called defaults. If a programmer relies on defaults to convey meaning, they are structurally strict. Structural rules govern the order of statements and sections of code and particular ways to begin, punctuate and end every program.

Redundancy: In English, very redundant, meaning can be ascertained despite many errors. Whereas programming languages are partly redundant, and the required redundancy serves as a way to identify errors: (i.e., overloading functions, blocks, explicit/duplicate type declarations). Implicit communication: English - read between the lines. However, when learning a new language, a programmer must learn its implicit assumptions, more commonly called defaults. If a programmer relies on defaults to convey meaning, they are structurally strict. Structural rules govern the order of statements and sections of code and particular ways to begin, punctuate and end every program.

Redundancy: In English, very redundant, meaning can be ascertained despite many errors. Whereas programming languages are partly redundant, and the required redundancy serves as a way to identify errors: (i.e., overloading functions, blocks, explicit/duplicate type declarations). Implicit communication: English - read between the lines. However, when learning a new language, a programmer must learn its implicit assumptions, more commonly called defaults. If a programmer relies on defaults to convey meaning, they are structurally strict. Structural rules govern the order of statements and sections of code and particular ways to begin, punctuate and end every program.
Chapter 8: Names are just an English string used to properly indicate syntax; for human ease to read code. Symbol table is a data structure that maintains all names and their definitions during run time. Type is stored here as well. Static binding creates an association (similar to a pointer) between a name (in the symbol table) and a storage object (an area of memory). For Dynamic binding (languages with scope) symbol table is called a dictionary and contains name field (name), link field (to organize dictionary to link to because a pointer to storage is used to refer to the data rather than the name). When multiple-name binding is used, storage is not allocated for the second name, but it is bound to the same address as the first and serves as a second way to refer to the same storage object. In a constant declaration, if the constant value is defined by an expression, that expression will be evaluated once at compile time. In a macro definition, the expression will be evaluated at run time every time the constant name is used. The scope of a name is that part of the program in which the name is known and will be recognized by the translator. Scope can be global (name is known throughout the program), local (it is only known within that program block which defined it). In a block structured language, a name becomes visible when it is born and invisible when it is masked by a declaration in an inner block, and it becomes visible again when the masking variable dies. All binding of symbolic name to storage location is done within the compiler closed.

*** Compiled Language Systems: enable the programmer to enter, compile, and link programs; when execution is done, control returns to the OS or to the shell. Interactive Language Systems (Prolog): These languages are embedded in subroutines. Semantics: The rules for interpreting the meaning of programming language items.

Chapter 5 -- Primitive Types

Computer Memory: Computer memory are arrays of bits organized into groups. These bits are grouped in 8 to form a byte. 2 and 4 bytes are called a word and a long word. These bytes and words form the basis for all computation. Character Code: Common encoding such as ASCII, EBCDIC are imposed on bit strings. There are several ways to represent a negative integer. For example, one bit can be interpreted as the sign and the rest as the magnitude to impose meaning beyond the bits. Ex: ASCII uses 7 bits to represent 128 characters. Packed Decimal: This is used to implement decimal (fixed-point arithmetic) which has two integer fields. One represents the magnitude and the other the position of the decimal point. Negative Numbers: There are several ways to represent a negative integer. For example, one bit can be interpreted as the sign and the rest as the magnitude. Floating Point: The IEEE has a standard for floating-point representation and computation which supports float of three lengths: 4, 8, and 10 bytes. The standard covers all aspects of floating point. The sign bit is always at the left end. Floating points also have an exponent and mantissa part. Builtin functions and comparison work with primitive type but user defined types are not as convenient or easy to use.
entry, otherwise the number of iterations is unpredictable. The most general form before entering the loop, an expression whose result must be interpreted as table and an increment after the scope of the loop. Implicit looping in functional is used in some pattern matching languages, such as Prolog. The implementation of time its parameter is used, and the result is stored for future use.196:197: *** Chapter 10 Basic Control...240: backtracking uses a stack to store pointers to the positions at which each t
241: 250: Call: This instruction, also called jump to subroutine, saves the current va
251: 254: GOTO the outside of the outermost loop.262: When exceptions occur, which include hardware errors, software errors, and l
263: handler takes action before resuming the continuation.266: -Functional languages do not have sequences of statements, but need a way to
267: 268: -Concept of continuation exists in all programming languages, but higher-ord
269: 270: be executed.271: -Packaging a continuation is like establishing a checkpoint; includes prog
- How exceptions arise; 1) Hardware error trigger interrupt signal, 2) System identifies an error (e.g. subscript outside of bounds), 3) User function identifies inconsistent situations
- Ignoring exceptions discourages locality of effects.
- Hardware detects exception and generates an interrupt signal which is processed by OS. OS will then set status flags
- Software Exception: some languages provide a general exception handling control structure
- Passing Control: When exception occurs it is more useful to propagate error to where it can be handled by passing the exception up the chain of calls.
- Downside of above is error handling code is intermingled with normal code, and intermediate routines need to have propagation code even though they have nothing to do with the error.
- Propagate by popping stack frames until handler is found or it returns to system.
- Handler code is translated in the context of its enclosing block.
- Being able to raise an exception by a specific name, provides more context about the cause of the problem.