

# *Glossary*

**absolute addressing** An address mode in the Mini architecture which stores a complete memory address in a single instruction field.

**action** An executable statement or procedure, often used in association with an automaton or program specification tool.

**action symbols** Symbols in a translation grammar enclosed in braces { } and used to indicate output or a procedure call during the parse.

**action table** A table in LR parsing algorithms which is used to determine whether a shift or reduce operation is to be performed.

**algebraic transformations** An optimization technique which makes use of algebraic properties, such as commutativity and associativity to simplify arithmetic expressions.

**alphabet** A set of characters used to make up the strings in a given language.

**ambiguous grammar** A grammar which permits more than one derivation tree for a particular input string.

**architecture** The definition of a computer's central processing unit as seen by a machine language programmer, including specifications of instruction set operations, instruction formats, addressing modes, data formats, CPU registers, and input/output instruction interrupts and traps.

**arithmetic expressions** Infix expressions involving numeric constants, variables, arithmetic operations, and parentheses.

**atom** A record put out by the syntax analysis phase of a compiler which specifies a primitive operation and operands.

- attributed grammar** A grammar in which each symbol may have zero or more attributes, denoted with subscripts, and each rule may have zero or more attribute computation rules associated with it.
- automata theory** The branch of computer science having to do with theoretical machines.
- back end** The last few phases of the compiler, code generation and optimization, which are machine dependent.
- balanced binary search tree** A binary search tree in which the difference in the heights of both subtrees of each node does not exceed a given constant.
- basic block** A group of atoms or intermediate code which contains no label or branch code.
- binary search tree** A connected data structure in which each node has, at most, two links and there are no cycles; it must also have the property that the nodes are ordered, with all of the nodes in the left subtree preceding the node, and all of the nodes in the right subtree following the node.
- bison** A public domain version of yacc.
- bootstrapping** The process of using a program as input to itself – as in compiler development – through a series of increasingly larger subsets of the source language.
- bottom up parsing** Finding the structure of a string in a way that produces or traverses the derivation tree from bottom to top.
- byte code** The intermediate form put out by a java compiler.
- closure** Another term for the Kleene \* operation.
- code generation** The phase of the compiler which produces machine language object code from syntax trees or atoms.
- comment** Text in a source program which is ignored by the compiler, and is for the programmer's reference only.
- compile time** The time at which a program is compiled, as opposed to run time. Also, the time required for compilation of a program.

- compiler** A software translator which accepts, as input, a program written in a particular high-level language and produces, as output, an equivalent program in machine language for a particular machine.
- compiler-compiler** A program which accepts, as input, the specifications of a programming language and the specifications of a target machine, and produces, as output, a compiler for the specified language and machine.
- conflict** In bottom up parsing, the failure of the algorithm to find an appropriate shift or reduce operation.
- constant folding** An optimization technique which involves detecting operations on constants, which could be done at compile time rather than at run time.
- context-free grammar** A grammar in which the left side of each rule consists of a nonterminal being rewritten (type 2).
- context-free language** A language which can be specified by a context-free grammar.
- context-sensitive grammar** A grammar in which the left side of each rule consists of a nonterminal being rewritten, along with left and right context, which may be null (type 1).
- context-sensitive language** A language which can be specified by a context-sensitive grammar.
- conventional machine language** The language in which a computer architecture can be programmed, as distinguished from a microcode language.
- cross compiling** The process of generating a compiler for a new computer architecture, automatically.
- DAG** Directed acyclic graph.
- data flow analysis** A formal method for tracing the way information about data objects flows through a program, used in optimization.
- dead code** Code, usually in an intermediate code string, which can be removed because it has no effect on the output or final results of a program.
- derivation** A sequence of applications of rewriting rules of a grammar, beginning with the starting nonterminal and ending with a string of terminal symbols.

**derivation tree** A tree showing a derivation for a context-free grammar, in which the interior nodes represent nonterminal symbols and the leaves represent terminal symbols.

**deterministic** Having the property that every operation can be completely and uniquely determined, given the inputs (as applied to a machine).

**deterministic context-free language** A context-free language which can be accepted by a deterministic pushdown machine.

**directed acyclic graph (DAG)** A graph consisting of nodes connected with one-directional arcs, in which there is no path from any node back to itself.

**disjoint** Not intersecting.

**embedded actions** In a yacc grammar rule, an action which is not at the end of the rule.

**empty set** The set containing no elements.

**endmarker** A symbol,  $\epsilon$ , used to mark the end of an input string (used here with pushdown machines).

**equivalent grammars** Grammars which specify the same language.

**equivalent programs** Programs which have the same input/output relation.

**example (of a nonterminal)** A string of input symbols which may be derived from a particular nonterminal.

**expression** A language construct consisting of an operation and zero, one, or two operands, each of which may be an object or expression.

**extended pushdown machine** A pushdown machine which uses the replace operation.

**extended pushdown translator** A pushdown machine which has both an output function and a replace operation.

**finite state machine** A theoretical machine consisting of a finite set of states, a finite input alphabet, and a state transition function which specifies the machine's state, given its present state and the current input.

- follow set (of a nonterminal, A)** The set of all terminals (or endmarker  $\$$ ) which can immediately follow the nonterminal A in a sentential form derived from S.
- formal language** A language which can be defined by a precise specification.
- front end** The first few phases of the compiler, lexical and syntax analysis, which are machine independent.
- global optimization** Improvement of intermediate code in space and/or time.
- goto table** A table in LR parsing algorithms which determines which stack symbol is to be pushed when a reduce operation is performed.
- grammar** A language specification system consisting of a finite set of rewriting rules involving terminal and nonterminal symbols.
- handle** The string of symbols on the parsing stack, which matches the right side of a grammar rule in order for a reduce operation to be performed, in a bottom up parsing algorithm.
- hash function** A computation using the value of an item to be stored in a table, to determine the item's location in the table.
- hash table** A data structure in which the location of a node's entry is determined by a computation on the node value, called a hash function.
- Helper** A macro in SableCC, used facilitate the definition of tokens.
- high-level language** A programming language which permits operations, control structures, and data structures more complex than those available on a typical computer architecture.
- identifier** A word in a source program representing a data object, type, or procedure.
- implementation language** The language in which a compiler exists.
- inherited attributes** Those attributes in an attributed grammar which receive values from nodes on the same or higher levels in the derivation tree.
- input alphabet** The alphabet of characters used to make up the strings in a given language.

- intermediate form** A language somewhere between the source and object languages.
- interpreter** A programming language processor which carries out the intended operations, rather than producing, as output, an object program.
- jump over jump optimization** The process of eliminating unnecessary Jump instructions.
- keyword** A word in a source program, usually alphanumeric, which has a predefined meaning to the compiler.
- language** A set of strings.
- left recursion** The grammar property that the right side of a rule begins with the same nonterminal that is being defined by that rule.
- left-most derivation** A derivation for a context-free grammar, in which the left-most nonterminal is always rewritten.
- lex** A lexical analyzer generator utility in the Unix programming environment which uses regular expressions to define patterns.
- lexeme** The output of the lexical analyzer representing a single word in the source program; a lexical token.
- lexical analysis** The first phase of the compiler, in which words in the source program are converted to a sequence of tokens representing entities such as keywords, numeric constants, identifiers, operators, etc.
- LL(1) grammar** A grammar in which all rules defining the same nonterminal have disjoint selection sets.
- LL(1) language** A language which can be described by an LL(1) grammar.
- load/store architecture** A computer architecture in which data must be loaded into a CPU register before performing operations.
- load/store optimization** The process of eliminating unnecessary Load and Store operations.
- local optimization** Optimization applied to object code, usually by examining relatively small blocks of code.

- loop invariant** A statement or construct which is independent of, or static within, a particular loop structure.
- LR** A class of bottom up parsing algorithms in which the input string is read from the left, and a right-most derivation is found.
- LR(k)** An LR parsing algorithm which looks ahead at most  $k$  input symbols.
- method** In Java, a sub-program with zero or more parameters, belonging to a particular class.
- multiple pass code generator** A code generator which reads the the intermediate code string more than once, to handle forward references.
- multiple pass compiler** A compiler which scans the source program more than once.
- natural language** A language used by people, which cannot be defined perfectly with a precise specification system.
- newline** A character, usually entered into the computer as a Return or Enter key, which indicates the end of a line on an output device. Internally, it is usually coded as some combination of 10 and/or 13.
- nondeterministic** Not deterministic; i.e., having the property that an input could result in any one of several operations, or that an input could result in no specified operation (as applied to a machine).
- nonterminal symbol** A symbol used in the rewriting rules of a grammar, which is not a terminal symbol.
- normal form** A method for choosing a unique member of an equivalence class; left-most (or right-most) derivations are a normal form for context-free derivations.
- null string** The string consisting of zero characters.
- nullable nonterminal** A nonterminal from which the null string can be derived.
- nullable rule** A grammar rule which can be used to derive the null string.
- object language** The language of the target machine; the output of the compiler is a program in this language.

**object program** A program produced as the output of the compiler.

**operator** A source language symbol used to specify an arithmetic, assignment, comparison, logical, or other operation involving one or two operands.

**optimization** The process of improving generated code in run time and/or space.

**p-code** A standard intermediate form developed at the University of California at San Diego.

**palindrome** A string which reads the same from left to right as it does from right to left.

**parse** A description of the structure of a valid string in a formal language, or to find such a description.

**parser** The syntax analysis phase of a compiler.

**parsing algorithm** An algorithm which solves the parsing problem for a particular class of grammars.

**parsing problem** Given a grammar and an input string, determine whether the string is in the language of the grammar and, if so, find its structure (as in a derivation tree, for example).

**pop** A pushdown machine operation used to remove a stack symbol from the top of the stack.

**postfix traversal** A tree-scanning algorithm in which the children of a node are visited, followed by the node itself; used to generate object code from a syntax tree.

**production** A rewriting rule in a grammar.

**programming language** A language used to specify a sequence of operations to be performed by a computer.

**push** A pushdown machine operation used to place a stack symbol on top of the stack.

**pushdown machine** A finite state machine, with an infinite last-in first-out stack; the top stack symbol, current state, and current input are used to determine the next state.

- pushdown translator** A pushdown machine with an output function, used to translate input strings into output strings.
- quasi-simple grammar** A simple grammar which permits rules rewritten as the null string, as long as the follow set is disjoint with the selection sets of other rules defining the same nonterminal.
- quasi-simple language** A language which can be described with a quasi-simple grammar.
- recursive descent** A top down parsing algorithm in which there is a procedure for each nonterminal symbol in the grammar.
- reduce/reduce conflict** In bottom up parsing, the failure of the algorithm to determine which of two or more reduce operations is to be performed in a particular stack and input configuration.
- reduce operation** The operation of replacing 0 or more symbols on the top of the parsing stack with a nonterminal grammar symbol, in a bottom up parsing algorithm.
- reduction in strength** The process of replacing a complex operation with an equivalent, but simpler, operation during optimization.
- reflexive transitive closure (of a relation)** The relation,  $R'$ , formed from a given relation,  $R$ , including all pairs in the given relation, all reflexive pairs ( $a R' a$ ), and all transitive pairs ( $a R' c$  if  $a R' b$  and  $b R' c$ ).
- register allocation** The process of assigning a purpose to a particular register, or binding a register to a source program variable or compiler variable, so that for a certain range or scope of instructions that register can be used to store no other data.
- register-displacement addressing** An address mode in which a complete memory address is formed by adding the contents of a CPU register to the value of the displacement instruction field.
- regular expression** An expression involving three operations on sets of strings – union, concatenation, and Kleene \* (also known as closure).
- relation** A set of ordered pairs.

- replace** An extended pushdown machine operation, equivalent to a pop operation, followed by zero or more push operations.
- reserved word** A key word which is not available to the programmer for use as an identifier.
- rewriting rule** The component of a grammar which specifies how a string of nonterminal and terminal symbols may be rewritten as another string of nonterminals and terminals. Also called a 'production'.
- right linear grammar** A grammar in which the left side of each rule is a single nonterminal and the right side of each rule is either a terminal or a terminal followed by a nonterminal (type 3).
- right linear language** A language which can be specified by a right linear grammar.
- right-most derivation** A derivation for a context-free grammar, in which the right-most nonterminal symbol is always the one rewritten.
- run time** The time at which an object program is executed, as opposed to compile time.
- SableCC** An object-oriented, Java-based compiler generator.
- scanner** The phase of the compiler which performs lexical analysis.
- selection set** The set of terminals which may be used to direct a top down parser to apply a particular grammar rule.
- semantic analysis** That portion of the compiler which generates intermediate code and which attempts to find non-syntactic errors by checking types and declarations of identifiers.
- semantics** The intent, or meaning, of an input string.
- sentential form** An intermediate form in a derivation which may contain nonterminal symbols.
- set** A collection of unique objects.
- shift operation** The operation of pushing an input symbol onto the parsing stack, and advancing to the next input symbol, in a bottom up parsing algorithm.

**shift reduce parser** A bottom up parsing algorithm which uses a sequence of shift and reduce operations to transform an acceptable input string to the starting nonterminal of a given grammar.

**shift/reduce conflict** In bottom up parsing, the failure of the algorithm to determine whether a shift or reduce operation is to be performed in a particular stack and input configuration.

**simple algebraic optimization** The elimination of instructions which add 0 to or multiply 1 by a number.

**simple grammar** A grammar in which the right side of every rule begins with a terminal symbol, and all rules defining the same nonterminal begin with a different terminal.

**simple language** A language which can be described with a simple grammar.

**single pass code generator** A code generator which keeps a fixup table for forward references, and thus needs to read the intermediate code string only once.

**single pass compiler** A compiler which scans the source program only once.

**source language** The language in which programs may be written and used as input to a compiler.

**source program** A program in the source language, intended as input to a compiler.

**state** A machine's status, or memory/register values. Also, in SableCC, the present status of the scanner.

**start condition** In the lex utility, a state which is entered by the scanner, resulting from a particular input; used to specify the left context for a pattern.

**starting nonterminal** The nonterminal in a grammar from which all derivations begin.

**stdin** In Unix or MSDOS, the standard input file, normally directed to the keyboard.

**stdout** In Unix or MSDOS, the standard output file, normally directed to the user's monitor.

**string** A list or sequence of characters from a given alphabet.

- string space** A memory buffer used to store string constants and possibly identifier names or key words.
- symbol table** A data structure used to store identifiers and possibly other lexical entities during compilation.
- syntax** The specification of correctly formed strings in a language, or the correctly formed programs of a programming language.
- syntax analysis** The phase of the compiler which checks for syntax errors in the source program, using, as input, tokens put out by the lexical phase and producing, as output, a stream of atoms or syntax trees.
- syntax directed translation** A translation in which a parser or syntax specification is used to specify output as well as syntax.
- syntax tree** A tree data structure showing the structure of a source program or statement, in which the leaves represent operands, and the internal nodes represent operations or control structures.
- synthesized attributes** Those attributes in an attributed grammar which receive values from lower nodes in the derivation tree.
- target machine** The machine for which the output of a compiler is intended.
- terminal symbol** A symbol in the input alphabet of a language specified by a grammar.
- token** The output of the lexical analyzer representing a single word in the source program.
- top down parsing** Finding the structure of a string in a way that produces or traverses the derivation tree from top to bottom.
- Translation class** An extension of the DepthFirstAdapter class generated by SableCC. It is used to implement actions in a translation grammar.
- translation grammar** A grammar which specifies output for some or all input strings.
- underlying grammar** The grammar resulting when all action symbols are removed from a translation grammar.

**unreachable code** Code, usually in an intermediate code string, which can never be executed.

**unrestricted grammar** A grammar in which there are no restrictions on the form of the rewriting rules (type 0).

**unrestricted language** A language which can be specified by an unrestricted grammar.

**white space** Blank, tab, or newline characters which appear as nothing on an output device.

**yacc** (Yet Another Compiler-Compiler) A parser generator utility in the Unix programming environment which uses a grammar to specify syntax.