Table 1: Definition of categories and choices for `grep`

#	**Category**	**Choice**
1	NormalChar - presence of any literal character	NormalAlNum - presence of any alphabetic or numerical literal (for instance “A”, “z”, or “5”)
		NormalPunct - presence of any punctuation character (such as “:”)
2	WordSymbol - presence of “word” or “non-word” metacharacters	YesWord - “\w” present
		NoWord - “\W” present
3	DigitSymbol - presence of “digit” or “non-digit” metacharacters	YesDigit - “\d” present
		NoDigit - “\D” present
4	SpaceSymbol - presence of any “whitespace” or “non-space” metacharacters	YesSpace - “\s” present
		NoSpace - “\S” present
5	NamedSymbol - presence of a symbol from a character group	ALPHA - presence of [:ALPHA:]
		UPPER - presence of [:UPPER:]
		LOWER - presence of [:LOWER:]
		DIGIT - presence of [:DIGIT:]
		XDIGIT - presence of [:XDIGIT:]
		SPACE - presence of [:SPACE:]
		PUNCT - presence of [:PUNCT:]
		ALNUM - presence of [:ALNUM:]
		PRINT - presence of [:PRINT:]
		GRAPH - presence of [:GRAPH:]
		CNTRL - presence of [:CNTRL:]
		BLANK - presence of [:BLANK:]
6	AnyChar - presence of the “.” metacharacter (matches any character)	Dot - dot (“.”) present

– continued on next page

Table 1 Definition of categories and choices for `grep` (continued)

#	**Category**	**Choice**
7	Range - presence of a pattern representing a character range	NumRange - number range present (for example "[1-7]")
		UpcaseRange - uppercase letter range present (for example "[C-G]")
		LowcaseRange - lowercase letter range present (such as "[s-w]")
8	Bracket - presence of patterns encompassed by [] or [^]	NormalBracket - "[]" pattern present
		CaretBracket - [^] pattern present
9	Iteration - presence of patterns that contain iterator symbols	Qmark - presence of the question mark metacharacter ("?"), which matches 0 or 1 iteration
		Star - presence of the star metacharacter ("*"), matching zero or more iterations
		Plus - presence of the plus metacharacter("+"), matching one or more iterations
		Repminmax - presence of min-max repetition form: for example, "{2, 3}" matches lines containing "aa" or "aaa"
10	Parentheses - used to group patterns for repetition, also "backreferencing"	NormParen - presence of a pattern surrounded by parentheses
		Backref - presence of a pattern with normal parentheses and a back reference
11	Line - presence of special characters relating to line boundaries	BegLine - presence of ("^") (matches beginning of line)
		EndLine - presence of ("$") (matches end of line)
		BegEndLine - presence of ("^"..."$") (matches beginning and end of line)
12	Word - presence of sequences that match word beginnings or ends	BegWord - presence of a ("\<") metacharacter (matches word beginning)
		EndWord - presence of a ("\>") metacharacter (matches word end)
		BegEndWord - presence of a ("\<" ... "\>") pattern (matches word end)

– continued on next page

Table 1 Definition of categories and choices for `grep` (continued)

#	**Category**	**Choice**
13	Edge - presence of sequences that match word boundaries	YesEdgeBeg - presence of a "\b" metacharacter (sequence must lie on a word edge at the beginning - for example "\babc" matches "abcde" but not "xabc")
		YesEdgeEnd - presence of the "\b" metacharacter (sequence must lie on a word edge at the end - for example "abc\b" matches "12abc" but not "abc12")
		YesEdgeBegEnd - presence of "\b" ... "\b" pattern - sequence must lie on a word edge at the beginning and the end (for example "\babc\b" matches "abc" only)
		NoEdgeBeg - presence of "\B" metacharacter - sequence must not lie on a word edge at the beginning (for example, "\Babc" matches "xabce" but not "abcde").
		NoEdgeEnd - presence of "\B" metacharacter - sequence must not lie on a word edge at the end (for example, "abc\B" matches "xabce" but not "xabc").
		NoEdgeBegEnd - presence of "\B" ... "\B" - sequence must not lie on a word edge at the beginning and the end (for example, "\Babc\B" matches "xabce" but not "abcdeabc").
14	Combine - combining multiple patterns	Concatenation - presence of a sequence of tokens (which must all appear in sequence in the text to match - for example, "ab" matches "abx" or "cab" but not "aaa", "axb", or "bax")
		Alternative - presence of two tokens separated by the "|" metacharacter (presence of either token will result in a match - for instance "a|b" matches "ast" or byz")