Change Log — GPLEX Scanner Generator

(*Version 1.2.1 July 2013*

John Gough QUT

June 29, 2013

Documentation Map

This paper is the change log for the *gplex* scanner generator.
The complete documentation for *gplex* consists of the files —

- Gplex.pdf – This file merges the information that prior to version 1.1.4 was in three separate files.
- Gplex-Changelog.pdf – change log for *gplex*, (this file)

1 Change Log

This section tracks the updates and bug fixes from the intial release candidate version RC0 of October 2006. RC0 was the first release candidate bootstrapped using a self-generated scanner and *gppg*-generated parser. Versions prior to RC0 used a handwritten scanner.

Changes in version 1.2.1 (gplex, July 2013)

- Fixed a bug in Utils.cs, which caused an index out of bounds exception to be thrown if a nul character was the last character in a pattern.
- Fixed a bug in AAST.cs, which caused a method not found exception to be thrown if a user-defined character predicate function was not the first or only method defined in its class.
- The `[:`*Predicate*`:]` mechanism for defining character class constructors has extended semantics. The enclosed name may either be the name of a character predicate or the name of a previously defined named regex, provided the named regex defines a character class.
- New character set operators have been implemented `{+}`, `{-}`, `{*}`, denoting character class set union, difference and intersection respectively. This assists in the compact definition of large Unicode character classes, and are compatible with *Flex* input syntax.

- The scanner runtime code that performs scanner backup has been re-implemented. The new version adds less runtime overhead to the execution.

Changes in version 1.2.0 (gplex, June 2012)

- Error messages are now in *MSBuild*-friendly format to interwork better with *Visual Studio*. This is the default behavior, with a new option allowing legacy messages to the console.
- This version changes the default file encoding to Unicode, with a fallback to raw (uninterpreted bytes) if there is no *BOM* prefix.
- New optional code may be used to allow *gppg* parsers to push back wrapped symbols to a *gplex* scanner.
- Generated scanners suppress several *CSC* warning messages.

Changes in version 1.1.6 (gplex, May 2011)

- Version 1.1.6 corrects an error in the handling of singleton character classes, like [z] in the presence of the `/caseInsensitive` option.
- The NFSA next state dictionary now uses 32-bits of state ordinal number, and 32 bits of symbol ordinal. *gplex* now works correctly for specifications that imply NFSA with more than 64k states.
- The distribution now comes with a contemporary copy of *gppg* executable, as a convenience. The *gppg* and *gplex* executables are both required to build *gplex* from its sources.

Changes in version 1.1.5 (gplex, November 2010)

- Version 1.1.5 corrects an error in the handling of literal strings in regular expressions.
- Code sections of the specification now allow the use of *C#* verbatim literal strings.
- The detection of errors in string literals is improved with specific error messages if a literal string in unterminated in a code section of the specification.

Changes in version 1.1.4 (gplex, May 2010)

- Version 1.1.4 corrects an error in the implementation of LineBuffers. The code is now protected against some cases where the setter of buffer position indexes out of bounds in the line-array.
- An error is corrected in the handling of surrogate character pairs. Scanner.GetCode() and Scanner.yyleng are affected.

Changes in version 1.1.3 (gplex, April 2010)

- Version 1.1.3 fixes an error in the implementation of *yyleng*. In the previous version if *yyleng* was called on a token that spanned a line-break the length of an immediately following token would be out by one.

Changes in version 1.1.2 (gplex, December 2009)

- Version 1.1.2 implements a new option that allows the generation of case insensitive scanners.
- The preferred source for future updates to *gplex* has moved to Code Plex.

Changes in version 1.1.1 (gplex, March 2009)

- Version 1.1.0 contained a bug in state backtracking on some input specifications. Version 1.1.1 fixed this.

Changes in version 1.1.0 (gplex, March 2009)

- File buffering in generated scanners has been completely re-implemented. Performance on large input files is much improved. File position semantics are now the same for all input buffer types. A new option [*no*]*PersistBuffers* allows buffer space to be reclaimed when that is appropriate.
- Unicode-mode scanners may now use any of the encodings supported by the *System.Globalization* namespace of the host machine.
- Scanners generated by *gplex* are now *FxCop*-friendly. However, some internal scanner members have been renamed, in what is a *potentially breaking change for some existing applications*.
- Generated scanner code is restructured to make it easier to avoid name-clashes in multi-scanner applications. In particular new keywords allow the visibility of the generated types to be either public or internal, and the *ScanBuff, Scanner* and *Tokens* types may all have their default names overridden.
- The frame file and the buffer code are embedded resources in the *gplex* executable, leading to simpler deployment.
- The *QUT.Gplex* namespace exports an interface *ICharTestFactory* which is implemented by generation-time *gplex* plugins that define user character predicates.
- Detailed code changes have been made to the *NextState* functions, to avoid arithmetic overflows when generated scanners are embedded in projects that build with */checked*.
- A new marker *%visibility* in *LEX* files allows the accessibility of the scanner class to be controlled.
- All option processing from the command line or from embedded *%option* markers is now case insensitive.
- A bug in the implementation of character class predicates (*%charClassPredicate* marker) when used with the */noMinimize* option is now fixed.
- The *buffer.Peek* method returned a surrogate character if the next code point to be read was above the 64k boundary. *Scanner.Peek* now always returns a valid unicode value or an *EOF* marker.
- String and Line buffers now correctly handle unicode surrogate pairs.

Changes in version 1.0.2 (gplex, January 2009)

- User-defined character class predicate functions are now supported. The predicates may be specified in any *.NET* language that supports managed *PE*-files.
- Right-anchored patterns in unicode scanners now recognize any of the unicode line-end characters.

Changes in version 1.0.1 (gplex, November 2008)

- A separate documentation file "`Gplex-Input.tex`" collects the details of the *gplex*-input language in one place.
- Previous versions prevented *C#* code from including pragmas, by notifying an error when finding a `#`. This is now permitted.
- Some compression options incorrectly dealt with the first few codepoints in the second character plane, that is, the first few characters that require surrogate pairs for their representation in strings. This is now fixed.
- Unicode scanners no longer skip "private use areas" in the unicode character set when compiling character classes from character predicates. (This is in anticipation of the introduction of user-specified predicates in the next refresh.)

Changes in version 1.0.0 (gplex, November 2008)

- New options for unicode scanners allow the user to specify the fallback code page to use if an input file does not have a valid *UTF* prefix.
- New facilities for unicode scanners allow the host application to set the fallback code page at scanner runtime.
- New facilities for unicode scanners allow the scanner to scan the input file to determine the probable encoding used.
- New options for byte-mode scanners allow the user to specify the code page mapping that is used to define the meaning of character set predicates.
- A separate documentation file "`Gplex-Unicode.pdf`" collects the details of the unicode-specific features in one place.
- The change log information has been separated out into this separate documentation file "`Gplex-Changelog.pdf`".
- An error in the code of *GetString* for unicode unicode buffer implementations has been fixed. Under some circumstances *yytext* would report an extra character with utf-8 files.
- Errors in the handling of the *utf8default* option have been corrected (but this option is now deprecated).

Changes in version 0.9.9 (gplex, October 2008)

- Multiple input sources are now allowed, using user-defined overrides of the *yywrap* predicate. A new buffer context type has been defined and context handling methods added to the scanner class. Overloads of *SetSource* can create any of the possible buffer types.
- Start condition scopes have been introduced, so that a group of rules may use the same start condition list. These scopes may be nested.
- *gplex* now checks on the version marker of the frame file. It is a fatal error to attempt to use an incompatible frame file.
- *C#*-style single line comments may be used anywhere in the lex file, and are treated as white space.
- A number of minor corrections and improvements of error reporting have been incorporated.

Changes in version 0.9.0 (gplex, August 2008)

- Surrogate pairs are now handled for string and UTF-16 input, and all automata deal with code-points rather than character values.
- New code-point to equivalence class map compression algorithms have been added to handle very large alphabet cardinality. Choice of compression mechanism(s) for maps in the */unicode* case is now adaptive at scanner generation time.
- A hard limit on state-space cardinality has been relaxed by changing the symbol table implementation.
- The implementation of the *NextState* function in the generated parsers has been improved.
- Unreachable rules and patterns that consume no input are now detected.
- Character predicates can be generated for named character classes.
- Scanner class is now "partial" in frame file.
- Better diagnostics, including shortest string reaching each *FSA* state.
- New flags */nofiles, /utf8default, /squeeze*.
- The semantics of the setter for *YY_START* now exactly match *BEGIN*. Both *currentScOrd and currentState* are updated.

Changes in version 0.6.2 (gplex, November 2007)

- New buffer implementation for lists of strings contributed by Nigel Horspool.
- The scanner class is marked `partial`, so that much of the user code of the scanner can be separated out into a separate file.
- A number of small bugs in the unicode support have been fixed.

Changes in version 0.6.1 (gplex, August 2007)

* Dead code warnings in semantic action dispatch of the automaton have been suppressed.
* Sharing of redundant rows is implemented for uncompressed next-state tables
* Treatment of un-escaped dash characters as the first or last character in a character class definition follows the *LEX* semantics, rather than being an error.
* Incorrect behavior with empty strings in semantic actions has been fixed.
* Bug in V0.6.0 text buffer code lost the last character of *yytext* if an unexpected end of file was encountered. Fixed.

Changes in version 0.6.0 (gplex, July 2007)

* Table compression using character equivalence classes is invoked by the */classes* option.
* New option */unicode* allows unicode scanners to be generated. *UTF-8*, and both byte-orders of *UTF-16* are supported.
* Option */minimize* is now on by default.
* The frame file has numerous small changes - the encoding of *EOF* has changed, the abstract buffer class has a property *ReadPos* for the file position of the current character (not the same as *Pos*–1 in the unicode case), and new buffer classes have been added for encoded text files.
* A bug in the parsing of regular expressions, if a character escape appears immediately after a range operator '–', is fixed.

Changes in version 0.5.1 (gplex, March 2007)

* Rules explicitly marked with the *INITIAL* start condition are **not** added to inclusive start states. This aligns *gplex* with the *Flex* semantics.
* The */babel* option produces incremental scanners compatible with the *Managed Babel* framework.
* The "frame" file, *gplexx.frame*, has been changed to allow for the */babel* option.
* To use a *gplex* scanner and a *gppg* parser with *Managed Babel* both tools must be invoked with the */babel* option.
* When working with string buffer input, *gplex* returned an *EOL* character if the scanner attempted to read past the end of the string. This behavior is now restricted to the */babel* option, where it is necessary for compatibility with the managed package framework. In previous versions attempts to fetch *yytext* for this virtual token threw an exception. The code is now guarded.

Changes in version 0.5.0 (gplex-RTM, February 2007)

- Parser is now generated by version 1.0.2.⋆ of *gppg*, using a modified version of *IScanner*. *Existing* gplex *scanners will require recompilation to work with parsers produced by the new version of* gppg.
- Version strings are now generated from an assembly attribute, and accessed via reflection.
- New facilities allow output to be sent to the standard output stream. This may involve redirecting verbose progress output to standard error.

Changes in version 0.4.2 (gplex-RC2, January 2007)

- Semantics of inclusive start states were incorrect. Now fixed.
- There was a bug in the analysis of right context patterns involving concatentation, leading to some cases being incorrectly analysed as having fixed length. Fixed.

Changes in version 0.4.1 (gplex-RC1, November 2006)

- Program failed on empty user code containing not even a comment. Fixed.
- Reflection is used in *gplexx.frame* to determine if the *maxParseToken* value has been defined in the parser specification.
- Must allocate a default error handler if *OpenSource* fails.
- Table compression algorithm failed for some corner cases where the longest run of equal next-state entries wraps around character 255.
- Within the constructed scanners token start and end position is now represented by private variables *tokPos, tokEPos, tokLin, tokELin, tokCol, tokECol* to allow single tokens to correspond to multi-line *LexLocation* values. The previous *tokLen* has disappeared, with *yyleng* now computed from *tokEPos* and *tokPos*. *This change may break user-defined YYLTYPE computations*.