Creating `pcb-tools` Haskell library

parsing Gerber RS-274X, Excellon and other Haskell tales

Bc. Adam Lučanský

February 2, 2018

Abstract

`pcb-tools` library sets goal to emit initial effort and to create fully featured PCB design files parser/pre-processor, as well as CAM tooling. Project is currently in development.

1 Introduction

In the first phase of the project, modules parsing and interpreting `Gerber RS-274X` [1] (layer description) format as well as `Excellon` (drilling) were created. Parsing is implemented by the Attoparsec LL(∞) parsec [2] library. Interpretation is performed in State monad.

This shall be the base point for the further work.

Second phase is basic tooling based on the library, such as utilities for pre-processing of the drawings/drillings, e.g.:

- **viewgerber** - programs for visualization of the designs
- **mergedrill** - drill file pre-process (rounding drill diameters to available drills)
- **gcoder** - CAM tooling, outlining path of the polygons as defined in the design (similar to `pcb2gcode`)

In the current implementation, `mergedrill` and `viewgerber` work as a proof-of-concept.

Following sections introduce grammar of parsed/interpreted languages with examples and structures emitted by the interpreter.

[1] `https://www.ucamco.com/files/downloads/file/81/the_gerber_file_format_specification.pdf`

[2] `https://wiki.haskell.org/Parsec`

2 Gerber RS-274X

Gerber RS-274X is a structured human-readable ASCII format describing vector graphics. Usecase for this format can be found in PCB manufacturing processes.

Listing 1: Example Gerber source file

```
%ADD10C,1.321*%
%ADD11OC8,1.321*%
%ADD12C,1.524*%
%ADD13C,1.270*%
D10*
X42164Y05283D02*
X42164Y06604D01*
X44704Y06604D02*
X44704Y05283D01*
X47244Y05283D02*
X47244Y06604D01*
X47244Y14224D02*
```

2.1 Grammar

Listing 2: Grammar rules of implemented Gerber parser in EBNF

```
<S> ::=                         <gerberCommands>
<gerberCommands> ::=            {"%" <extended> "%" | <standard> "*" | <eof> }
<char> ::=                      any ASCII char
<eof> ::=                       "M02*" {<anyChar>}
<anyCharExceptAsterisk> ::=     [ASCII] - ["*"]
<allowedChars> ::=              A-Za-z0-9,.#@$\n
<optionalNewLines> ::=          {"\n" | "\r"}
<takeTillAsteriskMany> ::=      (<anyCharExceptAsterisk>)* "*"
<takeTillAsteriskMany1> ::=     <anyCharExceptAsterisk> <takeTillAsteriskMany>
<singleBlockWrap> ::=           <singleBlockExtendedCommand> "*"
<multiBlockWrap> ::=            <multiBlockExtendedCommand>
<standard> ::=                  <comment>
<standard> ::=                  <toolChange>
<standard> ::=                  <operation>
<standard> ::=                  <quadrantMode>
<standard> ::=                  <interpolationMode>
<standard> ::=                  <regionBoundary>
<standard> ::=                  <unknownStandard>
<extended> ::=                  <singleBlockCommand> "*" | <multiBlockCommand>
<singleBlockCommand> ::=        <formatSpecification>
<singleBlockCommand> ::=        <setUnits>
<singleBlockCommand> ::=        <addAperture>
```

```
<singleBlockCommand> ::=      <unknownExtended>
<multiBlockCommands> ::=      <apertureMacro>
<quadrantMode> ::=            "G74" | "G75"
<interpolationMode> ::=       "G01" | "G02" | "G03"
<regionBoundary> ::=          "G36" | "G37"
<comment> ::=                 "G04␣" <commentChars> "*"
<commentChars> ::=            [ASCII] - ["*"]
<toolChange> ::=              "D" integer {integer}
<action> ::=                  "D01" | "D02" | "D03"
<coord> ::=                   ["X" integer] ["Y" integer] ["I" integer] ["J" integer]
<operation> ::=               <coord> <action>
<unknownStandard> ::=         (<anyCharExceptAsterisk>)* "*"
<unknownExtended> ::=         (<anyCharExceptAsterisk>)* "*"
<formtSpecification> ::=      "FSLA" "X" digit digit "Y" digit digit
<setUnits> ::=                "MO" ("MM" | "IN")
<addAperture> ::=             "ADD" integer ([A-Z0-9]+) "," ({scientific "X"} | scientific)
<apertureMacro> ::=           "AM" <allowedChars>* "*" <apertures>
<apertures> ::=               <singleAperture> {<singleAperture>}
<singleAperture> ::=          <allowedChars>* "*" <optionalNewLines>
```

2.2 AST

Implemented parser outputs AST in type `[Command]` where:

Listing 3: Structure of single Gerber command

```
data Command =
  -- STANDARD COMMANDS
  -- G04
  Comment ByteString |
  -- Dxx, xx >= 10
  ToolChange Integer |
  Operation Coord Action |
  AddAperture Integer ByteString [Scientific] |
  DefineAperture ByteString [ByteString] |
  EndOfFile |
  -- EXTENDED COMMANDS
  -- FSLAX
  FormatStatement FormatSpecification |
  -- MO
  SetUnits Unit |
  SetQuadrantMode QuadrantMode |
  -- G01/G02/G03
  SetInterpolationMode InterpolationMode |
  -- G36/G37
  SetRegionBoundary RegionBoundary |
  Deprecated DeprecatedType |
  SetOffset Integer Integer | -- Deprecated
  UnknownExtended ByteString |
  UnknownStandard ByteString
```

2.3 Interpreted output

Interpreter implements the state machine processing stream of commands. Final state represents output.

Listing 4: Gerber interpreter structure, acting as a result as well

```
data InterpreterState = InterpreterState
  { _formatSpecification :: Maybe FormatSpecification
  , _coordinateUnit :: Maybe Unit
  , _currentCoord :: Coord
  , _currentAperture :: ApertureID
  , _interpolationMode :: Maybe InterpolationMode
  , _quadrantMode :: Maybe QuadrantMode
  -- TODO: polarity
  -- TODO: LM, LR, LS
  , _apertures :: Apertures
  , _apertureTemplates :: ApertureTemplates

  , _draws :: [(ApertureParams, ApertureTemplate, Located (Trail V2 Double))]
  , _flashes :: [(ApertureParams, ApertureTemplate, Coord)]

  , _commandsParsed :: Integer
  , _unknownCommands :: Integer
  , _deprecatedCommands :: Integer}
```

3 Excellon

Excellon is a language used for defining drilling and slotting jobs for CNC machines. Although Excellon has no unified official specification, syntax can be derived from outputs of CAM software

(Eagle, KiCAD, Altium. . .).

3.1 Grammar

Listing 5: Grammar rules of implemented Excellon parser in EBNF

```
<S> ::=                     <excellonCommands>
<excellonCommands> ::=      <header> <body>
<header> ::=                "%" {<headerCommand> <newlines>} "%"
<body> ::=                  {<bodyCommand> <newlines>}
<commandM> ::=              "M" integer
<genericCommand> ::=        <commandM>
<headerCommand> ::=         <genericCommand> | <addDrill>
<addDrill> ::=              "T" integer "C" scientific
<bodyCommand> ::=           <genericCommand> | <setDrill> | <drillAt>
<setDrill> ::=              "T" integer
<drillAt> ::=               "X" integer "Y" integer
<newLines> ::=              {"\n" | "\r"}
```

3.2 AST

Output from the parser and basically AST is `[ExcellonCommand]`, and is as follows:

Listing 6: Structure describing Excellon command (not tied to any context)

```
data ExcellonCommand =
     -- Mxx command located in Body section
     M Integer |

     -- TxxCyy command in header (x - Tool identifier, y - diameter)
     AddDrill ToolIdentifier Diameter |

     -- Sets current drill (T01, T02, T3)
     -- T0 means no drill, usually at the end of program
     SetDrill Integer |

     -- Marks the drill position
     DrillAt {x :: Integer, y :: Integer}
```

3.3 Interpreted output

Due to the nature of the AST, invalid sequence of commands can be represented, therefore interpretation step is needed, resulting following structures:

Listing 7: InterpreterState, as well as result

```
type Diameter = Double
type ToolIdentifier = Integer
data Drill = Drill { diameter :: Diameter }

data DrillJob = DrillJob
  { drillUnit         :: Unit -- MM or IN
  , drillsDefinition :: Map ToolIdentifier Drill
  , drillings         :: [Located ToolIdentifier]
  , lastUsedDrill     :: ToolIdentifier
  }
```

4 Graphical output

Library `diagrams` [3] is used in order to render trails drawn by the Gerber interpreter. Proof-of-concept has been made to satisfy the critical path for viewing Gerber files, although not yet fully complaint with specification. Up to this point, further iterations shall be easier.

[3] `https://archives.haskell.org/projects.haskell.org/diagrams/`