Creating `pcb-tools` Haskell library

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

Bc. Adam Lučanský

February 1, 2018

Abstract

This documents briefly explains grammar of the parsed languages, as well as the Abstract-Syntax-Tree emitted by the parser.

1 Introduction

This project sets goal to initiate effort to create Haskell library supporting PCB manufacturing processes (up to certain use-cases) and sub-programs for specific tasks related to manufacturing/milling.

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.

Following sections introduce grammar of parsed/interpreted languages with examples.

2 Gerber RS-274X

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

[1]`https://www.ucamco.com/files/downloads/file/81/the_gerber_file_format_specification.pdf`
[2]`https://wiki.haskell.org/Parsec`

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

3 Excellon

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

3.1 Grammar

3.2 AST

4 Graphical output

Library `diagrams` [3] is used in order to render trails drawn by the Gerber interpreter.

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