Grammars and Parsers for Validating Binary File Formats

William Underwood and Sandra Laib

Information and Communications Laboratory, GTRI, Atlanta, GA

Introduction

Automated tools are required for identifying and validating the formats of the huge number of files ingested into digital data and record archives.

Validation is required because

- If a file has been damaged, it may be possible to obtain an undamaged copy, or repair the file.
- File might need to comply with a standard format, e.g., PDF/A.

JHOVE (JSTOR Harvard Validation Environment) is an example of a set of programs for validating file formats

- JHOVE supports validation of the following file formats: AIFF, ASCII, GIF, HTML, JPEG, JPEG 2000, PDF, TIFF, UTF-8, WAVE, and XML
- JHOVE2 supports the validation of the following additional formats: ICC, SGML, Shapefile and ZIP

Each validation program is manually created from the specification for the file format.

Specification of File Formats

File (or record) Layouts for Binary Formats

Offset	Description
0	"RIFF" 4-byte tag
4	size of data chunk starting at offset 8
8	"WEBP" the form-type signature
12	"VP8 " 4-bytes tag, describing the raw video format used
16	size of the raw VP8 image data chunk, starting at offset 20
20	the VP8 image data

C Data Structures for Binary Formats

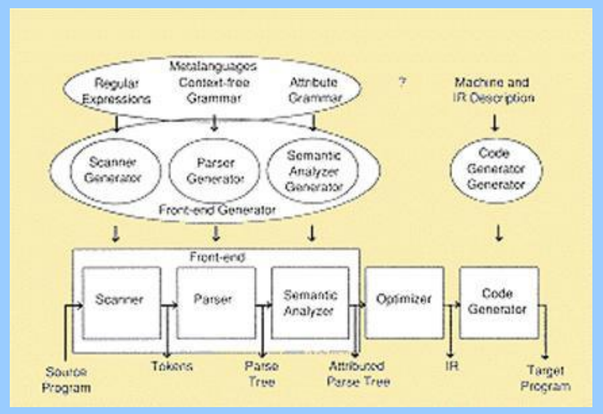
```
typedef struct {
        UWORD w, h;
        WORD  x, y;
        UBYTE nPlanes;
        Masking masking;
        Compression compression;
        UBYTE pad1;
        UWORD transparentColor;
        UBYTE xAspect, yAspect;
        WORD  pageWidth, pageHeight;
        } BitMapHeader;
```

Context-free Grammars for Textual Formats

Programming language syntax is usually defined using a combination of regular expressions (for lexical structure) and Backus–Naur Form rules (for grammatical structure)..

```
expression ::= atom | list
atom ::= number | symbol
number ::= [+-]?['0'-'9']+
symbol ::= ['A'-'Z''a'-'z'].*
list ::= '(' expression* ')'
```

Compiler-Compiler Technology



Given the specification of a programming language in terms of regular expressions and a context-free grammar, a compiler-compiler creates a program called a compiler that will take as input a source program and create an executable program in the target language.

One component of this technology is called a Parser Generator. It takes as input a context-free grammar for the language and generates a parser (syntax checker, validator) for the source programs.

Research Questions

Is it possible to extend the concept of context-free grammars from textual languages to binary file formats?

Is it possible to specify binary file formats using these extended context-free binary file grammars?

Is it possible to develop a parser generator that takes a binary file grammar for a binary file format and generates a parser that can validate the file format?

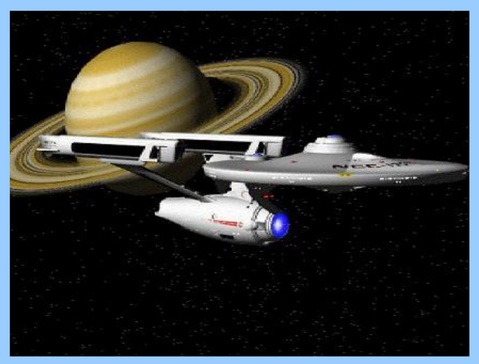
Family of Chunk-based Binary File Formats

Electronic Arts & Commodore-Amiga developed the Interchange File Format (IFF) in about 1985. It was the first chunk-based binary file format. A chunk consists of a chunk-id, a chunk-size and chunk-data. Chunk data can contain image, audio or text data. It can also contain sub-chunks and metadata. Sub-chunks can contain sub-sub-chunks

Examples of Chunk-based File Formats

(RIFF) – WAV, AVI, ANI, RMID, DIB, Webp
JPEG
Advanced Systems Format – WMA, WMV
Binary Interchange File Format (Microsoft Excel)
CorelDRAW Vector Graphics-cdw
Apple QuickTime - mo, qt
Portable Network Graphics -- PNG, MNG, JNG

Image in ILBM IFF File Format



Bytes 0-511 of the ILBM IFF File

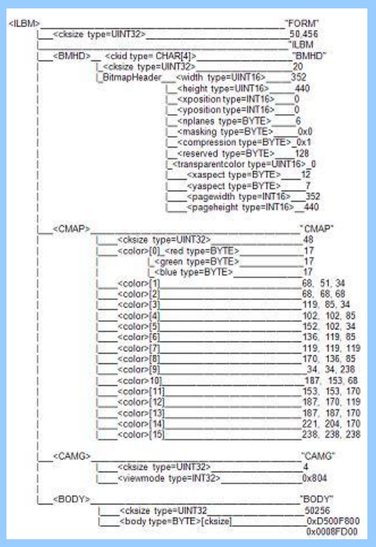
```
0000:   46 4F 52 4D 00 00 C5 18 - 49 4C 42 4D 42 4D 48 44   FORM....ILBMBMHD
0010:   00 00 00 14 01 60 01 B8 - 00 00 00 00 06 00 01 80   .....`..........
0020:   00 00 0C 07 01 60 01 B8 - 41 4E 4E 4F 00 00 00 48   .....`..ANNO...H
0030:   24 56 45 52 30 20 57 72 - 69 74 74 65 6E 20 62 79   $VER: Written by
0040:   20 41 53 44 47 27 73 20 - 41 72 74 20 44 65 70 61    ASDG's Art Depa
0050:   72 74 6D 65 6E 74 20 50 - 72 6F 66 65 73 73 69 6F   rtment Professio
0060:   6E 61 6C 20 49 46 46 33 - 2E 30 2E 34 20 28 31 39   nal IFF3.0.4 (19
0070:   2E 30 32 2E 39 35 29 00 - 43 4D 41 50 00 00 00 30   .02.95).CMAP...0
0080:   11 11 11 44 33 22 44 44 - 44 77 55 22 66 66 55 99   ...D3"DDDwU"ffU.
0090:   66 22 88 77 55 77 77 77 - AA 88 55 22 22 EE BB 99   f".wUwwww..U""...
00A0:   44 99 99 AA BB AA 77 BB - BB AA DD CC AA EE EE EE   D.....w.........
00B0:   43 41 4D 47 00 00 00 04 - 00 00 08 04 44 50 49 20   CAMG........DPI
00C0:   00 00 00 04 00 96 01 01 - 42 4F 44 59 00 00 C4 50   ........BODY...P
00D0:   D5 00 F8 00 00 08 FD 00 - 00 20 FE 00 00 80 FE 00   ......... ......
00E0:   03 81 00 80 10 FA 00 02 - 01 00 08 F9 00 D5 00 D5   ................
00F0:   00 00 BF F9 FF 00 F2 FD - FF 00 CB FE FF 00 2F FE   ............../.
0100:   FF 03 2E 5F 2F E5 FA FF - 02 FE 5F F2 F9 FF 00 DF   .../......._....
0110:   F9 FF 01 F3 7F FE FF 00 - CD FE FF 00 37 FE FF 03   ............7...
0120:   36 6F 37 E6 FA FF 03 FE - 6F F3 7F FA FF D5 00 FC   6o7.....o.......
0130:   00 00 20 FE 00 00 02 F9 - 00 02 20 00 01 FA 00 00   .. ....... .....
0140:   04 FC 00 00 20 FB 00 02 - 20 00 00 D5 00 D5 00 00   .... ... .......
0150:   BF FD FF 00 CB FE FF 01 - FC BF FA FF 03 CB FF FE   ................
0160:   5F FB FF 01 F9 7F FD FF - 00 CB FB FF 02 CB FF FF   _...............
0170:   00 DF FD FF 00 CD FE FF - 01 FC DF FA FF 03 CD FF   ................
0180:   FE 6F FB FF 01 F9 BF FD - FF 00 CD FB FF 02 CD FF   .o..............
0190:   FF D8 00 02 20 00 00 F8 - 00 00 08 FC 00 00 80 E5   .... ...........
01A0:   00 D5 00 D5 00 00 BF F9 - FF 00 F2 FC FF 00 2F E8   ............../.
01B0:   FF 02 CB FF FF 00 DF F9 - FF 01 F3 7F FD FF 00 37   ...............7
01C0:   E8 FF 02 CD FF FF D5 00 - D5 00 D5 00 D5 00 00 BF   ................
01D0:   D6 FF 00 DF D6 FF D5 00 - 04 00 40 00 00 08 FD 00   ..........@.....
01E0:   06 10 00 00 10 00 00 04 - F5 00 00 10 F8 00 00 04   ................
01F0:   FC 00 D5 00 D5 00 04 BF - 97 FF FF F2 FD FF 07 E5   ................
```

Grammar for ILBM Format

```
<ILBM> → "FORM" <cksize TYPE=UINT32> "ILBM" <BMHD> <CMAP> <CAMG> <BODY>
{ILBM.cksize = cksize.value}

<BMHD> → "BMHD" <cksize TYPE=uint32> <BitmapHeader>
<BitMapheader> →   <width type=UINT32>
                   <height type=UINT16>
                   <xposition type=INT16>
                   <yposition type=INT16>
                   <nplanes type=BYTE>
                   <masking type=BYTE>
                   <compression type=BYTE>
                   <reserved type=BYTE>
                   <transparentcolor type=UINT16>
                   <xaspect type=BYTE>
                   <yaspect type=BYTE>
                   <pagewidth type=INT16>
                   <pageheight type=INT16>
<CMAP> →"CMAP" <cksize type=UINT32> <color>[n] {n=cksize.val/3}
<CAMG> → "CAMG" <cksizetype=UINT32> <viewmode type=INT32>
<BODY> → "BODY" <cksize type=UINT32> <data type=BYTE>[cksize]
<color> → <red type=BYTE> <green type= BYTE> <blue type=BYTE>
```

Parse Tree for ILBM Binary File



Summary

It is possible to extend context-free grammars for textual languages to the specification of chunk-based binary file formats.

ANTLR, a parser generator for LL(k) grammars, has been successfully used to generate parsers for two chunk-based file formats.

Next Step: Binary file grammars for directory-based binary file formats, e.g., TIFF, OLE, OASIS Open Document, and Microsoft Open Office files.

Additional Information

W. Underwood and S. Laib. Attribute Grammars for Validating Chunk-based Binary File Formats. ICL/ITDSD Working Paper 11-03, GTRI, Atlanta, Georgia, July 2011 http://perpos.gtri.gatech.edu

Acknowledgement

This research was sponsored by the Army Research Laboratory (ARL) and the Applied Research Division of the National Archives and Records Administration (NARA) and was accomplished under Cooperative Agreement Number W911NF-10-2-0030.