Vol.2013-CE-122 No.30
Vol.2013-CLE-11 No.30
2013/12/15

プレゼンテーション用の TEX-PDF ファイル作成支援ソフトウェアの提案

黄 掣[†1,a)]

概要：Adobe 社が開発した PDF ファイルは情報の完全性, 高い信頼性, 優れた拡張性と各種システムに対応できるといった特徴で広く使われている. 一方, 数式や化学式などを多用するような場合には, 数学的な表現に優れる, 再利用・データベース化が容易と様々なシステムに対応できるなどといった特徴で TEX がよく使われている. TEX ソースをそのまま利用して, プレゼンテーション資料にするのがとても便利で, 定型のものを作るのに Prosper, Beamer などのライブラリが開発されている. しかし, これらの使い方は専門以外の人にとって難しいと思われる. そこで, わずかな命令を挟むだけで, 簡単にプレゼンテーション PDF が作成できるといった自由度が高い PPower4 に注目し, 今回その特徴を保持した互換性のあるプレゼンテーション用の TEX-PDF ファイル作成支援ソフトウェアを提案する.

キーワード：プレゼンテーション作成支援, TEX, PDF ファイル

Proposal of TEX-PDF file creation software support for the presentation

HUANG CHE[†1,a)]

Abstract: PDF files developed by Adobe company are widely used because of its good expansibility, integrity of information and high reliability. On the other hand, in using many Mathematical or Chemical formulae, the TEX system is often used because of excellent features of TEX, for instance, to express fine mathematical formulae or to treat the resources or databases easily. By the TEX system, we can easily build presentation PDF files, in fact, the libraries such as Prosper or Beamer have been used for stereotyped presentation PDF files. But it seems that it is difficult to learn these systems in general. So, we focus on PPower4 library which is a simple flexible system to build TEX presentation PDF files. In this paper, we propose a compatible TEX-PDF system to PPower4.

Keywords: Support for Presentation, TEX, PDF file

1. はじめに

数式交じりのプレゼンテーション資料が必要な場合はよくあるが, 多彩な数式表現は TEX に勝るものはないだろう. その点では, 教育現場でも需要は多いと思われる. このように, PDF ファイルをプレゼンテーション用に仕上げるライブラリとして, Prosper, Beamer などが LATEX で開発され, 数学系の研究会のプレゼンではほとんど, このような TEX で作成された PDF ファイルが使われている. これらのライブラリは, 定型のものを作るのには非常に威力を発揮するので, 人気が高い所以でもあるが, 逆に TEX の言語使用に加えて, 新たに専用の言語知識がいるために, 専門で使う人以外は敷居が高いだろう.

そこで, 開発は止まっているが PPower4 というライブラリに注目した. 基本的に必要な命令を（dvipdfm などの PDF 作成により）コメントとして PDF に埋め込み, それを後処理することで, 上記と同等のプレゼンテーション用 PDF を作成する仕組みのものである. したがって, 通常の

†1 現在, 大阪教育大学教育学研究科総合基礎科学専攻
Presently with Pure and Applied Sciences Graduate School of Education Osaka Kyoiku University
a) j139610@ex.osaka-kyoiku.ac.jp

TEX のファイル自身も, このライブラリで動くし, さまざまな \pause 命令を挟むとページ分割して順次表示するなど, わずかな命令を挟むだけで, 簡単にプレゼンテーション PDF が作成できる. それ以上に, 例えば LATEX 線画用の TPIC などもそのままサポートされているので, 簡単な線画およびそれに伴うアニメーションが作成できる. さらに埋め込みの命令を拡張すれば, 容易にバージョンアップも可能である. 以上のように自由度が高く, このままにするのはもったいない題材なので, あえてこのサポートおよび発展的な開発に取り組もうと考えた.

しかし, 欠点としては, 開発が中止されたことと, 後処理の変換用ソフトが JAVA の JAR ファイルで提供されており, 通常バッチファイルで JAVA のバージョンの変化を吸収して, 実行させることがデフォルトの仕様になっていて, 頻繁に更新される JAVA のバージョンアップに伴い, バッチファイルの変更が必要となる. 枠組みとしてはシンプルで優れたものであるのにもかかわらず, 多くの支持を得られることがなかったのは, 本来汎用的な JAVA ベースであることが裏目に出たと思われる. 私の研究室では, PPower4 及び TEX のファイルが簡単に作成できるよう, 専用のエディタも開発しており, JAVA のバージョンも自動で取得し, いくつかボタンを押せば, ファイル作成が簡単にできるようになっている. しかし, 発展性を考えると, やはり上位互換の変換プログラムがあることが望ましい.

そこで我々は, 特定の DLL を要したりすることなく実行可能なファイルを提供できる DELPHI で, 変換ファイルの作成を目指した. その作成には, PDF の仕様は公表されているものの, その解説は分かりにくく, また有料の解析ソフトでも表示できないような部分まで PDF の構造自体を解明する必要があり, それにかなりの時間を有したが. まだ完全ではないものの, PPower4 のライブラリで提供されている基本命令を埋め込んだ簡単なファイルであれば変換可能なレベルまでにはソフト開発できた. そして, 自作のいくつかの命令も使えるようになったので, 今回は簡単な解析部分とプログラム設計における重要部分を中心としてその報告をしたい.

2. PDF の解析

まず間単に PDF ファイルの構造を説明していく.PDF ファイルはヘッダ, ボディ, 相互参照表, トレーラの四つの部分に分かれている.

- Header　ファイルが準拠する PDF 仕様のバージョン番号を表す.
- Body　ページコンテンツやグラフィックスなどの文書の内容を表現する obj（オブジェクト）を格納するセクションである.
- xref　ファイル中の各オブジェクトの位置を一覧化し, ランダムアクセスを可能にするための情報が格納されたセクションである.
- trailer　相互参照表の位置およびファイルのボディ内にある一部の特別なオブジェクトの位置を示す.

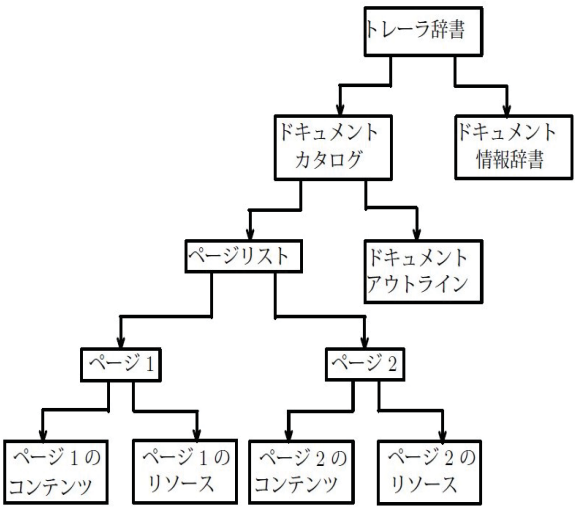
以下のコードは TEX から作成した PDF ファイルをテキストエディタで開いたものである (一部省略).

```
%PDF-1.4
%蓿□
2 0 obj
<</Creator( TeX output 2012.04.19:0741)
/Producer(dvipdfmx \(20100328\))
/CreationDate(D:20120419074140+09’00’)>>
endobj
1 0 obj
<</Pages 7 0 R/Type/Catalog>>
endobj
9 0 obj
<</Subtype/Type1C/Filter/FlateDecode/Length 362>>
stream
（バイナリ文字列部分）
endstream
endobj
4 0 obj
<</Type/Font/Subtype/Type1/Widths 8 0 R
/FirstChar 48/LastChar 48/BaseFont
/ELUSRH+CMR10/FontDescriptor
10 0 R>>
endobj
xref
0 11
0000000000 65535 f
0000000506 00000 n
0000000379 00000 n
0000000231 00000 n
0000001017 00000 n
0000000015 00000 n
0000000153 00000 n
0000000306 00000 n
0000000551 00000 n
0000000572 00000 n
0000001143 00000 n
trailer
<</Root 1 0 R/Info 2 0 R
/ID[<4d09c02cfe3e21e4e9557d58381ad408>
<4d09c02cfe3e21e4e9557d58381ad408>]/Size
11>>
startxref
1320
%%EOF
```

obj は番号付きで，文書構造，各ページの内容，フォント定義，画像データなどのファイルを構成する様々な要素を持っている．上のコードに 9 番 obj の stream から endstream までの間の部分がは非可読文字列である，このような文字列の部分は文字，埋め込みフォントや画像などの情報が主に FlateDecode という圧縮方式で圧縮さているものである．これは obj に格納されている情報が膨大化する恐れがあるため，今の PDF ファイルでは，このような情報が obj の stream から endstream までの間に圧縮されている．圧縮部分のデータを有料の解析ソフトでも表示できなかったので，TEX から生成される PDF ファイルによく出る Type1 と CFF の二種類埋め込みフォントの圧縮データと文字の圧縮データに対して，Ruby で作成したプログラムを用いて解析することができた．

PDF の文書構造は，obj の参照関係で構成されている．文書の構造情報，ページごとの描画情報や文字情報などはそれぞれ独立した obj として定義され，お互いに参照させられることで，階層構造となっている．

一般的な PDF の文書構造はトレーラ辞書，ドキュメントカタログとページツリーによって，構成されている．以下の図は 2 ページからなる一般的な PDF の文書構造である．



3. 対応できた PPower4 ライブラリの命令と自作命令および使用方法

PPower4 というライブラリでは提供されている基本の命令はページの背景，ページのトランジション効果とページの pauselevel に関する命令が提供されている．今のところで対応できた命令と自作の命令および使用方法を紹介する．

3.1 ページの背景

画像を背景として扱う場合では以下のコードを使う．

```
\usepackage[dvipdfm,bgadd]{background}
\bgadd{\includegraphics[設定サイズ]{画像名前}}
```

色を指定し，背景として扱う場合では，以下の三つの命令が使える．

- `\pagecolor{color1}` 単色の背景を指定するときに使う．
- `\vpagecolor[color1]{color2}` 上から下までのグラデーション背景を指定する．
- `\hpagecolor[color1]{color2}` 背景が左から右までのグラデーション背景を指定する．

3.2 ページのトランジション効果

ページのトランジション効果の部分は以下の命令が使える．

- `\pauseReplace` 単に元のページを新しいページに切り替える．pause と同じ．
- `\pauseDissolve` 元のページイメージをモザイクの様に漸次に分解させ，次のページが現れる．
- `\pauseHBlinds` ページと平行する等間隔の複数の直線が下に移動し，次のページが現れる．
- `\pauseVBlinds` ページと垂直する等間隔の複数の直線が右へ移動し，次のページが現れる．
- `\pauseHOSplit` ページと平行する二本の直線はページの中心から上と下に移動し，次のページが現れる．
- `\pauseHISplit` ページと平行する二本の直線はページの上と下から中心に移動し，次のページが現れる．
- `\pauseVOSplit` ページと垂直する二本の直線はページの中心から上と下に移動し，次のページが現れる．
- `\pauseVISplit` ページと垂直する二本の直線はページの左と右から中心に移動し，次のページが現れる．
- `\pauseOBox` ボックスの様な矩形領域が中心から外側に動いて，次のページが現れる．
- `\pauseIBox` ボックスの様な矩形領域が外側から内側に動いて，次のページが現れる．
- `\pauseWipe\{value\}` スクリーンを通し，一本の直線は片端から移動して次のページが現れ，その角度引数の可能な値は 0, 90, 180 と 270 である．
- `\pauseGlitter\{value\}` スクリーンの一方から広い範囲に広まっていくことで，次のページが現れ，その方向を指定している引数の可能な値は 0, 270 と 315 である．

3.3 ページのポーズレベル

ポーズレベル命令を使うことで，指定したポーズレベルナンバーによる，異なるレベルで異なるテキストが表示することが設定できる．以下にポーズレベルに関する用法を簡単に紹介する．(n, a, b, c, d は正整数である．)

- `\pause\pauselevel{=n}text\pause` ナンバー n の位置から示している．
- `\pause\pauselevel{=+n}text\pause` プラス n+1 の位置から示す．

- \pause\pauselevel{=-n}text\pause マイナス n+1 の位置から示す.
- \pause\pauselevel{=a : b}text\pause ナンバー a から b までの位置を示している.($a <= b$)
- \pause\pauselevel{=$\pm a$:$\pm b$}text\pause プラスかマイナス $a+1$ からプラスかマイナス $b+1$ までの位置に示している.($\pm a <= \pm b$)
- \pause\pauselevel{=$a : b, = c : d$}text\pause ナンバー a から b までの位置とナンバー c から d までの位置を示している.($a <= b$ かつ $c <= d$)
- \pause\pauselevel{=$\pm a$,:$\pm b$,=$\pm c$,:$\pm d$}text\pause プラスかマイナス $a+1$ からプラスかマイナス $b+1$ までの位置とプラスかマイナス $c+1$ からプラスかマイナス $d+1$ までの位置に示している.($\pm a <= \pm b$ かつ $\pm c <= \pm d$)

3.4 自作の命令

最新の PDF リファレンスを参考して, いくつの命令を作成し, 新しいページのトランジション効果ができるようにした.

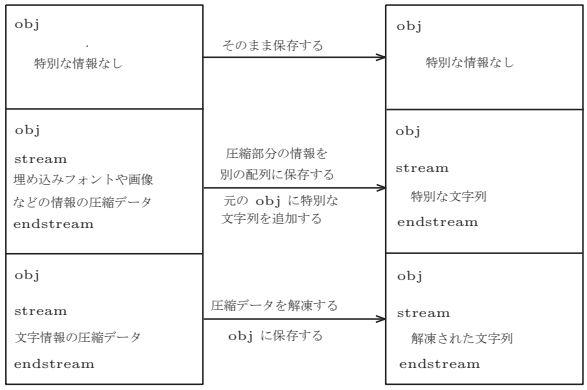
- \pauseIFly 外側から飛んでくる, 新しい画面が現れる.
- \pauseOFly 新しい画面に現れてから, その位置から外側に飛ばされる.
- \pauseOCover 新しい画面は現在の画面をかぶせる.
- \pauseOUncover 現在の画面を取って, 新し画面が現れる.
- \pauseOPush 新し画面がスライドに入って, 現在の画面が押されてスクリーンから離れる.
- \pauseFade 次のページは元のページを通して, 徐々に見えるようになる.

4. プログラムの設計

PDF ファイルの中身を変更するには, 各 obj を格納し, obj を処理する必要がある. しかし, PDF ファイルを一般のエディタで見ると, obj 部分にはテキストとバイナリのデータが混在している. 圧縮されている部分の情報が ascii コード範囲を超えている部分があるため, この部分のデータは文字型で収納できないので, Delphi のストリーム型に保存することでうまく処理することが可能になった.

全体の obj を格納する配列は動的配列を用いる. PDF ファイルを読み込んだら, obj の順番は昇順や降順ではないので, あとで, obj を管理しやくするために, obj を順番に保存必要がある. たとえば, 5 番 obj は先頭は 5 0 obj で始まり, endobj で終わる.temp にはもし 0 obj という文字列がヒットすれば, ここは obj の始まりだとわかる. そして, 前の番号を取得することで, 5 番 obj ということを認識し, ここからの temp を格納する動的配列の 5 番目に保存していく. temp に endobj という文字列にヒットしたら, この番号の obj の情報の保存をここで終了させる.

obj を格納するときに, もし圧縮部分がある場合は, stream が圧縮部分の始まりなので, ここから endstream までの情報を一旦 byte 型の配列に保存する. もしこの圧縮部分の情報が埋め込みフォントや画像の情報であれば, そのままストリーム型の動的配列に順番に保存していく. この時の obj の stream から endstream までの間に特別な文字列 (元は圧縮情報があると認識できる文字列とストリーム型の配列の番号) を追加する. もしこの圧縮部分の情報が文字情報であれば, この部分の情報を ascii コード範囲の文字列に解凍し, 元の obj の stream から endstream までの間の情報と置き換える. obj に対して以下の図ように三つの種類に分けて処理を行う.



プログラムの設計のほうでは, 主にポーズ (\pause), ページ効果 (\pause 指定子) とポーズレベル (\pauselevel{指定子}) といった命令の機能を実現する.

以下の TEX のソースは各機能を実現する命令を使った例である.

```
\begin{document}
A\pauseHBlinds
B\pause
\pauselevel{=+4}C\pauseVOSplit
\pauselevel{=1 :3}D\pause
\pauselevel{=1 :1, =4 :5}E\pause
\end{document}
```

この TEX ソースから PDF に変換すると, 文字情報および命令の部分は以下のコードとなる.

```
 Q BT/F1 29.74 Tf -19.92 19.19 TD[(A)]TJ ET
 q 1 0 0 1 -0.09 19.19 cm
%pause
 Q 0.90 0.50 0.00 rg 0.90 0.50 0.00 RG
 5.67 w 2.74 19.19 m 2.74 33.36 l
 S q 1 0 0 1 -0.09 19.19 cm
%esuap trans='Blinds,H'
 Q 1.00 1.00 1.00 rg 1.00 1.00 1.00 RG
 BT -19.92 -36.59 TD[(B)]TJ ET
 q 1 0 0 1 -0.09 -36.59 cm
```

```
%pause
 Q 0.90 0.50 0.00 rg 0.90 0.50 0.00 RG
 5.67 w 2.74 -36.59 m 2.74 -22.42 l
 S q 1 0 0 1 -0.09 -36.59 cm
%esuap
 Q 1.00 1.00 1.00 rg 1.00 1.00 1.00 RG
 q 1 0 0 1 -19.92 -36.59 cm
%pauselevel =+4
 Q BT -19.92 -92.36 TD[(C)]TJ ET
 q 1 0 0 1 -0.92 -92.36 cm
%pause
 Q 0.90 0.50 0.00 rg 0.90 0.50 0.00 RG
5.67 w 1.91 -92.36 m 1.91 -78.19 l
 S q 1 0 0 1 -0.92 -92.36 cm
%esuap trans='Split,V,O'
 Q 1.00 1.00 1.00 rg 1.00 1.00 1.00 RG
 q 1 0 0 1 -19.92 -92.36 cm
%pauselevel =1 :3
 Q BT -19.92 -148.14 TD[(D)]TJ ET
q 1 0 0 1 1.56 -148.14 cm
%pause
 Q 0.90 0.50 0.00 rg 0.90 0.50 0.00 RG
 5.67 w 4.39 -148.14 m 4.39 -133.97 l
 S q 1 0 0 1 1.56 -148.14 cm
%esuap
 Q 1.00 1.00 1.00 rg 1.00 1.00 1.00 RG
 q 1 0 0 1 -19.92 -148.14 cm
%pauselevel =1 :1, =4 :5
 Q BT -19.92 -203.91 TD[(E)]TJ ET
 q 1 0 0 1 -2.16 -203.91 cm
%pause
```

4.1 文字情報部分の分割

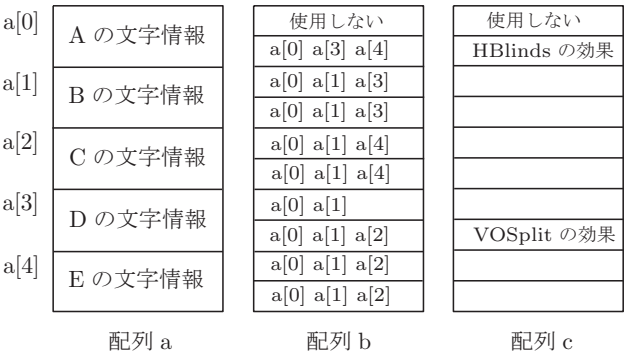
ポーズ効果などを各機能を実現するには, まず文字情報の部分を分割する必要がある. TEX ソースコードに\pauseというポーズ機能を実現するための命令を埋めこむ必要がある. この命令を文字の間に入れると生成した PDF ファイルの文字情報部分には%pause と%esuap という特別な文字列が埋め込まれる.

このコードを解析する, %pause から%esuap の部分が不要な情報であるため, 最初の部分から%pause の情報と%esuap から%pause までの情報だけを取って, 順番に設定した動的配列に保存する処理を行えば, 文字情報の分割ができる. しかし, 保存した PDF の文字情報描画命令を解析すると, q と Q のオペレータの命令が正しく設定されていないため, 保存した新しい情報の先頭に q を追加する処理と末尾に Q を追加する処理が必要である.

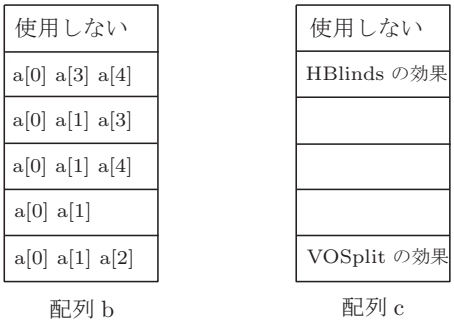
次に, 分割した各文字情報を格納する動的配列 a と各ページに対応する文字情報の参照番号を格納する動的配列 b と対応するにページ効果のデータを格納する動的配列 c を宣言する. この場合の例に対して, b と c の配列の長さの初期値を 10 とする (プログラムでは b と c の長さの初期値を 15 にしている). 配列 b と c の何番目に配列 a の文字情報の参照番号とページの効果データを追加するには一つの変数 x を宣言する. 配列 b と c の 0 番目を使用しないことにしているので, x の初期値を 1 とする.

文字表各部分の文字情報を分割するときには, %pause と%esuap という文字列を利用する. 一部の文字情報を配列 a に格納できてから, もしその後に%esuap の命令以外に別の命令がない場合では, 配列 b に x 番目から最後まで文字情報データの参照番号を追加する. もし%esuap trans という命令が出れば, 配列 c の x 番目にこの時のページ番号目に対応するページ効果のデータに追加する. もし, %pauselevel という命令が出れば, x を基準にして, 後ろの命令で指定した全てのページにこの時の文字情報データの参照番号を追加する.

このようなプログラムで上のデータを処理すると, 三つの配列は以下のようとなる.



上の配列 b の要素を確認すると, 格納している文字情報の参照番号の文字列が重複している部分があるため, 隣に重複している要素を削除する必要がある. もし配列 b の 3 番目が削除されたら, 配列 c に対して, 3 番目の削除処理も行わないと, ページ効果がずれてしまう. 上の配列 b と c をさらに処理すると以下のようになる.



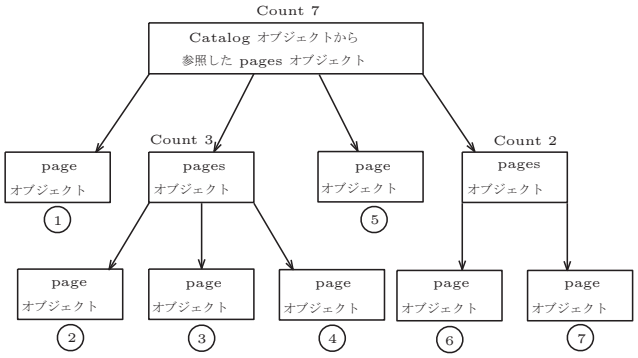
このように, 新しく追加するページが参照する文字情報

と対応するページ効果のデータを取得することができる.

4.2 ページの追加とページ参照処理

PDF ファイルの全体は相互参照するという形となっている. ページ構成は pages オブジェクトから pages オブジェクトまた page オブジェクトに参照し, page オブジェクトから文字に関する情報を格納するオブジェクトに参照するのである.

解析した PDF ファイルからよく見かける複雑なページ構成は以下の図のようになる. page オブジェクトから文字に関する情報を格納するオブジェクトに参照する部分を省略している.

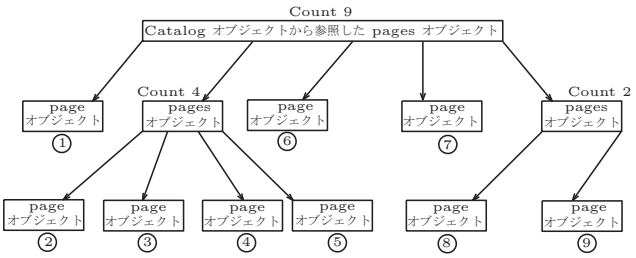


上の図のように, 一番上にあるオブジェクトは Catalog オブジェクトから参照した pages オブジェクトである. このオブジェクトから page オブジェクトまた pages オブジェクトに参照する. 一番下にあるのは必ず page オブジェクトである. pages オブジェクトには Count という要素もある. この要素は自分から参照する page オブジェクトの数を意味している. Catalog オブジェクトから参照した pages オブジェクトの Count は PDF 全体の page オブジェクトの数を意味する.

プログラムでは, まず Catalog オブジェクトから pages オブジェクトの番号を取得する. この pages オブジェクトから参照したオブジェクトを一つずつチェックする. もし page オブジェクトの場合であれば, この page オブジェクトから参照した文字情報のオブジェクトに特別な命令 (`\pause` や`\pauselevel` など) が埋め込まれているかどうかを一つずつチェックする. もしあれば, 前の節で紹介した文字情報の分割などのプログラムで処理を行う. 元の page オブジェクトを置き換えて, 新しい文字情報を格納するオブジェクトと page オブジェクトなどを作成し, 相互参照関係をきちんと設定して, ページ追加が実現できる. もし pages オブジェクトの場合であれば, 参照する page オブジェクトを同じような処理を行う.

注意が必要なところとしては, pages オブジェクトの場合では, Count 要素の設定をもう一回しなければならない, 新しい page オブジェクトを作成したとしても, Count の数を増やさないと PDF ファイルに新しいページが現れないことである.

もし上の図の 3 番と 5 番の page オブジェクトが 2 ページに分割すれば, 以下の図となる.



元の 3 番 page オブジェクトが置き換えられて, 3 番と 4 番 page オブジェクトに分割される. これらに参照する pages オブジェクトの Count の数が 4 に変わり, Catalog オブジェクトから参照した pages オブジェクトの Count の数を 8 に変わる. 元の 5 番 page オブジェクトが置き換えられて, 6 番と 7 番 page オブジェクトに分割される. この時は Catalog オブジェクトから参照した pages オブジェクトの Count の数を 9 に変わる. このように, 新しいページの追加とページの参照処理を実現する.

プログラムで文字情報の分割, 埋め込んだ各機能を実現するための命令の処理やページの追加と参照処理などで obj 部分に対して様々な処理を行うことで, 各機能を実現する. このようなに obj の変更や新しい obj の追加などの処理されているが, obj を格納する動的配列に同時に変更や保存の処理を行うことで, obj の管理が簡単にできる.

新しく作成する PDF ファイルの名前は元のファイルの名前の引用して, 後ろに`_HP` をつけることにした. 保存する場所は元の PDF ファイルと同じ場所となる. 例えば, test というフォルダに `text.pdf` という PDF ファイルをプログラムで処理したら, このフォルダ内に `text_HP.pdf` という名前のファイルが生成される.

5. 終わりに

本稿では, PPower4 システムを参考に, TEX からプレゼンテーション用の PDF ファイルを作成する支援ソフトウェアを提案した. PDF ファイルの解析を踏まえ, プログラムの設計と実装を行い, PPower4 ライブラリの基本命令と自作命令が使えるようになっている. PDF ファイルを設計したプログラムで処理する時に, 不要な情報を出来るだけ減らすことに心がけていた. PPower4 で作成した PDF ファイルと比べ, 自作の PDF ファイルのほうが不要な情報が少なくなり, 出来上がったプレゼンテーション用の PDF ファイルのサイズも若干小さくなっている.

今後の課題として, 最新の PDF リファレンスを参考にし, 新しい命令を追加し, さらなる効果が便利に使えるようにすることである. 可能であれば, 多言語化（中国語な

ど）についても研究し, 様々な国のフォントを使った PDF ファイルを 変換できるようにしたいと思う.

謝辞 本研究に際して、様々なご指導を頂きました藤井淳一先生に深謝いたします. また, PDF ファイルの解析を手伝ってもらった四回生の邱艷淋さんに感謝いたします.

参考文献

[1] Adobe Systems:「PDF リファレンス第 2 版」, ピアソン・エデュケーション, (2001).
[2] Adobe Systems:「PostScript リファレンス第 3 版」, アスキー, (2001).
[3] John Whitington:「PDF 構造解説」, オライリー・ジャパン, (2012).
[4] Adobe Developer Support: The Compact Font Format Specification,(1998).
http://www.adobe.com/content/dam/Adobe/en/devnet/font/pdfs/5176.CFF.pdf
[5] Adobe Developer Support: The Type 2 Charstring Format, (1998).
http://download.microsoft.com/download/8/0/1/801a191c-029d-4af3-9642-555f6fe514ee/type2.pdf
[6] Adobe Systems Incorporated: Adobe Type 1 Font Format, (1993).
http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF
[7] Adobe Systems Incorporated:Adobe-Japan1-6 Character Collection for CID-Key Fonts, (2008).
http://partners.adobe.com/public/developer/en/font/5078.Adobe-Japan1-6.pdf
[8] George Nagaoka: OpenType/CFF の仕様の解説,(2011).
http://d.hatena.ne.jp/project_the_tower2/20101204/1291482945
[9] 横浜工文社: 手書き PDF 入門,(1999).
http://www.kobu.com/docs/pdf/pdfxhand.htm
[10] K.Guntermann and C.Spannagel:PPower4 Manual,(2002).
http://www.tex.ac.uk/tex-archive/support/ppower4/manual.pdf#search='PPower4+Manual'