PDF 文件格式及其向 PS 文件转换的研究

On Converting PDF File to PS File

王 婉 韩逸秋 徐福培

(南京大学软件新技术国家重点实验室 南京210093)

Abstract Portable Document Format(PDF) is put forward by image manipulation software company Adobe. It's developed from Page Description Language PS. And PDF has properties which make it more fitful for transporting on network. Considering PS can export high quality page and its dominant status in press domain, converting PDF file to PS file is of significance. Since most popular e-journals are PDF files, PostScript printer needs the ability of printing PDF files directly. This paper introduces PDF and the implementation of translating PDF into PS.

Keywords PDF, PS, Page description, Interpreter

一、引言

结构化的文档格式 Portable Document Format (PDF),由美国排版与图像处理软件公司 Adobe 于1993年首次提出,它是 Adobe 继页面描述语言 PS (PostScript)之后,推出的最重要的电子文件规范,被广泛地应用于电子文件传送、交换和发行。由于其"高保真"的特性,已成为事实上的国际标准,目前流行的电子出版物基本上都是 PDF 格式的。

PDF 从 PS 而来,具有与 PS 语言几乎相同的页面描述能力和相似的描述方法,但与 PS 语言不同的是、PDF 除了能描述复杂版面外,还具有交互功能(如超链接、交互表单等)、页面随机存取及字体仿真描述等特性。

PS 语言可以生成高质量的图文印刷输出,已成为工业标准并在当前印刷领域占统治地位。因此,利用 PDF 文件生成 PS 文件从而得到高质量的印刷输出,具有重要的意义。

本文将介绍 PDF 文件格式以及 PDF 文件向 PS 文件转换的实现机制。

二、PDF 的文件格式

2.1 PDF 概述

(1)PDF 是一种与应用软件、硬件以及操作系统无关的文件格式。PDF 文件包含 PDF 文档以及其他的支持数据。此外,PDF 文件还包含 PDF 规范的版本以及文件中重要结构的位置信息等。PDF 从逻辑上可分为四部分:对象、文件结构、文档结构、页面描述。

第一部分是基本对象类型的集合。由 PDF 用来表示对象,这些类型中除少数例外,绝大部分都与 PS 中的数据类型相对应。

第二部分是 PDF 文件结构。文件结构决定了对象在 PDF 文件中的存储方式,对象是如何被访问、更新的。文件结构与对象的语义无关。

第三部分是 PDF 文档结构。文档结构指定了基本对象类型如何被用来表示 PDF 文档的组成部分:页面、注解、超文本连接、字库等。PDF 文档包含一页或多页的页面描述,每一页可以包含文本、图像以及与设备、分辨率无关的图形的任意组合。PDF 文档也可以包含其他电子类信息,诸如超文本连接、声音、电影等。

第四部分是 PDF 页面描述。一个 PDF 页面描述,作为 PDF 页面对象的一部分,可以与其他部分无关地被解释。PDF 页面描述仅与 PDF 文档的其他部分有有限的交互作用。这样就简化了 PDF 向 PS 语言程序的转化工作。

(2)PDF 使用与 PS 语言相同的着色模型。PDF 文件由一系列编号的对象组成,这些对象与 PS 程序中的对象类似。PDF 页面中的文本、图形、图像使用了从 PS 语言操作符发展过来的操作符描述,二者的页面描述极其相似,因此 PDF 文件虽然不是 PS 程序,不能直接用 PS 解释器来解释,但是 PDF 文件里的页面描述部分可以直接转换为 PS 程序。

虽然如此,PDF 与 PS 相比,还是有很多不同点。主要表现在以下几方面:

·PDF 是一种文件结构,而 PS 是一种编程语言。PDF 不是程序设计语言,不含有过程、变量和控制结构。PDF 定义了足以描述页面的高层操作符,这些操作符以机器码直接实现,而不是象 PS 那样以程序代码实现,因此 PDF 页面描述可以被很快地还原。由于具有非常规范的程序结构,应用程序可以高效、可靠地在 PDF 文档中定位文档对象。因此 PDF 具有比 PS 更高的处理效率。PDF 以缺乏灵活性换取效率的提高。

·PDF 的严格结构定义允许应用程序对其中的某个对象进行随机存取，而 PS 只能顺序存取。由于页与页之间的不相关性，一个成千上百兆的 PDF 文件就可以放在 Internet 上，读者可以通过能自动嵌入 IE/Netscape 的 PDF 浏览器，以页为单位、动态地最小量下载数据并立即显示。

·PDF 文件中可以包含交互对象如超链接、交互表单等，而 PS 没有。

·PDF 中包含有字库的规范尺寸等字库描述信息，以便在字库不存在时进行字库仿真（而非简单的字库替代），保证文档显示的一致性。

2.2 PDF 中的对象

PDF 支持七种基本对象类型：boolean，number，string，name，array，dictionary，stream 以及空对象，并引入了间接对象的概念。

（1）对象类型　前五种对象类型较简单，这里仅介绍 dictionary 和 stream

1. Dictionary：字典是包含对象对的表。每对的前一个对象称为 key，必须是 name 类型；后一个对象称为 value，类型不限。字典表示为用《和》包括的多个 key-value 对，通常用来说明一个复合对象的多个属性，每个 key-value 对对应一个属性的名和值。例如：

```
<< /Type /Example
   /Length 12
>>
```

字典对象是 PDF 文档的主要构成部分。PDF 文档的许多部分，诸如页面和字库，都使用字典表示。

2. Stream：stream 与 string 类似，由一连串字符组成，但 String 必须完全读入而 stream 可以每次只读入一部分。包含有大量数据的对象（诸如图像、页面描述）都可以表示为 stream。stream 由一个字典和由关键字 stream 和 endstream 包括的一段字符组成。

```
〈stream〉::=〈dictionary〉
    stream{〈lines of characters〉}*
    endstream
```

stream 一定是间接对象，而 stream 中的字典必须是直接对象，字典描述了 stream 的一些属性诸如长度、使用的过滤器等。stream 和 endstream 之间构成 stream 的一串字符可以被包含在外部文件中，此时，可通过 stream 中的 dictionary 来指定该文件。stream 可以使用过滤器压缩或由二进制转化为 ASCII 码。

（2）间接对象　任何对象都可以被标志为间接对象。PDF 文件中使用了大量的间接对象和间接引用。

间接对象是一个被标志过的对象，这样它就可以被其他对象引用。间接对象由对象标志符（object ID）、直接对象，关键字 endobj 组成。对象标志符由 object number、generation number 和关键字 obj 组成。每个间接对象的 object number 和 generation number 唯一，在该对象生成时被赋值。

```
〈object ID〉::=〈object number〉〈generation number〉
                    obj
〈indirect object〉::=〈object ID〉〈direct object〉endobj
```

对间接对象的引用称为间接引用。它由间接对象的 object number、generation number 和关键字 R 组成。

〈indirect reference〉::=〈object number〉〈generation number〉R

2.3 PDF 文件结构

PDF 的文件结构（即物理结构）包括四个部分：文件头、文件体、交叉引用表和文件尾。

文件头：出现在 PDF 文件的第一行，指明该文件所遵从的 PDF 规范的版本号。

文件体：由一系列的 PDF 间接对象（indirect objects）组成。这些间接对象构成了 PDF 文件的具体内容，如字体、页面、图像等。

交叉引用表：为了实现对间接对象的随机存取，PDF 中设立了一个间接对象地址索引表，即交叉引用表。

PDF 文件中只包含一个交叉引用表，由一段或几段组成。将一个交叉引用表分为若干段是为了文件修改的需要，如果文件未被修改过，表中只有一段，每次修改文件后，附加上一段仅包含被修改对象的交叉引用表，而不需修改原有的交叉引用表。

每段由关键字 xref 开始，后跟一或多个交叉引用子段。子段中包含了 object numbers 连续的数个间接对象的入口。

子段由包含有两个数字（分别是子段中第一个对象的 object number 和子节中的入口数）的 header line 开始，后跟所有的对象入口，每个对象一行：

```
〈cross-reference subsection〉::=
    〈object number of first entry in subsection〉
    〈number of entries in subsection〉
    〈cross-reference entry〉+
```

对象入口有两种类型：一种表示正在被使用的对象，另一种表示被删除、处于闲置状态的对象：

〈cross-reference entry〉::=〈in-use entry〉|〈free entry〉

对于正在使用中的对象，入口中包含了该对象在文件中的位置（通过该对象起始处与文件头偏移的字节数来表示）、该对象的 generation number，以及关键字 n：

〈in-use entry〉::=〈byte offset〉〈generation number〉n〈end-of-line〉

当一个间接对象被删除，它在交叉引用表中的对象入口将被标志为 free，成为闲置状态，这种对象入口

中包含了下一个空闲对象的 object number、generation number 及关键字 f：

```
〈free entry〉::=〈object number of next free object〉
    〈generation number〉f〈end-of-line〉
```

这样，交叉引用表中的空闲对象形成了一个单向链表。

交叉引用表(由初始表和修改表组成)必须为 object number 从0到最大值之间的所有对象保留一个入口，即使其中的一些对象未被使用。

文件尾：

```
〈trailer〉::=trailer
                <<
                〈trailer key-value pair〉+
                >>
                startxref
                〈cross-reference table start address〉
                %%EOF
```

文件尾的前面部分是 trailer 字典，由关键字 trailer 及一个字典构成。trailer 字典中包含了交叉引用表中所有的入口数(包括初始的和增加的)、上一个交叉引用表的位置、Catalog 对象的位置以及加密等信息。文件尾的倒数第三行是关键字 starxref，倒数第二行是从文件头到文件中最后一个交叉引用表中的关键字 xref 的字节偏移，最后一行包含了文件结束标志%%EOF。

应用程序应该从文件尾读起，根据文件尾提供的信息，找到交叉引用表和整个 PDF 文件的根对象，从而控制整个 PDF 文件。

PDF 文件的内容可以被更新而无须重写整个文件，所做的更新可以附加在文件的尾部，原文件部分原封不动。PDF 文件被更新后，新加入的或修改的对象被附上，同时将附加上一段交叉引用表和一个新文件尾。更新后的文件将呈现如下的结构：

```
〈Updated PDF file〉::=〈PDF file〉
                        {〈update〉}*
        〈update〉::=〈body〉
                    〈cross-reference section〉
                    〈trailer〉
```

PDF 文件更新时添加的一段交叉引用表中仅包含了被改变、替换或删除的对象的入口以及对象0的入口。新添加的文件尾包含了前面的文件尾中的所有信息，并在 trailer 字典中记录了上一个交叉引用表的位置。当一个文件改变多次后，可以包含几个文件尾。

2.4 PDF 文档结构

PDF 提供了文档的电子表示法——一系列包含文本、图形、图像的页画，连同其他的一些信息诸如 thumbnails(页面的缩略图)、文本注释、超文本链接以及书签等。PDF 文件的文件体由这些共同表示文档的对象构成。

PDF 的文档结构是 PDF 文件内容的逻辑组织结构，它反映了文件体中间接对象之间的层次关系。PDF 的文档结构是一种树型结构，树的根节点就是 PDF 文件的根对象 Catalog 字典。根节点下有四个子树：页面树(Pages Tree)、目录树(Outline Tree)、线索树(Article Threads)、名字树(Named Destination)。页面树中，所有页面对象都是树的叶节点，树中的子节点可继承父节点的各属性。每一页包含了对它的内容(contents)、缩略图(thumbnail)以及注释(annotations)对象的引用。目录树管理书签(Bookmark)。书签建立了书签名与一个具体页面位置的关联，用户可以按书签名来访问文档的内容。线索树将文章线索及线索下的文章块(Article Bead)按树型结构组织起来进行管理。名字树则是建立一种字符串(名字)和页面区域的对应关系，让 PDF 文件中的其他对象能够用字符串名字来代表某一个页面区域。

PDF 文件的文件尾 Trailer 字典在其关键字 Root 的值中指定了 Catalog 对象的位置。Catalog 字典包含了对文档中页面树对象的引用、对表示文档书签的对象的引用、对文档的线索的引用以及命名资源表。例如：

```
1 0 obj
<<
/Type /Catalog
/Pages 2 0 R
/Outlines 3 0 R
...
>>
endobj
```

可以通过 Catalog 字典中关键字 Pages 所对应的页面树对象对文档页面进行随机访问。树中的所有节点都是字典。节点可以是子页面树对象或是页面对象，页面树对象中定义了文档中页面的次序。(为了优化阅读器的性能，Acrobat Distiller 和 Acrobat PDF Writer 构造了平衡树，应用程序使用有限的内存就可以迅速地访问含有数千页的文档。)页面对象是页面树的一个节点，它定义了一个包含了文本、图形和图像的页面的字典。字典中还定义了页面的大小、所使用的资源、页面描述内容、旋转角度、注释、缩略等信息，是整个文件的重点。

2.5 PDF 文件中的页面描述

PDF 一共有60条页面描述指令。这60条页面描述指令描述了页面上的一系列图形对象。这些图形对象可分为五类：路径对象(Path Object)、文本对象(Text Object)、图像对象(Image Object)、外部对象(External Object)和 shading object。它们是构成所有页面的基本元素。

1. 路径对象是由直线和曲线段组成的轮廓。路径对象可以用来表示直线、曲线、区域等，路径由一系列路径片段组成，可以包含若干子路径。路径后跟路径绘

制操作符。

2. 文本对象由一个或多个字符串组成，可以以任意方向放置在页面的任意地方。和路径对象一样，可以作为勾勒，填充区域或作为裁剪路径。文本对象由指定字符串、当前点的移动以及文本状态的操作符组成，由关键字 BT 开始，ET 结束。

⟨text object⟩ ::= BT ⟨text operator or graphics state operator⟩* ET

3. 图像对象由一些指定颜色模式的图像的集合组成。内嵌图像对象：作为外部对象 image 的补充，描述页面上的图像。

4. 外部对象是在内容流外定义的对象，PDF 支持三种外部对象：image，form 和 PS 语言片段。

5. shading 对象描述了页面上某个区域的颜色的平滑渐变。

图形对象的精确渲染由相应参数来控制，诸如当前线宽、字库以及行距等。这些参数是图形状态的一部分。图形状态在每一页的开始部分被初始化。PDF 的图表状态与 PS 语言的图形状态非常相象，只有少许区别，这里不赘述。

三、PDF 的生成

要理解 PDF，必须了解 PDF 文件如何生成，如何使用。目前 PDF 的生成有两种途径：

1. 通过打印方式生成 PDF，即通过一个虚拟的 PDF 打印机将应用程序的文字和图形指令（如 Windows 下的 GDI 指令或 MAC 下的 QuickDraw™指令）转换为 PDF 指令，并保存在 PDF 文件中，参见图1。例如，在安装了 Adobe Acrobat PDF Writer 之后，从理论上说所有的具有打印功能的应用程序都能将待打印的内容打印输出到 PDF 文件。

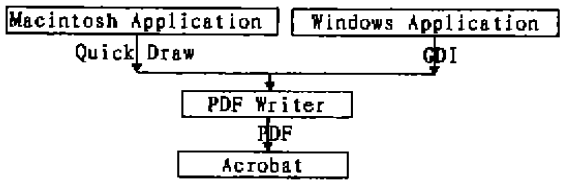

图1　通过打印方式生成 PDF

2. 由 PS 转换到 PDF 是另一种生成 PDF 的方法。由应用程序先将待打印的内容发排到 PS 文件，再由 Adobe Acrobat Distiller 将 PS 文件转换成 PDF 文件，参见图2。

两种生成 PDF 的方法各有利弊。通过打印方式生成 PDF 的优点是和应用程序能够紧密结合，在用户看来是从应用程序直接生成 PDF，但缺点是由于 GDI 指令集和 QuickDraw™指令集本身的局限，难以生成高精度的 PDF；而从 PS 转换到 PDF 虽然多了一道工序，但由于 PS 本身具有高精度的描述能力，因此生成的 PDF 可以达到印刷级的质量和精度。

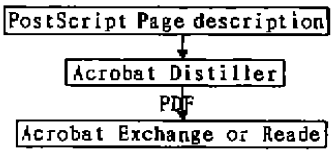

图2　从 PS 转换生成 PDF

生成 PDF 文件之后，用户就可以用 Acrobat viewer 来阅读和打印。用户可以通过页面缩像、超链接、书签（或目录）迅速访问文档中的任何指定部分，文档中的文本可以查找、摘录以便于其他应用程序使用，还可以给 PDF 文件增加如页面缩像、超链接、书签（或目录）、注释等交互属性。

四、PDF 文件向 PS 文件转换的主要技术

4.1　PDF 文件向 PS 文件转换的可能性

PDF 文件不是 PS 程序，不能直接被 PS 解释器解释。但 PDF 是在 PS 的基础上发展而来，二者具有的相似之处，决定了 PDF 与 PS 之间是可以相互转换的。PDF 和 PS 具有相同的成像模式以及相似的页面描述；大部分 PDF 页面描述操作符在 PS 中都有与其直接对应的操作符。PS 支持 PDF 中的字库。PDF 文件里的页面描述可以转换为 PS 程序，通过这种方式，可以使 PS 释器直接解释 PDF 文件，从而产生高质量的打印输出。

4.2　PDF 文件向 PS 文件转换的主要技术

PDF 文件转化为 PS 程序的过程主要由以下几个部分组成。

1. 构造 PS 文件的序言部分　PS 文档通常由两部分组成：序言部分（prolog）和描述部分（script）。序言部分中包含变量、过程定义；描述部分包含主程序部分。序言部分的执行不产生任何页面输出，描述部分根据序言部分的定义生成实际的页面输出。应在序言部分中用文档结构约定 DSC 简单说明版本号以及一些文档信息等。对于一些比较简单的、容易用 PS 语言定义的 PDF 操作符，就在序言部分直接用 PS 过程来定义。例如：/d{setdash}def 这样做的好处是减轻第6部分中页面扫描转换的工作量。

2. 构造交叉引用表信息　从 PDF 文件中读出交叉引用表，目的是通过 object number 得到对象的位置。

在2.3节中曾具体谈到 PDF 的交叉引用表以及文件尾。PDF 真正的文件尾中指定的都是主交叉引用表

的位置。可以从文件真正的文件尾中找出关键字startxref，从它的后面读出主交叉引用表的位置，然后根据该位置读出主交叉引用表中所有对象的入口信息。再从该交叉引用表后读出trailer字典中的信息，从中找出上一个交叉引用表的位置，读出上一个表中的入口信息以及表后面的trailer字典。如此循环，直到读出文件中所有交叉引用表中的入口信息。

注意：在PDF中，某个对象被修改过数次后，最近更新过的信息包括在最近附加到文件尾上的更新段中的交叉引用表中。因此，当构造交叉引用表信息时，必须用最近更新的入口信息覆盖到最初的交叉引用表中。

构造出PDF文件的交叉引用表信息后，就得到了文件中所有引用对象的位置信息，可以随时通过引用对象的object number定位该对象在文件中的位置，读出该对象。

3. *读出根对象Catalog字典和所有页面对象，构造页面对象数组* 在2.4中曾具体谈到Catalog词典以及页面树。从主交叉引用表后的trailer字典中可读出Catalog字典的信息，得到Catalog字典和交叉引用表信息就可以控制整个PDF文件。页面树由PDF文件中的所有页面节点组成，通过Catalog字典中Pages项找到页面树的根节点，根据根节点中的信息再找子节点直到找出所有页面对象。

页面树中各节点的组织具有前序深度优先的性质，可以用前序遍历算法读出所有的页面对象及其属性，依次写入预定义的页面对象数组。这样访问页面节点的次序与页面的实际页码是吻合的。具体实现伪代码如下：

```
BOOL readPage Tree(页面树对象 x){
 for(x 的所有子节点)
  {
    if (子节点为页面对象)
      将页面对象读入预先定义的页面对象数组;
    else if(子节点为页面树对象)
     readPage Tree(子节点);
    else{
     出错处理;
     return FALSE;
    }
    }
  return TRUE;
}
```

4. *插入资源* 主要是字库的创建以及替代，其余的资源在解释页面时再创建。

页面对象的页面属性中包含了该页面的资源字典，其中指定了本页中使用的字库，字库在PDF中以字典表示，字典中指定了字库类型、该字库的PS字库名、编码以及替代信息等。

PDF文件中可以使用嵌入字库，嵌入字库采用字库文件机制，用字库字典中的/FontDescriptor的对应值指定字库描述字典，字库描述字典中又指定了定义字库的流或文件。对于嵌入字库，可以读入字库对象以及嵌入的字库定义，利用这些信息重新构造新的字库字典。

5. *页面描述部分的扫描转换* 由于PS的交互操作能力不够，当PDF文件转化为PS程序时会丢失大量的交互信息，可以被完全转换的只有页面描述部分，因此这部分是转换工作的重点。

PDF中页面内容（如文字、图形、图像）都保存在页面对象Contents关键字所对应的内容流中。内容流可以使用过滤器过滤、压缩。对于那些由PDF中特有而PS中没有的过滤器所处理的内容流，在使用前必须进行解码。

解码后的内容流由一系列图形对象组成，每个图形对象由操作数和页面描述操作符构成。对于简单的PDF操作符，利用序言部分中定义的过程直接转换成对应的PS过程，对于序言部分中没有定义的，其他较复杂的操作符，则采用扫描转换，即可以采用与PS解释器相同的方法来解释这些图形对象，使用扫描器扫描内容流，根据PDF语法规则把字符组合成语言标记，然后根据这些语言标记创建对象并执行这些对象，如果该对象是操作数，则将其压入操作数栈，如果该对象是操作符，则从操作数栈中取出操作数，将该操作符以及与其对应的操作数解释为PS代码。

4.3 系统运行流程

1. 初始化处理。主要工作是初始化全局变量、出错信息以及读入cinfig文件等。

2. 打开PDF文件，用自定义的数据结构构造文件信息。主要工作是打开文件、创建文件流、检查文件合法性、构造交叉引用表信息、页面信息等。

3. 从文档信息中获得页面的范围。

4. 构造输出的PS文件并初始化。主要工作是建立文件，写入文件头，写入预定义的序言部分，根据页面信息中的资源字典创建用于替代的PS字库。

5. 逐页扫描转换，将页面描述中用PDF操作符描述的内容流转换为PS程序。

6. 结束处理。释放申请的内存空间等。

参 考 文 献

1 Adobe System Incorporated. Portable Document Format Reference Manual, Version 1.3, March 1999
2 Adobe System Incorporated. PostScript Language Reference, Third Edition. Feb. 1999
3 徐福培，潘志庚，等. 页面描述语言及其程序设计. 南京大学出版社，1994
4 张效祥，徐家福，等. 计算机科学技术百科全书. 南京大学出版社，1998
5 杨道良，等. 面向对象的中文PDF阅读器的设计与实现. 计算机应用，1999，19(6)
6 杨道良，等. PDF及其在电子出版领域的应用. 计算机应用，1999，19(1)