中文页面描述语言文本管理器的设计与实现*

徐福培 张炜

(南京大学计算机科学与技术系,南京 210093)

摘要 文本管理器是用于提高中文页面描述语言解释器效率而引入的前处理系统,它包括资源管理、异常处理、EPSF管理和文本优化等功能模块.本文详细论述了文本管理器的设计思想与相应实现方法.

关键词 中文页面描述语言,文本结构约定,文本管理器,被封装的PS文件,资源优化.

1. DM的由来

中文页面描述语言(CPDL)在处理图形、正文、图象等信息时显示出其它语言无法比拟的优越性.随着彩色空间概念及其一系列新操作符的引入,这些特点在我们自行研制开发的中文页面描述语言解释器CPDL-2中得到了更好的发扬.由于语言指令集合的扩展,用CPDL语言书写的程序越来越复杂,体现在指令序列变得冗长,程序结构也变得越来越不规范和明晰.

考察页面描述语言的发展历程和现状[1],就其两大代表产品Interpress(Xerox公司)和PostScript(Adobe公司)作一比较,可以发现:PostScript具有良好的成像模型,而Interpress具有清晰的文本结构.ISO于1992年通过的SPDL(Standard Page Description Language)标准草案(ISO/IEC DIS 10180)[2]就是集二者优势为一体的一个成功范例.考虑到与现有CPDL解释器的兼容性,即既要达到很强的页面描述功能,又要做到优良的文本结构,提出了文本结构约定(Document Structure Conventions,简称DSC)的概念,并相应引入了称为文本管理器(Document Manager,简称DM)的前处理系统,它们包括资源管理、扫尾处理、工具包等.

2. DM与CPDL解释器的关系

从系统的角度看,DM位于CPDL解释器的前端,它的输入为一嵌有符合DSC的注解语句的CPDL源代码级文本,输出为一经过DM处理的CPDL源代码级文本(DM与CPDL解释器的关系如图1所示).DM不能对文本中的CPDL代码作任何改动,也不能对其后端的CPDL解释器产生任何副作用.它根据文本中的DSC语句,产生一些CPDL语句代码并插入文本的相应位置.实质上DM完成了文本解释过程中的预处理环节.

* 本文1993-07-13收到

作者徐福培,55岁,副教授,主要研究领域为计算机图形学与页面描述语言.张炜,23岁,硕士生,主要研究领域为计算机图形学与页面描述语言.

本文通讯联系人:徐福培,南京 210093,南京大学计算机科学与技术系

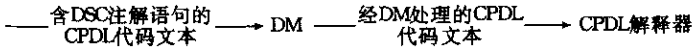

图1 DM与CPDL解释器的关系

对 DSC 的处理和对文本中 CPDL 代码的处理逻辑上分别进行,而在实现时这种分别处理并不是泾渭分明的,即 DM 存在于 CPDL 解释器中,它是一个相对独立的预处理系统.

1 文本结构约定

1.1 CPDL 的文本结构

1. CPDL 文本结构构成

CPDL 语言是一种高级解释语言,它为程序员提供了完备的控制语句集合,并提出了过程的概念. 在进行 CPDL 语言的程序设计时,同应用其他高级语言(如 Pascal)进行程序设计一样,采取自顶向下、逐步精化的结构化程序设计方法. 同时注意到 CPDL 语言所具有的页面描述语言的特征,对文本结构的概念进行了扩充.

一方面可以将 CPDL 文本分成结构和内容两部分. 文本结构独立于内容,且可以不依赖内容单独处理(CPDL－2 中此项处理称为预处理);文本内容以页面描述语言的格式出现,它的处理依赖于文本结构.

另一方面可以将 CPDL 文本分成序言(Prolog)和描述(Script)两部分[3]. 序言部分只能定义过程,它的执行不产生任何页面输出,也不对图形状态栈产生任何影响;描述部分根据序言部分的过程定义生成实际的页面输出. 序言部分包括头部分(Header Section)、默认部分(Defaults Section)和过程定义部分(Procedures Section);描述部分包括文本设置部分(Document Setup)、页面部分(Pages)和文本尾部(Document Trailer). 其大致结构如图 2 所示.

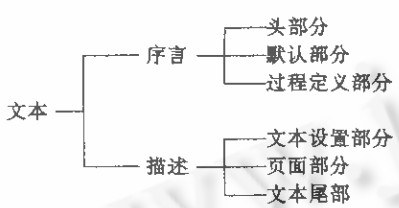

图2 CPDL文本结构

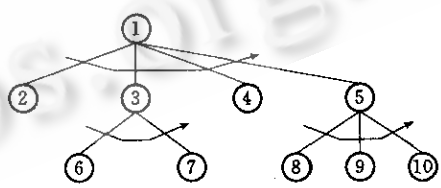

图3 CPDL文本结构序列次序

• 头部分:说明产生文本正确输出时所依赖的系统环境,必须全部用 DSC 书写;

• 默认部分:在若干页面重复出现的页面一级的注释,将它们统一放置在默认部分,以节省空间,同时也为文本管理器提供了资源优化的线索;

• 过程定义部分:包括一系列映像过程的定义;

• 文本设置部分:用于进行设备设置,包括对物理介质的选择,资源的下载操作及对某些图形状态参数的设置;

• 页面部分:该部分包括若干张页面的描述,页面之间应该是逻辑独立的,即这些页面描述可以以任意次序执行;

· 文本尾部:该部分进行一些扫尾处理(Post Processing)及对某些遗留的含有待赋(Atend)值的 DSC 语句进行处理.

2. CPDL 文本结构的特点

层次结构和序列次序是 CPDL 文本结构的两大特点. 可以将 CPDL 文本的结构划分为文本(Document)、页面(Page)、标记(Token)三大层次. 同一层次结构元素具有序列次序.

层次结构和同级结构元素的序列次序的结合,给出了组成文本的结构元素的一个关系序列. 层次结构和序列次序的关系具有两条原则:CPDL 文本的每个结构元素后面紧跟着它的所有下属;先于某同级结构元素的任何结构元素,其下属也先于该同级结构元素. 二者具有"前序深度优先(pre－order depth－first)"的性质(见图 3).

1.2 DSC 的语法描述

根据对 CPDL 文本结构的描述,可以作出结论:标准 CPDL 语言不能完整描述 CPDL 的文本结构,它还需要其他约束条件,文本结构约定(DSC)应运而生.

1. DSC 的 BNF 描述

DSC 语句中操作符与操作数的书写次序不采用逆波兰表示法,而采用传统的操作符前缀表示法(操作符在前,操作数在后),这在形式上有别于 CPDL 语言.

对 DSC 的语法描述采用严格的 BNF(Backus－Naur Form)描述,举例如下:

%%IncludeResource: ⟨resources⟩

$$\langle resources\rangle ::= font\ \{\langle fontname\rangle\}_1^{n1} \mid file\ \{\langle filename\rangle\}_1^{n2}$$
$$\mid procset\ \{\langle procname\rangle\}_1^{n3} \mid pattern\ \{\langle patternname\rangle\}_1^{n4}$$
$$\mid form\ \{\langle formname\rangle\}_1^{n5} \mid encoding\ \{\langle vectorname\rangle\}_1^{n6}$$

2. DSC 的数据类型

DSC 的基本数据类型包括文件名(filename)、字库名(fontname)、模版名(formname)、整数(int)、向量名(vectorname)、正文行(textline)、图案名(patternname)、过程名(procname)、实数(real)、正文(text)、资源(resource)、待赋(atend)等.

1.3 DSC 的分类

1. DSC 的功能分类

按 DSC 所能完成的功能可将其分为 6 类.

· 通用约定(General Conventions):该约定可以区分出 CPDL 文本中各种结构性元素,如序言和描述部分,它还对文本设置部分、页面设置部分及对各不同页面进行标识.

· 需求约定(Requirement Conventions):该约定用以对 CPDL 文本提供的和所需求的资源进行说明,文本管理器据之进行资源定义、资源下载(Resource Download)和优化操作.

· 分色约定(Color Separation Conventions):该约定对 CPDL－2 的彩色功能[4]进行了扩充,它能够标识出页面中使用不同分色的 CPDL 代码段,还可以对定制颜色(Custom Color)进行规格说明.

· 查询约定(Query Conventions):该约定用以标识查询 VM(Virtual Memory),解释器当前状态及获得资源能力的 CPDL 代码段,还可以交互地查询打印机的某些物理特点.

· 开放式约定(Open Structuring Conventions):该约定主要用于对 DSC 指令集进行扩充,是用户自定义的约定.

·特例约定(Special Conventions):该约定表明CPDL文本中含有exitserver或startjob操作符.这两个操作符的执行将中断正常的作业执行流,并对CPDL解释器产生影响,因此DM有必要对该文本进行某些处理.

2.DSC的结构分类

通用约定、需求约定、分色约定从结构上又可分为三大类,即头(Header)、体(Body)和页面(Page)三部分注释约定.

·头部分:文本中出现在第一条CPDL指令之前和过程定义(Procedure Defination)之前的DSC注释部分.

·体部分:该部分约定用于提供文本的结构性信息,它必须与头部分中的相关DSC语句一致.其形式一般为%%Begin…,%%End…或%%Include…

·页面部分:该部分约定为页面一级的约定.

1.4 DSC的最小遵从集

产生CPDL页面描述时,大的应用程序能指定所有DSC信息,而小型的应用只能指定一些基本的DSC信息(如%%EndProlog,%%Page,%%Trailer,%%DocumentNeededResources等),我们将这类基本DSC信息称为一个"最小遵从".可以认为DSC的最小遵从集是DSC指令集合的最小实现子集.一个符合"最小遵从"的页面描述必须提供有关程序结构和字库等资源需求的信息,其他DSC信息均可省略.

2 DM的总体设计

2.1 DM与CPDL解释器的接口

考虑到DM与CPDL解释器逻辑上的相对独立性,必须在现有的CPDL-2中添加独立的DM模块,使该CPDL解释器具有处理DSC的功能.根据这一思路,当在批处理方式下执行CPDL文件时,若加上开关参数-m,即表示将对CPDL文本中的DSC语句进行处理,否则解释器将不对其进行处理.

2.2 DM的功能模块

根据DSC的功能分类,相应地可以将DM划分为几大模块.流程图见图4.

1.通用约定处理模块

通用头部分DSC注释语句提供了文本的版权、来源、生成日期、作者等信息,此外它还提供了BoundingBox和文本包括的页面数等重要参数.

通用体部分DSC注释语句一般只用来标识CPDL文本中的结构元素,它体现了一种良好的程序设计风格.

通用尾部分DSC注释语句对应于头注释中的待赋(Atend)类型,主要用于处理源实用程序不能静态生成的动态DSC注解.

2.需求约定处理模块

根据DSC的规定,文本可以引用的外部信息有2种类型:资源和EPSF(见本文的第3、4节).

需求约定还提供了为用户设置打印机物理参数的开放式接口[5].这种方式通过DSC语句及在系统中配置的CPDF文件(CPDL Printer Description File)的相互作用来实现.该种

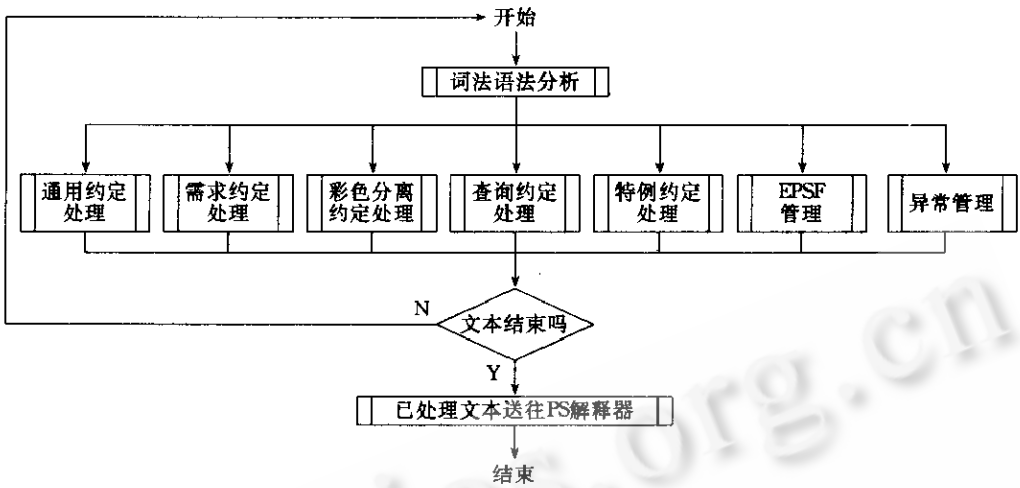

图4 DM的主要功能模块

方式的优点是用户不必了解 CPDL 解释器的内部结构,只需修改 CPDF 就可扩大 CPDL 解释器的应用环境.

3. 分色约定处理模块

首先引入 2 个名词:原色(Process Color)和定制色(Custom Color)[6].

原色:若 3 种颜色中任意两种颜色的组合都不能生成第 3 种颜色,则这 3 种颜色被称为原色. 如减色模型 CMY 中的青(Cyan)、洋红(Magenta)、黄(Yellow),加色模型 RGB 的红(Red)、绿(Green)、蓝(Blue).

定制色:在某彩色空间中按一定比例混合原色生成的颜色.

当 DM 被指定还原某种原色或定制色时,DM 只留下具有该种颜色的 CPDL 代码段,其它不同颜色的代码段一律不送 CPDL 解释器. 这样,CPDL 解释器的效率被大大提高了.

4. 查询约定处理模块

标以%! CPDL－2.0 Query 的 CPDL 文本,是用来查询的文本. 该文本中只包含查询约定注释语句(否则为非法文本),每一项查询都被%%? Begin…Query 和%%? End…Query 括起.

DM 配合 CPDF 对查询约定进行处理. 以下面 2 句为例:

%%? BeginFeatureQuery: ⟨featuretype⟩ [⟨option⟩]

－－－－用来查询的 CPDL 代码－－－－

%%? EndFeatureQuery: ⟨default⟩

DM 首先在 CPDF 中定位⟨featuretype⟩及相应⟨option⟩,若失败则返回默认值;否则使用 CPDL 代码进行查询处理(由 CPDL 解释器完成). 这种处理属于智能型的处理,因为如果 CPDF 中不含被查询项,DM 可以略去用于查询的 CPDL 代码而直接返回默认值.

5. 特例约定处理模块

若 CPDL 文本的开头为%! CPDL－2.0 ExitSever,则该文本将被当作一种未被封装的作业(Unencapsulated Job)来执行. 它将对 CPDL 解释器的作业执行环境产生不可挽回的

副作用,因此DM有必要将该文本从作业流中移去.

6. EPSF处理模块

当CPDL文本的开头为%!CPDL—2.0 EPSF—2.0时,DM调用EPSF处理模块.细节详见本文第4节.

7. 异常处理模块

DM对DSC进行处理时,若发生异常可以有三种管理方式:一是通知用户,并停止作业的执行;二是忽略,即发生异常后不进行处理,这样做的前提是用户可以完全控制和处理该异常;三是替换,如DM发现某个字库的定位出现异常,可用事先定义的替换策略进行更替.在初步实现的DM中,异常处理大都采用第一种方式.

3 资源管理

3.1 资源的概念

所谓资源,指一组具有共同特征的对象实例的集合及其定位策略.可以将资源分为字库、过程集、向量编码、图案、模版和文件等若干资源类.对于不在VM中的实例,每个类都有相应的定位策略;对于VM中的实例,每个类都有各自的管理方法.

资源是自定义的(Self Defined).通过执行findresource(CPDL—2)操作符或实用程序的资源插入步骤,使相应资源类的实例被定位,该实例是一段CPDL代码,自行完成VM加载的工作.

文本(Document)不是一种资源类,它是完全符合DSC的文件,这是其与文件资源类在概念上的最大区别.

3.2 资源的组织

在CPDL的文档中,资源的组织形式是由不同系统环境和解释器的具体实现决定的.

在目前的DM中,将资源按类组织在不同的子目录下.每一个资源实例对应一个CPDL文件.

3.3 资源定义

有些CPDL代码对应的图案或者编码向量可能被程序频繁调用,可以将其按类定义在资源库中.在引用时,从库中取出,这样可省去大量的源程序代码空间.

对应资源定义的DCS语句有如下3条:

%%DocumentSuppliedResources:〈类名〉〈实例名〉

该DSC语句出现在文本的头部分中,它对文本中定义的资源类名和实例名进行说明.

%%BeginResource:

…CPDL代码…

%%EndResource

DM将该DSC语句说明的资源实例名按类登记在数据结构suppliedres中

3.4 资源引用

在一个CPDL文本中引用外部资源可以有两种不同方式.一种是外部实用程序直接将其定义嵌入至作业流中去;另一种是CPDL解释器从外部源(如磁盘文件)将它加载到VM中去.

CPDL－2 的操作符 findresource 支持第 2 种方式.

对应资源引用有两条配对使用的 DSC 语句:

%%DocumentNeededResources:〈类名〉〈实例名〉

该 DSC 语句出现在文本的头部分,它对文本中所需的资源类及实例名进行规格说明.

%%IncludeResources:〈类名〉〈实例名〉

DM 在处理该 DSC 语句时,首先看该资源实例在 includeres 中有没有登记,若已登记则直接引用 findresource 这一操作符,否则报错.该操作符根据资源名按固定的策略先在 VM 中查找该资源实例.若未找到,则到磁盘的资源库中定位该资源文件,找到后插入 CPDL 文本中的资源引用处.若还未找到,则转异常处理.

3.5 资源优化

有些资源在文本的默认设置部分进行引用说明,其作用域为整个文本,有些资源为页面一级的引用,这样就出现重复引用.另外由于 VM 的大小有限,不可能无限制的将资源加载入 VM,所以出现了资源优化的问题.这里的"优化"并不是对资源的定义文件进行代码优化,而是对资源的引入点,引入次数进行优化.DM 的资源优化处理流程需要扫描文本 3 次.首先,优化文本一级引用的资源,在文本设置部分进行资源定位操作;其次,优化页面一级引用的资源,在页面设置部分进行资源定位操作;最后,在每一页面执行前插入 save 操作符,执行后插入 restore 操作符;在同一页面中,当某种资源下载 VM 时,若 VM 中的该类资源数已超限,则在引用该资源的语句前后插入 save,restore 操作符,对其进行"封装",以保护 VM.

4 EPSF 管理

4.1 EPSF 的概念

"被封装的 PS 文件(Encapsulated PostScript File)"是在各种系统环境中的实用程序用来输入(Import)和输出(Export)CPDL 文件的一种标准文件格式.这些实用程序分为两类.

一类称为源(Source),它们可以生成 EPSF,如 Coreldraw 3.0 等;另一类称为目标(Destination),它们对 EPSF 进行处理,如 DM,PageMaker 等.

4.2 EPSF 的特点

1.独立性:一个 EPSF 实际上是一个符合 DSC 的 CPDL 文本,它可被送往 CPDL 解释器进行解释.EPSF 是一种最终文件形式,实用程序不可对文件本身进行任何修改.

2.封装性:EPSF 是一种被封装了的文件.这里的"封装"指在 EPSF 执行后,不应对 CPDL 解释器产生任何副作用.根据这一特性,可以将一个 EPSF 嵌套于其它 PS 文本中.

4.3 EPSF 的预演机制

EPSF 中除了 PS 代码外,还可以包含一些用于实现预演(Preview)的位图映像数据.这些数据不对 EPSF 的输出结果产生任何影响.该机制可以对目标实用程序灵活排版起到辅助作用,达到所谓"所见即所得(WYSIWYG)"的效果.PC 机中映像数据的格式可以是 TIFF 格式,或是专用于 Windows 的元文件格式(Metafile).另外,为了达到较好的设备的无关性,引入了一种称作 EPSI 的文件交换格式.

预演不是必需的,它只是一种辅助手段.在 CPDL－2 中,批处理方式下运行 CPDL 文

件,若采用-v参数,即可在显示器上进行预演.

4.4 生成EPSF时的注意点

1. 必要的DSC注释

EPSF必须包含有两个头注解(Header Comments). 一个是%! CPDL-2.0 EPSF-2.0,该DSC语句表明该文本为EPSF,DM必须对其进行某些处理;另一个是%%BoundingBox: llx lly urx ury,该DSC语句为DM对EPSF进行裁剪提供了信息.

2. EPSF中受限的PS操作符

CPDL语言中的某些操作符在EPSF中不能使用,另外statusdict中的所有操作符以及userdict中与映像相关的操作符均不能被调用. 这类操作符的使用将影响EPSF的封装性,对作业的执行产生无法挽回的副作用.

4.5 DM对EPSF的处理

1. 栈和VM的保护措施:执行EPSF前,应对图形状态栈和VM执行save操作,并对图形状态栈中的参数进行初始化;将用户字典压入字典栈;将操作数栈中的操作数保存至一数组中,清操作数栈. 执行EPSF后,恢复操作数栈;弹出字典栈中处理EPSF前压入的字典;通过执行restore操作将图形状态栈和VM恢复.

2. 坐标变换:DM不能对EPSF的内容进行任何添删,但可以对EPSF在用户空间中的位置进行坐标平移、旋转、变比等操作. 在这一过程中,EPSF的头注解中必须提供其BoundingBox的信息.

3. 裁剪路径的设置:坐标变换结束后,执行EPSF前,可以设置裁剪路径.

4. 操作符showpage的重定义:操作符showpage隐含有erasepage和initgraphics这两个操作符的功能,而这些功能恰是执行EPSF时必须限制的,因此应对showpage进行重定义.

5 结束语

目前开发的CPDL-2是一个面向彩色印刷系统的软件模型,开发具备网络功能并且拥有Spooling管理和动态字库生成的DM正在进行之中. 此外,为减少后继处理者—CPDL解释器的负担,资源优化、分色及异常处理的智能化机制有待进一步深化. DM目前处理的DSC指令集合不是封闭的,可以根据系统环境的具体情况进行扩充.

参考文献

1 Spring, Michael B. Electronic printing and publishing: the document processing revolution. New York: Marcel Dekker, 1991.

2 ISO. SPDL (ISO/IEC DIS 10180). 1992.

3 Adobe System Inc.. PostScript Language Reference Manual Supplement for Version 1012. 1992.

4 David A Holzgang. Understanding PostScript. Adobe System Inc., 1992.

5 Adobe System Inc.. PostScript Language Reference Manual. 1991.

6 罗杰斯 D F. 梁友栋等译. 计算机图形学的算法基础. 北京:科学出版社,1987.

THE DESIGN AND IMPLEMENTATION OF A DOCUMENT MANAGER OF CHINESE PAGE DESCRIPTION LANGUAGE

Xu Fupei and Zhang Wei

(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

Abstract Document manager is a pre—processing system which is designed to improve the efficiency of chinese page description language interpreter. It includes such function modules as resource management, error management, EPSF management and document optimization programs. This paper discusses the details of the design of document manager and its corresponding method of implementation.

Key words Chinese page description language (CPDL), document structure conventions (DSC), document manager (DM), encapsulated postscript file (EPSF), resource optimization.