データベースシステム 101−12
（1995. 1. 27）

構造化文書を対象とした操作コマンド 紫

楠本 浩二　黒澤 宏　鈴木 克明

富士ゼロックス(株) ドキュメントシステム開発センター
アプリケーションシステム商品開発部

文書を自動編集するシステムについて述べる。文書を作成するとき，既存の文書の一部を利用する。構造化文書であるODAやSGMLは，体裁との独立から論理構造の一部を部品として再利用できる。文書の構成が明確かつ安定した文書を対象とすると，編集手続きを定式化できるので，プログラムの記述による自動編集が可能である。われわれは，構造化文書を操作するコマンド群「紫」を開発した。文書部品の複写，削除，生成，変更の機能を持つ編集コマンドを連携したプログラムを記述する。対象である部品の指定は，オブジェクトの属性値と論理構造におけるオブジェクト間の関係からなる"文脈情報"を利用する。本稿では紫の概要，記述例，性能，評価を示す。

Murasaki: A System Automating Document Manipulation

Koji Kusumoto　Hiroshi Kurosawa　Yoshiaki Suzuki

Document Systems Development Center
Fuji Xerox Co., Ltd.
KSP/R&D, 3-2-1, Sakado, Takatsu-Ku, Kawasaki-Shi, Kanagawa 213, Japan

This paper reports on System *Murasaki* comprising UNIX™ commands, a product automating document manipulation. Structured documents have the logical structure independent of their physical aspects. This feature enables us to produce new documents based on reusing previously existing documents and their components. *Murasaki* gives users editing operations such as copying, deleting, creating and changing components. *Murasaki*'s chief novelty is a "contextual information" query paradigm. Contextual information consists of object attributes and structural locations. The system is sufficient for editing long life cycle documents and compound documents. We also introduce some examples of programs, and evaluate this system.

1 はじめに

木構造の構造化文書モデル[1]を対象とした編集システム[2]は，文書部品の柔軟な再利用を目的としている。構造化文書の国際規格には，ODA[3]と SGML[4]がある。文書要素を線形に並べた WYSIWYG 文書モデルに対して ODA や SGML は，「交換性」「部品単位の機能性」「体裁との独立」の観点から再利用時に有効である[5]。

文書の論理構造は，その使用目的と密接に関係している。特に論文や特許，仕様書といった文書構成が明確で安定した，ライフサイクルの長い文書の再利用性は高い。このような文書を利用し，新たな文書を作成するとき，必要な文書部品を抽出したり，あるいは不必要な文書部品を削除するといった編集の手続きを定式化しておくことができる。これにより人間の介在を必要としない文書の自動編集が可能となる。自動編集のためのプログラムには，「対象文書」，「処理操作」，「操作対象範囲の指定」の記述が必要である。

以前，われわれは文脈情報に基づいた文書自動操作システムを提案した[6]。本稿では，文書を構成するオブジェクト内のタイプや属性情報と，構造上のオブジェクト間の関係情報を文脈情報と定義する。木構造の論理構造では，文脈情報は木構造の包含関係を表す。

文書データーベースシステム[7]では，文脈情報に基づく問い合わせを行なっている。またテキスト解析手法[8]では，主要な文書情報を抽出するとき，見出し情報から，論理的な文書フレームを検索するトップダウン的手法を併用している。見出し情報とその配下の段落を指定する条件記述は，文脈情報の典型例である。

文書の自動編集においても操作の対象範囲の指定として文脈情報を用いることは，有効な方式と考える。われわれは，文脈情報を用いた構造化文書の自動操作システムである紫(1を開発し，製品化した(2。実用的な文書操作システムの構築を目的として以下の要件に留意した。

(1) 文脈情報を簡潔に記述できる。

(2) 基本的な木構造操作が実行できる。

(3) 基本的な文書編集操作が実行できる。

本稿では，システムと紫の機能仕様の概要について述べる。さらにプログラム例，性能，比較および使用実績について言及する。

2 システム概要

構造化文書を対象とした対話型編集システム *Akane*[9]が製品化されている(3。*Akane* は，文書の編集，レイアウト，印刷といったオフィス向けの一連の処理への適用性と，論理構造および割り付け構造の両方の交換の容易性から ODA モデルを採用している。

Akane と紫は UNIX™ 上で稼働する(4。紫は構造化文書フォーマットのファイルを対象に操作するコマンド群である。ODA のファイルは，主にバイナリ形式であるが，*Akane* の文書ファイルは，*CForm* と呼ばれる図.7 に示すキャラクタ形式の公開フォーマット[10]である。文書部品ファイルは，文書の一部を格納したファイルである。紫はキャラクタ形式の文書または文書部品ファイルを入力し，操作した結果を文書または文書部品ファイルとして出力する。

紫は UNIX コマンド形式で呼び出し実行するので，入力ファイルにはいかなる影響も及ぼさない。コマンド名が処理操作に対応し，コマンド引数として，オプション，文脈情報，文書または文書部品ファイル名を指定する。インタープリタとして UNIX の標準シェルである Bourne shell を利用することによって，フロー制御機能，パイプ，リダイレクション，変数などプログラムを記述する上で有効な機能を利用することができる。また必要に応じて awk, sed といった UNIX 上で動作するテキスト処理言語と連携できる。紫の一般的なコマンド形式は以下である。$は UNIX 標準シェルプロンプトを示す。

```
$ command [options] 文脈情報 filename ....
```

文書は，その目的に応じた文書体裁が適用可能でなければならない。つまり，「文書の論理操作の柔軟性」

(1 万葉集の額田王の歌「あかねさす紫野行き標野行き野守は見ずや君が袖振る」に由来する。

(2 1993 年 9 月に文書操作コマンドセットとして製品リリース。

(3 1992 年 10 月に UNIX 版 *Akane* 第 1 版を製品リリース。*Akane* は，章節単位の編集機能や論理的視点からのビューイング等の特徴を持つ。本稿は *Akane* で作成した。

(4 1995 年 1 月に Windows 3.1 版 *Akane* を製品リリース。

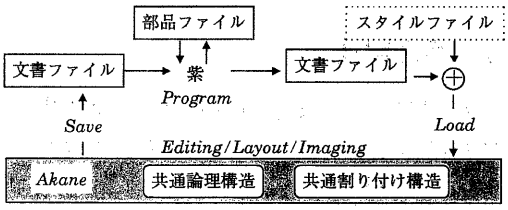


図.1　システムの概観

と同時に「文書レイアウトの柔軟性」も求められる。

木構造の論理構造は，フォーマッタによる自動レイアウト[11]により，レイアウト情報を文書内容に混在させないで別に管理できる。*Akane* では文書ファイルと独立したスタイルファイル内にレイアウト情報を保存できる。文書ファイル内には，文書プロファイルと特定論理構造と文書内容を必須として保存する。*Akane* はスタイルファイルを文書へ適用し，共通割り付け構造に従った特定割り付け構造を生成する。例えばスタイル情報から版面を1段か2段に制御する。

Akane は共通論理構造と共通割り付け構造をそれぞれ1つだけ規定している。ユーザ定義の新たな共通構造は定義できない。付録の図.5と図.6に *Akane* の共通論理構造を示す。

紫で操作し出力された文書が，*Akane* が持つ共通論理構造を満たす文書であるとき(5)，*Akane* においてレイアウト，編集可能な文書である。紫と *Akane* を合わせたシステムは，論理構造操作の柔軟性と文書レイアウトの柔軟性を兼ね備えている。システムの概観を図.1に示す。

3　操作機能

対話型編集システムでは，編集要素として「操作対象範囲の指定」と「編集操作」がある[13]。紫は自動編集を目的とするが，要求される機能は対話型編集と同様である。

3.1　操作対象

操作対象は，文書，論理構造の一部である文書部品とオブジェクト属性と属性値である。

a.　文書と文書部品

ODAの論理構造は，入次数0のオブジェクトである根(Root)が1個だけ存在し，残りすべてのオブジェクトの入次数が1の根付き有向木である。またオブジェクトは，それぞれ正整数 $1, 2, ..., i$ の標識付きの順序木である。あるオブジェクトを根とする文書の部分木を p とする。p は，根から辿れる出力次数0までの端点までである。また p の線形リスト $<p_1, p_2, ..., p_i>$ を森 $\boldsymbol{P}$ とする。p または森 $\boldsymbol{P}$ を文書部品と定義する。紫は文書部品を出力する。文書部品ファイルを木や森として保存することにより，文書の部分的な論理構造を保存するメリットがある。

b.　オブジェクト属性

文書，または文書部品を構成するオブジェクト内部の属性であるオブジェクト属性と，その値であるオブジェクト属性値が操作対象である。典型的な属性値として文書内容がある。

3.2　編集機能

copying, *deleting*, *creating*, *changing* が，自動編集のための基本操作である[13]。表.1にコマンドを示す。

a.　*copying*

条件に照合する文書部品またはオブジェクト属性値を検索し，複写する機能である。検索対象は，文書または文書部品である。例えば条件が段落(Type=Txt)であったとき，以下が複写の内容である。頂点である識別子"[3]"を付与して個々の木を区別し，1つのファイル内に森を表現する。

```
Basic[3]{Type=Txt;}
Char[3 0]{CntInfo='見出しの最初の段落です。
この行は，最初の段落に含まれます。';}
Basic[3]{Type=Txt;}
Char[3 0]{CntInfo='2番目の段落です。';}
```

条件に適合する木が複数存在し，ある2つ以上の木について包含関係にあるとき，最も大きい木のみを複写する。また識別子を対象とする演算コマンドとの連携のため，識別子のみを複写するオプションもある。

b.　*deleting*

条件に照合する文書部品，またはオブジェクト属性を検索し，削除する機能である。検索対象は文書または文書部品である。結果として文書または文書部品を出力する。オブジェクト内部の属性に空の属性値を代入する記述は，属性の削除を意味する。

(5 *Akane* の文書ファイルをチェックするコマンド[12]を使用する。これは論理構造から文書属性値にいたるまで不正箇所をチェックできる。

表.1 *紫主要コマンド一覧*

操作	対象	コマンド名	機　能	条件
copying	文書部品	***dselect***	検索& 複写	文脈情報
		iselect		識別子
	文字内容	***cget***		文脈情報
	属性値	***aget***		文脈情報
deleting	文書部品	***dremove***	検索& 削除	文脈情報
	文字内容	***cput***		文脈情報
	属性値	***aput***		文脈情報
creating	文書部品	***dinsert***	検索& 挿入	文脈情報
		iinsert		識別子
	文書	***gchapter***	作成	-
changing	文字内容	***cput***	検索& 変更	文脈情報
	属性値	***aput***		文脈情報
operation	識別子	***loperate***	演算	-

c. *creating*

条件に照合する箇所に文書部品またはオブジェクト属性と属性値を挿入する機能である。検索対象は，文書または文書部品である。結果として文書または文書部品を出力する。またテキストファイルをもとに見出し，段落を持つ文書を生成する機能がある。

d. *changing*

条件に照合する文書部品またはオブジェクト属性値を検索し，オブジェクト内部の属性に空でない属性値を代入する操作である。検索対象は，文書または文書部品である。文書または文書部品を出力する。

3.3 その他の機能

a. 識別子を対象とした演算機能

図.7 に示すように *CForm* では，すべてのオブジェクトに識別子が文書内で一意に付与される。ある条件に照合する文書部品の全識別子のリストを l とすると，識別子列 l を対象に以下の集合演算を定義できる。

$l_1, l_2, ..., l_n$ の合併(和集合) $l_1 \cup l_2 \cup ,..., \cup l_n$

$l_1, l_2, ..., l_n$ の共通(積集合) $l_1 \cap l_2 \cap ,..., \cap l_n$

$l_1, l_2, ..., l_n$ の差(差集合) $l_1 \setminus l_2 \setminus ,..., \setminus l_n$

また識別子は，10 進分類法で付与されるので，あるオブジェクト識別子は，頂点からの位置すなわち木構造における関係情報を表す。この情報を利用した直系関係や兄弟関係の問い合わせの演算機能[6]がある。識別子列の演算結果を条件に *copying* や *deleting*, *creating* 機能コマンドと連携することによって，文書部品を複写，削除，挿入することができる。

4 条件記述

文脈情報によって文書部品を特定し，文脈情報と属性操作の記述によってオブジェクト属性を特定する。

4.1 文脈情報

a. オブジェクトの指定

あるオブジェクトを指定するために，オブジェクト内部の名札情報である 1 個のラベルを記述する。ラベルは，オブジェクトタイプ，属性名，属性値などオブジェクトを特徴づける任意のものを利用できる。

b. オブジェクト間の関係指定

直接の親子関係述語 *Produce* と子孫関係述語 *Descendant*，直接の兄弟の関係述語 *Next* と親を同じくする兄弟関係述語 *brother* を定義する。これらを使用したオブジェクトの関係情報を条件として与える。

c. 操作対象の指定

文脈情報の中で検索条件と操作対象とを同時に指定できる。文脈情報の中から操作の対象となる木の頂点を選択する述語 *focus* を定義する。例えば，文脈情報「見出し番号が 1 である見出しの先頭段落において，操作対象は先頭段落」の記述は以下である(図.2 参照)。

```
Produce (Ttl, TtlNo)
Next (Ttl, Txt)
Produce (TtlNo, 1)
focus (Txt)
```

4.2 オブジェクト属性操作の記述

a. 属性指定

オブジェクトの属性値の代入 *assign* を定義する。ここで階層を持つ属性操作のため，述語 *sub* を定義する。

上の例の段落オブジェクトが持つの属性 x 内部の属性 y の値を z とする記述は，以下である(図.2 参照)。

```
focus (Txt)
sub (x, y)
assign (y, z)
```

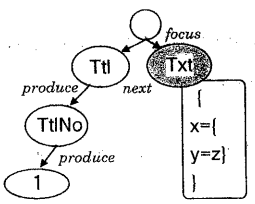


図.2 属性操作例 1

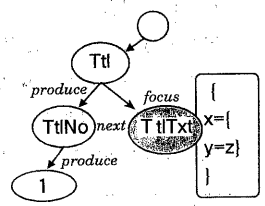


図.3 属性操作例 2

表.2 条件記述のための表記と意味

意味	表記	意味	表記
produce(a, b)	a / b	*any object*	.
next(a,b)	a # b	*empty object*	$
descendant(a,b)	a @ b	*focus*(a)	^a
brother(a,b)	a + b	*content*(a)	" a "
sub(a, b)	a . b	*assign*(a, b)	a = b

4.3 条件記述の表記

コマンド引数として条件を簡潔に指定できるように述語をシンボル化する(表.2 参照)。文脈情報とオブジェクト属性の指定記述は,区別のため ***separator*** としてシンボル「:」を定義する。また便宜上「"」内は,文字内容を表す。以下は図.2 の操作のシェルスクリプトである。「#」,「+」の直後のオブジェクトは,文脈情報の先頭オブジェクトが兄であることを表す。

```
$aput 'Ttl/TtlNo/"1"#^Txt:x.y=z' docname
```

文脈情報の先頭オブジェクトが兄でない場合,該当する兄オブジェクトの直前に括弧を置いて示す。以下は図.3 の操作のシェルスクリプトである。

```
$aput 'Ttl/(TtlNo/"1"#^TtlTxt):x.y=z' docname
```

a. 文脈情報の共有化

汎用的な条件式を定義,登録し,ライブラリとして利用できる。文脈情報は,引数部{?}を持つ key で呼び出す。例えば key は以下に示す「*」行のように自然言語で登録しておく。実行時に key に対応する文脈情報がコマンドに適用される。これにより,汎用部品を指定する文脈情報の共有と,理解容易な記述ができる。

```
*見出し{?}直下の全段落
NoSeg/(Ttl/TtlTxt/"?"+^Txt)
```

5 システム作成例

5.1 文書設計

文書の自動編集システム案件に対する一般的な開発の流れを図.4 に示す。以下シェード部分を説明する。

簡単な例として試験問題作成システムを説明する。ここでは講師が作成した解答記入済みの文書を入力して,問題用紙の作成,解答の別冊の自動作成を行なう。

まず入力文書のモデリングをする。試験問題は「番号つき問題文」,「問題説明文」,「解答部」という 3 つの

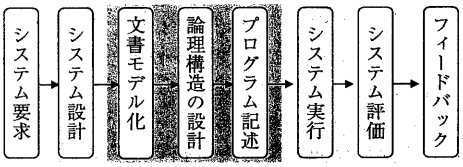


図.4 文書処理システム案件フロー

論理モジュールで構成される。

次にこの文書モデルを基に試験問題の論理構造の設計をする。「番号つき問題文」を「見出し情報」とする。「問題説明文」を「通常段落」とする。「解答部」を「行内テキスト枠内の段落」とする。

講師は,この共通論理構造を持つ文書で試験問題を作成し,解答を記入しておく。以下に Bourne シェルをインタプリタとして呼び出す文書操作スクリプト例を示す。ここでは英語の試験問題を対象とする。

5.2 削除&変更例

a. 試験用紙の作成

入力文書から,解答部分である行内テキスト枠内容を削除する。ライブラリを呼び出すオプション-L を使用して key「行内テキスト枠内容を{?}」を指定する。

```
#!/bin/sh -e
cput -L '行内テキスト枠内容を{}' Eng_test.source > Eng_test
exit 0
```

ライブラリ内の定義にしたがって,コマンドは対応する条件 Txt/Char/Fig/ArtTxt/^Char:?を適用する。変数部分「?」の値が空なので,解答内容を削除する。

```
*行内テキスト枠内容を{?}
Txt/Char/Fig/ArtTxt/^Char:?
```

行内テキスト枠は,内容部の下に構造を持つ。これは ODA の拡張である入れ子枠[14]で実現している[6]。入れ子枠は,内容部内に下位構造を持つ部分構造を埋め込み,複数の内容体系の融合を可能にする。入れ子枠も文脈情報を使って記述することができる。

b. 解答別冊の作成

```
#!/bin/sh -e
cput -e 'NoSeg/Ttl/TtlTxt/Char:' Eng_test.source | \
dremove '^Type=Txt/Char/$' - > Eng_answer
exit 0
```

(6 *Akane* の表も,この入れ子枠で実現している[15]。

問題内容を空欄に変更し，標準出力する(2行目)。

行内枠を持たない，すなわち解答部を持たない段落(説明文)を削除する(3行目)。

5.3 作成&複写例

a. 特定の問題を集めた試験問題の作成

```
#!/bin/sh -e
echo '単語にスペル"ther"を含む発音の正誤問題。' | \
gchapter -p - > new_doc
dselect 'NoSeg/(Ttl/TtlTxt/"発音"+\
^NoSeg/Txt@"ther")' Eng_test.source |\
dinsert -y Txt new_doc > result.doc
exit 0
```

文字列を標準出力する(2行目)。

文字列を段落に持つ文書を作成する(3行目)。

発音問題に含まれ，スペル"ther"を含む問題を検索し，文書部品を複写して標準出力する(4, 5行目)。

文書部品を3行目で作成した文書へ挿入し結果をファイルとして格納する(6行目)。

6 評価

6.1 性能

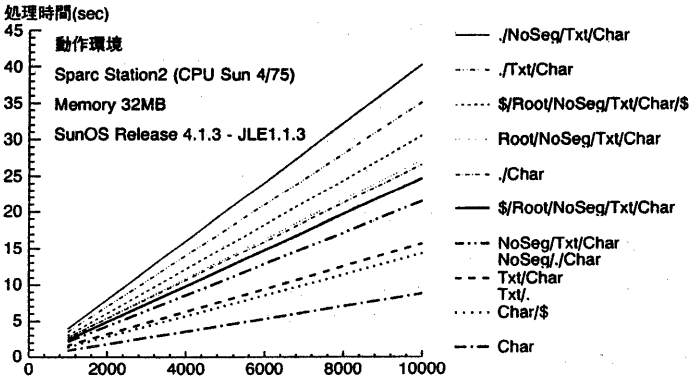
いくつかの文脈情報を与えて処理速度を実測する。ここでは1つの見出しとその直下にn個の段落を持つスタイル属性のないテスト用文書を使用した。1段落に128byte長の文字列を含む。1,000個の段落文書で180KB, 10,000個の段落文書で1.8MBのサイズとなる。dselectコマンドを対象に，内容部を検索する条件式をいくつか適用した。時間はUNIXのtimeコマンドで計測し，コマンドでの経過時間についてグラフ.1にまとめた。本稿(約136KB)を対象に測定すると，800個の段落を持つテスト文書相当の結果となった。

処理時間は，文書サイズ，条件のオブジェクト数(深さ）に比例して大きくなる。この他に条件の指定方法によっても異なる。例えば頂点のオブジェクトを任意指定とすると，オブジェクトの候補が絞れないので，速度低下の原因となる。逆に$/Root/NoSeg/Txt/Charのように親オブジェクトがない根であることを明示すると，検索候補が限定され，処理速度が向上する。

紫の検索アルゴリズムは，単純なバックトラック方式である。条件式に依存するバックトラックの頻度や検索空間に関係する。特に条件の頂点については，一意に決定できるオブジェクトのラベルがあるならば，それを陽に指定するのが高速となる。

実験スクリプト

```
$/usr/bin/time dselect 検索条件 test_doc_N > result_N
```



グラフ.1 検索条件式と処理時間の関係 (横軸は文書の大きさ-段落数-を表す)

6.2 システム比較

以前われわれが提案したシステム[6]の操作例の一部を以下に示す。このスクリプトは，章の見出し文字列に 'はじめに' を含み，かつ最下位であるような章を文書部品として新規の文書へ挿入するものである。

```
%(select_type -e 'Numbered Segment' test.doc ; \
(select_content -p 'はじめに' test.doc; \
select_type -e 'Title' test.doc) | \
below) | \
above | \
bottom | extract -s - test.doc | \
insert -yc -p '3' - skeleton.doc > result.doc
```

このシステムは，オブジェクト間の構造上の関係がわかりにくい。また実行プロセスがオブジェクト検索とオブジェクトの関係演算と分かれて発生するので，効率が悪い。本稿で紹介した紫は，記述面と実行の効率面で向上した。

6.3 実装

紫は文書ファイルを操作の対象としたことにより，エディタの実装と独立している。機能拡張が頻繁なエディタの影響が少ないので保守性が良い。

紫はコマンド方式により，基本機能のみを提供する。コマンドの連携によ

って，他のテキスト処理言語とを併用し，目的の文書操作を実現する。これは，性能面で不利であるが，市場の多様な文書操作の要求に応えられる。

紫のコマンド間プロトコルとしてテキストデータ，文書，文書部品，識別子列を規定した。このプロトコルを順守すれば，新規のコマンドを追加しても，従来のコマンドに影響しない拡張容易なシステムである。

6.4 使用実績

システムエンジニアの受注書と，表ソフトやデーターベースのデータと連係する見積り仕様書作成支援システムに導入された。このシステムにより，仕様書のページ数で約3分の1の分量の自動化を達成できた[16][17]。この他にCASEと連動した文書作成システム，試験問題作成システムに導入されている。

7 おわりに

文書の再利用を目的に，根付き有向木の構造化文書を自動編集する *copying, deleting, changing, creating* 機能のコマンド群を開発した。文書部品，内容，属性といった操作対象の領域を指定するとき，文脈情報を利用できる。文脈情報は，コマンド形式に見合う簡潔な表記方式を採用した。しかし，条件式記述には，共通論理構造と文書フォーマット上での表記に関する知識が必要である。そこで理解容易なキーで呼び出せるように条件式を隠蔽化，共有化できるようにした。今後は，条件式の入力支援や，処理性能向上といった拡張が考えられる。

謝辞

紫を設計，実装した富士ゼロックス(株)ドキュメントシステム開発センター古城慎太郎氏，松本天氏に感謝します。*Akane* の実装について説明頂いた富士ゼロックス情報システム(株)甲谷嘉英氏に感謝します。紫の開発，議論に係わったすべての方に感謝します。

参考文献

[1] Jacques André, Richard Furuta, and Vincent Quint: Structured Documents, Cambridge University Press, 1988.

[2] Richard Furuta, Vincent Quint, and Jacques André: Interactively Editing Structured Documents, Electoronic Publishing, 1(1), 19-44, 1988.

[3] ISO: Information Processing - Text and Office Systems - Open Document Architecture(ODA) and Interchange Format, ISO/IEC 8613, 1989.

[4] ISO: Information Processing - Text and Office Systems -Standard Generalized Markup Language(SGML), ISO/IEC 8879, 1986.

[5] David M. Levy: Document Reuse and Document Systems, Electoronic Publishing, Vol. 6(4), 339-348, (DECEMBER 1993).

[6] 中津山恒，楠本浩二，村田真: 構造化文書の文脈情報に基づく文書操作システム，情報処理学会テクニカルコミュニケーション研究会報告 '93 Jan.

[7] Ian A. Macleod: A Query Language for Retrieving Information from Hierarchic Text Structures. The Computer Journal, Vol. 34, No.3: 254-264, 1991.

[8] 高松忍，西田富士夫: 見出し情報を用いたテキスト解析と情報抽出，情報処理学会論文誌 Vol.29 No.8 Aug. 1988.

[9] 屋内恭輔，保科孝之，田口安男，小林晴法，西田賢一，黒澤宏: 構造化ドキュメントエディタ Akane, Fuji Xerox Technical Report, P98-105 No.8, 1993.

[10] Fuji Xerox: Akane 文書ファイルフォーマット説明書-CForm 説明書-1993.

[11] Richard Furuta, Jeffrey Scofield, and Alan Shaw, Document formatting systems: Survey, concepts, and issues. ACM Computing Surveys, 14(3):417-472, 1982.

[12] Fuji Xerox: CFormCheckTool ユーザーズガイド,1993.

[13] N. Meyrowitz, A. van Dam, Interactive Editing Systems: Part I . ACM Computing Surveys, 14(3):321-352, 1982.

[14] 村田 真，林 浩一，磯部 俊哉: 構造化文書体系(ODA)の拡張: Nested Content, 情報処理学会第46回全国大会論文集, 1992.

[15] 田口 安男: 構造化ドキュメントエデイター Akane における表内容体系，電子情報通信学会技報 OFS93-42(1994-03).

[16] 門馬敦仁，竹岡誠，陌間端: 構造化文書体系を用いた見積り仕様書作成支援システム，情報処理学会第46回全国大会論文集

[17] 畑中正明: 構造化文書体系を用いた見積り仕様書作成支援システムにおけるいちスプレッドシートインターフェース，情報処理学会第46回全国大会論文集

[18] ISO/IEC ISP 10610-1: 1992 FOD11

[19] ISO/IEC ISP 11181-1: 1992 FOD26

[20] ISO/IEC ISP 11182-1: 1992 FOD36

[21] Microsoft Corporation: Object Linking and Embedding: OLE 2.0 Design Specification, 1993.

[22] ISO: Information Processing Systems - Computer Graphics Metafile for Transfer & Storage of Picture Description Information, ISO/IEC 8632, 1987.

付録

ここでは、*Akane* が扱う共通論理構造と文書ファイルについて簡単に補足説明する。ISO は, ODA 文書の交換を容易にするため, 限定した機能のみを許す構造を国際標準プロフィルとして制定している。扱える構造と機能によって, 単純なレベル 1 から高度なレベル 3 のプロフィルがある。レベル 1 は 6 種類, レベル 2 は 18 種類, レベル 3 は 45 種類の論理オブジェクトを定義する。レベル 1 [18]は, G4FAX とのデータ互換を前提にした単純な構造のみを許すプロフィルである。レベル 2 [19]は, 文字内容のほか, 幾何学図形, ラスタ図形, 番号付き章節構造, 脚注参照構造を持ち, さらに章節番号, 脚注番号の自動生成が可能となる。レベル 3 [20]には, レベル 2 に箇条書きリスト, キャプション付きの図, テーブル, フォームを加える。

Akane は, 国際標準プロフィルレベル 2 相当の機能を持ち, レベル 3 のキャプション付きの図の構造と機能を実現する。扱う論理オブジェクトは、以下の 24 種類である。なお OLE[21]枠は, Windows 3.1 版 *Akane* から導入された。

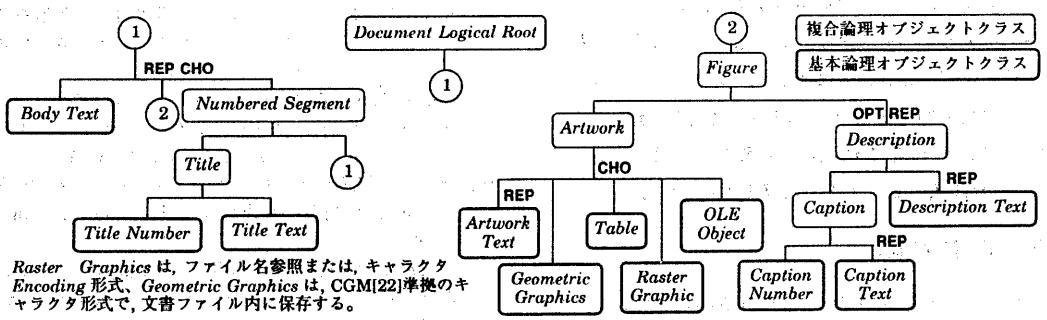


図.5 *Akane* が扱う共通論理構造 -文書構造-

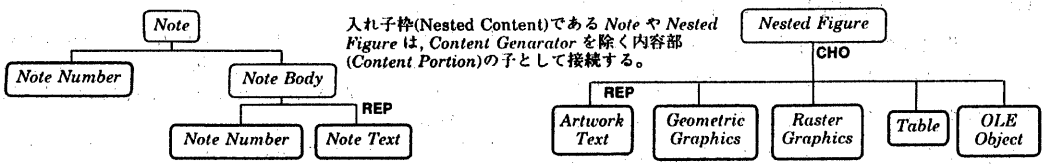


図.6 *Akane* が扱う共通論理構造 -Nested Content-

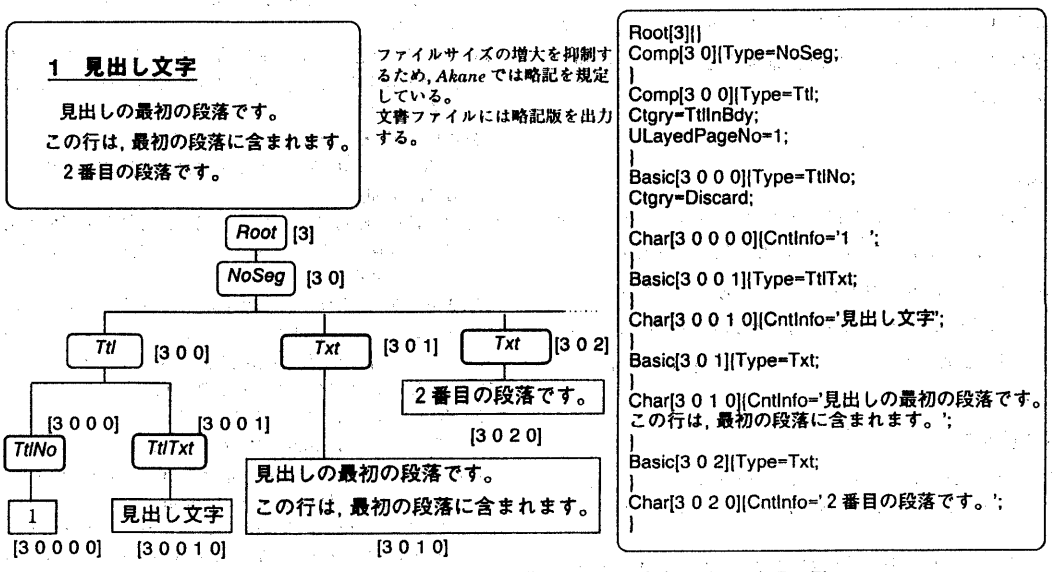


図.7 見出しと本文段落の特定論理構造、*CForm* 内容、ビュー表示の例