解　説

文書記述言語の標準化動向—III

国際規格；文書スタイル意味指定言語 (DSSSL) の概要†

安　達　　淳††

1. はじめに

文書スタイル意味指定言語 (ISO/IEC CD10179: Document Style Sematics and Specification Language: DSSSL)[1] は，構造化された文書を対象とした処理系において，処理方法を標準化された形式で指定するための言語である．

本著は，上記の一文を理解してもらうためのものである．説明の必要のありそうなキーワードとしては，「構造化された文書」(以降，標準化文書と称する)，「処理系 (および処理)」，「標準化された形式」の三つであろう．

本著での中心となるのは，最後の「標準化された形式」だが，その理解を深めてもらうために，前者二つについても概要を述べておく．

1.1 構造化文書とは

本著を例にとろう．

本著は，まず表題から始まり，著者名所属が続き，本文として 1., 1.1, … と続いている．さらに，本文の後には，情報処理学会の規定に従い，謝辞，参考文献，付録が続くかもしれない．これらの項目の並びは，解説記事や，論文では共通の形式であるが，少なくとも年賀状の形式とは異なっている．

さらに詳細に見ていくと，これらの内容項目は，単に順番になっているだけでなく 1 章の中に 1 節，2 節があるように，入れ子構造になっているのが分かる．このように，文書に自然に備わっている構造を文書構造と呼ぶ．

文書構造を規定すると二つの利点が生じる．まず，文書を書く場合に，必要最低限の形式を整えることができる．つまり，書く内容に項目単位での洩れや，重複がなくなる．また，文書の内容を参照する場合に，1 章 1 節の最初から何文字目といった形式で間違いなく参照できるので，校正などのときに指示がしやすくなるだろう．

さらに，文書をコンピュータ上で処理することを考えると，内容を理解しなくても文書構造が分かるようになっていると都合が良い．そのために，文書構造を示す目印をコンピュータで処理しやすい形で付与した文書を構造化文書と呼ぶ．

そして，そのような構造化文書を処理するシステムが増えた結果，構造化文書を異なるシステムとの間でも，交換を可能にする仕組みとして ISO/IEC 8879 (Standard Generalized Markup Language; SGML)[2] が制定されている．

SGML では，文書の構造を定義する部分を文書型定義，各文書構造を示す目印をタグ，また，タグにより示された文書構造の一要素をエレメントと呼んでいる．以降，説明のためこれらの用語をもちいる．

1.2 構造化文書を対象とする処理

1.1 で述べたように，構造化文書によりコンピュータでの文書処理が容易になった．さらに，SGML 規格により，構造化文書を交換することが可能となった．

しかし，これまでは文書のフォーマットの指定など，交換した文書をどのように処理するかについての指定は，システムごとに異なっていた．そのため，交換した文書の処理の指定は無視されるか，もしくは，受け手がそのローカルな環境に合わせて変更を行う必要があった．

DSSSL は，バッチ形式の処理系において，その処理の方法を標準化された形式で指定するための規格である．第一義的には，フォーマタに対して文書表示／印刷のスタイルを指定するための規格

† Overview on International Standard; Document Style Semantics and Specification Language (DSSSL) by Jun ADACHI (Oki Electric Industry Co. Ltd., Computer Systems Division, Software Development Dept.).

†† 沖電気工業(株)コンピュータシステム開発本部ソフトウェア開発第2部開発第2課

である．しかし，単にスタイルの指定だけに終わることなく，フォーマタの処理を構成する文書構成の変更処理，記述形式の変換処理を独立して使用することも考慮されている．

1.2.1 文書構成の変更処理

文書構成の変更処理の例を以下に示す．

例1：複数のバージョンを含む文書から特定のバージョンの文書を取り出す処理．

例2：参考文献を示す情報を集めて参考文献リストを作成するのに必要なエレメントを生成する処理．

例3：文中の注釈をまとめて巻末に移動したりする処理．

文書構成の変更処理は，入力 SGML 文書を指定に応じて出力形式の SGML 文書に変換する．単独で，もしくは，記述形式の変換処理と組み合わせて，処理を行う．

この処理は，これまで部分的には SGML のリンク機能を使って行われてきた．DSSSL は，このリンク機能を拡張し，より複雑な変換を指定できるようになっている．

1.2.2 記述形式の変換処理

記述形式の変換処理の例を以下に示す．

例1：文書をフォーマットし，標準ページ記述言語 (ISO/IEC DIS 10180 : Standard Page Description Language ; SPDL)[3] などの形式に変換する処理．

例2：文書をデータベースに格納しやすい形式に変換して，格納する．または，そのように格納された文書を取り出す．

この処理は，文書構成の変更処理と必ずしも切り離して考えることはできない．例1の，文書をフォーマットする処理では，それ以前にフォーマットに必要な情報が揃っている必要があるが，それは，構成を変更する処理を通して初めて得られる場合が多い．

1.2.3 DSSSL による利点

フォーマティングを例にとると，DSSSL が処理の指定を標準化したことで，ユーザレベルでは以下のような利点が生じる．

- 送り手は，意図したスタイルの文書をそのままのスタイルで送ることができる．
- 受け手は，フォーマティングを行うためにローカルな環境に合わせてスタイルの指定を調整する必要がなくなる．
- 一度作成したスタイル指定は再利用可能．しかも，従来の WYSIWYG ワープロなどと異なり，文字属性に至るまで処理系が自動的に付与可能である．
- プロの作ったスタイル指定を用いることで，だれでも一定の品質の文書を作成できる．

2. DSSSL の処理モデル

DSSSL では，前章の構成変換処理のことを一般言語変換プロセス，記述形式変換処理のことを意味固有プロセスと呼んでいる．

DSSSL のアプリケーションは，これらの処理の自由な組合せで構成できる．DSSSL の規格書にある Fig.2—DSSSL の概念モデル—を多少変更し図-1 に示す．

この図のうち，DSSSL が規定しているのは，一般言語変換プロセス，意味固有プロセスの入力

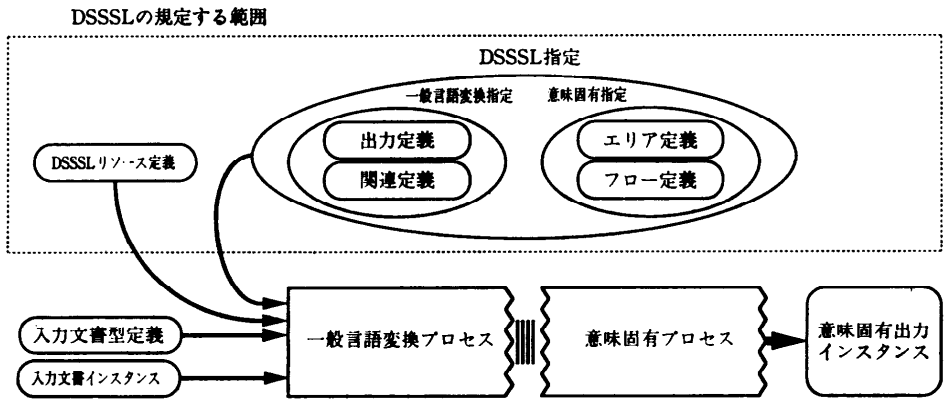

図-1 DSSSL 処理系のモデル

となる処理指定である．一般言語変換プロセスと意味固有プロセスの間の中間生成物や，入力文書の形式そのものは，DSSSL の規定の範囲外である．

以下に，これらのプロセスの詳しい説明と，それらが用いる入力情報の概略を示す．

2.1 一般言語変換プロセス

一般言語変換プロセスは，入力文書を内部処理で用いる言語に変換する処理であって，その過程で入力文書の構成を変えたり，新たに情報を追加したりする．

一般言語変換プロセスが，入力として処理する情報は，入力文書型定義，入力文書インスタンス，DSSSL リソース定義，一般言語変換指定である．

入力文書インスタンスは，文書型定義で定義された構造をもつ SGML 文書インスタンスである．

現在のところ，入力文書形式としてサポートされているのは SGML だけであるが，ISO/IEC 8613 (Office Document Architecture; ODA)[4] で定められている形式のドキュメントをパート 5 で規定されている Office Document Language (ODL) を介して入力とすることも検討中である．

DSSSL リソース定義は，DSSSL 指定の環境を変更するために用いられ，各種の指定を行う．

一般言語変換指定は，関数定義と，出力定義からなり，一般言語変換プロセスで行う変換処理を指定するものである．出力定義で定義される出力は，概念的なものであり，処理系に依存する．ある処理系では SGML の文書インスタンスを実際に生成するが，一般言語変換プロセスと意味固有プロセスを一つの処理にまとめた処理系の場合では，メモリ，一時ファイルなどの記憶域に非標準的な形でのみ存在する場合もある．

3. DSSSL 指定の概要

これ以降，DSSSL の指定を概観するが，DSSSL は今だ開発中の言語であるので，読者が実際に見るものは，これとはかなり異なる可能性もある．ここでは，DSSSL 指定でどのような情報を指定するかを汲み取っていただきたい．

3.1 共通記法

各指定で共通の記法としては，

- 位置モデル (Location Model)，
- アクセスモデル (Access Model)，
- 表現言語 (Expression Language)，
- 制約 (Constraint)

がある．

位置モデル・アクセスモデルは，入力インスタンス，出力インスタンス，意味固有出力インスタンスの中の特定のエレメントを指定するための記法である．

表現言語は，DSSSL で取り扱うデータタイプに対する演算，処理系組込みの関数などを用いるためのものである．

制約は，一連の指定に対して評価値を与える方法を規定するものであって，フォーマット処理のように複雑な処理の指定を可能とするものである．

3.1.1 位置モデル

位置モデルは，入力インスタンス，出力インスタンス，および，意味固有出力インスタンス内のエレメントの位置を記述する記法で，条件に適合するエレメントの集合を選択するための機構である．選択されたエレメントは，関数の適用により情報を抽出したり，関連の指定などに用いられる．

位置モデルが選択するのは，特定の実体，文書インスタンスのエレメント，エレメント内容のいずれかである．それぞれを選択するような，位置モデルの例を以下に示す．

a） 実体参照の例

```
ENTITY : dsssl
```

この例では．dsssl という参照名で定義された実体を参照しており，(おそらく) Document Style Semantics and Specification Language という正式名称に置き換えられる．

b） エレメントの選択の例

```
paragraph [[PARENT : note|
                PARENT : list-item]]
```

この例では，エレメント名が paragraph のエレメントであり，親エレメントが note である，親エレメントが list-item である，のいずれかのクォリファイア（資格要件）をみたすものを選択している．

ここで用いるクォリファイアには，親，子，上，下，直前，直後，前，後，属性，カウント，および，アンド，オア，ノットなどの論理演算で結んだ形式のものがある．

c） エレメント内容の選択の例

DATA : 18, 20 [PARENT : note]

この例では，note の中に含まれるキャラクタデータの 18 番目から 20 番目までを選択している．

エレメント内容の種別を示す先頭のタグとしては，DATA（通常のキャラクタデータ），NDATA（非 SGML データ），PI（処理命令）がある．

ここで用いるクォリファイアには，親，前，後，論理演算の位置に関するものと，文字概念そのものを示すジェネリックキャラクタのプロパティに関するものがある．

3.1.2 アクセスモデル

アクセスモデルは，入力文書インスタンス，出力文書インスタンス，意味固有出力インスタンスで適用可能なもので，現在処理中のエレメントを基準に周囲のエレメントなどにアクセスするためのものである．

アクセスモデルは，構文木の指定と，現在位置の一時的な変更と，クォリファイアからなる．以下に例を示す．

a） アクセスモデルの例

Root (Elder:) : SSRI : Child [Self : note]

この例では，意味固有出力インスタンスの構文木の note という名前の甥に当たるノードを選択している．（意味固有出力インスタンスの現在位置を一時的に兄ノードに変更して，その子ノードであって，自身の名前が note であるようなノードを選択している．）

アクセスモデルで使用可能なクォリファイアには，親，子，兄，弟，上，下，前，後，属性，プロパティ，データ内容などがある．

3.1.3 表現言語

表現言語は，DSSSL が標準に備えているデータタイプと，それらを扱う関数の組合せで構成された言語である．

以下に表現言語の例を示す．

a） 数値表現

1.5（数値）

1.2 cm（距離）

30 deg（角度）

b） 文字列表現

substr (“abcdefg”, 3, 2)

これは部分文字列を取り出す関数で，3 文字目から 2 文字を取り出す．この例では，“cd” が返る．

c） 選択表現

select (when (条件) value 値
otherwise value 値)

条件に適合する値を選択する表現で，入れ子構造をとることが可能である．

d） オブジェクト

規格書では，データタイプとしては扱われていないが，位置モデル・アクセスモデルで選択されるエレメント・属性・内容なども表現言語で取り扱う．

表-1 に，データタイプとそれらを扱う関数の一覧を示す．

3.1.4 制 約

DSSSL は，制約言語の一種である．アサインメントや，マッピングはあっても代入という概念はない．また，アクセスモデルや，表現言語では，まだ存在しないエレメントを参照することができる．DSSSL の処理系では，そのような値については，参照しているエレメントが生成されるまで保留しておくような機構も備えることになるであろう．

このような，暗黙のうちに解決される制約のほかに，陽に解決法を指定する制約がある．

意味固有プロセスがフォーマタである場合，エリアの属性を記述する部分には，属性の指定と条件，および，その条件を満たした場合の評価値をもつ制約セットを指定できる．DSSSL の処理系は，全体としての評価値が，最大となるように，制約セットの中から，属性の指定を選択する．

以下にエリア属性を定義する制約セットの例を示す．

a） エリア属性を定義する制約セットの例

area-constraints :
100 area-attributes :
Align-Top＝2 cm
50 area-attributes :
Align-Bottom＝2 cm

この例では，図表用のエリアなどの場合を想定している．可能であれば，そのエリアを含むエリアの上辺に揃え，それが不可能であれば，下辺に揃えるというものである．

この例で示したように，制約はより大きな評価

表-1 データタイプと関数のリスト

データタイプ	関　数	機能概要
数　値	数学関数群	
	abs ( )	絶対値を求める.
	acos ( )	cos の逆関数.
	asin ( )	sin の逆関数.
	atan ( )	tan の逆関数.
	ceiling ( )	与えられた値を下回らない最小の整数を返す.
	exp ( )	e の n 乗を返す.
	floor ( )	与えられた値を越えない最大の整数を返す.
	square ( )	2乗数を返す.
	sqrt ( )	2乗根を返す.
	数値 −> 文字列変換関数	
	sconvert ( )	数値を指定された方法で文字列に変換する.
	数値集合関数	
	max ( )	最大値を返す.
	mean ( )	平均値を返す.
	median ( )	中央値を返す.
	min ( )	最小値を返す.
	sum ( )	合計値を返す.
	角度関数	
	cos ( )	余弦関数.
	sin ( )	正弦関数.
	tan ( )	正接関数.
文字列	delete ( )	指定された句切り子で分割された文字列の残りを返す.
	delstr ( )	文字列の指定された位置から文字数分除いたものを返す.
	extract ( )	指定された句切り子で分割された文字列の一部を返す.
	find ( )	文字列中に含まれる部分文字列の区切り子による位置を返す.
	index ( )	文字列中に含まれる部分文字列の位置を返す.
	replicate ( )	指定された文字列をくり返したものを返す.
	size ( )	文字列中の文字数を返す.
	substr ( )	文字列の指定された位置から文字数分を返す.
	words ( )	文字列中の区切り子により区切られた数を返す.
位置／アクセスモデル	locationcount ( )	モデルに適合するエレメントの数を返す.
	npropval ( )	モデルに適合するエレメントの数値プロパティを取り出す.
	spropval ( )	モデルに適合するエレメントの文字列プロパティを取り出す.
	treecount ( )	構文木中のオプションに適合するエレメントの数を返す.
任　意	select ( )	条件に適合する値を返す.

値をもつ制約から評価していき，条件を満たすものを適用する．ただし，そこで適用された制約は，より上位の制約を適用することで覆される可能性もある．

3.2 一般言語変換指定

以下に，一般言語変換で用いる変換指定について，引用の処理で用いる文書型定義と入力文書インスタンスの例を用いて述べる．SGML の文書型定義については，他所の SGML の解説に詳しく記述されているので省略する．

入力文書型定義の定義：

```
<!ELEMENT cit  - 0 (#PCDATA)>
<!ATTLIST cit author CDATA #REQUIRED
          document CDATA #REQUIRED
          publish CDATA #REQUIRED
          memo CDATA #IMPLIED >
```

入力文書インスタンスの例：

```
<cit author = "安達　淳"
    document = "文書意味指定言語の概要"
    publish = "情報処理学会出版局"
    memo = "for example">
文書スタイル意味指定言語…
</cit>
```

この入力文書型定義で記述しているのは，引用エレメントであり，付随する属性情報として引用文献の著者，文献名，出版情報，メモがある．これらの情報は，文字データであり，メモ以外は必須なものとして宣言されている．入力文書インスタンスの例は，この記事の『はじめに』，の章を引用した場合に，どのような文書インスタンスが引用エレメントとして生成されるかを示したものである．

3.2.1 出力定義

出力定義は，SGML の文書型定義の形式で記述する．

出力定義：

```
<!ELEMENT cit-refs - 0
                (cit-string, cit-mark)>
<!ELEMENT cit-string 0 0 CDATA>
<!ELEMENT cit-mark - 0 CDATA>
<!ELEMENT cit-list - 0 (cit-info*)>
<!ELEMENT cit-info 0 0
      (cit-content,
       cit-author,
       cit-document,
       cit-publish, cit-memo?)>
<!ELEMENT (cit-author|
       cit-document|
       cit-publish) - 0 CDATA>
```

出力定義では，二つのグループのエレメントを定義している．一つは，引用を行っている cit-refs で，もう一つは，引用一覧を作るための cit-list である．cit-refs は，引用文そのものを含む，cit-string と，引用番号を示す cit-mark からなる．cit-list は，引用文も含んだ各種文献情報のリストを含んでいる．

3.2.2 関連指定

関連指定は，言語変換の処理を指定するものであって，入力文書インスタンスと出力文書インスタンスの各エレメント，各属性について関連づけを行うものである．

関連づけには，単純関連，複製関連，契機関連の三つのタイプがある．

- 単純関連は，入力文書インスタンスのエレメント（属性）に，出力文書インスタンスを関連づけるものである．
- 複製関連は，入力文書インスタンスのエレメント（属性）を，索引や，引用リストなどを作成するために2カ所以上と関連づけるためのものである．
- 契機関連は，そのエレメントの内容ではなく，そのエレメントの存在自体，もしくはその場所が，問題になる場合の関連づけを行うためのものである．

複製・契機関連は，一つの入力文書インスタンスのエレメントに対して，複数もたせることができる．

また，関連づけと同時に，入力文書インスタンスのエレメントの内容に新たに内容を加えたり，出力を完全に抑制したりすることもできる．

以下に関連指定の例を示す．

a） 関連指定の例

関連指定：

```
cit/CoPoFrag
-> cit-refs/cit-string,
      textbefore : '"',
      textafter : '"'
 + cit-list/cit-info/cit-content
```

```
> cit-refs/cit-mark,
    textafter : // '*',
     sconvert(
       number-to-string,
              treecount(
         cit, psiblings, element))
ATTR : cit--author[Parent : cit]
-> cit-list/cit-info/cit-author
ATTR : cit-document[Parent : cit]
-> cit-list/cit-info/cit-document
ATTR : cit-publish[Parent : cit]
-> cit-list/cit-info/cit-publish
ATTR : cit-memo[Parent : cit]
-> IGNORE
```

出力インスタンス例：

```
<cit-refs>"文書スタイル意味指定…"
<cit-mark>＊1</cit-refs>
<cit-list>
<cit-info>
<cit-content>
  文書スタイル意味指定…
<cit-autor> 安達　淳
<cit-document>
  文書スタイル意味指定言語の概要
<cit-publish> 情報処理学会
</cit-info>
<cit-list>
```

順番に見ていくと，まず，cit の内容が文字列生成の機能を用いて生成された前後のクォーテーションマークにくくられてエレメント cit-string に関連づけられる．

次に，cit の内容が複製(＋)されて，cit-content に関連づけられる．

さらに，cit を契機(>)として cit-mark が生成され，アスタリスクと構文木上で何番目になるかを数えた（treecount）結果を文字列に変換（sconvert）して結合(//)したものを内容とする．

メモ以外の各属性は，対応するエレメントに対応づけられる．メモは，出力を抑制されている．

これらの関連づけの結果，出力インスタンス例で示されるようなインスタンスが生成される．

3.3 意味固有指定

意味固有指定は，アプリケーションに固有のものであり，別種のアプリケーションであれば当然ながら，まったく異なったものとなるだろう．

現在開発されているのは，フォーマタを想定した，エリア定義とフロー定義だけである．一般言語変換指定の説明に続いて，意味固有指定でも，引用の例を用いていく．

3.3.1 エリア定義

エリアの指定は，DSSSL のアプリケーションであるフォーマタにとって最も重要な部分である．ここで定義するのは，エリアを生成するときの型となるエリアテンプレートである．

エリアテンプレートの定義では，各エリアの属性と，そのエリアが内部に含むエリアを記述する．

以下にエリア定義の例を示す．引用の処理に必要なエリア定義をすべて記述したいが，紙幅の都合上，そのうちの一つだけを示す．

a） エリア定義の例

```
area-name : CitArea
area-type : cit-1
area-contstraints :
 100 area-attributes :
  Width =
   npropval (SSRI : Child : [Self : CitCont],
             X-extent),
  Depth =
   npropval(SSRI : Parent :, Y-extent),
  Position-Point = (0 mm,
   npropval(SSRI : Parent :, Hposy)),
  Escapement-Point = (
   npropval(SSRI :, X-extent),
   nproval(SSRI : Parent :, Hescy)),
  QualLeft,
  Placement-Path-type = horizontal-path,
  Placement-Path-Start = (0 mm,
   npropval(SSRI : Parent :, Start-Y)),
  Placement-Path-End = (
   npropval(SSRI :, X-extent),
   nproval(SSRI :, Start-Y))
area-subdiv : CitConAr, CitMarkA[1]
```

この例で記述しているのは，図-2 で示した構

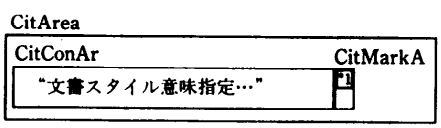

図-2 引用参照エリアのエリア階層構造図

造をもった引用エリアの一番外側にあたるエリアである．

area-name は，エリアの名前を指定するものである．area-type は，アプリケーションが必要に応じて参照するエリアタイプの指定である．続いて，制約セットによりエリアの属性を指定する．area-subdiv は，このエリアがどのようなエリアを含むかの指定で，この場合には，CitConAr と CitMarkA のエリアが内部に生成される．角括弧で括られているのは生成回数の指定であって，指定がない場合には1回以上，指定がある場合には，その指定に応じて適切な回数生成される（この場合は，一回のみ）．

3.3.2 フロー定義

フロー定義は，一般言語変換プロセスが生成した，複数のデータストリームをエリア定義で定義した，エリアテンプレートにマッピングすることで，実際のエリアの生成を指定することである．

a） フロー定義の例

```
cit-refs/cit-content -> CitArea/CitConAr
cit-refs/cit-mark -> CitArea/CitMarkA
```

この例では，出力定義で定義された各ストリームをエリア定義で定義したエリアテンプレートにマッピングしている．この例では，一つのデータストリームが一つのエリアに対応しているが，段組のフロー定義などでは，一つのデータストリームが，複数のエリアテンプレートに対応することもある．

4. DSSSL リソース

現在 DSSSL リソースとして定義されているものには，以下のものがある．紙幅の都合により，概要のみを述べる．これらのうち，導出単位，ビット-キャラクター-グリフマッピング，コレーティングシーケンス，コンバージョンタイプは，一般言語変換プロセスで用い，それ以外は，フォーマタである意味固有プロセスで用いられる．

- 導出単位

既存の単位を用いて導出単位の定義を行う．

- ビット-キャラクター-グリフマッピング

数値とジェネリックキャラクタのマッピング，および，ジェネリックキャラクタとグリフとのマッピングを指定する．

- コレーティングシーケンス

ソート処理時の順序をジェネリックキャラクタを用いて指定する．

- コンバージョンタイプ

整数を文字列に変換する方法を指定する．

- ハイフネーション辞書

ハイフネーション処理に用いる辞書名を指定する．

- ハイフネーション規則

ハイフネーション処理で用いる規則名を指定する．

- エリア分割規則

エリア分割の規則名を指定する．

- エリア調整規則

エリアの調整の規則名を指定する．複数のエリアを一定の長さにおさめるための規則である．

- エリア終了規則

エリアの終了の規則名を指定する．和文の場合には，禁則処理の指定はここで行うことになる．

- フォント名参照リスト

ISO/IEC DIS 9541-2[5] に適合したフォントセットを指定する．

- カラーモデル

ローカルなカラーと SPDL で規定されているカラーとのマッピングを指定する．

5. 日本語の処理について

日本語の処理にについて，問題となるのは，文字コードと，日本語に特有の組版である．

文字コードについては，ビット列とキャラクタとのマッピングテーブルをもっているので，日本語 EUC，MS-漢字（いわゆるシフト JIS）などのモードレスなコード系を扱うのは比較的簡単である．エスケープシーケンスを用いてコードを切り替えるコード系については，今後の検討が必要だと思われる．

日本語組版に特有なものとして，ルビ，縦中横，割注，圏点などの組版が存在する．

それら特殊な組版については，サンプルコーディングを行いながら，問題点を抽出する方式で DSSSL の表現力をチェック中である．このチェックの過程で必要性が認められ，アクセスモデルなどが新設されている．

現状では，圏点についてはチェックが済み，割注についても問題点の提出を非公式な形では終え

ている．残ったルビ，縦中横のように複雑なものの指定が可能になれば，それら以外については単に記述量の問題になるだろう．

6. 今後の動向

DSSSL は，1991年8月26日に DIS 投票を終了する．その結果を待って，さらに改良を続け，1992年12月に IS 化の予定である．

国内でも，JIS 化の作業が今年の7月には開始される予定である．他の規格と比べるとまだまだ，一般に知られているとは言えないが，JIS 規格が制定されるころには，読者諸氏にも馴染みのあるものになっていることを期待したい．

謝辞 私の拙い英語を真面目に理解し，日本語特有の規則を反映するために努力していただいている，DSSSL SWG の chair person である IBM の Sharon Adler，国内・国外を問わず SC 18/WG 8 の活動についてご教示くださっている松下電送の小町主査，および，SC 18/WG 8 国内委員会の方々に感謝する．

参 考 文 献

1) ISO/IEC CD 10179, Document Style Semantics and Specification language (1991).
2) ISO/IEC 8879, Standard Generalized Markup Language (1986).
3) ISO/IEC DIS 10180, Standard Page Description Language (1991).
4) ISO/IEC 8613, Office Document Architechture (1988).
5) ISO/IEC DIS 9541-2.2, Font information Interchange—part 2: Interchange Format (1990).

(平成3年6月18日受付)

安達 淳

1985年広島大学総合化学部総合科学科人間行動科学コース卒業．同年沖電気工業(株)入社．文書処理の開発に従事．1990年 SC 18/WG 8 国際会議初参加．以降，DSSSL 規格の開発に携わり，現在に至る．