データベースシステム 91－6
（1992. 11. 5）

HyTime: 文書の拡張としてのハイパーメディア記述言語

今郷 詔

（株）リコー 中央研究所

ISO は文書構造記述言語である SGML の考え方をハイパーメディアに拡張し、ハイパーメディアの論理構造を記述する言語として HyTime を 1992 年に制定した。

HyTime は SGML の構文を用いて、オブジェクトの様々なアドレス付けを可能にし、ハイパーリンクやオブジェクト間の同期関係の表現など、ハイパーメディアの持つ本質的な構造を記述する手段を提供する。メディアの符号化方式や表示方法、ユーザ・インタフェースなどは規定しない。

HyTime はハイパーメディア情報の交換の基盤であり、一つの情報が様々なアプリケーションから利用できるような開かれた環境の実現に役立つことが期待される。

HyTime: A hypermedia description language as extensions to documents

IMAGO Satosi

Research and Development Center, RICOH Co.,Ltd.

ISO extended the concept of SGML, a language to describe the document structures, to hypermedia and developed HyTime in 1992. Based on the SGML structures, HyTime describes the logical structures of hypermedia.

Among them are addressing schemes of objects and the intrinsic structures of hypermedia described as hyperlinks and synchronizations of objects. Encoding schemes of media and extrinsic information such as rendition and user interactions are out of its scope.

HyTime is the basis for the hypermedia information interchange and expected to contribute to realize the open environment in which different applications are able to access various information.

1 はじめに

ハイパーメディアは突然現われた新しい領域ではない。例えば本の中には、テキストや図、表などの複数のメディアが含まれていたし、図表番号や脚注などの内部世界でのリンクと参考文献という形での外部世界へのリンクも含まれていた。したがって文書の拡張としてハイパーメディアを捉えることができる [2]。

ISO では文書の論理構造を記述するための言語である **SGML** (*Standard Generalized Markup Language*) を国際規格として 1986 年に制定した [1, 3]。

論理構造という言葉はレイアウト構造に対する言葉で、ページやインデント、文字の大きさや飾りなどの文書の見せ方とは関係なく、章や段落などの文書に本質的に備わっている構造を指す。

SGML では文書を element の木構造として捕えており、element 構造をユーザが自由に定義できるため、極めて広い範囲の文書に適用することができる。SGML を使って文書を記述しておけば、様々なレイアウトで紙に印刷できるだけでなく、処理方法を変えることにより、データベースへの格納、CD-ROM への記録、機械翻訳、情報抽出などの文書を対象とするあらゆるアプリケーションで大きな力を発揮する。

HyTime (*Hypermedia/Time-based Structuring Language*) はこのような SGML の考え方をハイパーメディアに拡張し、SGML の構文を用いてハイパーメディアの論理構造を記述するための言語として開発が進められ、1992 年に国際規格となった [6, 9, 10]。

ISO ではさらに HyTime のアプリケーションとして、SMDL (*Standard Music Description Language*) という音楽記述言語も現在開発している [8]。SGML と HyTime、SMDL との現在の関係をまとめると、SGML のアプリケーションが HyTime であり、さらに HyTime のアプリケーションが SMDL であるということになる。

2 HyTime の思想

HyTime では文書を「一つの単位として識別され、人間が知覚できるような情報の集まり」と定義している。したがって、ハイパーメディアも文書の一種とみなす。

HyTime は、様々なハイパーメディア・アプリケーション同士で文書交換を可能にするための基盤となる規格である。もし文書が特定のアプリケーションでの表示や再生の方法を示すスクリプトを用いて表現されていたら、その文書を他のアプリケーションで利用することはできない。表示や再生の処理から独立した文書の本質的な構造をスクリプトから分離して HyTime で表現すれば、その文書が作り出されたアプリケーション以外からも利用が可能になる。

HyTime は一部のアプリケーションに必要とされる機能をすべてサポートするような規格ではなく、多くのアプリケーションに必要とされる機能の一部をサポートする規格である。したがって、HyTime の枠内では記述できない情報はアプリケーションが独自の構造を定義して記述しなければならない。

HyTime は、メディアの符号化方式や文書構造、DTD[1]、オブジェクトの表示・再生方法、ガイド方法などは対象としていない。これらの決定はアプリケーションに委ねられている。

ただし、HyTime とアプリケーションとの境界が明確に決まっているわけではない。情報の使われ方の予測や実行速度の問題などを考慮して、HyTime がサポートしている機能があっても、アプリケーション固有の方法で表現してもよい。標準化した方法で表現するかしないかは、アプリケーション・デザイナやユーザが判断すべきことである。

HyTime はモジュール化されているので、アプリケーション・デザイナは自分で標準に従って記述したいと思う特性に対応するモジュールだけを使うことができる。

2.1 ハイパー文書

図 1に示すように、直接あるいは web を通して間接的に結合している文書やオブジェクト全体が一つのハイパー文書である。

ここで web とは、一緒に使用されるハイパーリンクの集合を意味する。なお特にハイパーリンクと呼ぶのは SGML のリンク機構と区別するためで、一般には単にリンクとよばれているものである。

ハイパー文書の中で、そのハイパー文書に対するアクセスが始まる文書を hub document と呼ぶ。

[1] Document Type Definition

hub document は静的な存在ではなく、アプリケーション起動時に決まる動的な存在である。

web に結合されている文書やオブジェクトは必ずしもHyTime やSGML に適合している必要はなく、hub document だけがHyTime に適合していればよい。HyTime はSGML のアプリケーションであるので、HyTime に適合することはSGML に適合することも意味する。

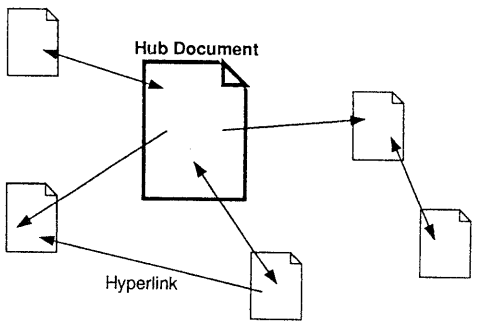

図 1: ハイパー文書の概念

2.2 SGML からの拡張

SGML は文書構造を記述する方法を規定する規格であって、文書構造そのものを規定する規格ではない。同様にHyTime はハイパーメディアの構造を記述する方法を規定する規格であって、構造そのものを規定する規格ではない。

HyTime はSGML のアプリケーションという位置付けであるが、DTD そのものを規定するのではなく、DTD の構成方法とその意味を規定している。

テキストを中心とした従来型の文書はSGML で記述できるが、それをハイパーメディアに拡張する場合に問題となる点が次の三つである。

オブジェクトの符号化方式 SGML では非テキスト・メディアの符号化方式自体は規定していないが、外部 entity としてメディアの内容と符号化方式名を組にして記述することで、画像や音声のような非テキスト・メディアを文書の一部として表現することができる。

ハイパーメディアでは様々なメディアが扱える必要があるので、メディアの符号化方式を規定することは現実的ではない。したがって、HyTime でもSGML の方式をそのまま用いることになる。

しかしこの方式では、ハイパーメディアの送り手が用いた符号化方式を受け手でも理解できないと表示や再生が行なえない。それぞれの世界で標準的な符号化方式を定めることによってこのような事態を避けることが期待される。

リンクの表現方法 リンクを表現するには、文書の内外のオブジェクトを指示するアドレス付け能力と、アドレス指定の集合を表現する手段が必要となる。

SGML ではID attribute を持つ element しかアドレス対象にできない。したがってHyTime ではアドレス付け機構の大幅な強化が必要である。アドレス付けができれば、その集合を表現する方式を定めるだけでリンクが表現できることになる。

HyTime のアドレス付け機構は一般化されており、自動的に解決できるアドレスやコンピュータ上に表現されているアドレスだけでなく、人間が介入しなければ解決できないアドレスも含んでいる。例えば、参考文献情報は自動的には解決することができないが、人間が図書館へ足を運ぶことで解決できる。このようなアドレスも扱えることは、ハイパーメディアの一般性を確保するうえで重要なことである。

オブジェクトの同期の表現方法 同期という概念はSGML にはない。HyTime ではオブジェクト間の同期を、ユーザが設定する座標空間上の位置を媒介とすることで表現する。

リンクと同期はまったく別の問題のように見える。しかし、HyTime ではこれを同じアドレス付けの問題と捉えている。すなわち、リンクはアドレス付けされたオブジェクトの集合であり、同期はアドレス付けされたオブジェクトの時間軸上への写像である。同期の機構は、時間的な位置関係だけでなく、軸を変えるだけで空間的な位置関係を現す機構となる。

3 HyTime の内容

3.1 全体構成

HyTime は図 2に示す六つのモジュールに分けられている。各モジュールは関連が深い様々な機構の

集合で、必須機構とオプション機構の二種類を含む。モジュール自体もbase module以外はオプションである。ただしモジュール間には、図2で矢印で表現した依存関係があり、矢印の根のモジュールを使用する場合は必ず矢印の先のモジュールも使用しなければならない。破線の矢印はモジュール内の特定の機構を使う場合にのみ発生する依存関係を現す。

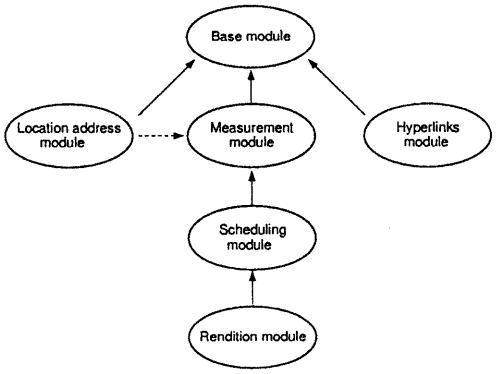

図 2: HyTime のモジュール間の依存関係

3.1.1 Architectural form

HyTime はDTD や文書構造そのものを決めているのではない。アプリケーション・デザイナが自身のDTD を作る時に基づかなければならない形式の集合を定めているだけである。この形式のことを **architectural form** と呼ぶ。

architectural form はDTD の中でelement やattribute を定義する時に基づかねばならない形式で、次の二通りがある。

element type form element type declaration とattribute definition list declaration の両方の形式を規定する。この形式に従うelement にはHyTime attribute が与えられ、その値でどのarchitectural form に対応するかを識別できるようになっている。

attribute list form attribute definition list declaration の形式のみを規定する。任意のelement type form で使用可能な形式と、指示されたいくつかのelement type form だけで使用可能な形式の二種類がある。

HyTime は architectural form によっていわばmeta-DTD を定義していることになる。アプリケーション・デザイナがそのmeta-DTD に適合するDTD を作り、そしてそのDTD に適合するインスタンスが作成される。architectural form とDTD およびインスタンスとの関係を図3に示す。

アプリケーション DTD は一般にarchitectural form に従ったelement と、アプリケーションが独自に使用するHyTime とは無関係なelement の両方を含む。また、アプリケーション独自のarchitectural form を定めてもよい。

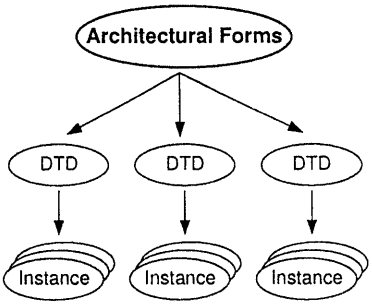

図 3: architectural form とDTD およびインスタンスとの関係

3.2 Base module

唯一の必須モジュールで、ハイパー文書の表現方法を規定している。その他に互いに独立したいくつかのオプション機構も提供している。

Hyperdocument management オブジェクトの表現・識別・アクセス方法とハイパー文書の交換フォーマットを規定している。

オブジェクトの表現とアクセス方法、内部オブジェクトの識別はSGML の規定をそのまま用いる。ハイパー文書の交換フォーマットにはISO 9069 (SDIF)[4] を、外部オブジェクトの識別はISO 9070[5] を用いる。

オブジェクトはSGML のelement あるいはentity として表現する。HyTime ではオブジェクトの符号化方式は規定せず、アプリケーションにその規定を委ねているが、内容が一般のテキストではないオブ

ジェクトは次のどれかの方法で指定することになる。

- SGML 記述による符号化
  表や数式、スプレッド・シートのデータなどはこの方法で表現できる。図 4に表の例を示す [7]。
- テキスト形式の符号化
  SGML character でオブジェクトを符号化して element の内容とする。図 5に例を示す。
- バイナリ形式の符号化
  任意の符号化を行ない、外部 entity として参照する。図 6に例を示す。

HyTime identification 文書が HyTime のインスタンスかどうか、文書中の element がどの architectural form に対応するかを識別するための機構である。

HyTime のインスタンスである文書は、対応する SGML 宣言の APPINFO パラメータに HyTime という値を与えることで識別する。

HyTime の architectural form に従っている element は、対応する DTD において HyTime という attribute を設け、その値として対応する architectural form の名前を指示しておく。例えば、HyTime のインスタンスの最上位 element には HyTime attribute に doc という値が DTD で固定される。

3.3 Measurement module

座標系を用いてオブジェクトのアドレスを表現するための基礎となる座標軸に関する機構を提供するモジュールである。座標軸上の位置を指定する機構と、座標軸の単位を指定する機構、オブジェクトの座標軸上の位置を参照するための機構を含む。このモジュールはオプションであるが、location address module の coordinate location 機構あるいは、scheduling module を使用する場合にはこのモジュールの使用が必須となる。

座標軸 HyTime は座標軸を有限で量子からなると定義している。つまり、座標軸には必ず両端が存在し、座標軸上の位置指定は必ず大きさを伴うことになる。

```
<!ELEMENT table - - (th1*, r*, tabcap)>
<!ELEMENT th1 - O ((#PCDATA|th2)*)>
<!ELEMENT th2 - O ((#PCDATA| th3)*)>
<!ELEMENT th3 - O (#PCDATA)>
<!ELEMENT r - O (c*)>
<!ELEMENT c - O (#PCDATA)>
<!ELEMENT tabcap - O (#PCDATA)>
```

DTD

```
<table id=arfhc>
<th1>Year
<th1>Annual Rainfall (in centimetres)
<th2>First Quarter <th3>Jan<th3>Feb<th3>Mar
<th2>Third Quarter <th3>Jul<th3>Aug<th3>Sep
<th2>Fourth Quarter <th3>Oct<th3>Nov<th3>Dec
<r><c>1986<c>3.8<c>3.6<c>4.4<c>5.0<c>6.0<c>6.2
<c>5.8<c>7.2<c>7.2<c>5.6<c>6.2<c>4.8<c>65.8
<r><c>1987<c>...
<tabcap>Annual Rainfall for Head City
</table>
```

インスタンス

図 4: SGML 記述による符号化

```
<!NOTATION eps SYSTEM>
<!NOTATION cgm SYSTEM>
<!ELEMENT fig - - CDATA>
<!ATTLIST fig
    dt NOTATION(eps|cgm) #REQUIRED>
```

DTD

```
<fig dt="EPS">
...
10 dict begin
/scanLine 71 string def
360.000000 216.000000 scale
...
</fig>
```

インスタンス

図 5: テキスト形式の符号化

```
<!NOTATION tiff  SYSTEM>
<!ELEMENT fig - O EMPTY>
<!ATTLIST fig
    file ENTITY #REQUIRED>
```

DTD

```
<!ENTITY foo SYSTEM "/sample/bar" NDATA tiff>
...
<fig file="foo">
```

インスタンス

図 6: バイナリ形式の符号化

位置指定　座標軸は量子の有限な順序列であるので、オブジェクトの座標軸上の位置 (dimension) を指定するには、先頭位置に対応する量子 (position) と、オブジェクトが占める範囲に対応する量子の数 (extent) の二つを規定すればよい。

座標軸上の位置は二つの数値の組で図 7に示すように指定する。この数値の意味は正負によって解釈が変わる。

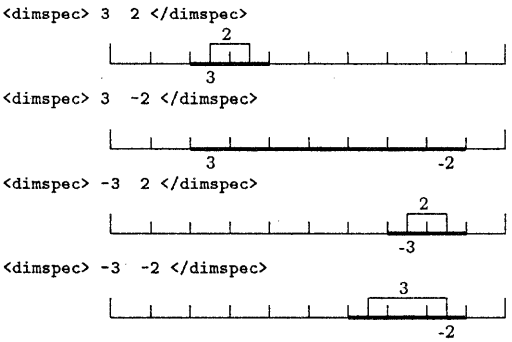

図 7: 座標軸上の位置指定

計測単位　座標軸の計測単位つまり量子の単位をどうとるかを指定する。任意に指定できるが、基本単位として次の五種類が提供されている。これらの基本単位はSMU (standard measurement unit) と呼ばれる。

- Generic quantum: 相対的な関係のみを表現するための単位
- Real time: 実時間を表現する単位 (秒)
- Real space: 実距離を表現する単位 (メートル)
- Virtual time: 仮想的な時間を表現する単位
- Virtual space: 仮想的な距離を表現する単位

実際にはSMU をそのまま扱うと不都合が多いので、SMU と正の有理数との積をとった granule を座標軸の単位として用いる。さらなる柔軟性のために、granule と正の有理数との積をとった HMU (HyTime Measurement Units) を用いることもできる。

位置参照　オブジェクトの座標軸上の位置を次の観点で参照する。

- 座標軸上の先頭位置
- 座標軸上の末尾位置
- 座標軸上の量子の数

あるオブジェクトの位置をこの機構を用いて参照し、その位置の関数として別のオブジェクトの位置を決定することで、オブジェクトの同期や位置合わせを表現することができる。

3.4 Location address module

アドレスを表現する基本的な SGML の形式は、element に与えられた ID attribute である。ID attribute が与えられている element は IDREF attribute によって参照できる。しかし、元々ID を持たない element やデータの一部に対して ID を割り当てる手段がないために、SGML のアドレス付け機構だけではハイパーメディアの記述に不十分である。

このモジュールは任意の element や外部のオブジェクトに対して ID を与える機構を提供することにより、アドレス付けを可能にする。

アドレスを表現する機構を大きく分けると、name space location と、coordinate location、semantic location の三種類がある。

Name space location　ID を持つオブジェクトと、ID を持たないが何らかの名前によって識別されるオブジェクトに対して ID を割り当てる機構である。

- named location address
  entity によって指示されるオブジェクトや、ID をもつ element、query によって返されるオブジェクトなどに ID を与える。

Coordinate location　ある座標系の中に位置するオブジェクトに対し、対応する座標を指定してそれを識別し、ID を割り当てる機構である。この機構を使うには、前述した measurement module を使用しなければならない。

一次元座標および木構造座標に基づく次のようなアドレスがある。ここで、location source とは、アドレス対象を含むオブジェクトのことである。

- data location address
  location source をバイト列と見なし、その部分

を指す。attributeの指定によって、バイト列を文字列と見なしたり、トークン列や数値列などと解釈することができる。

- tree location address, path location address
  location source を木の根とみなし、そのノードを指す。SGML 文書のある element を元に、その下位に位置する element を指定するために使う。
- list location address
  location source をノードのリストとみなし、その部分を指す。
- relative location address
  location source を木のあるノードとみなし、祖先や兄弟、子孫のノードを位置づける。

Semantic location 識別するための名前を持たず、座標系にも位置付けられていないオブジェクトにIDを与える機構である。

- Property location address
  attributeの値や、element名、entityのnotation名などにIDを与える。
- Notation-specific location address
  データの内容に依存する表記法でデータの一部にIDを付ける。例えば、内部ラベルを持つデータならそのラベルを指示してもよいし、CGMのデータなら「三番目の多角形」という指定でもよい。
- Bibliographic location address
  参考文献形式によって指示される任意のオブジェクトで、書籍やビデオ・テープ、テレビ番組、人や組織などのように、計算機外に存在しているオブジェクトを指すために利用できる。このアドレスの表現形式はアプリケーションが決める。このアドレスの内容は人間に提示され、人間が解決することになるであろう。

Location ladder あるアドレスを参照して新しいアドレスを作ることを繰り返し、任意のオブジェクトをアドレス付けすることができる。参照元となるアドレスをlocation sourceと呼び、こうして作成する一連のアドレスをlocation ladderという。location ladderの例を次に示す。

1. ID: local element を ID でアドレス付けする。
2. Tree location: その element の sub element をアドレス付けする。
3. Property location: その element の属性値をアドレス付けする。
4. Data location: その属性値の三番目のトークンをアドレス付けする。
5. Named location: そのトークンを entity 名とみなし、その entity が指すオブジェクトをアドレス付けする。
6. Data location: その最後 400 バイトをアドレス付けする。
7. Notation-spacific location: そのデータに query を行ない、結果をアドレス付けする。

3.5 Hyperlinks module

オブジェクト同士の接続を hyperlink という。SGML の ID attribute と IDREF attribute とによる参照関係の表現も hyperlink といえるが、それだけでは表現力が不足である。

図8に示すように、一つの hyperlink は一つの element で表現され、一つあるいは複数の link end を内容とする。link end の内容はオブジェクトに与えられた ID である。ここでオブジェクトとは element であったり location address であったり、hyperlink であったりする。link end によって指示されるオブジェクトを anchor と呼ぶ。一つの anchor は複数のオブジェクトを指すこともできる。

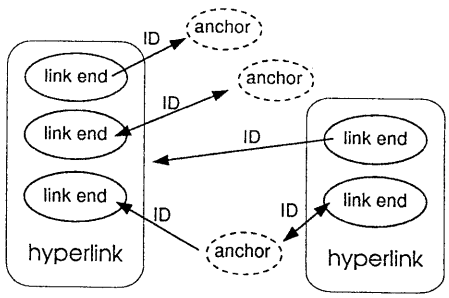

図 8: hyperlink の構成

Independent link　リンクが書かれる位置とリンクの内容とが無関係なhyperlinkである。すなわち、リンクの内容はlink endによってのみ規定される。変更することができないanchor (例えば書き込み権限のない文書内のelement) 同士にリンクを設定する場合などに有効である。図9のilinkという名前のelementがIndependent linkの例である。

Contextual link　リンクの内容と他のオブジェクトとを結び付ける。リンクの内容が空の場合は、そのリンクの位置と他のオブジェクトとを結び付けることになる。この形式のリンクでは指定できるlink endは一つだけである。図9のclinkという名前のelementがcontextual linkの例である。

```
<p id=sub> コメントされる内容 </p>
<comment id=mark> コメント </comment>
<ilink> sub mark </ilink>
<clink linkend=sub> コメント </clink>
```

図 9: リンクの記述例

Link traversal　対話的に使用するリンクでは、それぞれのlink endに対してのたどり方を指定できる。ただし、リンクのtraversalは必ずanchorから始まる。

指定は、リンクの外からそのanchorに入った場合と、リンクの中からそのanchorに入った場合の二つの場合に分けて、そのanchorからリンクの外へのたどりの可否と、リンクの中へのたどりの可否を記述する。

3.6 Scheduling module

同期や位置合わせの問題を扱うには座標軸の集合である座標空間が必要である。同期は時間軸上の位置合わせと捉えることができるので、両者は同じ機構で扱うことができる。なお、数学的な意味での座標空間は無限であるが、HyTimeで扱う座標空間は有限である。

オブジェクトには座標空間での固有の位置というものはない。後述するeventの中に組み込むことによって初めて座標空間上の位置が割り当てられる。

Finite coordinate space　オブジェクトはfinite coordinate space(以下ではFCSと略す)の中に位置づけられる。一次元のFCSの概念を図10に示す。FCSは座標軸の集合によって定義され、内部に後述するevent scheduleを複数持つことができる。それぞれの座標軸は、計測単位（量子の単位）と長さ（量子の数）によって定義される。もし、座標が絶対時間に対応する場合は、座標軸の最初の量子を絶対時間に結び付けておく。

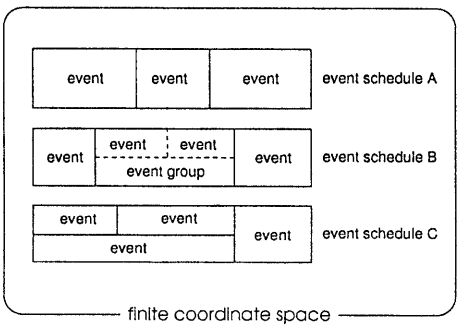

図 10: finite coordinate space

Event　オブジェクトに座標空間上の位置を結び付けるためのelementで、オブジェクトのbounding boxに当たる。eventは次の情報を持つ。

- extent specification: オブジェクトのそれぞれの座標軸上での位置および範囲
- extent reconciliation: オブジェクトが指定した範囲に足りないあるいははみ出す場合の処置
  - replacement: 別のオブジェクトに交換する。
  - vamping: 指定範囲を満たすまでオブジェクトを繰り返す。
  - filling: 指定範囲をmalleable objectで満たす。これは任意の大きさに合わせることができるオブジェクトで、pause、hold、wallpaperの三種類がある。
  - cropping: 指定範囲を越える部分を削る。
  - alignment: オブジェクトのどの部分をeventの範囲と合わせるかを指示する。

隣り合った一連の event をまとめて event group を作ることもできる。event group ではそれ自身の位置と範囲を定めることができる。

Event schedule 一連の event あるいは event group をまとめたものである。一つの FCS の中に複数の event schedule があってもよい。

アプリケーションはいろいろなタイプの event schedule を定義できる。例えば、複数の種類のメディアに対応する event をまとめて一つの event schedule を構成することもできるし、それぞれのメディア毎に event をまとめて複数の event schedule を構成することもできる。

一つの event schedule で、複数の event が同じ部分空間を占めてもよいし、どの event にも対応しない部分空間があってもよい。

FCS location address FCS をオブジェクトに被せ、その位置と範囲を指定することでオブジェクトの一部分を指示するためのアドレス付け機構である。

notation-specific location address でも同様のアドレス付けができるが、その解釈にはオブジェクトの notation を理解しなければならない。FCS location address では notation を理解する必要はない。

3.7 Rendition module

文書の描出方法は HyTime の対象範囲外であるが、描出の際のアプリケーションのパラメータの一部を標準的な手段で指定するための機構を提供するのが rendition module である。

rendition module には、event の中身であるオブジェクトの内容の変形を表現する機構と、event 自体の位置・範囲の投影を表現する機構がある。

このモジュールを使用する場合は scheduling module を必ず使用しなければならない。

Object modification この機構は、オブジェクトの内容を変形する手順を event schedule と結び付ける。この手順はオブジェクトの位置には影響を与えない。

オブジェクト内容の変形手順は図 11に示すように wand (魔法の杖) という一種のスケジュールを表現する element の中に置かれる。wand は modifier scope という element から構成され、座標空間上の位置と変形方法がその element の中で指示される。wand と modifier scope との関係は event schedule と event との関係に等しい。ただし event とは異なり、複数の modifier scope が同じ部分空間を占めてはならない。変形方法自体はアプリケーションによって定義される。変形方法の例として、グラフィックスに被せる色変換フィルタや、音声の増幅指示などが考えられる。

wand は同じ FCS の中の一つあるいは複数の event schedule と結合することによって、個々のオブジェクトの変形方法が指定されることになる。

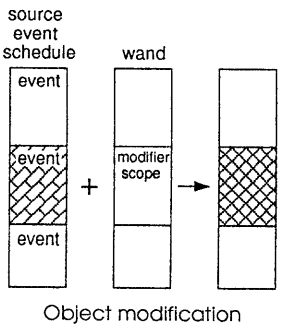

図 11: オブジェクトの変形

Event projection この機構は、event の位置を投影する手順を event schedule と結び付ける。この手順はオブジェクトの内容には影響を与えない。

event の投影手順は図 12に示すように baton (指揮棒) という一種のスケジュールを表現する element の中に置かれる。baton は projector scope という element から構成され、座標空間上の位置と投影方法がその element の中で指示される。baton と projector scope との関係は event schedule と event との関係に等しい。ただし event とは異なり、複数の projector scope が同じ部分空間を占めてはならない。投影方法自体はアプリケーションによって定義されるが、wand の場合とは異なり、HyTime で規定している投影方法もある。

baton は同じ FCS の中の一つあるいは複数の event schedule と結合することによって、個々の

event が別の FCS の event schedule の中に投影されることになる。

この機構の利用方法として、仮想時間空間に表現された event を実時間空間に写像したり、仮想距離空間に表現された event を実距離空間に写像することがある。例えば、グラフィックスの表示範囲の指示や、スライドの映写時間の指示、音楽のテンポの指示などが考えられる。

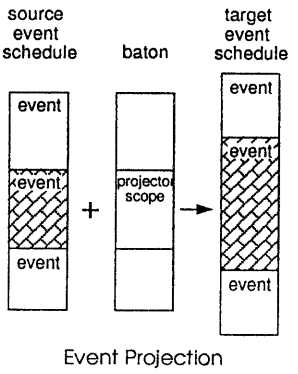

図 12: event の投影

4 おわりに

一つのハイパーメディアが様々なプラットフォームおよびアプリケーションで利用できると共に、より大きなハイパーメディアの一部としてそれを統合できるような環境が望ましい。HyTime はハイパーメディアをそのような開放・統合環境に導く上での重要な役割を果たす。

ISO では HyTime 以外にも MHEG や ODA 拡張、AVIs (*Audio Visual Interactive Scriptware*) といったハイパーメディアに関連する規格化活動が行なわれている。また個々のメディアの符号化方式である JPEG や MPEG などの標準化も進んでいる。

HyTime とこれらの標準や様々な業界標準をうまく組み合わせて、できるだけ標準化した基盤の上にハイパーメディアを表現し、利用していくことが重要である。特に今後のネットワーク社会では、計算機を利用した共同作業を分散環境で行なうことが多くなり、ハイパーメディアの統合・開放が不可欠になる。

謝辞 日頃より有益な議論や助言を頂いている日本事務機械工業会 SC18/WG8 国内委員会委員の方々に感謝いたします。

参考文献

[1] C. F. Goldfarb. *The SGML Handbook*. Oxford University Press, New York, 1990.

[2] C. F. Goldfarb. HyTime: A standard for structured hypermedia interchange. *IEEE Computer*, 24(8):81–84, Aug 1991.

[3] ISO 8879. *Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML)*, 1986.

[4] ISO 9069. *Information Processing — SGML Support Facilities — SGML Document Interchange Format (SDIF)*, 1988.

[5] ISO 9070. *Information Processing — SGML Support Facilities — Registration Procedures for Public Text Owner Identifiers*, 1990.

[6] ISO/IEC 10744. *Information technology — Hypermedia/Time-based Structuring Language (HyTime)*. will be published.

[7] ISO/IEC TR 9573. *Information Processing — SGML Support Facilities — Techniques for Using SGML*, 1988.

[8] S. R. Newcomb. Standard music description language complies with hypermedia standard. *IEEE Computer*, 24(7):76–79, Jul 1991.

[9] S. R. Newcomb, N. A. Kipp, and V. T. Newcomb. The "HyTime" hypermedia/time-based document structuring language. *Communications of the ACM*, 34(11):67–83, Nov 1991.

[10] 小町祐史, 田中洋一. HyTime と SMDL. 画像電子学会誌, 20(6):578–585, 1991.