情 報 学 基 礎 13 － 1
（ 1 9 8 9. 6. 2 ）

ハイパーテキストのデータモデル

西尾 信彦
東京大学 理学部 情報科学科

ハイパーテキストシステムをそれが基盤とするデータモデルに沿って研究することの意義について述べる．その例としてマルチユーザ／ネットワーク対応のハイパーテキストデータベースサーバのデータモデルである HAM，アイディアプロセッサ構築のために設計されたホルダネットワーク，TRON プロジェクトの BTRON で採用された統一的操作モデルである実身／仮身モデルの３つのハイパーテキストデータモデルを [1] で提出された次世代ハイパーテキストシステムが解決すべき７つの問題に即して比較／考察する．

A Comparison of Data Models in Hypertext Systems

NISHIO, Nobuhiko

Department of Information Science, Faculty of Science, the University of Tokyo

7-3-1 Hongo, Bunkyo-Ku, Tokyo 113, Japan

Data models that hypertext systems are based on are studied. The first one is HAM, Hypertext Abstract Machine, which is a hypertext database server for multi-users from network access. The second one is Holder Network, a hypertext-based data model designed for idea processors construction. Ths last one is the Real Object / Virtual Object Model designed as a unified model of human-computer interaction in the BTRON environment. A comparison of the power of expression and its limitation and a consideration on the systems based on these models are presented in the light of 7 issues presented by Halasz's paper [1].

1．はじめに

古くからあったハイパーテキストの考え方 [2] は近年の計算機環境の充実とともに実用的な使用が考えられるようになってきた．しかしその実用面に目をむける余りそのユーザインタフェース・デザインなどの研究に力が注がれ，まだハイパーテキストとはどのように定義できるかとか，今までのメディアとどこが異なっているかについて形式的に考察するような理論の研究はそれほどなされてはいない．本稿では各種のハイパーテキストシステムがバックボーンとしているそのデータモデルについて考察している研究についていくつかを提示し，それらを基にしてハイパーテキストシステムの可能性について考察する．

2．ハイパーテキスト・データモデル

ハイパーテキストの考え方が従来のデータと本質的に異なっているところは，データの中に陽に構造情報を持っているところであろう．構造情報をユーザが陽に入力できることの意義はいわば自然言語処理のような人工知能の研究の完成を待たずに計算機に人間の意図を伝えることができることにある．ここで構造 (structure) 情報という言葉が用いたがここではその定義を与えながらさらに詳しく考察する．

本来ハイパーテキストはメディアの一種であるため情報の伝達に用いられる．このとき情報を授受しあうものが人間同士である場合と人間と機械である場合に分けて考えてみる．人間同士の情報伝達は，自然言語（音声によるものもテキストによるものも）や図形情報（動画を含む）を中心的なメディアとして（その他には身振りなどが考えられる）行なわれる．人間はこれらのメディアを状況に応じて認識する能力を持ち，ハイパーテキストの考え方の導入によりこれら各種のメディアを統合化していく傾向にある．いわゆるマルチメディアという方向性である．またこの考え方を特に強調するときにはハイパーメディアという言葉が使われることが多い．ハイパーテキストデータモデルはこの統合化能力をはかるための基盤として研究することができる．一方，機械と人間とのあいだの情報伝達について考えてみるとハイパーテキストデータモデルの所為はさらに明らかになる．この場合には機械はまだ人間の用いている自然言語の理解や図形情報の理解（動画／シーン解析を含む）が満足にできる段階まで到達していない．そこで人間が機械と情報伝達を行なうときには機械のためにあらかじめ理解できる「プリミティブなもの」を定義しておく必要がある．この「プリミティブなもの」を特に本稿では構造（情報）と呼ぶ．但しここで機械がある「もの」を理解できるとは，機械に伝達された全情報の中からその「もの」がどこに存在しているかを判別できてその他に理解できる「もの」との間で区別がつけられることを意味する．すべてのメディアはこの構造情報とそれ以外の部分から構成されると捉え直すことができる．この「それ以外の部分」を以後リテラル情報と呼ぶ [3].

ハイパーテキストというメディアは機械と人間との情報伝達においてこの構造情報を機械と人間両方がより積極的に利用することによりメディアとしての効果を上げようとするものだと考えることができる．すなわちユーザが入力してくれた構造情報を自動的に解析することがシステムには要求されることになる．この構造情報を規定しているものがまさしくハイパーテキストデータモデルであり，これについての理論的研究がなされることには非常に意義があると考えられる．

ハイパーテキストをそれが基盤としているデータモデルを通して研究することのより具体的な意義には以下の点を挙げることができる．すなわち，

（1）扱っている問題の所在を明らかにすることにより機能の拡張／補充についての厳密な考察ができる．
（2）データモデルを解析することによりそのシステムのデータの表現能力について言及することができて，他のシステムとの比較が可能になる．
（3）基本となるデータベースサーバをモデルに基づき構築すれば複数のシステムの実現が可能となり異種システム間でデータの互換性を上げることが可能になる．
（4）ハイパーテキストをブラウズするための問い合わせ言語作成のための基礎となる．

等が考えられる．以下の節ではハイパーテキストデータモデルを中心にした研究のうち，HAM [5][6]，ホルダネットワーク [3][4]，実身／仮身モデル [7][8][9]を挙げて順に述べる．ハイパーテキストシステムについては最近の Halasz の論文 [1] に次世代のハイパーテキストシステムへの移行における問題点を揚げているものがある．簡単にそれらについてここで触れておく．

(Issue 1) ハイパーテキストネットワークに蓄えられた情報に対する検索や問い合わせのような参照のための計算をどのようにサポートすべきか？これはノードの内容に関する検索と構造情報に関する検索に分けられ後者に関する研究は余り見られない．
(Issue 2) 広範なハイパーテキストネットワークを階層的に構造化するために必要となる包含関係はどのように表わすべきか？
(Issue 3) 徐々に変遷していくネットワークの情報を整合性をもって管理していくためには，その過渡的な時期の情報をどのように蓄えておくべきか？
(Issue 4) ハイパーテキストネットワークに蓄えられた情報を能動的に変更するような計算（例えばデータの整合性をチェックしたり推論エンジンを動かしたり）をどのようにサポートすべきか？
(Issue 5) オブジェクトのバージョン管理をどうするか？
(Issue 6) 多人数の共同利用にどう対処すべきか？
(Issue 7) システムの機能の各調整や目的に合わせた調整をどう与えるか？

以上がそこであげられている問題点である．データモデルの研究ではハイパーテキストシステムの下部構造を対象としているためにこれらの問題全てをデータモデルのレイヤにおいて解決するわけではない．即ち (Issue 3) のようなネットワークの仮想構造はハイパーテキストシステムのインタフェースレイヤにおいて扱われることが多いだろうし，(Issue 5) や (Issue 6) のデータアクセスに関する部分はさらに下部構造のファイルシステムにおいて解決されることもある．以下の節ではそれぞれのモデルが特にどの問題解決を目指しているかについて明らかにしてから論を進めていくことにする．

3．HAM

HAM (Hypertext Abstract Machine) は計算機支援されたソフトウェア開発環境（最近では CASE/Computer Aided Software Engineering といわれている）をサポートする CAD システムである Neptune システムの下位レイヤとしてホストファイルシステムの上位レイヤとして構築された汎用のハイパーテキストデータベースである [5]．このレイヤの上にハイパーテキストシステムのアプリケーションレイヤがさらにその上にユーザインタフェースレイヤを構築して全体のシステムを完成させる．HAM は，トランザクションベースでネットワークからもアクセス可能なマルチユーザ対応のデータベースサーバとして実現されていて (Issue 6) の共同利用対応になっている．そのために実際いくつかの異なったハイパーテキストのアプリケーションに適用可能である．例えば最近の論文 [6] で

は，HAMの表現力を実証するためにGuideシステム [12] のButtonオブジェクト，Intermediaシステム [13] のMail Web機構，NoteCardsシステムのFileBoxオブジェクトをそれぞれ記述し実現することに成功している．明らかにハイパーテキストデータモデルを提言することにより得た恩恵である．HAMはハイパーテキストデータモデルを提出し実装された最初の例だということができる．

HAMで扱う情報の単位はノードと呼ばれ，ノード間にはそれらの関係を表わすリンクを付けることができる．ノードやリンクをまとめるための機構をコンテキストという．各コンテキストは一つの親コンテキストを持ち（もちろんルートコンテキストは自分を指す），任意個のコンテキスト，ノード，リンクを包含する．コンテキストは，作業領域を切り分けてブラウズの便を図ったりソフトウェアシステムの構成管理，バージョン管理ツリー等を実現するために使われる．コンテキスト，ノード，リンクには属性を付けることが可能である．属性はそれらのオブジェクトに意味を与える働きをする．また，これらのオブジェクトが作る全体の構造をグラフと呼んでいる．これらの形式的定義は論文 [5] の付録に詳述されている．この形式的記述からこのデータ構造のための問い合わせ言語の核の構築が容易になっていて後はアプリケーションレイヤで各アプリケーションに合わせたインタフェースを被せることとなる．

HAMを論ずるにはまずハイパーテキストシステム構築のためにレイヤード構造を採用しているのに注目すべきである．レイヤード構造であるということは (Issue 3) のネットワーク情報の解析計算ドライバの実現や (Issue 7) の拡張性についてはデータモデルの外のレイヤでの問題となる．

またHAMのレイヤにはあらゆるハイパーテキストシステムが基盤とすべき基本オブジェクトを用意する必要があり，そのレイヤ中にコンテキストという今までのノードとリンクで全てを構成するというハイパーテキストの考え方にはなかったものを入れている．これは事物の関係を表わすのには，意味ネットワークの研究で良く使われるように参照を表わすis-aリンクと包含を表わすis-part-ofリンクが本質的に必要であることを反映していると考えられる．またこれはもちろん先にあげたうち (Issue 2) に対しての解決法である．従来のハイパーテキストのリンクの考え方は非常に強力で包含関係を表わすリンクを参照関係を表わすリンクを用いて実現することで間に合わせていた．例えば，NoteCardsのFileBox機能 [10] は実際には参照関係を表わすリンクにより実現されている．しかし近年の研究 [1]により，この二つの関係は本質的に異なるものでありそれぞれ別にオブジェクトを用意すべきであると言われていて，コンテキストはそのための一手法である．しかしノードとは異なったものとして導入されているためにコンテキスト間のリンク付けというものをノードのときとは別に考え直す必要があるのかとか，ノードのバージョン管理とコンテキストのそれとの関係をどうするかといった問題も出てきている．

この他にHAMには，バージョン履歴の管理を行ない，アクセスするオブジェクトを制限するためのフィルタ機構やセキュリティ確保のためにアクセスコントロールリストの機構もこのデータモデルのレイヤで組み込まれている．

4．ホルダネットワーク

ホルダネットワークはアイディアプロセッサを実現するためのデータ構造としてハイパーテキストを用いることを前提に考案された [4]．このデータモデルは以下のような特長を備えている．

（1）ハイパーテキストを構成するオブジェクトはホルダしか存在しない．
（2）データのネットワーク構造の部分集合をノードとして扱うことができる．
（3）すでに定義されたリンクを用いて新しいリンクを構造化することができる．
（4）1本のリンク結合を6つのポインタを用いて実現しているために構造情報の検索を高速に行なうことができる．

次節では，まずホルダネットワークの構成を大まかに説明する．詳しい形式的定義はこれを用いてアイディアプロセッサを構築したときの機能とともに付録としたので参考にされたい．

4．1　ホルダネットワークの構成

ホルダネットワークではハイパーテキストを単一のオブジェクトホルダによって構成する．そのためのホルダはやや重い存在になる．ホルダは以下のものから構成される．

（1）パラフレーズリテラル（従来のハイパーテキストのノードの格納する情報及び型付きリンクの型を表わす）
（2）ホルダへのポインタのテーブル×2（それぞれをスーパホルダテーブル，サブホルダテーブルと呼ぶ）
（3）単一化環境表

パラフレーズリテラルにはあらゆるデータが載ることを仮定している．またその中にはスロットと呼ばれる特殊な要素を0個以上埋め込むことができパラフレーズリテラル中のどこにスロットがあるか判別できることになっている．スーパホルダテーブルやサブホルダテーブルは同じホルダを複数回指すことができるようにする．ホルダはそのサブホルダテーブルの指している各ホルダをホルドしていると言い，スーパホルダテーブルが指している各ホルダにホルドされていると言う．
単一化環境表はスロットへのポインタとスロットへのポインタの対，またはスロットへのポインタとホルダへのポインタの対を連ねたテーブルである．この対として現れるスロットは，
（1）そのホルダのパラフレーズリテラル
（2）そのホルダがホルドする各ホルダのパラフレーズリテラル
のいずれかに埋め込まれているものでなくてはならない．また対として現われるホルダは，
　そのホルダがホルドするホルダ
でなくてはならない．

4．2　ホルダネットワークのセマンティックス

ホルダネットワークにおいては，通常のハイパーテキストでいうノードをホルダとそのパラフレーズリテラルで表現する．ハイパーテキストにおける2つのノードを結ぶリンクは，スロットを2個埋め込んだパラフレーズリテラルをもつホルダを用いてそのスロットが実例化される環境で表現する．例としてホルダAとホルダBとの間に前者は後者の例になっているという関係が成り立っている状況をホルダネットワークで表してみる．（ここでは‘%’がリテラルのなかでスロットのはじめとおわりを検出するための特殊文字となっている）

（1）AとBの実体であるホルダAとホルダBと「%スロット1%は%スロット2%の例になっている」というパラフレーズリテラルをもつホルダRを生成する．
（2）上記3ホルダをホルドするホルダHを生成する．
（3）ホルダHの単一化環境表にホルダRの%スロット1

%とホルダA，ホルダRの%スロット2%とホルダBの2つの対を記録する．

以上の手続きにより，ハイパーテキストにおけるノード間のリンク付けを実現する．以後ホルダネットワークでリンクと呼ぶときは上のようにして実現されたものをさすこととする．このとき単一化環境表に連ねられる各対はそれぞれの単一化(unification)の情報を保持するものと考える．ここでは同じ単一化でもスロットにホルダが単一化したときには，それを実例化(instanciation)と呼び区別することにする．ここではホルダRは「クラスとしてのリンク」を，ホルダHは「インスタンスとしてのリンク」を表わしている．

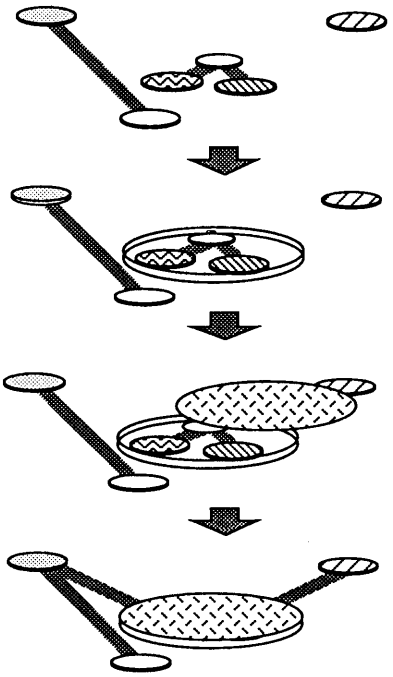

Fig.1 ノードとコンテキストを兼ねるホルダ
ノードの上蓋はパラフレーズリテラルを意味し下部構造を隠し抽象化を図っている

続いてホルダネットワークのサブネットワークをノードとして扱うことについて説明する．先ほどの例のようにして平面的にネットワークを構成していくとき，ホルダHはその全体のネットワークの中ではサブネットワークを表しており，一方ホルダHはホルダであるのでホルダAなどと同様にホルダネットワークのノードとなり得る．A—（R）→Bといったように小さな場合はホルダHがサブネットワークを表わすノードたりえるが，ホルダHのレベルのホルダを複数個使って表わすようなサブネットワークをノードにしたいときには以下に示すように三つの方法があって，それぞれに構造化の考え方が異なっている．

（1）小さなサブネットワークを表わす各ホルダがホルドするホルダすべてを単一化するひとつのホルダを生成する．
（2）小さなサブネットワークを表わすホルダのすべてをホルドするひとつのホルダを生成する．
（3）小さなサブネットワークを表わす各ホルダがホルドするすべてのリンクを表わすホルダをホルドし単一化するホルダを生成し，それをその他のホルダと実例化するホルダを生成する．

このようなサブネットワークをあらわすホルダのパラフレーズリテラルのもつ意味について考察する．このようなホルダの表わすサブネットワークは，その名のとおりホルダネットワークで表わされているわけであるが，単にスロットをホルダで再帰的に置き換えてできあがるリテラルデータでは全体を抽象化するときなどに不利である．そこでそのホルダがあらわす情報をもう一度パラフレーズしたリテラルデータをパラフレーズリテラルとして格納することにした．そのときそのホルダが実例化していないスロットを持つときパラフレーズリテラルもそれに単一化したスロットを埋め込むことにする．ただし，ここではすべての実例化されていないスロットが埋め込まれる必要はない．それはどのようなパラフレーズがされるかによる．以上でリンクを表わすホルダを複数個ホルドするホルダを生成して新たなリンクにパラフレーズすることが可能なことがわかる．これにより，リンクに潜在する介在者を隠ぺいすることもできるし，必要なときにはその隠ぺいされたものの他との関係も得ることができるようになる．

4．3 ハイパーテキストデータモデルとしてのホルダネットワーク

ホルダネットワークもハイパーテキストのデータモデルとしてシステムの下部構造を担当している．HAMの研究などと比べてより強力な表現力をより美しく簡潔な構造の中に持たせるために汎用性を追及したものになっている．ただしまだ実験的なシステムしかその上での実装がなされていず，全体として効率の良いものができるかどうかは研究中である．そのため今のところ(Issue 3)のハイパーテキストネットワークの仮想構造や(Issue 5)のオブジェクトのバージョン管理(Issue 6)マルチユーザ対応の部分はこのモデルの扱うレイヤでは一切考慮されていない．

形式的なデータ構造の定義とその中心的なアプリケーションであるアイディアプロセッサの機能の定義は明確に与えてあるので（付録参照）(Issue 1)の問い合わせ言語の核の部分はHAMの場合と同様に構築が容易なはずである．特に構造検索のための構造マッチングに関しては多くのポインタを用いてリンクを構成している分，効率の良い実現が可能になる．

(Issue 2)のネットワークの部分集合を包含関係によりまとめる部分がこのモデルの最も特長的なところである．前のHAMの部分で述べたようなコンテキストにあたる考え方であるが，このモデルではノード，リンク（従来の意味でのリンクオブジェクトは各型のリンクにつき一つしか存在しないことに注意されたい），コンテキストを全て同じオブジェクト即ちホルダで表現している．新たにオブジェクトの種類を増やすことなく拡張されている点ではHAMよりも美しく汎用性があるといえる．この考え方のためにホルダネットワークではコンテキストのようにコンポジットノードと普通のノードに何ら区別がない上に，このサブネットワークをノード化する機構を同様に用いてリンクのコンテキストと言える概念を導入しリンクを構造化することにも成功している．

以上のような意味ではこのモデルに基づいたシステムはいわゆるハイパーテキストシステムというよりも知識ベースシステムよりのものに向いている．すなわち今までリテラル情報であった部分に新たに構造情報を埋め込むことが可能になっているためメディア情報の抽象化の機

構をそのまま持っていることになる．このため(Issue 6)のようなハイパーテキストネットワーク上の情報を解析するシステムソフトウェアの構築に向いておりアイディアプロセッサのような自分でもどのように整理したらいいかわからないデータ構造の解析支援を必要とするシステム構築向けになると考えられる．サブネットワークを解析してそのバラフレーズリテラルを生成するネットワーク情報解析計算ドライバに暗黙計算が準備できれば煩雑な情報の構造化／組織化が半自動化することができる．

システムの拡張性については，基本オブジェクトがホモジニアスであるいう点から汎用性に富み広範な対応が可能になるが，全体的な実装効率はその分機性になると考えられる．

5．実身／仮身モデル

実身／仮身モデルはTRONプロジェクトの中のスーパーパーソナルワークステーションのためのOSを設計するBTRONサブプロジェクトで採用される統一的操作モデル[9]として提案された．BTRONはTRONトータルアーキテクチャの中で，多国語処理，ネットワーク構造ファイルシステム（実身／仮身フィイルシステム），統一的操作環境（MMI :Man-Machine Interface），アプリケーション間データ互換（TAD :TRON Application Databus）などによって特徴付けられる．実身／仮身モデルはこれら全てを統合する概念でありここでは特にその中のネットワーク構造ファイルシステムにより提供されるハイパーテキスト環境について述べることにする．

実身／仮身ファイルシステム（実身／仮身モデルの考え方から生まれたネットワーク構造ファイルシステム）では従来のファイルにあたる部分を実身と呼びそれを参照するタグを仮身と呼ぶ．これらはハイパーテキストシステムでのノードとリンクの起点に相当する。この仮身は実身中のデータの任意の場所に埋め込むことができ，これによりネットワーク構造のファイルシステムを構成する．実身中に格納するデータのフォーマット（埋め込まれる仮身のフォーマットも含む）はTADにより規定され，同時にアプリケーション間での共通のデータフォーマットとなり互換性を保証することとなる．TADにはデータの標準として文章データ，図形データ，仮身，付箋などがある．文章データは文字データセグメントが一次元的に並んでおり，その中に図形データセグメントや仮身を埋め込んだりすることができる．一方図形データは二次元データとも呼ばれ個々の図形データセグメントを二次元的にオーバラップした配置情報を付けて格納できる．この図形データセグメントの中にはテキストを格納するテキストボックスもあり，それ以外に仮身も置くことができる．実身の中には付箋を用いることにより各アプリケーション固有のデータを自由に拡張して埋め込むことができる（このような付箋を指定付箋という）．その他のアプリケーションがその実身を読むときには自分の知らない付箋による部分を読み飛ばすことができアプリケーション間でのデータ互換を保証できる．また機能付箋と呼ばれる付箋は実身／仮身モデルの考え方からいうとプログラム実体（生身という）をソフトウェア的にリンクする仮身であると捉えることもできる。実身にこの機能付箋が複数個ついているとこの実身をウィンドウに開くための実行メニューにその数だけの項目が出現するようになっている．ユーザインタフェースのレイヤからみるとこのハイパーテキストデータモデルはデータファイルだけではなくアプリケーションプログラムもリンクしているように見える．さらにこのファイルシステムや，TADに保証されたデータアクセス，アプリケーション起動などのための「お回し」言語としてTACL(TRON Application Control flow Language)という言語の研究[8]がTULS (TRON Universal Language System)の一貫としてなされている．以上簡単ではあるが実身／仮身モデルが，データフォーマット（TAD)という低レベルからネットワーク構造ファイルシステムへそしてメニューによるユーザインタフェースまで貫かれた思想であることを紹介した．さらに以降で特にその中の実身／仮身ファイルシステムにより提供されるハイパーテキスト環境についてこの階層構造をもとに詳しく見ていくことにする．

実身／仮身モデルはOSの基盤となる考え方なので何よりも効率を重視した作りになっている．すなわちOS自身がカーネルを2階層構造で実現しており，実身／仮身ファイルシステムはその内核のファイル管理機能と外核の実身／仮身マネージャによって実現されている．ファイル管理機能の部分は実身ファイルをレコードという単位の系列であると管理し，そのレコードの中でリンクレコードのみその内容を認識してそれ以外のレコード内のデータを読むことはない．リンクレコードはその実身ファイルがどの実身ファイルにリンクしているかを記録しディレクトリの役目をしている．テキストや図形中に仮身をどのように見せるかといったようなことは外核の実身／仮身マネージャに任されている．実身／仮身マネージャの機能は将来的にはTULS/TACLと呼ばれる言語体系の中でデータ互換管理をつかさどる部分によって拡張されていく可能性を残している．具体的には今まで実身／仮身マネージャによって提供されていた機能がTULS/TACLにより記述し直されることにより解釈実行できるようになる。現段階では機械語レベルまでこの機能がコンパイルして実行されていると解釈でき将来への互換性を保証している．

```
【TADデータ】::=《管理情報セグメント》【TAD本体】
【TAD本体】::=【文章データ】｜【図形データ】
【文章データ】::=《文章開始セグメント》【文章要素並び】《文章終了セグメント》
【文章要素並び】::=《空》｜【文章要素並び】｜【文章要素】
【文章要素】::= 【文字コード】｜《文章付箋セグメント》｜【図形データ】｜
                【文章データ】｜《画像セグメント》｜《仮身セグメント》｜
                《指定付箋セグメント》｜《機能付箋セグメント》
【文字コード】::=《通常文字コード》｜《制御コード》｜
                《言語指定コード》｜《特殊文字コード》
【図形データ】::=《図形開始セグメント》【図形要素並び】《図形終了セグメント》
【図形要素並び】::=《空》｜【図形要素並び】｜【図形要素】
【図形要素】::= 《図形描画セグメント》｜【文章データ】｜【図形データ】｜
                《画像セグメント》｜《仮身セグメント》｜
                《指定付箋セグメント》｜《機能付箋セグメント》
```

Fig. 2　TADの構造定義

このデータモデルを前からの問題に照らして考えてみる．通常BTRONマシンではデータランドエディタと呼ばれるシステム供与のアプリケーションがビジュアルシェルとして起動されており、以下ではこのシェル環境をハイパーテキストシステムと捉えて考察を進める。(Issue 1)の問い合わせ言語及び(Issue 7)のシステムの拡張性についてはTULS/TACLによりサポートされるはずであるが，いまのところまだ研究段階であり実装に関してもTRONCHIPを核としない現状のワークステーションにはいささか荷が重いところもある．今後の効率の良い実装

が待たれるところである．(Issue 2) のコンポジットネスに関しては，実身／仮身マネージャのレベルでサポートされる「開いた仮身」により実現されることになる．ただしこれも TULS/TACL 体系が完成するまでは包含したデータを見ることのみで編集機能まではつくことにはならない．(Issue 3) の仮想構造や (Issue 4) のネットワーク情報に関する計算ドライバはこのうえのアプリケーションのレイヤでまかなわれるはずである．(Issue 5) (Issue 6) のバージョン管理やネットワーク経由の実身／仮身ファイルシステムへのマルチアクセスについては内核のファイル管理機能によりサポートされている．

筆者は現在この実身／仮身モデルの上でのホルダネットワークスキームの実現を考えておりホルダネットワークの効率の悪さを最下位レイヤからハイパーテキストファイルシステムを考慮した実身／仮身モデルで補おうと考えている．

6．最後に

現在ハイパーテキストシステムの研究が盛んになり徐々に実用化されたものも出始めている．特に米国 APPLE 社の HyperCard [11] はその代表だと言われている．しかしここまで考えてきたデータモデルに照らしてこのシステムを考えると，これは実はハイパーテキストシステムであるとはいい難いものであることがわかる．たくさんのカード同士をつぎつぎにボタンオブジェクトなどによってリンクしていくことによりカードをノードとしたネットワーク構造を構築できるのは確かなのであるが，このシステムには明らかにリンクを表わすオブジェクトが存在しない．先に述べたようにボタンオブジェクトを用いてカード間をリンクするときもその間を結んでいるのはボタンオブジェクトに付随する HyperTalk で記述されたプログラムでありそのリンク先を決定するためにはプログラムの中に変数などがあるためにプログラムの実行を必要とする．すなわち明白なリンクオブジェクトを欠いているために静的にハイパーテキストの構造を解析することができない．このような解析が静的にできないということはいわゆるブラウザツール（ネットワークの全体像を見たり，問い合わせ言語を処理したりする）の作成ができないわけでハイパーテキストシステムとしての半分以上の機能を捨ててしまっているといえる．かと言ってこのシステムが劣悪なものであるというつもりはない．大規模なデータを処理しないかぎり問い合わせ言語（HyperTalk は一枚のカード内のデータの検索には使えるかも知れないがカードネットワーク情報についての問い合わせ言語にはならないことに注意されたい）のようなものを必要とせずポインティングデバイスによるダイレクトマニピュレーションですべてまかなえるのならば，このシステムは効果を発揮するはずである．

ハイパーテキストデータモデルの研究はこのように既存のシステムの評価と問題点の所在を明らかにすることができ，今後の次世代ハイパーテキストシステムは堅固なモデルの上に効率良く構築されたシステムになっていくはずである．

謝辞

日頃ご指導くださる坂村健先生，ならびに草稿を読み貴重な助言をいただいた高田広章氏に感謝いたします．

参考文献

1. Halasz, F. G. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. Commun. ACM 31, 7 (July 1988), 836-851.
2. Bush, V. As we may think. *Atlantic monthly* 176, (July 1945). Reprinted in CD ROM: The New Papyrus, Microsoft Press, 1986.
3. 西尾信彦，小野芳彦，山田尚勇：アイディアプロセッサには何ができるか？ *Proc. 4th Symp. on Human Interface*, pp. 1-8 (Nov. 1988).
4. 西尾信彦，山田尚勇：発想の計算機支援，情報処理学会研究会文書処理とヒューマンインタフェース，15-2, pp. 1-8 (Nov. 1987).
5. Delisle, N. and Schwarts, M. Neptune: A Hypertext System for CAD Applications, *Proc. SIGMOD 1986*, pp. 132-143 (1986).
6. Campbell, B. and Goodman, J. M. HAM: A General Purpose Hypertext Abstract Machine. *Commun. ACM* 31, 7 (July 1988), 856-861.
7. Sakamura, K. BTRON: The Business-oriented Operating System, *IEEE Micro*, Vol. 7, No. 2, pp. 53-65 (Apr. 1987).
8. Sakamura, K. TACL: TRON Application Control-flow Language, *Proc. 5th TRON Project Symp.* 1988, Springer-Verlag, pp. 79-92 (Dec. 1988).
9. 坂村健：BTRON における統一的操作モデルの提案，情報処理学会情報処理，Vol. 26 No.11, pp. 1321-1328 (Nov. 1985).
10. Halasz, F. G., Moran, T.P. and Trigg, R.H. NoteCards in a Nutshell. In *Proceedings of the 1987 ACM Conference of Human Factors in Computer Systems (CHI + GI '87)* (Toronto, Ontario, Apr. 5-9). 1987, pp. 45-52.
11. Goodman, D. *The Complete HyperCard Handbook.* Bantam Books, New York, 1987.
12. *Guide: Hypertext for the Macintosh Manual.* OWL International, Inc., Bellevue, Wash., 1986.
13. Meyrowitz, N. Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In *OOPSLA '86 Proceedings* (Portland, Or., Sept. 29-Oct. 2, 1986), 186-201.

付録：ホルダネットワークの形式的定義

本文中で直観的に与えたホルダ・ネットワークの形式的定義を与える．

［定義１］A を 256 種類の情報符号セットであるとする．A には含まれない要素 S をスロットと呼ぶ．このときリテラル L を以下のように定義する．

$$X = A \cup \{S\},$$
$$L = X^*.$$

関数 Slot#:L→N は以下のごとく定義される．

$$Slot\#(e) = 0,$$
$$\forall a \in A.\ \ Slot\#(a) = 0,$$
$$Slot\#(S) = 1,$$
$$Slot\#(a \cdot b) = Slot\#(a) + Slot\#(b). \quad \blacksquare$$

ただし a・b は要素 a と要素 b の連接 (concatenation) を表わす．

前ホルダ（well-formed ホルダの候補）の集合 H を次のような四つ組の集合で与える．ただし N は 0 を含む自然数全体，N+ は 0 を含まない自然数全体である．

［定義２］前ホルダの集合 H は，

$$H = L \times SP \times SB \times U.$$

ただし以下の条件を満たす．

SP = power set of ID*, SB = power set of ID*,
U = power set of {(h1, s1, h2, s2) | h1, s1, h2, s2∈N}.■

このとき，L はバラフレーズ・リテラルの集合を，SP はスーパーホルダテーブルの集合を，SB はサブホルダテーブルの集合を，U はユニフィケーションテーブルの集合を表わしている．特に U はホルダ h1 のスロット s1 とホルダ h2 のスロット s2 をユニファイしていることを表わす．ただしこの場合の h1, h2 はそのホルダのサブホルダテーブルにおけるインデックスによりホルダを特定している．ホルダが 0 の場合は自分自身のホルダのバラフレーズ・リテラルを表わし，スロットが 0 の場合はホルダ自身を表わす．前ホルダの集合 H について 4 つの射影関数 pri, i = 1..4 が存在する．それぞれ pr1:H→L, pr2:H→

SP, pr3:H→SB, pr4:H→Uで表わされるとする．また関数SPTop:SP→ID, SBTop:SB→IDはそれぞれの先頭の要素を返し，関数 SPRest:SP→SP, SBRest:SB→SBはそれぞれ先頭の要素を取り除いたものを返すものとする．前ホルダには自身の属性としてIDがあり，その集合IDは，

ID = N+.

である．

［定義3］関数 SPIndex:SP × N+→ID∪{0}, SBIndex:SB → N+→ID∪{0}, SPOcc:SP × ID→N, SBOcc:SB × ID→Nを以下のように定義する．

もし sp = e ならば SPIndex(sp, n) = 0, SPOcc(sp, id) = 0,
もし n = 1 ならば SPIndex(sp,n) = SPTop(sp),
それ以外のときは

SPIndex(sp, n) = SPIndex(SPRest(sp), n - 1),

もし SPTop(sp) = id ならば

SPOcc(sp, id) = 1 + SPOcc(SPRest(sp), id),

それ以外のときは SPOcc(sp, id) = SPOcc(SPRest(sp), id).
関数 SBIndex, SBOccに関しても同様に定義される． ■

これにより例えば関数 SPIndex(sp, n) を用いて，スーパーホルダテーブル中の n 番目のホルダの ID番号が得られ，関数 SBOcc(sb, id) によりサブホルダテーブル中に何回 id というホルダの IDが現われているかを得る．また演算記号∠を次のように決める．

id∠SP ⇔ SPOcc(sp, id) > 0,
id∠SB ⇔ SBOcc(sb, id) > 0.

続いて前ホルダ・ネットワーク（well-formed ホルダ・ネットワークの候補）の全体の集合 HNを次に与える．

［定義4］前ホルダ・ネットワークの全体の集合 HNは以下のような写像の集合である．

HN = (ID→H∪{nil}) ただし，∀hn ∈ HN. hn(i)≠nil なる i の個数は高々有限個． ■

以上より well-formed ホルダ・ネットワークと well-formed ホルダを定義する．

［定義5］well-formed ホルダ・ネットワークの全体の集合 WFHNとは，HNの元 hnで以下の条件を満たすものの集合である．また well-formed ホルダ・ネットワーク wfhn の値域をなす前ホルダのみが well-formed ホルダである．

∀id∈ID s.t. hn(id)≠nil,
∀id'∠pr2(hn(id)).
hn(id')≠nil,
SPOcc(pr2(hn(id)), id') = SBOcc(pr3(hn(id')), id),
∀id"∠pr3(hn(id))
hn(id")≠nil,
SBOcc(pr3(hn(id)), id") = SPOcc(pr2(hn(id")), id),
∀(h1, s1, h2, s2)∈pr4(hn(id)).
もし h1 = 0 ならば
h2≠0, SBIndex(pr3(hn(id)), h2)≠0,
1 ≦ s1 ≦ Slot#(pr1(hn(id))),
1 ≦ s2 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h2)))),
もし h2 = 0 ならば
h1≠0, SBIndex(pr3(hn(id)), h1)≠0,
1 ≦ s2 ≦ Slot#(pr1(hn(id))),
1 ≦ s1 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h1)))),
h1≠0 かつ h2≠0 ならば
SBIndex(pr3(hn(id)), h1)≠0,
SBIndex(pr3(hn(id)), h2)≠0,
0 ≦ s1 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h1)))),
0 ≦ s2 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h2)))),
s1≠0 または s2≠0. ■

以後 well-formed ホルダ・ネットワークと well-formed ホルダをホルダ・ネットワークとホルダとそれぞれ呼ぶ．

以上でホルダ・ネットワークは形式化できたわけであるが，このままでは計算機上での手続きが形式化しにくいので，いくつかの演算をここで導入し，それによって構築されるもののみが well-formed ホルダ・ネットワークであると形式化し直す．

［定義6］∀id∈ID. hn(id) = nil なる前ホルダ・ネットワーク hn はホルダ・ネットワークであり，以下の関数により構成される前ホルダ・ネットワークのみがホルダ・ネットワークである．

Generate:WFHN×ID→WFHN.
Generate(wfhn, id) = wfhn' とすると，
もし wfhn(id)≠nil なら，
Generate(wfhn)・(id) = wfhn(id),
もし wfhn(id) = nil なら，
Generate(wfhn)・(id) = (e, e, e, {}).
ただし演算 ・ はリストの連接とする．

SetL:WFHN×ID×L→WFHN.
特定のホルダのパラフレーズ・リテラルを変更する．
SetL(wfhn, id, l) = wfhn' とすると，
∀id'≠id. wfhn'(id') = wfhn(id).
if Slot#(pr1(wfhn(id))) = Slot#(l) then
pr1(wfhn'(id)) = l
else
pr1(wfhn'(id)) = pr1(wfhn(id)),
pr2(wfhn'(id)) = pr2(wfhn(id)),
pr3(wfhn'(id)) = pr3(wfhn(id)),
pr4(wfhn'(id)) = pr4(wfhn(id)).

Hold:WFHN×ID×ID→WFHN.
Hold(wfhn, id1, id2) = wfhn' とすると，
もし wfhn(id1)≠nil かつ wfhn(id2)≠nil ならば
i≠id1, id2 において wfhn'(i) = wfhn(i)&
pr1(wfhn'(id1)) = pr1(wfhn(id1)) &
pr2(wfhn'(id1)) = pr2(wfhn(id1)) &
pr3(wfhn'(id1)) = pr3(wfhn(id1))・id2 &
pr4(wfhn'(id1)) = pr4(wfhn(id1)) &
pr1(wfhn'(id2)) = pr1(wfhn(id2)) &
pr2(wfhn'(id2)) = pr2(wfhn(id2))・id1 &
pr3(wfhn'(id2)) = pr3(wfhn(id2)) &
pr4(wfhn'(id2)) = pr4(wfhn(id2))
それ以外のときは wfhn' = wfhn.

Unify:WFHN×ID×N×N×N×N→WFHN. Unify(wfhn, id, h1, s1, h2, s2) = wfhn'.i≠id においては wfhn'(i) = wfhn(i) である．

以降 i = id において wfhn(i)≠nil なる場合のみを考える．この場合，pr1(wfhn'(id)) = pr1(wfhn(id)) かつ pr2(wfhn'(id)) = pr2(wfhn(id)) かつ pr3(wfhn'(id)) = pr3(wfhn(id)).のようにユニフィケーションテーブル以外は変化がない．以後ユニフィケーションテーブルについて述べる．

h1≠0かつh2≠0 でありかつwfhn(SBIndex(pr3(id), h1))≠nil, wfhn(SBIndex(pr3(id), h2))≠nil であるときは，
s1, s2 のどちらかが非零でありかつ，
0 ≦ s1 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h1))))
かつ 0 ≦ s2 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h2)))) であるときのみ pr4(wfhn'(id)) = pr4(wfhn(id))∪{(h1, s1,h2,s2)}.

h1≠0 かつ h2 = 0 かつ
wfhn(SBIndex(pr3(id), h1))≠nil のとき，
1≦s2≦Slot#(pr1(id)) かつ，
1≦s1≦Slot#(pr1(hn(SBIndex(pr3(hn(id)),h1))))
であるときのみ pr4(wfhn'(id)) = pr4(wfhn(id)),
h1 = 0 かつ h2≠0 かつ
wfhn(SBIndex(pr3(id), h2))≠nil のとき，
1≦s1≦Slot#(pr1(id)) かつ，
1 ≦ s2 ≦ Slot#(pr1(hn(SBIndex(pr3(hn(id)),h2))))であるときのみpr4(wfhn'(id)) = pr4(wfhn(id))∪{(h1, s1, h2, s2)}.
以上の条件を満たさない場合は， wfhn' = wfhn. ■

さてこの［定義1］から［定義5］までで得られるホルダ・ネットワークと，［定義1］から［定義4］まではそのままで［定義5］の代わりに［定義6］を用いて得られるホルダ・ネットワークの関係について考察する．後者によって得られるものは前者の条件を満たしている．また前者の条件を満たすホルダ・ネットワークのうちでスーパーホルダテーブルやサブホルダテーブルの順列を換えることによりできるホルダ・ネットワークをすべて同じものであるとすれば，なんらかの後者の演算の系列が存在して，それを構成することができることが証明できる．このデータ構造は計算機で扱うためのものであるので，以後後者において導入した演算を基にした形式化を中心に話を進める．またその他に有限時間内に計算できる演算を定義することによりホルダ・ネットワークにより実現されるアイディア・プロセッサの機能を形式化する．

［定義7］以下の演算を定義する．

GetL:WFHN×ID→L∪{nil}.
特定のホルダのパラフレーズ・リテラルを得る．

if wfhn(id) = nil then GetL(wfhn, id) = nil
else GetL(wfhn, id) = pr1(wfhn(id)).

Unhold:WFHN×ID×N→WFHN.
以前に行われた特定の1回のHold演算により起こった変化を破棄する．Unhold(wfhn, id1, n1) = wfhn' として，
if pr4(id1)∋(a, b, c, d) s.t. a or b = n1
then wfhn'(id) = wfhn(id) else 以下のとおり

∀id∈ID. s.t. id≠id1, SBIndex(pr3(id1), n1).
wfhn'(id) = wfhn(id).
GetL(wfhn', id1) = GetL(wfhn, id1).
GetL(wfhn', SBIndex(pr3(id1), n1))) =
pr1(wfhn(SBIndex(pr3(id1), n1))).
pr2(wfhn'(id1)) = pr2(wfhn(id1)).
pr3(wfhn'(SBIndex(pr3(id1), n1))) =
pr3(wfhn(SBIndex(pr3(id1), n1))).
pr4(wfhn'(SBIndex(pr3(id1), n1))) =
pr4(wfhn(SBIndex(pr3(id1), n1))).
pr2(wfhn'(SBIndex(pr3(id1), n1))) は
pr2(wfhn(SBIndex(pr3(id1), n1)))の出現を1回を省いたもの．pr3(wfhn(id1)) は pr3(wfhN'(id1))の第n1番目の要素を省いたもの．
pr4(wfhn'(id1)) = {compact((a, b, c, d)) ¦
(a, b, c, d)∈pr4(wfhn(id1))}

ただし演算 compact は，a, c のうちで n1 より大きいものを1づつ減らすものである．

De-unify:WFHN×ID×N×N×N×N→WFHN.
現在より前に行われた特定の1回のUnify演算により起こった変化のみを破棄することである．De-unify(wfhn, id, n1, n2, n3, n4) = wfhn' として，
pr4(wfhn'(id)) = pr4(wfhn(id)) - {(n1, n2, n3, n4)}.
ただしここで用いた集合の引き算は次のように定義されているものとする．

A - B⇔ A∩¬B.

Remove:WFHN×ID→WFHN.
特定のホルダについて{l, e, e, {}}に到達するまで演算Unholdと演算De-unifyを繰り返し，そうなったらそのホルダをnilにする．すなわち wfhn'(id) = nil とする．

Supers:WFHN×ID→ID*.
特定のホルダのスーパーホルダテーブルを返す．

Supers(wfhn, id) = pr2(wfhn(id)).

Subs:WFHN×ID→ID*.
特定のホルダのサブホルダテーブルを返す．

Subs(wfhn, id) = pr3(wfhn(id)).

WhichAttached:WFHN×ID×N×N→$2^{N\times N}$.
WhichAttached(wfhn, id, n1, n2) = {(id1, s1), ... , (in, sn)} として，nil でないホルダ wfhn(id) で Subs(wfhn, id) 内の第n1番目のホルダの第n2スロットが Subs(wfhn,id) 内の何番目のホルダの何スロットとユニファイしているかを返す。n1, n2 が0の場合もありえ正しいが、インデックスの範囲を超えるようなときは空集合を返す。
WhichAttached(wfhn, id, n1, n2) = {(n, s) ¦
(n1, n2, n', s'}∈pr4(wfhn(id)) が存在するなら、
(n', s') = (n, s)
(n', s', n1, n2}∈pr4(wfhn(id)) が存在するなら、
(n', s') = (n, s)}. ■

この定義中で演算 Unhold, De-unify, Remove により返されるホルダ・ネットワークは well-formed であることは証明される．筆者らが[4]で与えたアイディア・プロセッサの機能とは，(1)ホルダ・ネットワークというデータ構造にデータを格納する，(2)ホルダ・ネットワーク中のパラフレーズ・リテラルに対しての入力，編集及び検索を行なう，(3)ホルダ・ネットワーク中のパラフレーズ・リテラル以外の情報（すなわちホルダ・ネットワークが保持する構造情報）に対しての入力，編集及び検索を行なう，(4)構造情報の解析をして(2)及び(3)の支援をする，ことであった．以下でこの機能を形式化する．

［定義8］アイディア・プロセッサの機能は以下の演算により定義される．

Generate:WFHN×ID→WFHN,
GetL:WFHN×ID→L∪{nil},
SetL:WFHN×ID×L→WFHN,
Hold:WFHN×ID×ID→WFHN,
Unify:WFHN×ID×N×N×N×N→WFHN,
Unhold:WFHN×ID×N×N→WFHN,
De-unify:WFHN×ID×N×N×N×N→WFHN,
Remove:WFHN×ID→WFHN,
Supers:WFHN×ID→ID,
Subs:WFHN×ID→ID,
WhichAttached:WFHN×ID×N×N→N×N. ■

リテラルLに対する編集のための演算はここでの定義に入れてはいないが，第一要素を取り出す，それ以外を取り出す，リテラルの長さを計算する，二つのリテラルを連結するといった演算が可能であると仮定している．またXに対しても符号の順序比較や等号演算ができることは仮定している．