Vol.2019-DC-113 No.3
2019/7/17

ドキュメント操作のアルゴリズムとデータ構造に関する一考察

大野 邦夫† 木村 登志子††

本報告では従来のドキュメント操作の基本とされた論理構造とレイアウト構造の関係付けという観点を掘り下げて、論理構造の基本である文字と文字列の操作、その総体としての文章の作成編集操作に関して検討を試みる。文章の作成編集操作は、 XeroxPARCのStuartCardがGOMSモデルを提唱しているが、そのモデルはコンピュータとの対話モデルとして検討、評価された。ここでは、その実装モデルをプログラミング言語C及びLISPの文字列処理に関係付けて考察すると共に、プログラミング言語における型、関数、変数の概念をオブジェクト指向プログラミングにおけるクラス定義並びに継承の枠組みに拡張し、オブジェクトモデルにおける要求・応答の操作を自然言語における語彙・文法の枠組みに関係付け、意味伝達の原始的概念モデルの検討を試みる。

キーワード：文字列操作, 文書構造, DOM, フレーム, IDL, オブジェクトモデル, SNA, 言語行為

A Study on the Algorithm and Data Structure of Document Operation

Kunio OHNO† Toshiko KIMURA††

Though the basic operation of the conventional document is the transformation of document from logical structure to layout structure, we have considered the basics of logical structure, which is the creation and editing operations of sentences with operation of characters and character strings. Sentence creation and editing operations are considered as a dialogue model with computer named GOMS by Stuart Card of Xerox PARC. The implementation model of character string processing is reviewed through the programming language C and LISP, and the concepts of type, function and variable in programming languages have been extended to the framework of class definition and inheritance mechanism in object−oriented programming language. Then the vocabulary and grammar framework in natural language can be related to the request and response operation mechanism of object oriented model. Finally, the primitive concept of semantic communication model has been considered.

Keywords：Character string operation, Document structure, DOM, Frame, IDL, Object model, SNA, Speech act

1. はじめに

本DC研の端緒であるテクニカルコミュニケーション研究グループを四半世紀前に設立した関係者として考えさせられるのは、情報処理分野におけるドキュメント研究の現状はこれで良いのかということである。同研究グループは、1996年にデジタルドキュメント研究会として情報処理学会の正式な研究会として発足し、1990年代末から2000年代前半はXMLの標準化を通じたWebの進展を背景に情報処理学会においても注目される研究発表が多かった。しかし2000年代後半、さらに2010年代以降は、研究内容の深化や新たな方向性の展開に乏しいのではないだろうか。

そのような懸念に関しては、9年前に”情報処理学会50年の歩み”の中で遠回しに指摘したが[1]、その懸念は払拭されずにむしろ深まりつつある。ドキュメントコミュニケーション研究会自体、今後の社会への真摯な貢献を目指すならば、何のため誰のために何をなすべき研究会なのかを心ある関係者は原点に立ち返って自問自答して頂きたいと思う。これはDC研だけでなく日本の多くの研究組織に該当する課題でもあろう。日本の研究が世界から取り残されつつある状況は組織運営が自己目的化して自由な議論の雰囲気が失われていることに起因すると思われる。ドキュメントの基礎に立ち戻った議論が必要であろう。

最近のAIやIoTの進展に対して、ドキュメントが果たす役割は小さくはない。AIやIoTの結果が人間の視覚に関係するならその多くはドキュメントを通じて認識される。それをドキュメントの論理構造やレイアウト構造の問題として処理してしていたのが2000年代前半までのSGML、HTML、XMLと関連プログラミング言語アルゴリズムでの検討レベルであった。

論理構造とレイアウト構造に関して、さらに前者を後者に変換する文書編集手法が主眼になされてきた背景には、人間が視覚を通じて認識するドキュメント体裁の品質がドキュメントの価値として評価されたためであった。文字、図形、画像を包含する文書構造とレイアウト構造に関しては、XeroxPARCのALTOに端を発するDTPシステムが登場して以来、コンポーネントウエアとしての複合文書（Compound Document）が技術的に検討され商品化された。さらにその相互運用性に関しては標準化団体であるOMGとW3Cの両者により標準化が試みられた。だがその標準化はマイクロソフトやグーグルのような既存の強者の技術に包含・吸収されて現在はHTML5に集約された[2]。さらに論理構造からレイアウト構造への変換は概念的に代数学的な操作に還元し得るものである[3]。

他方ドキュメントのデータ構造に関しては、論理構造が注目され、マークアップ言語としてのSGMLやXMLが使用されてき

†(株) モナビITコンサルティング
Monavis IT Consulting Co. LTD.
††横浜商科大学
Yokohama College of Commerce

た。意味的な論理構造をドキュメント体裁の品質を支配するレイアウト構造に変換する操作としては、SGMLに対するDSSSL、XMLに対するXSL・CSSが標準化された。この分野に関しては応用を目指して検討が試みられており、昨年技術的な総括を試みたが[3]、残された課題として自然言語に関連する文章の意味レベルの検討を挙げた。本報告では、その端緒として文章レベルの操作アルゴリズムとデータ構造に関しての初歩的な検討を試みる。

2. ドキュメント操作

2.1 レイアウト構造と論理構造

従来のドキュメント操作は、先に述べたとおり論理構造からレイアウト構造への変換が主要なテーマであった。その経緯と概要は、昨年のLOIS研との合同研究会で紹介した[3]。論理構造からレイアウト構造への変換は、$Y=F(X)$という代数学的な演算で概念的には代表される。概念的というのは、厳密な数学的ということではなく、コンピュータによるアルゴリズムで引数としての具体的なデータを伴ったプログラムにより実現されることを意味する。なおYはレイアウト構造のデータ集合であり、Xは論理構造のデータ集合である。

より厳密には、$Y:\rho=F(X:\sigma)$と記述され、ρはページ、コラム、行のようなレイアウト構造の属性集合、σは章、節、項のような論理構造の属性集合である。この発展経緯については、大ざっぱではあるが5年前に報告した「アプリケーション仮想化環境における複合文書」の研究報告で紹介した[4]。XがSGMLの場合は、YがDSSSLに対応し、XがXMLの場合は、YはXSL−FOに対応する。その中間のピボット的なフォーマットとしてHTMLに対応するXSLTがある。XSL−FOが煩雑なために、その製品がアンテナハウス社のフォーマッタしか出現しないのに対し、HTMLはCSSに基づきXSL−FOとは似て非なるレイアウト標準を定めており、HTML5が主流となった今日ではこのレイアウトが事実上の標準となり、スマホからPC、さたにはインターネットTV、デジタルサイネージまでを統合するデファクトスタンダードになっている[5]。

ビジネス文書や技術文書の世界は、DTPシステムやマイクロソフト・オフィスのような独自フォーマットが使用されている。DTPシステムには、アドビのInDesignやFrameMaker等があるが、以前のような熱気は感じられない。印刷出版ビジネスが衰退気味なので新技術の開発は行われず、趣味や骨董品のようなニッチな領域になりつつあるのが実感である。この分野に関しては、私見であるがハイエンド製品であったInterleaf社のDTPシステムが最も完成度が高かったと思われるが、現状ではかつてのドキュメント遺産のサポート以外には使用されていない。だが、この製品は要素モジュールをC言語で実装し、ユーザインタフェースを中心とする全体のシステム操作をLisp言語で行う高性能できめ細かい処理を可能としたシステムであった。

2.2 文章の処理

論理構造とレイアウト構造に関する操作は、SGML、XML、DTPシステムの技術で実現され終わったかもしれないが、情報の意味や人間の知識が関係する意味構造とも言うべき文章の処理が課題として残されている。この課題については、昨年の報告の最後に残された課題として提起した[3]。そこで文章の作成と編集の操作に関して検討を試みる。

文章の作成目的の一つは、アイデアを記録することである。アイデアとは言っても、個人的なメモレベルでの創造的な内容や日記のような記録、キーワードだけを記した手帳のメモ等のバラエティがあり、状況や目的、個々人の性格や執筆スキルなどにより多種多様である。アイデア記録以外の文章の作成目的は、他者に情報を伝達するためである。情報の伝達も、文章を作成するためにはアイデアを必要とする。客観情報の伝達であれば、5W1Hのような新聞記事の属性のような情報が要求される。常識として知られていない語彙を知らせるためには、その説明が要求される。このような意味情報を伝達させるには、相手が分かるように順序立てて語彙を配置して論理的に総合が取れるように作文する必要がある。相手が具体的に特定される場合と、不特定な場合とではその文章は異なる。さらに目上、目下といった関係を意識する場合は敬語やていねい語の使用が要求される。従って文章を作成するためには、背景文化や状況に応じた文脈を意識する必要が生じ、文章作成は極めて知的なアイデア伝達行為なのである。

さらにアイデアは詩的な感性的、経験的な場合と、科学的論理的な記述に大別される。論理的な記述は、演繹的な記述や帰納的な記述の他に、仮説を検証する記述が用いられる。以上の論理記述では、多くの場合、命題や語彙を要素還元して論証する。弁証法により矛盾を止揚するような記述もあるが、これは論理だけでなく、経験的、詩的な側面を含む。

執筆力は人間の才能にとっては非常に重要である。その背景には論理的思考能力、多様な知識と経験、美的感性、倫理的人格などの要因が関係するが、それが実践に結びつく内容でなければ机上の空論となる。

2.3 編集処理

いったん作成した文章は多くの場合、推敲され編集される。編集作業は、一人で行う場合とグループの場合があり、グループの場合は全体を統括する責任者が必要になる。電子化以前は原稿用紙に朱を入れる操作が編集文化として存在し、その専門家が知識人の代表のような人物として存在した。特に新聞や雑誌の編集長、編集員や論説委員、論説主幹といった人々である。新聞、雑誌の編集責任者は、編集方針が世論や政治的意見に関与するので、ある種の権力を持つが、編集は執筆と同様に、否場合によってはさらに大きな社会的影響力を持つ。

だがワープロが発明されて以降は、編集操作は電子化されて、コンピュータ・ツールで処理可能となった。その結果文書作成、編集は一挙に効率化されたが、電子的な編集だけでは履歴が残らないのは問題なので、印刷して朱を入れる時期もあった。その後、ワープロやDTPがワークフロー管理を行うグループウエアと結びついて、履歴情報の管理を行うようになったが、必ずしも普及はしていない。多くの場合、履歴管理ツールの操作が煩雑であり、決して使い勝手の良いツールとは言えないためである。その背景には、個人のアイデアによる文書作成が個人の責任で行えるのに対して、グループ作業は多くの意見の調整を伴うのでどうしても煩雑になることが挙げられる。この問題は民主主義の抱える本質的な問題で、人間社会のあり方に関与するので簡単な解が出せない重要な問題である。

2.4 GOMSモデル

編集操作は、歴史的に検討されモデル化されている。その代表は、Xerox PARCにおけるStuart Card 等による検討である。その基本モデルは、GOMSモデルと呼ばれる[6]。このモデルは、編集操作だけでなく、ユーザがコンピュータを操作する知的なプロセスを分析する。そのような操作において普遍的に観察される手順のステップは、以下の4段階の操作に大別される。それらは、(1)一群の目標（Goal）の設定、(2)一群の操作（Operator）抽出、(3)目標の実現手段（Method）抽出、(4)目標実現手段が競合する場合の選択（Selection）規則、の4ステップで、各々のキーワードの頭文字を取ってGOMSと呼ばれた。

文献では、具体的な事例としてPOETという古典的な行エディタを用いて編集操作を行う事例を紹介している。その状況は、図1のように記述されている。先ず最初の目標

```
GOAL: EDIT−MANUSCRIPT
    GOAL: EDIT−UNIT−TASK      repeat until no more unit task
        GOAL: ACQUIRE−UNIT−TASK
            GET−NEXT−PAGE      if at end of manuscript page
            GET−NEXT−TASK
        GOAL: EXECUTE−UNIT−TASK
            GOAL: LOCATE−LINE
                [ select: USE QS−METHOD
                          USE−LF−METHOD]
            GOAL:MODIFY−TEXT
                [ selet: USE−S−COMMAND
                          USE−M−COMMAND]
                VEIY−EDIT
```

図1　テキストエディタによる編集操作のGOMSモデル

が文書の編集（Edit−Manuscript）である。次のサブゴールは、ユニットタスクの編集（Edit−Unit−Task）である。ユニットタスクは、無意識的に行える一連のタスクであるが、要するにユニットタスクが無くなるまではこのサブゴールが継続する。次のサブゴールは、ユニットタスクの把握（Acquire−Unit−Task）である。このレベルでは仕事が終わりかどうかの判別を行う。終わりであれば、別のタスクを行う（Get−Next−Task）。そうでなければ、新たなサブゴールとしてユニットタスクを実行する（Execute−Unit−Task）。先ず、対象となる行にカーソルを移動させ、その行が行末でなければ、文字列挿入を挿入し、行末であればLFを選択する。次のサブゴールは、テキストの変更（Modify−Text）である。その場合は、置換（Substitute）と修正（Modify）が選択肢として存在する。その後は編集結果の確認（Verify−Edit）を行うというものである。

GOMSモデルは、人間がコンピュータに対して一連の業務を遂行する際の普遍的なモデルを想定しているが、それは先ず定められた業務（Task）があり、それを遂行するための一連の操作（Operator）の予測が行われ、それらの操作の具体的な手段（Method）が考案され、手段が競合する場合には選択（Selection）が行われるという階層モデルである。

このモデルは、人間の行動が、目標を決めて、そのためのサブゴールを設定し、さらに選択肢があればそれを評価ルールの下に選択するという、知的な活動をシミュレートしていると言える。ゴールからサブゴールの設定は、演繹的なプロセスであり、選択はデシジョンテーブルやエキスパートシステムの推論に対応する。プログラミング言語における制御構造とも類似であり、機械的繰り返しのfor、条件分岐で繰り返すwhile、一度実行して条件繰り返し設定するdo〜whileなどを彷彿させる。

2.5 キーストロークレベルモデル

GOMSモデルは、抽象モデルであるが、このモデルに基づくキーストローク・レベル・モデルはユーザインタフェース評価で実用的に使われた[7]。そのために、K（打鍵操作〜0.2秒）、P（マウスによる指示〜1.1秒）、H（マウス・鍵盤の手の移動〜0.4秒）、M（メンタルな思考〜1.35秒）という代表操作時間を設定し、マウスとキーボードによる一連のタスクの操作時間を見積もり、その操作時間でユーザインタフェースを評価する。ゴールのタスクをGOMSモデルで分析し、それをキーストローク・レベルまでブレイクダウンさせて具体的な数値に換算してその比較でユーザインタフェースを評価する。このモデルを使用して、Five−keyマウス、ELISのキーボードについて操作性を評価したことがある[8][9]。

キーストローク・レベル・モデルは、ユーザインタフェースの評価だけでなく、作業量の評価・見積もりにも使用可能である。その観点で、文書のデジタル化を、文字コードとして入力するか、スキャナのデータでイメージのままで入力するかの場合をいくつかのレベルで評価したことがある。これは作業見積もりに活用できるので概算評価のためには有効であった[10]。

3. 文字列操作

3.1 プログラミング言語

文書編集操作における文字列操作をプログラム実行の環境から見ることがツールの性能や、拡張性、柔軟性の観点において重要である。プログラミング言語のデータ型の観点で、文字列は、文字データの配列として扱われる。プログラミング言語は、整数、浮動小数、文字、文字列、配列、構造体といったデータ型を用意して、変数や定数はそれらの型に従って管理される。文字列を変数に代入する際

は、文字列が操作されるメモリ空間の先頭番地がその変数の値に代入される。メモリ番地に直接値が書き込まれる場合と、新たなメモリ空間にコピーとして書き込まれる場合があるが、それはケース・バイ・ケースである。

文字データは、1バイト系のASCIIから始まり、2バイト系のJIS6226、さらに2バイト系を改訂・拡張したUNICODE、UNICODEを包含したISO10646へと発展した。その結果、最近のプログラミング言語は、アルファベット以外の漢字やハングル、アラビア文字、インド文字などたいていの文字をコード的には扱えるようになっている。コンピュータによる文章操作は、文字列を文章の要素として扱うことであるがそれを典型的な2種類のプログラミング言語に関して検討する。

3.2 C言語

文字列の基本的な扱いはC言語で考察すると原理的に分かりやすい。C言語では文字列を配列・ポインタで処理することが可能である。例えば、

char a[]=”Ohno”;

とするとaという文字型の要素からなる配列として”Ohno”という文字列が定義される。この場合はa[0],a[1], a[2], a[3]が、’O’, ’h’, ’n’, ’o’というcharデータとして定義される。同様に、

char *p =”Kimura”;

とすると、ポインタデータにおけるp[0], p[1], p[2], p[3], p[4], p[5]は、’K’, ’i’, ’m’, ’u’, ’r’, ’a’となり、pは先頭文字’K’へのポインタとなる。C言語における string.h というヘッダライブラリに、文字列長を返すstrlen(), 文字列コピーのstrcpy(), 文字列結合のstrcat(), 文字列比較のstrcmp()といった文字列操作を行う関数群が定義されており、実装における効率的な文字列操作が可能となっている[11]。

データ型の概念は先の報告[3]で説明したが、その実体はC言語では記憶領域の管理の問題に還元される。例えば文字型と整数型は意味的には異なるが、ASCIIの文字型は、実際には整数型として扱う。例えば、char a =’A’ とchar a=65はC言語のプログラム的にはまったく等価であり、文字型は一種の幻想のようなものである。これは英数字の文字集合が7bitで区別されるので1バイトの整数型として処理しているのであるが、それを人間の視覚は異なる文字として識別する。コンピュータの論理演算的な区分と人間の感覚経験的な区別は、本来異質なものであるが、データ型はそれを実用的に融合しているのである。

3.3 Lisp

文書の編集作業は文字の挿入削除を自由に行わせる必要がある。そのためには、Lisp言語の基本データ構造であるセルは柔軟で自由度が大きい実装技術を実現している。Lispのセルは、car部とcdr部の対である。carは、Content of Address part of Registerの略、cdrは、Content of Decrement part of Registerの略で、car部にデータ、cdr部に後続するデータのアドレスの値、すなわちC言語におけるポインターが格納される。文字の追加は、新たに挿入される文字のアドレスを、挿入される直前のcdrのアドレスに挿入し、挿入された文字群の最後のcdrのアドレスに、追加のために書き換えられた元のアドレスを挿入すれば良い。逆に削除する場合は、削除される文字の一つ手前の文字のcdrアドレスを、削除される文字群の次の文字のアドレスに書き換えれば良い。削除されて不要になったセルは、ガベージコレクション処理のプロセスにより回収され新たなセルとして再利用される。

以上の例は、文字をセルのデータとして扱う古典的・原理的なLispの場合で、現実には文字列データはC言語の場合同様に文字の配列としてセルとは異なるデータ領域で管理される。Lispは柔軟なのでC言語よりも多様な文字列操作が可能である、例えば文字列の等価性を検証するCommonLisp関数として、string=とstring−equalがあり、前者は大文字、小文字を区別するが、後者は区別しない。string−trimという関数は、対象文字列から指定文字を削除する。string−upcaseは対象文字列の小文字を大文字に変更した文字列を返し、string−downcaseは、逆に大文字を小文字にした文字列を返す。string−capitalizeは、文字列の最初の文字を大文字にし、それ以外を可能な範囲で小文字にした文字列を返す[12]。

このようにして、文字列を柔軟に処理することが可能である。なお以上のような処理は、C言語でも不可能ではないが、データ型を伴う変数や関数の定義、メモリの効率的な使用のための配列定義や不要になったデータの回収などを考慮するとLispの柔軟性、拡張性、生産性には及ぶべくもない。高機能テキストエディタとして有名なEMACSは元々Lispで開発された。その後処理性能の改善や汎用性、移植性を考えて、GNU EmacsはCに書き換えられ、モジュール化されたが、柔軟性、カスタマイズを可能とするために、C言語のモジュールを呼び出して処理可能とするEmacs−Lispがサポートされている[13]。

同様のDTPシステムとしてはInterleafが存在する。これはGNUEmacsが文字だけのテキストエディタで実現した手法を文字、図形、画像を含むDTPの世界で実現したものである。個別の機能はC言語で実現され、そのモジュールをInterleaf−Lispで処理可能としている。InterleafのDTPシステムは、章、節、項の番号を自動的に付与したり、図番、文献番号を一元的に自動的に付与するオートナンバー、オートリファレンス機能、文書レイアウト属性を、別の文書に自動的に対応させるブック・カタログ機能、目次や索引の自動生成機能など、他のDTPシステムが実現していないか、専用のアプリケーションで実現している機能を、Lisp言語レベルで実現していた[14][15]。四半世紀前の製品であるが、Interleafに匹敵するDTPシステムが、その後出現していないのは淋しい限りである。

3.4 DOM

文章の論理構造は、木構造であり、そのデータのアクセスには独特のインターフェースを必要とする。木構造へのアクセスは、先ずルートを掴み、子のノードを辿ることになる、そのようなAPIの関数がInterleaf−Lispには、get−first−child, get−last−childのように用意されていた。同列のノードを移動するためには、get−previousget−nextが用意され、上位のノードに戻るには、get−parentという関数が用意されていた。論理構造だけでなくレイアウト構造も木構造なので、上記の関数群はInterleaf−Lispにおいては、レイアウト構造のトラバースにも適用された。

この操作は、SGML、XMLでも同様である。1990年代後半に、INSエンジニアリングで開発したSGMLデータブ

レード並びに、SGML/XMLデータカートリッジにおいても、上記の木構造の操作はC言語のミドルウエアで実装された[16]。XMLのW3Cによる標準化に当たり、木構造の操作はDOM（Document Object Model）として仕様が制定された。この仕様は、言語独立とするために各種言語へのマッピングが可能なCORBAのIDL（Interface Definition Language）が用いられている。IDLは、言語仕様とは言っても、文法的な概念は、引数を伴った操作の結果を引数のデータ型と命令の名称に関して決めただけである。むしろ重要なのはIDLの型（IDL Type）である。IDL型は、C言語のデータ型を参照して決められたが、それはCORBAを最初に制定した1990年当時、C言語がネットワーク関連の基本的なアプリケーションで使用されていたからである。各種言語へのマッピングは各種言語がサポートするデータ型にIDL型を対応付けることであり、かつIDLのインタフェース、オペレーションの名称は、個々の言語のインタフェースやクラス、手続き名や関数名に対応付けられる。従ってDOMをIDLで定義しておけば、IDLマッピングが定義されているプログラミング言語によるXMLデータへの木構造的なアクセスが可能になるのである。CORBAは、肝心のクライアントサーバシステムの相互運用では、基幹系の一部を除いては殆ど使用されなかったが、XMLやHTMLのDOMとして活用されたのは興味深い。技術というものは、本来の狙いとは別の領域で活用されるという最近の具体例である。

3.5 代数学的演算

CORBAのIDLの本質は、データ型とオペレーション・シグニチャである。後者は、$y:\rho = F(\chi:\sigma)$と記述可能であるが、具体的には、下記のような演算として表現できる。

$$F：（\chi_1:\sigma_1,\ \chi_2:\sigma_2,\ \cdots\ \chi_n:\sigma_n）$$
$$\rightarrow（y_1:\rho_1,\ y_2:\rho_2,\ \cdots\ y_m:\rho_m）$$

ただし、*F*は演算の名称、χは型σのリクエストパラメータ、yは型ρの結果のパラメータである。パラメータ数nは1以上、mは0以上である。

オペレーション・シグニチャは代数学的な演算そのものである。これはプログラミング言語が、型を規定した引数の操作を、演算結果の型と値に関係付けるという概念であり、言語が意味を伝達する媒体であると考えるなら極めて示唆的な内容を含む。CORBAのIDLにおけるオペレーション・シグニチャはクライアント・サーバ系のリクエスト・レスポンスの抽象化である。この仕様に基づいて、各種のプログラミング言語によるリクエスト・レスポンス記述の相互運用を図るというのだから画期的であった。しかし画期的、革命的な企画が成功する機会は乏しい。結局XML、HTMLのDOMとして残存したに過ぎなかった。

4. 会話と文章

4.1 SNA

CORBAのIDLにおけるオペレーション・シグニチャは、自然言語においては文章記述の世界よりは、インタラクティブな会話の世界に適合する。多くの命題的表現を、接続詞や時制を用いてその整合性を通じた内容理解を要求する文章よりは、単一の命題を文としてやりとりする対話の方がオペレーション・シグニチャに対応するからである。なお、自然言語における会話操作の発展の枠組みとしては表1で示すソーシャルネットワーキングアプローチ（SNA）が有効と思われる。

SNAに関しては、昨年の9月のDC研の研究会で英語スキルの獲得過程に関係付けて紹介した[17]。SNAの枠組みは表1の枠組みで考えることができる。表1で示すとおり、SNAのパラダイムは、横方向に「わかる」、「できる」、「つながる」、縦方向に「言語」、「文化」、「グローバル社会」という展開パターンを用意しているが、これは、図2に示すノーマンの認知モデルとも適合すると思われる。

表1　SNAにおける3×3のマトリクス内容

	わかる	できる	つながる
言語	自他の言語がわかる	学習対象言語を運用できる	学習対象言語を使って他者とつながる
文化	自他の文化がわかる	多様な文化を運用できる	多様な文化的背景を持つ人とつながる
グ社会	グローバル社会の特徴や課題がわかる	21世紀型スキルを運用できる	グローバル社会とつながる

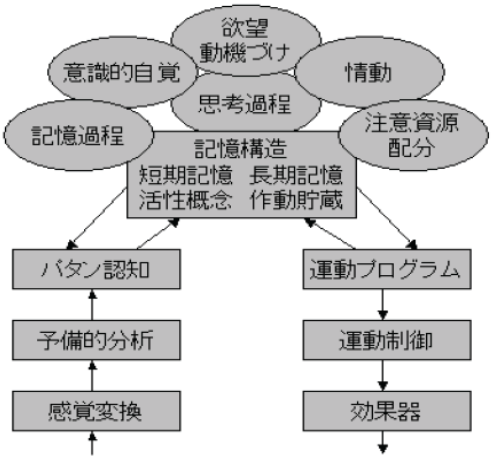

図2　ノーマンによる人間の認知行動モデル

ノーマンの認知モデルにおいては、視聴覚のような感覚情報をパターン認知して記憶・思考過程を経て、自覚、情動、欲望・動機付けといったプロセスを経て、運動プログラム、運動制御といった実践的なプロセスに移行する。この感覚情報から思考を経て実践に至るプロセスは、「分かる」から「できる」への展開である。さらに「できる」段階が、組織的・社会的に展開することにより「つながる」段階へと移行する。他方、言語、文化、グローバル社会という展開は、言語行為論と言語コード論が適合する。

4.2 言語行為論

言語行為論（Speech act theory）は、言語を対話する相手に行動を起こさせる媒体として位置づけるもので、状況に応じた外面的なプロトコルとして分析する手法で、発案

者のオースティンは、言語行為を、「約束」「任命」「警告」「宣言」に分類した。この内容に関しては、2015年10月に開催されたDC研の研究会で紹介した[18]。この概念を、人工知能間の対話モデルとして拡張したのが、KQMLやFIPAのACL（Agent Communication Language）である。SNAにおける「つながる」レベルの関係は、ACLの観点で考えると興味深い。KQMLは、言語行為に相当するエージェント間の通信カテゴリを表2のように分類し[19]、通信行為（Communicative Act）と呼んだが、SNAの「つながる」という対話状況をさらに分析すると、「約束」「任命」「警告」「宣言」のレベルを超えた通信行為に近いカテゴリが得られると思われる。

表2　エージェント通信言語KQMLにおける通信行為（Communicative Act）のカテゴリ分類

Category	*Name*
Basic query	evaluate, ask-if, ask-about, ask-one, ask-all
Multi-response (query)	stream-about, stream-all, eos
Response	reply, sorry
Generic informational	tell, achieve, cancel, untell, unachieve
Generator	standby, ready, next, rest, discard, generator
Capability-definition	advertise, subscribe, monitor, import, export
Networking	register, unregister, forward, broadcast, route,

4.3 言語コード論

言語コード理論（Speech Code Theory）[20]は、その名前で呼ばれる以前にコミュニケーション分野における民族学（Ethnography of Communication）としてフィリプセン（Gerry Philipsen）により提案された。その目標は、コミュニケーションと文化の関係を言語的な背景で捉える理論を開発することにあった。従ってオースティンの言語行為に関する理論の一つの分野を形成すると言える。このアプローチの背景には、言語は文化そのものであり、意味的価値観を本来的に伴う手段であるという考え方が存在する[18]。別の言い方をすると、言語の意味は、「真・偽」という命題論理的な区別ではなく、「適・不適」という集団の文化的価値観が重要という指摘である。ということから異なる言語の対応語彙に正解はなく、翻訳も近似でしかないということになる。従って言語理解には、文化理解が必要であることを意味するのである。以上はSNAにおける言語→文化の対応の妥当性を物語る。さらに、多様な文化の坩堝であるグローバル社会への参与は、とてつもないチャレンジと思われる。SNAの文化、グローバル社会への展開は、以上のような背景を考慮する必要がある。

4.4 自然言語とクライアントサーバ方式

異なる自然言語で翻訳を通じて意味伝達が可能ということは、意味伝達の本質は文法にではなく語彙関係にあることを物語る。さらに語彙関係が形成する意味が操作名（演算名）と型を伴った引数にあると考えれば、自然言語を通じた発話、行為は、クライアントサーバ・システムにおける、リクエストとレスポンスでモデル化可能ではないかという発想が生まれる。さらに多様なプログラミング言語によるリクエストとレスポンスが、CORBAのIDLによるシグニチャで定式化可能であるならば、自然言語による意味伝達とプログラミング言語によるリクエストとレスポンスは、CORBAのIDLによるシグニチャに還元可能なことを示唆する。

ところで、オペレーションシグニチャーは、要求としての型が示された引数が型を伴う結果を返す名称を持った演算であることに他ならない。演算としては、数値演算と論理演算が常識の範囲内であろう。数値演算は数学でお馴染みであるが、論理演算は、命題論理としての、AND, OR, NOT,IF~THEN などと、述語論理における全称命題と存在命題などがあり、IDLのシグニチャは、これらの組み合わせになると考えられる。その基本的なモデルは、ミンスキーのフレームの枠組みで考えることが可能であろう。

4.5 ミンスキーのフレーム

ミンスキーのフレームは、Lisp言語の属性リストの一般化、入れ子になった連想リストという古典的な概念であるが[21]、抽象データ型やオブジェクト指向のクラスはミンスキーのフレームのコンピュータにおける実装と考えられる。さらにSGMLやXMLも、その入れ子になった複合的な概念であり、RDFやOWLのクラスもその部分的な実装と言える。オペレーションシグニチャーが記述する内容は、基本的にフレームの属性に対する値の代入や変更である。この関係は、文書を包含する自然言語操作への、データ構造と意味に対しては概略図3のように考えられる。こ

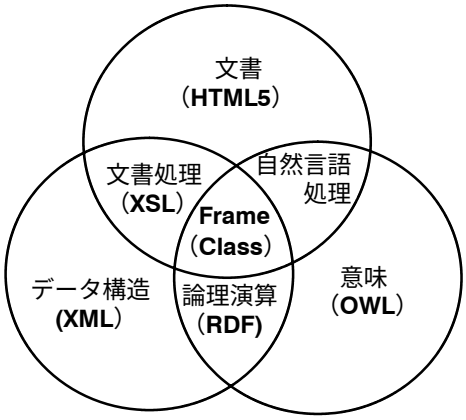

図3　文書・データ構造・意味の関連

の概念を、XMLの世界の具体的なボキャブラリで括弧内に記述したが、具体的な内容記述にするとゲーデルの不完全性定理により適用範囲が狭まるので一般性は減じられる。

なお、短文のやり取りの会話であれば、文書のウエイトは低下して、意味とデータ構造との関係になり、その中間に論理演算とフレームが介在する形式になる。

4.6 UML、オブジェクト、オントロジ

ミンスキーのフレームは、語彙の意味概念であり、意味ネットワークを形成する。そのネットワーク構造は。、個々の語彙相互間がis−a、has−a等の関連により形成される、関連を属性として扱うことが可能であれば、オブジェクト指向プログラミングにおけるクラス図として構築可能である。そのようにしてクラス図を描くことができれば、その関連のドメインをコンピュータとして意味的に扱うことが可能になる。この観点でのシステム構築は、オブジェクト分析・設計の世界が扱う分野であり、最近も日本酒醸造分野を取り上げてUMLで分析し、関連クラスの構築を試みた[22]。このようなシステムの構築は、最初から仕様を決めて実装するのではなく、作成しつつ問題点を修正してゆくプロトタイピング手法に近いのであるが、人間の知識獲得もこのようなプロトタイピング的な試行錯誤プロセスに適合すると思われる。

UMLにおけるクラス図の構築は、ユースケース図やアクティビティー図を通じて構築されるが、このプロセスにおいては語彙の意味概念が重要な役割を果たしている。それは常識に基づく分野毎の専門知識の集合論的な概念階層と概念相互間の命題論理・述語論理的な記述であろう。UMLのクラス図は、自然言語とプログラミング言語をつなぐ重要な役割を果たしているが、制約も多い。クラス概念自体、集合論という数学的な原理に支えられているように見えるが視点を変えると極めて確実な概念と信じられている数の体系ですら視点を変えると不確実な面があることを昨年の画像関連学会連合大会で指摘した[22]。

5. まとめおよび考察

5.1 まとめ

以上の検討を総括すると、自然言語における意味伝達は、クライアントサーバシステムにおける要求（リクエスト）と応答（レスポンス）の関係をモデルに扱うことが可能と思われる。このモデルは、記述される文章よりは、命題レベルの短い文章による対話に適合するが、記述される文章も、命題レベルの文章の集積と考えられるので、会話から文章への展開が課題となる。その観点から考えると、言語における文法は、クライアントサーバ的な骨子の修飾に過ぎないと見なすことが可能であろう。さらに以上の観点をSNAによる言語習得モデルに結び付けられる可能性があるので、そのアプローチに関しても検討を試みたいと考えている。

5.2 文書・概念・語彙関係への考察

さらに図3で描いた文書・データ・意味の関係についての展開を試みる。ゲーデル的アプローチの詳細化とは逆の一般化を図3に適用する。そのアプローチによる仮説モデルを図4に示す。この図では、意味を語彙として位置づけたが、語彙化されない意味もあることは承知の上である。語彙化されない意味は、複数の語彙の関係でありそれは論理演算や概念構造に包含される。従って、右下の語彙・意味のサークルは図3における意味のサークルの部分的な概念である。左下の概念構造のサークルは図3のデータ構造のサークルを概念構造としてより一般化させ拡大したもの

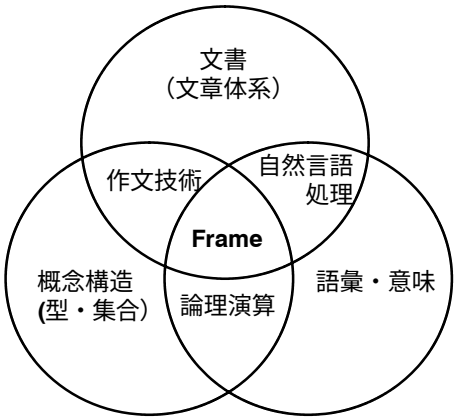

図4　文書・概念・語彙の関連

である。この図における文書の概念は、図3の文書がレイアウト化された文書実体を意識していたのに対して、むしろ文章体系としての文書概念であり、左下との共通概念は、アウトライニングや接続詞の使い方のようなレトリック的な作文技術ということになるであろう。

5.3 初歩的対話モデル

さらに図4の文章体系としての文書概念を消去した初歩的な対話や会話のモデルを図5に示す。

初歩的な対話や会話は、図5のモデルで可能と思われる。例えば、乳幼児が母親と対話するような場面を考えれば良い。食事や排泄のような初歩的な対話は、文法無しの

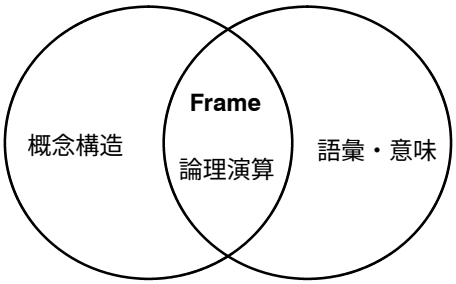

図5　概念・語彙の関連

単語の語彙レベルで行われる。言語がこのような乳幼児の発する単語レベルの発声が語彙となって発展するプロセスを図6に示すモデルで一昨年の研究会で報告した[23]。さ

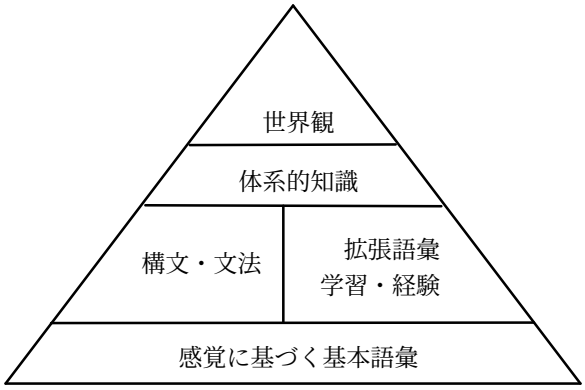

図6　人間の言語と知識階層

らにこのモデルが、図7に示すCommonLispにおけるアプリケーション構築と類似しており、プログラミング言語と自然言語の類似性についても考察した。

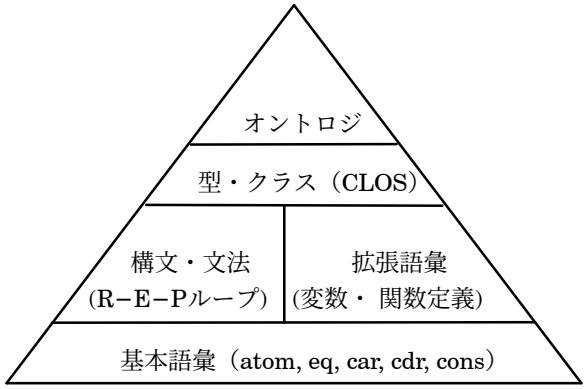

図7　Common Lisp上の意味階層モデル

この状況は、異文化の人々が最初に接触する状況での会話レベルに適用可能であろう。ピジンのような初歩的な言語は、一つの言語体系に別の言語体系が語彙レベルで交流し、両者がミックスした新たな言語体系が形成されたものである。その最初の異文化接触では、単語レベルの語彙の把握から開始されたであろうことは想像に難くない。この状況は、第二言語として最初に外国語を学ぶ際でも同様であろう。従って、この状況から語彙と文法を獲得していく状況が分析され、学習や教育にフィードバックされるべきであると考える。さらに別の次元で考えると、今日の世界で使用されている言語は、人類の歴史を通じた異文化交流の多重継承結果として出来上がったものである。

そのように考えると、最も初歩的な言語は図5のような語彙と概念関係の関係付けから形成される原始的なモデルで表され、それが長文の記述を可能とする図4のような体系になり、その最新状況が図3であると考えることができると思われる。

6. 今後の課題

以上のモデルは仮説段階なので検証を通じた具体的なブレイクダウンを行う必要がある。先ず、会話レベルについて、上記の図5による意味伝達を検証したいと考えている。要するに、文法を用いないで、動詞＋目的語のレベルで内容を把握する可能性状況の分析・検討を試みる。そのための環境要因の設定にはSNAによるアプローチを考えている。

文章レベルに関しては、既存の構文解析をベースとする複文の単文化、さらに物語展開上のレトリックの問題に還元されると思われるが、上記の会話レベルから文章レベルへの展開、その移行のモデル化の可能性を検討する。具体的には、アウトラインイング、物語的展開手法をベースに、語彙の活用法やその形容詞による修飾、接続詞、副詞の活用などに関して段階的に扱うことが可能と思われる。

7. おわりに

以上、本報告では、オブジェクトモデルにおける要求・応答の操作を自然言語における語彙・文法の枠組みに関係付け、意味伝達の原始的概念をオブジェクト指向プログラミング概念を踏まえて考察することを試みた。未だ漠とした内容であるが、SGML、XMLの時代に展望されたセマンティックWebからの進展として考えている。シンギュラリティが話題になる最近のAIは、人類の歴史的な遺産としての自然言語の語彙の蓄積に対する関与が乏しいと感じる。深層学習は、関連統計データのブラックボックスによる集積で、語彙的な検討が軽視されている。本検討は拙いながらその分野に対する一つの挑戦を試みると共に、実践的なSNAによる語学学習の活用とそれへの寄与を目指すものである。なお最後に本研究で触れたSNA分野の研究でご指導頂くノーザンコロラド大学の清水秀子教授に感謝します。

文献

[1] 大野邦夫；”情報処理学会50年のあゆみ:デジタルドキュメント”, 情報処理学会, pp269−272,（2010.11）
http://www.ipsj.or.jp/50anv/50nenshi/data/pdf/000050.pdf
[2] 大野邦夫；”複合文書の標準化経緯 − その登場から HTML5に至るまで − ”, 画像電子学会誌,Vol.47, No.4, pp.488−491 (2018.12)
[3] 大野邦夫;”ドキュメントの定義と型に関する考察”,情報処理学会研究報告, DC109−4（2018.7）
[4] 大野邦夫，新麗；”アプリケーション仮想化環境における複合文書”,情報処理学会研究報告, DD95−1（2014.10）
[5] 大野邦夫;”地域情報共有のためのデジタルサイネージ・コンテンツに関する一検討”,情報処理学会研究報告, DC112−1（2019.3）
[6] Stuart K. Card, Thomas P. Moran, Allen Newell; “The Psychology of Human−Computer Interaction”, Lawrence Erlbaum Associates, Inc. pp.139−192, (1983)
[7] Stuart K. Card, Thomas P. Moran, Allen Newell; “The Psychology of Human−Computer Interaction”, Lawrence Erlbaum Associates, Inc. pp.259−312, (1983)
[8] 大野邦夫,杉山広幸；”ユニバーサルコマンドを有するマウスの研究（その2）”, 信学技報, IE85−74（1985.9）
[9] 大野邦夫；”キーストロークレベルモデルのプログラム開発環境への適用例”, 情報処理学会「計算機システムのヒューマンインタフェース − モデル・評価・展望 −」シンポジウム（1988.4）
[10] 大野邦夫，金子哲也；”キーストローク・レベル・モデルによるドキュメント電子化ﾌ?ロセスの検討”,情報処理学会研究報告, DD4−2,（1996.11）
[11] 柴田望洋; “新・明解C言語−入門編”, SB Creative（2014）
[12] Guy L. Steele Jr.（井田昌之ほか訳）; “COMMON LISP 第2版”, bit別冊, 共立出版（1991）
[13] Richard Stallman（竹内郁雄, 天海良次監訳）; “GNU Emacs マニュアル”, bit別冊, 共立出版（1988）
[14] 大石進; ”オーバービューオブInterleaf5−Part1”, Super-ASCII, Vol.3, No.10（1992.10）
[15] 大石進; ”オーバービューオブInterleaf5−Part2”, Super-ASCII, Vol.3, No.11（1992.11）
[16] 大野邦夫；”ミドルウエアによるSGML/XML文書管理の枠組みの検討〜ネットワークとドキュメントのオブジェクト技術による融合”, DD−Symposium’98資料（1998.1）
[17] 大野邦夫;”ソーシャル・ネットワーキング・アプローチによる英語スキルの展開とグローバル社会での活動に関する考察”,情報処理学会研究報告, DC110−12（2018.9）
[18] 大野邦夫,芥川一則；”エージェント通信と異文化コミュニケーションの類似性に関する検討”, 情報処理学会研究報告, DC99−1（2015.10）
[19] https://www.csee.umbc.edu/~finin//papers/kqml.acl.pdf
[20] Wikipedia; “Speech code theory”, https://en.wikipedia.org/wiki/Speech_code_theory,（2015.7）
[21] P.H.ウインストン, B.K.P.ホーン,（白井良明, 安部憲広訳）; “LISP”, 培風館（1982）
[22] 大野邦夫；”オブジェクト分析設計におけるクラス継承とシステムの意味的概念の形成に関する考察 ”, 第5回画像関連学会連合会大会講演論文（2018.11）
[23] 大野邦夫, 木村登志子;”言語習得プロセスのモデル化に関する一検討〜自然言語の習得とプログラミング言語Lispによる開発の類似性”,情報処理学会研究報告, DC105−10（2017.7）