10　ソリッドモデリング

計算機上に物体をモデル化する手法は形状モデリングと呼ばれ、数多く研究されてきた。しかし、現在ではほぼ境界表現による形状モデリングに実用上統一されたと言ってよい状況である。境界表現は形状の境界を明示的に位相要素データとして保持し、境界の隣接関係を記述する。従来の境界位相構造は、必要な位相要素を異なる次元の多様体それぞれについて、異なるデータ構造として定義・表現し実現していた。１次元の線の境界は頂点、２次元の面の境界は Loop、３次元のソリッドの境界は Shell という具合である。場合により隣接位相要素としてさらに別の位相要素を定義利用する[2]。非多様体モデルと呼ばれる、異なる多様体次元の物体が混在するモデルの表現のためには、ある境界の周りに隣接する位相要素の順序関係を表現する。通常われわれが扱う３次元多様体までのモデリングのためには０から３次元までのそれぞれの対応する位相要素をすべて定義する必要が生じ、システムは複雑化する。

しかし、点、線、面、ソリッドなどの形状要素を Cell（単体）と抽象化して捉え、ある多様体次元の Cell の境界は、その Cell のパラメータ空間を表現する一つ低い多様体次元の Cell の順序化された集合である、と考えると、すべて Cell の境界表現（境界および境界の順序関係の表現）はひとつのデータ構造で表現することができる。われわれはこの表現を**複体モデル**と呼び、C++オブジェクト指向プログラミング言語によりシステム化した。本論文ではこの**複体モデル**を紹介する。

オブジェクト指向プログラミングにおけるクラス継承の考えは、集合の汎化、専化の関係をプログラミングするためのツールである。複体モデルにおける Cell の考えは、点、線、面、ソリッドなどの形状要素を抽象化したものであり、点、線、面、ソリッドなどのそれぞれの形状要素は Cell の具象クラスと考えることができる。

複体モデルの考えは Rossignac ら[5][6] の SGC（Selective Geometry Complex）モデルを起源とする。Ram D Sriram ら[1] は Rossignac らの SGC モデルを C++プログラミング言語と ObjectStore という OODB(Object Oriented Database) System を用いて実現している。SGC モデルでは形状要素を Cell として抽象化しているが、明示的な境界表現というものを持っていない。この点で SGC モデルは実用的なシステムになりにくいと筆者らは考える。そこで、この Cell に Complex（複体）と Boundary（境界）という二つの考え（クラス）を付加し、境界要素を導入した。さらに、Binder Cell(接合 Cell)と Binder Cell により接合される（順序つけられた）Partner Cell 列という考えを導入して、すべての次元の多様体に共通して用いることのできる境界の隣接関係、および隣接位相要素の順序関係の表現を可能とした。これらの考え（クラス）はすべて多様体次元フリーであり、すべての多様体次元の形状要素に共通して使用する。われわれは具体的な利用方法をすぐには考えつかないが、この複体モデルは、必要であれば、３次元以上（４次元、５次元など）の多様体次元のモデリングにも拡張適用可能である。

複体モデルは winged edge data structure[8]を起源とするいわゆる half edge data structure を Edge という１次元多様体だけではなく、すべての多様体次元に関して一般化したものである。

デージーテクノロジーズではMGCLと呼ぶ空間次元フリーな幾何クラスライブラリを所有している。また、近年のプログラミング言語の発展は目覚ましく、現在、C++というオブジェクト指向プログラミング言語と、この言語に標準ライブラリとして含まれる各種コンテナクラス(list, vector, deque, map など）を利用することができる。ＭＧＣＬとＣ＋＋の標準ライブラリを利用することにより、標準 C++プログラミング言語以上のツールを利用することなく、本論文で記述する複体モデルを実現することができた。

10.1　背景

いわゆるソリッドモデリングの手法は次のように三つに分類できる（文献(7)）：

（１） Decomposition models
立方体などのある特定の基本的なオブジェクトをつなぎ合わせソリッドを記述する。平面内の絵を pixel で埋め尽くして書くように、３次元空間内のソリッドを voxel と呼ぶ空間要素で埋め尽くして表現する。

（２） Constructive models
立方体、球、円柱、などの基本ソリッドと基本ソリッド同士の和、差、積などの集合演自体を表現とすることにより、ソリッドを記述する。

（３） Boundary models
ソリッドを囲む境界を定義、表現することにより間接的にソリッドを記述する。

本稿で紹介する複体モデルは上記（３）Boundary models による表現（境界表現）のひとつである。境界表現でオブジェクトを記述する場合、

- ソリッドの境界が Shell
- Face の境界が Loop
- Edge の境界が Vertex

であり、従来、図８のようなデータ構造階層が適用されてきた。この場合いわゆる Half-Edge Data Structure が主要な役割を果たす。Half-Edge Data Structure は図４に例示されるような構造で、面を囲む境界はその面に属する Edge の集合（Loop）と捉える。ふたつの面を接続する場合、それぞれの境界を表現する Edge 同士を表裏の関係にある Edge(Co-edge, Twin Edge などと呼ばれる)として見ることにより、ふたつの面はこのふたつの Half Edge を通して接続されているとする。Half Edge は線であり、面の境界のパラメータ値（ｕ，ｖ）を表現する線と考えられる。それぞれの面の境界はすべてそれぞれの面を表現するパラメータ値（u，v）を表す線により表現されると考えてよい。

従来のソリッドモデリングでは Edge に対してのみこのような考えを適用してきた。ソリッドの境界構成要素である Face、または Edge の境界構成要素である点に対しては異なった構造の

ものとして扱ってきた。しかし、次のように考えればこれらの構造はすべて（多様体次元の違いを無視すれば）同一であることがわかる。

（1） Edge の境界は Edge の線表現のパラメータ値を表現するふたつの点（Vertex）である。
（2） 面を囲む境界 Loop は面の境界のパラメータ値（ｕ，ｖ）を表現する線(Edge)の集合である。
（3） ソリッドの境界 Shell は（中身の詰まった）ソリッドのパラメータ値を表現する Face（曲面）の集合である。
（4） さらにソリッドより高い多様体次元のオブジェクトに対しても、そのオブジェクトの境界は、そのオブジェクトのパラメータ表現である、ひとつ低い多様体次元のオブジェクトの集合である、と敷衍することができる。

複体モデルではこれらの点、線、面、ソリッドを Cell というクラスで抽象化し、ソリッドの境界である Shell、面の境界である Loop、Edge の境界である Vertex を Boundary というクラスで抽象化している。Cell および Boundary という二つの抽象クラスによりソリッド、面、Edge の境界をすべて同一構造で扱うことができた。

10.2　複体モデルの理論的モデル

図１は複体モデルをＵＭＬ(Unified Modeling Language)で表現したものである。

10.2.1　Cell（単体）

Cell は和名で単体(文献(2)では胞体と呼んでいるが、ここでは複体に対比して単体と呼ぶ)と呼び、複体モデルの基本位相要素である、頂点、エッジ、面、ソリッド（または更に高次元の多様体）などの形状要素を抽象化するクラスである。Cell は同じ表現の Geometry(Extent とも呼ぶ)の均一次元の多様体で、Cell を構成する Geometry、その Cell を囲む境界（Boundaries、一般には複数ある）、および Binder Cell となったときのパートナーメンバーCell などのメンバーデータから成る。

(1) Parameter Cell と Binder Cell、および Binder Cell の Partner メンバー Cell

一般に線、面、ソリッドなどの Cell はすべて一つ低い多様体次元の Cell を境界要素として持つ。

- 線の境界は頂点であり、ふたつ（以上）の異なる表現の線がある頂点で接合されているとき、その頂点自身は線と同じ世界座標空間の表現であるが、線の境界としてはそれぞれ異なるパラメータ値の頂点と考えられる。（図２）
- 面の境界は線であり、ふたつ（以上）の異なる表現の面がひとつの境界線で接合されているとき、境界線自身は面と同じ世界座標空間の表現であるが、面の境界はそ

れぞれの面のパラメータ値（u，v）を表現する、異なる２次元の線である。（図３参照）

- ソリッドの境界もまったく同様である。（図４参照）

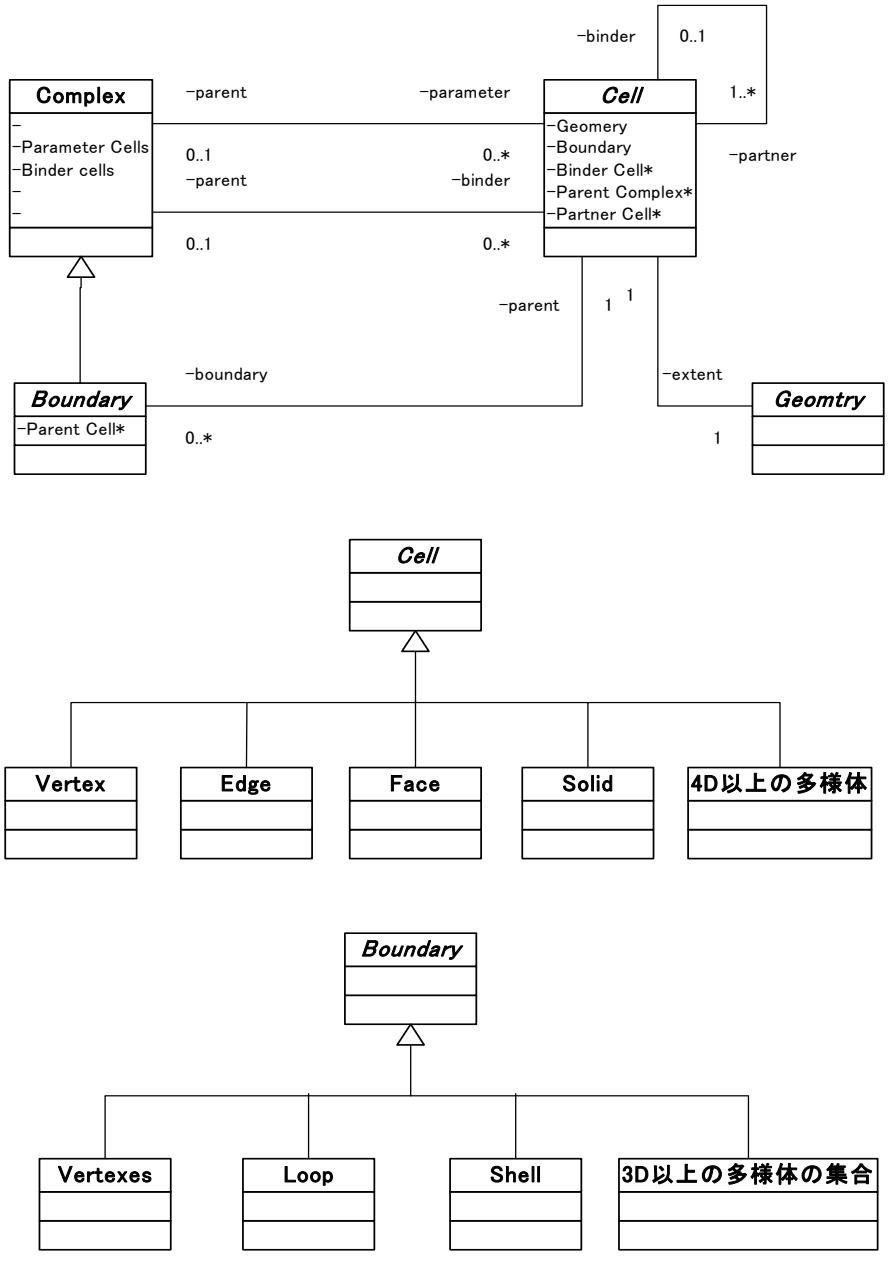

図１ 複体モデルの理論的構造

そこで、ある Cell::SC の境界を表現する（ひとつ低い多様体次元の）Cell::PC と、その境界で接合されている複数の Cell の‘接合’を表現するための Cell::BC を考える。Cell::PC を Cell のパラメータ値を表現する Cell であるから Parameter Cell(PCell と略す)と呼び、Cell::BC を複数の Cell を接合する Cell であるから Binder Cell（BCell と略す）と呼ぶ。

Cell::SC は Cell::PC の Star Cell(星状単体)と呼ばれ、Cell::PC は Cell::SC の Boundary Cell（境界単体）と呼ばれる[3]。

Binder Cell は接合対象の Cell と同じ多様体次元とする。Binder Cell が接合する Cell は一般的には Parameter Cell である。Binder Cell の座標は、接合対象の Parameter Cell の座標をパラメータ空間とする Star Cell の座標と同じものとなる。Parameter Cell がある Star Cell の境界となっていれば、Parameter Cell の座標はその Star Cell のパラメータ空間であるから、Parameter Cell の座標の空間次元は常に“Parameter Cell の多様体次元＋１”である。すなわち Parameter Cell を境界とする Star Cell の多様体次元と等しい。

- 頂点の多様体次元は 0 であり、頂点の Parameter Cell の空間次元は 1（線のパラメータ値=一つの実数）である。これは線の多様体次元である。
- 面の境界線の多様体次元は 1 であり、面の境界の Parameter Cell の空間次元は 2（面のパラメータ値＝(u, v)）である。これは面の多様体次元である。

Binder Cell によって接合された同一多様体次元の複数の Cell 同士は partner 関係の Cell と呼び、この Binder Cell の Partner メンバー cell となる。Partner メンバー cell は一般的には Parameter Cell であるが Binder Cell も Partner メンバー cell になり得る。

Binder Cell は順序がつけられた複数個の Partner メンバー cell を表現し、この境界に接続される隣接位相要素の順序関係を表現する。この順序関係から、いわゆる Disk cycle、Radial cycle 等の隣接位相要素の関係をすべて（異なる多様体次元の Binder Cell の Partner メンバー cell から）求めることができる。参考文献（２）で述べられている隣接位相関係は、この Binder Cell の Partner メンバー cell という一つの構造で表現することができる。我々は C++の std::vector によりパートナーメンバーの順序関係を表現した。

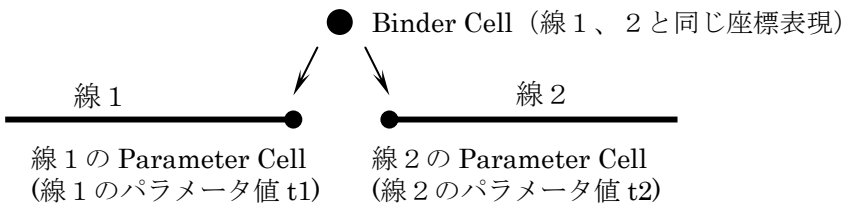

図２　線の Parameter Cell と Binder Cell

(2) Cell の基本関係

- star() / boundaries()

Cell のメンバー関数 star()は その Star Cell（唯一である）を求め、boundaries()はその境界 Cell（によって構成される Boundary の vector）を求める。ある Cell の boundary を求め、その boundary の Star Cell を求めれば、元の Cell が求められる。すなわち、ある Cell::SC がある Cell の Star Cell となっているとすると、次の式が成立する：

SC=(SC.boundary(i)->pcell(j))->star() for all i, j （式１）

(SC のすべての境界の、すべてのメンバーCell の Star cell は SC である)

- neighbours()

ある Cell（n+1 多様体次元とする）の境界を構成する、同じ Boundary 内の隣接 Cell（同じ境界を構成する隣の Cell、従って Cell の境界構成要素）を neighbours と呼ぶ。neighbours はすべて n 次元多様体の Cell である。

面（多様体次元２）の境界である Loop のメンバーCell (Edge)（多様体次元１）を例にとると、Loop のメンバーCell の neighbours は同じ Loop 内の隣接する（前と後に接続されている）二つの Edge である。Loop がクローズしたものではなくオープンであれば、最後の Edge の neighbours は前の Edge ひとつであり、最初の Edge の neighbours は後の Edge ひとつである。

- partners()

それぞれ異なる Cell の境界構成要素であるが、Binder Cell を通して接続されている Cell 同士を Partner Cell と呼ぶ。Partner 関係である Cell 同士は、それぞれ異なる親 Cell の Boundary を構成し、この Partner 関係を通して（この Partner cell を共通の境界として）接続されていることを表現する。

neighbour と partner の関係は、実は、多様体次元を無視すれば同一の関係（同一のデータ構造で表現される関係）である。ある Cell C に対して次の式が成り立つ：

C.neighbours() ={ C.boundary(i)->pcell(j)->partners()[k]->star() } for all i, j, k （式 2）

(ある Cell Cの neighbour は、C の境界を構成する Cell の partner の Star Cell の集合である)

（例１）Edge を通して接続された二つの面 F1 と F2 を例にとる（図３）。

面 F1 と F2 はそれぞれの Parameter Edge E11, E21 が（Binder Cell—Binder Edge—により）接続されており、E11 と E21 はお互いに partner 関係にある。また、E11 の neighbours は（pre_edge である）E16 と（aft_edge である）E12 である。面 F1 の boundary は Loop L0 である。Loop L0 は E11 から E16 で構成されている。Parameter Cell E11 から E16 の Star Cell

はすべて F1 である。

F1 の neighbour は F2 であるが、式２を図３に適用すると次のようになる：

F1 の boundary L0 のメンバーCell E11 の partner E21 の Star Cell F2 が、F1 の neighbour である。

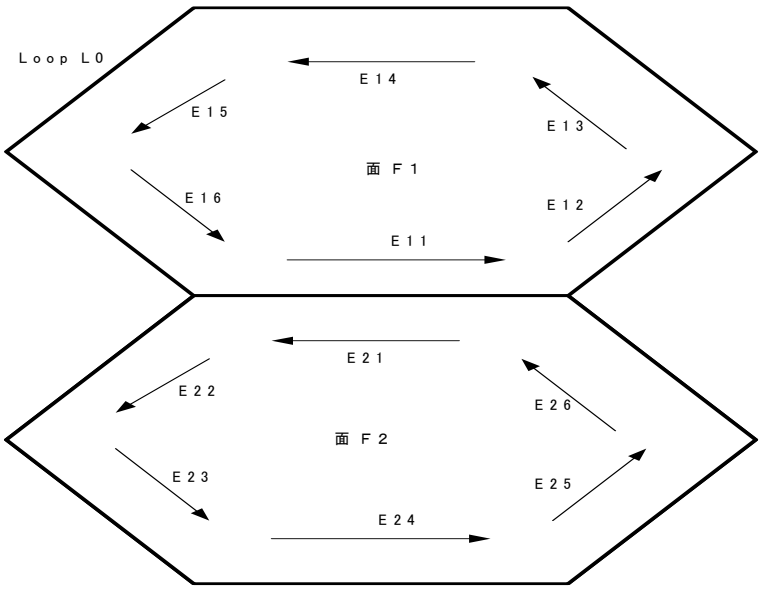

図３ Edgeを通して接続された二つの面

（例２）２段に重ね合わされた（面で接続された）二つの立方体 A と B を例にとる（図４）。

立方体 A の boundaries は Shell S0 で（この場合、一つの Shell)、６つの面 FA1 から FA6 で構成される。これらの Cell（２次元多様体である面）は隣接する Edge を通して一つに接続され、Complex を形成し、立方体の境界(Boundary、Shell)となっている。FA1 から FA6 の Star Cell はすべて立方体 A である。二つの立方体は A 側の面 FA1 と B 側の面 FB1 とで接続されている。この時、面 FA1 と面 FB1 はお互い partner 関係にあり、面 FA1 の neighbours は４つの面、FA2, FA3, FA4, FA5 である。

立方体 A の neighbour は立方体 B であるが、式２を図 4 に適用すると次のようになる：

A の boundary Shell S0 のメンバーCell FA1 の partner FB1 の Star Cell 立方体 B が、立方体 A の neighbour である。

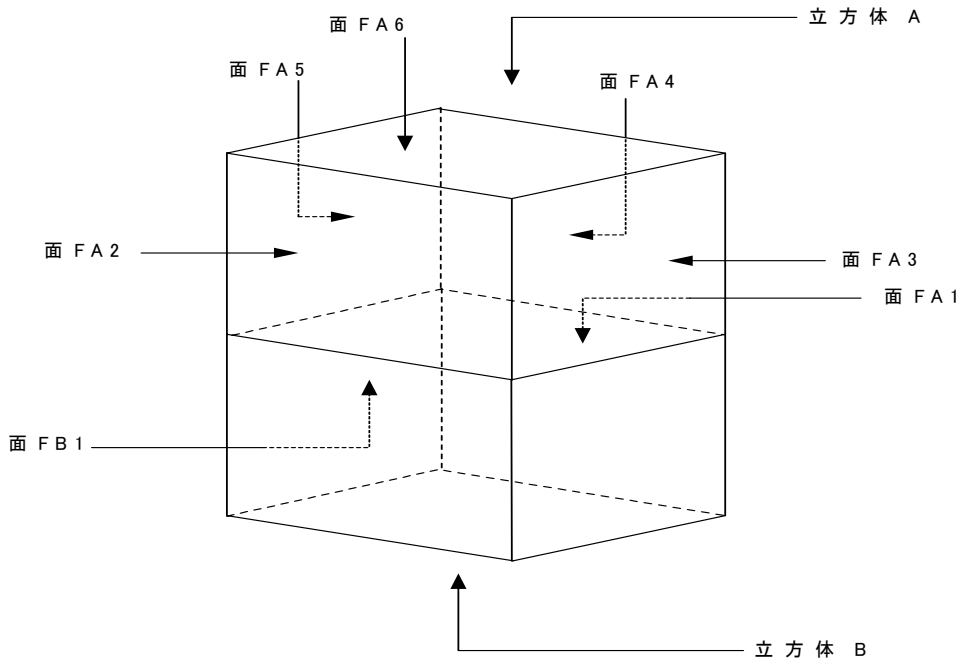

図４ 重ね合わされた二つの立方体

(3) Cell の Boundary

Cell の Boundary が存在しないことがある。Cell の Boundaries().size()=0（境界の数が０）の時、Cell の境界はなく、Cell の extent(Geometry)全体が Cell を構成する。

10.2.2 Complex(複体)

Complex は位相要素である Cell（Parameter Cell と Binder Cell）の集合である。Complex に含まれる Cell の多様体の次元は必ずしも均質である必要はなく、いわゆる**非多様体モデル**を表現することができる。Complex に含まれる Cell は、Parameter Cell、Binder Cell いずれも、たとえ多様体次元が異なっていても、Geometry の座標は同じ世界のものを表現する。例えば、その Complex がある世界座標を表現するものであれば、含まれる Cell はすべて同じ世界座標を表現する。また、その Complex が、ある Cell の境界表現（すなわち Boundary）用のものであれば、含まれる Cell の座標値はすべてその Cell のパラメータ空間を表現する。

Complex は Cell の境界要素である Boundary の親クラスである。Boundary は同一多様体次元の Parameter Cell だけから構成される特別な Complex である。Complex は Geometry をもつことはなく、Geometry は Cell のみがもつ。Complex は Cell の順序つけられた集合を表現する。

10.2.3Boundary

Boundary は Cell の境界を表現する為の特殊な Complex であり、Complex のサブクラスであ

る。Boundary はある Cell(Star Cell)の境界である。Star Cell よりも１次元低い均一な多様体次元の Cell 列であり、Star Cell を n 次元多様体とすれば、Boundary は向き付け可能な(n-1)次元多様体である。Boundary 内のメンバー Cell の表現する座標空間は Star Cell のパラメータ空間である。Star Cell の Geometry をそのパラメータ位置においてふたつに分断し、Boundary のいずれか一方だけが active（興味の対象）となる。Boundary 内のすべてのメンバー Cell（境界構成要素）は、メンバー Cell のすべての境界で（たかだか）ひとつの partner と接続されている。すべてのメンバー Cell のすべての境界で partner に接続されていればその Boundary は閉じている。Partner を持たない境界を持つメンバー Cell が Boundary に含まれていれば、その部分で閉じていない Boundary となる。

たとえば、面（Face=２次元多様体）の境界である Loop のメンバー Cell は Edge(１次元多様体)である。Loop 内のすべての Edge は Star Cell である、あるひとつの面のパラメータ値(u, v）を表現している。これらの Edge はそのパラメータ空間内で同一方向を向くよう向き付けされていて、Edge の境界である両端点においてたかだかひとつの Edge と接続されている。すべての Edge が両端点でそれぞれひとつの Edge(partner Edge)に接続されていれば、Loop は閉じた Loop であり、円（２次元球面 S[1]）と同相である。これらの関係はソリッドの境界 Shell に関しても同様である。

Boundary は抽象クラスであり、具象クラスとしては Loop（Face の Boundary）、Shell（Solid の Boundary）がある。Edge の Boundary は Vertex であるが、次の実装で述べるようにメモリー効率、処理効率のため Boundary のサブクラスとしていない。

10.3 実装

抽象クラスである Complex, Boundary, Cell の関係は理論的モデルのとおりに実装されるが、0 次元多様体である頂点(Vertex)、１次元多様体であるエッジ(Edge)もすべてこの構造体の中に実装することは無駄がある。

・頂点は 0 次元多様体であり、境界を持たない。
・エッジは１次元多様体であり、境界は始点と終点のたかだか２個である。

この無駄を解決するために Cell を３つのクラス階層に分解し、頂点、エッジのための抽象クラス CellBase、CellNB 導入する。 更に Vertex を PVertex（Parameter Cell である Parameter Vertex）と BVertex(Binder Cell である Binder Vertex)に分解する（図５参照)。

10.3.1 CellBase

CellBase は自身の Cell が Binder Cell によって接続される関係のみを表現し PVertex と CellNB の親クラスとなる。
CellBase のメンバーデータは自分を接続する Binder Cell へのポインターのみである。

10.3.2 PVertex (Parameter Vertex)

PVertex は Edge の Parameter Cell であると同時に Edge の Boundary という二つの性格を持つ特殊な Cell である。CellBase のサブクラスである。

PVertex のメンバーデータは Edge（Edge の Geometry であるの曲線）のパラメータ値と、その Edge へのポインター（Boundary が持つ親 Cell へのポインター）である。

PVertex は Edge の Parameter Cell であり、同時に Edge の Boundary であるが、Binder Cell とはならない。PVertex の Binder Cell は BVertex である。

10.3.3 CellNB (Cell with No Boundaries)

CellNB は一般的な Cell から Boundary data を除いた構造を表現する。CellBase の子クラス。Complex のメンバーには CellNB（の子クラス）のみがなり得る。また、Binder Cell も CellNB の子クラスのみがなり得る。

BVertex （0 次元多様体であり、境界がない)、Edge（PVertex を境界とし、一般的な境界を持たない)、および一般的な Cell の親クラスとなる。

メンバーデータとして Geometry（Cell の表現する extent)、Complex のメンバーとなった際の親 Complex へのポインター、および Cell が Binder となった際のメンバーの Partner Cell 列である。

10.3.4 BVertex(Binder Vertex)

Binder Vertex は 0 次元多様体（点）であり、境界がない。 PVertex の Binder Cell 版である。BVertex はメンバーデータを持たない。

10.3.5 Edge

Edge は 1 次元多様体（線）であり、一般的な境界（Boundary）をもたないで、PVertex を境界とする。PVertex の 2 個の配列（始点と終点）をメンバーデータとして持つ。

edge の方向

edge の方向は edge の Geometry である Curve の方向と同一である。Parameter edge の方向は常にこの Curve の方向が Edge の方向となる。Loop 内の Edge の方向はすべて同一方向に整列される。loop の方向は parameter edge の方向に見て、左側が face の内部で、右側が空な領域である。loop の edge(メンバーedge)は面の normal 方向に対して反時計回りに整列して格納される。(右ネジを締めるときに右ネジが回る方向が反時計回りで Edge, loop の方向、右ネジが進む方向が loop を境界とする面の方向)。

Binder edge には方向性の制約がない。

10.3.6 Loop

Loop は Parameter Edge の集合であり、 1 次元多様体の Boundary(Complex)である。Boundary のサブクラスである。Loop 内の Edge は前後に高々ひとつの Edge しか接続されていない。最初と最後の Edge が接続された Loop はクローズした Loop であり、円と同相である。面の境界を表現する。ある面 F の境界としての Loop に(Parameter Edge として)含まれる Edge の世界座標値はすべて面 F を構成する Geometry のパラメータ値(u, v)を表現する。

面のパラメータ空間は(u, v)の２次元の世界であるが、この中で Loop は必ず方向性があり、進行方向の左側が境界の内部、右側が境界の外部である。

10.3.7 Cell

Cell は 2 次元以上の多様体（例えば、面、ソリッド、その他のもっと大きな多様体次元の単体）を表現する抽象クラスである。具体的なサブクラスとしては Face、Solid がある。
メンバーデータとして Boundary の列を持つ。 Boundary の列の最初の一つは outer boundary であり(存在しない場合がある)、他の複数は inner boundary である。非 active な Boundary が含まれる場合もある。boundary 同士が交差することはない。

10.3.8 Face

Face は２次元多様体（面）であり、Loop を境界（Boundary)とする。Cell のサブクラス。Shell の構成要素である。

10.3.9 Shell

Shell は複数の Face を Face の境界である Parameter Edge を通して接合したもので、Solid の境界となる。Boundary のサブクラスである。

10.3.10 Solid

Solid は 3 元多様体（立体）であり、Shell を境界（Boundary）とする。

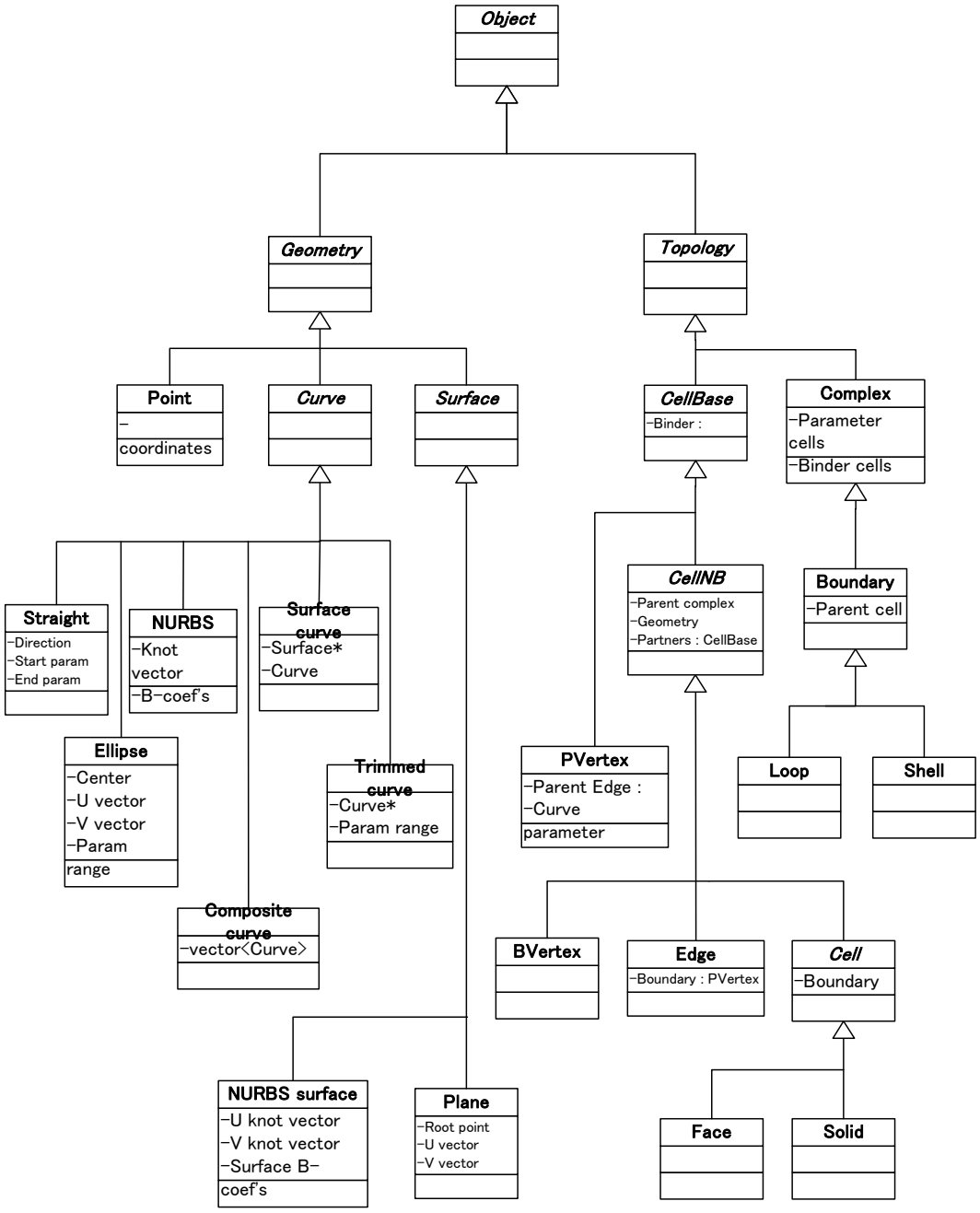

図5　完全なObjectのクラス継承階層

10.4 例

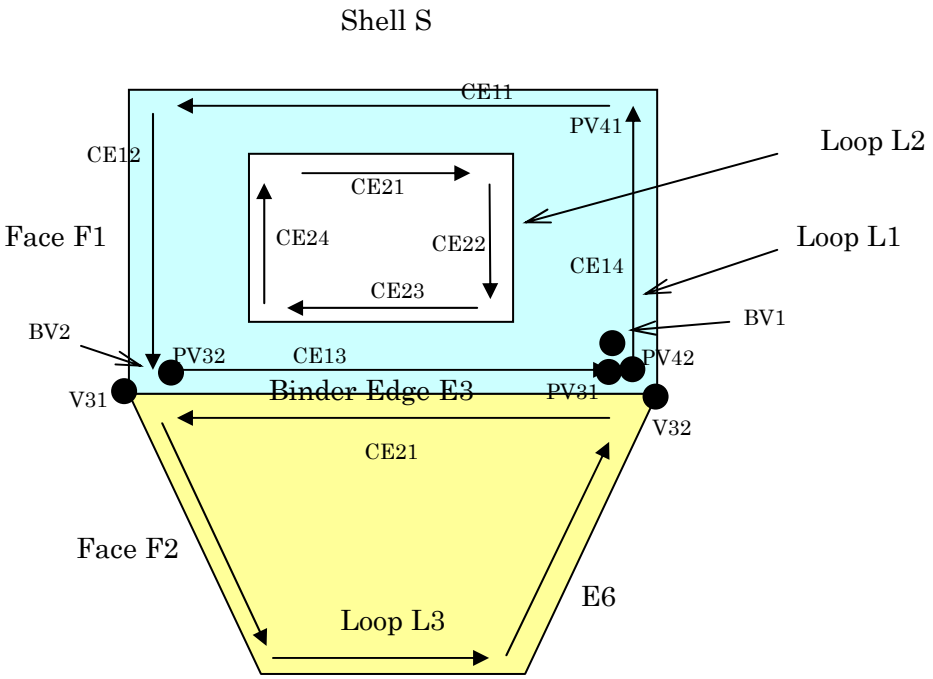

図６ Shell の表現例

(1) この例では、Shell S が 2 つのメンバーCell F1, F2（2 次元多様体）から成り立っている。

S.star()=null （Root Complex である）
S.pcells()={F1, F2} （最終製品の世界座標表現）
S.closed()=False （Shell のメンバー cell である F1, F2 ともに open なものがある）

(2) Face F1 は 2 次元多様体の Parameter Cell（最終製品の世界座標表現）であり、Loop L2 で表現される穴がひとつある。

F1. parent_complex()=S
F1.extent()=F1 の曲面表現 Geometry(Surface)
F1.boundaries() ={L1, L2} （L1 は outer boundary であり、L2 は inner boundary）
F1.star()=null （最終製品の世界座標表現である）
F1.neighbours()={F2} （おなじ Shell のメンバーとなる F1 のある境界で接続された隣接面）
F1.partners()=null （F1 の partner face=F1 に接続される隣接 Shell の face はない）

(3) Loop L1, L2 はそれぞれ Boundary であり、 1 D Complex

L1.star()= F1 （L1 の Star Cell は Cell F1）
L1. pcells()={CE11, CE12, CE13, CE14} （F1 曲面のパラメータ空間(u, v)表現）
L1.active()=True （L1 は有効＝閉じている）
L2.star() = F1 （L2 の Star Cell は Face F1）
L2.pcells()={CE21, CE22, CE23, CE24}（CE21,222,3,24 は F1 曲面のパラメータ空間(u, v)表現）

L2.active()=True　　（L2 は有効）

L1 と L2 とは同じ F1 曲面のパラメータ空間(u, v)表現であるが、向きは反対である。

(4) Edge CE13 は Face F1 の Parameter Cell（１次元多様体）である。

CE13.parent_complex() ={L1}　　（CE13 は L1 のメンバーCell）
CE13.extent()= CE13 の曲線表現 Geometry
　　（F1 曲面のパラメータ空間(u, v)を世界座標とする２Ｄ曲線）
CE13.boundaries() ={PV32, PV31}
　　（CE13 の境界は PVertex　PV31, 32= CE13.extent()のパラメータ値）
CE13.star()=F1　　（CE13 は面Ｆ１のパラメータ空間が世界座標値）
CE13.neighbours()={CE12, CE14}　　（おなじ面 F1 の境界を表現する隣の Edge）
CE13.partners()=CE21　　（binder E3 により、CE13 と接続されている隣の面 F2 の境界）
CE13.binder()=E3　　（CE13 と CE21 は binder E3 で接続されている）

(5) E3 は Binder Cell である。

E3.extent()=F1, F2 と同じ世界座標表現の曲線表現 Geometry
E3.star()=null　　（E3 は最終製品の世界座標表現である）
E3.partner_members()={CE13, CE21}　（CE13 と CE21 は binder E3 で接続されている）

(6) PV32, PV31 は CE13 の境界である(PVertex)。

PV31.t()={PV31 を表す線 CE13 のパラメータ値}
PV31.star()={ CE13}　　（PV31 の star cell は Cell CE13）
PV31.partners()=PV42　　（BV1 により、PV31 と接続されている隣の Edge の境界）
PV31.binder() =BV1　　（PV31, PV42 は binder BV1 で接続されている）

(7) BV1 は Binder Cell である。

BV1.extent()=PV31, PV42 の Star cell CE13, 14 と同じ空間座標（面 F1 のパラメータ空間座標）
BV1.star()=F1　　（BV1 の世界座標は面 F1 のパラメータ空間）
BV1.partner_members()={PV31, PV42}（PV31 と PV42 は binder BV1 で接続されている）

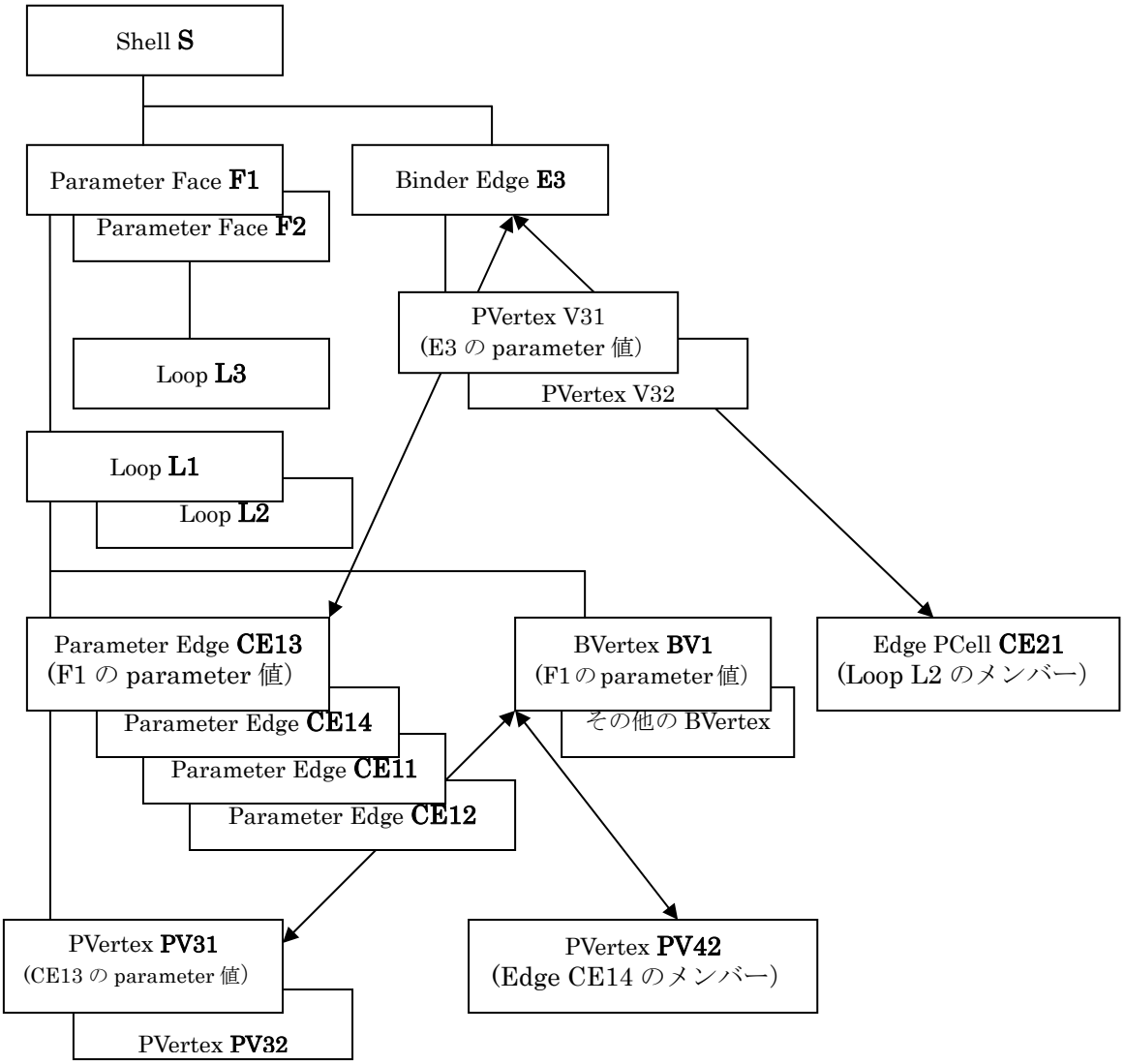

図７ Shell（図６）の内部表現例

10.5　従来の Shell Structure との対比

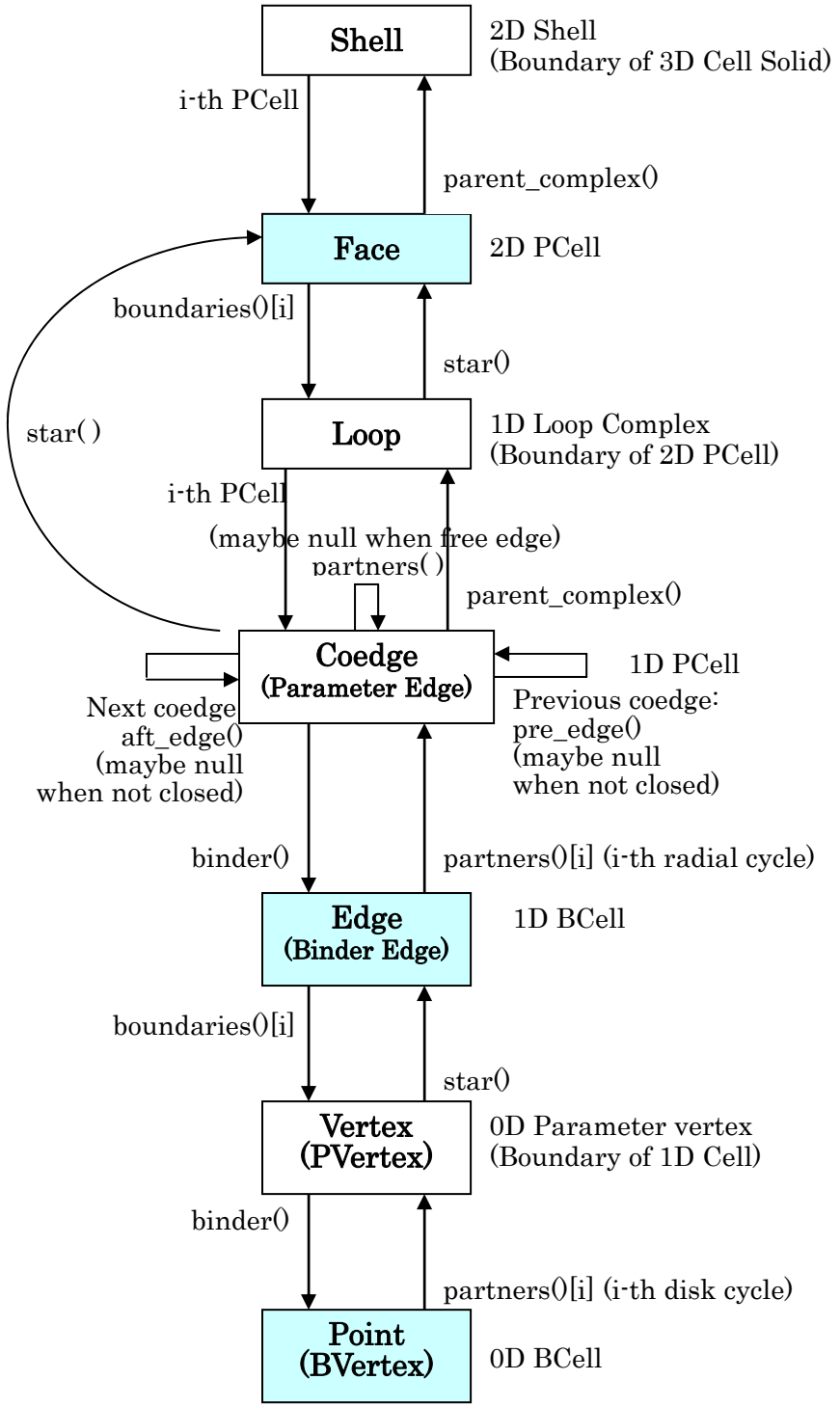

図８　従来のモデルとの比較

従来のモデルの隣接位相要素が複体モデルではどのように求められるかを見てみる[9]。

(1) Disk cycle

Disk cycle は vertex 回りの edge とその edge を境界としている face の順序集合である。ある vertex を特定するにはある edge の境界である始点または終点の parameter vertex を特定するか、または edge と同じ座標系をもつ binder vertex を直接特定するかである。Binder vertex は Parameter vertex から binder()関数によりもとめられる。Binder vertex は partner_members としてすべての disk cycle の parameter vertex (従来のモデルでは parameter vertex と binder vertex を区別していない)を明示的に保持している。Parameter vertex からその vertex を境界とする edge, さらにその edge が境界となっている face を求めることができる。これを binder vertex 内の partner_members の parameter vertex に順に適応すれば disk cycle のすべての情報が求められる。

(2) Radial cycle

Radial cycle は edge 回りの face とその face を境界としている solid の順序集合である。ある edge を特定するには、ある face の境界である loop から、その構成要素である parameter edge を特定するか、または face と同じ座標系をもつ binder edge を直接特定するかである。Binder edge は Parameter edge から binder()関数により求めることができる。Binder edge は partner_members としてすべての radial cycle の parameter edge (従来のモデルでは coedge、または twin edge と呼ばれている)を明示的に保持している。Parameter edge からその edge を境界とする face, さらにその face が境界となっている solid を求めることができる。これを Binder edge 内の partner_members の parameter edge に順に適応すれば radial cycle のすべての情報が求められる。

(3) Loop cycle

Loop cycle は face の境界である loop 回りの edge と vertex の順序集合である。Loop cycle は複体モデルにおける loop そのものであり、loop に含まれる parameter edge により順に edge, vertex が求められる。

10.6 MGCL のソリッドモデルの評価

MGCL はいわゆる 2 次元多様体である shell モデル（曲面モデル）の実現をターゲットとして開発を進め、shell までを、ここで記述されている複体モデルで実現した。オイラーオペレーションと呼ばれるソリッド操作のための汎用オペレーションは実現していない。この意味でいわゆるソリッドモデルは実現していない。

文献（１）では、次の三つの基本操作を多様体次元に共通な操作として定義し、オイラーオペレーションに使用しようとしている。

（1） Subdivide

（2） Select

（3） Merge & simplify

MGCL は、これらの走査が共通操作として使用できるかどうかに及んでいない。それぞれの多様体次元専用に（線は線用に、面は面用に）考え、実装した。これらの操作が Cell の共通操作として利用できれば利用価値が高い。ソリッドの操作の実装と共に今後の研究課題である。

いずれにしても、背景のモデル、すなわちデータ構造は、すべて Cell の境界とその Cell の境界を構成する Cell の Binder cell およびそのパートナーメンバーCell という共通構造を使用することができる。しかも冗長度の少ないデータ構造である。従来のデータ構造を飛躍的に単純化できた。

10.7 参考文献

(1) Ram D Sriram, Albert Wong and Li-Xing He “GNOMES: an object-oriented nonmanifold geometric engine” Computer-Aided Design, Vol.27, No.11, pp.853-868, 1995

(2) 山口泰、木村文彦 「非多様体位相の隣接関係の表現と操作」 情報処理学会論文誌 Vol.32 No.6, June 1991

(3) 森元勘治 著 「３次元多様体入門」 培風館, 1996

(4) Spatial Technology ‘ACIS 3D Toolkit Technical Overview’

(5) Rossignac, J. and Requicha, A. A. G. ‘Constructive non-regularized geometry’ Computer Aided Design Vol.23 No 1 (1991)

(6) Rossignac, J. and O’Connor, M. ‘Selective geometry complex: a dimension-independent model for point sets with internal structures and incomplete boundaries’ in Wozny, M J, Turner, J U and Preiss, K. eds. Geometric Modeling for Product Engineering, North-Holland (1990)

(7) Martti Mantyla ‘An introduction to Solid Modeling’, Computer Science Press(1987)

(8) Baumgart, B. ‘Winged-edge Polyhedron Representation, Stanford A.I. Report No. CS-320 (1972)

(9) Gursoz, L., Choi, Y. and Prinz, F.B. ‘Vertex-based Representation of Non-manifold Boundaries, Geometric Modeling for Product Engineering, Wozny, M.J., Turner, J., and Preiss, K. eds., pp.107-130, North-Holland (1990)