非多様体形状モデルのための形状操作とその応用に関する研究

増田　宏

概要

これまでソリッドモデルは設計や生産を支援するために広く用いられてきたが、表現できる幾何形状が 2 -多様体に制限されており、設計生産で現れる様々な幾何形状を一貫して表現するための形状モデラとしては不十分であった。そのため近年では、より広い定義域を持った非多様体形状モデルが注目されている。非多様体形状モデルは、ワイヤフレーム、サーフェス、ソリッドが統一的なデータ構造で扱えるため、様々な形状表現の行なわれる設計生産の支援に適していると考えられる。

しかしながら、非多様体形状モデリングでは、これまでデータ構造の提案はなされているものの、ソリッドモデリングに比べて理論的な基盤が貧弱であり、基本位相操作や集合演算の実現法などは知られていなかった。また応用という観点からみても、非多様体位相構造を有効に利用したシステムの提案がほとんどなされていなかった。そこで本研究では、

- 非多様体形状モデリングシステムの構築に不可欠な基礎技術を確立する、
- 非多様体形状モデルを設計生産に関する問題に応用し、その有効性を実証する

ことを目的とする。

基礎技術について論じるにあたっては、非多様体形状モデルの表現対象を「適当な胞体分割により複体となる幾何形状」とする。この定義域はワイヤフレーム、サーフェス、ソリッドを含むものであり、設計生産を一貫して支援する目的に十分適合するものである。ここで提案する基礎技術は、以下に示すような、オイラー操作の導出法、データ構造、集合演算の実現法の三点である。

- オイラー操作は、形状モデルの位相構造を変更するための原始的操作であり、ソリッドモデリングシステムを構築するために不可欠のものであった。しかしながら、非多様体形状モデルでは従来のオイラー操作に相当する位相操作は知られていない。そこで本研究では、非多様体形状モデルの位相的な拘束式である

$$v - e + (f - r) - (V - Vh + Vc) = C - Ch + Cc$$

(v, e, f, V はそれぞれ vertex, edge, face, volume の個数、r, Vh は face, volume の穴の個数, Vc は volume の空洞の個数, C, Ch, Cc はそれぞれ形状モデルの連結成分, 貫通穴, 空洞の個数) を導出し、この式に基づいてオイラー操作が定義できることを示す。この方法によって、非多様体形状モデルでは９種類の位相操作が独立であり、それらの操作を組み合わせて任意の非多様体形状モデルが生成できるという定理を導くことができる。

- 定義域の幾何形状を計算機で表現するためのデータ構造について、主として効率の面から考察を行ない、データ構造の設計を行なう。本論文で述べる形状操作や応用システムをこのデータ構造に基づいて実装することで、本データ構造の実用性を確認する。

- 非多様体形状モデルの集合演算を実現するために、併合操作、抽出操作、簡略化操作の三種類の操作を組み合わせた新しい手法を提案する。本手法の特徴は、併合操作を用いることで製品形状には現れない位相要素を内部表現として保持できる手段が提供でき、さらに抽出操作を用いることで内部表現から部分集合を取り出す手段が実現できることである。また、必要に応じて簡略化操作を用いることで必要最小限の位相要素以外を消去してデータ量を軽減することもできる。非多様体位相構造に基づく集合演算では、最小限の位相要素で演算結果を表現する方法と、元のプリミティブの位相構造を保持する表現法の二通りが考えられるが、本手法はこの両方に対応できるものであり、非多様体形状モデルの持つ位相的な柔軟性を十分に引き出せる集合演算が実現できる。

そして次に、以上のような基礎技術を利用することで構築された非多様体形状モデリングシステムを従来のソリッドモデルでは解決が困難であった問題に応用することを考える。本論文で示す応用は、試行錯誤的な形状生成法、形状特徴表現法、三面図からのソリッド合成法の三つである。

- 集合演算は形状生成手段として最も一般的なものであるが、これまでは大規模な形状モデルでは修正に非常に時間がかかり試行錯誤的な形状生成が困難であるという問題があった。これは、集合演算がそもそも演算順序に依存する演算であるために、途中の演算を取り消すために他の多くの集合演算も再計算しなければならないためである。本研究では、併合操作が順序に依存しないという性質を利用することで、演算順序に依存しない高速な集合演算の修正操作が可能となり、試行錯誤的な設計環境に適した形状操作が実現できることを示す。

- 形状特徴は幾何形状の特定の部分に工学的意味を記述したものであるが、従来のソリッドモデルでは表現可能な形状特徴が２-多様体の部分形状として表現できるものに限られ、また形状特徴間に干渉が生じる場合には形状特徴が適切に保持できないという問題があった。このような問題は非多様体位相構造を利用するこ

とで解決できる。ここでは、形状特徴表現で必要となる位相要素も保持した位相構造が併合操作によって生成でき、また形状特徴や製品形状などの幾何形状が適宜抽出操作で取り出せることを利用して、形状特徴の表現・管理に適したシステムが構築できることを示す。

- 三面図からソリッドモデルを自動合成する問題において、非多様体位相構造と推論システムを利用した新しいソリッド合成法を提案する。ソリッド合成問題では、処理の過程でワイヤフレームやサーフェス、ソリッドが現れるので、本問題には非多様体形状モデルが適していると考えられる。本手法は、従来の方法に比べると処理が単純で、処理効率がよいという利点がある。また、本手法が誤りを含んだ図面からのソリッド合成問題にも適用できることを示す。

目 次

図 目 次

Chapter 1

序論

本章では、本論文で解決しようとする問題および本研究の目的について述べる。また、論文の構成について示す。

また、３次元形状処理に関する研究について概観し、非多様体形状モデリング研究において本研究の果たしてきた役割を述べる。

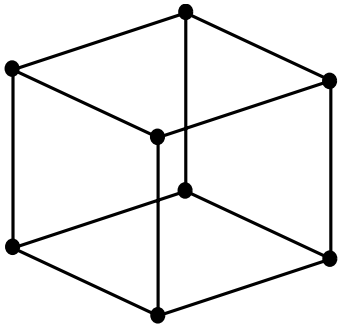
1.1 はじめに

３次元形状モデルは、設計や生産の場において広く用いられてきた。設計においては、製品の幾何形状を決定することは最も重要な作業の一つであり、また、体積や応力分布など、幾何形状に依存する工学的性質も多く存在する。そのため、製品の幾何形状を計算機によって表現した３次元形状モデルは、設計作業の支援や様々な工学的なシミュレーションを行う上で大変重要な役割を果たしてきた。

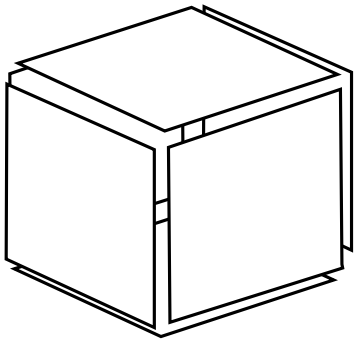
３次元形状モデルとしては、これまでワイヤフレームモデル (wireframe model)、サーフェスモデル (surface model)、ソリッドモデル (solid model) が用いられてきた。図 1.1 は、これらの表現を示している。ワイヤフレームモデル (図 1.1 (a)) は、直線と曲線だけで構成されており、面という概念は持っていないため、３次元形状の表現としては曖昧である。サーフェスモデル (図 1.1 (b)) は、面の表現を保持しているため、隠面処理表示や NC 加工データの生成などに利用することができる。ソリッドモデル (図 1.1 (c)) は、面のどちら側が実体であるかという情報を保持することにより、立体の完全な幾何情報を保持した形状モデルである。

保持されている情報量からいえば、明らかにソリッドモデルが優れており、マスプロパティの計算や FEM 解析、NC データの作成など非常に広い範囲での応用が可能となる。しかし、現実の設計生産の場では、目的に応じて形状モデラが選択され、ワイヤフレームモデル、サーフェスモデル、ソリッドモデルが使い分けられてきた。なぜなら、ワイヤフレームモデル、サーフェスモデル、ソリッドモデルの順で形状モデルが作成しにくくなり、コストが増大していくため、形状モデルをどう利用するかによってこれらの表現法を使い分けていくことが必要になっているからである。

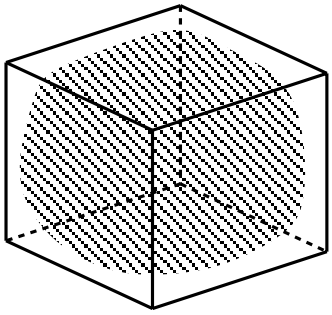
そのような現状に対応するため、近年では非多様体形状モデリング技術が注目されている。非多様体形状モデルはワイヤフレームモデル、サーフェスモデル、ソリッドモデルが統一的に表現できるデータ構造を持った３次元形状モデルで、1986 年に K. Weiler によってそのデータ構造が提案された [Weiler86b]。ソリッドモデルでは位相構造の制限からワイヤフレームやサーフェスを表現することができなかったが、非多様体形状モデルを利用することで、従来は別々の形状モデラによって表現されていた幾何形状を単一の表現形式に統一することができる。また広範な表現空間を利用することで従来の形状モデルでは困難であった応用にも利用できることが期待できる。



(a) wireframe model



(b) surface model



(c) solid model

図 1.1: 幾何形状の表現.

1.2 ソリッドモデルの制限

ソリッドモデルは、立体形状を表現する手段としては大変優れており、設計作業やプロセスプラニングなどで広く利用されてきた。しかし、設計において広く用いられている境界表現ソリッドモデルには、2-多様体でない形状は表現できないという本質的な制限がある [Braid75, Mantyla86]。

2-多様体とは、図 1.2 に示したように、立体の任意の境界上に十分小さな球を置いたとき、境界面が球を実体とそうでない側に二分するような形状と考えることができる。図 1.2 (a) では、立体の境界面が球を二分するので 2-多様体であり、(b) のように一本の稜線が三枚以上の面に共有される場合は 2-多様体ではない。また、ワイヤフレームや閉じていないサーフェスモデルでは 3 次元の実体という概念を持たないので、2-多様体ではない。2-多様体でない形状はソリッドモデルの表現対象外である。

ソリッドモデルが 2-多様体を定義域とすることによって生じる問題には次のようなものがある。

1. 設計を一貫して支援するための中核となる形状モデルとしては十分ではない。
2. 集合演算でソリッドモデルを生成していく過程で、2-多様体でない立体形状が現れることがある。
3. 形状モデルを工学的応用に用いる場合、形状特徴の記述が必要になることがあるが、2-多様体形状モデルでは保持できる形状特徴が限定されている。
4. いくつかの応用分野では、分割モデル (decomposition model) やサーフェスとソリッドの混在などが有効であるが、ソリッドモデルでは対応することができない。

これらの問題は、2-多様体よりも広い定義域を持つ形状モデリングシステムを実現することで、解決できることが期待できる。そのような形状モデルについて述べる前に、ここに示したソリッドモデルの制限について説明を加えておく。

1. 設計支援における制限

現実に存在するすべての実体形状は、2-多様体の条件を満たしているので、製品形状を表現する目的のためにはソリッドモデルは十分な能力を持っているともいえる。しかし、現実には、ワイヤフレームモデルやサーフェスモデルも設計を支援する上で不可欠なものとなっている。

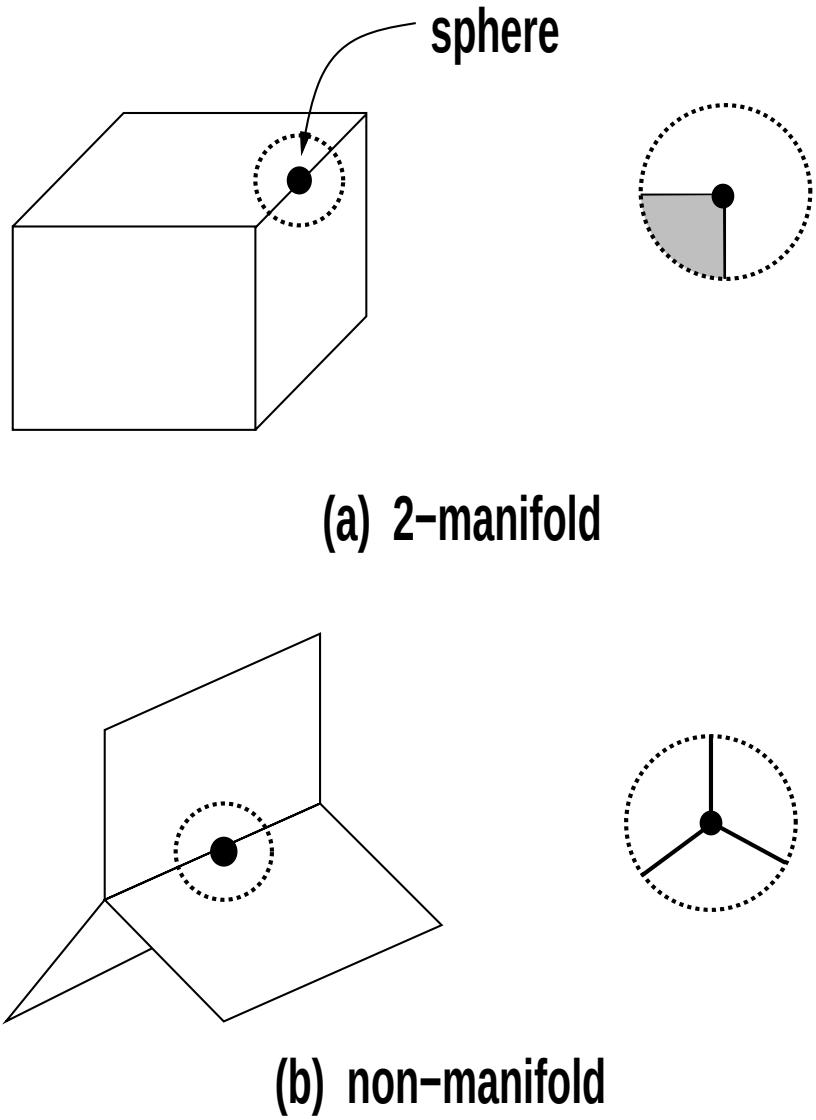


図 1.2: 2-多様体の条件.

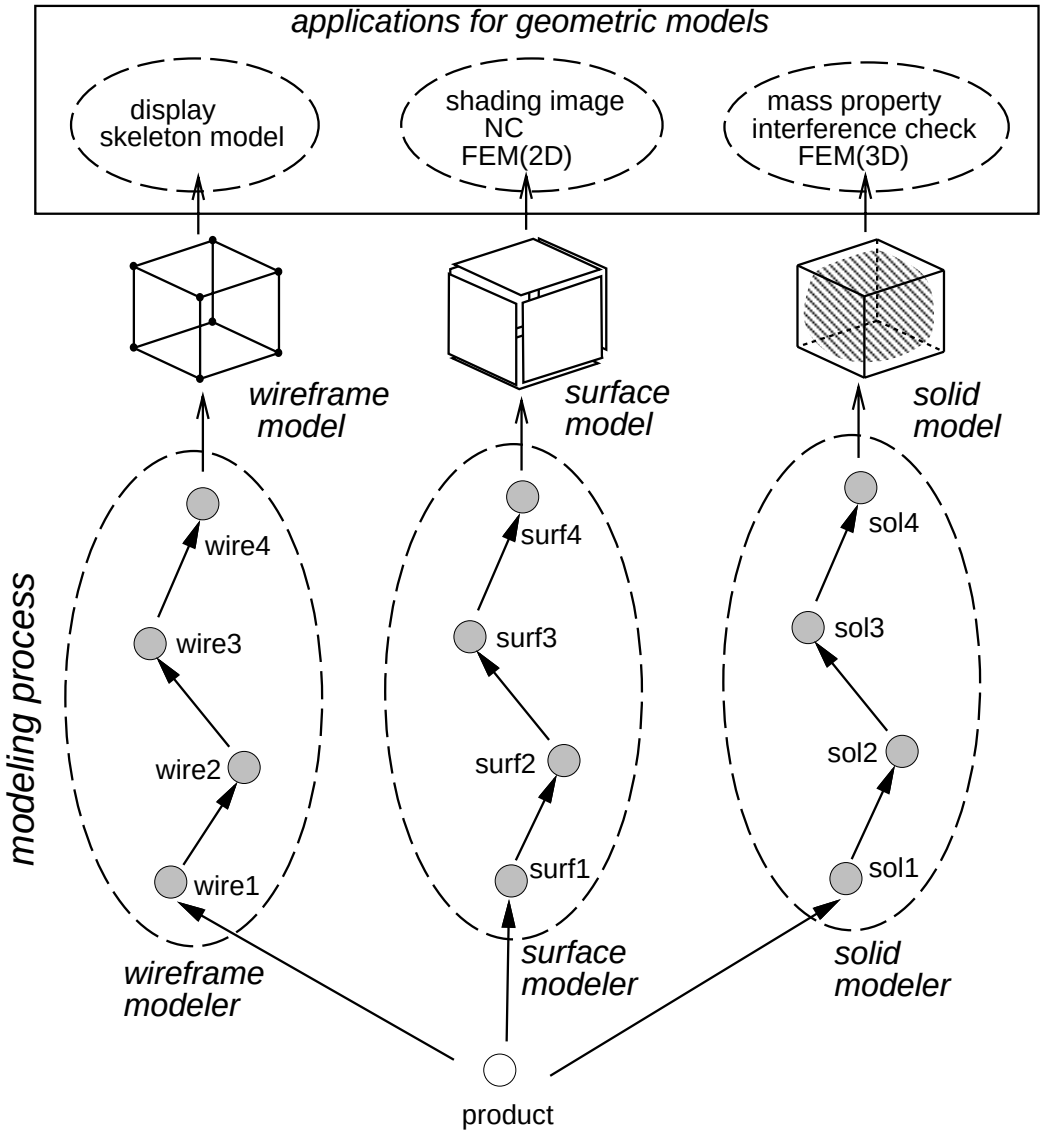


図 1.3: 3次元形状モデルの生成と利用.

図 1.3 は、製品 (product) の形状モデルが作成されていき、利用される様子を示している。通常は、どのような応用に利用するかによって形状モデラを選択し、ワイヤフレームモデル、サーフェスモデル、ソリッドモデルを別々のデータ構造で作成していく。そして、作成された形状モデルは、矩形で示したような応用分野で利用されることになる。この図で示しているすべての応用はソリッドモデルが作成できていれば対応可能であるが、完全な立体情報を必要としない応用分野も多く存在する。たとえば、表示だけが目的ならワイヤフレームモデルでも十分かもしれないし、曲面設計であれば一部の曲面だけが重要で、必ずしも完全な立体を表現する必要はないかもしれない。このような場合には、作成の大変なソリッドモデルを作成するのは無駄である。一般に、ワイヤフレーム、サーフェス、ソリッドの順で形状モデルの生成作業に長い時間を要し、作業コストが増大していく。そのため、ソリッドモデルだけですべての作業を行なうことは、現時点においては一般的であるとはいえず、設計作業全般を支援するための形状モデルとしては十分とはいえない。

また、設計の比較的初期の段階では、製品の完全な立体情報が決まっていないため、ソリッドモデルを生成することは困難だという問題がある。大雑把な仕様から徐々に細部を詳細化していく設計作業においては、ソリッドモデルが利用できるようになるのは形状の大部分が確定した設計の後半になってからである。そのため、設計の初期の段階で形状モデルを利用しようとするならば、立体としては不完全な形状も表現できることが必要となる。また、あえてソリッドモデルを試行錯誤的に製品形状を決めていく目的に利用しようとすると、形状修正の際の操作性が非常に悪いという問題が生じてくる。この点からも、形状が完全に確定していない場での利用がしにくくなっている。

2. 集合演算の制限

立体の表現に限定して考えてみても、ソリッドモデルには、集合演算を繰り返していく過程で、しばしば 2-多様体でない図 1.4 (a), (b) のような形状が生じてしまうという問題がある。これは、ソリッドモデルの集合演算の定義である正規化集合演算 [Requicha80] に関して、2-多様体が閉じていないためである。

このことは、ソリッドモデルを利用する現場で以前から指摘され、問題になっていた。これらの形状は「非多様体」と呼ばれており、境界表現ソリッドモデルでは表現できないので、集合演算を施す際に基本立体の位置を本来の位置からわずかにずらしたり、演算順序を変更するなど、非多様体を避けるための面倒な操作が必要であった。

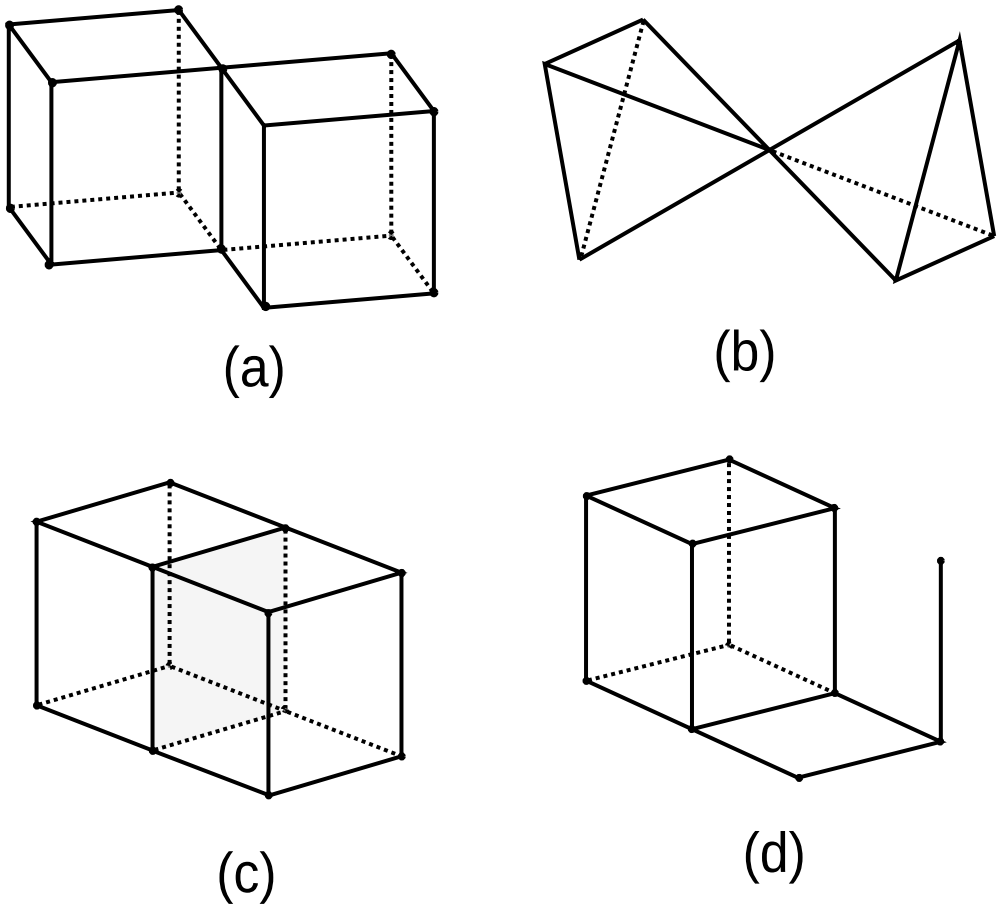


図 1.4: 2-多様体ではない立体形状.

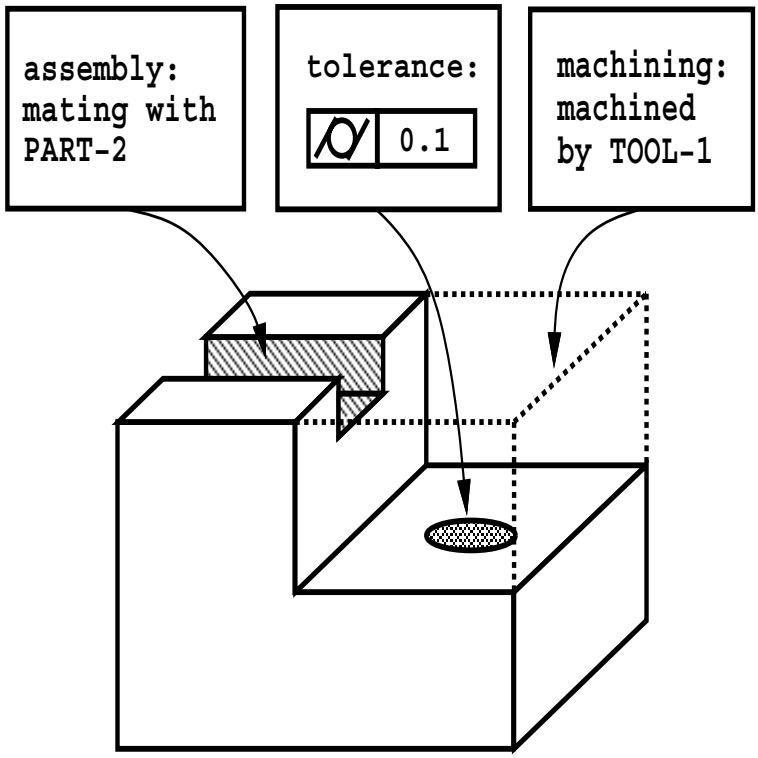


図 1.5: 形状特徴の記述された形状モデル.

3. 形状特徴表現の制限

形状モデルの工学的応用のためには、形状の特定の部分に対する加工手段や精度、接触面、寸法、材料などの工学的情報の記述が必要なことが多い。形状特徴はそのための概念であり、穴や溝など形状モデルの特定の部分に対して工学的属性を付加したものとして定義される [Shah88, Wilson88, Pratt88]。図 1.5 は、形状特徴の記述された形状モデルを示しており、このような情報はプロセスプラニングなどに多く利用されている。

しかし、境界表現ソリッドモデルでは、形状特徴が位相要素の集合によって表現されるため [Shah88, Wilson88, Pratt88]、表現可能な形状特徴は 2-多様体の部分形状となるものに限定される。また、形状モデルが徐々に変更されていく場合には、形状特徴の表現に用いられた位相要素が消失し、形状特徴が保持できなくなる場合がある。このような問題は、形状特徴に応じた付加的な位相要素を保持することで解決可能であるが [Masuda92a]、そのためには 2-多様体以外の位相構造が必要となる。

また、2-多様体では、体積特徴を記述することができない [Pratt88]。体積特徴とは、

加工で除去される立体領域などに対して工学的意味を付加したものである。しかし、図 1.4 (c) のように、立体をいくつかの領域に分割した形状は、三枚以上の面に共有される稜線を含むので 2 -多様体形状モデルでは表現できない。

4. 応用分野の制限

さらに、応用によっては、分割モデルや、サーフェスとソリッドの混在した形状モデルなど有効となりうる。図 1.4 (c) は、立体を二つの部分空間の和として表現したものであるが、このような形状は、複数材料でできた製品形状や FEM 解析のための四面体分割された形状モデルの表現などで有効なものである。また、FEM 解析では、形状をすべてソリッドで表現すると、解析時間が長くなったり、解析結果が不安定になったりするため、図 1.4 (d) のように非常に薄い部分を簡略化してサーフェス表現やワイヤフレーム表現にした方が好ましい結果が得られることがある。

1.3 非多様体形状モデリング

1.3.1 非多様体位相構造を用いた形状表現

以上述べたような制約は、定義域が 2-多様体であることから生じたものであるから、より広い定義域を持つ形状モデリングシステムが構築できればこれらの問題が解決できると考えられる。

そのような 3 次元形状モデルとして、非多様体形状モデル （non-manifold geometric model）が注目されている [Weiler86]。これは、直観的には、ワイヤフレームモデル、サーフェスモデル、ソリッドモデル、また図 1.4 の例で示した 2-多様体以外の形状も含んだ、従来よりも広い定義域の幾何形状が表現できる形状モデルである。非多様体形状モデルは、様々な形状の現れる設計生産を支援するための中核形状モデラとして、また、これまでのソリッドモデルでは困難であった問題を解決する手段として非常に有望なものと考えられる。

非多様体形状モデルを設計作業に利用した場合の形状処理を 図 1.6 に示す。非多様体形状モデリングでは、図の点線の矩形で示した広範な形状処理を単一の表現空間で扱うことができる。そのため、従来は別々のプロセスで行なわれてきた、ワイヤフレームモデリングやサーフェスモデリング、ソリッドモデリングを統一的に扱えるようになるため、非常に広範な範囲で利用ができるものとなる。また、必要に応じてワイヤフレーム、サーフェス、ソリッドの各表現の間を行き来することも可能となるため、ワイヤフレーム からサーフェス、ソリッドという変換もスムーズに行なえる。さらに、非多様体を利用した新しい応用分野にも用いることが期待できる。

1.3.2 非多様体形状モデリングシステム構築のための課題

非多様体形状モデルは、以上述べたように大変有望であり、多くの可能性を持っている。しかし、非多様体形状モデルは、ソリッドモデルに比べて研究の歴史が浅く、汎用的な形状モデリングシステムとして用いるためには、理論、形状処理手法、表現法、応用など、多くの面において研究が不十分である。

非多様体形状モデルではデータ構造の提案は初期の段階からなされていたものの [Weiler86b]、その基盤は非常に貧弱である。多くの点で、非多様体形状モデルはソリッドモデリング技術の ad hoc な拡張となっており、理論的な裏付けが十分とはいえない。またシステムを試作した上での評価がなされていなかったため、非多様体データ構造の現実的

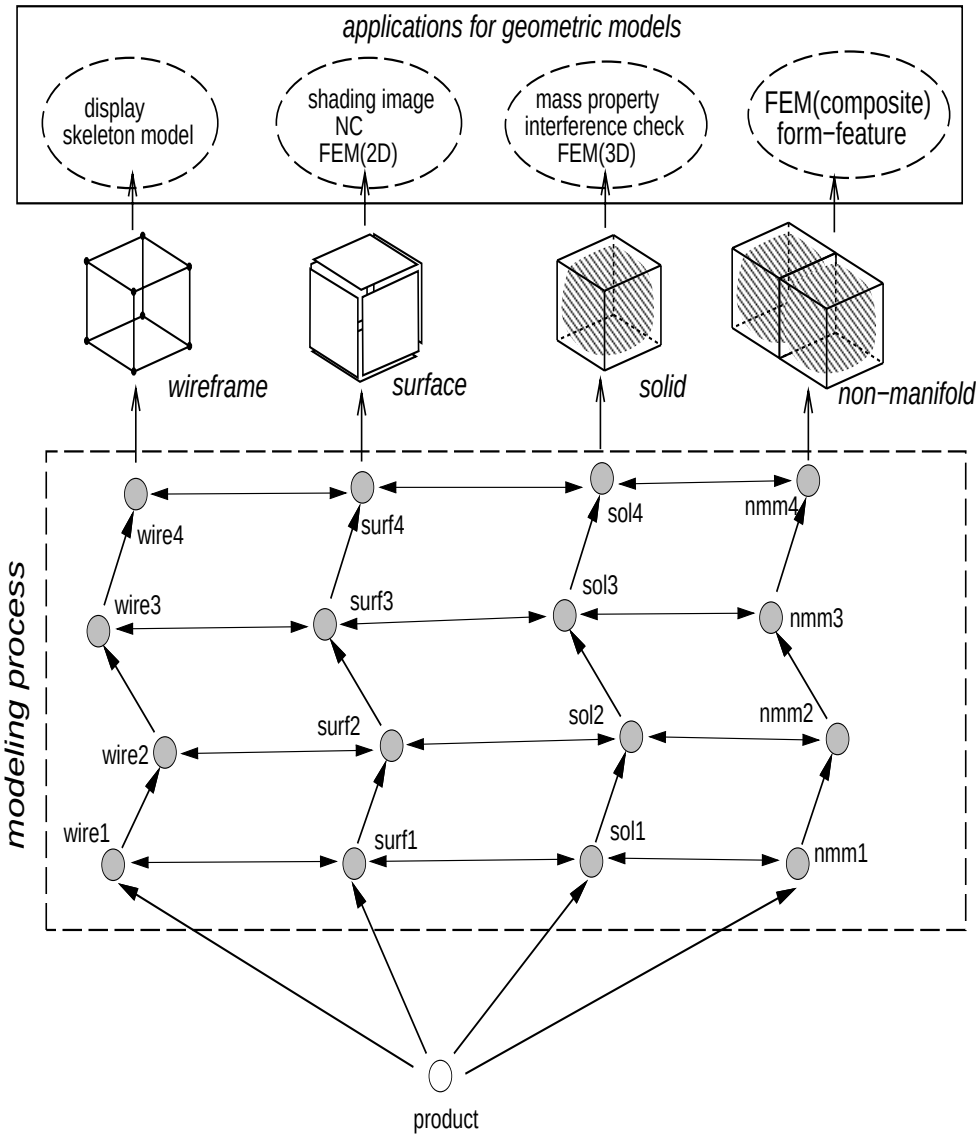


図 1.6: 非多様体形状モデラによる形状生成.

な有用性にも疑問の声が少なくなかった。さらに、利用可能な非多様体形状モデリングシステムがなかったので、非多様体形状モデルの応用システムも存在しなかった。

筆者は非多様体形状モデルの基礎研究と応用研究を、日本アイビーエム東京基礎研究所において、1987 年以来続けてきた。本論文はその成果をまとめたものであるが、研究当初は非多様体形状モデリングシステムを構築するにあたって、理論や形状処理手法に関して未解決な問題が多く、それらを解決することが不可欠であった。また、非多様体位相構造を用いることでどのような利点があるのかを、応用システムを構築することにより実証することも必要であった。

今日ではソリッドモデリングシステムが広く用いられているが、そこに至るまでには非常に多くの研究がなされている。本章の後半で述べるように、ソリッドモデリングでなされてきた主要な研究としては、

- [Requicha80] などによる形状モデルの数学的な定式化、
- [Baumgart74],[Braid80], [Mantyla82], [Chiyokura83b] などによる位相変形操作に関する研究、
- [Baumgart74], [Braid73], [Okino73], [Hosaka74], [Spur75], [Brun76], [Voelcker77], [Kimura84] などによる内部表現やシステム構築、工学的応用に関する研究、
- [Braid75], [Tilove80], [Requicha85],[Mantyla86], [Kobayashi87] などによる集合演算アルゴリズムに関する研究、

などが挙げられるが、これらの研究はソリッドモデルの定義域である 2-多様体または正則集合を対象としているため、非多様体形状モデルにそのまま適用することはできない。非多様体位相を考える場合には、従来の理論やアルゴリズムが適切であるかを形状モデリング研究の原点に戻って検討することが不可欠である。そして、従来の形状処理手法をより広範な幾何形状に適用できるように拡張していく必要がある。

非多様体形状モデリングにおいて、本質的に重要であり解決を必要とする問題には、以下のようなものがある。

1. 非多様体形状モデルの定義が明確でない.

   非多様体形状モデルについて考えるとき、まず必要となるのは、どのような幾何形状を対象に考えるかという問題である。非多様体形状モデルは、直観的には、ワイヤフレーム、サーフェス、ソリッドが混在した形状を統一的なデータ構造で

表現したものと考えることができる。しかしながら、ソリッドモデルが数学的に２-多様体として定義されていたのに比べて、この直観的な定義は曖昧であり、非多様体形状モデルがどのような位相的な性質を持った形状の集合を扱うのかが明確ではない。どのような条件の制約の元でワイヤフレーム、サーフェス、ソリッドの混在を許すのか、また、すべての形状モデルが共通に持つ性質は何なのかといったことが明確になっていないと、表現や操作の問題に関して厳密な議論をすることは困難である。

非多様体形状モデルの定義については、どこまで定義域を広げるかについて必ずしも統一的な見解があるわけではない。そのため、議論の前提条件を明らかにするためにも、非多様体形状モデルの定義を明確にすることが必要である。

2. 位相操作のための理論がない.

ソリッドモデリングにおいては、位相操作としてオイラー操作 [Baumgart74, Braid80] が広く用いられてきた。しかし、これらは２-多様体でのみ成立するものであり、非多様体形状モデルでは成立しない。オイラー操作はソリッドモデリングでは最も重要な位相操作であったため、非多様体形状モデルでそれに該当する操作が知られていないことは大きな問題であった。オイラー操作の利点の一つは複雑なデータ構造を隠蔽できることであるが、非多様体形状モデルのデータ構造はソリッドモデルよりもさらに複雑であるので、データ構造を隠蔽する必要度はソリッドモデルよりも高い。また、これまで逆操作などのオイラー操作の利点を利用した様々なアルゴリズムが開発されてきたが [Mantyla82, Chiyokura83b]、それらが利用できなくなることはモデリングシステムを構築する上で大きな制約となる。

そこで、非多様体形状モデルにおいても、オイラー操作に相当する位相操作群を導出する必要がある。

3. 非多様体データ構造で十分実用的な形状処理ができるか検証されていない.

非多様体形状モデルのデータ構造については既に Weiler が提案しており [Weiler86]、データ構造の基本的な部分は一般にこれに準じたものになると考えられる。ただし、形状モデリングがデータ構造に変更を加えていく操作であることを考えると、データ構造の管理のしやすさが非多様体形状モデラの性能に大きな影響を与える。しかしながら、Weiler の提案したデータ構造に基づいて形状処理システムを構築した場合、位相データの整合性を管理するための処理に時間がかかり、特に、サーフェスモデルを扱う際に、処理が重くなる傾向があることがわかっている。また、データ構造上、面で囲まれる閉空間と３次元点集合を同一視しているため、様々な形状処理の実装が複雑になると考えられる。このため、非多様体形状モデルの実用性については疑問の声が少なくなかった。

そこで、実用的に十分な効率で動作するためのデータ構造を設計し、その有効性を実際に形状モデリングシステムを構築することによって実証する必要がある。

4. 集合演算が未定義である．また、実現アルゴリズムが明らかになっていない．

集合演算は、幾何形状を生成していく上で最も広く用いられている操作である。ソリッドモデルでは集合演算に対して正規化集合演算という定義を与えているが、非多様体形状モデルにはこの定義は不適切であることがわかっている。したがって、まず、非多様体形状モデルの集合演算がどのような操作なのかを明らかにする必要がある。

さらに、非多様体形状モデルでは立体の内部構造も保持できるが、これまでの集合演算では境界位相要素のみしか考慮されていなかった。そのため、境界位相要素以外の内部構造を位相構造に反映させるための集合演算操作についても考慮する必要がある。

また、非多様体形状モデルでは、ワイヤフレーム、サーフェス、ソリッドが混在する形状も扱う。そのため、従来のソリッドモデリングよりも柔軟な集合演算アルゴリズムが必要となる。

5. 非多様体形状モデルの応用システムが示されていない．

非多様体形状モデルは柔軟な位相構造を持っているが、この性質がどのように利用できるかを示し、新たな応用システムを開発することは、非多様体形状モデルの有効性を示す上で非常に重要である。しかし、非多様体形状モデリングの潜在的な可能性についてはしばしば述べられるものの、具体的な応用システムについてはほとんど示されていなかった。

非多様体形状モデルが実際に有用であることを示すためには、非多様体位相構造の性質を用いた応用システムを開発し、その有効性を明らかにすることが重要である。

1.4 研究の目的

本研究は、機械設計のための３次元形状モデリングにおいて、ソリッドモデルよりも広範な範囲での利用が期待できる非多様体形状モデリングの基礎技術の確立とその応用を目的とする。そのために、非多様体形状モデルの表現や操作など、モデリングシステム構築に不可欠な基礎技術の研究と、その工学的応用に関する研究を行なう。

本研究の目的は以下の通りである。

1. 非多様体形状モデルをソリッドモデルの ad hoc な拡張と考えるのではなく、理論的な裏付けを明らかにし、非多様体形状モデリングのための理論的基礎を確立する。

2. 非多様体形状モデリングシステムを構築するために必要な基礎技術を確立し、実際にシステムを試作する。それによって、十分実用的な非多様体形状モデリングシステムが実現できることを確認する。

3. 非多様体形状モデルの広範な位相空間を利用することで、従来のソリッドモデリングでは実現が困難であった応用が実現できることを示す。それにより、非多様体形状モデルの有効性を示す。

以上の目的を達成するために以下に述べるようなアプローチを取るものとする。

まず、理論的基礎として、以下に示すように非多様体形状モデリングのための数学的な定義と位相変形操作の数学的導出法を示す。また境界表現形状モデルの集合演算の性質について考察する。

1. 非多様体形状モデルの定義域を議論の前提として明らかにする。

2. 非多様体形状モデルの集合演算に関して、従来の正規化集合演算に代わる定義を示す。

3. 非多様体形状モデルの位相構造を変形するための理論的基礎を明らかにし、非多様体形状モデルのための位相変形操作群が数学的に導出できることを示す。

   オイラー操作はソリッドモデリングでは最も基本的な位相変形操作であったが、非多様体形状においてはそれに相当する操作は知られていなかった。本研究では、非多様体形状モデルの位相構造を変更する手段として、複体のオイラー・ポアンカレの式を応用したオイラー操作の導出法について提案し、任意の非多様体形状

モデルの位相構造が９種類の基本操作群の組み合わせで実現できることを証明する。

次に、これらの理論的基盤に基づいて、非多様体形状モデリングシステムを試作する。そのために、以下のように、データ構造と集合演算の実現法を明らかにする。

1. 効率的な非多様体形状モデルのデータ構造について考察を行ない、本論文で示した定義域の形状を表現するために適したデータ構造を示す。

2. 非多様体形状モデルにおける集合演算の実現法を示す。ここでは、非多様体形状モデルの位相的な柔軟性を十分に反映させることのできる集合演算操作を考え、併合操作、抽出操作、簡略化操作を組み合わせた新しい集合演算アルゴリズムを示す。

次に、試作した非多様体形状モデリングシステムを中核としていくつかの応用システムを構築し、非多様体形状モデルの有用性を示す。ここでは、以下の形状処理が非多様体形状モデルを用いることで実現できることを示す。

1. 従来のソリッドモデルでは、集合演算操作の修正に時間がかかるために、試行錯誤的な形状生成の目的に利用することが困難であった。ソリッドモデルの修正では、操作の履歴の再実行と、直前の操作を順次取り消していく undo 操作の二つが知られていたが、いずれの場合も直接には関係ない他の操作も再計算しなければならないため、複雑な形状モデルにおいては非常に効率が悪かった。理想的な操作は、演算順序に依存せずに集合演算の取り消しが行なえる取消操作であるが、そのような操作を実現する手法はソリッドモデリングでは提案されていなかった。

   本研究では、取消操作が非多様体位相構造を利用することによって実現できることを示し、試行錯誤的な設計環境に適した形状処理として有効であることを示す。

2. 形状特徴は幾何形状の特定の部分に工学的意味を記述したものであるが、そのためには形状モデルの部分形状が適切に表現・管理されていることが必要である。しかし、従来のソリッドモデルでは、表現可能な部分形状に制限があった。そこで、非多様体位相構造を利用することにより、広範な形状特徴を表現・管理できる方法を提案する。

3. 非多様体形状モデルを応用した、三面図からソリッドモデルを自動合成する方法を提案する。本手法では、中間形状を非多様体形状モデルを用いて表現することにより、多数の候補ソリッドの記述を単一の形状モデルで行なうことが可能にな

り、また、処理の過程で現れるワイヤフレーム、サーフェス、ソリッドが統一的に表現できる。これらの利点を利用することで、簡潔で効率のよりアルゴリズムが実現できることを示す。

1.5 論文の構成

本論文の構成は以下の通りである。

第２章「非多様体形状モデルの定義」では、非多様体形状モデルの定義域を明確にし、以後の章における議論の前提条件を明らかにする。ここでは、まず非多様体形状モデルの数学的な定義について述べ、次に、非多様体形状モデルの集合演算について定義する。集合演算は、定義域について閉じていることが必要であるが、その条件を満たす集合演算を提案する。

第３章「非多様体形状モデルのオイラー操作」では、非多様体形状モデルのためのオイラー操作について提案する。ここでは、基礎となる数学的概念としてホモロジー群について述べ、複体のオイラー・ポアンカレの式を拡張することによって、非多様体形状モデルのための位相的な関係式が導出できることを示す。次に、導出された関係式に基づいてオイラー操作が定義でき、独立な９種類のオイラー操作があれば、その組合せで任意の非多様体形状モデルが生成できることを示す。また、導出した関係式を限定的に適用することによって、ワイヤフレームやサーフェスなど、様々な定義域の形状でのオイラー操作が導出できることも示す。最後に、オイラー操作の実現法についても論じる。

第４章「非多様体形状モデルの内部表現」では、本研究で試作した非多様体形状モデルのデータ構造について述べる。まず、これまでの非多様体データ構造の効率上の問題について論じ、３次元位相要素として volume 要素を用いることが効率の点で有利なことを述べる。その上で、非多様体形状の表現に必要な位相要素を定義し、そのデータ構造を示す。

第５章「非多様体形状モデルの集合演算と形状特徴表現への応用」では、非多様体形状モデルの集合演算に関して、集合演算の実現法と、その応用として、試行錯誤的環境での形状操作と、形状特徴表現の問題について論じる。本章では以下のことが述べられる。

- 非多様体形状モデルの集合演算として、併合操作、抽出操作、簡略化操作に基づいた集合演算操作を提案する。
- 集合演算の取り消しが、プリミティブの個数や演算順序に依存せず、極めて高速に行える手法を提案する。また、プリミティプの優先順位を反映した集合演算の評価法についても示す。
- ソリッドモデルを用いた形状特徴表現の問題点について論じ、非多様体形状モデ

ルを用いた形状特徴表現法について述べる。

- 集合演算を実装するための具体的なアルゴリズムを示す。

第６章では、非多様体形状モデルを応用した例として、三面図からのソリッドモデル生成問題について述べる。ここでは、三面図からソリッドモデルを作成する問題に非多様体形状モデルが有効に利用できることを述べ、セル分割表現と推論システムを利用したソリッド合成アルゴリズムを提案する。また、誤りを含んだ三面図からのソリッド合成法についても論じる。

最後に第７章で、本研究のまとめを行なう。

1.6 ソリッドモデリングの基礎技術

非多様体形状モデルに関する研究は、これまでなされてきたソリッドモデルの研究が基礎となっている。そこで、ソリッドモデルに関する主要な研究について本節で説明しておく。

主要なソリッドモデリング研究

３次元形状モデリングの研究は、1960 年代半ばからいくつかの大学で行なわれてきたが [Braid93]、1973 年に Budapest で開かれた PROLAMAT Conference で、境界表現と CSG 表現のソリッドモデルが提案されて以来、多くの研究者の注目を集めるようになった。この会議では、Braid と Lang が Cambridge University で開発された形状モデラ BUILD に関する発表を行ない、幾何形状を境界表現 (boundary representation) によって記述する方法を提案した [Braid73] 。Braid らは、その後 BUILD-2 の開発を行ない、穴や空洞のある形状が扱えるように、データ構造と位相操作を拡張した [Braid80]。また、沖野らは北海道大学で開発された形状モデラ TIPS に関する発表を行ない、CSG 表現 (constructive solid geometry) による立体表現を提案した [Okino73]。

1974 年には、Baumgart が Stanford University において視覚認識の目的で形状モデラ GEOMED を開発し、今日の境界表現ソリッドモデルの基礎となっている winged-edge 構造とオイラー操作の概念を提案した [Baumgart74]。

1970 年代には、実験的なソリッドモデリングシステムの開発が盛んに行なわれた。1974 年には、東京大学で穂坂と木村らが境界表現モデラ GEOMAP の開発を行なった [Hosaka74]。このシステムはその後、曲面処理を含んだ GEOMAP-III へと発展し [Kimura84] 、ソリッドモデルがロボットの軌道生成や NC 加工、視覚認識など、設計生産の自動化に有効なツールであることを示した。また、東京大学の千代倉と木村は、形状生成過程が管理できる形状モデラ MODIF を開発し、オイラー操作の逆操作を利用した undo 操作が試行錯誤的な形状生成に有効であることを示した [Chiyokura83b]。1975 年には Technical University of Berlin で Spur ら が COMPAC の開発を行なっており [Spur75, Spur82]、1976 年には University of Grenoble で Brun が EUCLID の開発を行なった [Brun76]。また、1976 年には、University of Rochester で Voelcker と Requicha が CSG 表現に基づいた形状モデラ PADL-1 を開発した [Voelcker77] 。Voelcker らのグループは、ソリッドモデルの数学的扱い [Requicha80] や集合演算アルゴリズムの開発 [Tilove80] などにおいて重要な役割を果たしている [Voelcker93]。また、Mantyla らは Helsinki University of Technology で オイラー操作に基づいた形状処理

システム GWB (Geometric Work Bench) を開発している [Mantyla82]。これらの大学主導で開発されたシステムの基礎理論やアルゴリズムは、1980 年以降に現れた商用のソリッドモデラ ROMULUS, GMSOLID, UNISOLIDS, BRAVO などに大きな影響を与えている [Voelcker93]。

これらの先駆的な研究を通して確立されてきたソリッドモデリングの基本技術としては、以下に説明するようなものが挙げられる。

境界表現と CSG 表現

ソリッドモデルの表現法としては、Braid らによって提案された境界表現 [Braid73] と沖野らによって提案された CSG 表現 [Okino73] が広く用いられている。立体形状の CSG 表現と境界表現を図 1.7 に示す。

CSG 表現は、直方体や円筒などの単純な立体である基本立体 (primitive) と、それらの和、差、積といった集合演算や剛体変換を組み合わせた式を保持することによって立体を表現する方法である。この式は、図 1.7 に示したような、基本立体を葉 (leaf)、集合演算や剛体変換を節 (node) とする CSG ツリーと呼ばれる木構造で表現することができる。CSG 表現で保持された形状モデルは評価されていない形状 (unevaluated shape) といわれる。

境界表現は、立体の内部と外部を隔てる境界面によって立体形状を表現する方法である。境界面は閉じた殻（境界のない 2-多様体）となっており、それらは頂点、稜線、面といった位相要素に分けられ、それらの位相要素間の隣接関係がグラフ構造によって保持される。この表現では、立体形状の位相要素が陽に保持されているので、評価された形状 (evaluated shape) といわれる。解析やプロセスプラニングなどの応用システムでは、形状データを直接参照する必要があるため、境界表現が一般に用いられる。

CSG 表現と境界表現を比較すると、CSG 表現では保持するデータ量が少なく、表現も単純であるという利点があるが、立体の面や稜線の情報を得るためには集合演算式を評価して境界表現データに変換する必要がある。 一方、境界表現ではデータ量が多くなり、管理するグラフ構造も複雑であるが、境界データを陽に保持しているために解析など多くの応用分野に適用ができるという利点がある。現在ではこれらの表現はどちらも必要であり、CSG 表現 は必要に応じて境界表現に変換できるので、境界表現モデラと CSG モデラの差は、どちらをマスターモデルにするかという問題に過ぎないと考えられている [Wilson86]。

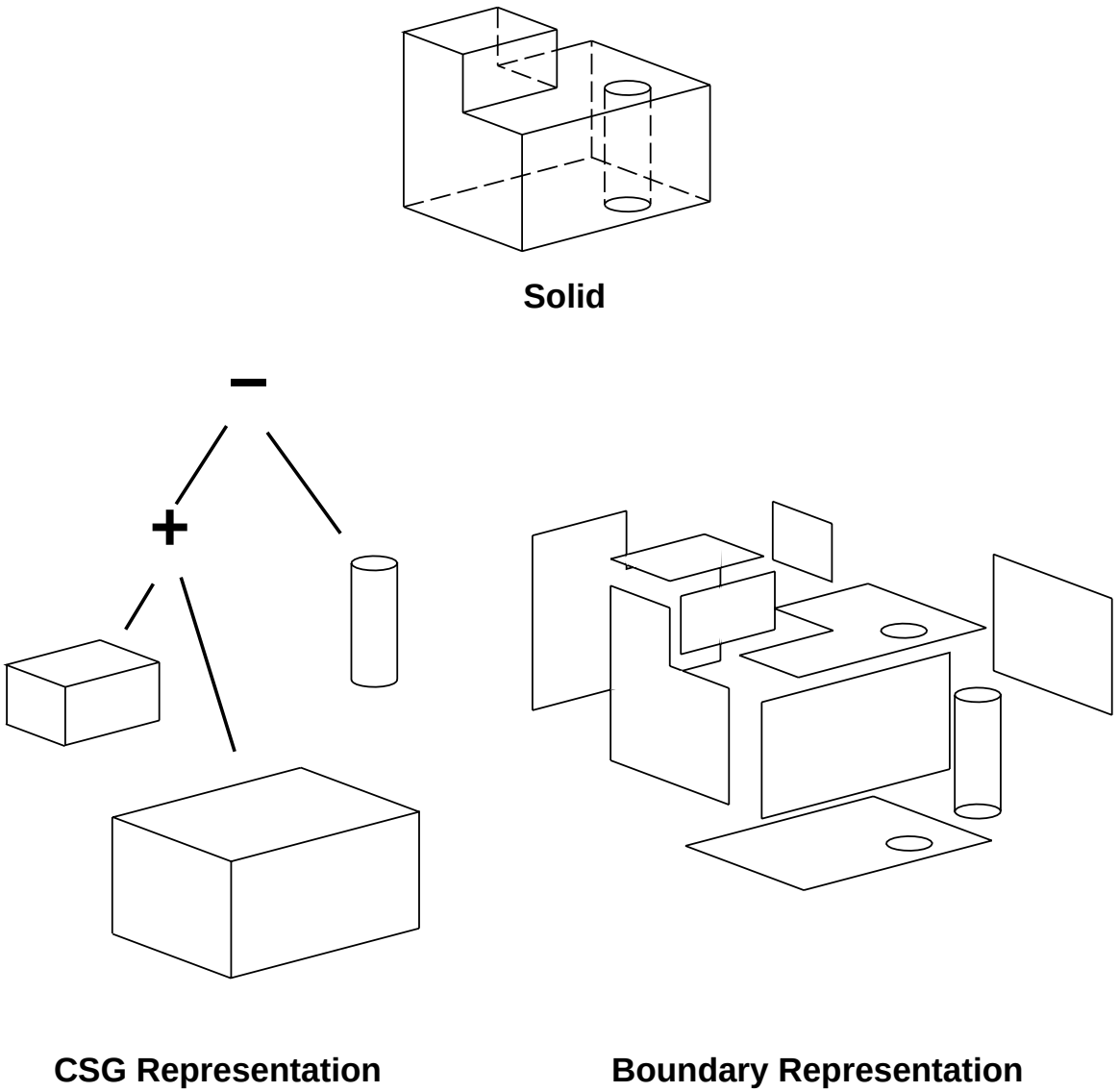


図 1.7: CSG 表現と境界表現.

ソリッドの定義域

Requicha は、1980 年にソリッドモデルの数学的な定義を示し、「有界」「正則」「半解析的」な形状の集合をソリッドモデルの定義域とした [Requicha80]。そのような形状の集合は正則集合と呼ばれる。ただし、境界表現においては、通常、正則集合の部分集合である 2-多様体 (2-manifold) を定義域とする [Braid75, Mantyla88]。2-多様体とは境界表現で保持する立体の殻の部分が球面と同相でなければならないという条件のことである。稜線が常に二つの面に共有されることが保証される 2-多様体形状は比較的単純なデータ構造で表現できるので、境界表現にとって都合がよいと考えられる。

また、Requicha は正則集合である形状間の集合演算についても数学的な定義を行ない、集合演算の結果が正則集合となることが保証される正規化集合演算を提案した [Requicha80]。この定義は今日のソリッドモデリングで広く認められている。ソリッドモデリングで用いられている集合演算とは一般に正規化集合演算の意味である。

位相操作

境界表現ソリッドモデルでは、データ構造が複雑なため、どのようにして位相構造の整合性を保ちながらグラフ構造を変更していくかが大きな問題となる。そのための位相操作としては、Baumgart の提案したオイラー操作 [Baumgart74] が広く用いられている。オイラー操作は形状変形する際の最も原始的な操作であり、オイラー操作を組み合わせることで任意の位相変形操作が記述できる。オイラー操作の利点は次のようにまとめることができる [Braid80, Mantyla82, Chiyokura83a]。

- オイラー操作は位相要素の個数に関して整合性を保持するので、それらに基づいて実装された形状変形操作でも位相要素の個数の整合性が保てる。
- オイラー操作は実際のデータ構造を隠蔽するので、アルゴリズムが特定のデータ構造に依存する度合が少なくなる。
- オイラー操作の組合せで形状処理を実装することによりシステムのモジュラリティが高まり、プログラムの信頼性の向上や保守の容易さが期待できる。
- オイラー操作は一対一に対応する逆操作が存在するため、形状変形後に位相データを元の状態に復元する逆操作が容易に実現できる。

Baumgart の提案したオイラー操作は、Euler の多面体定理（L. Euler, 1752）に基づいて形状変形する方法である。この定理は、球面と同相な多面体の頂点、稜線、面の

個数をそれぞれ、v, e, f としたとき、$v - e + f = 2$ が成立するというものである。オイラー操作は、これらの変数を単位量変化させるような原始的な操作群として定義される。今日広く用いられているオイラー操作は、Braid らが穴や空洞を含んだ形に拡張したもので [Braid80] 、面の穴の個数 r と貫通穴の個数 h、連結位相要素 (shell) の個数 s を変数に加えた式、

$$v - e + f - r = 2\ (s - h)$$

に基づいた操作群である。

図 1.8 は Mantyla らによって提案されたオイラー操作の組である [Mantyla82]。図 1.8 において、(a) の mvsf, kvsf は それぞれ 'make vertex, shell, and face' と 'kill vertex, shell, and face' の略で、vertex, shell, face を同時に生成または消滅させる操作である。また、(b) の mev, kev は 'make edge and vertex', 'kill edge and vertex' を意味している。このような make/kill の対は一方が他方の逆操作となっている。図 1.9 はこれらのオイラー操作を用いて貫通穴のあるソリッドモデルを生成する手順を示したものである。この図では、図 1.8 に示したオイラー操作を組み合わせることで、ソリッドモデルの位相構造が順次生成されていく様子が示されている。

内部表現

境界表現ソリッドモデルでは、立体を面、稜線、頂点の集まりとして表すので、それらの間の関係をどのように記述するかが重要な問題である。その記述を立体の内部表現、またはデータ構造と呼ぶ。

境界表現ソリッドモデルの内部表現においては、Baumgart の提案した winged-edge 構造 [Baumgart74] が広く用いられている。winged-edge 構造は、比較的少ない保持データで位相要素の隣接関係と順序関係が表現できるという特徴がある。このデータ構造では、図 1.10 に示すように、ある edge に関して隣接する face や edge を、立体を外から見たときの左右の区別をつけて保持する。

また、winged-edge 構造と等価な位相情報を持つデータ構造として、位相データの探索が高速にできる half-edge 構造も用いられている [Mantyla88]。このデータ構造では、図 1.11 のように、face を反時計回りに回るときの向きに応じて edge を二本に分けることによって、face を一周廻るときの効率を向上させている。

図 1.8: ソリッドモデルのオイラー操作 (Mantyla,1982).

図 1.9: オイラー操作を用いた形状生成 (Mantyla,1982).

図 1.10: winged-edge 構造 (Baumgart,1975).

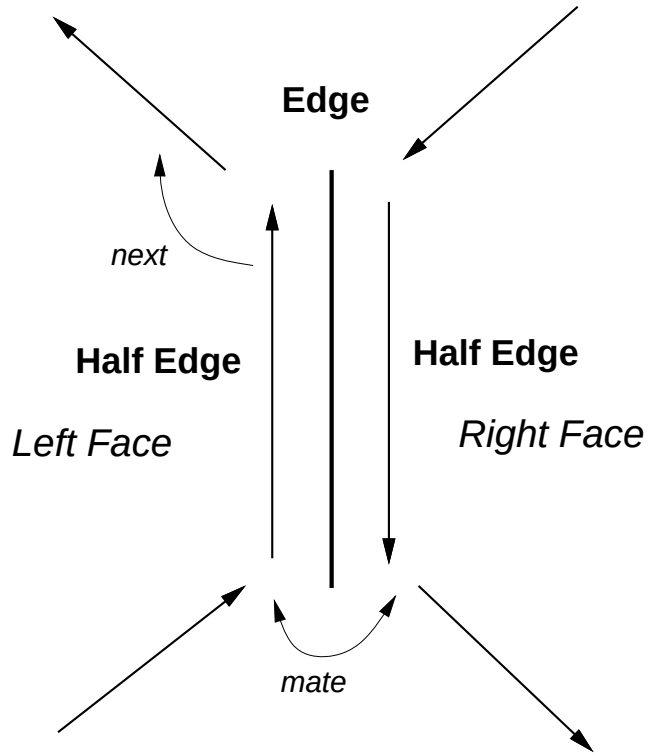


図 1.11: half-edge 構造.

集合演算

ソリッドモデルを作成する手段として、最も広く用いられている操作が集合演算である。集合演算による形状生成では、立体の和、差、積を計算していくことによって、徐々に複雑な幾何形状を作成していく。集合演算の実装においては、曲面間の干渉計算や位相操作、数値計算の安定性など非常に多くの問題があり、これまで多くの研究がなされてきた [Braid75, Tilove80, Requicha85, Mantyla86, Kobayashi87]。

比較的よく用いられていると思われる集合演算アルゴリズムは、1980 年に Tilove によって提案された分類に基づくものである [Tilove80]。この方法では、二つの立体の包含関係を IN, ON, OUT に分類し、それらの組み合わせによって和、差、積に相当する立体形状を作成する。図 1.12 は、二つの立体 A, B から作成される断片形状を示しているが、この例題においては、A と B の和、差、積はそれぞれ、($AoutB$ & $BoutA$), ($AoutB$ & $BinA^{-1}$) , ($AinB$ & $BinA$) として得ることができる。$BinA^{-1}$ は、面の向きの反転を意味している。

Tilove のアルゴリズムは多面体を対象としているが、現在では自由曲面を含んだ形状の集合演算に拡張されている [Chiyokura83a, Kimura84, Kobayashi89]。また、winged-edge 構造を用いたソリッドモデルで、サーフェスと立体との集合演算を行なう手法も提案されている [Satoh91]。

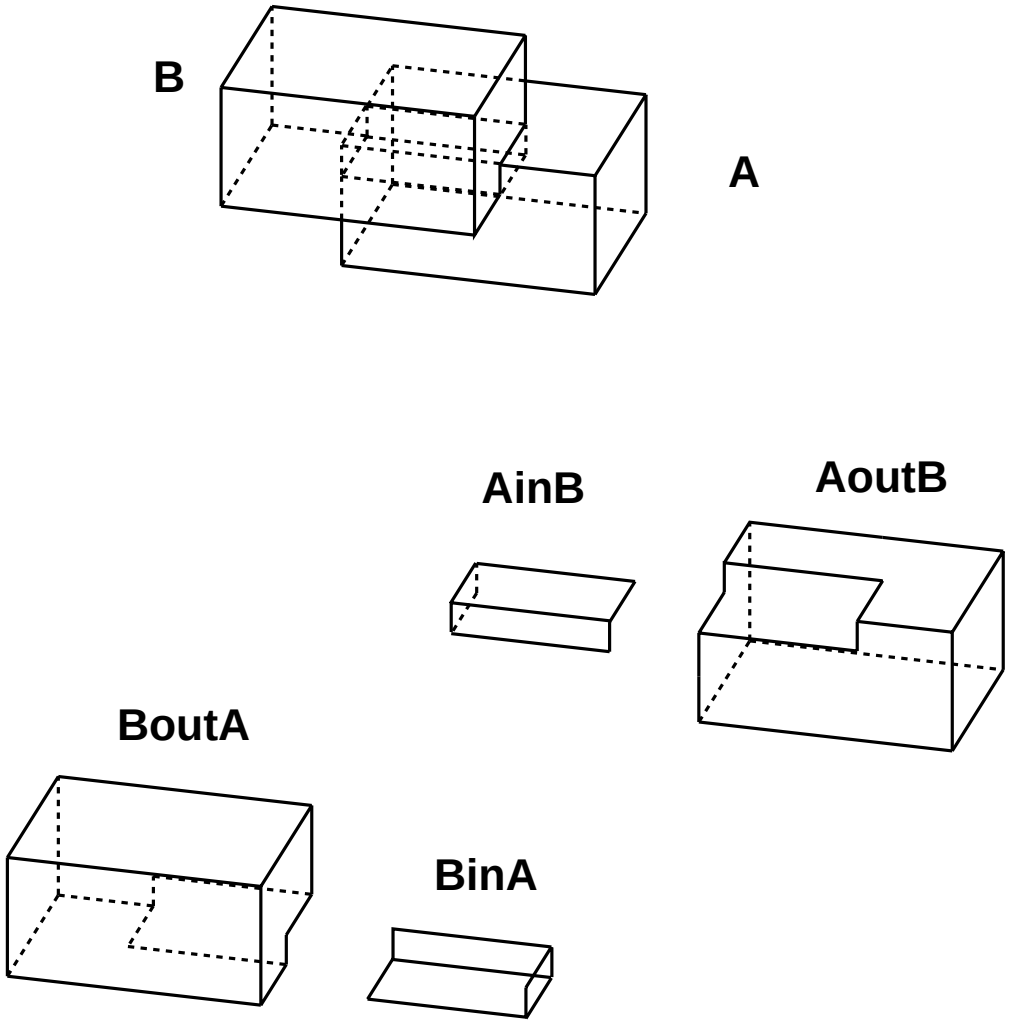


図 1.12: ソリッドモデルの集合演算アルゴリズム (Tilove, 1980).

1.7 非多様体形状モデリング研究

本節では、前節で述べたソリッドモデリングの基礎技術に対応させ、非多様体形状モデリング研究について概観する。すなわち、(1) 定義域、(2) 位相操作、(3) 内部表現、(4) 集合演算、(5) 応用システムに関してこれまでの非多様体形状モデルの研究について述べ、さらに本研究の果たしてきた役割についても概観する。ここでは本研究の位置付けを簡単に述べるだけにとどめ、詳細は２章以後に述べるものとする。

1.7.1 定義域

非多様体形状モデルについて考える場合、どのような範囲の幾何形状を表現対象とするかを規定することは極めて重要なことである [Weiler86]。ただし、どのような定義域を取るべきかは、形状処理システム全体としての整合性や、応用分野に応じて考えていくべきもので、一概にどれが優れているかを論じることはできない。ここでは、いくつかの研究で採用されている定義域を、主として設計生産の支援という観点から見ていくことにする。

非多様体位相を含む幾何形状の数学的な記述については、1986 年に相澤が提案した胞複体鎖を用いた幾何形状の数学モデルがある [Aizawa86, Aizawa89]。この研究では、幾何形状を胞複体鎖を用いて表現することにより幾何形状を代数的手法によって扱う方法を示しており、胞複体鎖によって定義域を規定している。

また、Weiler は 1986 年に、面の位相的条件やグラフ構造の制約などの観点から非多様体形状モデルの定義域を示した。この定義では、(1) 非多様体サーフェスが定義域に含まれ、(2) face が多様体で平面に写像可能であり、(3) 位相要素が互いに交わらず、(4) 有界であり、またグラフ構造として (5) セルフループなどの擬グラフ (pseudograph) や、(6) 面の穴などの不連続グラフ (disconnected graph) が許され、(7) グラフの構成要素がラベル付けされている、ということを規定した。同様の定義域は、[Gursoz88, Kobayashi89, Yamaguchi90, Crocker91] など、多くの非多様体研究で用いられている。

ただし、Weiler の示した定義の記述では、定義域の幾何形状が共通にもつ位相的な性質が必ずしも明らかになっていず、正則集合や２-多様体といった定義に比べると曖昧であることが増田らによって指摘された。そして、非多様体形状モデルの定義域を複体の定義に準じて定めることで位相操作などの問題が理論的に扱えるようになることを示した [Masuda88b, Masuda89a]。また、増田らはその定義域に関して閉じている集合演算についても定義を示している [Masuda89b, Masuda93a]。これらの研究は、本論

文の第２章で述べられる。

Wu も複体を定義域とする形状モデル表現を提案しているが [Wu89]、円のように境界を持たない位相要素も許している一方で、貫通穴や空洞を持つ位相要素は許していない。そのため、機械部品の表現には必ずしも適しているとはいえない。

Rossignac と Requicha の研究では、複体の構成要素である胞体の部分集合を考えることにより、閉じていない形状も定義域とした [Rossignac91] 。この定義域には有界で構成要素が有限であるほとんどの幾何形状が含まれるが、境界を持たない形状を定義域に含めることによりどのような利点が生じるのかは明らかにしていない。

1.7.2 位相操作

非多様体形状モデルの位相操作では、1986 年に Weiler の提案した NMT オペレータが知られている [Weiler86c]。この研究では、基本変形操作を非多様体データ構造を変更するためのマクロ関数として捉えており、非多様体形状モデルの生成や変更のための基本的な位相操作群を提案している。図 1.13 は、NMT オペレータの一部を示している。この図において、たとえば K_V は 'kill vertex' の略で、kvfle, kve などは K_V のバリエーションを示しており、それぞれ 'kill vertex, face, loop, and edge'、'kill vertex and edge' の意味である。これらの操作はソリッドモデルのオイラー操作を参考にして決められたが、オイラー式のような位相的な関係式に基づいているわけではなく、理論的な背景はない。そのため、(1) 操作の決め方が場当たり的であり、操作の個数が非常に多い、(2) どのような基本操作を用意すれば十分かが不明、(3) NMT オペレータが定義域に関して閉じているかが不明、いう問題点がある。

増田らは、1988 年に、非多様体形状モデルに関する位相的な関係式を導出することによって、ソリッドモデルのオイラー操作と同等の性質を持つ操作群を導出する手法を示した [Masuda88b, Masuda89a]。この方法では操作群が数学的に導かれるので、NMT オペレータで生じたような問題点は解決される。この研究では、複体におけるオイラー・ポアンカレの式 [数学 2, 数学 3] を貫通穴や空洞のある場合に拡張した関係式、

$$v - e + (f - r) - (V - V_h + Vc) = C - Ch + Cc$$

(v, e, f, V はそれぞれ vertex, edge, face, volume の個数、r, Vh は face, volume の穴の個数, Vc は volume の空洞の個数, C, Ch, Cc はそれぞれ形状モデルの連結成分, 貫通穴, 空洞の個数) を導出し、９種類のオイラー操作の組み合わせで任意の非多様体形状モデルが作成できることを証明した。図 1.14 に９種類のオイラー操作を示す。この研究に関しては、本論文の第４章で述べられる。

図 1.13: NMT オペレータ (Weiler,1986).

図 1.14: 任意の非多様体形状モデルの生成に十分なオイラー操作群 (Masuda,1988).

また、Wu は、1989 年に増田らとは独立にオイラー・ポアンカレの式に基づいた複体モデルのための関係式を求めている [Wu89]。ただし、この研究では、形状を位相的にいくつかの集合に分類した上でそれぞれのオイラー式を導出しており、式に現れる変数もかなり多い。そのため、式の適用条件の判別が難しいという問題がある。

また、木村, 山口, 小林は、非多様体形状モデリングを無限空間の分割と捉えることによって増田の式の変数の個数を減らせることを指摘した。そして、s を閉空間の個数、h をハンドル (handle) の個数、c を空洞 (cavity) の個数として、

$$v - e + (f - r) - (s - h + c) = 0$$

という関係式に基づくオイラー操作を提案した [Kimura89]。さらに、山口と木村は、非多様体形状モデルの位相データが補助的な位相要素を介して記述されていることに着目して、連結位相要素の個数を変数にした関係式についても提案している [Yamaguchi90]。

また、正則集合 (r-set) を対象としたオイラー操作の研究もある。正則集合には非多様体稜線を含んだ形状も含まれるため、2-多様体のオイラー式は一般には成立しない。Desaulniers らは、正則立体に仮想的な面を生成することで、2-多様体を組み合わる問題に帰着させた [Desaulniers92]。そして、仮想的な面を必要とする頂点 V^*, 稜線 E^*, 面 F^* のためのオイラー操作を加えることによって正則立体のオイラー操作を求めた。また、東らは、非多様体の稜線や頂点に縮退したループや稜線があると仮定して、2-多様体に帰着させる方法を示した [Higashi90] 。

1.7.3 内部表現

非多様体形状モデルのデータ構造としては radial-edge 構造がよく知られている。図 1.15 に radial-edge 構造を示す。この研究では、稜線回りの面の順序を陽に表現することの重要性を示し、また、use と呼ばれる補助的な位相要素を導入することにより、固定長データで形状表現ができることを示した。また、形状表現のための階層構造として、図 1.16 のようなデータ構造を提案した。

また、Gursoz らは、1988 年に vertex 回りの隣接関係に着目したデータ構造を提案し、頂点接触する場合の位相構造も表現できる tri-cusp 構造を提案した [Gursoz88]。この表現は、非多様体形状モデルの vertex-based 表現ともいわれる。図 1.17 に、tri-cusp 構造による vertex 回りの位相表現を示す。

また増田らは、面に囲まれた閉空間として定義される位相要素 region を用いた図 1.16 のようなデータ構造では、3 次元点集合と閉じた 2 次元点集合を同一視しており、ま

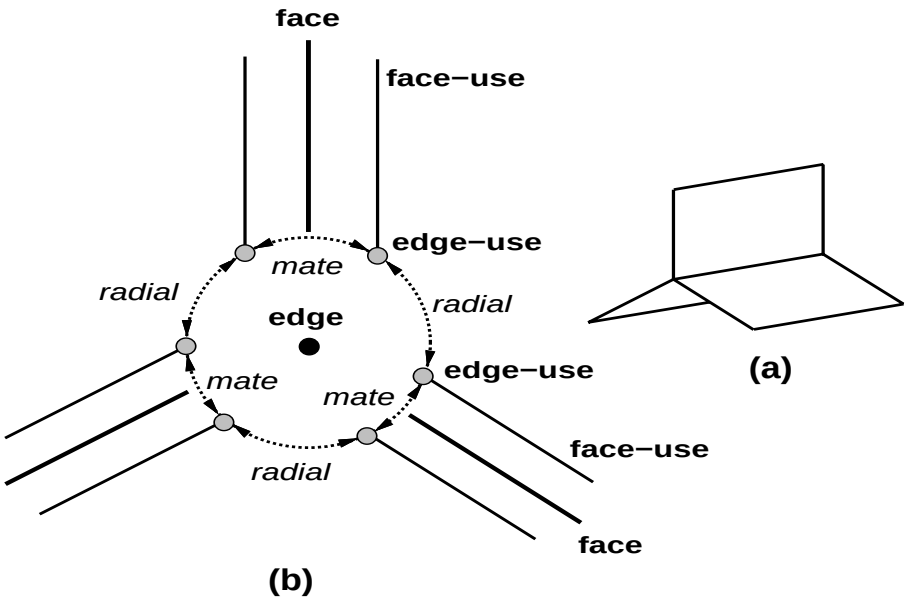


図 1.15: radial-edge 構造 (Weiler,1986).

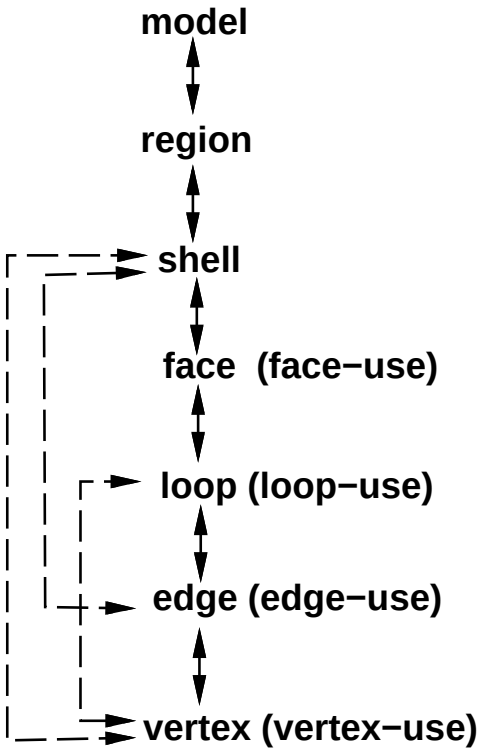
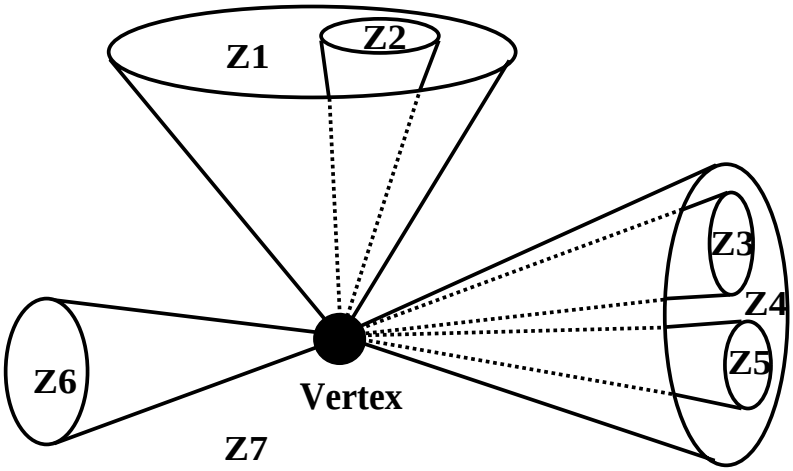
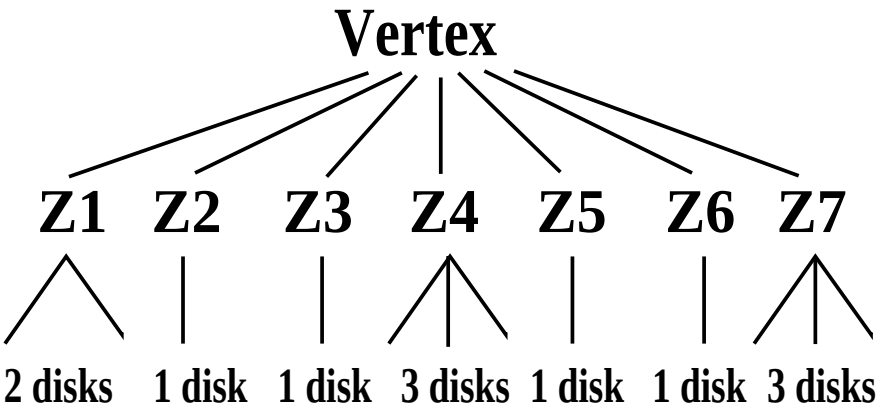


図 1.16: radial-edge 構造の階層構造 (Weiler,1986).



(a) A vertex shared by disks.



(b) Structure

図 1.17: tri-cusp 構造による vertex 回りの位相構造 (Gursoz,1988).

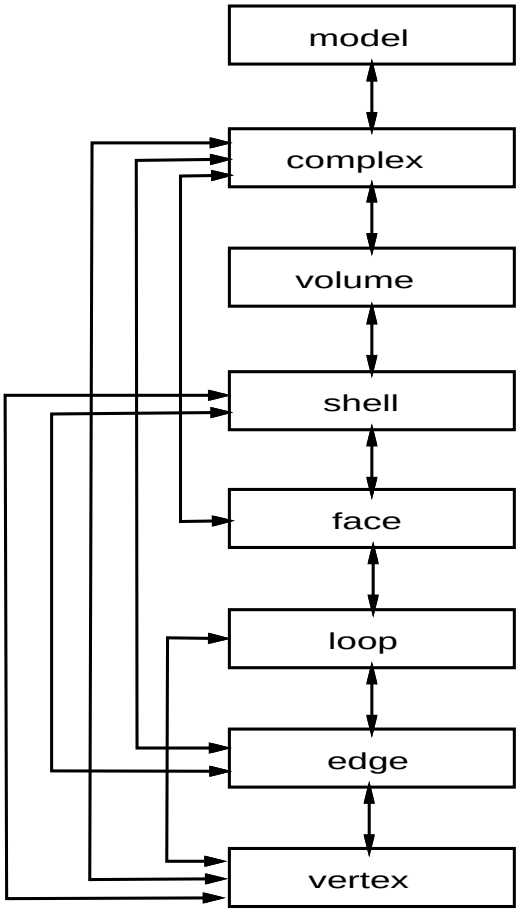


図 1.18: ３次元位相要素 volume を用いたデータ構造 (Masuda,1988).

た位相構造の整合性の管理に閉空間探索という大域的な操作が必要となるために効率上問題があることを指摘した。そして、ソリッド部分を陽に表現する位相要素 volume を用いた図 1.18 のような階層構造を持つデータ構造によってこれらの問題が解決できることを示した [Masuda88a, Masuda89a]。このデータ構造については、本論文の第３章で述べられる。

また、山口は非多様体位相を表現するための隣接関係を分類し、非多様体データ構造を隣接位相要素という考え方によって整理する方法を提案した。また、隣接位相要素に基づいたデータ構造の提案も行なっている [Yamaguchi90]。

また、形状モデルの次元を一般化した高次元形状モデルのデータ構造の研究もなされている。ただし、高次元形状モデルについては、現時点では工学的な有用性が不明であるため、ここでいくつかを簡単に紹介するにとどめる。

Lienhardt は、 n-G-map を用いて胞体分割された n 次元形状を表現する方法を提案した [Lienhardt89]。Brisson は定義域を多様体の胞体分割に限定しているが、一般次元の形状が統一的に扱える cell-tuple 構造を提案した [Brisson90]。Brisson のデータ構造では、位相要素間の隣接関係と順序関係が簡潔に表現されており、従来の２-多様体ソリッドモデルのデータ構造を n-多様体に一般化したものとなっている。また、Paoluzzi は、n 次元の単体的複体に限定した形状処理について論じている [Paoluzzi93]。Rossignac と O'Connor は、n 次元形状の隣接関係を保持する表現を提案しているが [Rossignac90]、順序関係が表現できていないため形状モデリングのための表現には適していない 。

1.7.4 集合演算

非多様体形状モデルの集合演算についても研究がなされている。

増田らは、1989 年に、Requicha がソリッドモデルで正規化集合演算を定義した方法 [Requicha80] を拡張し、非多様体形状モデル間の集合演算結果が定義域内に収まるような集合演算の定義を示した [Masuda89b, Masuda93a] 。それについては、本論文の第２章で述べられている。また、Rossignac と Requicha は、境界を持たない形状も含んだ定義域での集合演算の定義を示している [Rossignac91] 。

また、増田らは 1988 年に、非多様体形状モデルの集合演算として、演算結果を非多様体位相構造を用いて表現し、非多様体形状モデルの位相要素を選択的に取り出して集合演算を評価した形状を得る方法を示した [Masuda88b, Masuda89a]。この研究では、図 1.19 に示したような二つのプリミティブＡ, Ｂの集合演算の結果として、中央に示した非多様体位相構造を保持し、集合演算の種類に応じて、和集合、差集合、積集合に相当

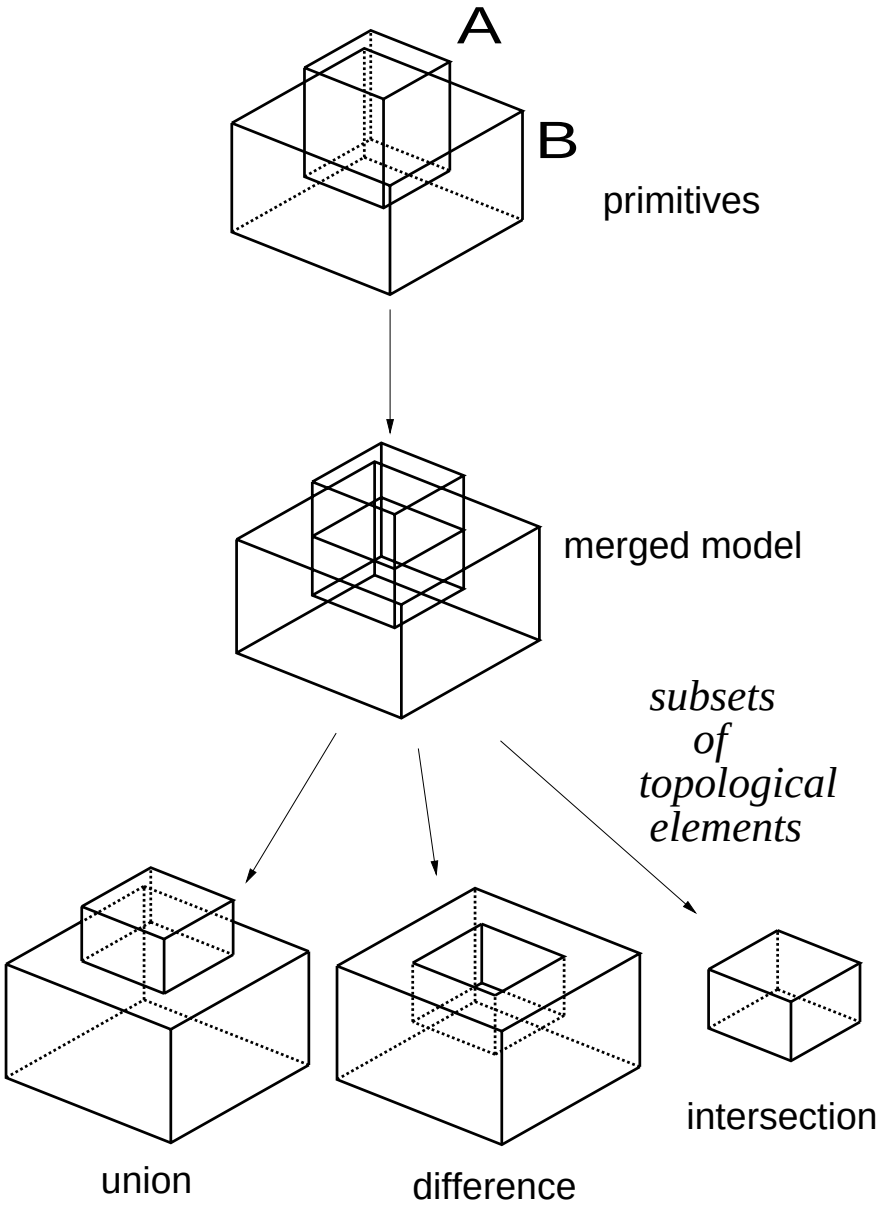


図 1.19: 非多様体位相構造から抽出される和集合、差集合、積集合 (Masuda,1988).

する位相要素を抽出する方法を示している。また、この位相表現の応用として、集合演算を演算順序に関係なく、局所変形操作だけで取り消す操作を実現する方法も示した。これらの集合演算手法については、本論文の第５章で述べられる。また、Rossignac らは位相要素を選択的に取り出す方法を境界を持たない幾何形状にまで拡張した場合について論じた [Rossignac90]。

Wilson は、1986 年に、非多様体位相構造を用いて境界表現と CSG 表現を融合した Brep/CSG ハイブリッド表現について提案した [Wilson86] 。Wilson の方法は、CSG ツリーの節 (node) に対して、非多様体位相構造との関連付けを行なっている。増田らは、Wilson の方法では対応関係を管理する負荷が極めて大きくなるという問題点を指摘し、CSG ツリーの葉 (leaf) のみを非多様体位相構造と関連付けた Brep/CSG ハイブリッド表現を示し、境界表現の位相構造に CSG 表現のプリミティブを埋め込む方法について示した [Masuda88a]。それについても、本論文の第５章で述べられる。

集合演算を実装するためのアルゴリズムを提案した研究としては、Gursoz らの研究 [Gursoz91] 、増田の研究 [Masuda91, Masuda93a] 、Crocker らの研究 [Crocker91] がある。いずれも、ワイヤフレーム、サーフェス、ソリッドの混在に対応して、vertex, edge, face 間の干渉を分類し、位相構造を構築するものとなっている。ただし、増田の研究は、集合演算を併合操作と抽出操作の二つの操作に分解し、それぞれの実現法を示した点で他の方法とは異なったものとなっている。図 1.20 は、増田の示した併合操作を実現するための処理手順である。これらについても、本論文の第５章で述べられる。

1.7.5 応用システム

形状特徴

Pratt は、[Pratt88, Pratt89] において、非多様体形状モデルが体積特徴の表現に必要な付加データの保持に利用できることを示した。しかし、Pratt の提案した表現法では対象が体積特徴に限定されており、また、形状特徴が干渉した場合の対処が難しいという問題がある。

増田は、形状特徴をユークリッド空間における点集合に対して工学的属性を記述したものと考え、線特徴、面特徴、体積特徴を統一的に表現でき、干渉が生じる場合でも整合性を保持できる表現方法について提案した [Masuda92a, Masuda93b]。図 1.21 は、製品形状には現れない付加的な位相要素を保持することによって、形状特徴が位相構造中に保持できることを示している。これについては、本論文の第５章で述べられる。

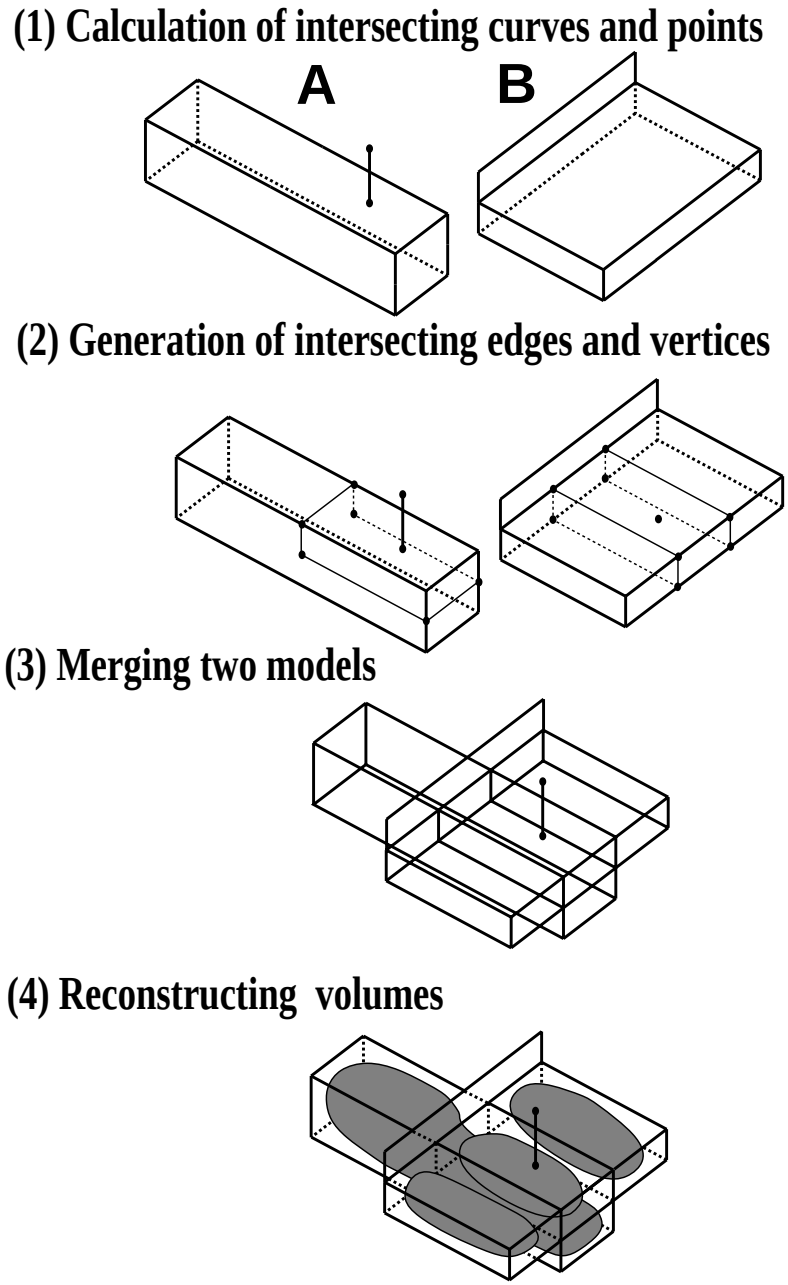


図 1.20: 併合操作による非多様体位相構築のための処理手順 (Masuda,1991).

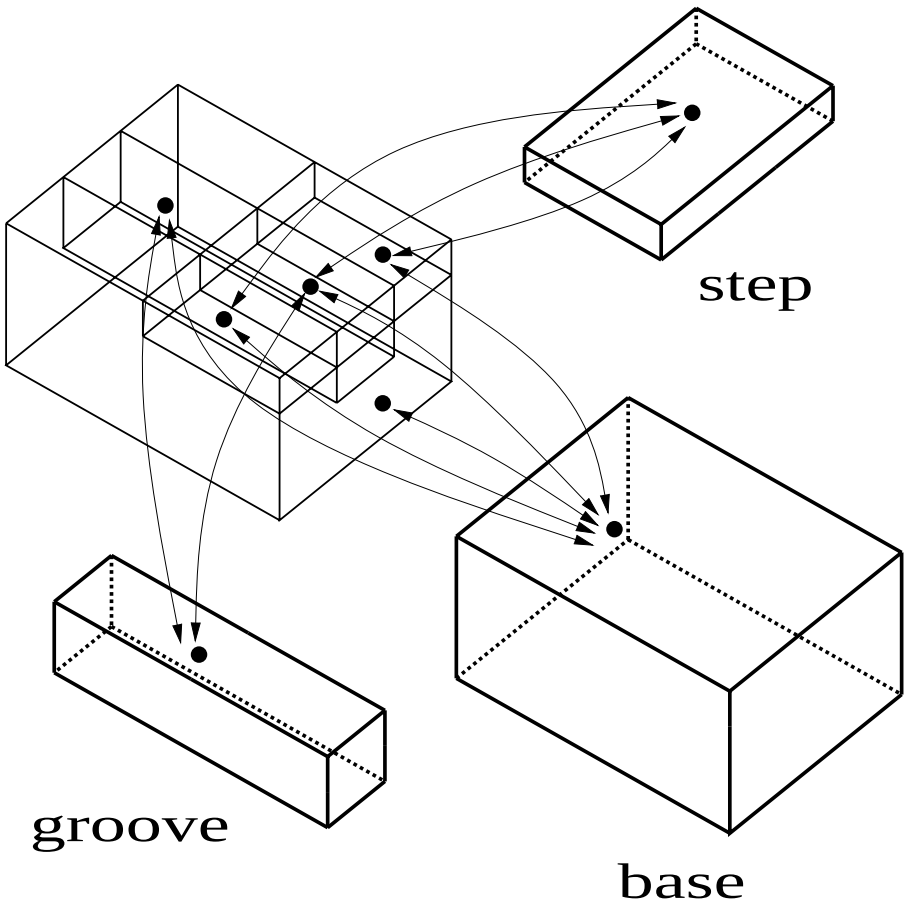


図 1.21: 非多様体形状モデルを用いた形状特徴の保持 (Masuda,1992).

ソリッド合成

また、増田らは、三面図からソリッドモデルを合成する問題を非多様体形状モデルと推論システムを組み合わせて解く手法を提案している [Masuda94a]。この手法では、図 1.22 に示すように、三面図から非多様体位相構造を用いたセル分割モデルを作成し、解ソリッド探索をセルの組み合わせを求める方法に帰着させている。セル分割モデル作成の過程では、ワイヤフレームモデルとサーフェスモデルが現れるが、これらが統一的に表現できることの利点についても示されている。本手法については、第 6 章で述べる。

有限要素法

Weiler は、有限要素法解析のためのモデルが非多様体形状モデルを用いて表現できることを述べている [Weiler86]。有限要素法では立体を四面体の集合に分割することが必要であるが、そのような形状は 2-多様体ではないので、非多様体形状モデルが有効であると考えられる。また、薄膜状や線状の立体部分があると非常にいびつな四面体が生成されてしまうという問題がある。それらの部分をサーフェスやワイヤフレームに近似することで解析の効率や精度を向上させることができるが、そのためのモデルとしても非多様体形状モデルが有効であると考えられる。

その他に、Gursoz らによるラピッドプロトタイピングへの応用の提案がある [Gursoz91]。この提案では、ソリッドとサーフェスとの積集合を取ることによってスライス形状を作成し、ラピッドプロトタイピングの入力データを作成するための方法について述べている。

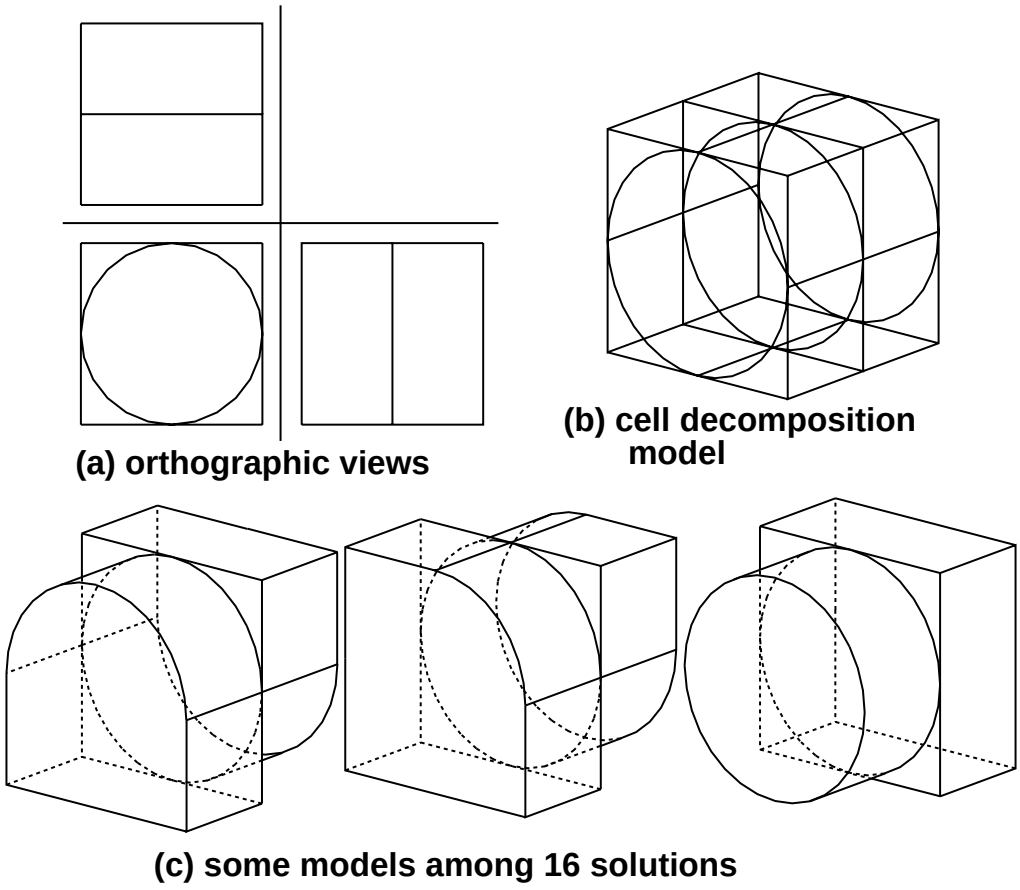


図 1.22: 非多様体位相構造を用いた三面図からのソリッド合成手法 (Masuda,1994)

1.8 まとめ

本章では、以下のことについて述べた。

- ソリッドモデルの表現上の制限について述べ、設計作業全般を支援するためには十分とはいえないことを説明した。
- 設計作業全般を支援するために、非多様体形状モデルが有望であることを述べた。
- 非多様体形状モデリングにおける課題を述べ、(1) 理論的な基礎の確立、(2) 非多様体形状モデリングシステム構築のための基礎技術の確立、(3) 応用システムの構築による有用性の実証が重要であり、その解決法を示すことが本論文の目的であることを述べた。
- これまでになされてきたソリッドモデリング研究について述べることで３次元形状処理の基礎技術について説明を行なった。
- 本研究も含めたこれまでの非多様体形状モデル研究について概観し、本研究の果たしてきた役割について説明した。

Chapter 2

非多様体形状モデルの定義

本章の目的は、以後の本論文における非多様体形状モデルに関する議論の前提条件を明らかにすることにある。そのために、非多様体形状モデルの定義域と集合演算の定義について論じる。

非多様体形状モデルは、直観的には、ワイヤフレーム、サーフェス、ソリッドが混在した形状を扱うと考えることはできる。しかし、形状モデルの厳密な議論を行うためには、どのような制約の元でワイヤフレーム、サーフェス、ソリッドの混在が許されるのか、言い換えれば「非多様体形状モデル」の最低限の条件は何なのかが示されていなければならない。そこで、本論文では、位相幾何学における「複体」に基づいた非多様体形状モデルの定義を採用する。なお、本定義の有用性に関しては、本論文の以後の章において、本定義に基づいた形状処理システムを実装し、工学的応用に適用していくことを通して実証していくものとする。

また、非多様体形状の集合演算についても定義が必要である。従来のソリッドモデルの集合演算は正規化集合演算として定義されてきたが、この演算は立体でない面や稜線を常に排除するため、ワイヤフレームやサーフェスも定義域に含まれる非多様体形状には適していない。そこで、本章では非多様体形状の集合演算について考察し、非多様体形状モデルの定義域に適した集合演算を提案する。

2.1 形状モデルの数学的記述

2.1.1 形状モデルの定義域

非多様体形状モデルは、直観的には、ワイヤフレーム、サーフェス、ソリッドモデルを含む形状モデルと考えることができるが、このような定義は曖昧であり、非多様体形状モデルがどのような性質をもった形状の集まりなのか、ということを規定してはいない。形状モデリングにおいては、「非多様体」という用語がしばしば用いられてきた。しかし、ソリッドモデルの定義域として用いられてきた「正則集合」や「2-多様体」という用語が厳密な数学的裏付けを持っているの対し、形状モデリングで用いる「非多様体」という用語は数学的には厳密な用語ではない。従来、形状処理の分野では、三枚の面に共有される稜線など、ソリッドモデルで表現できない部分は、しばしば非多様体状態 (non-manifold condition) と呼ばれた。非多様体形状モデルという用語はこのような背景から慣用的に用いられているが、実際には多様体でない一部の形状も表現できるという意味に過ぎない。当然のことながら、非多様体形状モデルは多様体でない任意の形状が表現できるわけではないし、それを目指しているわけでもない。

そこで、非多様体形状モデルがどのような条件を満たす形状を扱うのかを明確にする必要がある。このような議論は、非多様体形状モデルの性質を研究していく上で本質的なものである。正しい形状モデルであるために満たすべき条件が厳密に定義できていなければ、形状表現も位相変形操作も場当り的にならざるを得ない。たとえば、ソリッドモデルではオイラー操作 [Baumgart75] と呼ばれる位相変形操作が広く用いられているが、これは2-多様体が持つ位相的性質を利用した操作である。このような操作に関する議論は定義域が明確に規定されていることが前提となる。

2.1.2 正則集合と正規化集合演算

ソリッドモデルにおいては、Requicha の提案した正則集合と正規化集合演算が広く知られている。しかし、これらの定義は非多様体形状モデルには適していない。ここでは、これらの定義について説明し、なぜワイヤフレームやサーフェスを含んだ非多様体形状モデルに適用できないかを示す。

用語の定義

まず、形状モデルの定義域の説明でしばしば用いられる用語である「閉包」「内部」「境界」について、その意味を図 2.1 を用いて説明しておく。

・ Mの **閉包** (closure) とは、点 x を中心とする半径 $e(>0)$ の球 $\rho_e(x)$ が e の値によらず、$M \cap \rho_e(x) \neq \phi$を満たすような x の集合として定義される（図 2.1(a))。Mの閉包を $c(M)$ または $[M]$ と書く。形状モデリングにおいては、境界を持たない部分に境界を付け加える、という意味でこの用語を用いることが多い 。

・ M の **内部** (interior) とは、図 2.1(b) に示すような点集合で、点 x を中心とする半径 $e(>0)$ の球 $\rho_e(x)$ を考えたとき、十分小さい e をとれば $\rho_e(x) \subset M$ となる点 x の集合のことである。M の内部を $i(M)$ と書く。３次元ユークリッド空間において形状 M の内部というときは、立体の境界とはならないぶら下がり面やぶら下がり稜線は除去される。

・ M の **境界** (boundary) とは、M に属し、内部でない点の集合のことで、$b(M)$ と書く (図 2.1c)。

正則集合

Requicha はソリッドモデルのモデリング空間を「有界」「正則」「半解析的」な形状の集合であると定義し、そのような形状の集合を正則集合 (r-set) と呼んだ。半解析的であるとは、図 2.2のように端点で振動して収束しないような曲線・曲面を除外するための条件である。この例では、$x \to 0$ のとき無限に振動するので、$x = 0$ で境界を定義できない。また、正則な形状とは、数学的には次の性質を持つ形状 M として定義される。

$$M = c(i(M))$$

この式において、i は内部、c は閉包を示している。M の内部 $i(M)$ を取ることによって立体でない部分や立体の境界が除去され、さらにその閉包 $c(i(M))$ をとることによって、立体部分とその境界のみが残される。すなわち、この式が成立するということは、形状に立体でない部分が存在しないことを意味している。M がワイヤフレームやサーフェスであれば、$i(M)$ は空集合となるので、正則の条件は満たさない。

図 2.3に正則集合に属する形状の例を示す。正則集合は、図 2.3 (c) (d) のような多様体でない立体も定義域に含むが、ワイヤフレームやサーフェスは含んではならない。

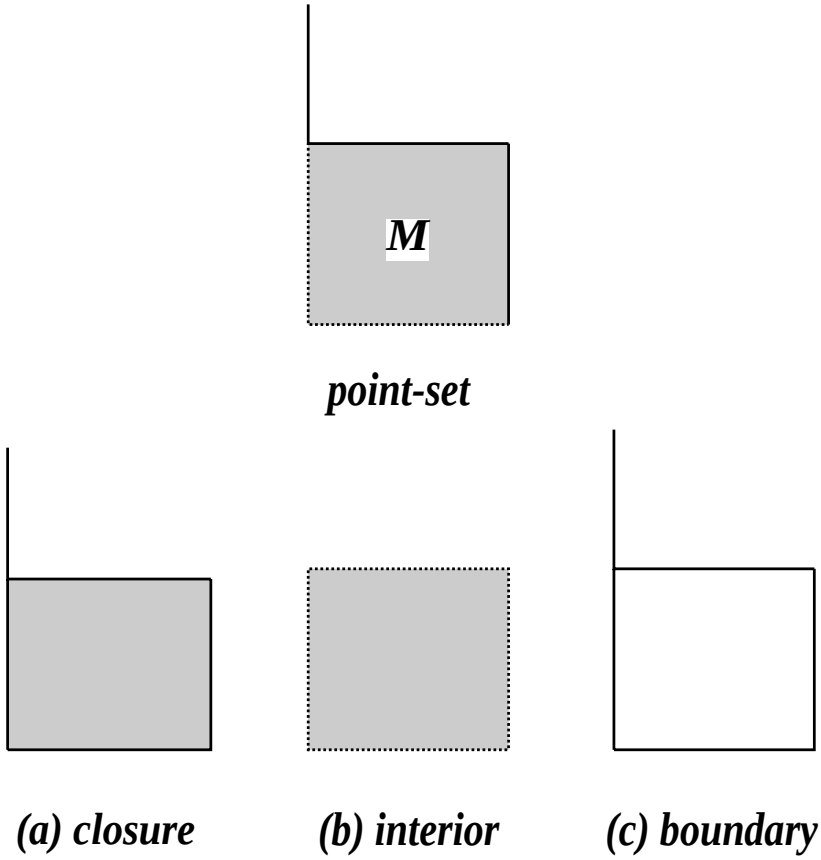


図 2.1: ２次元形状 Mの (a) 閉包、(b) 内部、(c) 境界.

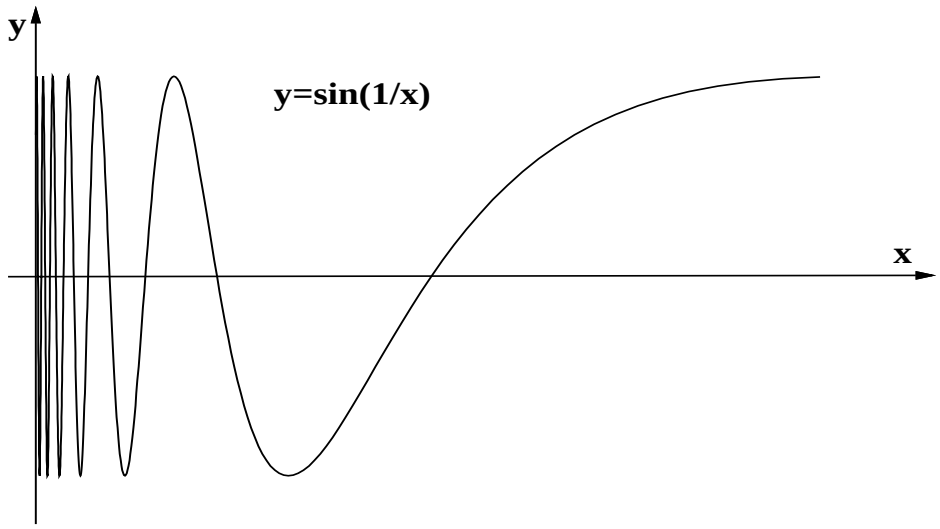


図 2.2: 半解析的でない曲線の例.

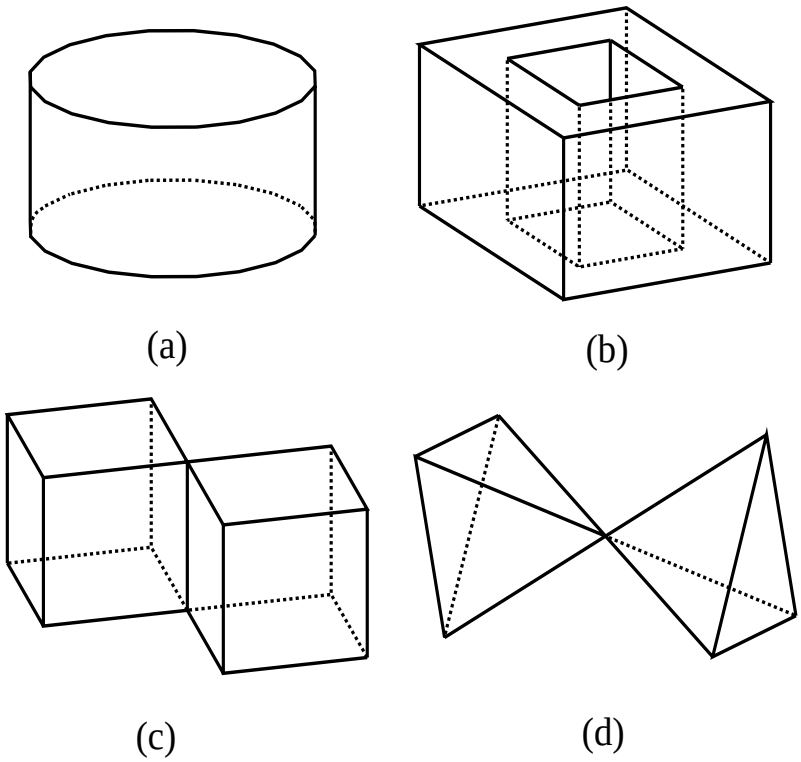


図 2.3: 正則集合.

正規化集合演算

次にソリッドモデルの集合演算の定義について述べる。正則なソリッドモデルに数学で通常用いられている集合演算（$\cup, \cap, -$）を適用するとき、演算結果は必ずしも立体とはならず、定義域から外れてしまうことが知られている。図 2.4 では正則なソリッドモデル A, B の積集合を示している。A と B の共通部分は斜線で示した形状であるが、この部分を積集合としてとると演算結果は正則ではない形状となる。

そこで、Requicha は集合演算の結果が常に正則となるような定義を提案し、その演算を正規化集合演算とした。点集合 A の正規化 $r(A)$ とは、A の閉包、内部をそれぞれ $c(A), i(A)$ として、

$$r(A) = c(i(A))$$

と定義される。正規化集合演算を、和集合、積集合、差集合に応じて $\cup^*, \cap^*, -^*$ と書くとき、次のように記述することができる。

- $A \cup^* B = r(A \cup B)$
- $A \cap^* B = r(A \cap B)$
- $A -^* B = r(A - B)$

このような定義により、図 2.4 の集合演算の結果は空集合となる。

しかし、正規化集合演算は、非多様体形状モデルのための集合演算としては明らかに適していない。たとえば図 2.5(a) のように、サーフェスとソリッドの混在した形状 A と B との正規化和集合演算では、正則ではない部分がすべて除去されるので、図 2.5 (b) のような形状となってしまう。非多様体形状モデルでは、ソリッドやサーフェスとの混在が表現できるという利点があるが、正規化集合演算ではこの利点を損なってしまう。

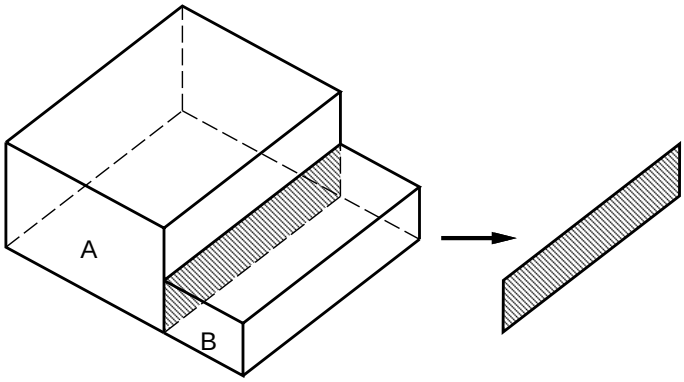


図 2.4: 積集合に縮退した形状が現れる例.

2.2 本研究の方針

ソリッドモデルの数学的記述として用いられてきた正則集合や正規化集合演算は、非多様体形状モデルには適していない。そこで、非多様体形状モデルについて論じるにあたっては、これらに相当するものを定義しておく必要がある。

定義域

非多様体形状モデルを定義する際には、いたずらに定義域を広くするのではなく、数学的な性質の優れた形状の集合が望ましい。形状モデルの数学的背景である位相幾何学は長い歴史を持っており、代表的な幾何形状については非常に詳細な研究がなされてきた。したがって、非多様体形状モデルを数学の分野で十分研究された幾何形状に準じて定義することができれば、数学的扱いが容易になり、幾何形状の性質などに関する様々な数学の定理が利用できることになる。

本研究では、このような観点から、数学の分野で一般に用いられてきた「複体」(complex) に準じて非多様体形状モデルを定義する。複体とは n 次元多様体が混在した幾何形状を扱うために数学者 Poincare (1854-1912) が導入した概念で、ワイヤフレーム、サーフェス、ソリッドの混在するような幾何形状を記述する概念として適していると考えられる。このような定義域を採用する利点としては、以下のようなことが挙げられる。

- 位相的性質がよくわかっている。
- ほとんどの非多様体形状モデル研究 [Weiler86, Masuda88a, Gursoz88, Kobayashi89,

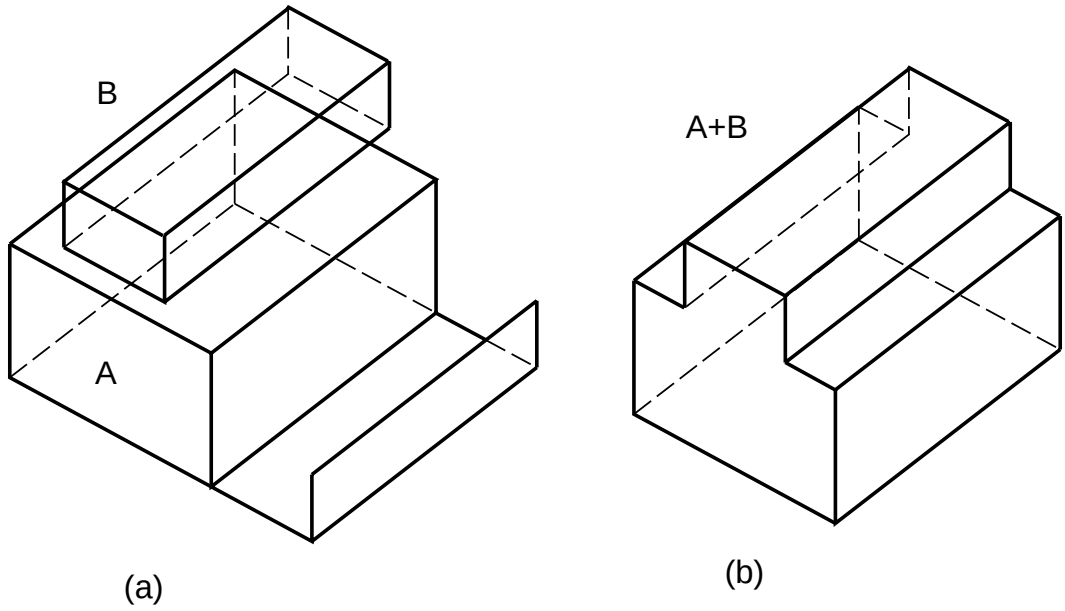


図 2.5: 正則でない形状A, Bの正規化和集合.

Yamaguchi90, Crocker91] で表現対象としている幾何形状を、数学的な言葉で記述したものとなっている。

- 胞体と呼ばれる構成要素の集合で形状が記述されるので、境界表現に適している。

また本章で述べる定義は、実際に非多様体形状モデリングシステムを構築し、様々な応用に適用していく上でも非常に有用なものである。それについては本論文の３章以降で、本章の定義に基づいた形状操作や形状表現、またその応用などの研究成果を通じて実証していくものとする。

なお、非多様体形状に関して異なる定義を用いた研究もあることを述べておく。Rossignac と Requicha の研究 [Rossignac91] では、複体の部分集合を考えることにより、境界を持たない開いた形状も定義域に含めた。しかし、このような定義域は位相的な性質が素直ではないために形状操作が極めて複雑なものとなり、また境界を持たない形状を定義域に含めることによりどのような利点が生じるのかも明らかではない。また、相澤 [Aizawa89] の研究のように、胞複体鎖に基づいて形状モデルを定義した研究もある。この研究は代数的手法によって形状処理を記述することを目的としており、一般の形状モデリング研究とは大きく異なったものであるのでここでは論じない。また、形状モデルの定義域として一般次元の形状を扱った研究として [Rossignac90, Lienhardt89, Brisson90, Paoluzzi93] などがある。ただし、４次元以上の高次元形状においては、形状設計における利点は

明らかではない。

集合演算

ソリッドモデルで用いられてきた正規化集合演算は、集合演算の結果が必ず正則なソリッドとなるので、ワイヤフレームやサーフェスの集合演算には適用できない。したがって、非多様体形状モデルにおいても適用できる集合演算を考えることが必要である。[Rossignac91] のように、境界のない開いた形状まで含めた定義域を採用するのであれば、数学で用いられる集合演算の定義をそのまま流用できるが、本論文を含めた多くの非多様体形状モデル研究では閉じた幾何形状のみを扱っており、数学で用いられる集合演算をそのまま用いることはできない。

また、たとえ定義域に収まっていても、集合演算を形状生成の手段として考えると、不用意に縮退した面や稜線を生成することは望ましいことではない。たとえば、図 2.4 のような境界を接するような２つの立体 A, Bの積集合を考えるとき、非多様体モデルではこのような正則でない形状も矛盾なく扱うことができるが、立体形状の作成手段としては縮退した部分は多くの場合不要である。したがって、図 2.4 のような場合、縮退した面ではなく空集合を演算結果とした方が望ましいことが考えられる。

以上のような観点から、次に示すような条件を満たす集合演算が、非多様体形状モデルの集合演算として望ましいものであると考える。

1. 任意の形状モデル間で集合演算（和集合、差集合、積集合）が定義できる。
2. 集合演算で作られた形状は、形状モデルの定義域におさまる。
3. 任意の形状モデルは、他の形状モデル間の集合演算結果として得ることができる。
4. 正則なソリッドモデル間の集合演算では、正則集合演算と一致する結果を得ることができる。

４節では、これらの条件を満たした集合演算の定義を提案する。

2.3 非多様体形状モデルの定義域

本節では、まず複体について説明した後、本研究で用いる非多様体形状モデルの定義について述べる。

2.3.1 用語の定義

複体とは、異なる次元の多様体を組み合わさせて生成される幾何形状を扱うためにPoincare が導入した概念である。ここでの複体に関する説明は、位相幾何学の参考書である [数学 1, 数学 3, 数学 5, 数学 4] に基づいている。なお、ここで述べる複体とは、ユークリッド空間の部分集合で、構成要素が有限個であるもののみを考える。

胞体

３次元ユークリッド空間においては、複体は、図 2.6 に示したような０胞体、１胞体、２胞体、３胞体を組み合わせることによって構成される。胞体はいずれも境界を含まず、また穴や空洞があってはならない。厳密には胞体は次のように定義される。
「ユークリッド空間の部分空間 e に対し、連続写像 $\varphi : V^n \longrightarrow [e]$ で $\varphi(S^{n-1}) \subset e^{\cdot}$ となり、かつ $\varphi : V^n - S^{n-1} \longrightarrow e$ となる φ がとれるならば e を n 次元胞体 (n-cell) または開胞体 (open cell) と呼ぶ。」
ここで、V^n, S^{n-1} をそれぞれ n 次元単位球体、$n-1$ 次元単位球面、また $[e]$ を e の閉包、$e^{\cdot}$ を $[e]-e$ とする。この定義では、$e^{\cdot}$ に関する条件も規定しており、端点で無限に振動するような半解析的でない曲線は、端点で集積点を持たないので胞体の定義から除外される。

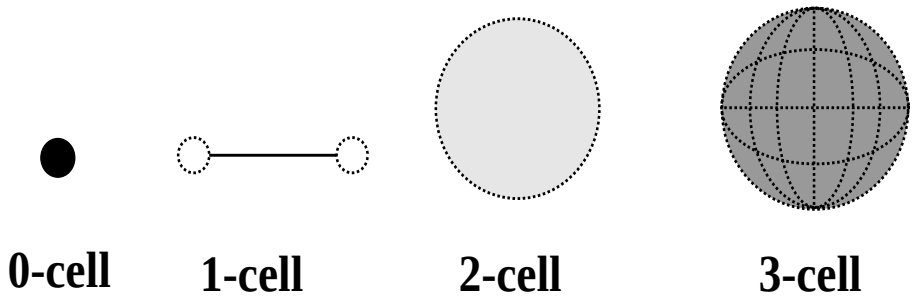


図 2.6: ０胞体、１胞体、２胞体、３胞体.

複体

複体とはこれら胞体が、互いに交わらず、次元の低い胞体が次元の高い胞体の境界上に存在するように組み合わされたものである。数学的には、ユークリッド空間の部分空間 X が次に示す条件を満たすとき、X が複体であると定義される。ここで、Λ は胞体のための添字集合、X^n は次元が n 以下の e_μ $(\mu \in \Lambda)$ 全体の集合である。

1. $X = \bigcup_{\lambda \in \Lambda} e_\lambda$
2. $e_\lambda \bigcap e_\mu = \phi$ $(\lambda \neq \mu)$
3. e_λ の次元が $n+1$ ならば　$[e_\lambda] - e_\lambda \subset X^n$

図 2.7は、これらの条件を具体的に説明するために示したもので、複体に反する形状の例である。形状 (a) は面に空洞があり、２胞体ではないので条件１に反する。形状 (b) は１胞体と２胞体が交点を持つので、条件２に反する。また、形状 (c) は 端点が存在しないので、条件３に反している。

一方、図 2.8は複体の例である。

2.3.2 非多様体形状モデルの定義

点集合としての定義域

本論文においては、非多様体形状モデルの定義域を、複体がユークリッド空間に占める点集合で表される幾何形状と定義する。すなわち、幾何形状 X が複体の条件を満たす胞体の集合に分割できるとき、X を非多様体形状モデルの表現対象と考える。

境界表現非多様体形状モデルの定義

次に、非多様体形状モデルを境界表現で表す場合の位相要素の定義と、位相要素の満たすべき条件を複体に準じて定めることにする。複体は胞体の集合によって定義されているので、2-多様体や正則集合が単に点集合として定義されていたのと異なり、幾何形状を境界表現で表すときの位相構造についてもある程度規定していると考えることができる。

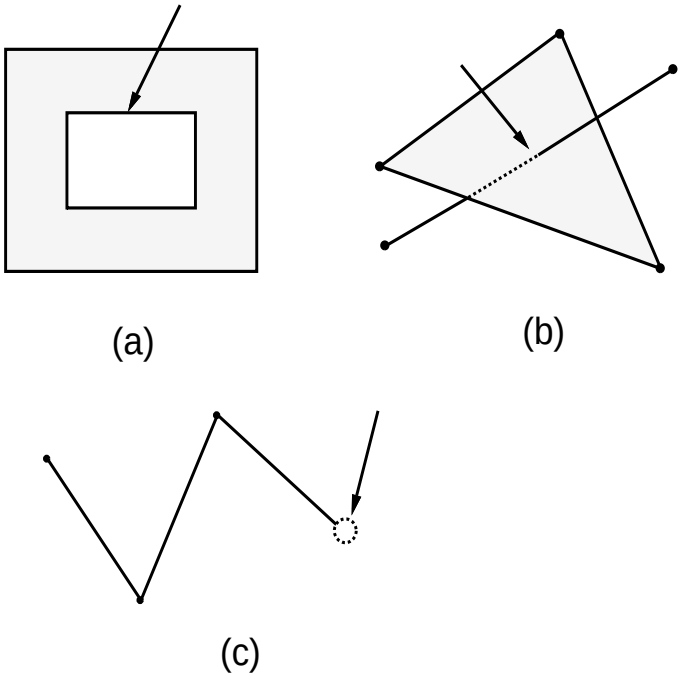


図 2.7: 複体でない形状.

３次元複体は、０胞体、１胞体、２胞体、３胞体の集合で構成される。境界表現形状モデルも同様に、頂点、稜線、面といった位相要素の集合で表現されてきた。ここでは、境界表現形状モデルの位相要素を vertex, edge, face, volume と呼ぶことにする。volume とは、面に囲まれた立体部分に対応する位相要素である。

従来の境界表現形状モデルでは空洞や貫通穴を持った位相要素を認めているため、位相要素の定義は胞体とは異なっている。すなわち、図 2.9 のような形状を表現するとき、図 2.8 (d), (e), (f) に示した形状のように空洞や貫通穴が生じないように分割して表現することは境界表現形状モデルでは一般的ではない。そこで、複体が胞体の集合として定義されているのに準じて、非多様体形状モデルを以下のような位相要素の集合で構成されているものとする。図 2.10 に位相要素の図を示す。

vertex ０胞体。E_3 の一点

edge １胞体。両端点を含まない有界で連結な線分。

face 境界を含まない有界で連結な面。すべての点の近傍が２次元単位開球と同相。有限個の穴があってもよい。向き付け可能（裏と表の区別があること）で、境界は空集合でない。すなわち、メビウスの輪や球面は除外される。

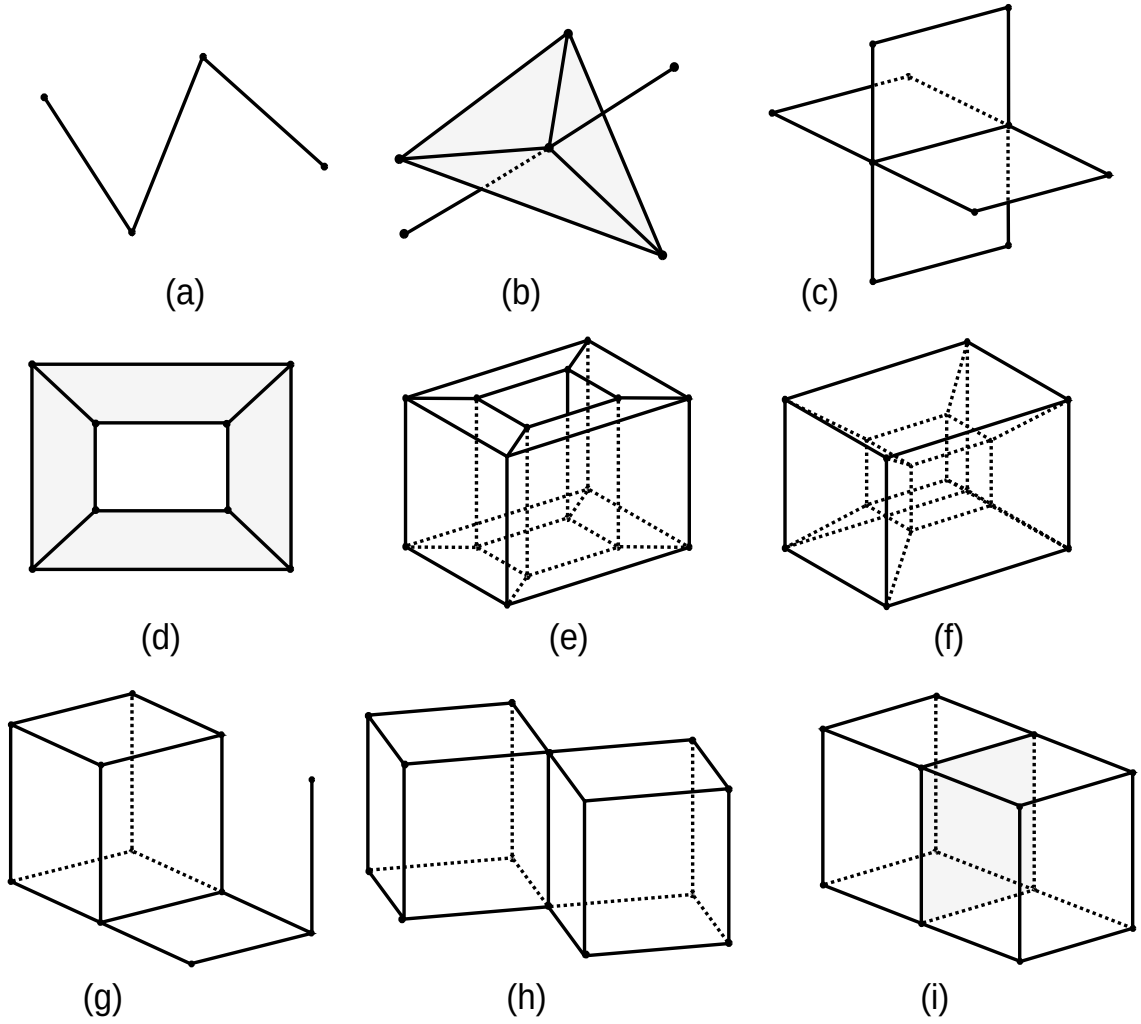
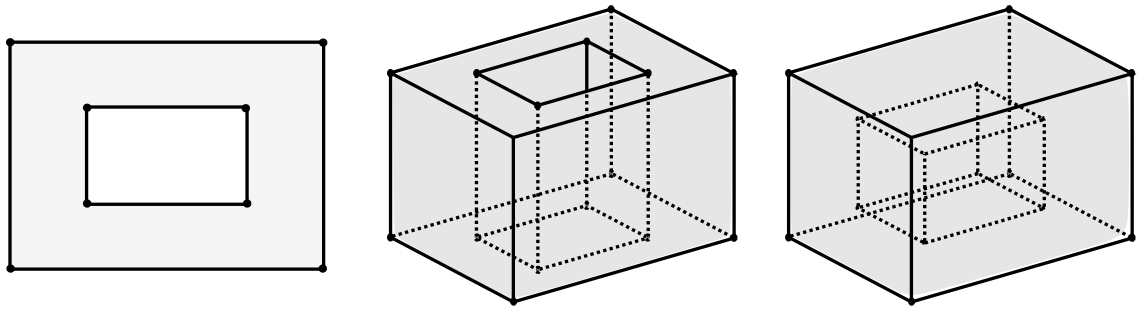


図 2.8: 複体の例.

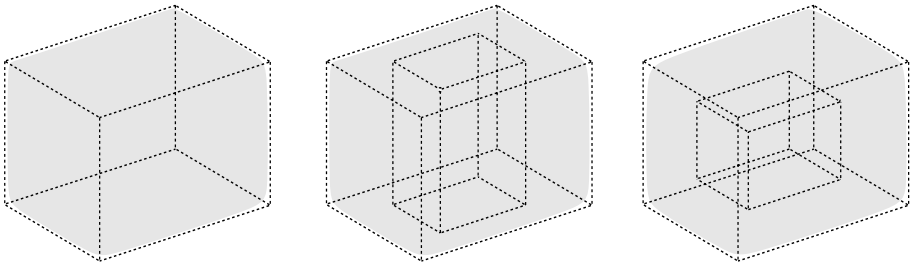


(a) face with a cavity **(b) volume with a through-hole** **(c) volume with a cavity**

図 2.9: 空洞と貫通穴を持つ形状.

(a) vertex **(b) edge**

(c) face



(d) volume

図 2.10: 位相要素.

volume 境界を含まない有界で連結な３次元領域。すべての点の近傍が３次元単位開球と同相。有限個の貫通穴や空洞があってもよい。

本論文では、境界表現非多様体形状モデル M を次の条件を満たす位相要素の集合 $\{\epsilon_\lambda |\ \lambda \in \Lambda\}$ で表現されるものと定義する。ここで、M^n は次元が n 以下の$\epsilon_\mu\ (\mu \in \Lambda)$ 全体の和集合である。

1. $M = \bigcup_{\lambda \in \Lambda} \epsilon_\lambda$
2. $\epsilon_\lambda \bigcap \epsilon_\mu = \phi\ (\lambda \neq \mu)$
3. ϵ_λ の次元が $n+1$ ならば　$[\epsilon_\lambda] - \epsilon_\lambda \subset M^n$

これらの条件は、これまでの境界表現でも用いられてきた制約であり、自然に受け入れることのできるものである。また、本定義に基づいて境界表現形状モデルを構成すれば、face, volume を適当に分割することで複体に帰着させることができるので、数学的扱いも容易である。

2.4 集合演算の定義

本研究では、正規化集合演算に代わる非多様体形状モデルの集合演算として、図 2.11 に示されるような集合演算を用いる。この図では、左側に二つの形状A、Bをそれぞれ実線、破線で示しており、右側に演算結果を示している。ここでは、定義される集合演算の和、差、積をそれぞれ $\oplus, \ominus, \otimes$ と書き、数学で用いられる集合演算 $\cup, \cap, -$ と区別するものとする。これらの集合演算は、

- 任意の形状モデル間で集合演算（和集合、差集合、積集合）が定義できる、
- 集合演算で作られた形状は、形状モデルの定義域におさまる、
- 任意の形状モデルは、他の形状モデル間の集合演算結果として得ることができる、
- 正則なソリッドモデル間の集合演算では、正則集合演算と一致する結果を得ることができる、

という条件を満たしている。

以下に、それぞれの定義について述べる。

2.4.1 和集合

非多様体形状モデルA、Bの占める点集合の和を和集合の定義とした場合、非多様体形状モデルの定義域に関して閉じている。また、正則な形状モデル間の和集合の結果は正規化集合演算と一致する。したがって、非多様体形状モデルの和集合を以下のように定義することができる。

$$A \oplus B \equiv A \cup B$$

図 2.11(a) に、和集合の例を示す。

2.4.2 差集合

点集合間の差集合 $-$ の場合は、境界を持たない部分が生じるために、本論文の非多様体形状モデルの定義域に納まらない。図 2.12 に例では、立体AとBとの差集合は図 2.12(b) の破線で示した部分において境界が存在しない。

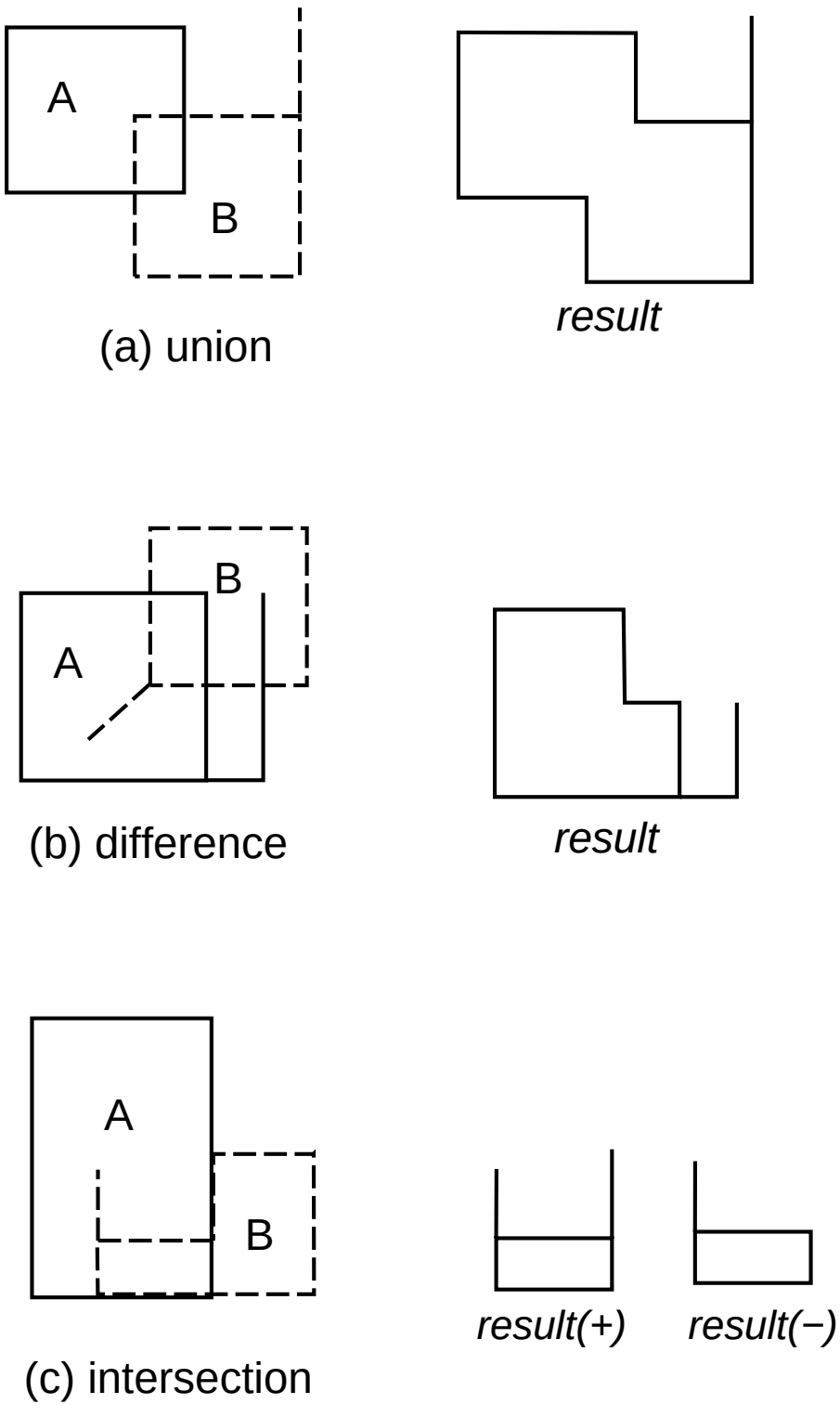


図 2.11: 非多様体形状モデルの集合演算の例.

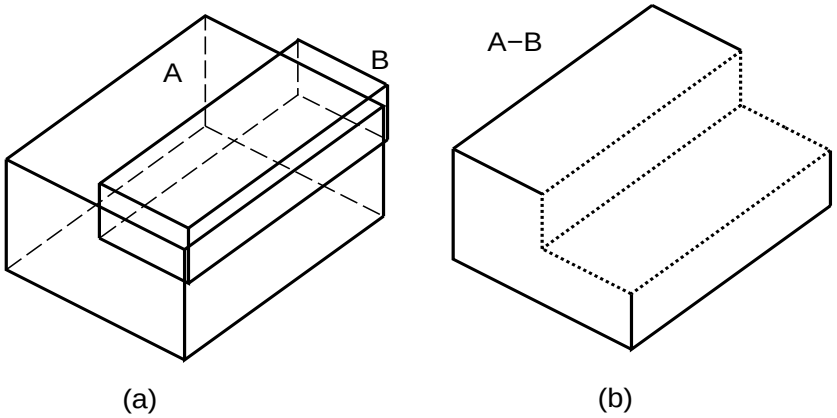


図 2.12: 差集合が開集合となる例.

そこで、差集合が定義域に納まるように、図 2.12(b) に集積点を加える演算を差集合演算 $\ominus$ と定義する。この演算は、正則なソリッド間の差集合の場合は正規化集合演算の結果と一致する。非多様体形状モデルの差集合は $A - B$ の閉包として以下のようになる。

$$A \ominus B \equiv c(A \cup B)$$

2.4.3 積集合

数学で用いられる積集合で生成される形状は、非多様体形状モデルの定義域に納まっている。したがって、点集合間の共通部分を非多様体形状モデルの積集合と考えることは可能である。

しかし、設計作業の支援という観点から考えると必ずしも適当な定義ではないように思われる。たとえば、図 2.4 のように、境界を接するような 2 つの立体 A, B の積集合を考えると、演算結果は 2 次元の面となる。非多様体モデルではこのような正則でない形状も矛盾なく扱うことができるが、立体形状の作成手段として集合演算を用いる場合には縮退した部分は多くの場合不要である。

ただし、形状モデリングの一般のアプリケーションを考えた場合は、縮退した部分が重要となることも考えられる。たとえば、ソリッド間の干渉チェックを積集合を用いて求める場合には縮退した面、線、点には意味があり、それぞれ面接触、線接触、点接触を表わしている.

従って、積集合の縮退部分を残しても除去しても積集合の定義としては矛盾しないこ

とから、本論文では、積集合には２つのバージョンがあるものとし、縮退した部分も残す積集合を $\otimes^+$、除去する演算を $\otimes^-$ と書くことにする。図 2.11(c) には縮体した部分を残す演算と除去する演算の例が示されている。なお、$\otimes^-$ で除去するのは縮退した部分だけなので、この例から明らかなように、常に正規化集合演算と一致するとは限らない。

数学的には、縮退した部分を削除する積集合は次のように定義することができる。いま、非多様体形状モデルの占める点集合を A, 点 p を中心とする十分小さい半径 ϵ の３次元開球 $\{x \mid |x-p| < \epsilon\}$ を $\rho_\epsilon(p)$、また、$p \in A$ で $A \cap \rho_\epsilon(p)$ が n 次元開球と同相となるような点 p の集合を $i^n(A)$ と書き、

$$I(A) = i^1(A) \cup i^2(A) \cup i^3(A)$$

とする。このとき、積集合を $c(I(A) \cap I(B))$ とすれば、縮退部分を含まない演算結果が得られる。正則なソリッド R をこの演算に当てはめた場合は、$i^1(R) = \phi, i^2(R) = \phi$ なので演算結果は正則となり、正規化集合演算の結果と一致する。

以上から、積集合演算は以下のような定義となる。

$$A \otimes^- B \equiv c(I(A) \cap I(B)) \quad \text{（縮退した部分を残さない）}$$

$$A \otimes^+ B \equiv A \cap B \quad \text{（縮退した部分を残す）}$$

2.5 まとめ

本章では、以下のことについて述べた。

1. 本論文で扱う非多様体形状モデルの定義を示した。

   非多様体形状モデルには広く一般的に認められた定義がないため、非多様体形状モデルについて論ずるためには、どのような定義域の幾何形状を扱うのかを明確にする必要がある。本論文では、非多様体形状モデルで表現対象とする幾何形状を複体に準じて定義する。

2. 非多様体形状モデルのための集合演算について定義した。

   非多様体形状モデルには従来の正規化集合演算はなじまないため、集合演算の定義を示す必要があった。そこで本研究では、非多様体形状モデルに適した集合演算の定義を行った。本定義に基づいて集合演算を行った場合、演算結果は非多様体形状モデルの定義域に納まっており、さらに、正則なソリッド モデル間の演算では正則集合演算の結果と一致する。

Chapter 3

非多様体形状モデルのオイラー操作

オイラー操作はソリッドモデルの位相データを変更する操作として広く用いられてきたが、非多様体形状モデルに適用できるオイラー操作は知られていなかった。本章では、非多様体形状モデルにおいて、ソリッドモデルのオイラー操作と同等の性質を継承した基本操作群を導出するための数学的な手法を提案する。本研究は、非多様体形状モデルのオイラー操作を導出した初めての研究 [Masuda88a, Masuda89b] として知られている。

本論文では複体のオイラー・ポアンカレの式を利用して非多様体形状モデルで成立する位相的な関係式を導出し、その式に基づいてオイラー操作を定義する方法を示す。そして、任意の非多様体形状モデルは、理論的には９種類のオイラー操作を組み合わせることで生成できることを証明する。

また、本章の数学的道具を用いて、様々な定義域の形状モデルでのオイラー操作についても論じる。さらに、オイラー操作を設計する方法についても論じる。

3.1 形状モデルの位相操作

本節では、形状モデルの位相変形操作であるオイラー操作について説明し、オイラー操作が形状処理において非常に重要な操作であることを述べる。

3.1.1 オイラー操作

境界表現形状モデルでは、face, edge, vertex といった位相要素の集まりによって形状を表現する。これらの位相要素間の関係はグラフ構造で表現されているが、形状の位相構造を変更する際には、このグラフ構造を矛盾なく変更することが必要である。しかしながら、境界表現の場合、グラフ構造は大変複雑であり、形状として矛盾を生じないように位相構造を変更していくことは難しい作業である。そこで、ソリッドモデリングでは、位相要素間の整合性を保持する基本操作としてオイラー操作がよく用いられている。

オイラー操作とは、Baumgart がソリッドモデルの位相変形を行なうために 1975 年に提案した操作群であり、ソリッドモデルの最も基本的な位相操作である [Baumgart75] 。この操作群は、2- 多様体ソリッドが位相的に満たす必要条件であるオイラー式を壊さない原始的な操作によって構成される。Baumgart は当初、貫通穴や空洞を含まないソリッドモデルを対象としたが、Braid らが貫通穴や空洞を含む場合に拡張し [Braid80]、次の式がソリッドモデルのためのオイラー式、またはオイラー・ポアンカレの式として一般に用いられている。

$$\begin{aligned}&[\text{頂点の個数}] - [\text{稜線の個数}] + [\text{面の個数}] - [\text{面の穴の個数}] \\ &\quad = 2\ ([\text{連結成分の個数}] - [\text{貫通穴の個数}])\end{aligned}$$

一般に、ソリッドモデルのオイラー操作とは、この式の関係を維持し、各変数の増減が 1 であるような原始的な操作をいう [Baumgart75, Braid80]。

ソリッドモデルのオイラー操作の基本的な性質として、

「すべての 2-多様体ソリッドモデルの位相構造は、いくつかの原始的な操作を組み合せることによって生成できる」

という定理が知られている [Braid80, Mantyla82]。したがって、十分なオイラー操作を用意しておけば任意の位相変形操作はオイラー操作を組み合せることによって記述できるので、形状処理システムの実装に大変都合がよい。

オイラー操作の利点は次のようにまとめることができる [Braid80, Mantyla82, Chiyokura83a]。

- オイラー操作は位相要素の個数に関して整合性を保持するので、それらに基づいて実装された形状変形操作でも位相要素の個数の整合性が保てる。
- オイラー操作は実際のデータ構造を隠蔽するので、アルゴリズムが特定のデータ構造に依存する度合が少なくなる。
- オイラー操作の組合せで形状処理を実装することによりシステムのモジュラリティが高まり、プログラムの信頼性の向上や保守の容易さが期待できる。
- オイラー操作は一対一に対応する逆操作が存在するため、形状変形後に位相データを元の状態に復元する逆操作が容易に実現できる。

3.1.2 非多様体形状モデルの位相操作

オイラー操作にはこれらの優れた性質があり、境界表現形状モデリングシステムの実装には必須のものとして利用されてきた。しかし、従来のオイラー式は、開いた shell や非多様体形状では成立しないことが知られている。そのため、非多様体形状モデルが研究されるようになると、これまで形状モデリングの基本的な道具として利用してきたオイラー操作が使えないという深刻な問題が生じるようになった。

非多様体形状モデルの基本操作としては 1986 年に NMT オペレータが提案されている [Weiler86c]。しかし、これらの操作群はオイラー操作のような数学的な背景を持たず、拘束式に基づいた操作ではない。そのため、ここで述べたようなオイラー操作の利点は継承していず、また操作群の選択も場当たり的で基本操作群としては不十分であった。任意の位相操作を実現するためにはどのような NMT オペレータが必要か、また NMT オペレータが位相の整合性をどれだけ保証できるのか、逆操作の存在は保証されるのかなどに関して十分な答えを提示できていなかった。

オイラー操作の利点の一つは複雑なデータ構造を隠蔽できることであるが、非多様体形状モデルのデータ構造はソリッドモデルよりもさらに複雑であるので、データ構造が隠蔽できる基本操作群の必要度はソリッドモデルよりも高い。また、これまで、逆操作などのオイラー操作の利点を利用した様々なアルゴリズムが開発されてきたが、それらが利用できなくなることはモデリングシステムを構築する上で大きな制約となる。

3.2 オイラー操作に関する従来の研究

本節では、これまで提案されてきたオイラー操作に関する研究を概観することにより、オイラー操作の基礎となるオイラー式について説明する。また、オイラー操作ではないが、非多様体形状モデルの位相変形操作として提案された NMT オペレータについても説明しておく。

3.2.1 ソリッド モデルのオイラー操作

Baumgart のオイラー式

Baumgart によって最初に示された オイラー操作は、多面体の頂点の数（v)、稜線の数（e)、面の数（f）の間に関係式：

$$v - e + f = 2 \tag{3.1}$$

が成立することを利用する [Baumgart75]。この式はオイラー式と呼ばれ、多面体の満たすべき必要条件である。オイラー操作とは、この関係式を壊さない基本操作である。図 3.1 にオイラー操作の例を示す。ここでは、稜線 e を e_1 と e_2 に分割し、さらに面 f を f_1, f_2 に分割している。それぞれの操作がオイラー操作に相当する。この過程ではそれぞれ、式 3.1 の変数 v と e が 1 ずつ、及び、変数 e と f が 1 ずつ増加するが、

$$(v+1)-(e+1)+f = v-(e+1)+(f+1) = v-e+f$$

となるので、操作適用前の形状がオイラーの式を満たしているならば適用後も必ず式を満たす。このように、オイラー操作によって生成されたデータ構造では、位相要素の個数は常にオイラー式を満たしているので、位相構造の整合性を容易に保持することができる。

Braid のオイラー式

式 3.1 は、face の空洞や、貫通穴を含まない立体に対する関係式であるが、この式は Braid らによって、貫通穴を含んだソリッド モデルで成立するように拡張された [Braid80]。この場合、頂点の数（v)、稜線の数（e)、面の数（f)、面の空洞の個数（r)，貫通穴の個数（h)，連結成分の個数（s）の間に次の関係式が成立する。

$$v - e + f - r = 2\,(s - h) \tag{3.2}$$

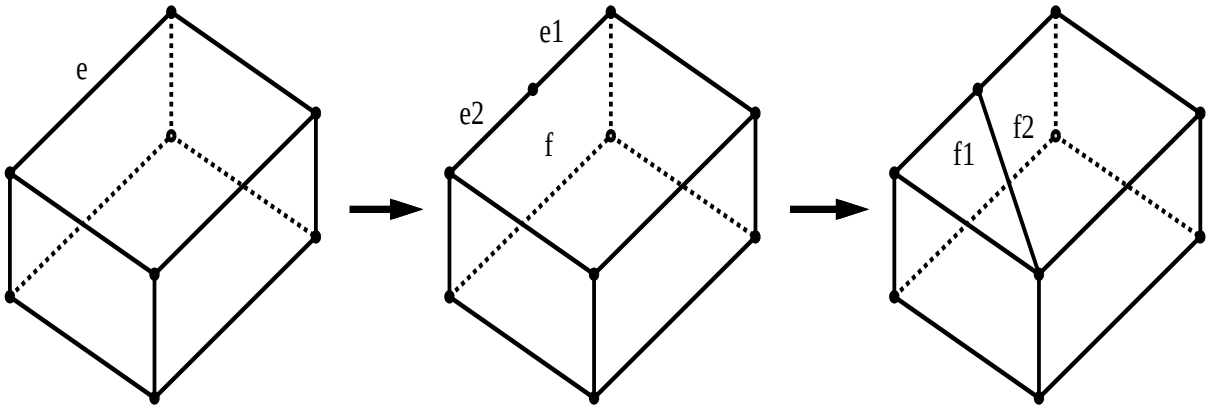
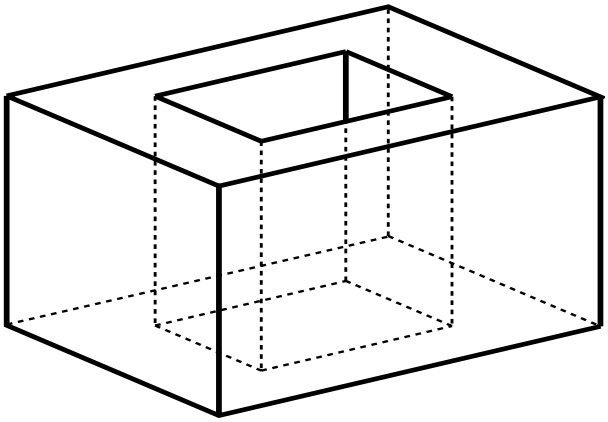


図 3.1: オイラー操作の例.



v=16, e=24, f = 10, r = 2,
s = 1, h = 1

v−e+f−r=2(s−h)=0

図 3.2: オイラー式：$v - e + f - r = 2\ (s - h)$ の適用例.

図 3.2 に式 3.2の適用例を示す。

また、Braid はオイラー式とベクトル空間との類似性に着目して、任意の 2 - 多様体ソリッドは、５種類のオイラー操作の組み合せて生成できることを示した [Braid80]。このことは以下のように説明できる。

式 3.2 は６個の変数を持つので、これらを主軸とする６次元空間を考えることができる。この場合、式 3.2 は６次元空間における超平面 E を表しており、任意のソリッドモデルは、超平面上の点 $P(v,e,f,r,s,h)$ に対応する。Eは自由度が５であるので、E 上の任意の点は独立な５個の基底ベクトルの線型結合で表現することができる。ここで、基底ベクトルをオイラー操作とみなせば、独立なオイラー操作は５種類であることがわかる。

オイラー操作の応用システム

オイラー操作に基づいた形状モデリングシステムの研究としては、GWB [Mantyla82] や MODIF [Chiyokura83a] が知られている。GWB では、式 3.2 に基づいたオイラー操作やその逆操作によって形状変形操作を実現している。また、千代倉はオイラー操作に一対一に対応する逆操作が存在する性質に着目して、変更した形状を以前の状態に戻す undo 操作が行なえるソリッドモデラ MODIF を開発した。すべての形状変形操作をオイラー操作の組合せによって記述し、オイラー操作や幾何データ変更の履歴を操作ツリーとして保持することによって、形状生成のどの段階にでも後戻りすることが可能となる。

非多様体位相への適用例

しかしながら、このようなオイラー操作は非多様体形状モデルで用いることはできない。なぜなら、非多様体形状ではオイラー式 3.1 、3.2 は成立しないからである。図 3.3 に式 3.2 が成立しない例を示す。立体ではない例として図 3.3(a) を考えてみると、vertex は６個、edge は７個、face が２個、shell が１個であるが、

$$v - e + f - r = 1$$

$$2\ (s - h) = 2$$

となるので、式 3.2は成立しないことがわかる。また、立体であっても、図 3.3 (b) のように４枚の face に共有される非多様体稜線を含む形状では、vertex が １４個、edge

が２３個、face が１２個、shell が１個となり、

$$v - e + f - r = 3$$

$$2\ (s - h) = 2$$

となるので、式 3.2が成立しないことがわかる。

3.2.2 ワイヤフレームと紙モデルのオイラー操作

また、２-多様体以外の形状に対しては、Wilson が紙モデルとワイヤフレームモデルのそれぞれについてオイラー操作を示した [Wilson85]。

紙モデルのオイラー式

紙モデルとは、一枚の紙からハサミを使って作ることのできる平面形状を (1) 他の紙とは端でのみ接続し、(2) ３枚以上の面は同一稜線で接続しない、という条件で生成できる形状であり、位相的には、境界のある２-多様体と、閉じた殻となる境界のない２-多様体の２種類が生成される。紙モデルが閉じた殻であるときは既に述べた２-多様体ソリッドモデルと見倣せるので式 3.2 が成立する。一方、境界のある２-多様体のときには、頂点の数（v)、稜線の数（e)、面の数（f)、面の穴の数（r)、形状モデルの穴の個数（h)、連結している位相要素の個数（s）の間に、

$$v - e + f - r = s - h \tag{3.3}$$

が成立する。したがって、紙モデルが閉じているかどうかによって式 3.2 と 3.3 を使い分けることになる。図 3.4(a) は、開いた紙モデルで式 3.3 が成立していることを示している。

ただし、この研究では、２-多様体ソリッドと開いた shell のそれぞれについてオイラー操作を示しているが、非多様体稜線など、多様体でない形状は定義域に含まれていない。また、境界のない２-多様体と境界のある２-多様体の場合でオイラー式を使い分けなければならないため、両者にまたがる位相操作は定義できないという問題がある。

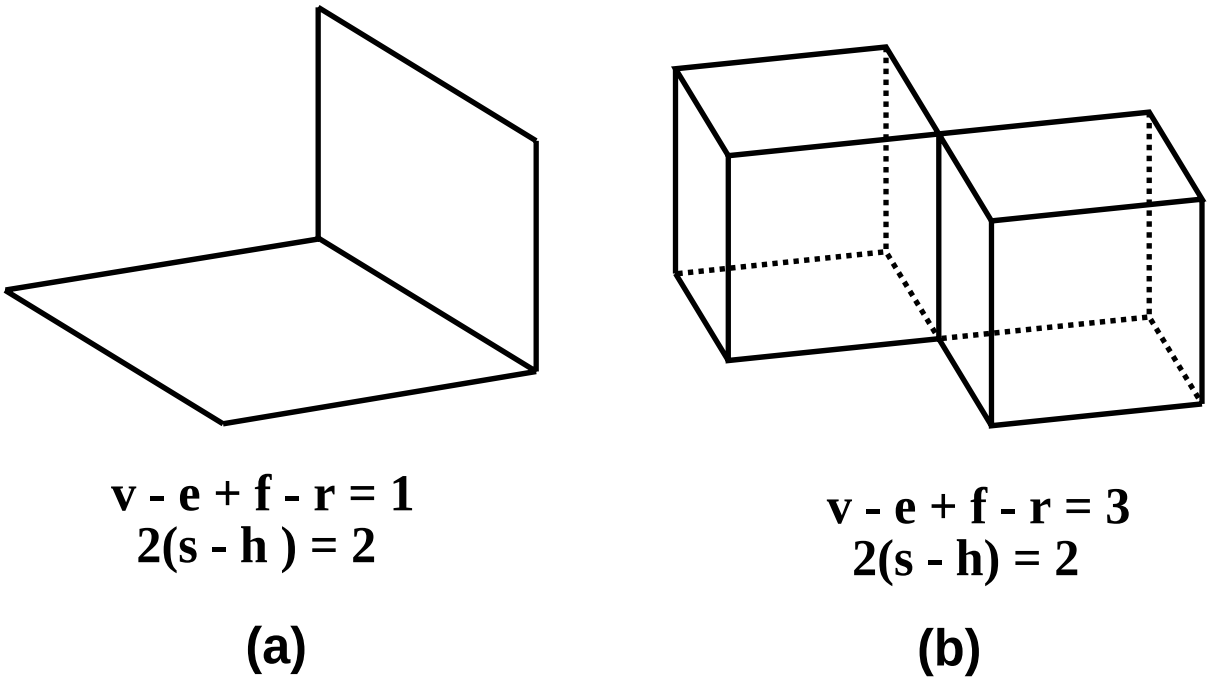


図 3.3: オイラー式：$v - e + f - r = 2\ (s - h)$ の成立しない形状.

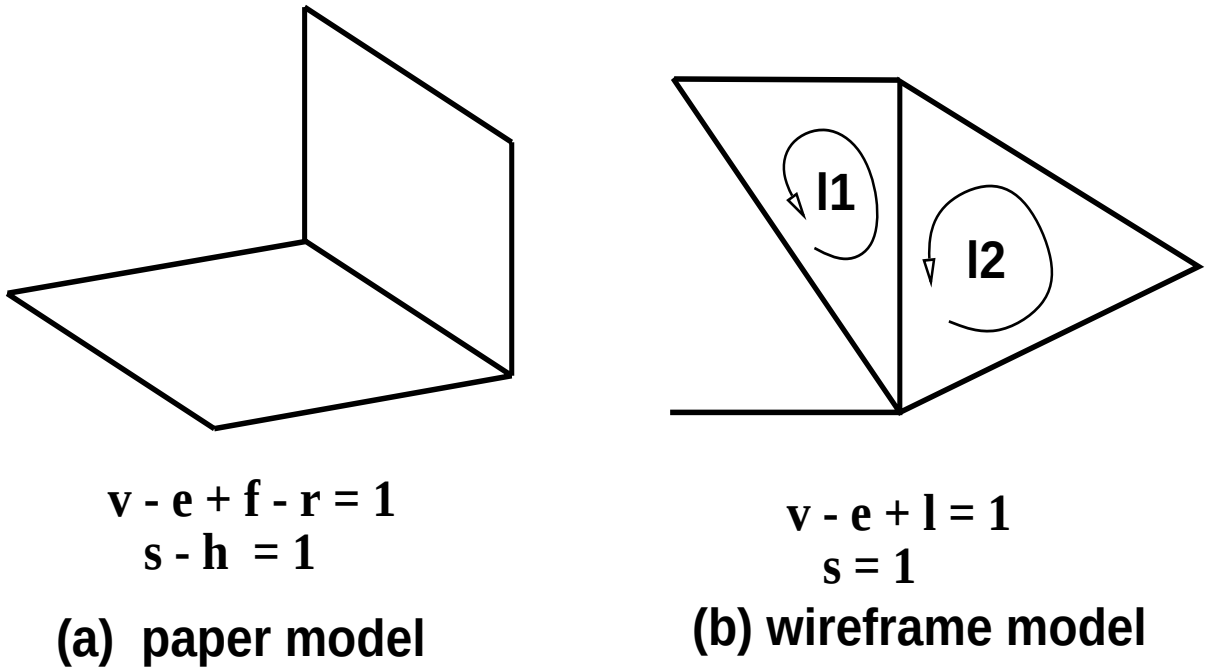


図 3.4: 紙モデルとワイヤフレームモデルのオイラー式の適用例.

ワイヤフレームモデルのオイラー操作

また、Wilson はワイヤフレームモデルを、vertex を岐点、edge を枝とするグラフと考え、独立なループの個数を l としたときのオイラー式を次のように求めた。

$$v - e + l = s \tag{3.4}$$

図 3.4 (b) の例では、独立なループは l_1 と l_2 の二つなので、$l = 2$ となり、式 3.4 を満たしている。この式は、頂点が 3 本以上の稜線に共有されても成立するため、1 次元の非多様体形状に関するオイラー式とみなすことができる。(これは、本章３節で述べるように、式 3.4 における独立成分の個数 s と独立なループの個数 l が１次元非多様体形状の位相的な特性を記述するのに十分な位相不変量であることによる。)

3.2.3 NMT オペレータ

オイラー操作ではないが、ワイヤフレーム、サーフェス、ソリッドが混在した非多様体形状を対象とした位相操作としては、Weiler の提案したＮＭＴオペレータがある [Weiler86c]。この操作は、ソリッドモデルのオイラー操作からの類推により、経験と直感に基づいて決められており、オイラー操作のような位相的な関係式に基づいてはいないため、オイラー操作の優れた性質は受け継いでいない。そのため、次のような問題点があった。

1. 位相操作の決め方が場当たり的である。
2. どのような基本操作をあらかじめ用意しておけば非多様体形状モデリングにとって十分なのかが不明である。
3. NMT オペレータを組み合わせて実現される位相操作が定義域に関して閉じているかが不明である。
4. １対１に対応する逆操作の存在が保証されていない。

3.3 本研究の方針

非多様体形状モデルの基本操作として最も望ましいと考えられるのは、ソリッドモデリングのオイラー操作の利点をすべて引き継いでいる操作群である。そのためには任意の非多様体形状で成立する関係式を導出し、その式に基づいて非多様体形状モデルのオイラー操作を定義する必要がある。

しかし、そのような関係式はこれまで提案されていなかった。そこで、本論文では、非多様体形状モデルの定義からオイラー式に相当する関係式を導出し、それに基づいたオイラー操作を提案する。

オイラー操作に関する議論の構成

非多様体形状モデルのオイラー操作に関する議論は、以下のような構成でなされる。

まず、次の４節では位相幾何学の基礎的事項について解説する。ここでは、ホモロジー群の概念について述べ、複体のオイラー・ポアンカレの式 3.16 が導かれる数学的背景について述べる。

５節では、複体のオイラー・ポアンカレの式を利用することによって、非多様体形状モデルのための位相的な関係式を導出する方法について述べる。

６節では、導出した関係式に基づき、独立な９種類のオイラー操作があれば任意の非多様体形状モデルが生成できることを証明する。また、非多様体形状モデリングのための基本位相操作群についても論じる。

７節では関係式を限定的に適用することによって、様々な定義域の形状でオイラー操作が導出できることを示す。

８節では、オイラー操作の実装について論じる。

3.4 数学的背景

本節では、複体のためのオイラー・ポアンカレの式とその数学的な背景について説明する。ここでの目的は主として、位相幾何学の基礎概念について説明し、複体のオイラー・ポアンカレの式

$$\sum_{p=0}^{dimK} (-1)^p \alpha_p = \sum_{p=0}^{dimK} (-1)^p \beta_p$$

（α_p はp胞体の個数, β_p はp-Betti数）を導出することなので、既に十分な知識がある場合や、この式の導出法に関する詳細な議論に興味のない場合には、次節に飛んでも差し支えない。

以下において、まずオイラー式の基礎となっているホモロジー群とBetti数の概念について説明する。次にこれらの概念が3次元形状モデリングでどのような意味を持つかについて述べる。最後に、複体のオイラー・ポアンカレの式について説明する。なお、ここでの数学的な定義や記述法は、主として、位相幾何学の参考書 [数学 1, 数学 4, 数学 7] に基づいている。

3.4.1 ホモロジー群

オイラー式の基本的な考え方は、形状を分類するための指標を導入することである。境界が2-多様体のソリッドモデルの取りうる形状には無限の種類がある。しかし、連結成分の個数, 貫通穴の個数をそれぞれ x 座標と y 座標とすると、様々なソリッドモデルが同一の格子点に対応する。たとえば、四面体は (1, 0) に対応するが、この対応付けでは多角柱、多角錐や球などはすべて同一の点に対応する。このように、位相が異なっても同じになる量を位相不変量と呼ぶ。

位相不変量による分類を一般化したものがホモロジー群である。連結成分と貫通穴による分類は、2-多様体のホモロジー群に相当している。ここでは、複体のホモロジー群について考えるが、そのためにまず鎖と輪体の概念について説明し、ホモロジー群について定義する。

数学用語の定義

複体では形状は胞体の集合で構成される。ただし、ここで扱う複体は、有限個の胞体から構成され、ユークリッド空間の部分集合である有限ユークリッド胞複体を指すも

のとする。本論文では、単に複体と呼ぶ。

まず、胞体として、頂点 (0-胞体) $\boldsymbol{v}_i$、辺 (1-胞体) $\boldsymbol{e}_j$、面 (2-胞体) $\boldsymbol{f}_k$ を考える。これらを組み合わせることによって幾何形状が作られるが、幾何形状を胞体の代数的計算に帰着させるために、
頂点 $\boldsymbol{v}_1, ..., \boldsymbol{v}_p$ の整数係数1次結合として、

$$\boldsymbol{c}_0 = \sum_{i=1}^{p} a_i \boldsymbol{v}_i$$

辺 $\boldsymbol{e}_1, ..., \boldsymbol{e}_q$ の整数係数1次結合として、

$$\boldsymbol{c}_1 = \sum_{i=1}^{q} b_i \boldsymbol{e}_i$$

面 $\boldsymbol{f}_1, ..., \boldsymbol{f}_r$ の整数係数1次結合として、

$$\boldsymbol{c}_2 = \sum_{i=1}^{r} c_i \boldsymbol{f}_i$$

を考える。このとき、$\boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2$ をそれぞれ **0-鎖**, **1-鎖**, **2-鎖** と呼ぶ。

例として図 3.5 のように胞体分割された形状を考えると、すべての折れ線は1-鎖の形で書くことができ、たとえば折れ線 BDC であれば、辺の線形結合として、$\boldsymbol{c}_1 = BD + DC$ と記述できる。領域に関しても同様で、例えば領域 ABDC であれば二つの面 ABC と CBD の和なので、$ABC + CBD$ と書くことができる。

また、p-鎖からその境界である (p−1)-鎖に対応させる写像を**境界作用素** ∂ と書くことにする。例えば、図 3.5 における面 ABC, 辺 AB, 頂点 A の ∂ はそれぞれ次のようになる。

$$\partial(ABC) = AB + BC + CA$$

$$\partial(AB) = B - A$$

$$\partial A = 0$$

一般に、p-鎖 $A_0 A_1 A_2 ... A_p$ の ∂ は、

$$\partial(A_0 A_1 A_2 ... A_p) = \sum_{k=0}^{p} (-1)^k A_0 A_1 ... \overline{A_k} ... A_p \tag{3.5}$$

となる。ただし、$\overline{A_k}$ は A_k を除くという意味である。

次に輪体について定義する。まず、複体 K におけるすべての p-鎖の集合を $C_p(K)$ と書くものとする。このとき、$\boldsymbol{c} \in C_p(K)$ でかつ $\partial \boldsymbol{c} = 0$ となるとき、$\boldsymbol{c}$ を K の

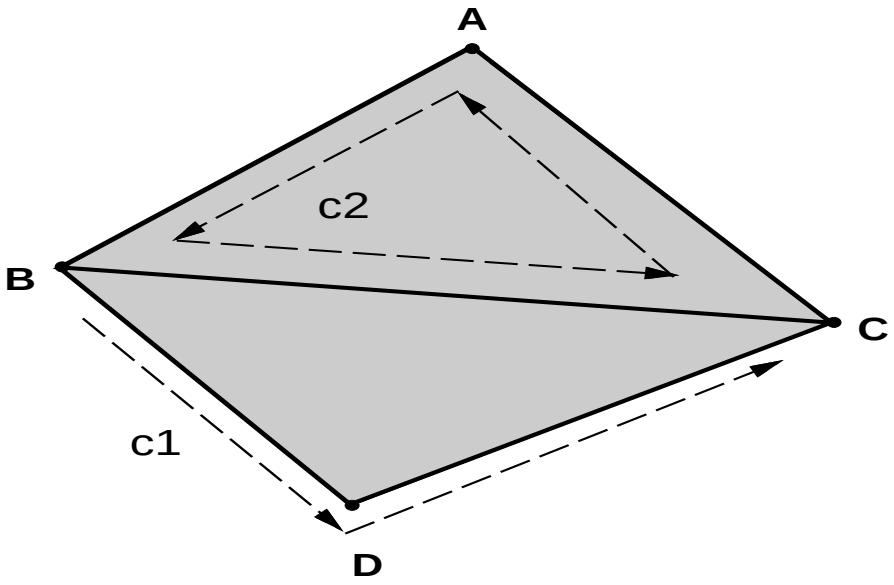


図 3.5: 1-鎖 (C1,C2) と 1-輪体 (C2).

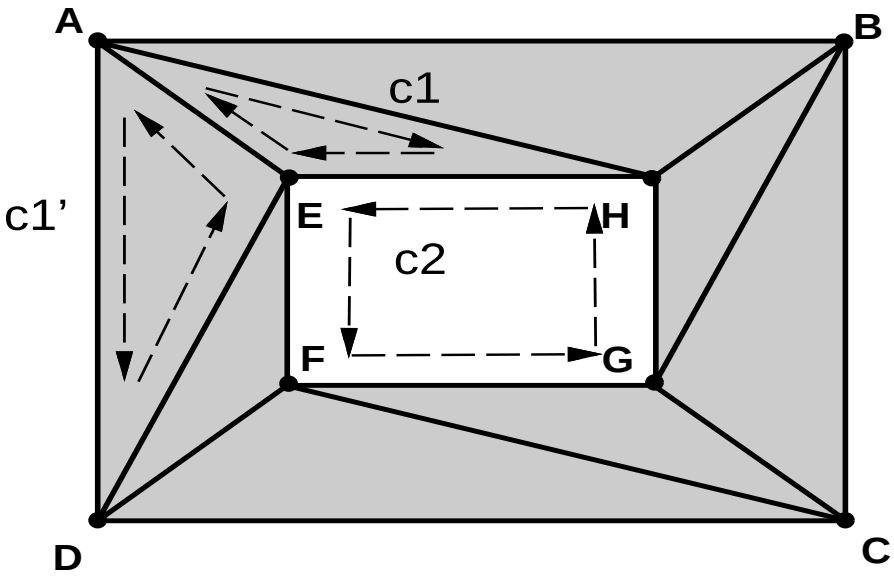


図 3.6: ホモローグな 1-輪体 (C1,C1' はホモローグ, C1,C2 はホモローグでない).

p-輪体 という。図 3.5 の例では、1-鎖 $\boldsymbol{c}_1 = BD + DC$ は輪体ではないが、1-鎖 $\boldsymbol{c}_2 = AB + BC + CA$ は、

$$\partial(AB + BC + CA) = (B - A) + (C - B) + (A - C) = 0$$

となるので輪体である。

また、p-輪体 $\boldsymbol{c}$ に対して $\partial \boldsymbol{c} = \boldsymbol{b}$ となる (p+1)-輪体 $\boldsymbol{b}$ が存在するとき、$\boldsymbol{c}$ を**境界輪体**という。図 3.5 では、2-輪体 ABC、BDC の境界上の辺は境界輪体となるので、すべての1-輪体は境界輪体である。また、この例では3-輪体が存在しないので、2-輪体 ABC, BDC は境界輪体ではない。

ホモロジー群の定義

ここで、輪体の同値関係を次のように決める。二つの p-輪体 $\boldsymbol{c}_1, \boldsymbol{c}_2$ に対し、$\boldsymbol{c}_1 - \boldsymbol{c}_2$ を境界とする (p+1)-輪体が存在するとき、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ は同値と考える。このとき、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ は **ホモローグ** であるという。このような同値関係によって全体を分類したとき、同値類の全体を**p次元ホモロジー群** といい、$H_p(K)$ と書く。

図 3.6 の例では、1-輪体 $\boldsymbol{c}_1 = (AH + HE + EA)$ と $\boldsymbol{c}'_1 = (AD + DE + EA)$ を考えると、$\boldsymbol{c}_1 - \boldsymbol{c}'_1 = (AH + HE + ED + DA)$ は 2-輪体 $AHE + ADE$ の境界なので、$\boldsymbol{c}_1$ と $\boldsymbol{c}'_1$ はホモローグである。また、$\boldsymbol{c}_1$ 自身も境界輪体なので、$\boldsymbol{c}_1 - 0$ もまた境界輪体である。すなわち、$\boldsymbol{c}_1$ と $\boldsymbol{c}'_1$ は 0とホモローグでもある。一方、1-輪体 $\boldsymbol{c}_2 = (EF + FG + GH + HE)$ を考えると、$\boldsymbol{c}_1 - \boldsymbol{c}_2 = (AH + HG + GF + FE + EA)$ となるが、四角形 $EFGH$ に相当する部分が空となっているので、$\boldsymbol{c}_1 - \boldsymbol{c}_2$ を境界に持つ2-輪体は存在しない。したがって、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ はホモローグではない。

次に Betti 数を定義する。複体 K の p-鎖の全体を $C_p(K)$ としたとき、p-輪体の集合 $Z_p(K)$、p 次元の境界輪体の集合 $B_p(K)$、p- 次元ホモロジー群 $H_p(K)$ は次のように記述できる。

$$Z_p(K) = [\boldsymbol{c} \in C_p(K); \partial \boldsymbol{c} = 0] \tag{3.6}$$

$$B_p(K) = [\partial \boldsymbol{c}; \boldsymbol{c} \in C_{p+1}(K)] \tag{3.7}$$

$$H_p(K) = Z_p(K)/B_p(K) \tag{3.8}$$

ここで、$Z_p(K)/B_p(K)$ は、輪体の同値関係による類の全体を意味しており、$Z_p(K)$ の $B_p(K)$ による商空間と呼ばれる。

このとき、p次元ホモロジー群の次数 $dim H_p(K)$ を **p次元 Betti 数**と呼び、β_p と書く。図 3.5 の複体では、すべての1-輪体は0にホモローグなので、$\beta_1 = 0$ となる。

一方、図 3.6 に示した複体の１次元 Betti 数を考えると、輪体 $\boldsymbol{c}_2$ が境界輪体でなく、すべての輪体が０か $\boldsymbol{c}_2$ にホモローグなので、$\beta_1 = 1$ となる。

3.4.2 ３次元形状モデルの Betti 数

ホモロジー群や Betti 数という考え方は直観的には捉えにくいものであり、機械設計のための形状モデリングの用語としては馴染みにくいと思われる。そこで、３次元形状モデルにおける Betti 数の幾何学的な意味を考えてみたい。

３次元形状モデリングでは、４次元以上の位相要素は存在しないから、０次元、１次元、２次元の Betti 数を考えればよい。

０-Betti 数

０次元ホモロジー群 $H_0(K)$ は次のように解釈できる。いま、頂点 $v_1, v_2, ..., v_l$ を持つ形状モデル K を考える。このとき、任意の２頂点 v_i, v_j に関して、K の実体部分またはその境界を通って v_i から v_j にいくパスがあるかどうかを考える。このパスが存在するときに v_iと v_j が同じ類になると考えて頂点全体を分類したものが０次元ホモロジー群である。このような指標で頂点を分類していくと、連結な頂点は同じ類になるから、頂点全体は互いに非連結な類の集合に分類されることになる。すなわち、０次元ホモロジー群は、形状モデル K が連結かどうか、また連結でないなら連結成分の個数はいくつであるかを示す指標であるといえる。したがって、０-Betti 数は連結成分の個数と一致する。

１-Betti 数

１次元ホモロジー群では１-輪体について考える。ここでは、図 3.7のような貫通穴を持った形状を例に考える。図において、$\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ は輪体を示している。

まず、この形状の内部が詰まっている状態、すなわち、立体の内部に３-胞体が存在する場合を考えてみる。このときの形状を K_1^{fill} と書くものとする。輪体 $\boldsymbol{c}_1$ は面の境界となっているので、明らかに境界輪体であり、０とホモローグである。また、$\boldsymbol{c}_3$ も内部が詰まっているという仮定から境界輪体となる。一方、$\boldsymbol{c}_2$ は境界輪体ではない。なぜなら、$\boldsymbol{c}_2$に囲まれる領域は K_1^{fill} に含まれないからである。K_1^{fill} では、すべての１-輪体は $n\boldsymbol{c}_2$ (n は任意の整数) とホモローグになる。すなわち、$H_1(K_1^{fill})$ は $\mathcal{Z}\boldsymbol{c}_2$ の

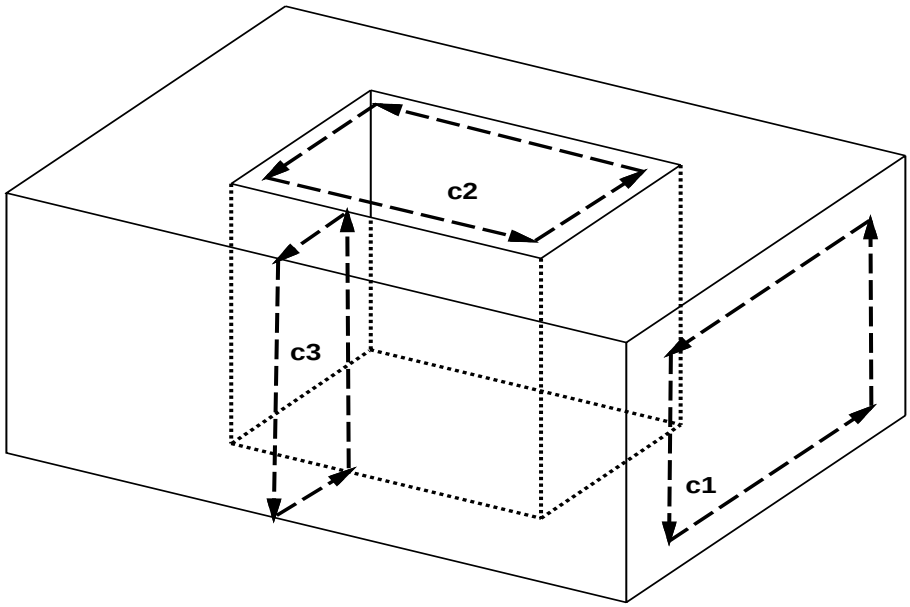


図 3.7: 貫通穴を持った形状における 1-輪体. (内部の詰まったソリッドならば独立な輪体は c2 のみ.)

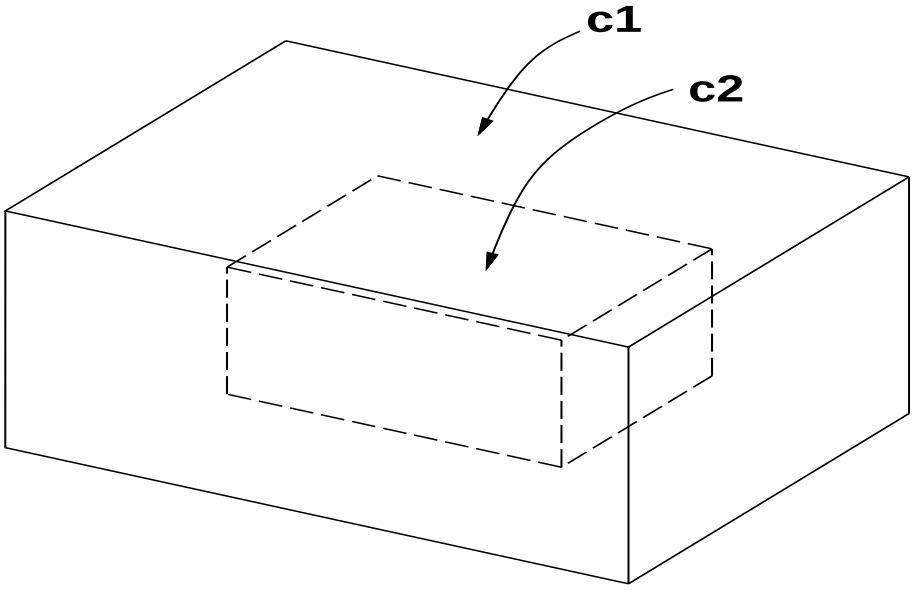


図 3.8: 複数の 連結成分 を持った形状における 2-輪体. （空洞を持ったソリッドならば独立な輪体は c2 のみ.）

形に書けるので、

$$H_1(K_1^{fill}) \cong \mathcal{Z}$$

となり、K_1^{fill} の 1-Betti 数は 1 となる。ここで、$\cong$ は 同型であることを示す記号であり、また $\mathcal{Z}$ は整数全体を示している。$\boldsymbol{c}_2$ に相当する 1- 輪体は、貫通穴の個数が 1 個増えるごとにその周囲を一周するように 1 個作れるので、立体の 1-Betti 数は貫通穴の個数に一致することがわかる。

一方、図 3.7 を閉曲面で、その内部に 3-胞体が存在しない形状 K_1^{void} と考えることもできる。これはソリッドモデルを 2-多様体と考える定義に相当する。このように考えた場合では $\boldsymbol{c}_3$ を境界とする 2-輪体は K_1^{void} 上には存在しないので、$\boldsymbol{c}_3$ は境界輪体ではない。また、$\boldsymbol{c}_2 - \boldsymbol{c}_3$ を境界とするような 2-輪体も存在しないので、$\boldsymbol{c}_2$ と $\boldsymbol{c}_3$ はホモローグでない。この例では、すべての 1-輪体は $m\boldsymbol{c}_2 + n\boldsymbol{c}_3$ (m, n は任意の整数) にホモローグなので、1 次元ホモロジー群は、$\mathcal{Z}\boldsymbol{c}_2 + \mathcal{Z}\boldsymbol{c}_3$ の形に書け、

$$H_p(K_1^{void}) \cong \mathcal{Z} + \mathcal{Z}$$

となる。したがって、1-Betti 数は 2 である。$\boldsymbol{c}_2, \boldsymbol{c}_3$ に相当する 1-輪体は貫通穴の個数が 1 増えるごとにそれぞれ 1 個ずつ増えるので、閉曲面の 1-Betti 数は貫通穴の個数の 2 倍に等しい。オイラー式 3.2 の右辺の h に係数 2 がつくのはこの理由からである。

2-Betti 数

次に 2 次元ホモロジー群を考える。図 3.8 は、空洞を持った形状モデルの例である。この図で、$\boldsymbol{c}_1, \boldsymbol{c}_2$ は 2-輪体で、それぞれ立体の外側の境界面と、空洞の境界面を示している。この場合も、形状の内部が詰まった場合と、境界面についてのみ考える場合の二通りについて示す。

立体内部に 3-胞体が存在するときは、$\boldsymbol{c}_1 - \boldsymbol{c}_2$ を境界とする 3-輪体が存在し、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ はホモローグとなる。ここで、$\boldsymbol{c}_2$ は空洞の境界であり、$\boldsymbol{c}_2$ を境界とする 3-輪体は存在しないので、$\boldsymbol{c}_2$ は境界輪体ではない。K_2^{fill} では、すべての 2-輪体は $n\boldsymbol{c}_1$ とホモローグであり、2 次元ホモロジー群は $\mathcal{Z}\boldsymbol{c}_1$ の形に書け、

$$H_2(K_2^{fill}) \cong \mathcal{Z}$$

となるので、2-Betti 数は 1 となる。$\boldsymbol{c}_2$ に相当する 2-輪体は立体に空洞が 1 個できるごとに 1 増えるので、立体の 2-Betti 数は空洞の個数に一致する。

一方、図 3.8 を閉曲面で 3-胞体が存在しない形状 K_2^{void} と考えた場合は、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ は非連結なので、2 個の閉曲面と考えることができる。当然、$\boldsymbol{c}_1$ と $\boldsymbol{c}_2$ はホモローグで

ない。しかも、3-胞体が存在しないので、$\boldsymbol{c}_1$ も$\boldsymbol{c}_2$ も境界輪体ではない。したがって、２次元ホモロジー群は、$\mathcal{Z}\boldsymbol{c}_1 + \mathcal{Z}\boldsymbol{c}_2$ の形に書け、

$$H_2(K_2^{void}) \cong \mathcal{Z} + \mathcal{Z}$$

となり、2-Betti 数は２になる。$\boldsymbol{c}_1$ に相当する２-輪体は閉曲面１個につき１個存在するので、閉曲面では 2-Betti 数は連結成分の個数と等しくなる。したがって、２-多様体では、2-Betti 数と 0-Betti 数はともに shell の個数に等しい。オイラー式 3.2 の右辺の 2s は 2-Betti 数と 0-Betti 数 の和に相当している。すなわち、ソリッドモデルのオイラー・ポアンカレの式 3.2 の右辺は、p-Betti 数を β_p、連結成分の個数 を s、貫通穴の個数を h とするとき、

$$\chi = \beta_0 - \beta_1 + \beta_2 = s - 2\ h + s = 2\ (s - h) \tag{3.9}$$

となっている。 χ はオイラー特性数と呼ばれる位相不変量である。

3.4.3 複体のオイラーポアンカレの式

次に複体のオイラー特性数を考えることによって、複体のオイラー・ポアンカレの式を導く。p次元ホモロジー群 $H_p(K)$ の次数を β_p とし、また、複体 K のオイラー特性数 $\chi(K)$ を

$$\chi(K) = \sum_{p=0}^{dimK} (-1)^p \beta_p \tag{3.10}$$

と定義する。

このとき、Kにおけるp-胞体の個数を α_p 、Kの p-鎖の全体を $C_p(K)$ とすると、$C_p(K)$ はすべてのp-胞体 $\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_{\alpha_p}$ の１次結合として、

$$C_p(K) = \sum_{i=1}^{\alpha_p} a_i \boldsymbol{e}_i \quad (a_i\text{は任意の整数})$$

と表されることから、

$$\alpha_p = dim C_p(K) \tag{3.11}$$

となる。また、線形写像として境界作用素 ∂ を考え、$C_{l-1}(K) \xleftarrow{\partial} C_l(K)$ とするとき、p-輪体の全体 $Z_p(K)$ は定義（式3.6）より写像 ∂ によって0 に移されるので、∂ の核となる。また、$\partial C_p(K)$ の像はp 次元の境界の全体 B_{p-1}である。したがって、線形代数の階数と核の次元の定理 [数学 6] を用いることにより、

$$\alpha_p = dim C_p(K) = dim(\partial^{-1}\ 0) + dim(\partial C_p(K)) = dim Z_p(K) + dim B_{p-1}(K) \tag{3.12}$$

と変形できる。また、商空間 V/W の次元は $dimV - dimW$ であるから [数学 6]、

$$\beta_p = dimH_p(K) = dim(Z_p(K)/B_p(K)) = dimZ_p(K) - dimB_p(K) \tag{3.13}$$

となる。また、最高次元である $dimK$ 次元の境界輪体は存在しないことから、

$$B_{dimK}(K) = 0 \tag{3.14}$$

である。以上の式より、式 3.10は次のように変形できる。

$$\begin{aligned}
&\chi(K) \\
&= \sum_{p=0}^{dimK} (-1)^p \beta_p \\
&= \sum_{p=0}^{dimK} (-1)^p \{dimZ_p(K) - dimB_p(K)\} && (\text{式 } 3.13 \text{より}) \\
&= \sum_{p=0}^{dimK} (-1)^p dimZ_p(K) + \sum_{p=0}^{dimK} (-1)^{p+1} dimB_p(K) \\
&= \sum_{p=0}^{dimK} (-1)^p dimZ_p(K) + \sum_{p=1}^{dimK} (-1)^p dimB_{p-1}(K) && (\text{式 } 3.14 \text{より}) \\
&= \sum_{p=0}^{dimK} (-1)^p \{dimZ_p(K) + dimB_{p-1}(K)\} && (B_{-1}(K) = 0 \text{より}) \\
&= \sum_{p=0}^{dimK} (-1)^p \alpha_p && (\text{式 } 3.12 \text{より})
\end{aligned} \tag{3.15}$$

よって、p胞体の個数 α_p と p-Betti 数との関係式として、次式が導かれる。

$$\sum_{p=0}^{dimK} (-1)^p \alpha_p = \sum_{p=0}^{dimK} (-1)^p \beta_p \tag{3.16}$$

この式は、n 次元有限ユークリッド胞複体のオイラーポアンカレの式と呼ばれる。

3.5 非多様体形状モデルのためのオイラー・ポアンカレの式の導出

本節では、複体のオイラー・ポアンカレの式を利用して、非多様体形状モデルの位相要素間に成立する位相的な関係式を導出する。

複体では空洞や貫通穴を持つ位相要素は許されないが、形状モデリングでは空洞や貫通穴持った位相要素を許している。そのため、非多様体形状モデルに、直接、複体のオイラー・ポアンカレの式を適用することはできない。そのため、Betti 数に影響を与える空洞や貫通穴の個数も考慮した関係式を算出する。

3.5.1 位相的な関係式の導出

関係式のパラメータ

非多様体形状モデルのオイラー特性数に影響を与えるパラメータとしては次のものがある。

- $\boldsymbol{v}$: vertex の個数
- $\boldsymbol{e}$: edge の個数
- $\boldsymbol{f}$: face の個数
- $\boldsymbol{r}$: face の空洞（ring）の個数
- $\boldsymbol{V}$: volume の個数
- $\boldsymbol{Vh}$: volume の貫通穴の個数
- $\boldsymbol{Vc}$: volume の空洞の個数

ここで、貫通穴、空洞という用語の意味を説明しておく。図 3.9 は立体、サーフェス、ワイヤフレームにおける貫通穴を示している。いずれも破線の矢印で示したように、境界輪体ではないただ一つの独立な 1 -輪体を持つので、１個の貫通穴を持つと考える。また、図 3.10 は空洞を示している。(a) は空洞を持ったソリッド、(b) は閉じたサーフェスである。(a) (b) いずれの場合も形状の内部に空のスペースが１個存在するので、空洞が１個存在すると考えることができる。

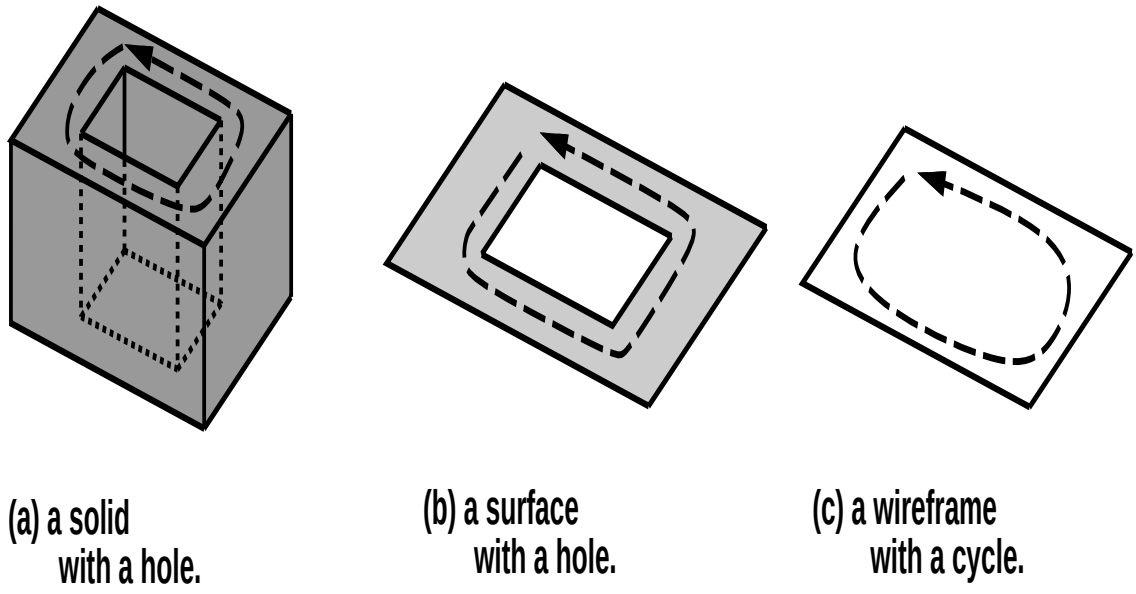


図 3.9: ソリッド , サーフェス, ワイヤフレームにおける貫通穴.

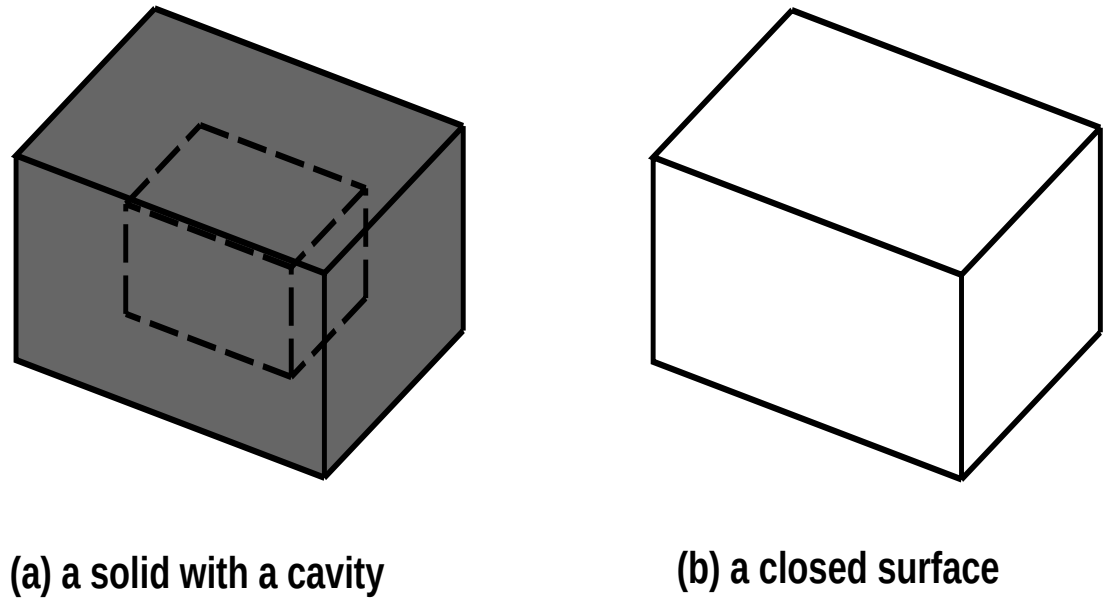


図 3.10: ソリッド , サーフェス, ワイヤフレームにおける空洞.

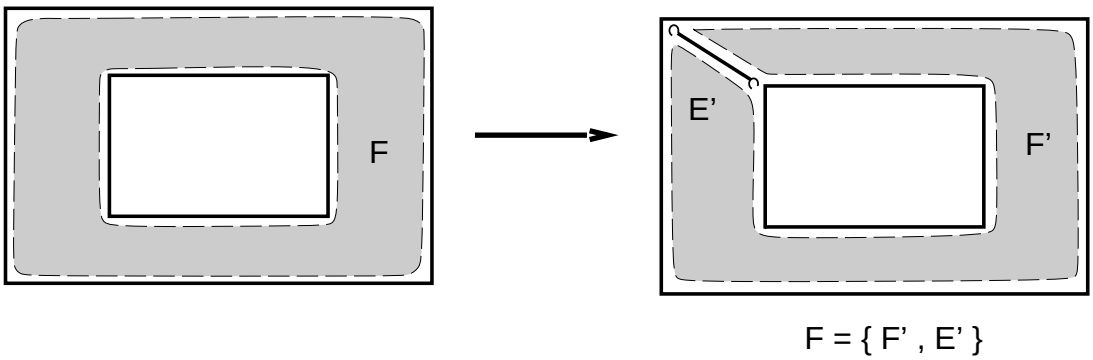
また、形状モデルのオイラー特性数は、その Betti 数によって、$\beta_1 - \beta_2 + \beta_3$ と表されるが、ここでは直感的な理解のしやすさを考慮して、Betti 数と等価な次の量をパラメータとして考える。

- $\boldsymbol{C}$: 形状モデルの連結成分の個数 (0 -Betti 数)。
- $\boldsymbol{Ch}$: 形状モデルの貫通穴の個数 (1 -Betti 数)。
- $\boldsymbol{Cc}$: 形状モデルの空洞の個数 (2 -Betti 数)。

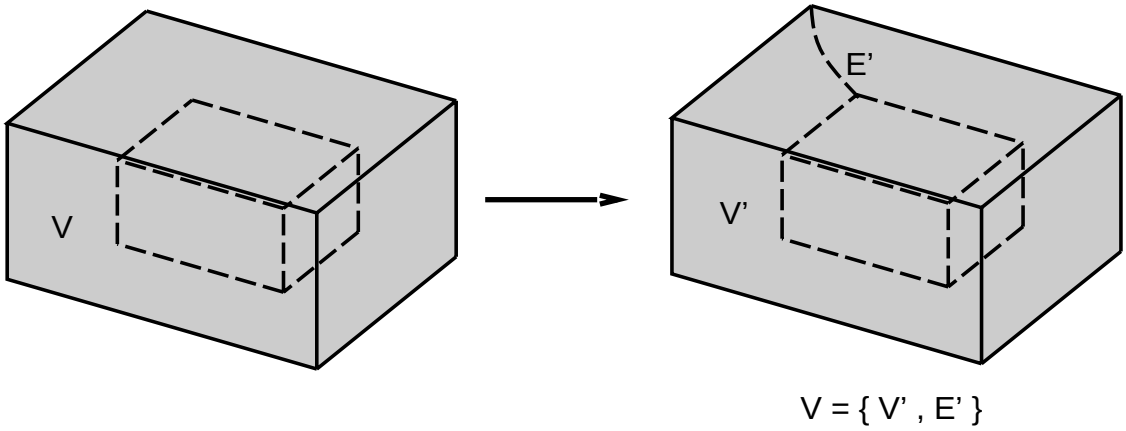
非多様体形状モデルの胞体分割

さて、このとき、これらのパラメータ間に成立する関係式を求める。複体のオイラー・ポアンカレの式では、パラメータが胞体の個数なので、この式を利用するためには、空洞や貫通穴を持った face や volume を胞体分割する必要がある。非多様体形状モデルは以下の手順によって胞体分割することができる。

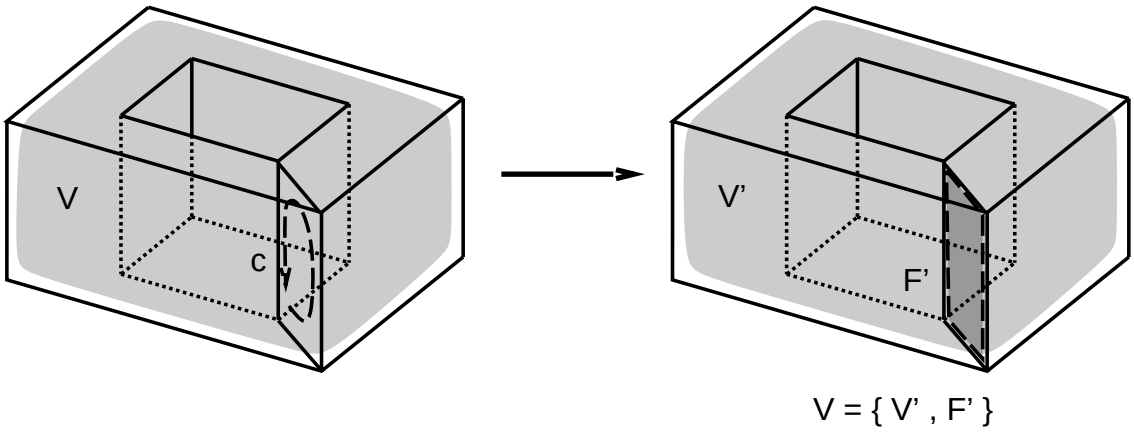
1. 空洞 を持った face は、図 3.11 (a) に示すように、各 loop 上の一点を結ぶ edge を加えることによって胞体の集合に分割することができる。このとき、面 F は、 2 -胞体 F' と 1 -胞体 E' に分割される。このような分割をすべての空洞を持った face に施せば、face の集合は 2 -胞体と 1 -胞体の集合となる。このとき、ring の総数が r 個であるとすると、この分割によって新たに r 個の edge が加えられたことになる。したがって、空洞を持った face を胞体分割すると edge の総数は r 個増えて ($e + r$) 個となる。
2. さらに、空洞を持った volume では、volume の境界の連結成分が複数あるが、図 3.11 (b) に示すように、非連結な境界位相要素を結ぶ edge E' を新たに加えることによって、 3 -胞体 V' と 1 -胞体 E' に分割することができる。volume の空洞の総数が Vc 個とすると、この分割によって新たに Vc 個の edge ができるので、edge の総数は ($e + r + Vc$) となる。
3. 次に、貫通穴を持った volume を分割する。volume が貫通穴をもつ場合には、まず、図 3.11 (c) のような 1 -輪体 c ができるように volume 境界上の face や edge を適当に胞体分割する。volume 境界上の face は手順 1 によりすでに胞体分割されているので、さらに分割しても volume の境界位相要素のオイラー特性数 $v - e + f$ の増減は 0 である。このとき、volume V を胞体に分割するためには、 1 -輪体 $\boldsymbol{c}$ を境界に持つような face F' を新たに付加すればよい。それによって、volume は 3 -胞体 V' と 2 -胞体 F' に分割することができる。volume の貫通穴の



(a) a face with a cavity.



(b) a volume with a cavity.



(c) a volume with a hole.

図 3.11: 貫通穴や空洞を持った位相要素の胞体分割.

総数が h 個のときは、すべての volume を胞体分割するには Vh 個の face を新たに加える必要があり、 face の総数は、$(f + Vh)$ 個となる。

関係式の導出

以上によって、すべての位相要素は胞体に分割された。ここで、胞体分割された非多様体形状モデルの p-胞体の個数 α_p を考えると、

$$\alpha_0 = v,\ \alpha_1 = e + r + Vc,\ \alpha_2 = f + Vh,\ \alpha_3 = V \tag{3.17}$$

となる。また、形状モデル全体において、連結成分の個数を C、貫通穴の個数を Ch、空洞の個数を Cc として、

$$\beta_0 = C,\ \beta_1 = Ch,\ \beta_2 = Cc \tag{3.18}$$

とおく。以上を複体 K のオイラー・ポアンカレの式：

$$\sum_{p=0}^{dimK} (-1)^p \alpha_p = \sum_{p=0}^{dimK} (-1)^p \beta_p$$

に代入すると次式を得ることができる。

$$\underline{\mathbf{v} - \mathbf{e} + (\mathbf{f} - \mathbf{r}) - (\mathbf{V} - \mathbf{Vh} + \mathbf{Vc}) = \mathbf{C} - \mathbf{Ch} + \mathbf{Cc}} \tag{3.19}$$

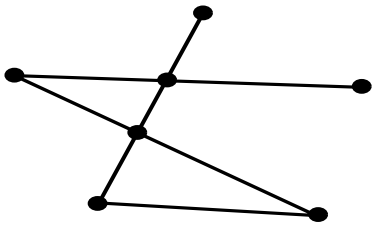
この式が本論文で定義したすべての非多様体形状モデルで成立する関係式である。式のパラメータは位相要素、貫通穴、空洞の個数という形状モデリングにおいて一般に用いられてきた用語から構成されており、直観的にも捉えやすい。本論文では、式 3.19 を非多様体形状モデルのオイラー・ポアンカレの式として定義し、この式の関係を壊さない位相変形操作をオイラー操作と定義する。

3.5.2 適用例

式 3.19 を幾つかの形状に対して適用した例を示す。

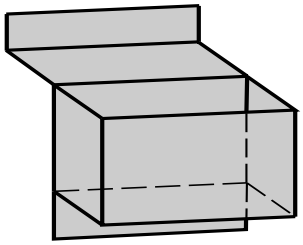
図 3.12 は、式 3.19 をワイヤフレームモデルやサーフェスモデルに当てはめた例である。

- 図 3.12 (a) は、式 3.19 をワイヤフレームモデルに適用した例である。



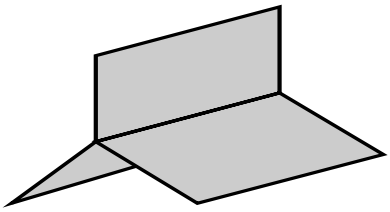
v = 7, e = 8,
f = r = 0,
V = Vh = Vc = 0
C = 1, Ch = 2, Cc = 0

(a) v−e+(f−r)−(V−Vh+Vc) = C − Ch + Cc = −1



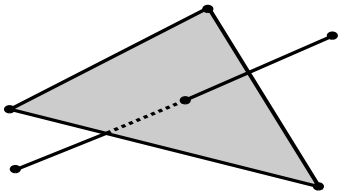
v = 14, e = 21,
f = 7, r = 0,
V = Vh = Vc = 0,
C = 1, Ch = 1, Cc = 0

(b) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 0



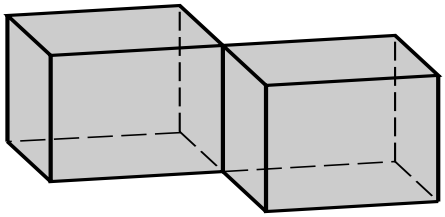
v = 8, e = 10,
f = 3, r = 0,
V = Vh = Vc = 0,
C = 1, Ch = Cc = 0

(c) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1



v = 6, e = 5,
f = 1, r = 1,
V = Vh = Vc = 0,
C = 1, Ch = Cc = 0

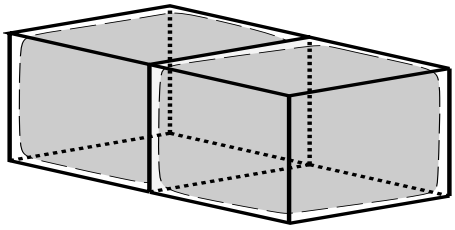
(d) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1



v = 14, e = 23,
f = 12, r = 0,
V = Vh = Vc = 0,
C = 1, Ch = 0, Cc = 2

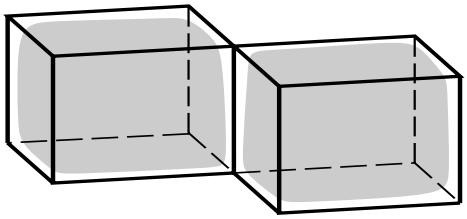
(e) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 3

図 3.12: オイラー・ポアンカレの式の適用例 (ワイヤフレームとサーフェスモデル)



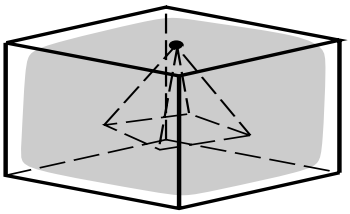
v = 12, e = 20,
f = 11, r = 0,
V = 2. Vh = Vc = 0.
C = 1, Ch = Cc = 0

(a) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1



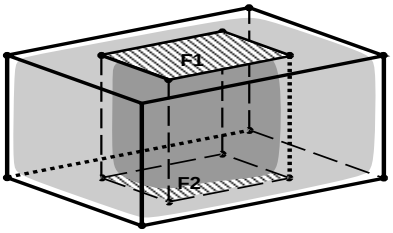
v = 14, e = 23,
f = 12, r = 0,
V = 2, Vh = Vc = 0,
C = 1, Ch = Cc = 0

(b) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1



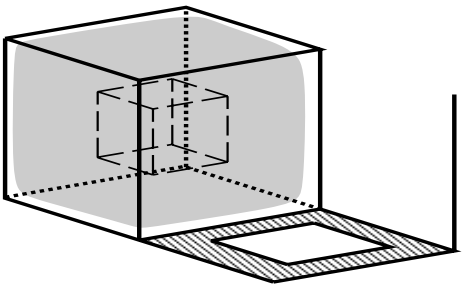
v = 13, e = 20,
f = 11, r = 1,
V = 1, Vh = Vc = 0,
C = 1, Ch = 0, Cc = 1

(c) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 2



v = 16, e = 24,
f = 12, r = 2,
V = 2, Vh = 1, Vc= 0,
C = 1, Ch = Cc = 0

(d) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1



v = 23, e = 32,
f = 13, r = 1,
V = 1, Vh = 0, Vc = 1,
C = 1, Ch = 1, Cc = 1

(e) v−e+(f−r)−(V−Vh+Vc) = C−Ch+Cc = 1

図 3.13: オイラー・ポアンカレの式の適用例 (非多様体を含んだソリッド).

- 図 3.12 (b) は 紙モデル の例である。
- 図 3.12 (c) は３枚の面に共有された非多様体稜線を含んだ形状である。
- 図 3.12 (d) はサーフェスとワイヤが混在した例である。この形状では face に微小な穴があいているため、ring の個数は１個となる。
- 図 3.12 (e) は二つの閉じたサーフェスが稜線を共有してできる形状である。この例では、面に囲まれた閉空間が２個存在するので、空洞の個数 Cc は２である。

図 3.13は立体形状に対して適用した例である。

- 図 3.13 (a) は、二つの立体が面を共有した形状である。したがって、２個の volume が存在する。
- 図 3.13 (b) は、立体が稜線を共有した形状である。この形状では各閉空間が volume で埋められているので volume の個数 V は２となる。
- 図 3.13 (c) は、直方体の境界上に頂点を持つ四角錐の空洞が存在する形状を示している。
- 図 3.13 (d) は貫通穴のある volume を含む形状の例である。この形状は全体としては直方体であるが、それを構成する位相要素は貫通穴を持つ volume と、その貫通穴を上面 F1 と 下面 F2 で塞いでできる閉空間を埋める volume から成っている。
- 図 3.13 (e) はソリッド、サーフェス、ワイヤフレームが混在した例である。

以上の例で示したように、式 3.19 はワイヤフレーム、サーフェス、２-多様体、正則集合、胞体分割された多様体、またそれらが混在する形状でも成立していることがわかる。

3.6 非多様体形状モデルのオイラー操作

非多様体形状モデルのオイラー操作とは、関係式

$$v - e + (f - r) - (V - Vh + Vc) = C - Ch + Cc$$

を保持するような位相操作群として定義される。本節では、非多様体形状モデルのオイラー操作について考察する。

3.6.1 最小限のオイラー操作

非多様体形状モデルのオイラー操作の基礎となる関係式 3.19 は１０個の変数によって構成されている。そこで、これらの変数で張られる１０次元空間

$$R^{10}: \quad (v, e, f, r, V, V_h, V_c, C, C_h, C_c)$$

を考える。この空間において、$v - e + (f - r) - (V - Vh + Vc) = C - Ch + Cc$ は超平面 E を表していると考えることができる。このとき、任意の非多様体形状モデルはこの平面上の点に相当する。この超平面 E 上の基底ベクトルの個数は９個であり、E の任意の点はこれらの基底ベクトルの線型結合によって表すことができる。ここで、オイラー操作とは式 3.19 の変数の値を変化させる操作であるから、R^{10} では、E 上のベクトルと見做すことができる。したがって、次が成立することがわかる。

「非多様体形状モデルでは、９個の独立な位相操作とその逆操作があれば、その組合せで任意の位相構造を生成することができる。」

逆操作とは、基底ベクトルに -1 を乗じたベクトルに相当する操作のことである。

９種類のオイラー操作

独立な９種類のオイラー操作には様々な選び方が可能であるが、一般には関係式の変数が単位量変化するようなものが選ばれる。図 3.14 に示した９種類のオイラー操作は独立であり、かつ各変数の増減が単位量となるような操作群である。それぞれの操作が変数をどのように変化させるかを示したのが表 3.1 である。

ここで、図 3.14 の各操作の意味は次の通りである。

図 3.14: 独立な９種類のオイラー操作.

操作	v	e	f	r	V	V_h	V_c	C	C_h	C_c
mvC	$+1$	0	0	0	0	0	0	$+1$	0	0
mev	$+1$	$+1$	0	0	0	0	0	0	0	0
meC_h	0	$+1$	0	0	0	0	0	0	$+1$	0
$mfkC_h$	0	0	$+1$	0	0	0	0	0	-1	0
mfC_c	0	0	$+1$	0	0	0	0	0	0	$+1$
mvr	$+1$	0	0	$+1$	0	0	0	0	0	0
$mVkC_c$	0	0	0	0	$+1$	0	0	0	0	-1
mvV_c	$+1$	0	0	0	0	0	$+1$	0	0	0
meV_h	0	$+1$	0	0	0	$+1$	0	0	0	0

表 3.1: ９種類のオイラー操作における変数の増減

mvC ただ一つの vertex を生成することによって形状モデルを新たに作成する操作。

mev vertex から edge を伸ばす操作。ただし、この操作で生成される edge がどの位相要素の境界になるかによって (1) edge がどの位相要素の境界にもならない場合、(2) face 境界になる場合、(3) volume 境界になる場合の３通りがある。境界表現形状モデルではそれぞれで位相構造が異なるので、後節で述べるように実装に際してはこれらを区別して扱う必要がある。

meCh 同一の連結成分に属する２つの vertex を結ぶ edge を生成する操作。この操作では貫通穴が一つ増える。

mfkCh edge のループに face を定義する操作。この操作によって edge のループが塞がれるので貫通穴が一つ減少することになる。

mfCc この操作も edge のループに face を定義する操作であるが、結果として閉空間が生成される操作である。

mvr face の中に vertex を定義する操作。この操作によって face に微小な穴が開くので、ring が１増加する。

mVkCc face に囲まれた閉空間に volume を定義する操作。この操作によって、面で囲まれた閉空間がソリッドとして定義される。

mvVc volume の中に vertex を定義する操作。この操作によって、volume に微小な空洞ができる。

meVh volume の連結な境界上の２点を結ぶ edge を定義する操作。この操作によって、volume に微小の径を持った貫通穴ができる。

3.6.2 オイラー操作の適用例

以上のべたオイラー操作を組み合せて位相変形操作を構成する例を示す。ここでは形状モデルを生成する操作と形状モデルの位相構造を変更する操作の二つの場合について示す。

６面体生成

図 3.15 はオイラー操作を組み合せて六面体を生成した例である。６面体においては関係式 3.19 の各変数は、

$$(v, e, f, r, V, Vh, Vc, C, Ch, Cc) = (8, 12, 6, 0, 1, 0, 0, 1, 0, 0)$$

となるが、このベクトルは表 3.1 に示したオイラー操作の１次結合として一通りに表現できて、

$$\text{６面体：}(8, 12, 6, 0, 1, 0, 0, 1, 0, 0)$$

$$= [mvC] + 7\ [mev] + 5\ [meC_h] + 5\ [mfkC_h] + [mfC_c] + [mVkC_c]$$

となる。図 3.15 に示したように実際これらの操作を組み合わせることによって６面体を生成することができる。

以下に各手順について説明する。

1. まず、６面体の底面を定義するために、mvC によって、vertex を１個生成し (a)、
2. mev 適用することによって３個の edge を生成する (b,c)。
3. さらに、meC_h を適用すれば、(d) に示すような底面の四角形の輪郭ができ、
4. そこに $mfkCh$ を適用して face を定義すれば底面ができる (e)。
5. 同様にして mev, meC_h によって残りの edge を生成し (f,g)、
6. それらを境界とする face を $mfkC_h$ を４回適用して生成すれば (h) に示すような 紙モデル となる。

7. さらに、mfC_c によって閉じた箱を作り (i)、

8. その空洞部分に $mVkC_c$ によって volume を定義すれば６面体のソリッドモデルができる (j)。

６面体は２-多様体であるから、ソリッドモデルのオイラー操作の組合せでも生成することができるが [Mantyla82]、２-多様体を定義域とするソリッドモデルの場合は中間状態もすべて２-多様体の位相構造をとらなければなければならないため、中間状態として不自然なソリッドモデルが現れてしまう。図 3.16 は mvC, mev に相当する操作を２-多様体のオイラー操作で行った例である。いずれも位相的には球面となっておりソリッドであるが、直方体の生成過程でこのような中間形状を想定するのは直観的とはいえない。一方、非多様体形状モデルのオイラー操作では中間形状がソリッドである必要はなく、より把握しやすい位相操作が可能である。図 3.15 の一連の操作は、ワイヤフレーム → サーフェス →ソリッド、という手順で定義していくために従来のオイラー操作群くらべて直観的に理解しやすくなっている。このように、ワイヤフレームやサーフェスを許すことによって、人の思考過程に合った位相変形手順が実現できるため、比較的容易に位相操作の実装ができる。

立ち上げ操作

同様に、位相変形操作として、形状をある方向に立ち上げる操作をオイラー操作で実現することを考える。操作の手順を図 3.17 に示す。この立ち上げ操作では、ワイヤフレーム部分は面となり、面の部分は立体になる。

この操作では、まず、mev によってすべての頂点から edge を伸ばし（b)、伸ばした edge の端点に meC_h によって edge を張る (c)。さらに、edge の立ち上げた際の軌跡に相当する面を $mfKC_c$ で張っていく（d)。次に面を定義して閉空間を作る（e)。最後に３次元領域に volume を定義し、面の立ち上げ部をソリッド化する (f)。

3.6.3 その他のオイラー操作

次に、実用的な観点からどのようなオイラー操作を選択したらよいかを考えてみる。理論的な観点からは９種類のオイラー操作で十分であるが、これだけでは実用的には不十分である。たとえば、二つの立方体間に一本の edge を張って連結させた形状モデルを生成する操作を考える。この場合、表 3.1 に示したように変数 C に影響を与えることができるのは mvC のみなので、二つの立方体を連結して変数 C を１減少させるた

図 3.15: オイラー操作による六面体ソリッドの定義.

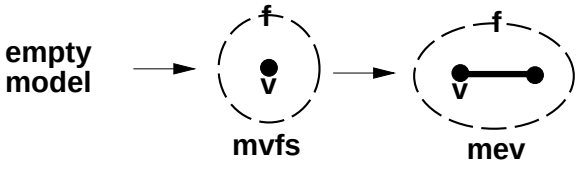


図 3.16: 2-多様体ソリッドにおける六面体ソリッドの生成過程.

図 3.17: 立ち上げ操作.

図 3.18: 非多様体形状モデリングのためのオイラー操作.

めには、一方の立方体を完全に消去した上で形状モデルを再構築しなければならない。変数 r, V_c の場合も影響を与える操作が mvr、mvV_c しかないので、二つの非連結な境界位相要素を連結する操作では、一方を消去して位相構造を再構築するしかない。貫通穴に関しても同様である。したがって、処理効率の観点からこれらを補うための操作を用意することが不可欠である。

また、edge, face, volume を分割/併合する操作は、変数の増減では９種類の操作に含めて考えることができるが、操作に必要となる入力や出力のデータが異なっていることから、別の操作として考えた方が都合がよいことが多い。

以上のような点を考慮してオイラー操作を分類したものが図 3.18 である。この図で逆操作は [ ] で囲んで示している。この図では９種類の操作に加えて、位相要素の分割併合操作として、

- edge の分割併合操作 $(split\_edge, merge\_edge)$
- face の分割併合操作 $(split\_face, merge\_face)$
- volume の分割併合操作 $(split\_volume, merge\_volume)$

の３種類、また非連結な境界位相要素の併合分割操作として、

- face の空洞の併合分割操作 $(make\_edge\_kill\_ring, kill\_edge\_make\_ring)$
- volume の空洞の併合分割操作 $(make\_edge\_kill\_Vcavity, kill\_edge\_make\_Vcavity)$
- complex の併合分割操作 $(make\_edge\_kill\_complex, kill\_edge\_make\_complex)$
- volume の貫通穴の併合分割操作 $(make\_face\_kill\_Vhole, kill\_face\_make\_Vhole)$

の４種類が追加されている。これらの操作による変数の増減を表 3.2 に示す。これらの操作を加えることによって、すべての変数に関して影響を与え得る操作が複数存在することになり、オイラー操作列への展開しやすさが改善されていることがわかる。

操作	v	e	f	r	V	V_h	V_c	C	C_h	C_c
mvC	$+1$	0	0	0	0	0	0	$+1$	0	0
mev	$+1$	$+1$	0	0	0	0	0	0	0	0
meC_h	0	$+1$	0	0	0	0	0	0	$+1$	0
$mfkC_h$	0	0	$+1$	0	0	0	0	0	-1	0
mfC_c	0	0	$+1$	0	0	0	0	0	0	$+1$
mvr	$+1$	0	0	$+1$	0	0	0	0	0	0
$mVkC_c$	0	0	0	0	$+1$	0	0	0	0	-1
mvV_c	$+1$	0	0	0	0	0	$+1$	0	0	0
meV_h	0	$+1$	0	0	0	$+1$	0	0	0	0
$mekC$	0	$+1$	0	0	0	0	0	-1	0	0
$mekr$	0	$+1$	0	-1	0	0	0	0	0	0
$mekV_c$	0	$+1$	0	0	0	0	-1	0	0	0
$mfkV_h$	0	0	$+1$	0	0	-1	0	0	0	0
spl_e	$+1$	$+1$	0	0	0	0	0	0	0	0
spl_f	0	$+1$	$+1$	0	0	0	0	0	0	0
spl_V	0	0	$+1$	0	$+1$	0	0	0	0	0

表 3.2: 分割併合のためのオイラー操作を含む変数の増減

3.7 代表的な定義域の形状におけるオイラー式

本論文で導いた関係式はこれまでの形状モデルに比べて広範な形状に対して成立する式である。したがって、非多様体形状モデルの関係式に適当な制約をつけることによって限定された定義域の形状で成立する関係式を求めることができる。そこで、非多様体形状モデルの関係式

$$v - e + (f - r) - (V - Vh + Vc) = C - Ch + Cc$$

を用いることにより、これまで比較的一般に用いられてきた定義域の形状についての位相的な関係式を導出する。

2-多様体ソリッド モデルのオイラー式

境界が2-多様体であるソリッドでは、複数の volume が連結した形状はあり得ない。よって、異なる volume は非連結でなければならない。したがって、volume の個数 V と 連結成分の個数 C は一致する。また、ソリッド モデルは、すべての位相要素は立体の境界となっており、形状モデルにおける貫通穴と空洞は volume の貫通穴、空洞と一致する。よって、

$$V = C, \quad Vh = Ch, \quad Vc = Cc. \tag{3.20}$$

ここで、volume の境界の連結成分を shell と呼び、shell の個数を s とすると、s は形状モデルの個数と空洞の個数の和に一致するので、$s = C + Cc$ である。さらに、貫通穴の個数 Ch を h で置き換えて、非多様体形状モデルの関係式に代入すると、

$$v - e + f - r = 2\ (s - h) \tag{3.21}$$

となり、Braid の提案したオイラー・ポアンカレの式 3.21 が導出できることがわかる。

紙モデルのオイラー式

紙モデルは既に述べたように一枚の紙を非多様体部分が生じないように切り貼りした形状である。紙モデルが閉曲面となる場合は2多様体ソリッドと同じなので、ここでは shell が閉じていない場合のみ考える。

開いた shell では、volume が存在せず、また空洞はないので、

$$V = Vh = Vc = 0, \ \ Cc = 0. \tag{3.22}$$

また、変数 C, Ch をそれぞれ s, h で置き換えると

$$v - e + f - r = s - h \tag{3.23}$$

となり、紙モデルのオイラー式 3.3 が導けることがわかる。

ところで、Wilson は 紙モデルにおいて非多様体稜線は許さないと定義した上でオイラー式を提案したが [Wilson85]、ここでのオイラー式の導出過程では 2-多様体という条件は用いていない。すなわち、式 3.3 は face を閉空間を作らないように張り合わせてできる形状であれば、実は非多様体稜線や非多様体頂点が存在する場合でも成立する式であることが証明できたことになる。

ワイヤフレームモデルのオイラー式

式 3.19でワイヤフレームの場合を考えると、face, volume が存在しないことから、

$$f = r = V = Vh = Vc = Cc = 0\,. \tag{3.24}$$

したがって、連結成分の個数を s、独立なサイクルの個数を c と置いて式 3.24 を非多様体形状モデルの関係式に代入すると、

$$v - e = s - c \tag{3.25}$$

と書くことができる。この式は、Wilson の提案したワイヤフレームのオイラー式 3.4 と等価である。

胞体分割された 3-多様体

ロボットシミュレーションなどではしばしば空間を立方体のような単純な立体で分割したボクセル表現が用いられるが、このような形状は位相幾何学的には胞体分割された 3-多様体に相当する。ボクセル表現ではボクセル (volume) が貫通穴や空洞を持たず、また face に ring もないので、

$$Vh = Vc = 0\,,\ \ r = 0\,. \tag{3.26}$$

となる。よって、関係式はセルの個数を V として

$$v - e + f - V = s - h + Cc \tag{3.27}$$

となる。必要最低限のオイラー操作の個数は 6 種類である。

正則集合

正則集合 (r-set) はソリッドモデリングでよく用いられる分類であり、代表的な定義域である。正則集合には非多様体も含まれており、2-多様体よりも広い定義域を持つ。ここで、非多様体形状モデルの関係式に基づいて考えてみると、正則集合では 10 個の変数がすべて独立である。したがって、正則集合の関係式は非多様体形状モデルの関係式と同じになり、オイラー式は簡略化されない。

このことから、正則集合は、境界表現形状モデルの定義域としては位相的な性質があまりよくないということができる。正則集合のオイラー式を扱うためには、[Higashi90] や [Desaulniers92] のように、正則集合の位相要素の個数を直接扱うのではなく、仮想的な位相要素を導入して 2-多様体に帰着させる方法が有効だと思われる。

3.8 オイラー操作の実装

本節では、本論文で述べたオイラー操作を、位相要素を引数にとる位相操作関数として実装するときの仕様について述べる。入出力はできる限りデータ構造に依存させないことを考え、原則として v (vertex), e (edge), l (loop), f (face), s (shell), V (volume), C (complex) のみで構成するようにした。vertex, edge, face, volume については既に説明したが (図 2.10)、complex, shell, loop は、以下のようなものである (詳細は、図 4.10, 4.11 参照のこと。)

complex (C) 連結な形状。形状モデルを二つに分離するような操作を行なうと、complex が二つになる。

shell (s) volume の境界の連結成分。たとえば、volume に空洞が一つある場合には、volume 境界の連結成分が外殻と空洞との二つになるので、shell は二つになる。

loop (l) face の境界の連結成分。face に空洞 (ring) が一つある場合は、face 境界の連結成分は二つである。

ただし、オイラー操作によっては、face の裏か表の一方を指定しなければならないことがあり、その場合にはこれらの位相要素だけでは引数が記述できない。このときには便宜上、図 3.19 の radial-edge 構造に用いられている補助的な位相要素 face-use または edge-use を引数に用いることにした。face-use は face の裏または表に相当する補助位相要素であり、edge-use は face-use の境界位相要素を表している。(radial-edge 構造の詳細な説明については第４章で述べる)

ここで示した関数は実際に本論文で試作した非多様体形状モデリングシステムに実装されており、第５章と６章で述べられる形状処理に用いられている。

以下に関数を記述する際の規則について示す。

- 関数名が同じであっても引数のタイプが異なれば別の関数である。
- オイラー操作とその逆操作を対にして記述する。
- 引数のうち入力は ↓、出力は ↑ で示す。
- 状況に応じて指定する必要のあるものは () で、必ずしも指定してもしなくてもよいものは {} で示す。
- 入力された位相要素に矛盾があり操作が行えないときはエラーを返すものとする。

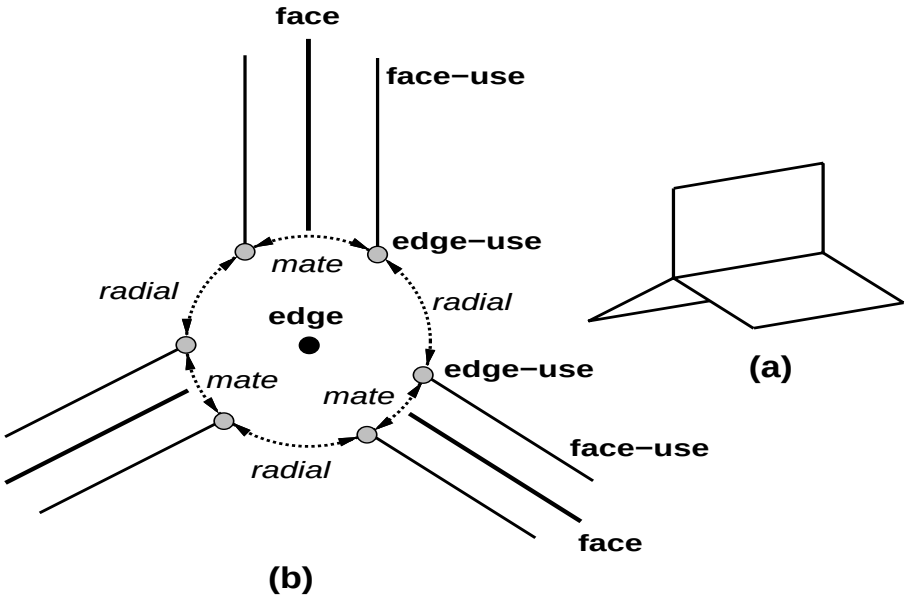


図 3.19: radial-edge 構造.

９種類のオイラー操作

まず、図 3.14 の９種類の操作とその逆操作を示す。

1. **make_vertex_complex とその逆操作：**

   mvC では、孤立点からなる形状モデルが新たに生成される。kvC では孤立点を消去するが、v が孤立点でないときはエラーとなる。

   - $mvC(new\_v \uparrow, new\_C \uparrow)$
   - $kvC(v \downarrow)$

2. **make_edge_vertex とその逆操作：**

   mev では、生成される edge が wire edge なのか、face の境界となる strut edge なのか、volume の内部の dangling edge なのかによって引数が異なる。また、edge が face 中の strut edge となる場合には face のループサイクルの再構成が必要となるので、そのための引数として、生成される strut edge から見て反時計回りのもっとも近い稜線を ccw_e として指定する必要がある。

   逆操作 kev では消去すべき vertex と edge を指定すればよいが、v が e 以外の edge に接続しているときはエラーとなる。

- $mev(v\downarrow, new\_v\uparrow, new\_e\uparrow)$ // a wire edge
  $mev(v\downarrow, f\downarrow, (ccw\_e\downarrow), new\_v\uparrow, new\_e\uparrow)$ // a strut edge in a face
  $mev(v\downarrow, V\downarrow, new\_v\uparrow, new\_e\uparrow)$ // a dangling edge in a volume
- $kev(v\downarrow, e\downarrow)$

3. **make_edge_Chole とその逆操作：**

   meC_h では 同じ complex に属する２頂点間に wire edge を張る。また、逆操作 keC_h では edge を消去するが、e が wire edge でないときや e の端点が開いているときはエラーとなる。

   - $meC_h(v1\downarrow, v2\downarrow, new\_e\uparrow)$
   - $keC_h(new\_e\downarrow)$

4. **make_face_kill_Chole とその逆操作：**

   $mfkC_h$ では、面のループサイクルの順に並べられた edge のリスト e_list を境界とする face を生成する。このとき、稜線周りの face の順序関係であるラジアルサイクルを再構成する必要があるが、そのためには生成される face に関して、向い合うのがどの face のどちら側なのかの情報が必要である。このことは、各稜線が２枚の face に共有される場合は自明であるが、３枚以上の面が存在する場合は引数として指定しなければならない。ここでは、向かい合う側の面を edge_use で指定することにする。逆操作 $kfmC_h$ では、指定された face を消去するが、face が volume の境界のときはエラーとなる 。

   - $mfkC_h((int)n\downarrow, e\_list[n]\downarrow, (edge\_use[n]\downarrow), new\_f\uparrow)$
   - $kfmC_h(f\downarrow)$

5. **make_face_Ccavity とその逆操作 ：**

   mfC_c では、face を定義することによって閉じた空の領域を作る。(ただし、４章で述べるデータ構造では、空の領域を陽に管理していないので、この操作は $mfkC_h$ と同等となる。しかし、データ構造によっては、空の領域を位相要素として陽に管理することもありうるので、ここでは区別した。) 一方、逆操作 kfC_c では、指定された face を消去するが、face が volume の境界のときや閉空間を構成していないときはエラーとなる。

   - $mfC_c((int)n\downarrow, e\_list[n]\downarrow, (edge\_use[n]\downarrow), new\_f\uparrow)$
   - $kfC_c(f\downarrow)$

6. **make_vertex_ring とその逆操作 ：**

   mvr では、face の内部に vertex を生成し、生成された vertex の属する loop を返す。また、逆操作 kvr では、face 中の vertex を消去するが、vertex が face 中の孤立点でないときはエラーとなる。

   - $mvr(f\downarrow, new\_v\uparrow, new\_l\uparrow)$
   - $kvr(v\downarrow)$

7. **make_volume_kill_Ccavity とその逆操作 ：**

   $mVkC_c$ では、face で囲まれた閉空間に volume を生成する。このとき、face のどちら側に volume を生成するのかを指定する必要がある。そこで、face の片面である face_use を引数として指定する。 また、逆操作 $kVmC_c$ では volume を消去して、空の３次元領域にする。

   - $mVkC_c(face\_use\downarrow, new\_V\uparrow)$
   - $kVmC_c(V\downarrow)$

8. **make_vertex_Vcavity とその逆操作 ：**

   mvV_c では、volume の内部に孤立 vertex を生成し、生成された vertex の属する shell を返す。また、kvV_c では、volume 中の vertex を消去するが、vertex が volume 中の孤立点でないときはエラーとなる。

   - $mvV_c(V\downarrow, new\_v\uparrow, new\_s\uparrow)$
   - $kvV_c(v\downarrow)$

9. **make_edge_Vhole とその逆操作：**

   meV_h では、２頂点を通り、volume を貫く edge を生成する。また、逆操作 keV_h では、指定された edge を消去するが、edge が volume 境界でない場合や端点が開いているときはエラーとなる。

   - $meV_h(V\downarrow, v1\downarrow, v2\downarrow, new\_e\uparrow)$
   - $keV_h(e\downarrow)$

併合・分割のためのオイラー操作

次に、図 3.18 に示したような、９種類のオイラー操作の組合せで実現可能ではあるが効率の点から用意した方が望ましい操作について述べる。ここでは、edge, face, volume

の分割併合を行う操作と、形状モデルの連結成分や loop, shell を分割併合するための６種類の操作について示す。なお、位相要素の分割併合のための操作は位相要素の個数の増減の上では９種類の操作に含めて考えることができるが、形状モデリングでは区別して扱われることが多い。例えば、$split\_edge$ は mev のバリエーションの一つであるが、これらは必要となる引数が異なるので、実装においては分けて考えることにする。

1. **edge の分割、併合操作 ：**

   $split\_edge$ は edge を二つに分割する操作である。引数として分割する稜線 e の端点 $survived\_v$ が指定されたときは、分割後もこの頂点のある側が稜線 e となり、他方が新たに生成された edge となる。また、逆操作の $merge\_edge$ では vertex を消去することによって edge を併合する。vertex に接続する二本の edge のうちどちらを残すかを指定したいときは $survived\_v$ を指定すれば、この頂点を持つ側 の edge が消去されずに残る。なお、消去される vertex に接続する edge が二本でないときはエラーとなる。

   - $split\_edge(e\downarrow, \{survived\_v\downarrow\}, new\_v\uparrow, new\_e\uparrow)$
   - $merge\_edge(\{survived\_v\downarrow\}, v\downarrow)$

2. **face の分割、併合操作：**

   $split\_face$ は、同一の loop に属する２頂点 v_1, v_2 間に edge を張ることによって face を分割する操作である。このときループサイクルを再構成する必要があるが、これらの頂点がこの loop に属する strut edge に接続する場合には 頂点周りの edge の順序を陽に指定するための引数が必要である。そこで、このような場合には、生成される edge から半時計回りに見て最も近い edge を ccw_e1, ccw_e2 として指定することにする。また、逆操作の $merge\_face$ は edge を消去して face を併合する操作である。edge の両側の face のうちどちらを残すかを指定したい場合には $survided\_f$ を指定する。なお、この操作では指定された edge が唯二つの面に共有されなければエラーとなる。

   - $split\_face(f\downarrow, v1\downarrow, v2\downarrow, (ccw\_e1\downarrow), (ccw\_e2\downarrow), new\_e\uparrow, new\_f\uparrow)$
   - $merge\_face(e\downarrow, \{survided\_f\downarrow\})$

3. **volume の分割、併合操作：**

   $split\_volume$ は volume の境界上の edge を境界とする face を生成することによって volume を分割する操作である。新たに生成される volume が face のどちら側にくるかを指定したい場合は $survived\_f$ を指定する。$mfkC_c$ のときと

同様、ラジアルサイクルを保持するために edge_use の指定が必要となる。また、$merge\_volume$ は 二つの volume の境界 face を消去することによって volume を併合する操作である。このとき、$survived\_V$ が指定されたときは、指定された側の volume が消去されずに残る。

- $split\_volume(V\downarrow, (int)n\downarrow, e\_list[n]\downarrow, (edge\_use[n]\downarrow), \{survived\_f\downarrow\}, new\_V\uparrow)$
- $merge\_volume(f\downarrow, \{survived\_V\downarrow\})$

4. **loop を分割、併合する操作：**

   $kemr$ は loop を edge で構成されるグラフとみたときにカットセットとなっている edge を消去して loop を生成する操作である。新たに生成される loop が edge のどちら側にくるかを指定したいときは $survived\_v$ を指定する。なお、edge がカットセットでないときはエラーとなる。また、逆操作である $mekr$ では異なる loop に属する vertex 間に edge を張ることによって loop を併合する操作である。残したい側の loop は $survived\_l$ で指定することができる。 vertex が同一 face の異なる loop に属していないときはエラーとなる。

   - $kemr(e\downarrow, \{survived\_v\downarrow\}, new\_l\uparrow)$
   - $mekr(v1\downarrow, v2\downarrow, \{survived\_l\downarrow\}, new\_e\uparrow)$

5. **shell の分割、併合操作：**

   $kemV_c$ は edge を消去することによって shell を分割する操作である。指定された edge を消去しても shell が二つに分割されないときはエラーとなる。$survived\_v$ を指定することによって edge のどちら側に新たな shell を生成するのかを指定することができる。また、$mekV_c$ では、同一の volume の異なる shell 間に edge を生成することによって shell を連結する。vertex が同一の volume の異なる shell に属していないときはエラーとなる。

   - $kemV_c(e\downarrow, \{survived\_v\downarrow\}, new\_s\uparrow)$
   - $mekV_c(v1\downarrow, v2\downarrow, \{survived\_s\downarrow\}, new\_e\uparrow)$

6. **complex の分割、併合操作：**

   $kemC$ は edge を消去して、一つの complex を二つの連結成分 (complex) に分割する操作である。どちら側を保持するかは $survived\_v$ で指定する。edge を消去しても二つに分割できないときはエラーとなる。また、$mekC$ は二つの異なる complex に属する vertex 間に edge を張ることによって、それらを連結する操作である。消去されない方の complex を指定することができる。

- $kemC(e\downarrow, \{survived\_v\downarrow\}, new\_C\uparrow)$
- $mekC(v1\downarrow, v2\downarrow, \{survived\_C\downarrow\}, new\_e\uparrow)$

3.9 まとめ

形状モデリングでは、位相構造を変更するための基本変形操作としてオイラー操作が非常に重要であるが、非多様体形状モデルではオイラー操作と同等の性質を持つ変形操作は知られていなかった。本研究では、非多様体形状モデルで成立する位相的な関係式を複体のオイラー・ポアンカレの式を利用することによって導出し、非多様体形状モデルのためのオイラー操作を求めた。

本章の要点は以下の通りである。

1. 非多様体形状モデルにおける位相的な関係式を複体のオイラー・ポアンカレの式に基づいて導出する方法を提案した。
2. 導出した式に基づき、非多様体形状モデルでは９種類のオイラー操作が理論上必要最小限の位相変形操作であることを証明した。
3. 導出した式に基づき、非多様体形状モデリングのためのオイラー操作群を求めた。
4. 非多様体形状モデルの関係式に制約を加えることによって、代表的な定義域の形状群についての位相的な関係式が導出できることを示した。
5. オイラー操作を実装するための関数を設計した。

Chapter 4

非多様体形状モデルの内部表現

本章では、非多様体形状モデルを表現するための内部表現について論じる。形状モデリングシステムの実装においては、データ構造の管理がしやすく、実用的に十分効率的に動作することが重要となる。

非多様体形状モデルのデータ構造の研究としては Weiler の提案した radial-edge 構造がよく知られている。しかし、Weiler の提案したデータ構造に基づいて形状処理システムを構築した場合、３次元位相要素 region の整合性を管理するための処理に時間がかかり、またサーフェスモデルにソリッドの構造を持ち込んでいるためにサーフェスに関する処理が重くなるという問題がある。そこで、本論文では、region とは異なる定義をもつ位相要素 volume を用いた新しい階層構造を導入した内部表現を用いることでこの問題を解決する。

4.1 形状モデルのデータ構造

境界表現形状モデルでは、vertex, edge, face といった幾何形状の構成要素や、連結した face の集合である shell などをノードとしたグラフ構造によって幾何形状を表現する。このように、幾何形状を計算機データとして保持するための表現形式をデータ構造と呼ぶ。データ構造の設計は、形状モデリングシステムを実装する上で最も基本となるものである。

従来の境界表現ソリッドモデルでは、Baumgart の提案した、winged-edge 構造 [Baumgart75] が主として採用されてきた。このデータ構造では、図 4.1 に示すように、矢印で示した edge に関して、隣接する face, edge, vertex を、立体を外側から見たときの左右の区別を付けて保持する。また、位相データの探索が高速にできる half-edge 構造 [Mantyla88] や、winged-edge 構造と等価な位相情報を持つ vertex-edge 構造や face-edge 構造 [Weiler85] などのデータ構造も提案されている。しかし、これらのデータ構造は 2-多様体ソリッドモデルを定義域としたデータ構造であり、すべての稜線が二枚の面の共有され、すべての面はソリッドを内部と外部を分ける境界となるという性質に基づいている。そのため、非多様体形状の表現には適していない。

非多様体形状モデルのデータ構造の研究としては、Weiler の提案した radial-edge 構造 [Weiler86b] がよく知られている。この研究では、非多様体形状モデルを表現するために補助的な位相要素を導入し、ワイヤフレーム、サーフェス、ソリッドを統一的に表現するためのデータ構造を提案した。また、非多様体形状を vertex-base で表現したデータ構造も Gursoz らによって提案されている [Gursoz88]。

しかし、これらのデータ構造は実用上いくつか問題点があり、必ずしも実用性まで考慮して設計されたものではないと思われる。現実に非多様体形状モデリングシステムを実装するためには、様々な形状処理を行なっていく上での効率に関して考察を行ない、実用に適したデータ構造を考える必要がある。

4.1.1 radial-edge 構造

まず、代表的な非多様体データ構造である radial-edge 構造について説明する。

図 4.2 は、radial-edge 構造における位相要素の階層構造を示している [Weiler86b]。この図において、shell は連結な位相要素の集合、region は面に囲まれた閉空間を表す位相要素である。非多様体形状モデルでは 2-多様体ソリッドと異なり、面に囲まれた閉空間が一つであるとは限らないため、region を陽に扱っている。図 4.3 は region の例を

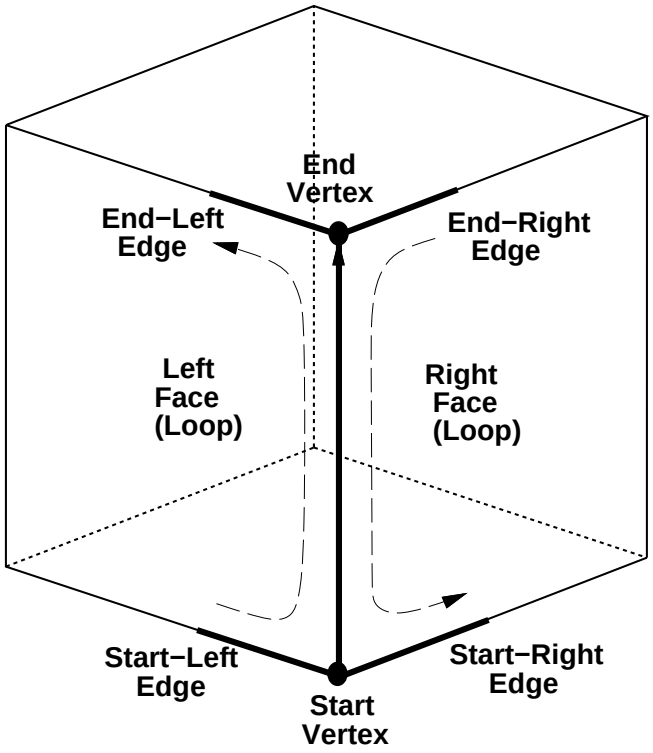


図 4.1: winged-edge 構造で保持されるデータ.

示している。六面体は、空間を六面体の内部と外部に分割するので、それぞれが region として管理される。

これらの位相要素の間には、階層構造とは別に順序関係も存在する。３次元形状モデルにおける順序関係には、図 4.4 に示されたように、ループサイクル、ラジアルサイクル、ディスクサイクルの三つがある。ループサイクルは、face の境界位相要素を face の内部からみて半時計回りに回る順序、ラジアルサイクルは、edge 回りの face の順序、ディスクサイクルは、頂点の回りの face に沿って一周するような順序である。

非多様体形状モデルにおいては、これらの階層構造と順序関係の記述は２-多様体の場合に比べて複雑になる。２-多様体のためのデータ構造である winged-edge 構造では、階層構造において上位の位相要素と下位の位相要素との関係は１対多の関係で記述できるので、単純なリスト構造で位相的な関係が記述できた。しかし、非多様体形状モデルでは、非多様体稜線を許すので、edge の上位の位相要素となる face の個数は二つだけとは限らない。また、vertex においても、ワイヤフレームの場合を考えると、連結するすべての edge を保持しなければ vertex と上位の位相要素との双方向の関係は記述できない。したがって、上位の位相要素と下位の位相要素の関係は一般に多対多

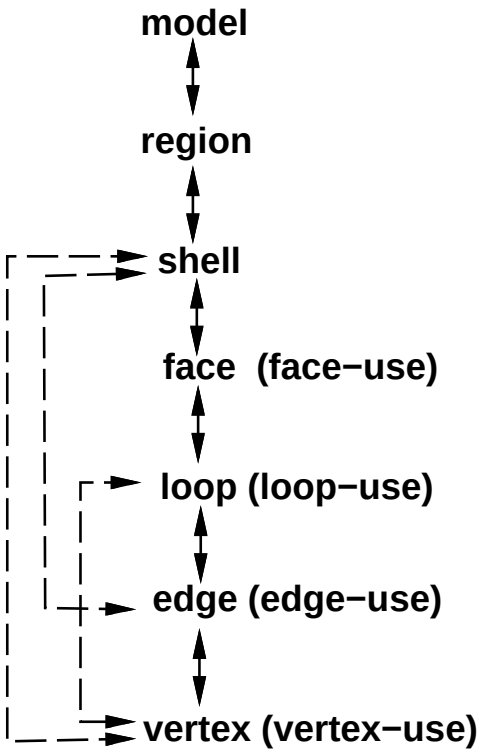


図 4.2: radial-edge 構造の階層構造.

となり、単純なリスト構造では双方向の関係が表現できない。

このような問題を解決する方法として、Weiler は補助的な位相要素を導入し、非多様体形状の位相構造を固定長のデータの集まりで表現する方法を示した。radial-edge 構造では、face, loop, edge, vertex の補助的な位相要素として face-use, loop-use, edge-use, vertex-use を導入し、隣接する位相要素の関係付けをこれらの use を介して記述した。図 4.5 では、図 4.5(a) の形状の use による表現を図 4.5(b) に示している。face は面の裏表のそれぞれで volume の境界と成りうる。そこで、裏表に応じて 二つの face-use を生成すれば、face-use が唯一つの上位の位相要素を持つようにできる。この場合、face は、 二つの face-use の集まりとなる。また、loop-use は face-use に対応して生成される。同様にして、edge の上位の位相要素の個数に応じて edge-use を生成すれば、edge-use が唯一つの上位の位相要素に連結するようにできる。この場合、edge は edge-use の集合を保持することになる。veretx についても同様に、上位の位相要素の個数に応じて vertex-use を生成すればよい。このような補助的な位相要素を導入すると、補助的な位相要素は唯一つの上位の位相要素と関係付けられるので、固定長のデータによるリスト構造で隣接関係が表現できる [Weiler86]。

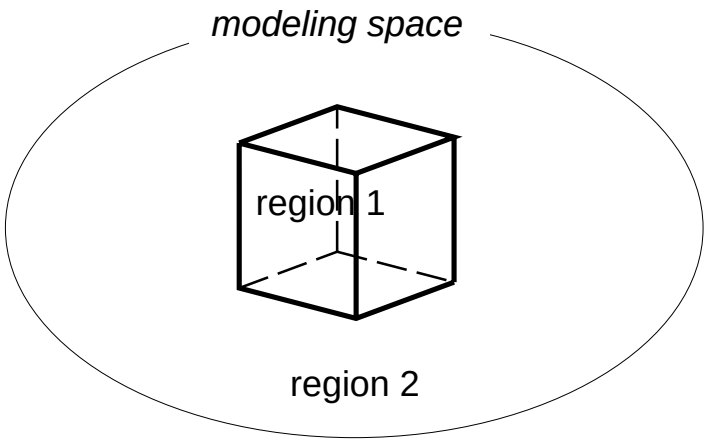


図 4.3: radial-edge 構造の位相要素：region.

サイクルもこのような補助的な位相要素の関係として記述することができる。ループサイクルについては、face の境界上の edge に関して edge-use が唯一つの loop-use の境界となるように生成されるので、loop-use を反時計回りに回るための次の edge-use を *counter-clockwise-edge-use* として各 edge-use が保持すればループサイクルを表現することができる。

また、補助位相要素を用いたラジアルサイクルの表現は 図 4.5 に示した通りである。この例では、形状 (a) におけるラジアルサイクルは関係 *mate* と *radial* を用いて (b) のように表現される。ラジアルサイクルは edge 回りの face の順序であるが、edge-use の *mate* と *radial* を交互に辿っていくことにより、edge 周りの face を一周することができる。すなわち、各 edge-use が *counter-clockwise-edge-use*、*mate*、*radial* という３通りのポインタを保持することによってループサイクルとラジアルサイクルが表現できる。

4.1.2 vertex-based 構造

非多様体形状モデルの順序関係を vertex-use 間の関係として保持する vertex-based データ構造 [Gursoz88] も提案されているので、ここで説明しておく。

このデータ構造では、図 4.6(a) のようなサーフェスの頂点接触を表現するために、新たな位相要素として disk (ディスクサイクルを構成する面)、及び zone (図で $Z_1, Z_2, ...$ で示されたサーフェスに挟まれた領域) を導入することにより、vertex, disk, zone の関

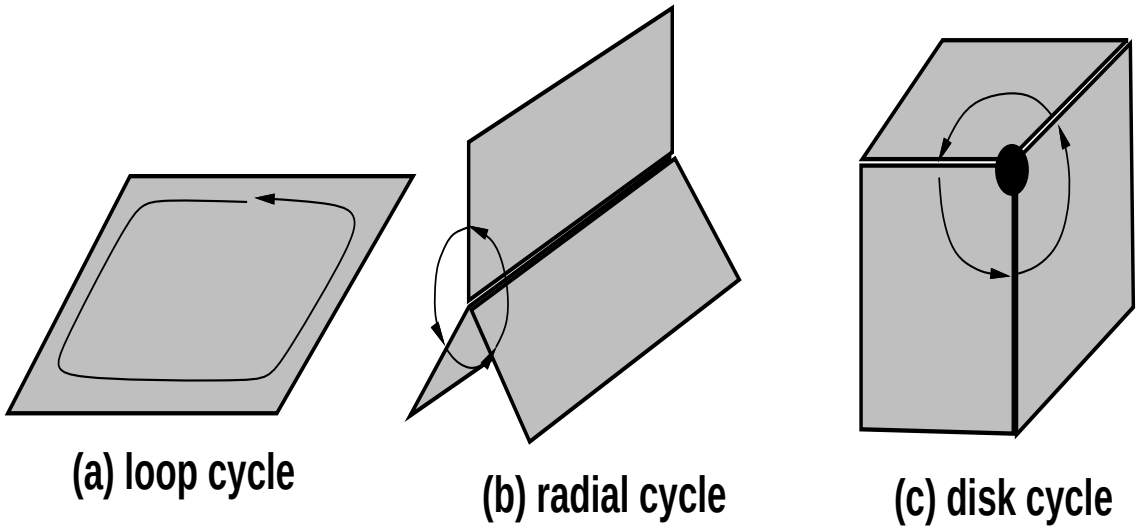


図 4.4: ３次元形状に現れる順序関係.

係を図 4.6(b) のような階層構造で表現した。このデータ構造の利点は、点接触している場合でも位相構造の探索によって、face に囲まれた閉空間が検出できることである。

しかし、実用的な観点から考えると、多くの場合、点接触のような例外的なケースが生じた場合だけ幾何計算でディスク間の関係を算出した方が、データ構造の整合性を保持するための不必要な計算をしなくて済む。図 4.6(b) のようなデータ構造を用いる場合、ワイヤフレームモデリングやサーフェスモデリングで edge や face を生成する度に、生成される位相要素がどの zone に属するかを計算しなければデータ構造の整合性が維持できないという問題点があり、モデリングの効率が非常に悪くなる。

したがって、本研究の場合のように形状処理の効率を重視する立場からは、disk や zone を陽に保持することは避けるべきだと思われる。

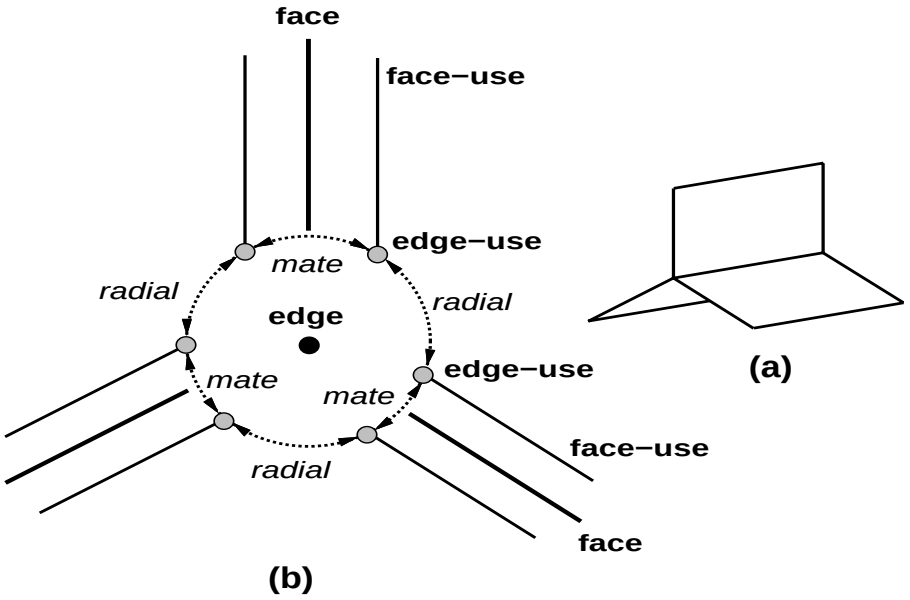


図 4.5: 補助位相要素によるラジアルサイクルの表現.

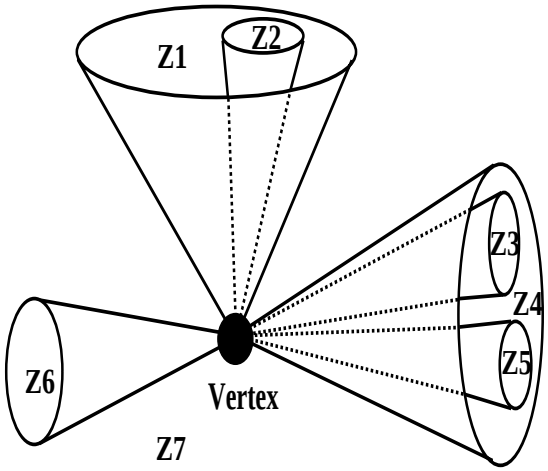
4.2 従来の非多様体データ構造の問題点

非多様体形状モデルのデータ構造については、既に Weiler の研究があり、具体的なデータ構造が提案されている。しかしながら、このデータ構造によって形状モデラを実装した場合、処理効率が悪くなり、また、データ構造上、ソリッドの内部と外部が同等に扱われるため処理が複雑になるといった問題がある。小林はこのような問題を避けるために、対象をサーフェスモデルに限定することにより、3次元位相要素である region を持たない非多様体形状モデラ *surmon* を提案しているが [Kobayashi89]、立体形状を扱おうとするならば、region に相当する3次元位相要素が必要である。

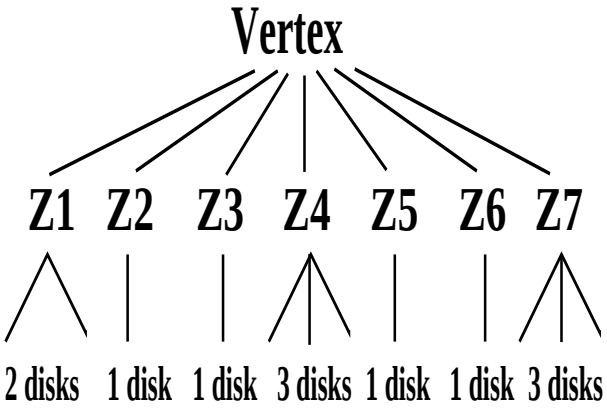
radial-edge 構造で生じる問題の原因は、face に囲まれた閉空間である region を基本的な位相要素として管理している点にあると思われる。region を内部表現に用いたデータ構造の問題点として、以下のようなことが挙げられる。

1. region の整合性を保持するための効率上のコストが大きい。

region を幾何形状の基本的な構成要素とした場合には、データ構造の整合性を保持するために、連結な face が閉じているか開いているかを監視していることが必要であ



(a) A vertex shared by disks.



(b) Structure

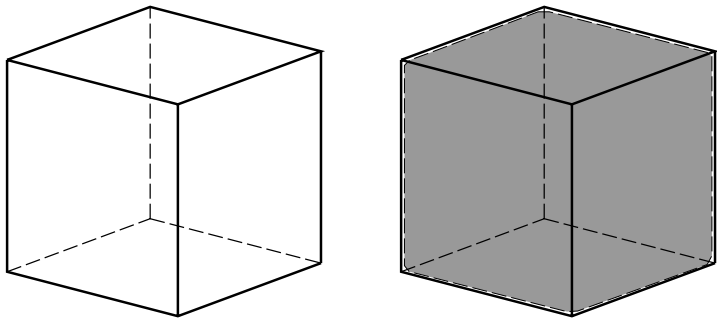
図 4.6: vertex 回りの位相構造 (vertex-based 構造).

る。なぜなら、face が閉空間を構成するかしないかが位相要素 region の有無に影響し、データ構造が大きく変わるからである。そのため、サーフェスモデリングであっても、face を生成する度に閉空間ができるのかどうかを確認しなければデータ構造の整合性は維持できないことになる。この点については、Weiler 自身も論文中で指摘しているが [Weiler86]、face 生成という局所的な操作を記述するために、閉空間探索という大局的な操作が不可欠ということになり、効率が大変悪くなる。

閉空間の情報を必要としないサーフェスモデリングなどでは region は管理せず、必要に応じて閉空間探索を行なってソリッド化するようなデータ構造を採用すべきである。しかし、そのような実装を行なった場合には region の定義に反することになり、データ構造の整合性が維持されないことになる。

２．表現できる形状の範囲が本論文の定義域よりも狭い。

region を基本的な位相要素とすることにより、形状モデルの定義域の問題も生じる。３次元ユークリッド空間で異なった点集合に対応する二つの形状モデルは、データ構造上区別できることが必要である。しかし、region による表現では、異なった点集合でありながら、データ構造上、区別できない場合が存在する。図 4.7にその例を示す。(a) の形状は面の集合で構成される２次元の点集合であり、(b) の形状は中身の詰まった３次元の点集合である。形状を点集合と考えるとき、両者は明らかに異なる形状であるが、region を位相要素とすると、形状 (a)(b) はどちらも同じデータ構造となってしまい、形状モデルとしては区別することができない。



(a) 2D point−set (b) 3D point−set

図 4.7: 面の集合（２次元点集合）とソリッドの立方体（３次元点集合）.

３．データ構造上、ソリッドの内部と外部が同等に扱われる。

region によってソリッドを表現した場合、図 4.3 のように、二つの region が生成される。一方が立体の内部、他方が外部の無限領域を示している。データ構造上はこれらはともに region として管理される。いいかえれば、このデータ構造だけからではどちらが立体の内部かわからない。そのため、立体が占める領域を表現するためには、region に立体の内部であることを示すフラグを用いることなどが必要となる。しかし、このような位相的にみて本質的な性質を付加情報によって表現することは適切とは思えず、また付加情報の管理が形状処理の最も低レベルな部分で必要となり、形状処理が複雑になる。

4.3 本研究の方針

4.3.1 位相要素 volume を用いた内部表現

region とはモデリングを空間分割と捉えた場合に用いられる位相要素である。すなわち、始めに実体の詰まった無限空間があり、それらを分割していったときの個々の部分空間が region となる。このようなモデルを考える限り、サーフェスモデリングの場合でも、新たに定義した面が空間を分割しないかどうかを監視していることが必要となる。

そこで本研究においては、このような考えはとらず、形状モデリングを何もない空間に位相要素を配置していくものと考え、位相要素全体の集合が実体を表現するという考え方をとる。そのために、region の代わりに位相要素 volume を考え、図 4.8 に示したような位相要素 vertex, edge, face, volume が空間に占める点集合の和として実体が表現されるものとする。この考え方は、本論文の第 2 章で述べた非多様体形状モデルの定義に沿ったものである。

3 次元位相要素 volume とは、立体領域を表現するための位相要素である。この位相要素は region とは異なり、立体の占める領域を表現しようとする場合にのみ生成される。図 4.9 を例にとって説明する。図 4.9(a) は、volume の生成されていない形状モデル、図 4.9(b) は、volume の生成された形状モデルを示している。形状 (a) は単に連結した face の集まりなので、実体は 2 次元の点集合である。一方、形状 (b) は、面に囲まれた閉空間の内部に 3 次元位相要素 volume が定義されているので、実体は立方体に相当する 3 次元の点集合である。このように、volume の有無によって 2 次元点集合と 3 次元点集合を表現し分けることができるので、face が閉空間を構成する場合でも、必ずしもソリッドの構造になるとは限らない。

内部表現という観点からいうと、volume が生成されない場合、face が閉じているかどうかといった情報はデータ構造に反映されず、また、ソリッドモデルの本質的性質である、face のどちら側が実体か、という情報を管理しないので、位相データの扱いが簡単になる。一方、図 4.9(b) は中が詰まっているので、ソリッドの構造となり、個々の face は volume の境界となる。したがって、face のどちらの側が実体かという情報を持つことになり、ソリッドモデルと同等の位相情報が管理される。

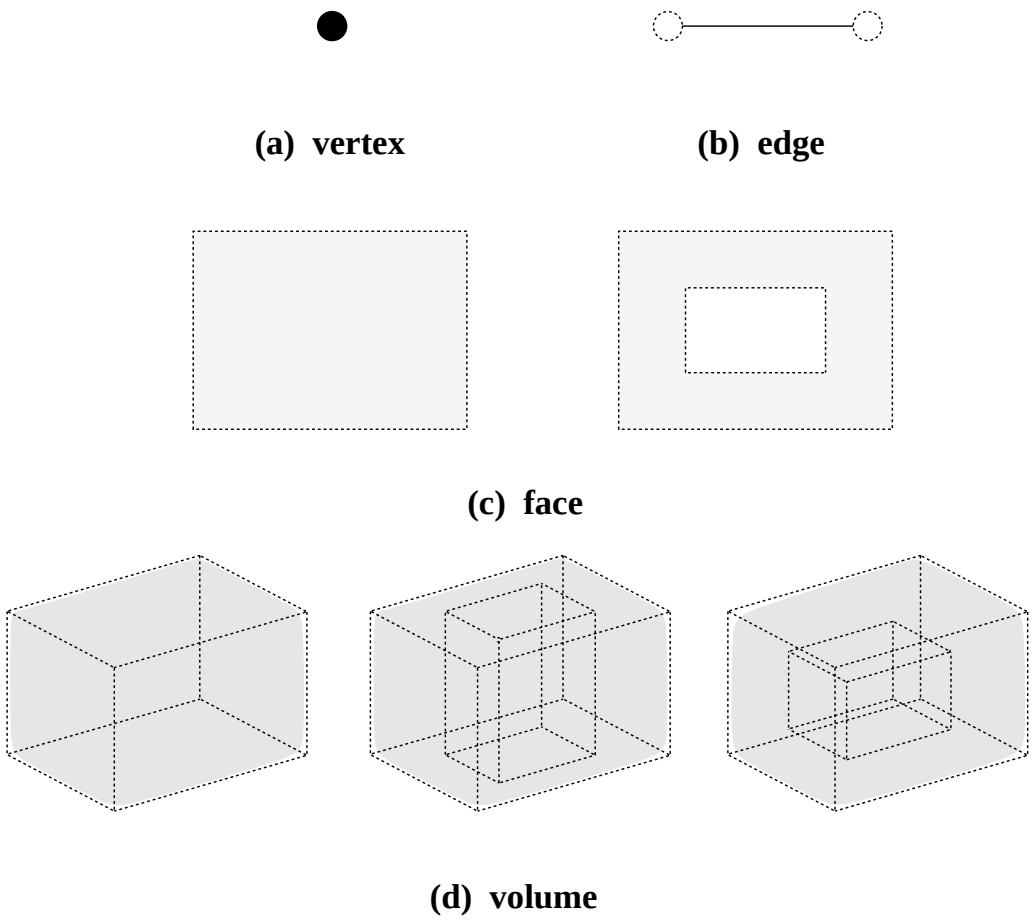


図 4.8: 位相要素 vertex, edge, face, volume.

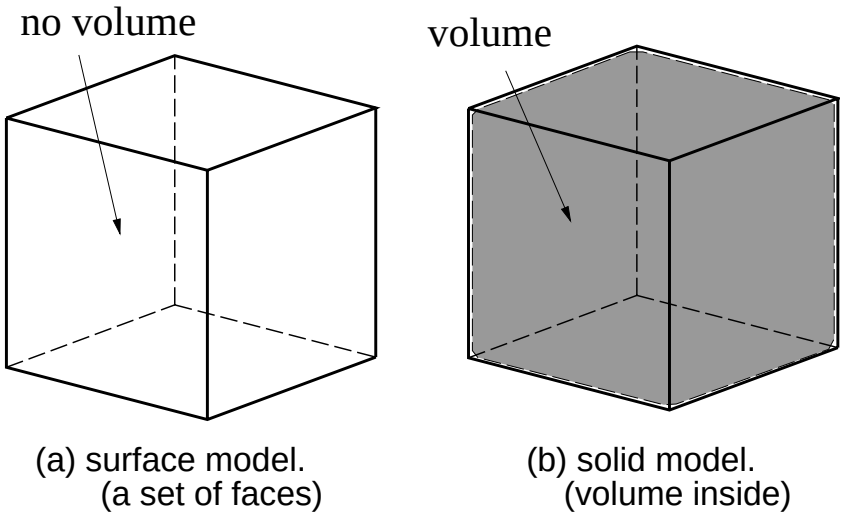


図 4.9: volume の有無による 2 次元点集合と 3 次元点集合の区別.

4.3.2 volume を用いる利点

このようなデータ構造をとることによって、前節で述べた問題点は解決できる。すなわち、

1. volume を用いたデータ構造では、閉空間が生成されるかどうかは陽には管理されないので、サーフェスモデリングにおいて face が生成されるたびに閉空間探索をする必要がない。したがって、サーフェスモデルを用いたシステムの実情に近くなる。

   一方、region の場合のように閉空間を直ちにソリッドの構造にしたい場合には volume を定義することによって、region を内部表現に用いた場合と同等の処理ができる。本データ構造では、閉空間を volume の有無によってソリッド領域として扱うかどうかを選択的に決めることができるので、応用システムに応じた効率的な形状処理が可能となる。

2. 形状モデルの表現する点集合を考えると、volume に基づいたデータ構造では、閉じた 2 次元点集合と 3 次元点集合を位相表現上区別することができる。region を用いたデータ構造では、このような区別はできなかった。

3. volume は実体のある部分にのみ生成されるので、region のように一つの立方体の内外に応じて二つ生成されるということがなく、実体部分をフラグをつけて区別する必要がない。

4.4 内部表現として管理される位相要素

本研究では、位相要素 volume を用いた内部表現を用いることについて述べた。本節では、volume 以外の位相要素について述べ、本研究で実装した非多様体形状モデルの内部構造を示す。

4.4.1 連結位相要素

本データ構造では、非多様体形状を vertex, edge, face, volume の集合によって表現する。これらの定義については既に示した通りである。ただし、face と volume については空洞を許しているので、連結な境界位相要素を管理することが必要である。また、形状モデル全体としても連結な位相要素の集合が管理できていることが望ましい。そこで、これらを管理するために、位相要素 $shell$, $loop$, $complex$ を導入する。

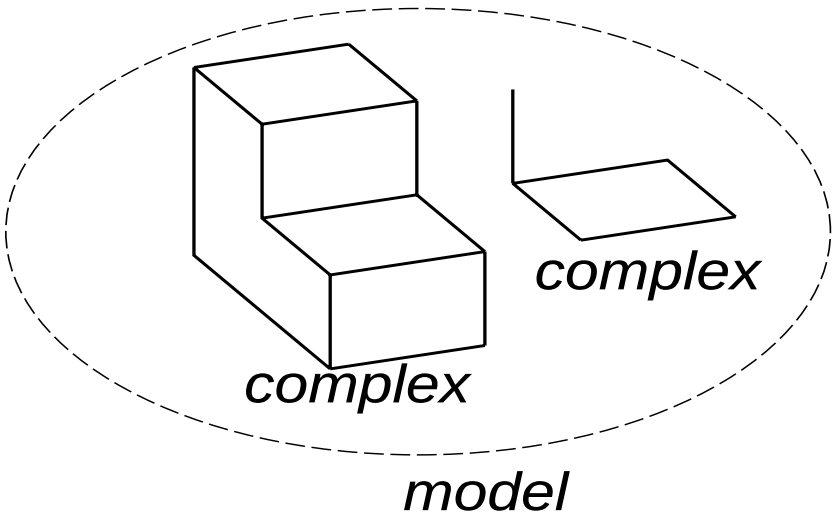


図 4.10: 位相要素：$model$ と $complex$.

complex

まず、連結な位相要素の集合を管理するために $complex$ を導入する。集合演算で形状モデルが分断された場合や、複数の連結成分を同一の形状モデルとして扱いたい場合

には、*complex* は複数となる。また、*complex* の集合を示す位相要素を *model* とする。図 4.10 に二つの *complex* から成る *model* を示す。

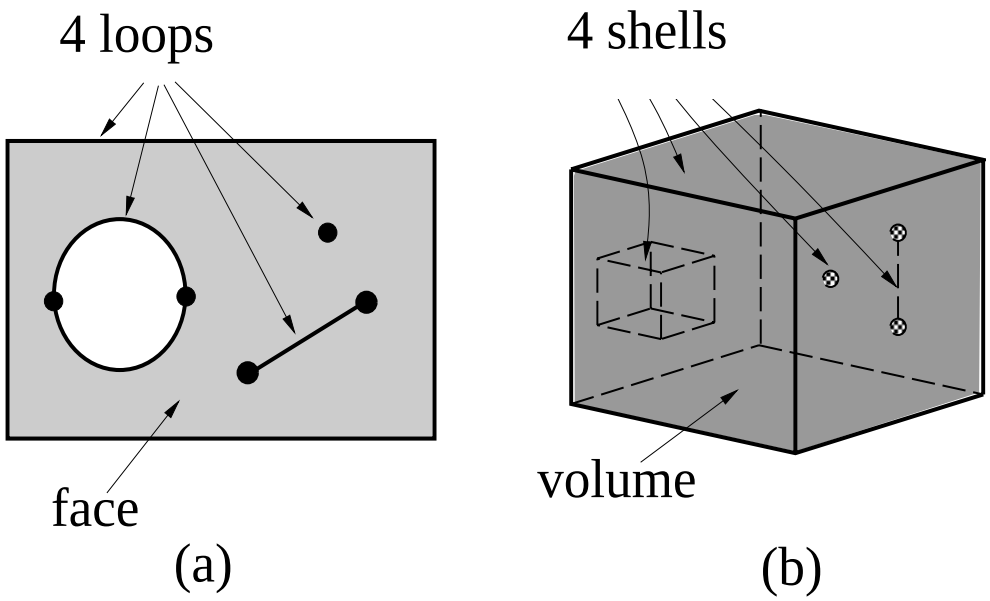


図 4.11: volume, face の空洞の表現.

shell と loop

face, volume に空洞を許す場合、これらの境界位相要素が非連結となる場合を考えなければならない。図 4.11にこれらの例を示す。

図 4.11(a) は空洞を持つ face の例であるが、この face の境界を構成しているのは、外側のループの他に、一つの閉ループ、閉じていない edge、孤立した vertex である。そこで、*loop* という位相要素を導入し、これらの連結成分の管理を行う。face は少なくとも一つの *loop* を持ち、*loop* のうち少なくとも一つは edge の閉路を持っている。

また、volume についても、図 4.11(b) のように空洞を持つ形状では、境界が複数の連結成分から構成される。この例では、孤立 edge、孤立 vertex を含めて、４個の連結成分から境界が構成されている。これらを *shell* として管理する。したがって、 volume は少なくとも一つの *shell* を持ち、また *shell* のうち少なくとも一つは閉空間を構成する。

4.4.2 位相要素の定義

以下に、非多様体形状モデルで管理される位相要素とその意味を示す。

model モデリング空間のすべての vertex, edge, face, volume の集合。

complex 連結な vertex, edge, face, volume の集合。

volume 3次元の実体を表す位相要素。境界を含まない、有界で連結な3次元領域。有限個の貫通穴や空洞があってもよい。

shell volume の境界の連結成分を表す位相要素。volume の外郭の shell は、 face で囲まれる閉空間を持たなければならないが、それ以外の shell は閉空間を構成しなくてもよい。

face 2次元の面を表現するための位相要素。境界を含まない有界、連結な2次元の領域。有限個の穴があってもよい。向き付け可能（裏と表の区別があること）であることが必要。

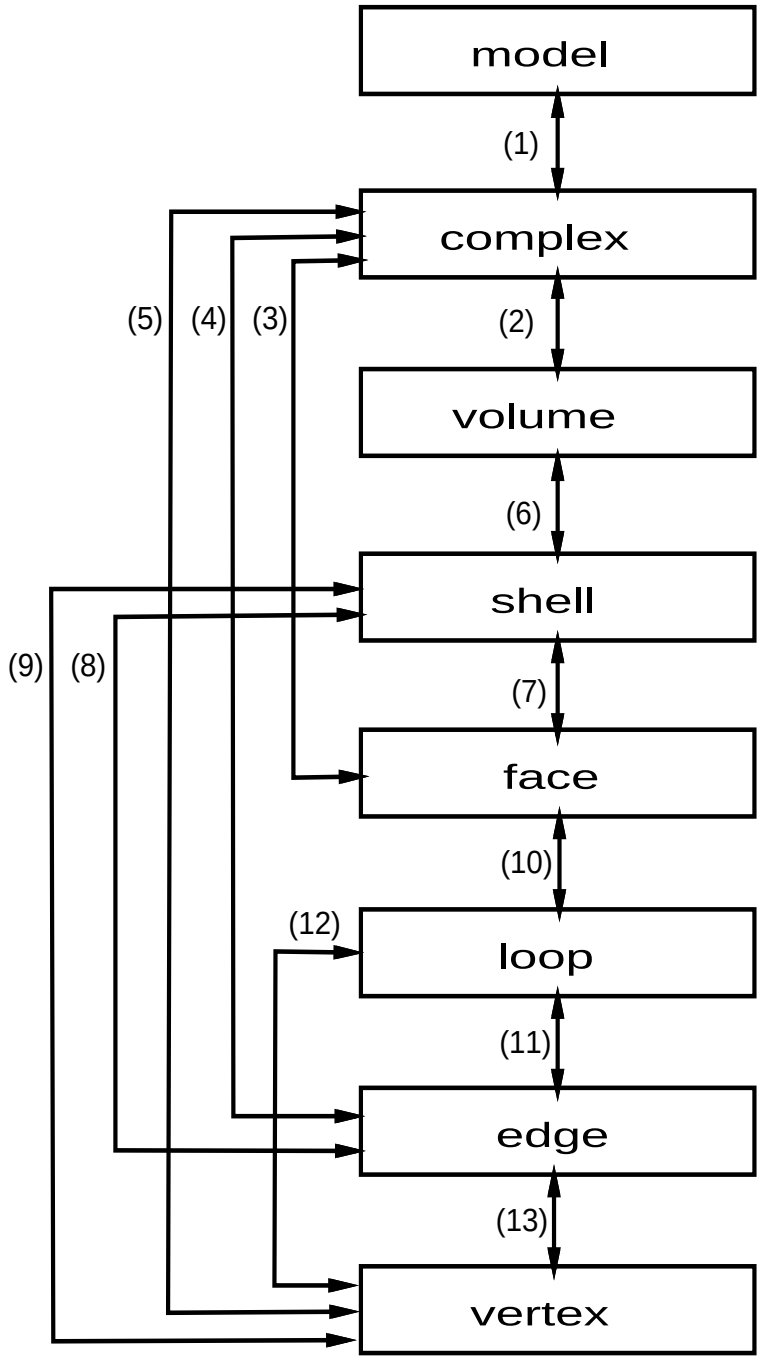
loop face の境界の連結成分を示す。face の外郭となる loop は、edge の閉路を含む。また、loop には向きが定義され、外殻ループでは face の法線に関して反時計回り、それ以外は時計回りとする。

edge 1次元の線分を表現するための位相要素。両端点を含まない有界、連結な線分。

vertex 0次元の点を表現するための位相要素。

4.4.3 階層構造

非多様体形状モデルの位相要素、volume, shell, face, loop, edge, vertex の関係を考えると、これらは、低次元の位相要素が高次元の位相要素の境界位相要素となっている。したがって、位相要素間の隣接関係は図 4.12のような階層構造で表現することができる。この図で、双方向の矢印で示したのは、境界や連結成分の要素となりうることを示している。たとえば、shell の構成要素は図 4.11(b) の空洞を持つ立体の例で示したように、face 、孤立 edge、または孤立 vertex となるので、図 4.12 の階層構造ではこれらの間に双方向の矢印 (7) (8) (9) が示されている。非多様体形状モデルでは、ワイヤフレームやサーフェスモデルなど様々な形状の混在を扱うため、位相要素間の隣接関係の種類は図に示した 13 通りとなる。



.

図 4.12: 位相要素間の階層構造.

位相要素間の関係の意味

以下に図 4.12の階層構造のそれぞれの関係の意味を示す。番号は図中の関係を示す番号に対応する。

(1) complex の集合を保持する。

(2),(3),(4),(5) complex を構成する位相要素の集合を保持する。関係付けられる位相要素は、volume, face, edge, vertex のうち、他の位相要素の境界となっていないもの（位相幾何学的にいえば境界輪体ではないもの）である。ソリッドの部分は、関係 (2) のように complex を volume と関係付けることによって表現する。また、(3) のように complex が直接 face と関係付けられるときは、その face がサーフェス部分であることを意味する。同様に、(4) のように edge と直接関係付けられるときはワイヤフレームの edge を、(5) のように vertex と直接関係付けられるときは、唯一つの vertex で構成される complex を表現する。すなわち、complex の下位にくる位相要素に応じて、ソリッド、サーフェス、ワイヤフレーム、孤立点が表現し分けられる。

(6) volume の境界連結成分である shell の集合を保持する。

(7),(8),(9) shell を構成する位相要素の集合を保持する。(7) のように、shell の下位となる face は、ソリッドの境界上にあることを意味する。また、(8) のように、shell が edge と関係付けられるときは、volume の占める空間中のワイヤ edge となる。volume 中に存在する孤立 vertex は (9) の関係で表現される。

(10) face の境界連結成分である loop の集合を保持する。

(11),(12) loop を構成する位相要素を保持する。(11) のように edge と関係付けられる場合は face の境界の edge の閉路または開路となる。(12) のように、vertex と関係付けられる場合は face 中の孤立 vertex である。

(13) edge の両端点の vertex を表現する。

図 4.13 は、ワイヤフレーム、サーフェス、ソリッドが混在した形状の階層構造を示している。この例は連結した形状モデルなので、complex は一つである。そして complex の下位には、ソリッド部分、サーフェス部分、ワイヤフレーム部分に応じて、それぞれ V_1, f_1, e_1 がくる。さらに、volume, face, edge, vertex 間の隣接関係を示すとこの図のような階層構造が生成できる。

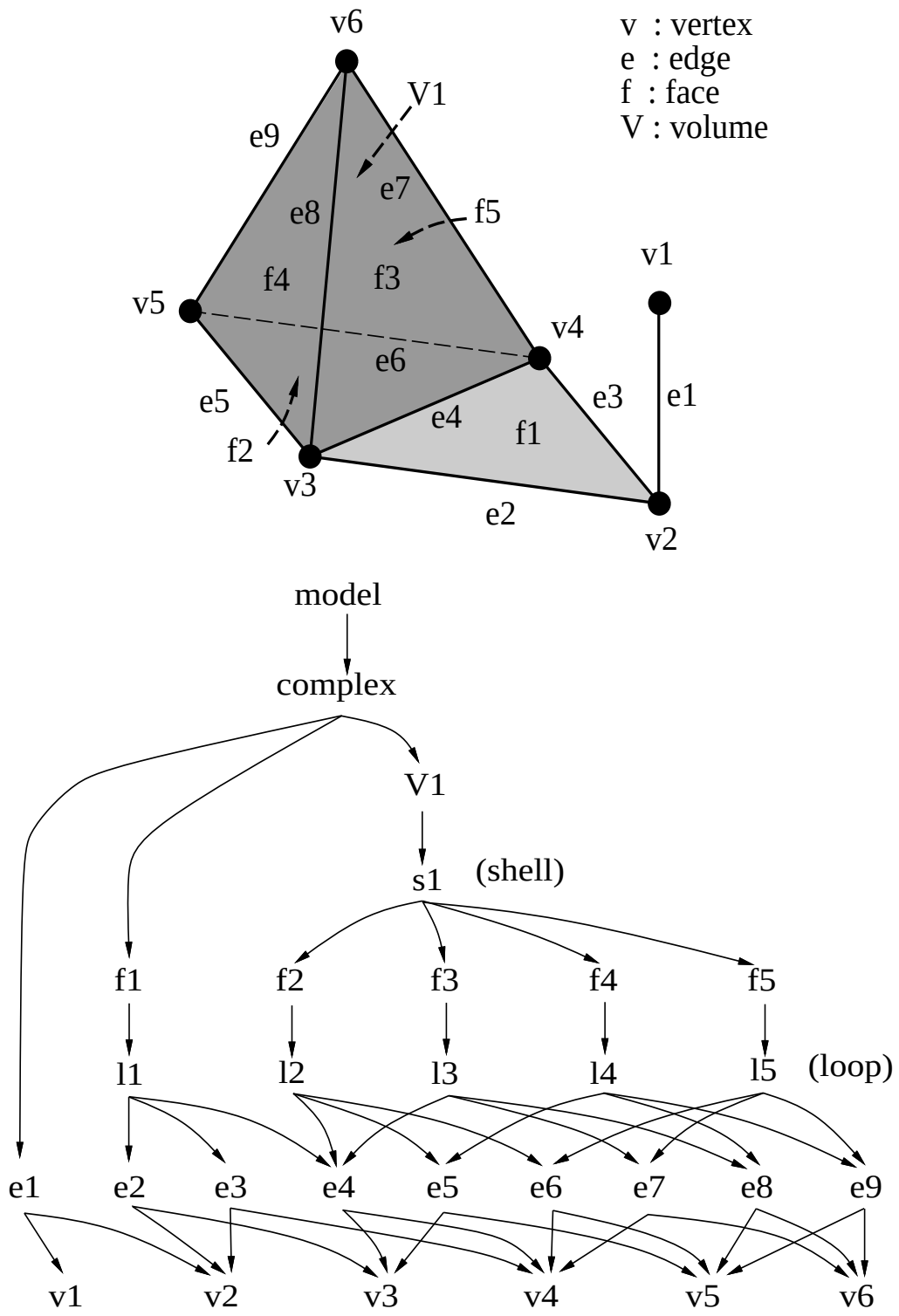


図 4.13: ワイヤフレーム、サーフェス、ソリッドが混在した形状の階層構造.

4.5 データ構造の設計

本節では、これまでの議論を踏まえて設計された非多様体形状モデルの具体的なデータ構造について示す。このデータ構造に基づいた非多様体形状モデラは C++ 言語を用いて実装され、本論文の例題で用いられている。その有効性については第５, ６章で示されており、形状処理において十分実用的な処理速度が得られている。

4.5.1 データ構造の構成

データ構造の記述法

ここではデータ構造を C または C++ 言語で用いられる構造体として記述する。まず、共通の事柄について述べておく。

- 位相要素 E が、下位の位相要素の集合 $\{e_1, e_2, ..., e_n\}$ を保持するとき、図 4.14 のようなリスト構造によって表現する。すなわち、上位の位相要素 E はリストの最初と最後の下位要素へのポインタを $first$、$last$ として持ち、また下位の位相要素 e においては、リストの次、及び一つ前の要素へのポインタとして、$next$、$previous$ を持つ。なお、$e_1, e_2, ...$ の順序は任意であり、順序自体に意味はない。
- 上位の位相要素へのポインタは、$upper$ と記述する。
- 各位相要素は、その種類を示すフラグ $type$ を持ち、同一種類の位相要素では同じ値をとる。$type$ にはあらかじめ定義された定数が位相要素の生成時に代入される。$type$ は、位相要素がどのように利用されているかを調べるときに用いられる。たとえば、edge は volume や face の境界、またはワイヤフレームの edge として利用されるが、$upper$ で記述された上位の位相要素の $type$ を参照すればどのように利用されているかを知ることができる。
- 非多様体形状モデルの階層構造では、上位または下位の位相要素の種類が必ずしも一通りに決まらないことがある。そこで、位相要素の種類が不定のときは、ワイルドカード的な位相要素として、ELEMENT というタイプを用いてデータ構造を記述する。実際に何が代入されたかは $type$ を参照することによって知ることができる。
- geometry についてはここでは詳細に述べず、単に geometry へのポインタを示す GEOMETRY として記述する。

- 形状モデリングを様々なアプリケーションでもちいるときは、属性データを持たせることが多い。そこでは属性へのポインタを ATTIBUTE として記述する。
- 形状モデリングのためのアルゴリズムを実装する際には、位相要素に一時的なマークをつけておくと便利なことが多い。そこで、作業領域として、integer の領域 $work$ を持たせる。

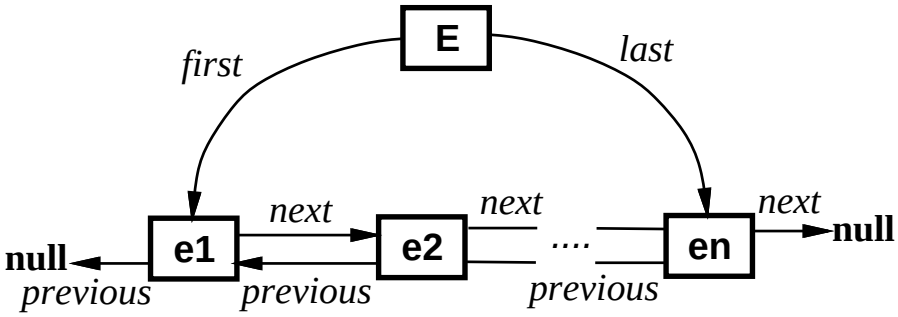


図 4.14: リストの表現.

管理される位相要素

本システムにおいても radial-edge 構造 [Weiler86] と同様、 face, loop, edge, vertex に関して補助位相要素を導入する。ただし、ここでは、補助位相要素は face-use, loop-use, edge-use, vertex-use という名前ではなく、co-face, co-loop, co-edge, co-vertex と呼んでいる。

したがって、位相的な関係は model, complex, volume, shell, co-face, co-loop, co-edge, co-vertex の間の関係として記述でき、それぞれの上位の位相要素は唯一つである。また、位相要素 face, edge, vertex は補助位相要素の集合の管理と geometry の記述に用いられる。

4.5.2 データ構造の定義

以下に、非多様体形状モデルの位相を表現するためのデータ構造を述べる。

ROOT

ROOT は 生成された model の集合を保持しておくために用いられる。

```
struct ROOT {
   int               type;         /* 生成時に１が代入される */
   int               work;         /* 作業領域 */
   ATTRIBUTE*        attribute;    /* 属性 */
   MODEL*            first;        /* model の集合 */
   MODEL*            last;
};
```

MODEL

MODEL は model のための構造体であり、complex の集合を保持する。また、すべての位相要素を高速に辿る目的のために、冗長ではあるが、生成されたすべての face, edge, vertex の集合をリストとして保持する。

```
struct MODEL
{
   int            type;         /* 生成時に２が代入される */
   int            work;         /* 作業領域 */
   ATTRIBUTE*     attribute;    /* 属性 */
   ROOT*          upper;        /* 上位の ROOT */
   MODEL*         previous;     /* ROOT に保持されるリスト */
   MODEL*         next;
   COMPLEX*       first;        /* COMPLEX のリスト */
   COMPLEX*       last;
   FACE*          first_face;   /* すべての face のリスト*/
   FACE*          last_face;
   EDGE*          first_edge;   /* すべての edge のリスト*/
```

```
   EDGE*          last_edge;
   VERTEX*        first_vertex; /* すべての vertex のリスト*/
   VERTEX*        last_vertex;
};
```

COMPLEX

COMPLEX は形状モデルの連結成分を表現する構造体で、下位の位相要素として、volume、volume の境界でない co-face、face の境界でない co-edge、または孤立した co-vertex を保持する。

```
struct COMPLEX {
   int               type;         /* 生成時に３が代入される */
   int               work;         /* 作業領域 */
   ATTRIBUTE*        attribute;    /* 属性 */
   MODEL*            upper;        /* 上位の model  */
   COMPLEX*          previous;     /* MODEL に保持されるリスト */
   COMPLEX*          last;
   ELEMENT*          first;        /* 構成要素のリスト */
   ELEMENT*          last;
};
```

VOLUME

VOLUME は shell の集合を保持する。

```
struct VOLUME {
   int               type;       /* 生成時に４が代入される */
   int               work;       /* 作業領域 */
   ATTRIBUTE*        attribute; /* 属性 */
   COMPLEX*          upper;      /* 上位の complex */
   SHELL*            first;      /* shell のリスト */
   SHELL*            last;
   ELEMENT*          previous;  /* complex に保持されるリスト */
   ELEMENT*          next;
};
```

SHELL

SHELL は volume の連結な境界位相要素のための構造体である。下位の位相要素は、co-face、face の境界でない co-edge、または孤立した co-vertex である。

```
struct SHELL {
   int               type;       /* 生成時に５が代入される */
   int               work;       /* 作業領域 */
   ATTRIBUTE*        attribute;  /* 属性 */
   VOLUME*           upper;      /* 上位の volume */
   ELEMENT*          fitst;      /* 構成要素のリスト */
   ELEMENT*          last;
   SHELL*            previous;   /* volume に保持されるリスト */
   SHELL*            next;
};
```

FACE

face は二つの co-face から構成されるが、これらの co-face は *mate* という関係で結ばれているので、一方の co-face を保持すればよい。また、曲面の方程式を保持するため、GEOMETRY へのポインタを持つ。

```
struct FACE {
   int             type;       /* 生成時に６が代入される */
   int             work;       /* 作業領域 */
   ATTRIBUTE*      attribute;  /* 属性 */
   CO_FACE *       co_face;    /* 一方の co_face */
   FACE*           previous;   /* MODEL に保持されるリスト */
   FACE*           next;
   GEOMETRY*       geometry;   /* 曲面の方程式 */
};
```

CO_FACE

CO_FACE は、face と他の位相要素との関係を記述するための補助位相要素の構造体である。下位の位相要素として co-loop のリストを持つ。対になる co-face は *mate* で

記述される。また、co-face は face の裏表のいずれかに対応するが、co-face の向きが FACE で記述された gometry の曲面の向きと一致するかを *code* の正負として記述する。上位の位相要素は、SHELL または COMPLEX である。

```
struct CO_FACE {
   int               type;       /* 生成時に７が代入される */
   int               work;       /* 作業領域 */
   ATTRIBUTE*        attribute;  /* 属性 */
   ELEMENT*          upper;      /* 上位の位相要素  */
   ELEMENT*          previous;   /* 上位の位相要素のためのリスト */
   ELEMENT*          next;
   CO_LOOP*          first;      /* co-loop のリスト */
   CO_LOOP*          last;
   FACE*             face;       /* face へのポインタ */
   CO_FACE*          mate;       /* 対となる co-face */
   int               code;       /* co-face の向き（+/-）*/
};
```

CO_LOOP

co-loop は、co-face の境界の連結成分を管理するための構造体である。co-face の境界は、外郭を構成するものと face の穴を構成するものに分けることができる。この区別を *code* の正負として管理する。下位の位相要素は、co-edge の集合、または co-vertex である。co-edge の集合を下位要素として保持する場合、連結な co-edge の集合はループサイクルを辿ることによって得られるので、一つの co-edge を保持しておけばよい。なお、本モデラでは、co-loop の対で構成される loop はデータ構造として保持していない。

```
struct CO_LOOP {
   int               type;       /* 生成時に８が代入される */
   int               work;       /* 作業領域 */
   ATTRIBUTE*        attribute;  /* 属性 */
   CO_FACE*          upper;      /* 上位の co-face */
   CO_LOOP*          previous;   /* co-face に保持されるリスト */
   CO_LOOP*          next;
   ELEMENT*          lower;      /* 下位の位相要素 */
```

```
    CO_LOOP*            mate;       /* 対となる CO_LOOP */
    int                 code;       /* face の外郭か空洞か（+/-）*/
};
```

EDGE

EDGE は co-edge の集合で構成されるが、ラジアルサイクルの表現である co-edge の *mate* と *radial* を辿ればすべての co-edge を得ることができるので、一つの co-edge を保持するだけでよい。曲線の方程式は、*geometry* として保持される。

```
struct EDGE {
    int                 type;       /* 生成時に9が代入される */
    int                 work;       /* 作業領域 */
    ATTRIBUTE*          attribute;  /* 属性 */
    EDGE*               previous;   /* MODEL に保持されるリスト */
    EDGE*               next;
    CO_EDGE*            co_edge;    /* 一つの co-edge */
    GEOMETRY*           geometry;   /* 曲線の方程式 */
};
```

CO_EDGE

co-edge は向きを持っているので、始点に相当する co-vertex を *start*、また edge の *geometry* で記述された曲線の向きと同じかどうかを *code* に記述する。

なお、CO_EDGE の上位の位相要素は、COMPLEX、SHELL、CO_LOOP のいずれかである。この構造体では、上位の位相要素によって保持すべきデータが異なる。

上位の位相要素が co-loop のときは、サイクルの情報が保持される。ループサイクルとして、反時計回りに辿る co-edge を *ccwe* (counter-clock-wise co-edge) として保持し、また、ラジアルサイクルとして、向かい合う face-use を *radial* として保持する。さらに、*mate* で繋がれた co-loop を上位に持つ co-edge 同士を *mate* で連結する。

一方、上位が COMPLEX、SHELL のときはサイクルを表現する必要がなく、代わりに、上位の位相要素に保持されるリストに繋がらなければならない。したがって、*ccwe*, *radial* を保持する代わりに、リストを表現するための *previous*, *next* を保持すること

が必要である。そこで、ここでは共用体 union を用いてデータ構造を記述する（なお、この union の用法は C++ で匿名の共用体と呼ばれるもので、C では認められていない。）

```
struct CO_EDGE{
   int             type;       /* 生成時に１０が代入される */
   int             work;       /* 作業領域 */
   ATTRIBUTE*      attribute;  /* 属性 */
   ELEMENT*        upper;      /* 上位の位相要素 */
   union {
     CO_EDGE*      ccwe;       /* loop で反時計回りの co-edge */
     ELEMENT*      previous;   /* shell, complex に保持されるリスト */
   };
   union {
     CO_EDGE*      radial;     /* ラジアルサイクル */
     ELEMENT*      next;       /* shell, complex に保持されるリスト */
   };
   CO_VERTEX*      start;      /* 始点の co-vertex */
   CO_EDGE*        mate;       /* 対となる co-edge */
   EDGE*           edge;       /* edge へのポインタ */
   int             code;       /* co-edge の向き (+1/-1)*/
};
```

VERTEX

VERTEX は co-vertex の集合をリストとして保持する。また、座標値を x, y, z として保持する。

```
struct VERTEX {
   int               type;       /* 生成時に１１が代入される */
   int               work;       /* 作業領域 */
   ATTRIBUTE*        attribute;  /* 属性 */
   VERTEX*           previous;   /* MODEL に保持されるリスト */
   VERTEX*           next;
   CO_VERTEX*        first;      /* co-vertex のリスト */
   CO_VERTEX*        last;
```

```
   double            x, y, z;     /* 座標値 */
};
```

CO_VERTEX

CO_VERTEX では、上位の位相要素は COMPLEX, SHELL, CO_LOOP, CO_EDGE のいずれかとなる。上位が、COMPLEX, SHELL, CO_LOOP になる場合は、孤立点となる。

```
struct CO_VERTEX {
   int               type;        /* 生成時に１２が代入される */
   int               work;        /* 作業領域 */
   ATTRIBUTE*        attribute;   /* 属性 */
   ELEMENT*          upper;       /* 上位の位相要素 */
   CO_VERTEX*        previous;    /* VERTEX に保持されるリスト */
   CO_VERTEX*        next;
   VERTEX*           vertex;      /* VERTEX へのポインタ */
};
```

4.6 まとめ

本章では、非多様体形状モデルを表現するためのデータ構造について述べた。要点は以下の通りである。

- 従来の非多様体データ構造を実装する上での問題点を論じ、位相要素として volume を管理するデータ構造を提案した。

  このデータ構造では、サーフェスモデルの処理で大局的な閉空間探索が行なわれないので、従来の場合に比べて効率的な処理が可能となる。また、volume の有無によって閉じた２次元点集合と３次元点集合をデータ構造上区別することができるため、本論文の第２章で論じた非多様体形状モデルの定義域に沿った形状表現ができる。

- 非多様体形状モデルのデータ構造の設計を行い、非多様体形状を記述するデータ構造を具体的に示した。

また、第５章と第６章で述べられる形状処理は本データ構造に基づいて実装されており、それらの例題を通じて、本データ構造が実用的に十分な効率で動作することを実証している。

Chapter 5

併合/抽出操作に基づく集合演算と形状特徴表現への応用

本章では、非多様体形状モデルの集合演算について論じる。

非多様体形状モデルにおいては、集合演算を実現するための手順を、演算順序に依存する併合操作と依存しない抽出操作に分離できるという特徴がある。本研究では、非多様体形状モデルの集合演算を実現する新しい手法として、併合操作、抽出操作、簡略化操作を組み合わせた集合演算の実現法を提案する。また、その実装方法についても明らかにする。

さらに、本研究で提案する集合演算を、試行錯誤的な形状生成、及び形状特徴表現に応用する方法についても示す。

本章は、本論文で最も長い章となっているので、ここで各節の内容と関係を簡単に紹介しておく。

第１-３節は前置きであり、本章で用いる位相構造がなぜ必要かを説明する。ここでは、ソリッドモデリングでは、試行錯誤的環境での形状生成と形状特徴表現に問題があり、それらが非多様体位相を用いた表現によって解決できることが説明される。

第４節は、本論文で提案する集合演算の実現手法の全体構成について述べている。本章の本質的な部分である。

第５-７節では、前置きで述べた問題点の解決法を具体的に示していく。第５節は、集合演算の取消操作の実現法が提案され、実際にいくつかの例題に適用した結果が示される。第６節は、集合演算のバリエーションを示したもので、優先順位を考慮した集合演算操作が可能なことが述べられる。第７節は、形状特徴の表現法が述べられる。

第８-１０節では、非多様体形状モデルの集合演算の実装法を説明する。第８節は、抽出操作の実装法、第９節は、併合操作の実装法、第１０節は、簡略化操作の実装法を示している。

第１１節は、結論である。

5.1 集合演算

集合演算は、境界表現ソリッドモデルを作成する手段として最も初期から用いられてきた操作であり [Braid93]、今日でも最も広く用いられている。集合演算は、非多様体形状モデリングにおいても、重要な形状生成手段となる。ただし、非多様体形状モデルにおいては、ワイヤフレーム、サーフェス、ソリッドが混在する。図 5.1 に、ソリッドと平面の積集合から、断面形状を作成した例を示す。非多様体形状モデルの集合演算としては、このようにソリッドモデルよりも広い定義域で集合演算を考えなければならない。

ここでは、集合演算で用いられる形状モデルを **プリミティプ**、集合演算で得られる形状を **評価形状** (evaluated shape) または **結果形状** (resultant shape) と呼ぶことにする。

非多様体形状モデルの集合演算については、本論文の「非多様体形状モデルの定義」の章において既に定義を行なった。本章では、この定義に基づいた和集合、差集合、積集合について考える。この集合演算の定義では、評価形状を図 5.2 に示したような点集合としている。

ただし境界表現においては、これらの点集合をどのような位相構造で表現するかという問題が残されている。もちろん、従来のソリッドモデルの集合演算のように、必要最小限の位相要素で評価形状を表現するとしてもよいが、一方で、非多様体形状モデルの自由度を積極的に利用することによって、従来のソリッドモデルでは解決が困難であった問題に対処する可能性を検討することも必要である。

そこで、まず、従来のソリッドモデルでは解決が困難であった問題について述べ、それらが非多様体形状モデルを用いることでどのように解決できるかを述べていく。

なお、非多様体形状モデルの集合演算に関して、実現法にのみ興味のある場合には、本章の 4, 8, 9, 10 節のみを読めばよい。これらは非多様体形状モデルのための集合演算を実現するためのアルゴリズムについて述べた節である。

図 5.1: ソリッドとサーフェス間の集合演算.

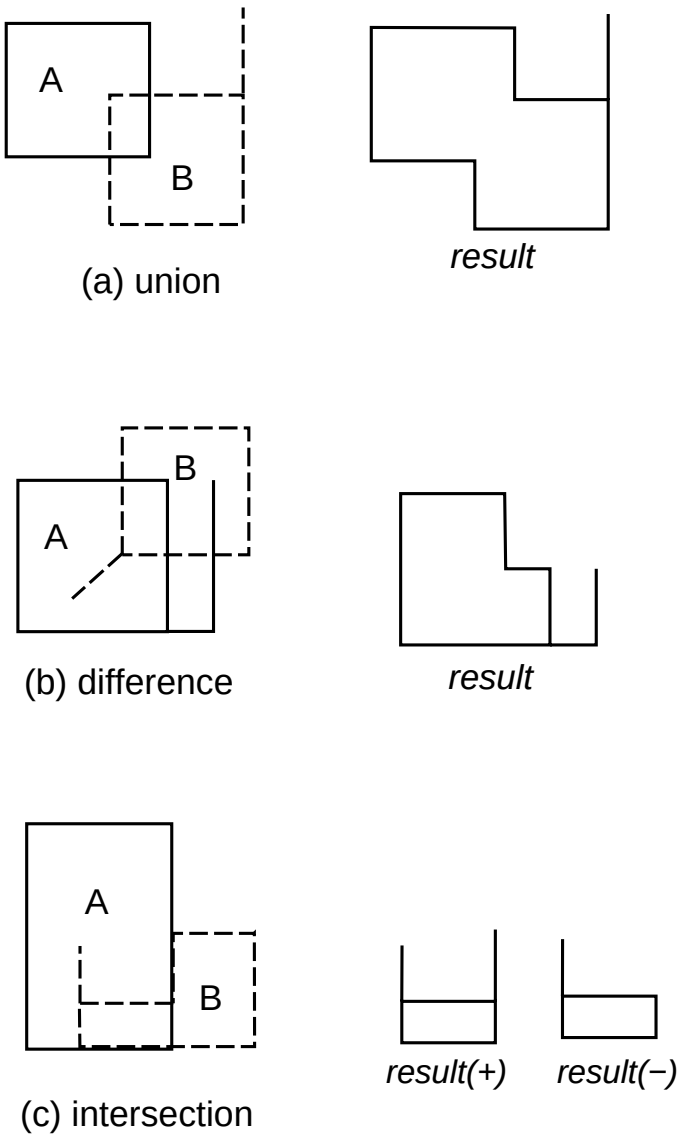


図 5.2: 非多様体形状モデルの和集合、差集合、積集合.

5.2 集合演算の応用上の問題点

非多様体形状モデルの集合演算の実現法について述べる前に、従来のソリッドモデルにどのような問題点があったかについて述べておく。ここでは、以下に示す二つの問題について論じる。これらの問題は、非多様体形状モデルの集合演算を応用することによって、解決することが期待できる。

試行錯誤的な形状生成

ソリッドモデリングでは集合演算が広く用いられてきたが、いくつかの問題点も指摘されている。その一つは、多くのプリミティブを組み合わせて生成された境界表現ソリッドモデルに修正を施すためには、非常に計算時間がかかるということである。これは、多くの形状モデラでは、形状の一部を修正するだけでも、一旦形状モデルを破棄して、過去の操作履歴を用いて初めから形状を作り直しているためである。そのため、比較的大規模な形状モデルを、パラメトリック設計のような試行錯誤的な形状生成法で作成しようとすると非常に操作性が悪いという問題があった。

さらに、集合演算による形状生成を考えると、集合演算を施す順序によっては、以前に開けた穴を塞いでしまったり、必要でない部分まで除去してしまうこと場合がある。その場合には演算順序を変えて集合演算をやり直したり、塞がれた穴を再度開け直すなどの処理が必要であった。すなわち、形状生成と集合演算の順序とは深く関係しているために、形状が確定した部分から優先的に形状定義を行っていく試行錯誤的なモデリングとは折り合いが悪かった。

形状特徴表現

形状モデルをプロセスプラニングなどに用いようとする場合、形状データだけでは不十分であり、加工情報、公差、材質など様々な工学的情報が保持されていなければならない。近年では、形状モデルにおいて穴や溝など工学的に意味がある部分形状を形状特徴として管理したり、形状特徴を陽に用いて形状定義を行なうといったことが一般的になってきている。実際、いくつかの商用システムでは、形状特徴を利用したプロセスプラニングを行なっている。

しかし、従来のソリッドモデルでは位相的な制約から表現できる形状特徴が限られており、たとえば加工などでしばしば用いられる体積特徴などは管理が難しい。また形状特徴間で干渉が起きる場合などでは、形状特徴が部分的に消去されることがあり、形

状特徴としての整合性が維持できなくなるという問題がある。

以下において、この二つの問題に関して、従来のソリッドモデルが持つ問題点について詳細に論じていくことにする。

5.2.1 集合演算の修正における問題点

試行錯誤的な設計の場でのソリッドモデルの利用を考えると、変形操作の取り消しや修正が高速にできることが重要になる。しかし、従来は、ソリッドモデルの修正は非常に時間にかかる操作であった。

図 5.3 に、集合演算 $(A \ominus B) \ominus (C \oplus D)$ で定義された形状から、プリミティブ B, C, D を取り消して生成される 3 通りの形状モデルを示す。

形状モデルの修正のための手順としては、それ以前に適用したすべての集合演算を始めからやり直すか、undo 操作で 1 ステップずつ後戻りするという二通りの方法が存在する。しかし、これらの方法は、この図のようにプリミティブの個数が比較的少数の場合には有効であるが、形状モデルが複雑になり、プリミティブの個数が多くなってくると修正に要する時間が長くなり、操作性が低下していくという問題がある。

集合演算の再実行

商用システムで多く用いられている修正方法は、集合演算の履歴を保存しておいてそれを始めからやり直す、という方法である。図 5.3 の例では、B を取り消すためには、$A \ominus (C \oplus D)$ を再計算するわけである。

しかし、この方法では、複雑な形状モデルの穴の径を一つ変更するだけでも、いままでの集合演算をすべてやり直して形状モデルを作り直さなければならない。大雑把にいえば、集合演算の再実行で修正を行なう場合、n 個のプリミティブから生成された形状モデルの場合、以前の集合演算を一つ修正するのに要する計算時間は、n^2 に比例すると考えられる。これは以下のような計算による。

- n 個のプリミティブから作られた形状モデルに $(n+1)$ 個目のプリミティブを加えたときの集合演算の計算時間 t_n を k を定数として、$t_n = k \cdot n$ と仮定する。すなわち、面の個数に比例して、集合演算の計算時間が増大するものとする。(この仮定は理想的な実装手段を用いた場合に成立するもので、最悪の場合では面の

図 5.3: 集合演算を取り消して得られる形状モデル.

個数の自乗に比例する。）このとき、プリミティブの一つを変更して形状の修正を行うことを考えると、以前の n 回の集合演算操作をすべてやり直す必要があるので、修正に要する計算時間 T は、

$$T = \sum_{i=1}^{n} t_n = (k/2)n(n+1)$$

となり、計算時間はプリミティブの個数の自乗に比例する。

- 図 5.35 は、多数のプリミティブから成る形状モデルを示しているが、プリミティブ１０個の場合の集合演算の再実行に ０.７２秒、プリミティブ１００個のときは４８秒かかっている。すなわち、プリミティブの個数が１０倍になることにより再計算に要する時間は６７倍になっており、この傾向が確認できる。

問題なのは、唯一つの操作を取り消すためであっても本来関係のないすべての集合演算を再計算しなければならないことであり、修正に要する時間がプリミティブの個数の自乗に比例して増大していくという性質である。形状生成の初期段階では修正に要する時間はそれほど気にならなくても、プリミティブの個数が多くなるに従って、たとえ形状のごく一部を変更しただけであっても多大な時間がかかるようになる。図 5.35 の例題のように、一部を変更する度に、再計算に４８秒もかかっていたのでは、試行錯誤的な環境でのモデリングは困難である。

undo 操作

このように、すべての集合演算を始めからやり直す方法は、計算コストが高く、応答性を損なうので、直前の集合演算操作を高速に取り消すことのできる undo 操作が千代倉と木村によって提案された [Chiyokura83b]。なお、undo 操作の実現方法には、オイラー操作の逆操作を利用するものと、データベースの復元機能を利用する方法とが知られている。

集合演算をオイラー操作の列によって実現した場合、すべてのオイラー操作が逆操作を持つという性質から、集合演算を適用する以前の状態に戻す逆操作が実現できる [Mantyla82, Chiyokura83a]。ただし、逆操作に必要なデータはオイラー操作の適用順序に依存するため、今まで行なってきた操作を逆向きに順次復元していかなければならない。

したがって、この方法では、直前の操作の取り消しは非常に高速であるが、モデリングの初期に適用された集合演算を取り消すには時間がかかる。なぜなら、取り消すべき集合演算以降のすべての形状を順次復元し、その上で本来は取り消す必要のなかっ

た集合演算を再度適用しなければならないからである。そのため、修正したい部分によっては、やはり膨大な計算時間がかかってしまう。

同様の undo 操作は、汎用的なデータベースの機能を用いることによっても実現できる。この場合、形状データが変更される度に、データベースのどの部分がどう変更されたかのログに取っておき、必要に応じてデータベースを以前の状態に復元する。この場合も、修正に要する時間が演算順序に依存するので、オイラー操作と同様の問題点が生じる。

また、 undo 操作以外の修正方法としては、CSG 表現の各ノードに中間状態の境界表現データを対応させておき、必要に応じて境界表現を計算し直す方法 [Turner87] も考えられるが、この場合でも初期の集合演算を取り消した場合は、各ノードでのプリミティブの組合せが変わるので、取り消したプリミティブを含むすべてのノードに対応する境界表現データを再計算しなければならない。また、中間段階のデータも保持するため、データ量が膨大になるという問題点がある。

再実行、undo 操作、取消操作の比較

図 5.4 は、集合演算で生成された境界表現形状モデルを修正する方法をまとめたものである。

(a) では、システムは、集合演算の履歴のみを保持している。この場合では、直前の操作を取り消す場合でも、すべての集合演算を再実行することになる。したがって、複雑な形状モデルを試行錯誤的に生成すると非常に操作性が悪くなる。

(b) では、オイラー操作の逆操作やデータベースの機能を用いて後戻りできる場合である。集合演算とその後戻り操作は、Δ, Δ^{-1} で示されている。後戻り操作においては、幾何計算を行なうことなく、データベース上の操作だけが行なわれる。この方法では、直前の状態に戻すことが容易であるが、任意の集合演算を取り消す場合、計算時間は演算の適用順序に依存して計算時間がかかる。

(c) は、理想的な修正操作である。これはどの集合演算を取り消す場合であっても、操作順序に関係なく、幾何計算を行なわずに形状の修正ができる操作である。 このような操作が実現できれば、これまで述べてきたような問題点が解決でき、大規模な形状モデルでも非常に高速な修正操作が可能になる。しかしながら、従来のソリッドモデリングにおいてはこれまで実現されていなかった。

本章では、図 5.4 (c) のような演算順序に依存しない取消操作を実現し、比較的大規模

な形状モデルでも試行錯誤的形状生成が可能となる形状操作を提案する。そのような取り消し操作については本章の第５節で論じる。

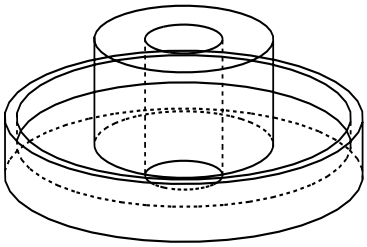
図 5.4: 集合演算を取り消すための３通りの手順.

5.2.2 演算順序における問題点

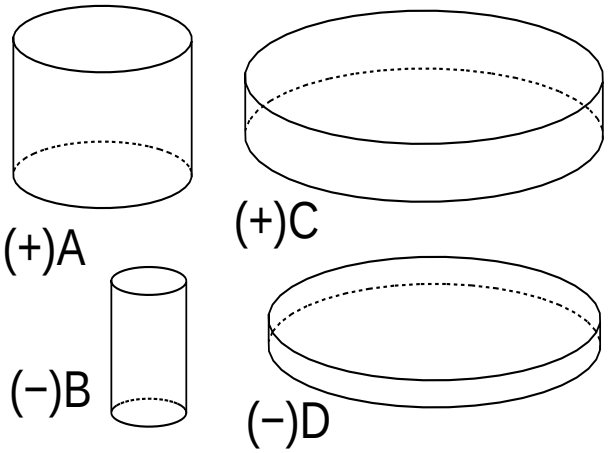
次に、集合演算のもう一つの問題として、順序依存性について考えてみる。集合演算は、一般に、

$$(A \oplus B) \ominus C \neq A \oplus (B \ominus C)$$

であるから、和集合、差集合、積集合の組み合せからなる集合演算では、演算の順序を変えると評価形状も変わってしまう。このことも、操作性のよい形状処理操作が容易に実現できない大きな理由の一つであると考えられる。



(a) geometric model R.



(b) primitives.

図 5.5: 集合演算の順序によって形状が変わる例.

ここで、図 5.5 (a) の形状モデルをプリミティブを組み合わせることによって生成することを考える。必要なプリミティブを図 5.5(b) に示す。AとCは足すべき形状、Bと

Dは引くべき形状である。しかし、これらのプリミティブの演算順序については、多くの組み合わせがある。この例題のようにプリミティブ間に干渉部分のある場合には、一度空けた穴が他のプリミティブに埋められてしまったり、また不必要な部分まで引いてしまうということが起こり、正しい形状の得られる演算順序は限られている。この例では正しい形状 (a) を得るためには、$C \ominus D \oplus A \ominus B$ の順で操作を行わなくてはならない。

従ってもし、軸と円盤とを別々に $A \ominus B$, $C \ominus D$として作ったり、また、D の部分は機能上重要ではないから最後に D の高さを決めたいような場合は、途中で CSG ツリーを編集して集合演算をやり直したり、埋まってしまった穴を再度引き直すといった余計な手間を余儀なくされる。実際、多くの商用の形状モデラでは、CSG ツリーを編集し、演算順序を入れ換えて形状修正を行なうことが不可欠な作業となっている。

集合演算はその定義上、演算順序に依存する。そのため、幾つかのプリミティブに集合演算を施して目的の形状を作るには、集合演算を施す順序に十分気を配らなければならない。

このことは試行錯誤的に形状を決めていく設計作業のサポートには非常に都合が悪い。設計者は寸法や形の決まった部分から順に形状モデルを作っていくべきであって、モデラに都合のよい順に形状を生成するのは妥当であるとはいえない。図 5.5 での C, D, A, B という演算順序には、設計作業においては何の必然性もなく、ただ単にモデラの制約による都合に過ぎない。

そこで、設計作業に適した形状生成を考えるならば、演算順序の影響が制御できるような集合演算が望ましい。そのような集合演算操作については、本章の第 6 節で論じる。

5.2.3 形状特徴表現における問題点

次に形状特徴表現の問題点について考えるが、それについて論じる前に、形状特徴に関する基本的な事柄について概観する。

・形状モデルにおける形状特徴

形状特徴

形状特徴 (form-feature) という用語は、設計・生産においてしばしば用いられてきた。形状特徴という考え方は、切削加工に適した部分形状を形状モデルに記述する研究から発展したといわれているが [Shah88, Wilson90]、現在一般に多く用いられているのは、Wilson と Pratt の用いた

「部品形状において興味のある部分形状」

という定義である [Wilson88] 。

一般に機械設計では、設計者は要求された機能の実現のために、公差、加工、組み立て性など様々な工学的問題を考慮する必要があり、形状はそれらの要求から決定されると考えられる。形状特徴とは、形状が本来持つ意味を記述したものであり、より抽象度の高いレベルで幾何形状を捉えたものということができる [Ostrowski87]。

陽表現の形状特徴

形状特徴は、陰表現と陽表現に分けて考えることができる [Pratt88]。

陰表現とは幾何形状として現れていない形状で、たとえば、「点 P を中心に半径 R で深さ H の円筒穴を垂直に開ける」という記述自体が陰表現となる。陽表現とは陰表現を評価して得られるもので、この場合は形状モデルの穴の部分が陽表現の形状特徴となる。

ここでは、陽表現の形状特徴について考える。陽表現の形状特徴は形状モデルのデータを必要とする解析や加工データの作成などで広く用いられる。陽表現の形状特徴においては、形状特徴を工学的な意味を持った「形状モデルの部分形状」と考えることができる [Pratt88]。

形状特徴の視点依存

何を形状特徴として捉えるかは、形状をどのような視点で捉えるかに依存する。たとえば、加工を考えるならば、穴やポケットなどの加工単位を形状特徴と見做すのが自然であるし、加工精度を考えるならば公差の与えられた部分が形状特徴となるであろう。組み立てにおいては、接触部分や接触の仕方などが形状特徴と見做される。図 5.6 は、形状モデルを工学的な評価に用いようとしたときに必要とされる形状特徴の例を示している。形状モデルを利用して設計の評価を行なう場合、評価しようとする対象に依存した工学的情報が必要となる。

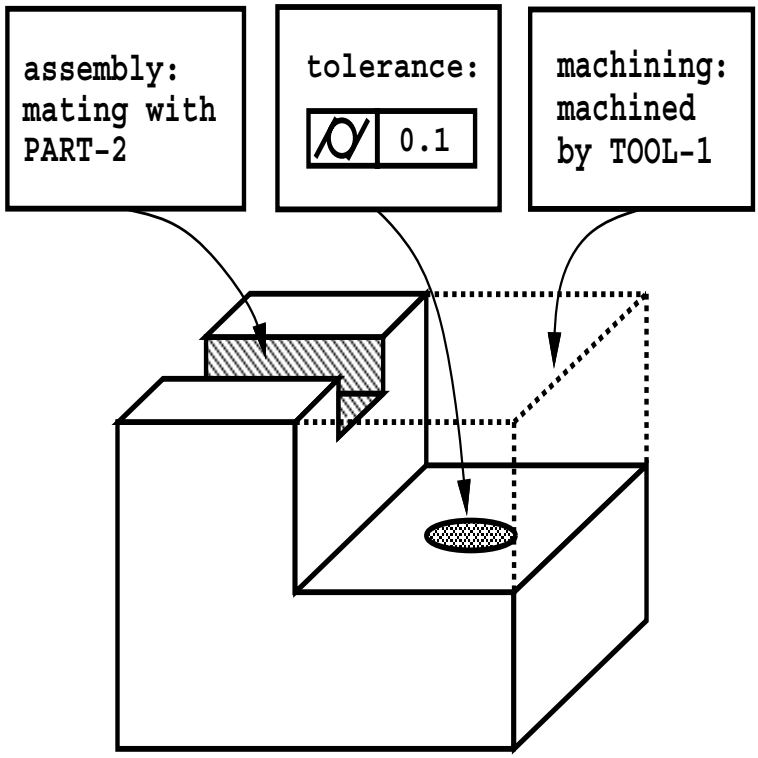


図 5.6: 形状モデルに対して記述される工学的情報.

形状特徴の獲得手段

形状特徴には、このように視点に依存するという問題もあるが、もし形状モデルに対して対象に応じた形状特徴が適切に保持されていれば、組み立て性評価や構造解析、プロセスプラニングなどを計算機によって有効に支援できることが期待できる。ここで、形状特徴をどのように獲得していくかということが問題になるが、そのための手段と

して「特徴抽出」(feature extraction) と「形状特徴による設計」(design by feature) がよく知られている。

形状特徴抽出

形状特徴抽出とは、既存の形状モデルから穴や溝などの形状特徴を自動的に抽出する方法である。一般的には、穴や溝に相当する位相や幾何をグラフ構造で表現し、そのパターンが形状モデルの境界表現データに存在するかを検索するパターンマッチングの手法が用いられる。

ただし、この方法は比較的単純な位相構造を持つ形状特徴には有効であるが、形状特徴間の干渉がある場合には検出が難しいという問題がある [Sakurai90]。また、位相や幾何のデータを探索することで形状特徴を抽出するので、平面部の公差のようなありふれた位相を持った形状特徴は特定が難しいという問題もある。

一般にすべての形状特徴抽出を自動的に行なうのは限界があり、解析を行う技術者が形状特徴を構成する面や稜線を直接選択するといった補助的な手段も必要となる。

形状特徴による設計

形状特徴を獲得するためのもう一つの方法は、形状を定義するときに「形状の意味」も一緒に保持する方法である。設計者は、形状設計の段階では形状の意味について認識しているが、形状モデルを作成する段階ではそれらの意味は捨てられ、位相と幾何の情報のみが残される。当然のことながら、一般には、位相と幾何から形状の意味を再生することは不可能である。そこで、形状特徴はできる限り形状生成の段階で獲得すべきだとする考え方があり、「形状特徴による設計」(design-by-feature) と呼ばれている [Dixon87]。図 5.7 は、設計においてよく用いられる形状特徴を示している。形状特徴による設計では、このような形状特徴を組み合わせることによって形状モデルを作成していく。

ただし、形状特徴による設計にも問題点がある。既に述べたように、形状特徴に関しては視点に応じて何が形状特徴かが変わるという問題がある。そのため、場合によっては形状生成の段階で獲得した形状特徴を別の形状特徴に変換する必要がある。たとえば、図 5.8 では、形状生成の段階で、突起 (protrusion) として定義された場合、溝 (groove) は形状特徴として陽には現れていない。しかし、加工の視点からは削りとる部分である「溝」が重要な形状特徴となる。このような場合には、必要に応じて形状

図 5.7: 形状特徴の例.

特徴抽出などを行なって溝の部分を検出する必要がある。

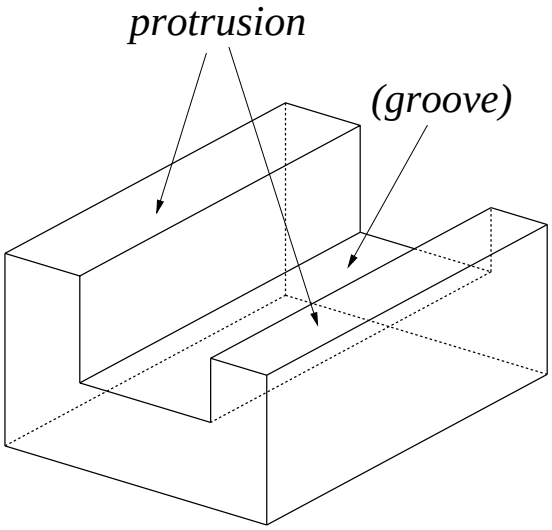


図 5.8: 形状定義の段階では現れない形状特徴 (groove).

境界表現モデルにおける形状特徴

さて、ここで、いずれかの手段で獲得された形状特徴をどう保持していくかについて考えていく。ただし、形状特徴に関する研究は膨大であり [Shah88, Wilson90]、工学的な意味まで考慮した形状特徴の表現法を一般的に論じることは困難なので、形状特徴の幾何的情報を保持する問題に限定し、工学的意味は「属性」として抽象化して考えるものとする。すなわちこの場合、形状特徴は「属性の付加された、形状モデルの部分形状」と見做される [Wilson88, Pratt88]。そのための表現手段としては、境界表現形状モデルが適していることが広く認識されており [Wilson88, Pratt88, Shah88, Ostrowski87]、形状特徴は境界表現モデルを構成する face, edge, vertex の部分集合として表現される [Wilson88, Shah88]。

・形状特徴表現における問題点

従来のソリッドモデルを用いて形状特徴設計システムを実現した場合、次のような、問題点が生じることが知られている [Masuda89a, Kawabe88, Shimada87]。

- 形状特徴は、工学的に興味のある部分が対象となるため、ワイヤフレーム、サーフェス、ソリッドなど様々な形態で表現できることが望ましいが、2-多様体ソリッドモデルでは表現できる形状が限られている。
- 最終的に形状モデルの境界位相要素として残らない位相要素には工学的情報を付加することができないため、設計の途中で現れる形状特徴が保持できないことがある。
- 形状特徴を構成する位相要素に干渉が生じた場合、位相要素と形状特徴との対応の保持が難しい。

境界表現ソリッドモデルでは、形状特徴は位相要素の集合として表現されるが、このような表現法では表現できる形状特徴が限定されている。なぜなら、形状特徴として記述したい部分が境界表現の位相要素として現れていない場合には、当然のことながら、部分形状として保持できないからである。たとえば、図 5.9 に斜線で示したような体積特徴は形状モデルの位相要素の集合だけでは表現できない。

そのため、従来の形状特徴表現では閉じた領域を作るために、仮想的な面を境界表現の位相構造とは別に保持することも行われていた [Pratt88, Gomes91]。しかしこの方法の欠点は、境界表現データと仮想面のデータという二つの異なる表現があるために、形状が変形されたときに両者の整合性を保持する処理が大変複雑になるということである。これに対して、Pratt は、仮想面を非多様体位相構造によって保持し、この問題を解決する方法を提案している [Pratt88]。しかし、Pratt の方法でも表現できる形状特徴に制限があり、また、図 5.10 のように体積特徴間に干渉が生じる場合には対処できない。この方法では、正の形状特徴から削り取られる部分しか形状特徴として保持されず、対象が加工のための形状特徴に限定されているためである。

さらに、様々な工学的情報を扱おうとすると、形状特徴は、線特徴、面特徴、体積特徴に応じて、ワイヤフレーム、サーフェス、ソリッドといった様々な形態を取ることも考えられる。しかし、従来のソリッドモデルでは、2-多様体ソリッドに限定されていたため、これらは必ずしも表現することができなかった。例えば、中心線や補助面などはソリッドモデルでは表現できないので、形状特徴としては定義できなかった。

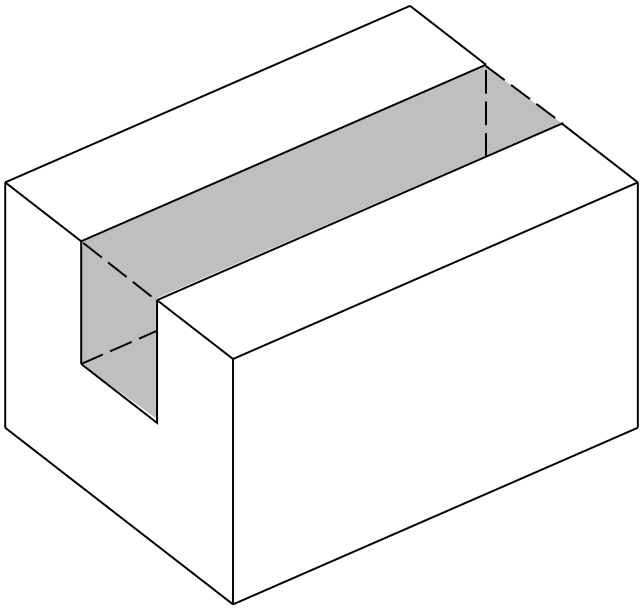


図 5.9: 付加的 face (斜線部) による体積特徴の表現.

また、設計の途中で工学的情報が付加された場合には、その形状特徴を維持することが難しいという問題がある。これは、形状特徴による設計 (design-by-feature) において、形状特徴間で干渉が起こる場合にしばしば生じる問題である。図 5.10 のように、二つの溝が干渉するような場合には、「溝」という属性の付けられた部分の位相要素が一部なくなる。このような場合、位相要素の集合として保持されている形状特徴の整合性を維持することは困難であった。

これらの問題に対処するための形状特徴表現については、本章の第 7 節で論じる。

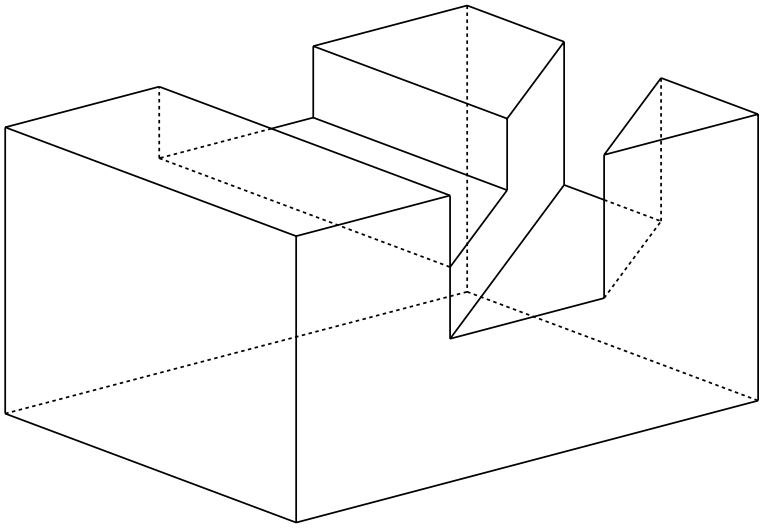


図 5.10: 形状特徴 (溝) が干渉する例.

5.3 本研究の方針

本研究では、以上述べた試行錯誤的形状生成と形状特徴表現の問題を、非多様体位相構造を用いた集合演算操作の応用として解決することを考える。本節では、併合操作と抽出操作を用いた新しい集合演算の基本的な考え方について述べ、それが前節の問題にどのように適用できるかについて述べる。

5.3.1 併合操作/抽出操作/簡略化操作を用いた集合演算

非多様体位相構造を用いると、集合演算の評価形状の表現に二通りの位相構造が可能となる。ここでは、二通りの表現に応じた集合演算がそれぞれ併合操作＋抽出操作、併合操作＋抽出操作＋簡略化操作の組み合わせで実現できることを示す。

併合操作

図 5.11 の立体A、Bの和集合について考えてみる。図 5.11(a) は、ソリッドモデルの和集合の評価形状であり、図 5.11(b) は本章で導入する「併合操作」によって生成された位相構造である。図 5.11(b) では、立体の内部にも面が残されており、三つの閉空間から構成されている。形状モデルの表現している形状は、位相要素の集合が占める空

間であると考えれば、図 5.11 (b) の形状モデルは細分化されてはいるものの、どちらの表現も同じ空間を占めている。従って、表現 (a) と (b) は同じ実体を表している別表現ということになる。

しかし、これらの位相構造には、プリミティブの位相情報が保持されているかどうかという本質的な違いがある。形状 (a) では、和集合を表現するための必要最小限の位相要素以外は消去されている。その結果、この位相構造だけから、元のプリミティブA, Bに関する情報は得られない。一方、形状 (b) では、立体の内部にある位相要素も残した位相構造を持っている。余分に残された部分は和集合を求めるという本来の目的からすれば不要であるが、これらの位相要素は元の形状A, Bの位相構造を保持するためには有用である。もし、形状 (b) の各位相要素にAとBのどちらに由来してできたが属性として記述されていれば、形状 (b) から元の立体A, Bを復元することも可能である。

ここでは、プリミティブから 形状 (b) のような位相構造を作る操作を **併合操作** と呼ぶ。なお、図 5.11(b) のような位相構造は 2-多様体ではないので、従来のソリッドモデルでは表現できない。プリミティブA、Bの干渉で生じた４本の稜線は４枚の面に共有されている非多様体稜線となっている。

抽出操作

集合演算には、和集合、差集合、積集合がある。図 5.12に、図 5.11 に示したプリミティブ A, B の和集合 (union)、差集合 (difference)、積集合 (intersection) と併合操作で作成された位相構造との関係を示す。この図からわかるように、和集合、差集合、積集合はいずれも併合形状の位相要素の部分集合となっている。したがって、もし、併合形状から適当な手続きによって和集合、差集合、積集合が抽出できるのであれば、併合形状と抽出手続きを保持することによって集合演算の評価形状が表現できることになる。

具体的なアルゴリズムは以後の節で述べるが、併合操作で作成された位相構造は、各位相要素がどのプリミティブに由来するものであるかを記述しておけば、それらのプリミティブ間の任意の集合演算の評価形状を抽出することが可能である。ここでは、併合形状から評価形状を抽出する操作を **抽出操作** とよぶ。

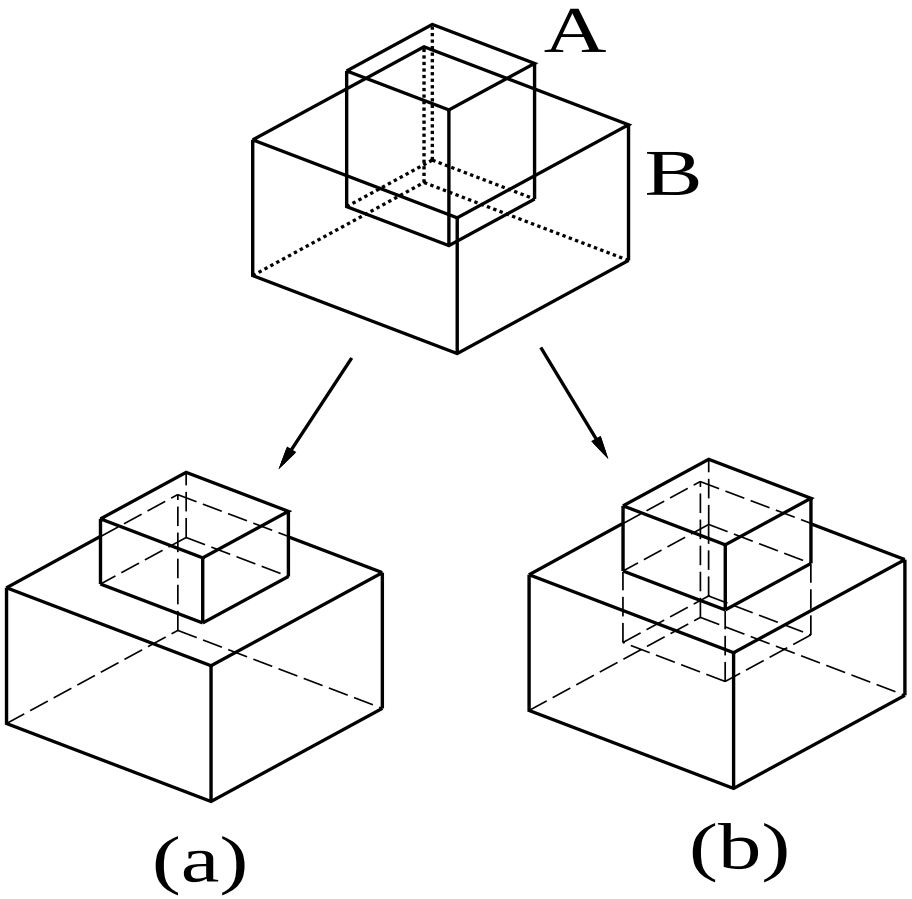


図 5.11: 従来の位相表現 (a) とプリミティブの位相要素を保持する位相表現 (b).

簡略化操作

併合操作と抽出操作にさらに **簡略化操作** を組み合わせることで、図 5.11 の表現 (a) のような必要最小限の位相要素からなる位相構造を生成することができる。簡略化操作とは、抽出操作で選ばれなかった位相要素を形状データから削除する操作である。本論文では、併合形状が試行錯誤的な形状生成や形状特徴表現に適していることを示しているが、これらの利点を必要としない場合には簡略化操作を適用してデータ量を軽減する。簡略化操作の実現法については、第 10 節で述べる。

併合操作/抽出操作/簡略化操作による集合演算

図 5.13 に三つの操作を用いた集合演算の手順を示す。

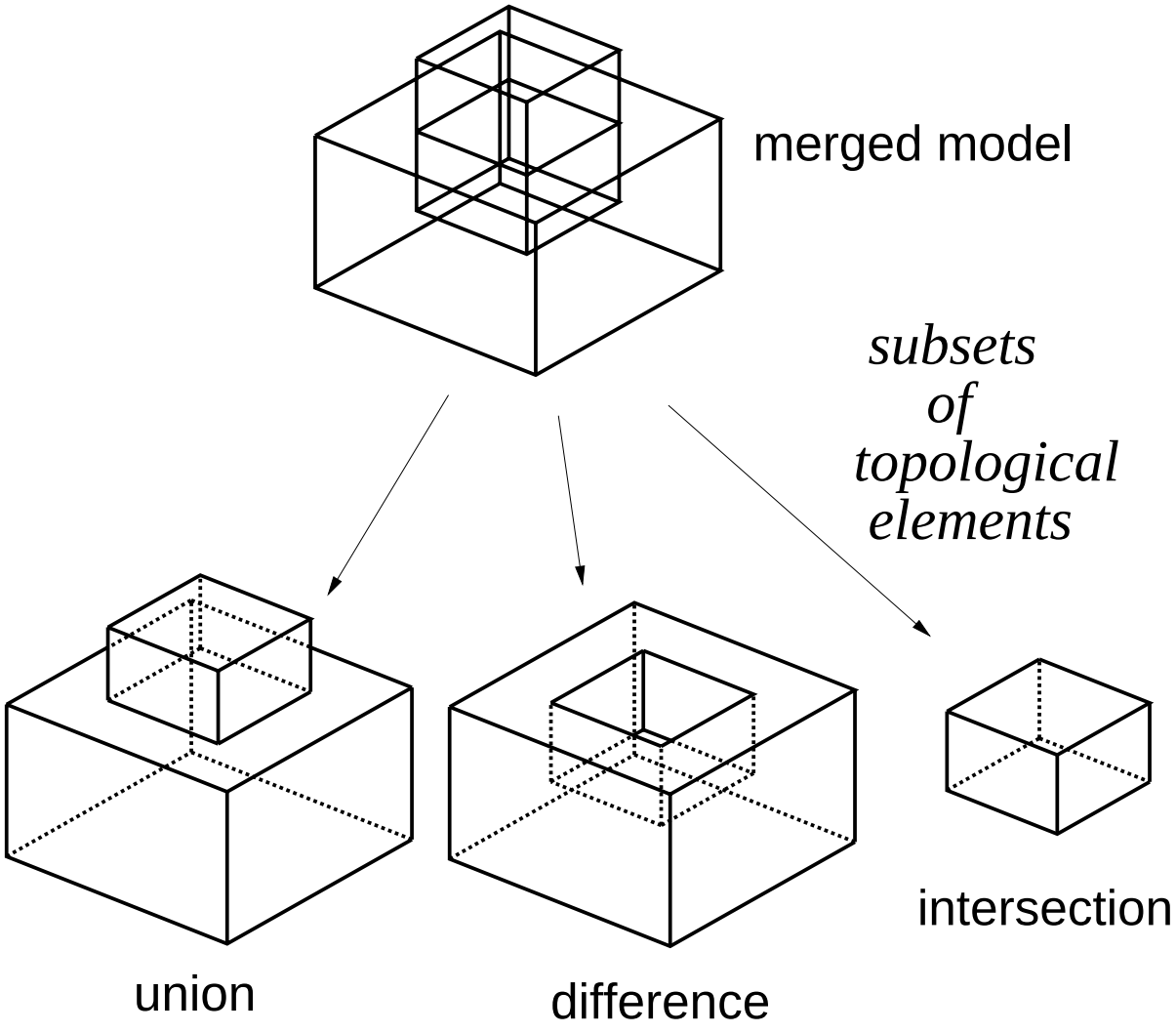


図 5.12: 併合形状から抽出される和集合、差集合、積集合.

1. まず、併合操作によって、プリミティブ A, B, C の併合形状 (merged model) を作成する。

2. 次に、抽出操作によって集合演算式を評価し、併合モデルから目的の形状 (extracted model) を取り出す。

3. 最後に、必要に応じて、評価形状に現れない位相要素を除去し、必要最小限の位相要素のみからなる形状モデル (simplified model) に変換する。

すなわち、本アルゴリズムにおいては、集合演算は、併合操作と抽出操作、また必要に応じて簡略化操作を組み合わせることで実現される。

併合操作と抽出操作の応用

次に、併合操作と抽出操作を利用して従来のソリッドモデルでは解決が困難であった問題を解くことを考える。ここでは次のような問題を考える。

試行錯誤的環境に適した集合演算

第２節で述べたように、従来の集合演算は試行錯誤的環境での形状生成には適していなかった。この問題には、併合操作と抽出操作による集合演算法が有効であると思われる。

集合演算を併合操作と抽出操作に分けるということは、集合演算を演算順序に依存しない操作と順序に依存する操作に分離することを意味している。

併合操作では、どのような順序でプリミティブを併合していっても、最終的にできる位相構造は同じである。したがって、集合演算が演算順序に依存するという性質が試行錯誤的環境に適した形状操作の実現を妨げていることを既に述べたが、併合操作ではそのような問題は生じないことになる。

一方、集合演算の順序依存の性質はすべて抽出操作に吸収されている。抽出操作は併合操作と違って、時間のかかる幾何計算や位相操作を伴わないので、全体の計算コストから考えるとほとんど無視できるほど負荷の軽い操作である。そのため、演算順序を入れ換えて抽出操作を再実行するといったことはほとんど計算時間を要さずに行なえる。

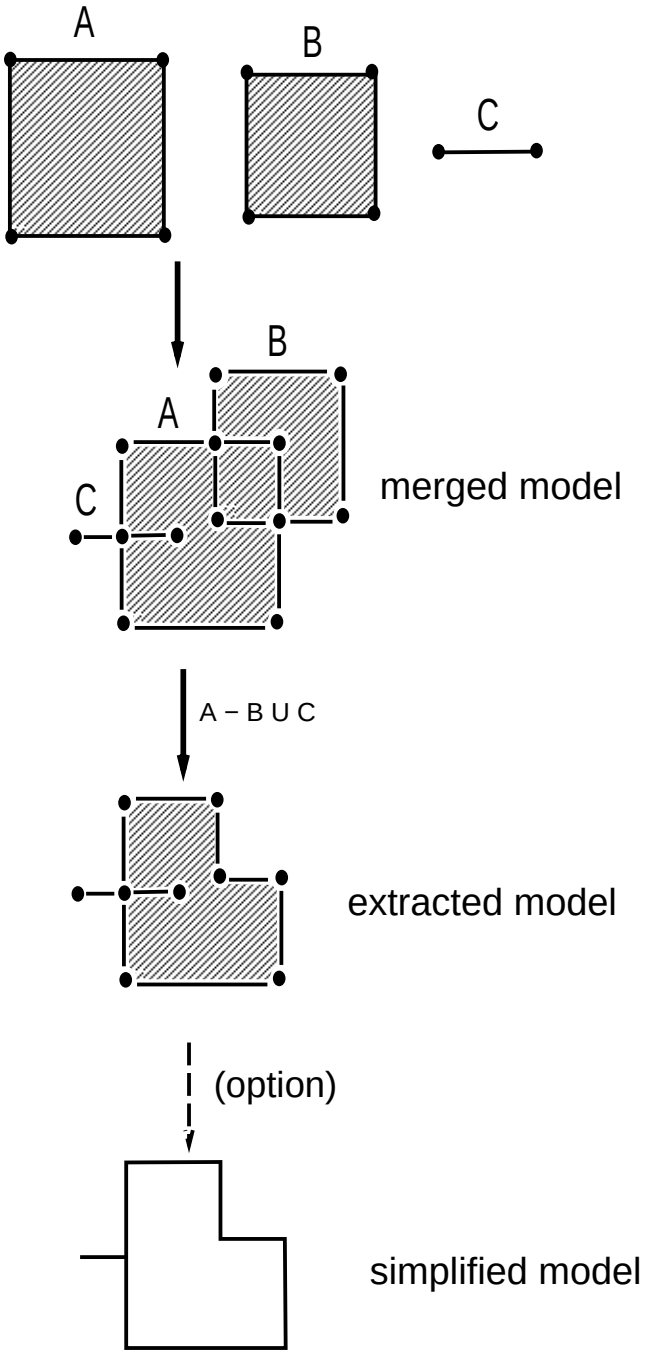


図 5.13: 集合演算を構成する三つの操作：併合操作、抽出操作、簡略化操作.

以上の性質を利用することで、試行錯誤的環境に適した形状操作を実現する。第５節では、演算順序に依存しない集合演算の取消/修正操作について、第６節では演算順序を柔軟に決めることのできる集合演算について述べる。

非多様体位相構造を用いた形状特徴表現

併合形状では、プリミティブの位相構造が保持されている。この考え方を発展させると、形状特徴で表現したい形状、すなわち、工学的に意味があるすべての部分が境界表現の位相要素の集合として得られるような位相構造を保持しておけば、形状特徴を適切に保持することができる。

図 5.14 に、形状特徴を保持した位相構造を示す。この図では、形状特徴と併合形状の volume との関係を示している。併合操作では、形状特徴表現に必要な位相要素は消去しないので、形状特徴間の干渉が生じても以前の形状特徴の情報が失われることはないし、必要に応じて形状特徴を併合形状から抽出することもできる。

ここでは、プリミティブは任意の非多様体形状モデルでよいので、形状特徴は、面、線、点であってもよい。したがって、体積特徴だけでなく、面特徴や線特徴なども扱うことができる。また、形状特徴抽出で獲得された形状特徴も同様に保持することができる。

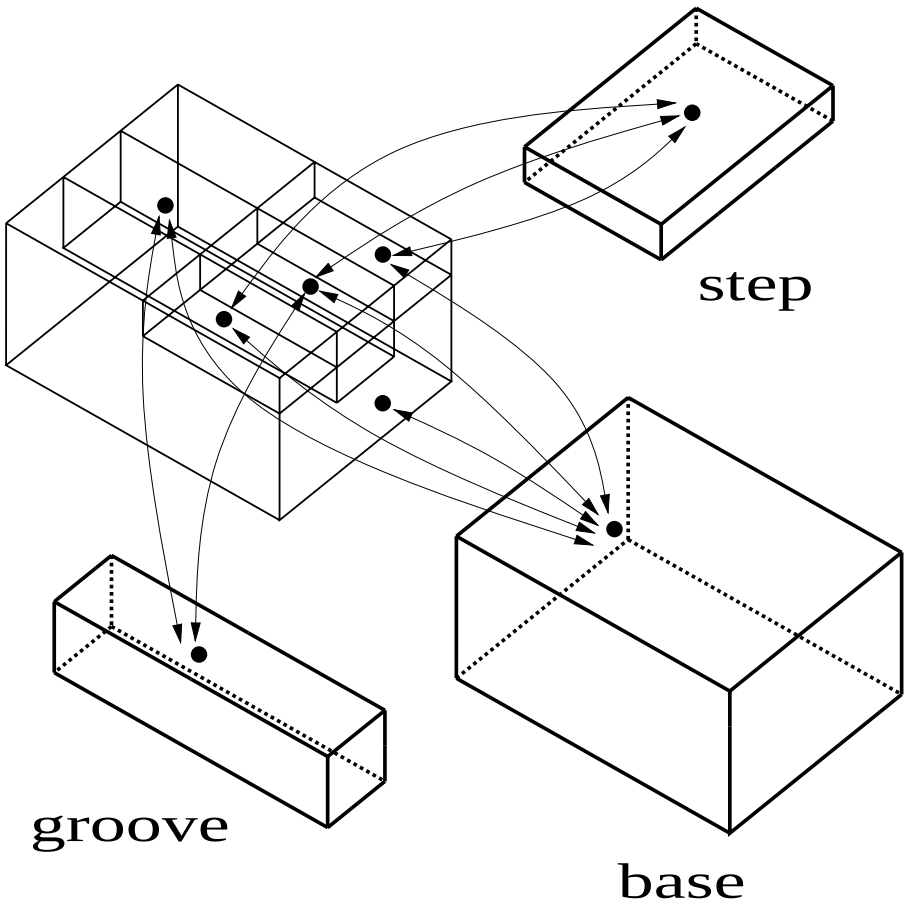


図 5.14: 形状特徴を保持するための位相構造.

5.4 併合操作と抽出操作に基づく集合演算

本節では、併合操作と抽出操作に基づく集合演算について説明する。ここでは、基本的な考え方について示し、詳細な実装手段については第８節以降で述べるものとする。

なお、本章でいう集合演算とは、本論文の「非多様体形状モデルの定義」の章で定義した集合演算を示し、和集合、差集合、積集合に応じて、$\oplus$, $\ominus$, $\otimes$ と書かれるものとする。

5.4.1 保持される形状データ

まず、本研究で提案する集合演算操作の構成を図 5.15 に示す。ここでは、プリミティブ A,B,C に対して、集合演算 $A \oplus B \ominus C$ を施す操作を示している。図において、上段は元のプリミティブ、中段は保持される形状データ、下段は抽出される形状である。

上段は、位相要素 f_a, f_b, e_c を持つプリミティブ A, B, C である。これらのプリミティブから、中段の stored data で示された三つの表現を作成する。一つは併合形状 (merged model) で、位相構造として保持されるものである。また、それ以外に、$(A \ominus B \oplus C)$ を表すブール式 または CSG ツリー、および、元のプリミティブ A, B, C と併合形状モデルとの関係を記述するための $A = \{f_a = \{f_1, f_2\}, ...\}$ といった表現を保持しておく。この表現は、「元のプリミティブは $\{f_a, ...\}$ といった位相要素の集合で構成され、プリミティブの面 f_a は、併合形状モデルでは $\{f_1, f_2\}$ に分割されている」という意味である。また、ブール式は併合形状から目的の形状を抽出するための記述であり、通常は集合演算の履歴が保持される。

併合形状から、集合演算を評価して形状を取り出すためには、中段の stored data に示した三つのデータがあれば十分である。集合演算 $(A \ominus B \oplus C)$ で得られるべき形状は、図 5.15 の下段に示した抽出形状 (extracted shape) である。評価形状は、併合形状モデルから適当な位相要素を選択することによって得られる。また、必要に応じて境界位相要素を抽出することによって評価形状の境界位相要素 (boundary elements) が得られ、表示データなどとして用いることができる。

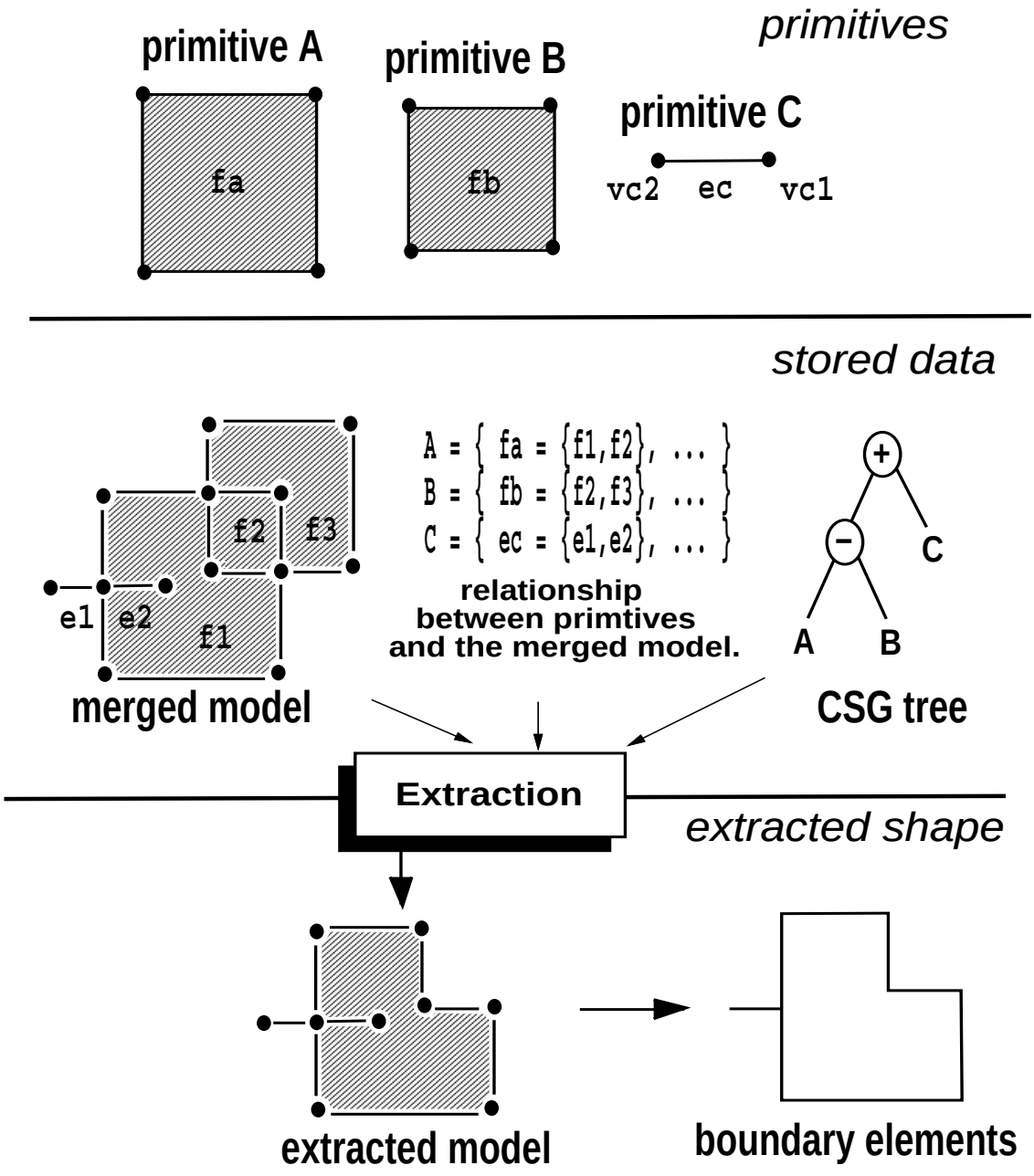


図 5.15: プリミティブの併合と抽出

5.4.2 併合形状モデルの位相構造

併合形状モデルは、併合操作でプリミティブを埋め込んでいくことによって作成される。ここでは、併合形状モデルがどのような位相構造を持つかについて、図 5.16 を用いて説明する。この図は、プリミティブA, B, Cを順次併合していった場合の併合形状モデルとプリミティブを示している。

併合操作でプリミティブを埋め込むときには、併合形状とプリミティブの位相要素間での干渉計算を行ない、二つの形状モデルを併合する。この際、位相要素の分割と、一致する位相要素の併合は許されるが、位相要素の消去は行なわれない。この例では、プリミティブA, Bを構成していた face をそれぞれ F_A, F_B 、またワイヤフレームのプリミティブCの edge を E_C としているが、これらは図に示したように、分割されてはいるものの、併合形状モデルには残されており、併合形状モデルの位相要素の集合として記述することができる。

このような位相構造を保持し、また元のプリミティブとの対応関係を保持しておけば、併合形状においてプリミティブの情報を残すことができる。

5.4.3 位相要素の抽出

ところで、併合形状から和集合、差集合、積集合を取り出すためには、集合演算を評価することが必要である。ここではまず、これらの位相要素を取り出す方法について説明する。

プリミティブの記述

図 5.17 にその考え方を示す。図 5.17 (a) は、図 5.11 に示したプリミティブ A, B の併合形状である。この形状で面に囲まれた閉空間をそれぞれ、図 5.17 (b) に示すような3つの volume、V_1 , V_2 , V_3 とする。

まず、併合形状の volume とプリミティブとの関係を考えれば、Aの占める空間を $|A|$、その閉包を $[|A|]$ とかくとき、A, B は、

$$|A| = [|V_1| + |V_2|]$$

$$|B| = [|V_2| + |V_3|]$$

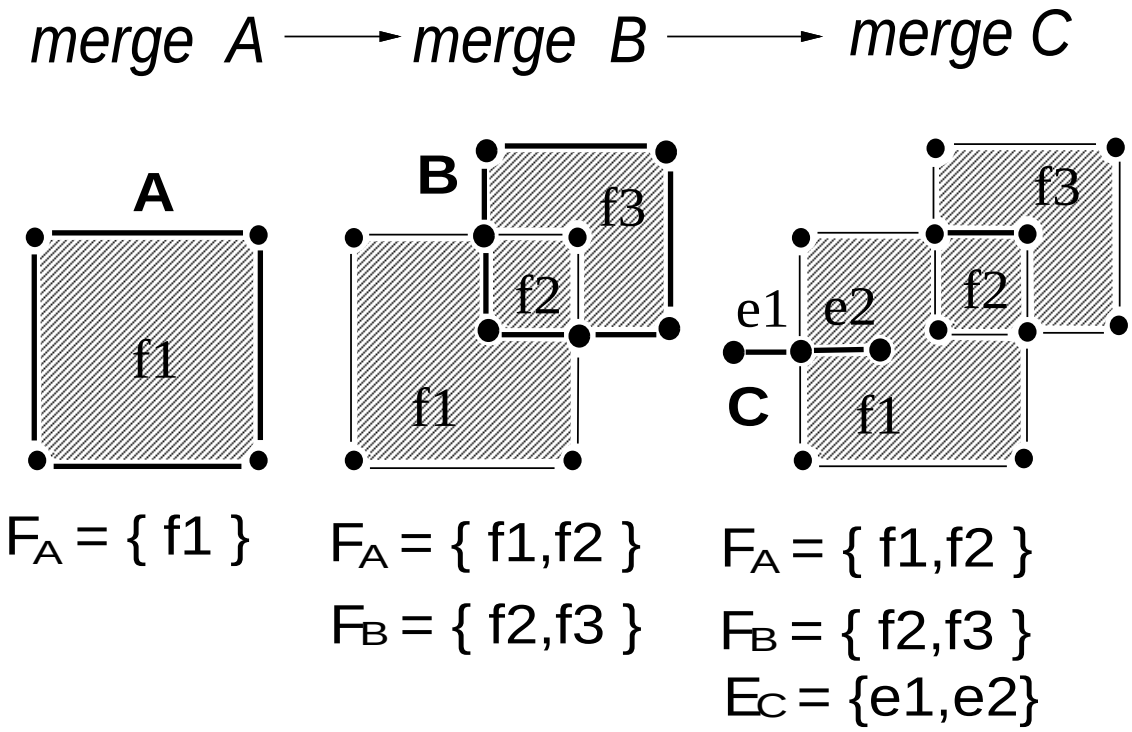


図 5.16: 併合形状とプリミティブの関係

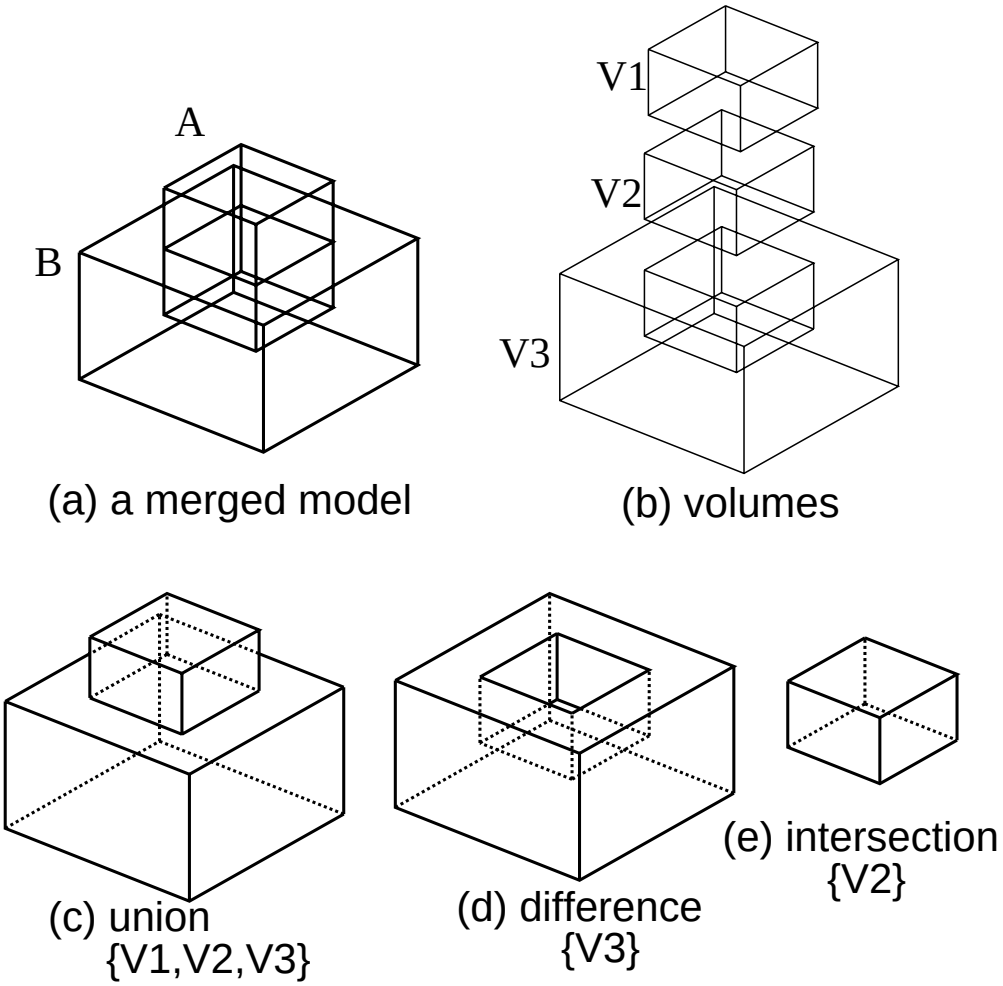


図 5.17: 和集合、差集合、積集合の抽出.

と記述することができる。したがって、本論文で定義した和集合、差集合、積集合をそれぞれ $\oplus$, $\ominus$, $\otimes$ と書くと、それらに対応する volume の集合は次のような単純な演算で求めることができる。

$$|A \oplus B| = |A| + |B| = [|V_1| + |V_2| + |V_3|]$$

$$|B \ominus A| = |B| - |A| = [|V_3|]$$

$$|A \otimes B| = |A| \cap |B| = [|V_2|]$$

これらの占める形状の境界位相要素を取り出した形状を図 5.17(c) (d) (e) に示す。

このような集合演算の評価は、プリミティブの個数が 2 個以上の場合であっても、上記の演算を順次施していくことで、評価形状を得ることができる。また、ここではソリッドの場合について述べたが、サーフェス形状であっても、上記の議論で volume を face に置き換えることで対応することができる。抽出操作に関するさらに詳細な議論に関しては、第 8 節で論じる。

5.4.4 プリミティブと併合形状の対応関係の記述法

ここでは、プリミティブと併合形状の関係がどのように記述できるかについて述べる。

併合形状においては、形状モデルが本来持っていた位相要素は分割されることはあるが、消去はされることはない。したがって、元のプリミティブのすべての位相要素は、少なくとも一つの併合形状の位相要素と対応させることができる。

プリミティブの位相要素と併合形状の位相要素の関係

図 5.18 はプリミティブ A と併合形状モデル M について示している。この例を用いて、プリミティブ A を併合形状モデル M の部分形状として保持しておくためにはどのような関係を保持することが必要かを説明する。なお、この図において、プリミティブ A の位相要素は斜体で、併合形状 M の位相要素はゴシックで示している。

プリミティブ A と併合形状モデル M との関係を考えてみると、プリミティブ A の面 f_1, f_2 は、併合後ではそれぞれ $\{\mathbf{f_1}, \mathbf{f_2}\}, \{\mathbf{f_3}, \mathbf{f_4}\}$ に分割されている。そこで、

$$f_1 \leftrightarrow \{\mathbf{f_1}, \mathbf{f_2}\}$$
$$f_2 \leftrightarrow \{\mathbf{f_3}, \mathbf{f_4}\}$$

という関係を記述することにより、面 f_1, f_2 を併合形状モデルの部分形状として保持することができる。volume, edge, vertex に関しても同様に元のプリミティブと併合形状を関係付けることができる。

一般に、併合後の形状モデルの位相要素 ϵ_m がプリミティブが併合前に持っていた位相要素 ϵ_p と次の関係を持つとき、ϵ_m と ϵ_p が相互に関係付けられるものとする。ここで、$|\epsilon_m|$ は、ϵ_m の占める点集合を意味する。

- $dim(\epsilon_m) = dim(\epsilon_p)$
- $|\epsilon_m| \subseteq |\epsilon_p|$

さらに、すべての位相要素について関係を双方向で記述するために、干渉によって生成された稜線や頂点も考慮する。図 5.18 の併合形状においては、$\mathbf{e_5}, \mathbf{e_6}$ が他の面との干渉で生じた edge である。これらは明らかにプリミティブ A に起因して生成されたものであるが、プリミティブ A には存在しない edge である。そこで、プリミティブ A の位相要素として、Ie_1, Ie_2 を仮想的に考え、

$$Ie_1 \leftrightarrow \{\mathbf{e_5}\}$$

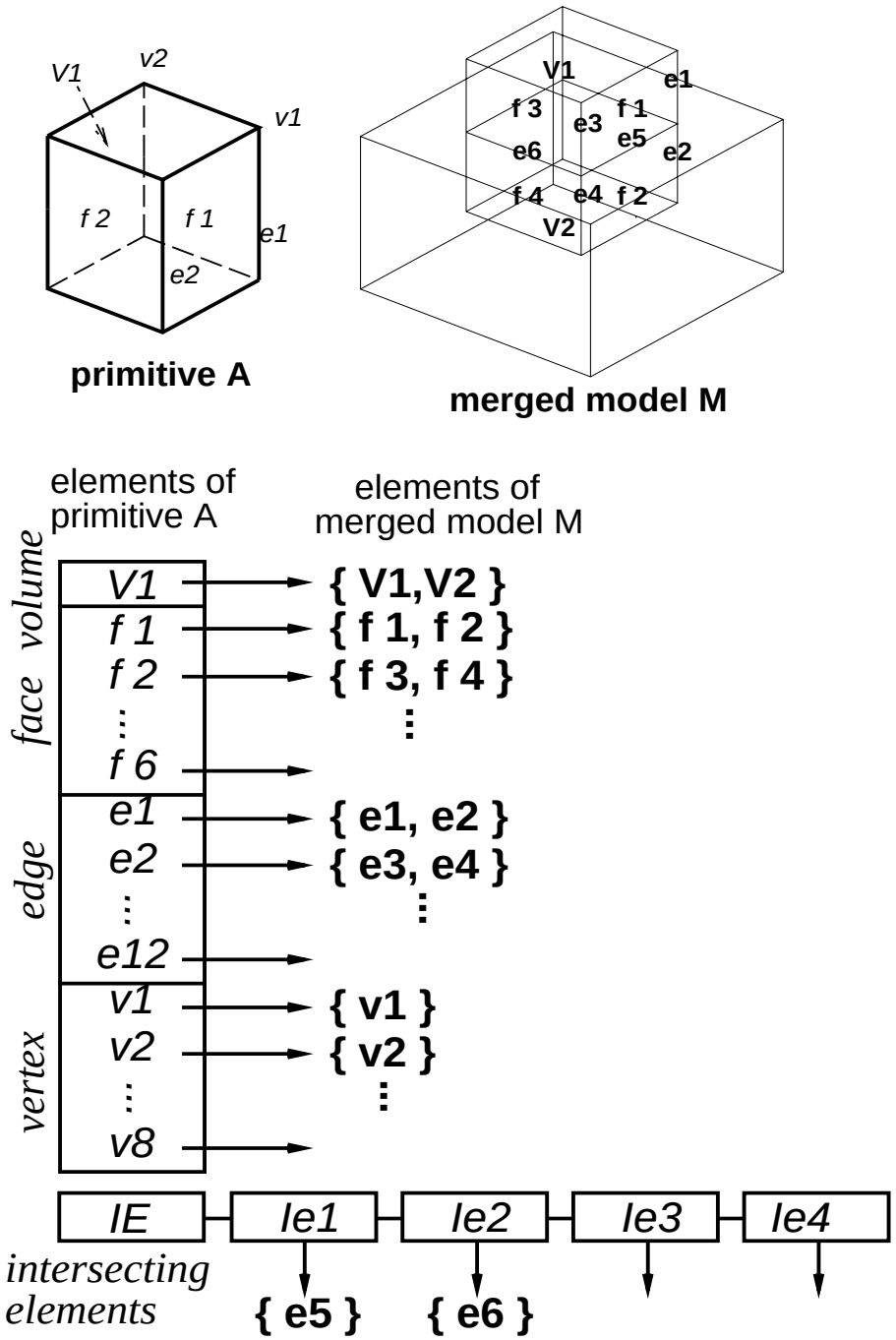


図 5.18: プリミティブを保持するためのデータ構造と、併合形状との対応.

$$Ie_2 \leftrightarrow \{\mathbf{e_6}\}$$

のような関係を記述する。

このような干渉位相要素を保持する理由は、併合形状の位相要素の由来を明確に記述しておくためである。例えば、edge e_5 がプリミティブ A, B の干渉として生じたことがわかっているならば、形状生成の過程で A, B の一方が除去されたとき、e_5 が存在理由を失い、除去される対象であることがわかる。このような情報は後の節で述べる試行錯誤的形状生成の支援において大変有用なものとなる。

関係記述のためのデータ構造

以上述べたような関係を保持するためのデータ構造は図 5.18 に示されている通りである。プリミティブ A のための記述としては、vertex, edge, face, volume の各個数分のスロットと、干渉位相要素のリストを保持するためのスロットがあり、それぞれ斜体で示されている。干渉位相要素の個数は他のプリミティブとの干渉によって決まるので、干渉位相要素は付加や除去が容易なように、図の IE で示したリスト構造によって保持されている。

これらのスロットと、太字で示した併合形状の位相要素との関係を記述し、管理しておけば、併合形状からプリミティブAに相当する位相要素のみを部分形状として抽出することができる。プリミティブ A の位相構造は、関係付けられた併合形状の位相要素を参照すれば得られるので、位相データとしては併合形状のみを保持しておけばよい。

5.4.5 併合操作/抽出操作のまとめ

併合操作と抽出操作について整理しておく。

併合操作はモデリング空間にプリミティブを埋め込む操作であり、プリミティブの位相構造を反映した併合形状モデルが保持される。併合操作では以下の処理が行なわれる。

- プリミティブを構成していた各位相要素が、併合形状モデルの位相要素の和として得られるような位相構造を作成し、
- 元のプリミティブを、図 5.18 に示したようなデータ構造によって併合形状モデルと関係付ける。

埋め込むプリミティブは、本論文で定義した任意の非多様体形状モデルが対象となる。

抽出操作は、対象となる形状を、併合形状から位相要素の集合として取り出すための操作である。ここでは、併合形状モデルからプリミティブを取り出したり、また、集合演算式を評価して得られる形状を取り出す操作として用いられる。抽出操作は、併合形状モデルの各位相要素とプリミティブとの関係を調べることによって、比較的単純な演算によって実現することができる。

なお、併合操作と抽出操作の実装方法を含むさらに詳細な議論は第８節以降で述べる。

5.4.6 CSG/Brep ハイブリッド表現

ここでは、併合/抽出操作で実現される集合演算を少し別の観点から論じてみる。併合/抽出操作では、境界表現の併合形状と CSG 表現の両方を保持しており、それらを併合形状とプリミティブの関係付けの記述が橋渡ししている。したがって、このデータ保持法は、境界表現と CSG 表現を密に関連付けた CSG/Brep ハイブリッド表現 [Wilson86] となっている。

境界表現と CSG 表現の二重表現

従来の形状モデラでも境界表現と CSG 表現の両方を形状データとして保持することはあったが、それらは独立に保持されていた。その場合、境界表現と CSG 表現との整合性を保持するという観点から、どちらかをマスターモデルとする必要があった。

図 5.19 に、従来の形状モデラにおける境界表現と CSG 表現の関係を示す。(a) では、CSG 表現がマスターモデルとなっている例で、必要に応じて CSG 表現から境界表現モデルを生成する。(b) は、境界表現がマスターモデルである形状モデラの構成例で、CSG 表現は境界表現を生成するための履歴を管理する補助的なデータとなっている。

また、[Gomes91] のように、 CSG 表現のプリミティブを形状特徴として表現するために、境界表現と CSG 表現を関連付けて保持する研究もある。しかし、そのような場合、境界表現と CSG 表現との整合性を保持するために多大な労力を払う必要がある。

本論文の図 5.15 で示した形状データにおいては、境界表現と CSG 表現の両方を保持したものとなっているが、両者が位相要素間の記述を介して、密に結びついた表現になっている。そのため、これまでの表現法のように、境界表現と CSG 表現のどちらがマスターモデルであるかという問題はそれほど意味がなくなり、両者の整合性は自動的に保持される。

Wilson はこのような境界表現と CSG 表現を密に関連付けた表現法を CSG /Brep ハイブリッド表現と呼んでいるが [Wilson86]、図 5.15 で示した表現法は、この CSG /Brep ハイブリッド表現を実現する一手法と見做すことができる。

Wilson の CSG/Brep ハイブリッド表現

Wilson は、 CSG/Brep ハイブリッド表現を実現する手法の一つとして、図 5.20 に示したような、非多様体位相を用いる方法を提案した。Wilson の方法では、 CSG 表現

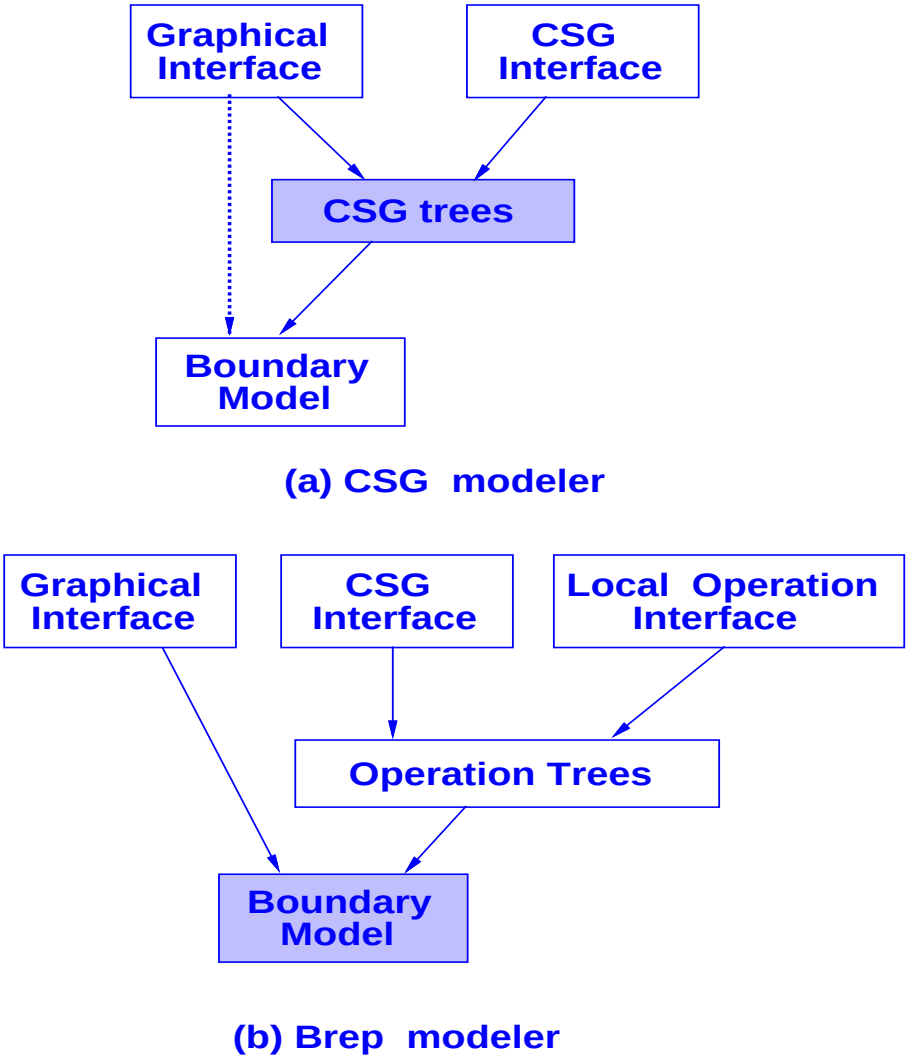


図 5.19: 従来の CSG モデラと境界表現モデラの構成

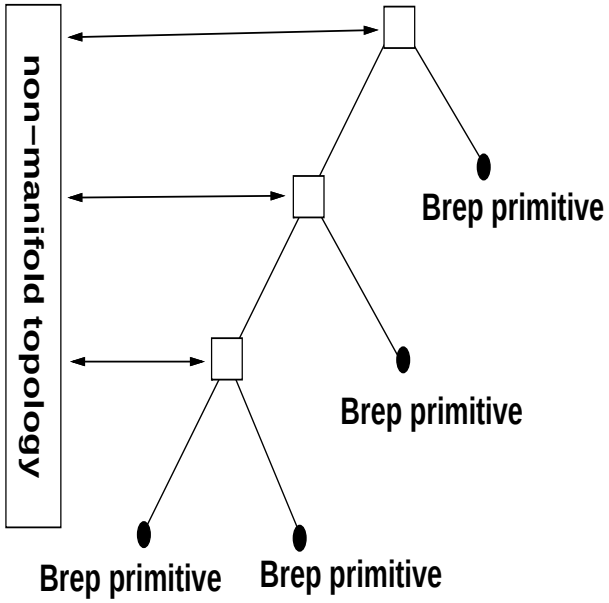


図 5.20: Wilson の提案したハイブリッド表現

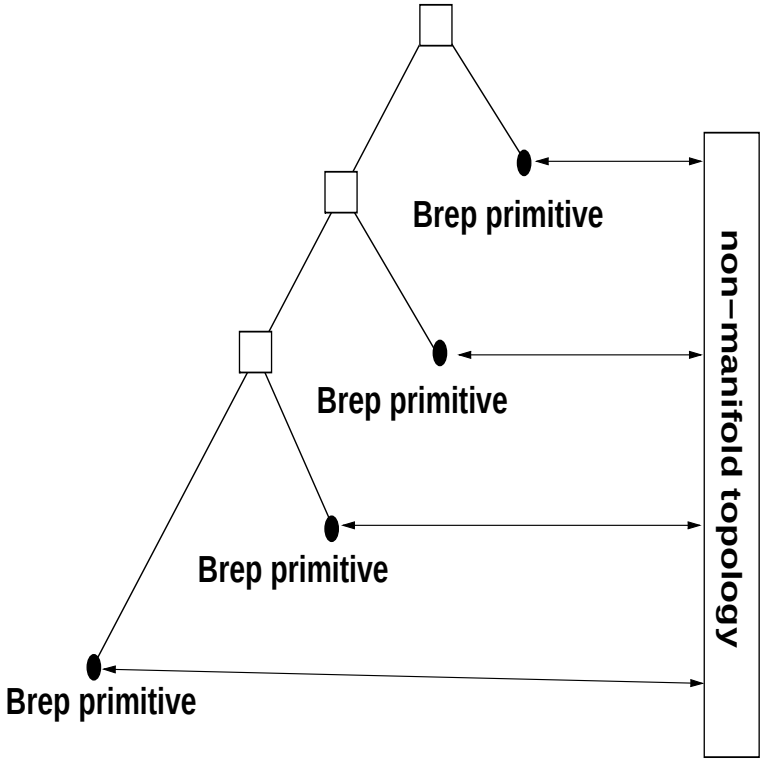


図 5.21: 併合・抽出法における境界表現と CSG の関係.

の節 (node)、すなわち、集合演算の過程で生成される中間形状を非多様体位相と関連付けた。

Wilson の研究は CSG/Brep ハイブリッド表現の実現に非多様位相が利用できることを示唆した点で注目すべきものであるが、実用上は対応関係を管理する負荷が大きいという問題がある。試行錯誤的な形状修正操作を考えてみると、Wilson の方法では、CSG と Brep の関連付けが集合演算の順序に依存してしまい、たとえば、初期のプリミティブを取り除く操作を行なった場合、そのプリミティブを子に持つすべての節の記述を書き換えなければならないため、ハイブリッド表現の整合性を保つための負荷が大きくなってしまう。本論文で提案した抽出操作を用いれば CSG 表現の節に対応する中間形状が容易に得られるので、節の記述の管理に負荷をかけるのは無駄である。

併合/抽出操作に基づく CSG/Brep ハイブリッド表現

本手法を CSG/Brep ハイブリッド表現の実現手段としてとらえた場合は、図 5.21 に示したように、CSG 表現の節ではなく葉 (leaf) が非多様体位相と関連付けられ、節に相当する部分は必要に応じて抽出操作を施して生成される。

この方法だと、Brep と CSG の関係付けは個々のプリミティブに関して独立になされており、プリミティブ間の相互関係を考慮する必要がない。したがって、特定のプリミティブに対する形状操作で影響の及ぶ範囲が限定されるという利点がある。さらに、本手法では、集合演算の評価された形状を陽に持たず、必要に応じて任意のブール式を評価した形状を得るための抽出操作を備えている。そのため、CSG の節に相当する中間形状も容易に得ることができるので、本論文の表現法は、Wilson の方法よりも汎用的なものとなっている。

以上から、併合/抽出操作に基づく形状データ表現は、対応関係の管理が容易であり、より汎用的な CSG/Brep ハイブリッド表現を実現するための方法として優れたものとなっていることがわかる。

5.5 集合演算の取消操作と修正操作

本節では、第２節で述べたソリッドモデルの問題点のうち、集合演算の取り消しや修正が演算順序に依存しないで行なえる手法を、併合操作と抽出操作の応用として実現できることを示す。

5.5.1 演算順序に依存しない取消操作

併合操作では、プリミティブが併合形状モデルの位相要素の集合として直接に表現されている。したがって、以前併合した形状モデルを取り除きたい場合、修正すべき箇所をただちに知ることができる。したがって局所変形操作によって特定のプリミティブに依存する位相要素のみを取り除くことにより、極めて処理の軽い取消操作を実現することが可能となる。

図 5.22を例に取り、取消操作を実現するための処理を示す。この図は、プリミティブ A, B, Cを併合して作成された併合形状モデルを示している。併合形状モデルにおいて、斜線で示された部分が取り除くべき位相要素である。併合形状モデルから、プリミティブAを取り除く手順は以下の通りである。

１. 関連する位相要素の除去

まず、プリミティブAのみと関係付けられている位相要素を併合形状モデルから削除する。削除は、volume, face, edge, vertex の順で、オイラー操作を用いて行なわれる。図 5.22 で、併合形状 (merged model) の face F_1, F_2 とプリミティブとの関係についてみると、

$$F_1 \leftrightarrow \{F_a\}$$
$$F_2 \leftrightarrow \{F_a, F_b\}$$

のような関係がある。ここで、プリミティブAがなくなると F_a が消滅し、対応関係は以下のように書き換えられる。

$$F_1 \leftrightarrow \{\}$$
$$F_2 \leftrightarrow \{F_b\}$$

この場合、F_1 は存在理由が失われているので、F_1 は併合形状モデルから削除すべきだということがわかる。同様にして、vertex: v_1, v_2, v_3, v_4 および点線で示した edge も、プリミティブAがなくなると存在理由がなくなるので、併合形状から除去される。

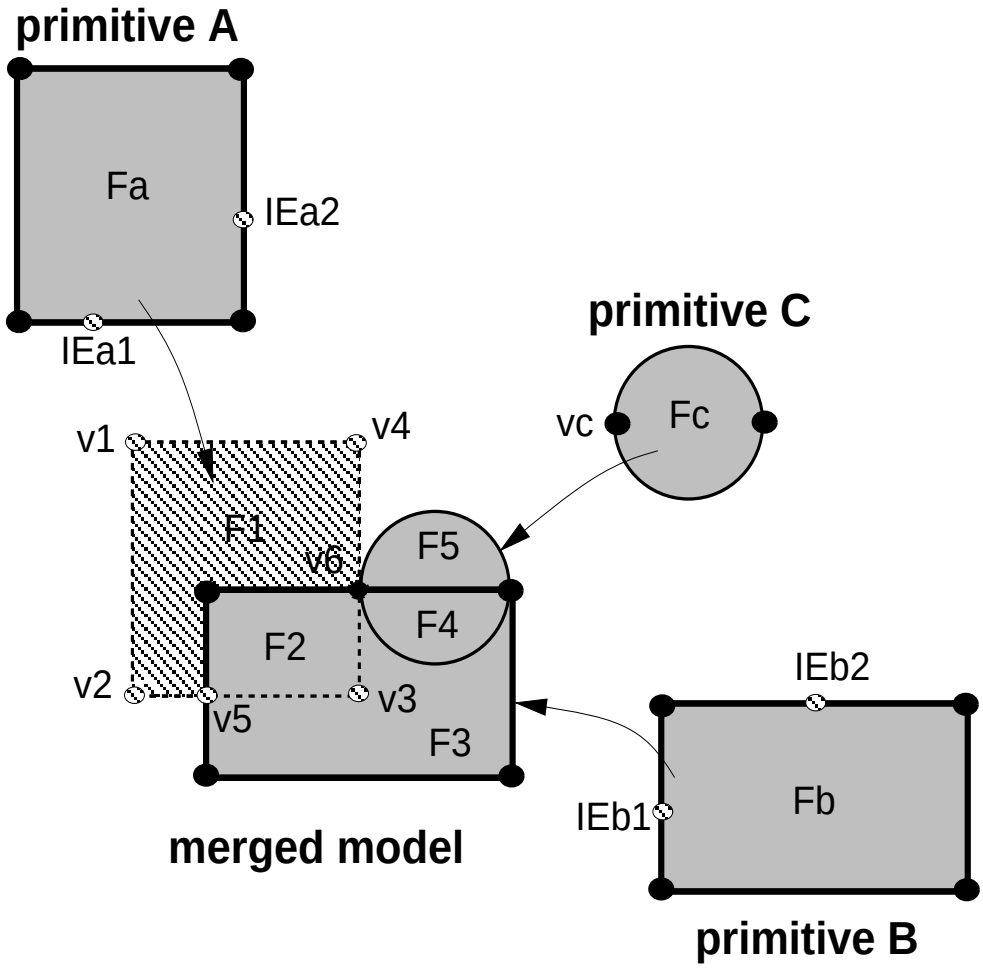


図 5.22: プリミティブAの併合形状からの除去.

次に、干渉位相要素を除去する。そのために、プリミティブAとそれ以外の唯一つのプリミティブとの干渉で生じた vertex と edge を削除する。v_5 と v_6 について考えると、それぞれ対応関係は、

$$v_5 \leftrightarrow \{IEa_1, IEb_1\}$$
$$v_6 \leftrightarrow \{IEa_2, IEb_2, v_c\}$$

である。ここで、IE はプリミティブの記述として保持されている干渉位相要素である。干渉位相要素は干渉する相手がいなくなったときに存在理由を失うから、v_5 は IEa_1 が消失するので除去されなければならない。一方、v_6 は、IEa_2 がなくなると、関係が

$$v_6 \leftrightarrow \{IEb_2, v_c\}$$

に書き換えられて保持される。

2. ブール式の変更と再評価

次に、抽出操作のためのブール式を書き換える。ここでは、削除すべきプリミティブの効果を無効にするように式を書き換える。そのためには、和集合と差集合においては、消去されたプリミティブを空集合とみなし、また積集合では消去されたプリミティブをモデリング空間全体とみなせばよい。したがって、取消操作における集合演算の評価規則は次のようになる。ここで、X は削除されるプリミティブ、Φ は空集合である。

$$P \oplus X = X \oplus P = P,$$

$$P \ominus X = P,\ X \ominus P = \Phi,$$

$$X \otimes^{+} P = P \otimes^{+} X = P$$

$$X \otimes^{-} P = P \otimes^{-} X = P$$

ブール式にこのような修正を施した後、集合演算の評価を行ない、評価形状を得る。

以上の処理を施すことによって、任意のプリミティブの除去を、集合演算の順序に関係なく、干渉計算を一度も行わずに局所変形処理だけで行うことができる。この処理では、たとえ大規模な形状モデルであっても修正すべき部分が限定されるので、短時間で処理を行うことができる。

5.5.2 取消操作の例題

ここで述べた取消操作を用いて、形状修正を行なった例を示す。使用した計算機は RC 6000/730 である。(IBM RS 6000 シリーズでは低速な計算機である。)

図 5.23 は、比較的大規模な形状モデルから一つのプリミティブを取り消して形状の再評価を行なったものである。この例題では、５０個のプリミティブの和集合から２５番目に足されたプリミティブを取り除く操作を施した。本手法では、プリミティブを取り消して形状の再評価を行なうのに要した時間は、CPU time で 0.25 秒であり、極めて高速な形状修正が実現できていることがわかる。同等な操作を undo 操作で行なおうとすれば、２５番目以降の集合演算をすべて取り消してやり直す必要があるため、数十秒を要する。

図 5.24 では、差演算の場合の取消操作を示している。図 5.24 (a) では、５０個のプリミティブからなる形状モデルＡと１２角柱Ｂを示しており、図 5.24 (b) では、集合演算 Ａ－Ｂ を実行した結果を示している。この差演算を取り消して、形状Ａ－Ｂから元の形状Ａを復元する操作を行ったものを (c) に示す。本手法では消された部分もデータ構造に保持されているため、形状の復元は非常に高速にでき、(b) の形状から (c) の形状を得るのに要した CPU time は、0.50 秒である。

図 5.23: 取消操作の例題１ （CPU Time : 0.25 秒）

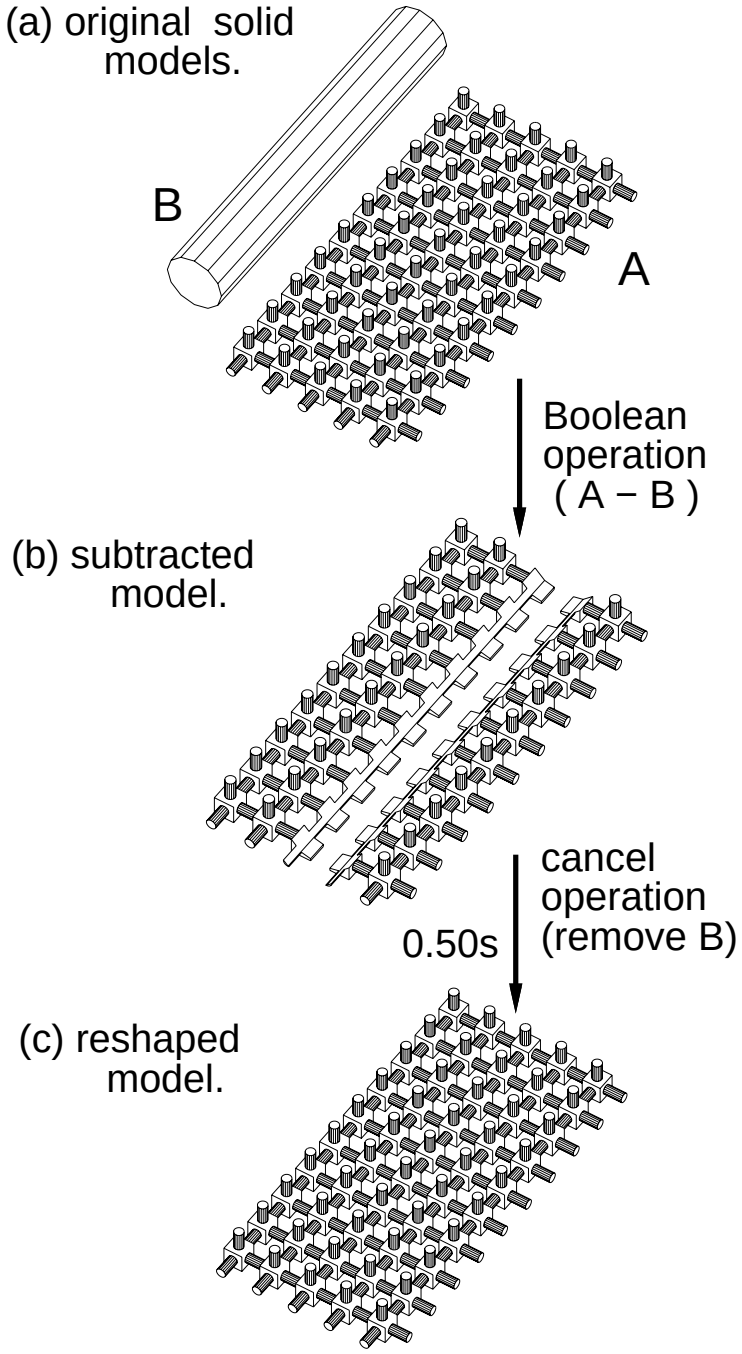


図 5.24: 取消操作の例題 2 (CPU Time : 0.50 秒)

5.5.3 修正操作

次に、修正操作について考える。修正操作は、以前適用したプリミティブを別のプリミティブで置き換える操作である。例えば、形状モデルの円柱穴を別の径を持った円柱で置き換える操作などがこれにあたり、パラメトリック設計で重要な操作である。

集合演算では、プリミティブ A を取り消したのちに修正したプリミティブ A' を後から足したとしても、一般に、

$$(A' \oplus B) \ominus C \neq (B \ominus C) \oplus A'$$

であるため、正しい形状ができるとは限らない。修正された形状を正しく得るためには、A を A' に置き換えた $(A' \oplus B) \ominus C$ を計算することが必要である。

この場合でも併合操作が演算順序に依存しないという性質を利用することによって、効率的な修正操作が可能になる。プリミティブ A を A' に置き換える修正操作の手順は以下のようになる。

1. プリミティブ A を取消操作によって、併合形状モデルから取り除く。
2. プリミティブ A' を併合形状モデルに併合する。
3. 抽出操作のためのブール式において、A を A' に置き換える。たとえば、全体形状が $R = A \oplus (B \ominus C)$ と表現されており、A が A' に変更されたとすると、ブール式は $R = A' \oplus (B \ominus C)$ と変更される。
4. 修正されたブール式を抽出操作によって評価し、評価形状を取り出す。

この修正操作では、どのプリミティブを修正する場合であっても、集合演算をやり直す回数が一回だけである。したがって、プリミティブの個数が多くなった場合でも、集合演算の順序によって再評価すべき集合演算の数が増大することがないため、短時間で形状修正ができる。

5.5.4 修正操作の例題

図 5.25 は、プリミティブ１５０個を組み合わせて生成したプリンタヘッドの形状モデルである。この形状モデルにおいて、いくつかの部分に対して取消操作と修正操作を適用してその計算時間を測定した。ここに示したように、本手法を用いることにより

非常に短時間で形状修正ができることがわかる。この計算時間は、従来手法のように集合演算の再実行を行って形状修正を行った場合に比べて、２桁ほど処理時間が早くなっている。従来の方法ではこの例題程度の複雑さであっても、形状のごく一部を変更するだけでも約５０秒の計算時間がかかっていた。

以上のことから、本手法を用いることにより、比較的大規模な形状モデルでも試行錯誤的な形状生成が可能な環境を提供することが可能となることがわかる。

図 5.25: 複雑な形状モデルに対する取消/修正操作.

5.6 プリミティブの優先順位の反映法

集合演算は適用順序に依存する演算である。そのため、幾つかのプリミティブに集合演算を施して目的形状を作るには、集合演算を施す順序に十分気を配らなければならないという問題があった。このことは、設計者が寸法や形の決まった部分から順に形状を作っていくためには障害となっていた。

ここでは、集合演算の順序依存の問題を解決するために、

- プリミティブに優先順位を設け、優先順位を反映させた形状評価を行なう手法、
- 集合演算の影響の及ぶ範囲を限定することによって、不用意な干渉を避ける方法

の二つの手法を提案する。

5.6.1 優先順位を考慮した抽出操作

通常の集合演算では、演算を適用した順に形状を評価していく。この場合、後に集合演算を施されたプリミティブほど優先されることになる。しかし、プリミティブに優先度を設定することができれば、複数のプリミティブが空間を共有する場合に、どのプリミティブを優先させるかを柔軟に決めることができる。

そこで、プリミティブに優先度を設定することを考える。優先度は足されるプリミティブ（正のプリミティブと呼ぶ）と引かれるプリミティブ（負のプリミティブと呼ぶ）について設定されるものとし、また、優先順位を設定しないプリミティブを優先度０とする。ここで、集合演算において、優先度は次のような意味を持つものとする。

- 優先度の高い正のプリミティブから、優先順位の低い形状を引くことはできない。
- 優先度の高い負のプリミティブに、優先順位の低い形状を足すことはできない。

図 5.26 に例を示す。ここでは、軸形状 (P1 − P2) の軸穴が塞がれないように、この部分の優先順位を高くして集合演算を行なっている。この図において、プリミティブ P_1 の優先度が n のとき、$P_1 : n$ のように書かれおり、優先度 n が大きいほど、優先順位が高いものとする。したがって、プリミティブ P_1, P_2 が優先順位が高く、P_3, P_4 は優先順位が低い。また、集合演算の結果を Rn として示している。このとき、P_1 から P_2

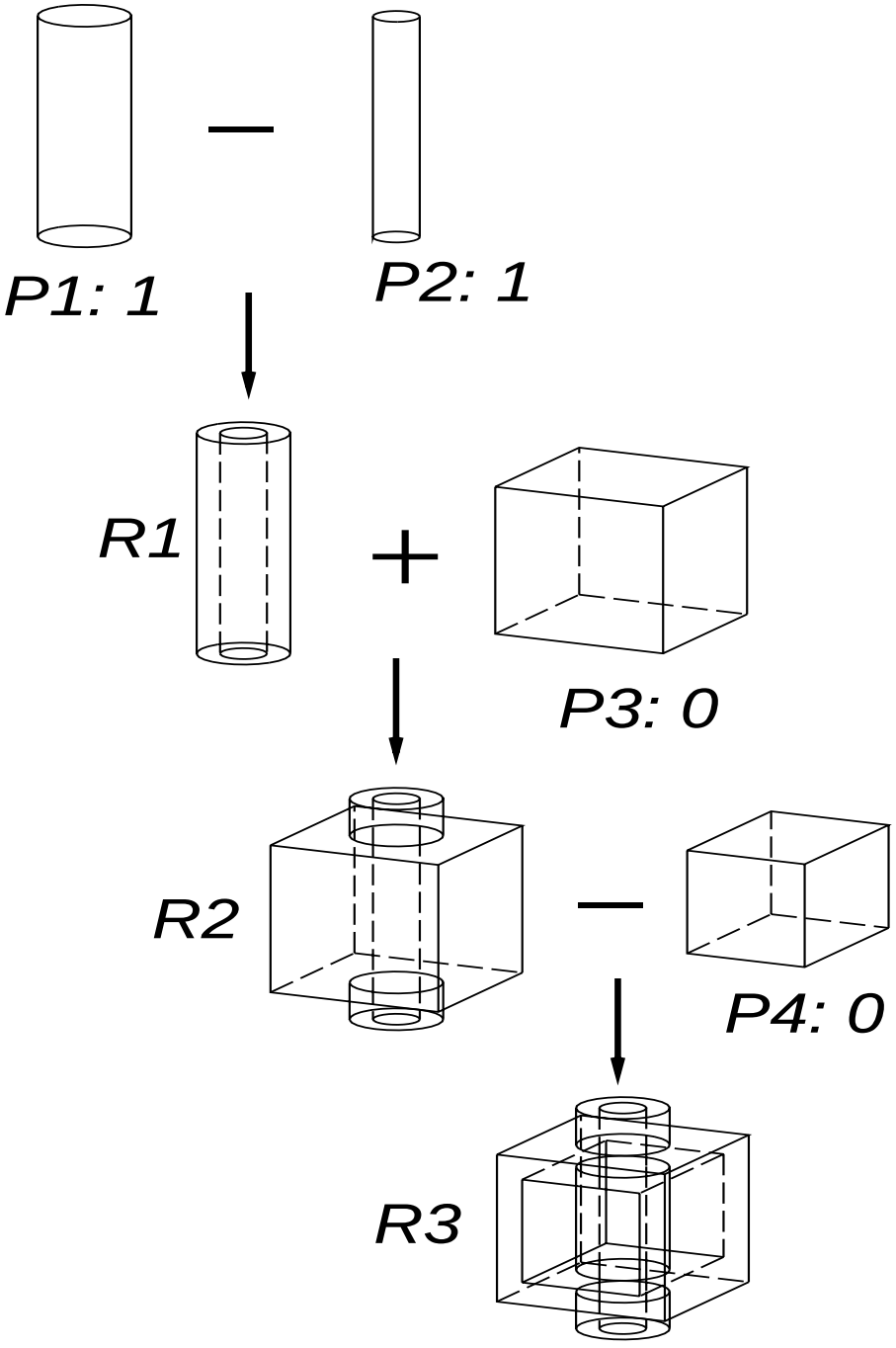


図 5.26: プリミティブの優先度を考慮した集合演算.

を引いて生成された R_1 に P_3 を足す操作を考える。通常の集合演算では、P_3 を足すことによって、軸穴を埋めてしまうが、この場合は P_3 の優先順位が低いので、穴を埋めることができない。したがって、R_2 のように、穴が保存された形状が得られることになる。さらに、P_4 を引く演算を考えてみると、P_4 は P_3 からは形状を差し引くことができるが、優先度の高い P_1 からは引けない。したがって、R_3 のように、軸の部分が保存された形状となる。

このように優先順位を反映させるためには、優先順位の低い順に集合演算の評価を行なっていけばよい。併合・抽出操作に基づく集合演算では、適当な抽出操作を行なえばプリミティブの任意の組み合わせで定義できる形状を得ることができるから、優先順位の問題は優先順位を考慮した抽出操作を定義する問題に帰着させることができる。

そこで、プリミティブに優先順位が設定されているとき、優先順位を反映させた形状モデルを得るためのブール式を算出し、それに基づいて抽出操作を行なうことを考える。

集合演算の適用順序に応じて記述したブール式を B_s とする。B_s は、従来用いられてきた CSG ツリーと同等なものである。このとき、優先順位を反映させた形状モデルを得るためのブール式 B_p を算出する。B_s から、B_p を算出する手順は以下の通りである。

1. プリミティブのうちで優先度が n のもののみを取り出したブール式を B^n とする。B^n は、B_s において優先度が n でないプリミティブをすべて X とおき、次の演算規則を用いて B_s を変形する ことによって得られるものとする。

   - $A \oplus X = X \oplus A = A$,
   - $A \ominus X = A,\ X \ominus A = -A$,
   - $X \otimes^{+} A = A \otimes^{+} X = A$
   - $X \otimes^{-} A = A \otimes^{-} X = A$

2. 優先度を反映させるためには、優先度の低いものを先に評価すればよいから、優先度の最大値を m とするとき、優先順位を考慮した形状評価を行なうためのブール式 ：

   $$\mathrm{B}_p = \mathrm{B}^{\mathrm{o}} \ldots \mathrm{B}^m$$

   を算出する。

B_p に基づいて抽出操作を行なえば、優先順位を反映させた集合演算を評価した結果を得ることができる。このように優先順位が定義できると、設計において重要な部分

の優先順位を高く設定することによって、設計意図を反映しやすい形状定義が可能になる。

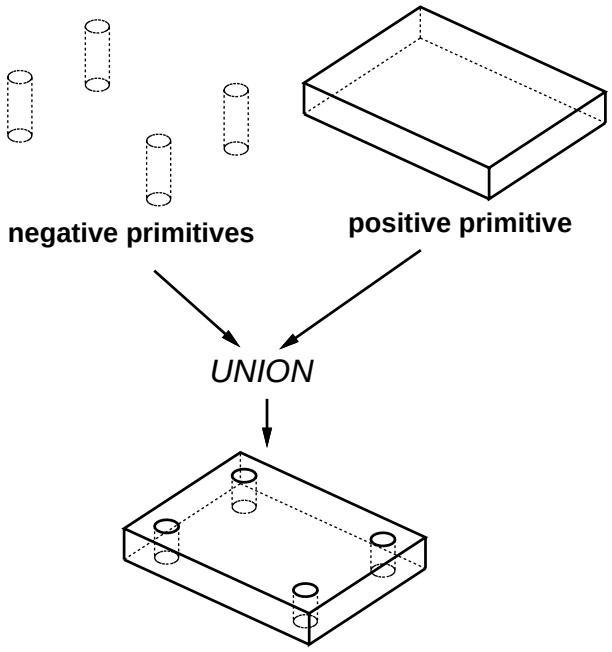


図 5.27: 優先度の高い負の形状との和集合.

また、この方法を用いれば、負の立体も定義できる。たとえば、5.27 ように、穴のモデリングを先に行い、あとから実体部分の寸法を決めて足し合わせたとしても、穴の優先度を実体部分より高めておけば穴は埋められることなく保存される。この手法を用いれば、複数部品の設計において、他の部品との関係から実体であってはならない部分をあらかじめ負の空間として表現しておくこともできる。

5.6.2 局所集合演算

従来の集合演算では、演算は常に形状モデル全体に対してなされていた。しかし、併合/抽出操作では、プリミティブの情報が保持されているので、形状モデルの特定の部分のみに集合演算を適用することが可能である。

図 5.5 の例で考えると、もしプリミティブ D を最後に引くと、軸に相当する A の部分が分断されてしまう。しかし、もし、D をプリミティブ C のみから引くことができれば図 5.28 のような演算が可能となる。ここで、プリミティブ C に対してのみ影響を与えるような集合演算を局所集合演算と呼び、$\ominus_C$ のように書くものとする。

図 5.28: 局所集合演算.

局所集合演算もまた、抽出操作のためのブール式を制御することによって実現することができる。いま、プリミティブ A, B, C からなるブール式を $T(A, B, C) = C \oplus A \ominus B$ とする。ここで、プリミティブ C のみに影響を与える局所集合演算は、

$$T(A, B, C) \ominus_C D = T(A, B, (C \ominus B)) = ((C \ominus D) \oplus A \ominus B)$$

を評価することによって得ることができる。このように、局所集合演算を用いることにより、集合演算の対象範囲を限定することができるので、プリミティブ間の不用意な干渉を回避することができる。

5.7 非多様体形状位相を用いた形状特徴表現

本節では、第２節で述べたソリッドモデルの問題点において、形状特徴に関する問題を非多様体位相構造を用いて解決する方法について述べる。

位相要素の集合による形状特徴表現

境界表現形状モデルにおいて形状特徴を表現する場合、部分形状を形状モデルの位相要素の部分集合として表現する方法が一般に用いられてきた [Shah88]。 本研究でも基本的にはこの方法を用いる。

ここでは境界表現形状モデルの位相要素として volume, face, edge, vertex を考える。すなわち、部分形状はこれらの位相要素の集合として表現される。形状特徴を表現するためには、部分形状として保持したい部分に相当する位相要素の集合が得られればよい。

ここでは、任意の位相要素の集合で表現される形状特徴を一般的に扱うために次のように書くものとする。

- 位相要素を α_λ として表す。
- 形状モデルを構成する位相要素の集合を $\{\alpha_\lambda | \lambda \in \Lambda\}$ とし、これらによって構成される形状モデルを $M(\Lambda)$ とする。
- $M(\Lambda)$ がユークリッド空間に占める点集合を $S(\Lambda)$ とする。

２-多様体形状モデルの問題点

このとき、形状モデルの部分形状がどのように表現できるか考えてみると、部分形状は、形状モデル $M(\Lambda)$ を構成する位相要素の部分集合としてしか表現できないので、表現できる部分形状 F は、

$$F = S(\Lambda_k) \quad (\Lambda_k \subset \Lambda)$$

となるような位相要素の集合 Λ_k が存在するものに限られる。

このことから、従来の境界表現のソリッドモデルによる形状特徴表現の問題点が明らかになる。 従来のソリッドモデルでは、データ構造の制約から、全体形状の境界上に

ある位相要素しか保持することができないので、表現できる形状特徴はその部分集合で表現できるものに限られる。従って、 vertex, edge, face の集合である Λ_k をどのように選んでも、体積特徴を表現することはできないし、また形状生成の途中で失われた位相要素は Λ に含まれないので形状特徴として保持できない。

そこで、本研究では、形状特徴の表現に適した位相構造を持つ境界表現形状モデル $M(\Lambda)$ を生成することによって、形状特徴を保持する方法について提案する。

5.7.1 形状特徴表現のための位相構造

ここで、形状特徴の表現方法を説明するために形状特徴による設計 (design-by-feature) の例を考えてみる。 形状特徴による設計では、形状特徴を組み合わせていくことによって形状モデルを定義する。 ここでは位相構造について説明するため、図 5.29 に示すように、形状特徴A, B, Cを定義していく過程を考える。

ここで問題となるのは、形状特徴A, B, Cがそれぞれ部分形状 $M(\Lambda_A), M(\Lambda_B), M(\Lambda_C)$ として表現できるような境界表現形状モデル $M(\Lambda)$ にはどのような位相構造が必要かということである。図 5.29 の併合形状モデルはそのための位相構造を示している。この図で、F_A は元の形状特徴 Aの face、 f_i, e_i, v_i はそれぞれ併合形状の face, edge, vertex を表している。いま、F_A が形状特徴を埋め込んでいく過程でどのように分割されていくかを考えると、最初は併合形状モデルの f_1 のみに対応しているが、形状特徴 B を埋め込んだ位相構造では面が分割され、$\{f_1, f_2\}$ が対応することになる。さらに、形状特徴 C が埋め込まれた場合は、面が分割されないので、対応は元のままである。このような対応関係は、位相要素の分割が起こる度に関係を付け変えることで整合性が維持できる。

5.7.2 形状特徴の位相

形状特徴がユークリッド空間に占める点集合は位相要素の集合 $S(\Lambda_k)$ として表現できる。 しかし、多くの応用分野では、点集合 $S(\Lambda_k)$ が位相構造を持った方が都合がよい。 たとえば、図 5.30 のように円筒を形状特徴として扱うには、全体を一つの参照単位とするよりもむしろ、 円筒を構成する上面、下面、側面といった単位で扱いたい場合が多い。

このような場合、形状モデルの face に対して直接に「円筒の上面」といった属性を付ける方法が一般的であったが、この方法では face の分割や削除が行われた場合に部分

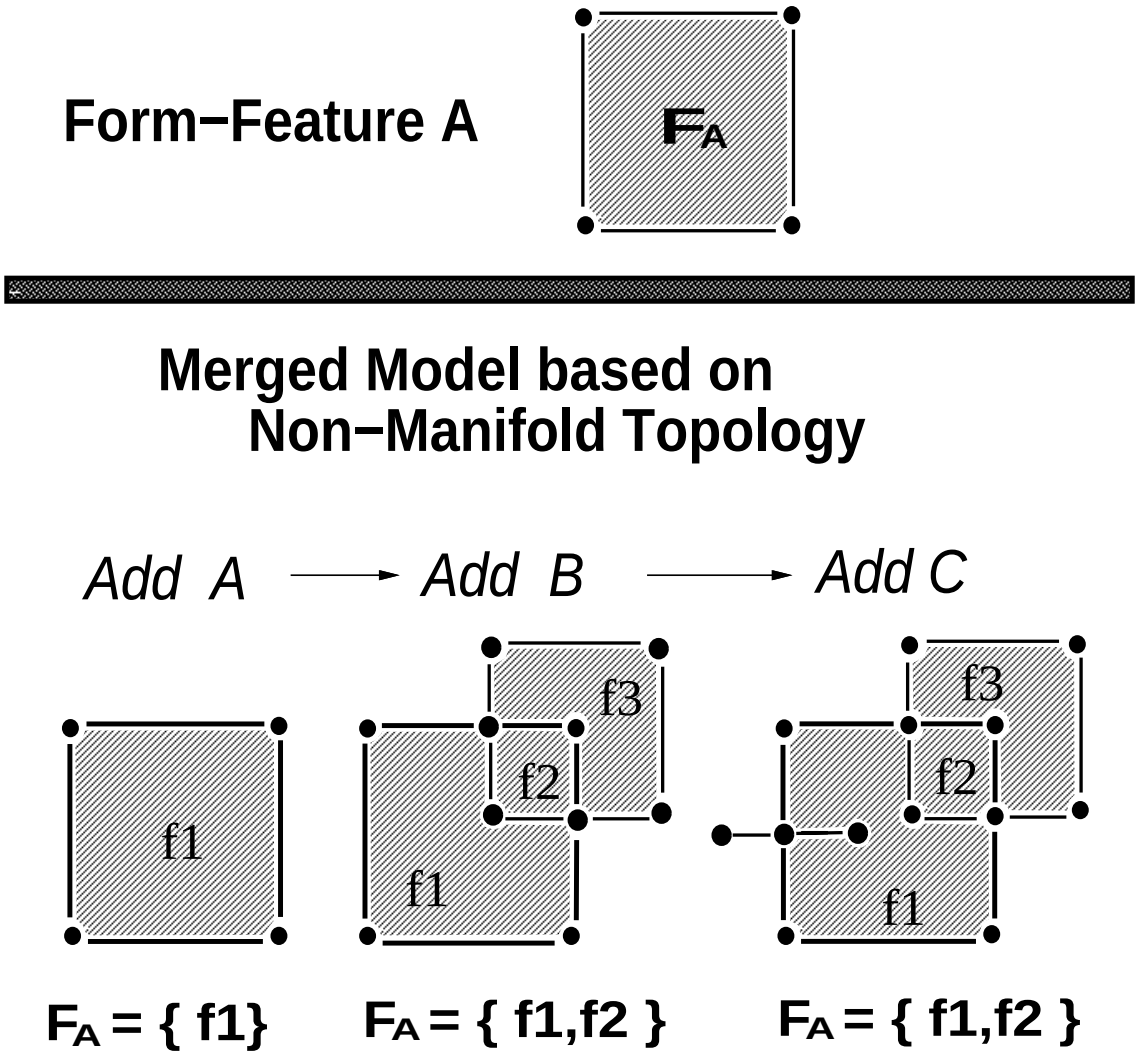


図 5.29: 形状特徴を保持した位相表現.

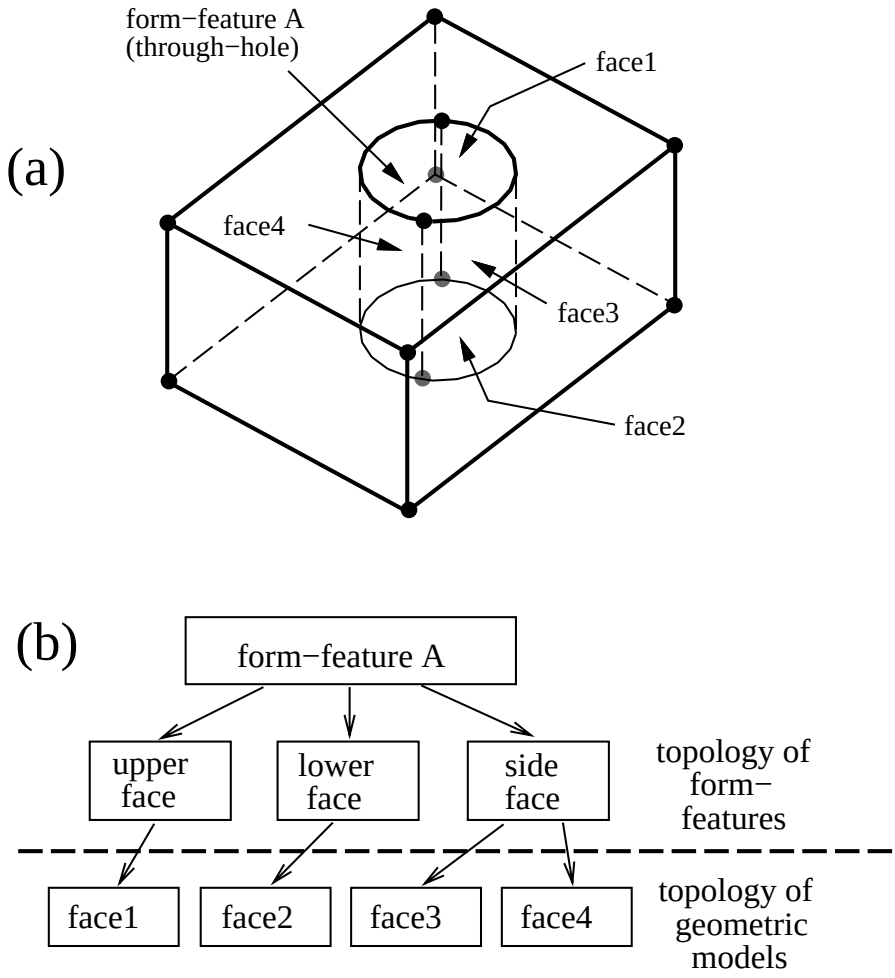


図 5.30: 形状特徴の位相と併合形状の位相の関係.

形状に対する属性を管理するのが大変であるという問題がある。本手法では、属性を「形状モデルの位相要素」ではなく、「形状特徴の位相要素」に持たせることによってこの問題を解決する。

形状特徴 F の占める点集合を「形状特徴の位相要素」の集合 $\{\beta_\gamma\}$ $(\gamma \in \Gamma)$ によって表現することを考える。 たとえば、図 5.30 では形状特徴Aは、一つの volume、 上面、側面、下面の３つの face、およびそれらの境界の edge と vertex によって構成されると考えることができる。 これらの位相要素も点集合であると考えると、適当な形状モデル $M(\Lambda)$ を用意すれば、すべての $\gamma \in \Gamma$ に対して、

$$\beta_\gamma = \{\alpha_\lambda\} \ (\lambda \in \Lambda_{F\gamma} \ , \Lambda_{F\gamma} \subset \Lambda)$$

と表現することが可能である。 一般に、位相要素 β_γ はモデリング空間に埋め込まれ

ると、既存の位相要素との干渉などによって複数に分割されることになるので、β_γ の占める空間は、境界表現形状モデルの位相要素 α_λ の集合で表現される。 図 5.30 (b) は face に関して形状特徴の位相要素と形状モデルの位相要素との対応関係を示している。形状特徴の位相要素 β_γ が占める点集合は、形状モデルの位相要素 $\{\alpha_\lambda\}$ がどのように分割されてもその部分集合として保持されることになるので、部分形状を安定して保持することが可能となる。

5.7.3 複合形状特徴の表現

複合形状特徴とは、他の形状特徴の組み合わせによって記述されるものである。形状特徴による設計では、必要なすべての形状特徴をあらかじめ用意することは困難なので、既存の形状特徴を組み合わせて新たな形状特徴を定義することが必要であり、そのための手段として複合形状特徴が用いられる。複合形状特徴を許すことによって、図 5.31 に示すように盲穴と貫通穴を組み合わせて段付き穴を容易に定義できるようになる。

このような複合形状特徴は、形状特徴間の集合演算で記述することができる。図 5.31 の例では、複合形状特徴は、$F_{compound} = (b\_hole \oplus t\_hole)$ というブール式によって、表現・保持することができる。複合形状特徴を、既存の形状特徴間の集合演算で定義できる形状と考えるとき、製品の全体形状 R も複合形状特徴と同様に表現ができる。図 5.31 の場合では、全体形状が $R = base \ominus (b\_hole \oplus t\_hole)$ または、$R = base \ominus F_{compound}$ と記述される。

複合形状特徴に相当する部分を併合形状モデルから抽出するためには、集合演算を評価することが必要である。そのためには、抽出操作を用いて、形状特徴を埋め込んだ形状モデル $M(\Lambda)$ から位相要素を適当に選び出す。抽出操作については次節で詳述する。

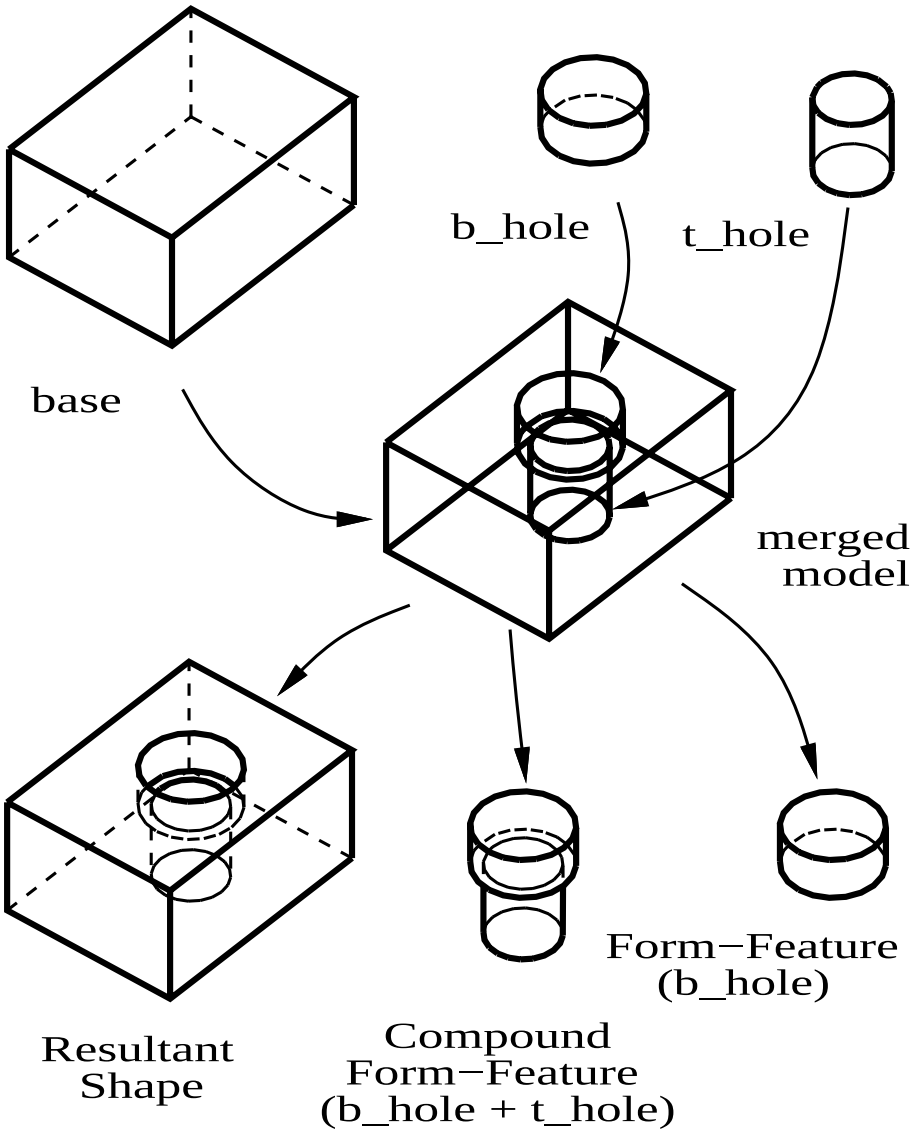


図 5.31: 併合形状から抽出される形状特徴.

5.8 抽出操作の実現法

本節では、任意の非多様体形状モデルにおける抽出操作を考えるために、抽出操作を数学的に定義し、それを実現する方法について論じる。

5.8.1 位相要素の集合間の演算

集合演算として、和集合、差集合、および前に述べたような縮退を認める積集合（$\otimes^{+}$）と認めない積集合 ($\otimes^{-}$) の４通りのものを考える。

プリミティブに併合操作を施した場合、プリミティブは、併合形状の位相要素の部分集合として表現される。従来の集合演算の定義では、Requicha の定義のように、プリミティブを３次元ユークリッド空間における点集合として扱い、集合演算を「点集合間の演算」としていた。しかし、ここでは、すべてのプリミティブは併合形状モデルの位相要素の集合として表現されているので、集合演算を「位相要素の集合間の演算」と考える。

そこで、本研究では、位相要素の集合間の演算として、集合演算を評価する方法について提案する。

5.8.2 抽出操作の数学的記述

まず、抽出操作を数学的に定義するために、次のような記法を導入する。

- 位相要素を α_λ とするとき、位相要素の集合 $\{\alpha_\lambda|\lambda \in \Lambda\}$ で構成される併合形状モデル M を $M(\Lambda)$ とかく。
- $M(\Lambda)$ においてプリミティブ A と関係付けられている位相要素の集合を、$\{\alpha_\lambda|\lambda \in \Lambda_A\}$ とする。このことを、$A(\Lambda_A) = \{\alpha_\lambda|\lambda \in \Lambda_A\}$ とかくものとする。
- $M(\Lambda)$ において、位相要素 α_λ の境界上にある位相要素の集合を $\Upsilon_\Lambda(\alpha_\lambda)$、また、$(\Upsilon_\Lambda(\alpha_\lambda) \cup \alpha_\lambda)$ を $\Phi_\Lambda(\alpha_\lambda)$ とかく。すなわち、位相要素 α が空間中に占める点集合の閉包を $[|\alpha|]$ とかくとき、$[|\alpha|] = |\Phi_\Lambda(\alpha_\lambda)|$ となる。また、位相要素の集合Aにおいて、

$$\Phi_{\Lambda_A}(A) = \bigcup_{\lambda \in \Lambda_A} \Phi_{\Lambda_A}(\alpha_\lambda)$$

$$\Upsilon_{\Lambda_A}(A) = \bigcup_{\lambda \in \Lambda_A} \Upsilon_{\Lambda_A}(\alpha_\lambda)$$

と定義する。

- α_λ が $M(\Lambda)$ に属するどの位相要素の境界上にもないとき、α_λ は境界位相要素ではない、とよぶことにする。このとき、$A(\Lambda_A)$ の位相要素のうち、境界位相要素でないものの集合を A^o または $A^o(\Lambda_{A^o})$ とかく。すなわち、

$$A^o \equiv A(\Lambda_A) - \Upsilon_{\Lambda_A}(A)$$

以上の記法を用いて、位相要素に基づいた集合演算の定義を以下のように行なう。

$$(A \oplus B) \equiv \Phi_\Lambda(A^o(\Lambda_{A^o}) \cup B^o(\Lambda_{B^o}))$$

$$(A \ominus B) \equiv \Phi_\Lambda(A^o(\Lambda_{A^o}) - B^o(\Lambda_{B^o}))$$

$$(A \otimes^- B) \equiv \Phi_\Lambda(A^o(\Lambda_{A^o}) \cap B^o(\Lambda_{B^o}))$$

$$(A \otimes^+ B) \equiv A(\Lambda_A) \cap B(\Lambda_B)$$

以上の定義によって集合演算を位相要素の集合間の演算に置き換えることができるので、任意のブール式で記述された形状モデルを併合形状から抽出することができる。

5.8.3 抽出操作の例

ここで定義された集合演算に基づいた位相要素の抽出を図 5.32 の形状を例にとって示す。この図において、(a) に示されたプリミティブを併合することによって、(b) の併合モデルを生成し、さらに、抽出操作によって (c) に示した形状モデルを求めることを考える。(d) は併合形状モデルを中心軸を通る平面で切断した図であり、$V_1, V_2, ...$ は併合形状モデルの volume、また、A, B, C, Dはプリミティブが埋め込まれた部分を示している。

形状モデルRは、集合演算

$$R = C \ominus D \oplus A \ominus B$$

によって定義されているものとする。集合演算の定義により、演算 $\ominus, \oplus$ では、A^o, B^o, C^o, D^o を考えればよい。ここでは、プリミティブを構成する vertex, edge, face はすべて volume の境界となっているので、境界位相要素でない位相要素は volume のみで

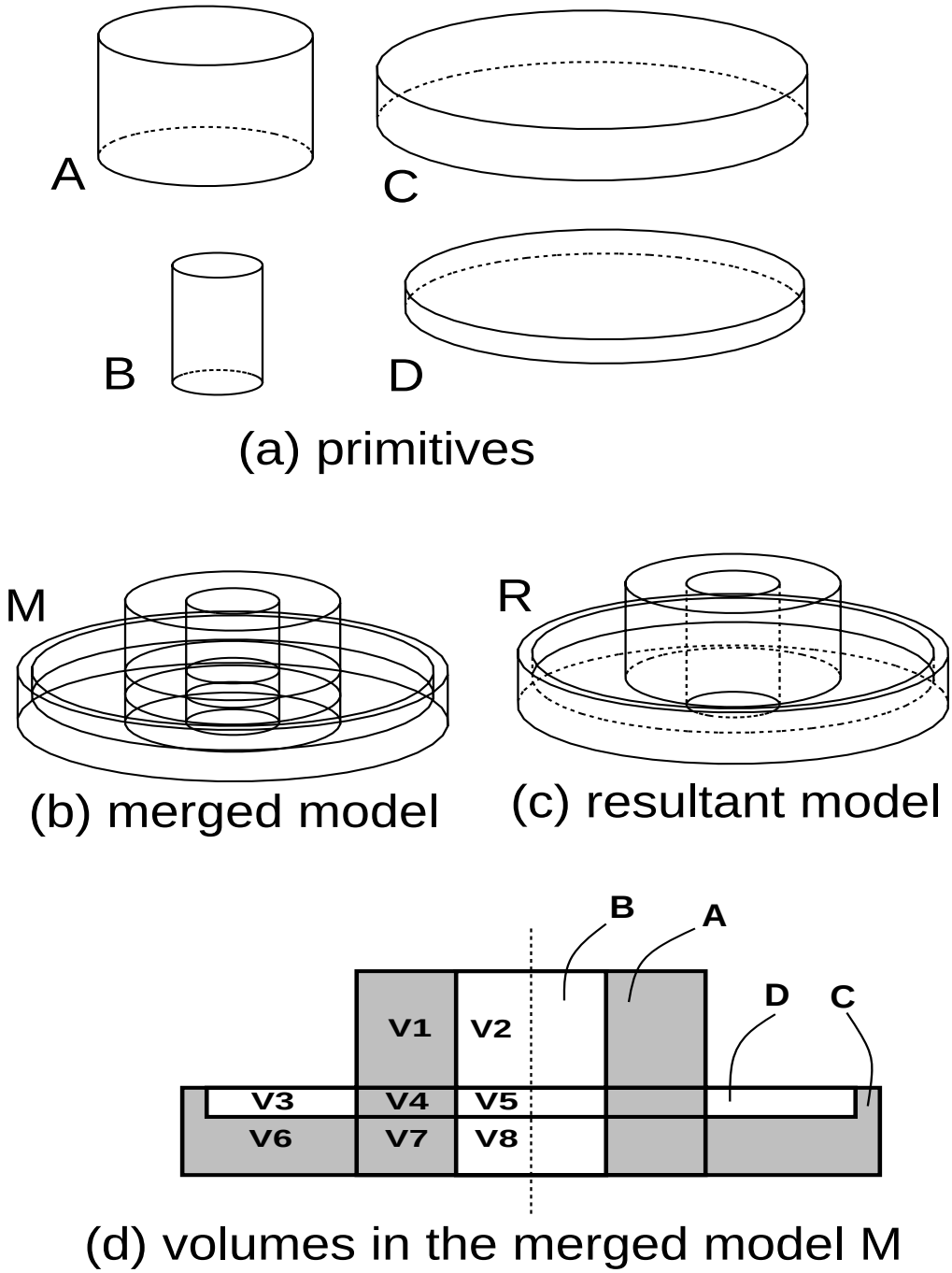

A
C
B
D
(a) primitives
M
R
(b) merged model
(c) resultant model
B
A
D
C
V1
V2
V3
V4
V5
V6
V7
V8
(d) volumes in the merged model M

図 5.32: 抽出操作の例

ある。したがって、プリミティブAの位相要素で境界位相要素でないものを A^o のようにかくとき、

$$\begin{aligned} A^o &= \{V_1, V_2, V_4, V_5, V_7, V_8\} \\ B^o &= \{V_2, V_5, V_8\} \\ C^o &= \{V_3, V_4, V_5, V_6, V_7, V_8\} \\ D^o &= \{V_3, V_4, V_5\} \end{aligned}$$

となる。形状モデルRを構成する位相要素は定義により、

$$R = \Phi(((C^o - D^o) \cup A^o) - B^o)$$

となる。ここで、この式に上で求めた集合を代入すると、

$$\begin{aligned} &((C^o - D^o) \cup A^o) - B^o \\ &\quad = (((\{V_3, V_4, V_5, V_6, V_7, V_8\} - \{V_3, V_4, V_5\}) \cup A^o) - B^o \\ &\quad = (\{V_6, V_7, V_8\} \cup \{V_1, V_2, V_4, V_5, V_7, V_8\}) - B^o \\ &\quad = \{V_1, V_2, V_4, V_5, V_6, V_7, V_8\} - \{V_2, V_5, V_8\} \\ &\quad = \{V_1, V_4, V_6, V_7\} \end{aligned}$$

となるので、$R = \Phi(\{V_1, V_4, V_6, V_7\})$ となる。したがって、volume の集合 $\{V_1, V_4, V_6, V_7\}$ とそれらの境界位相要素を加えたものが、集合演算の結果Rとなることがわかる。これは、図 5.32(d) の斜線で示した部分に相当し、(c) の形状モデルと一致する。

本定義に基づいて形状の評価を行うことにより、任意の非多様体形状モデル間の集合演算の評価形状を抽出することができる。図 5.33 は、一つの併合形状モデルから、3通りのブール式で表現された形状を抽出した例である。

図 5.33: 併合形状から抽出された形状.

5.9 併合操作の実現法

本節では、併合形状を非多様体形状モデルで実現するためのアルゴリズムについて論じる。

5.9.1 全体構成

従来のソリッドモデル間の集合演算に関しては非常に多くの研究がなされいるが、[Braid75, Tilove80, Chiyokura83a, Requicha85, Chiyokura85, Mantyla86, Eastman86, Pegal88, Mantyla88, Hoffman89]、基本的には、干渉線の生成、不要な位相要素の除去、二つのソリッドの併合、という手順で集合演算操作が実現される。ここで述べる併合操作においては、プリミティブ間の干渉計算を行ない、モデルの位相構造を変更するという点では従来の手順と同じであるが、不要な位相要素の除去は行なわず、また、ソリッド以外の形状でも併合操作が適用可能なようなアルゴリズムとなっている。

併合操作の手順

図 5.34 は、併合操作の大まかな手順を示している。左に示した図は形状モデルの位相変形の手順について、また右に示したものはプリミティブとの対応関係の記述について示している。

図 5.34 (a) は、対象となる二つのプリミティブA、Bを示している。まず、これらの間の干渉を計算することにより、図 5.34 (b) に示すような干渉点、干渉線を生成する。さらに、(c) に示したように、一致する vertex, edge, face を併合して、一つの形状モデルにする。最後に、face に囲まれる閉空間を検出し、volume の再構成を行なう。

図 5.34 の右に示したのは、併合形状の位相要素と、プリミティブの位相要素との関係を維持するための処理である。位相要素が分割されたときは、プリミティブとの関係を分割された個々の位相要素に分配し (図 5.34(b))、また二つのプリミティブの併合において、対応する位相要素が併合されたときは、プリミティブとの対応関係も併合する (図 5.34(c))。

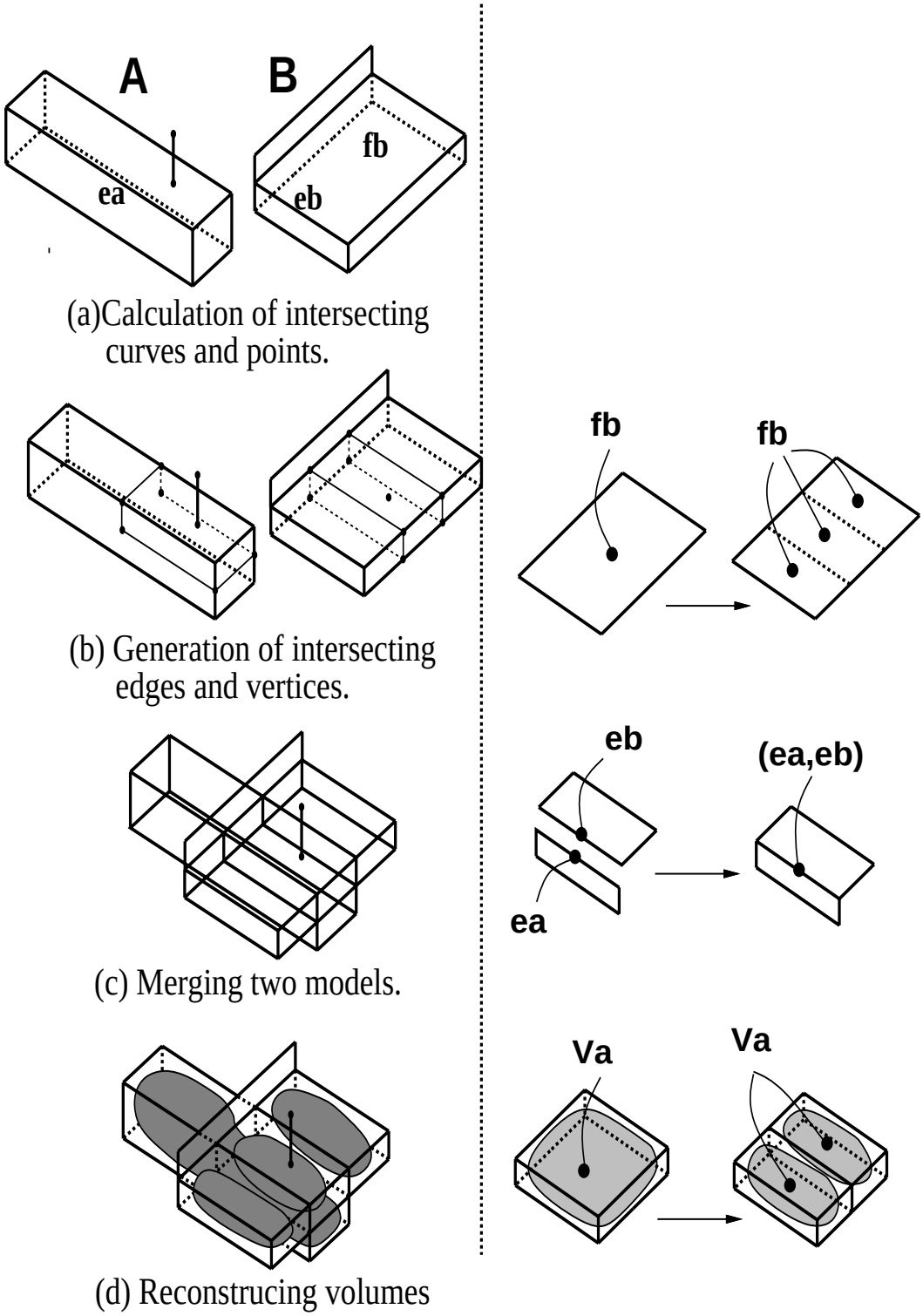


図 5.34: 併合操作の手順

5.9.2 処理の手順

具体的なアルゴリズムについては、以下に述べる通りである。

a. 干渉線/干渉点の計算

このプロセスは、干渉する可能性のある位相要素を選びだすラフチェックと、それらの干渉を計算する干渉計算の二つから成る。ラフチェックでは、非多様体位相構造を利用した階層的ラフチェックが有効である。干渉計算については、孤立 edge や孤立 vertex が存在しうるために、干渉計算の分類が多くなることを除けば、ソリッドモデリングで用いられてきた方法がそのまま適用できる。

a.1 階層的ラフチェックによる処理の高速化

位相要素間の干渉計算は比較的時間がかかる処理なので、ラフチェックによって干渉する可能性のある位相要素を絞り込むのが一般的である。ラフチェックの方法としては、位相要素を含んだ直方体や球を求め、それらの間で干渉が生じる場合にのみ位相要素間の干渉計算を行なうという方法が取られる。位相要素を囲む直方体や球は比較的軽い計算で求まるので、ラフチェックによって計算時間の短縮をはかることができる。

しかしながら、干渉計算の効率を向上させるためには、これだけでは必ずしも十分ではない。なぜならば、ラフチェックの回数が膨大になると、計算時間の大半をラフチェックに要する時間が占めてしまうことも少なくないからである。特に、face の個数が多く、かつ、干渉する位相要素が少ない形状モデルにおいては、ラフチェックに要する時間が最も多くなる。

図 5.35 は、 face の個数が多くなるときに、集合演算における計算時間においてどのような処理が最も重いかを調べる実験を行なったものである。この図において、プリミティブＡは円筒部を１６角形で近似しており、計５７枚の face から成っている。形状モデルＢは、プリミティブＡを多数個足し合わせて生成したものである。ここで、プリミティブＡと形状モデルＢを足し合わせるのに要する時間を測定する。このとき、形状モデルＢの面の数に応じて、集合演算一回あたりの計算時間がどのようになるかをグラフで示す。図 5.36 のグラフで横軸は形状モデルＢの面の個数、縦軸はＣＰＵタイムである。さらに、このグラフに、プリミティブＡと形状モデルＢとのラフチェックのみの計算時間も示した。ラフチェックとしては、face を囲む直方体同士で交わりがあるかを調べた。これによって、集合演算の計算時間のうち、ラフチェックに要する時間

図 5.35: プリミティブＡと、Ａを１００個併合した形状モデルＢ

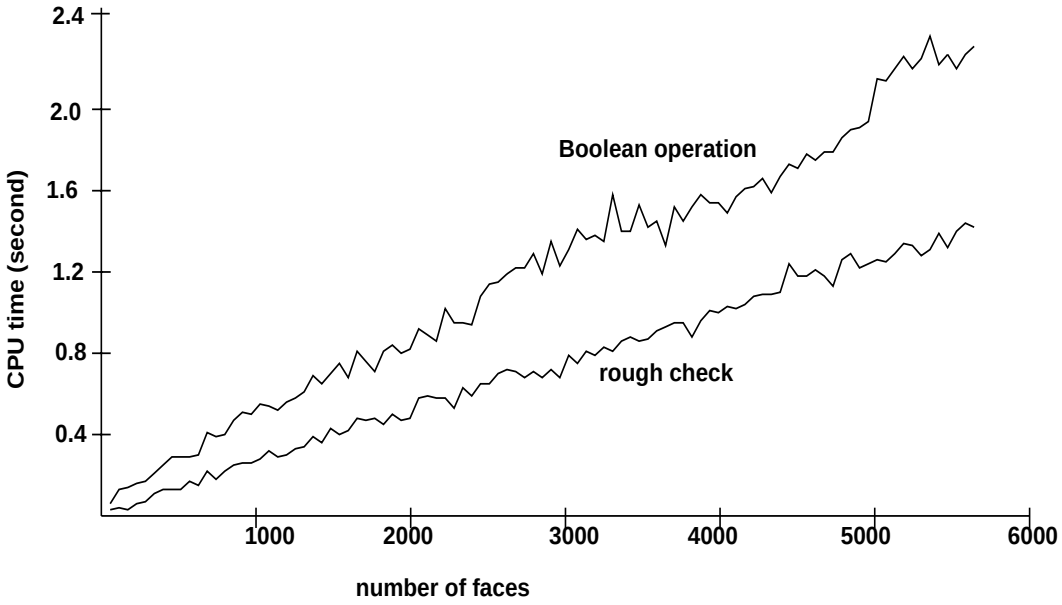


図 5.36: 集合演算とラフチェックの計算時間

がどの程度なのかがわかる。このグラフから、この例題においては、ラフチェックが集合演算の全体の計算時間の相当部分を占めていることがわかる。この例題では、１００個のプリミティブＡを足し合わせた形状モデルＢに１０１個目を足す際、$(５７ \times １００) \times ５７ = ３２４,９００$ 回のラフチェックが必要であり、この負荷が非常に大きなものとなっている。したがって、大規模な形状モデルの集合演算において計算時間を短縮しようとする場合、ラフチェックの回数をいかに減らすかが重要である。

この問題を解決するために、階層的ラフチェックを行なうことによって、ラフチェックの回数を低減させる手法を示す。手順を以下に示す。

1. まず、集合演算の対象となる形状モデル A, Bから、volume, 孤立 face (volume の境界でない face), 孤立 edge (volume, face のいずれの境界でもない edge), 孤立 vertex (volume, face のいずれの境界でもない edge) を取りだし、それらをそれぞれ $\{a_i\}, \{b_j\}$ とする。

2. すべての (a_i, b_j) の組み合わせについて干渉の有無をラフチェックを行なって調べる。

3. ラフチェックではじけない場合、もし、a_i が volume である場合には、volume の境界上の位相要素のうち、 face, face の境界でない edge, face や edge の境界で

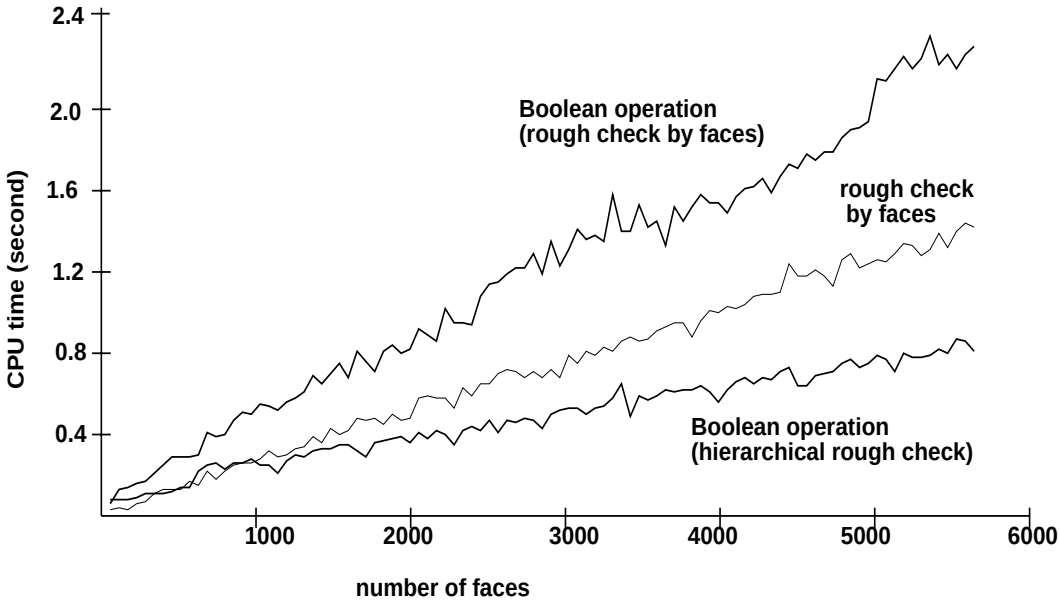


図 5.37: 階層的ラフチェックを用いた集合演算の計算時間

ない vertex を求め、これらと b_j との間でさらにラフチェックを行なう。b_j が volume である場合も同様。

4. 最終的には、ラフチェックではじけない位相要素の組み合わせとして、(face,face), (face, 孤立 edge), (face, 孤立 vertex), (孤立 edge, 孤立 edge), (孤立 edge, 孤立 vertex), (孤立 vertex, 孤立 vertex) が残る

以上のラフチェックを行なうことによって、干渉する部分が全体にくらべて少ない例題において、ラフチェックに要する時間を大きく低減させることができる。図 5.35 で、１００個のプリミティブＡを足し合わせた形状モデルＢに１０１個目を足す例で考えると、１００回の (volume,volume) のラフチェックで２個の volume に絞ることができ、さらに volume を face の集合に分解し、６４９８回の (face,face) のラフチェックを行なえば干渉し得る face の組み合わせを１６組に絞り込むことができる。この結果、(face,face) のみでラフチェックを行なった場合に比べて、ラフチェックの回数を４９分の１に減らすことができる。

図 5.37 のグラフは、階層的ラフチェックを用いた場合の集合演算の計算時間を (face,face) のラフチェックのみを用いた場合と比較したものである。また、細線で示したのは、階層的ラフチェックを用いなかった場合に、ラフチェックのみに要する時間である。このグラフより、本例題においては、階層的ラフチェックを用いた場合、従来のラフチェッ

クによる集合演算に比べて、約３５%の計算時間で処理が済んでいることがわかる。

このラフチェックは、volume が近接する face をまとめる役割を果たしていることを利用したものであり、非多様体形状モデルの特徴を生かしたものである。この手法は全体に比べて干渉する部分が少ないときに有効なものとなる。機械製品のモデリングにおいてはプリミティブが全体にまたがるようなことはそれほど多くないので、この方法は多くの場合有効であると思われる。

a.2 干渉計算

ラフチェックで残った位相要素の組に対して、干渉点、干渉線を求める。これに関しては、ソリッドモデルにおいて用いられてきた方法 [Kobayashi87] と同等の手法を用いる。ただし、非多様体形状モデルでは、ワイヤエッジや孤立点も存在するため、これらに対する処理を付加する必要がある。以下に処理の手順を示す。なお、ここでは face の方程式を area, edge の方程式を wire と呼んでいる。計算された交線、交点がプリミティブのどの位相要素上に乗っているか、または一致しているかは後の計算のために保持しておく。

1. (face1,face2) に関して、それぞれの面の方程式として、area1, area2 を得る。ここで、area1 と area2 が一致していないとき
   (a) area1 と area2 との交線 wire12 を求める。
   (b) wire12 と face1 の境界位相要素との交線、交点を求める。交線は face1 の境界 edge が wire12 上に乗っているときに生じる。wire12 と face2 との間でも同様の計算を行なう。
   (c) 交点、交線を wire12 上でソートする。
   (d) 隣合う交点を結ぶ線分が face1, face2 の両方に乗っていれば、その線分は face1, face2 の交線となる。face に乗っているかを判断する方法としては、三重積を取る方法や線分の中点が face 上に乗っているかを調べる方法が知られている [Kobayashi87] 。
2. area1, area2 の方程式が一致するときは、face1 のそれぞれの境界位相要素と face2 の交線、交点を求める。face1 の境界位相要素が edge のときは、その edge の wire を考えることで、１ で述べた方法に帰着させる。孤立 vertex のときは、その vertex が face2 に乗っているかを調べ、もし乗っていれば vertex の座標が交点となる。face2 の境界位相要素についても同様に face1 との交わりを調べる。area の一致する face については、(face1, face2) のペアとして保持しておく。

3. (孤立 edge, face) との間で干渉計算では、edge の wire を考えることで、1 に帰着する。

4. (face, 孤立 vertex) においては、vertex が face 上に乗っていれば交点となる。(孤立 edge, 孤立 vertex), (孤立 vertex1, 孤立 vertex2) でも同様。

5. (孤立 edge1, 孤立 edge2) では、それぞれの方程式 wire1, wire2 の交点が edge1, edge2 の両方に乗っていれば、その交点は (edge1,edge2) の交点である。

b. 干渉位相要素の生成

この操作では、計算された干渉点と干渉線を形状モデルA、Bのそれぞれに干渉 vertex, 干渉 edge として生成する。

図 5.34 (b) は、それぞれに干渉位相要素を生成したものである。この操作は、本論文で述べたオイラー操作において、split_edge (edge の分割)、 split_face (face の分割)、make_edge_vertex (face 中に strut edge を伸ばす)、make_vertex_ring (face 中に孤立点を生成する)、make_edge_kill_ring (face の 異なる loop 間に edge を生成する)、make_edge_kill_Vcavity (volume の異なる shell 間に edge を生成する) を用いることによって実現できる。

ここで、edge や face が分割されるときは、プリミティブと位相要素との対応関係が変更される。プリミティブBの f_b について考えると、干渉 edge を生成することにより、図 5.34 (b) の右の図に示したように3つの face に分割される。このときは、元々のプリミティブBの面 f_b が分割された3つの face に対応するように、対応関係を付け変える。同様の処理はすべての分割された位相要素に対して行なわれる。

また、干渉点、干渉線を計算した際に、プリミティブA, Bのどの vertex と edge が一致しているかは既に求まっている。そこで、それらについて一致するペアを保持しておく。これは、後にこれらを併合するときに利用するためである。また、干渉位相要素を生成するときには、図 5.18 で説明したように、プリミティブA, Bの干渉位相要素リストとして保持する。

c. 一致する位相要素の併合

次に形状モデルA, Bを併合して一つの形状モデルにする。そのために、形状モデルA, Bで、対応する vertex, edge, face を併合する。一致する位相要素のペアの併合処理は、face, vertex, edge の順で行なう。

併合処理の手順は以下の通りである。

- 一致する (vertex1,vertex2) のペアの併合を行なう。
- edge は co-edge の集まりとなっているので、edge の併合は、co-edge の集合を足し合わせればよい。ただし、co-edge の併合のときは、edge 廻りの face の順序関係であるラジアルサイクルを管理する必要があるため、co-edge の併合後に edge の廻りの face の順序を求め直し、ラジアルポインタを付け変えることが必要である。
- 面の方程式の一致する (face1,face2) のペアにおいて、
  1. face1, face2 の境界位相要素を順次比較していき、face1, face2 が一致するかどうかを調べる。
  2. 一致するときは、併合形状とプリミティブとの関係を更新する。たとえば、併合形状モデルの face1, face2 がそれぞれ、プリミティブの面 f_a, f_b と、face1 $\leftrightarrow f_a$, face2 $\leftrightarrow f_b$ のように関係が付けられていた場合、face2 の関係を face1 を併合し、face1 $\leftrightarrow \{f_a, f_b\}$ という関係に更新する。
  3. face1 と face2 の一方を消去する。

d. volume の再構成

二つの形状モデルを併合することによって、元々あった volume は分割される。図 5.34 (d) では、プリミティブの併合によって、face に囲まれる閉空間が４個生成される。そこで、volume を求め直し、データ構造を変更する必要がある。また、孤立 vertex, 孤立 edge が volume の内部にあるかを調べる必要がある。

処理の手順は以下の通りである。

1. 併合形状モデルにおいて、face に囲まれた閉空間を、ラジアルポインタを辿ることによって求める。
2. 得られた閉空間に位相要素 volume を定義する。この際、生成される volume は、プリミティブの volume を分割したものとなっているので、図 5.34 (d) の右の図で示したように、プリミティブの volume との関係を付け直す。
3. volume 間の包含関係を調べ、volume の空洞が生成されるかどうかを調べる。

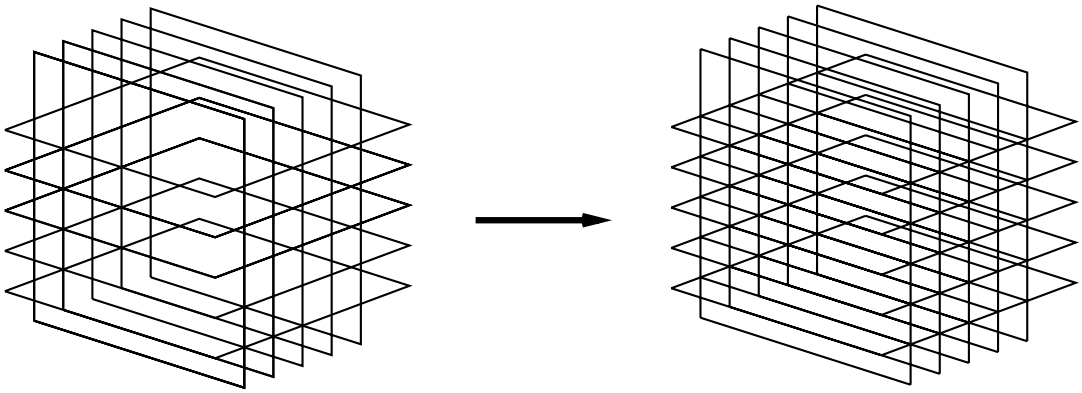
4. 併合された形状モデルにおいて、孤立 edge, 孤立 vertex は volume の内部に含まれいる可能性がある。そこで、これらが生成された volume の内部にあるかどうかを調べ、もし内部ならば volume の境界となるようにデータ構造を変更する。

例題

ここで述べているプリミティブは一般の非多様体形状モデルでよいので、以上の a-d の処理を施すことによって、任意の非多様体形状モデルの併合形状を生成することができる。ここで述べた処理は実装されており、その処理のいくつかの例を図 5.38 に示す。この例題は、サーフェス、ソリッドにおける併合操作である。

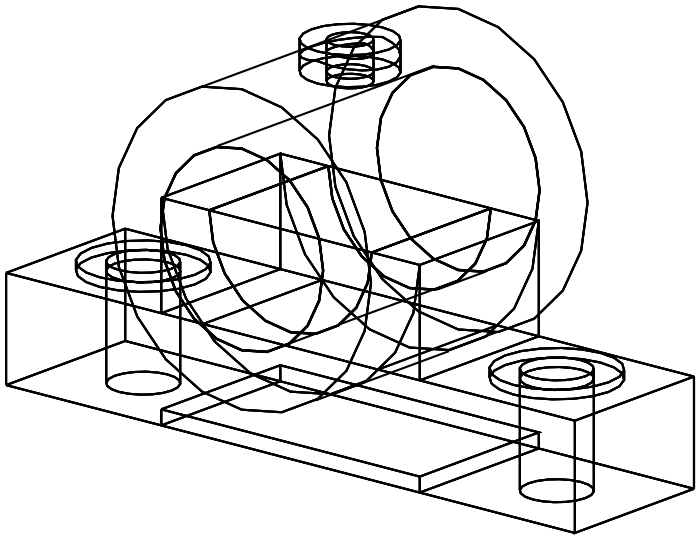
(a) は、１０個の平面モデルを併合して生成された形状モデルである。

(b) は １２個のプリミティブを併合して生成した形状モデルである。プリミティブ間の干渉によって volume が分割されており、２２個の volume を含んでいる。



10 plane models　　a merged model

(a) merging surface models.



a merged model of 12 solid primitives

(b) merging solid models.

図 5.38: 非多様体形状モデルの併合.

5.10 簡略化操作の実現法

これまで、併合形状モデルを保持する方法について論じてきた。しかしながら、このような位相構造が常に必要となるとは限らない。たとえば、形状特徴の表現や試行錯誤的な形状生成の必要がないときは、この表現はデータ量の点からみて無駄である。

また、集合演算によっては、併合操作が非効率になることがある。図 5.39 は、そのような典型的な例を示している。この例題では、ｎ個の立方体を ９０/ｎ 度ずつ回転させて足し合わせた形状を示している。このような場合、互いに干渉する部分が指数的に増えていき、評価形状に現れない位相要素が非常に多くなる。このような場合に置いては、部分的に評価形状に現れない位相要素を除去して処理を軽くすることが望ましい。

そこで、本節では、評価形状に現れない補助的な位相要素を除去する方法について論じる。この操作を簡略化操作と呼ぶ。

5.10.1 不要な位相要素の除去

併合操作では評価された形状には現れない位相要素も保持している。しかし、併合操作による付加的な位相要素は消去することもでき、形状特徴の表現や試行錯誤的な形状生成の必要がないときは除去してもよい。ここでは、評価形状の境界位相要素のみを残し、それ以外を消去する操作について述べる。

簡略化操作では、評価形状の境界位相要素以外の位相要素を消去し、また、併合操作の過程で分割された face や edge を併合することによって実現される。この操作の手順は以下の通りである。

1. 抽出操作で選ばれなかった位相要素を、volume, face, edge, vertex の順で、オイラー操作によって消去していく。

2. 併合操作の過程で分割された volume を併合する。そのために、face の両側が volume に隣接する face を除去する。

3. volume 中に孤立 edge, 孤立 vertex があれば消去する。

4. 併合操作の過程で分割された face を併合する。そのために、edge が同一の方程式を持つ二つの face の境界のとき、それらの face が併合できるならば、その edge

図 5.39: 併合操作が不利な例題： 2^n個の立方体の和集合.

を消去することにより併合する。また、face 中の strut edge や孤立 vertex が他の face や edge の境界でないならばそれらを消去する。

5. 併合操作の過程で分割された edge を併合する。そのために、vertex が二つの edge に共有され、それらの edge が同一の方程式を持つならば、vertex を消去する。

6. 抽出操作と部分形状が関連付けられている場合は、関連付けのためのデータを破棄する。

5.10.2 例題

図 5.40 に、併合形状モデルに簡略化操作を施した例を示す。この例では、２７個の volume から構成されていた併合形状モデルが 2-多様体形状モデルに簡略化される。それにより、プリミティブの情報が破棄され、形状モデルのデータ量は約半分になる。また、CPU Time は、IBM RS6000/730 を用いて 0.07 秒であり、簡略化操作が非常に軽い操作であることがわかる。

次に、図 5.39 の場合について考える。この例題では、併合操作＋抽出操作で集合演算を行った場合と、併合操作＋抽出操作＋簡略化操作で集合演算を行った場合とを比較してみる。その結果は、表 5.1 に示す通りである。この図では、足し合わせる立方体の個数に応じて、面の個数と処理時間がどのように変わるかを示している。この結果からわかるように、干渉線が指数的に増加するような例題においては、併合操作に要する処理時間が著しく増大する。そのような場合には簡略化操作を施すことが必要となる。

	簡略化操作なし		簡略化操作あり	
立方体の個数	面の個数	計算時間	面の個数	計算時間
2	42	0.08 s	18	0.08 s
4	210	0.25 s	34	0.12 s
8	930	1.66 s	66	0.30 s
16	3906	14.55 s	130	0.77 s

表 5.1: 集合演算の計算時間 (IBM RS/6000-980).

併合操作／抽出操作の直後に簡略化操作を行なうと、付加的な位相要素を保持しない従来の集合演算と同等な位相構造を得ることができる。このように、必ずしも部分形

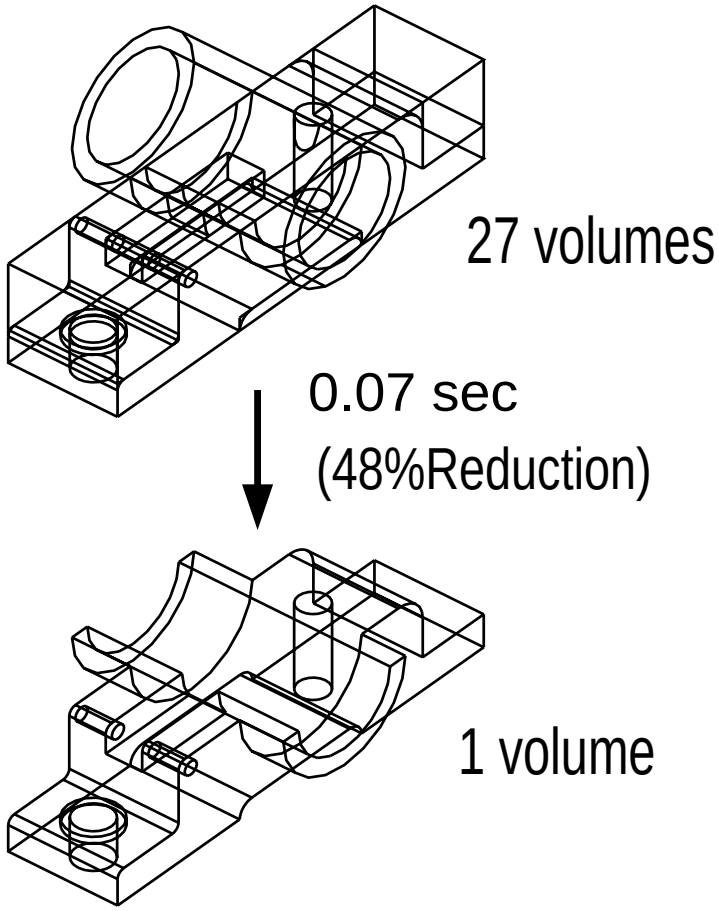


図 5.40: 簡略化操作の例.

状を保持する必要がないときは、併合操作、抽出操作、簡略化操作を組み合わせた集合演算が有利である。

最後に、実際の製品の形状モデルにおいて、併合操作、抽出操作、簡略化操作を施した場合の処理時間を図 5.41 に示す。ここでは、形状モデルを併合操作によって生成し、形状が確定した後で簡略化操作を施してデータ量を軽減させている。

図 5.41: 併合操作、抽出操作、簡略化操作の処理時間 (IBM RS/6000-980).

5.11 まとめ

本章では、非多様体形状モデルのための集合演算として、併合操作と抽出操作に基づいた集合演算を提案し、その実装法を示した。そして、従来の３次元形状モデルで解決が困難であった、試行錯誤的な形状変形操作と、形状特徴表現のための部分形状の保持の二点について論じ、非多様体位相構造を用いた解決法を示した。

要点は以下の通りである。

- 集合演算を演算順序に依存しない併合操作と、演算順序に依存する抽出操作の二つを組み合わせて実現する手段を示した。また、簡略化操作を組み合わせることで必要最小限の位相要素からなる位相構造が得られることを示した。
- 集合演算の適用された任意のプリミティブを、集合演算順序によらず、高速に修正する方法を示した。任意のプリミティブに関して、取消操作では局所操作のみ、修正操作では唯一度の集合演算の実行のみで形状モデルの修正ができるアルゴリズムを明らかにし、大規模な形状モデルでも短時間で形状の修正が行えることを示した。
- 集合演算の柔軟性を増すために、プリミティブに優先順位を設定し、優先順位を反映した評価形状を得る方法を示した。
- 非多様体位相構造を用いた形状特徴表現法について提案した。本表現により、線特徴、面特徴、体積特徴など様々な形状特徴が表現でき、また最終的に形状モデルの境界位相要素として残らない場合や、形状特徴間に干渉が生じた場合でも、整合性を失わず、形状特徴として保持できることを示した。
- 非多様体形状モデルにおける併合操作の実装法を示した。
- 抽出操作のための演算規則を定義し、併合形状モデルから、任意のプリミティブの和、差、積の組み合わせでできる形状を、位相要素の集合として取り出す方法を定式化した。
- 併合形状から付加的な位相要素の除去ができる簡略化操作の実装法を示した。

Chapter 6

非多様体形状モデルを利用した三面図からのソリッド モデル合成法

本章では、三面図からソリッド モデルを合成する問題において、非多様体形状モデルが有効に利用できることを示す。

三面図からソリッド モデルを作成する問題においては、処理の過程でワイヤフレーム、サーフェス、ソリッドが現れるため、これらを別々のデータ構造で表現した場合、三面図との対応の管理などの処理が複雑になるという問題点があった。また、一般に三面図は立体形状の表現としては曖昧であるが、多数のソリッド解の生成には計算時間を要するという問題がある。さらに、現実の三面図においては誤りが含まれることが多く、そのような図面に対しては従来の手法では対応できていなかった。

この問題に非多様体形状モデルを用いた場合、処理の過程で現れるすべての幾何形状が統一的なデータ構造で表現可能となるため、ソリッド合成に必要な３次元形状処理が単純化できる。さらに、中間形状に非多様体データ構造で表現されたセル分割モデルを用いることにより、すべての解候補を単一の形状モデルによって保持することが可能になる。そのため、従来の手法に比べて非常に処理効率のよいソリッド合成システムが実現できる。また、本手法は、従来の手法では対応できなかった三面図に誤りが含まれる場合のソリッド合成にも拡張することができる。

6.1 三面図からのソリッド合成

本節では、三面図からソリッドモデルを生成することの意義について述べる。

三面図からのソリッド合成の利点

機械製品の設計や生産においては、３次元形状モデルが有効であることは広く認識されている。特に、計算機を用いたシミュレーションや解析においては多くの場合、３次元形状モデルが不可欠である。

しかしながら、現状では、実際の設計において３次元形状モデルが利用されている場は限られている。特に日本の設計の現場においては２次元図面を利用している場合が多い。このような現状の理由としては、３次元形状モデラに意図した通りの形状を入力するのが大変であること、２次元ＣＡＤの方が安価であること、既存の２次元図面による製品データの蓄積が膨大であること、また、設計現場での３次元形状モデルの利用に対する抵抗感などを挙げることができる。

このような観点から考えると、２次元図面から３次元形状モデルを作成できれば非常に有用であると思われる。それにより、既存の図面データをソリッド化して解析等を行なったり、ソリッドモデル生成のための補助的手段として利用することができる。また、２次元ＣＡＤから３次元ＣＡＤへ移行するための過渡的な手段としても期待できる。

本章では、２次元図面のうち、特に三面図について考える。三面図は、互いに垂直な三方向からの投影図によって立体を記述したもので、一般的に広く用いられている。三面図からソリッドモデルを生成する研究はこれまでもなされてきたが、当初は図面データの精度がソリッドモデルと比較して著しく悪いという本質的な問題があった。しかし最近では、２次元ＣＡＤを用いて精度のよい三面図が作成されるようになってきており、特に、三面図作成をサポートする２次元ＣＡＤを用いた場合には、線分の端点が微妙に離れていたり、線分の対応がずれているという問題はほとんど生じなくなっている。そのため、図面データを元にソリッドモデルを生成することが可能な状況になってきている。

6.2 従来のソリッド合成の研究

本節では、これまでなされてきた三面図からのソリッド合成問題の研究について概観する。

２次元図面からソリッドモデルを生成する問題に対しては様々な研究がなされており、その代表的なものとして、この問題を面の組み合わせ問題として捉えた 出沢の方法と基本立体の組み合わせ問題として捉えた Wesley の方法の二つのアプローチがある。

出沢 の方法

出沢 は、三面図から３次元の頂点、稜線、面を順次生成し、立体としての条件を満たすような面の組み合わせを求める方法を提案した [Idesawa72, Idesawa73]。この方法の特徴は、頂点、稜線、面生成の各段階で生成される立体になり得ない位相要素（虚図形）を、適当なルール群を設定することによって除去していることである。図 6.1 (a) は、虚図形を含んだ形状で、(b) は、虚図形を排除した形状である。この方法は、ソリッドモデルを面の組み合わせ問題に帰着させて求めているので、組み合わせが多くなり探索領域が大きくなる。そのため、推論に負担がかかり、また正しい立体になるための条件と立体が三面図に一致するための条件をすべてルールとして記述して解かなければならないので処理が重くなるという問題がある。

出沢 の方法を拡張したものとして、高橋や西田らの研究がある。これらの研究では、虚図形を除去し、ソリッドを構成する面を探索するためのルール群について提案しており、複数のソリッド解を常識的な判断で絞り込むためのルール [Takahashi91] や、常識的な知識を用いて探索領域を絞り込むためのルール [Nishida91] を示している。

Wesley の方法

Wesley らはソリッド生成問題を、面の探索ではなく、基本立体の組み合わせ問題に帰着させる方法を提案した [Wesley81]。図 6.2 にこの方法を示す。この方法では、三面図から作成されたワイヤフレームモデル (図 6.2(a)) に面を生成したのち、面によって囲まれる最小の閉領域を基本立体として表現した（図 6.2(b)）。そして基本立体を適当に組み合わせた和集合として求まる立体形状のうち、投影図が三面図に該当するものを解ソリッドとした。図 6.2(c) は解ソリッドの一部である。

この方法では、基本立体を生成した時点で正しい立体となるための条件は満たしてい

図 6.1: 出沢 のソリッド合成法 [Idesawa72]. 虚図形を含んだ形状 (a) と虚図形を除去した形状 (b)

るので、出沢の方法のように解探索に立体の条件を記述した多数のルールを適用する必要がなく、探索が単純になるという利点がある。しかしこの場合でも、n 個の基本立体の組合せで生成可能な解候補の個数は 2^n 個となるため、基本立体の組に集合演算を施して候補ソリッドを生成するための処理に非常に時間がかかる。この問題は本質的に NP 完全問題なので、三面図に該当するソリッドを求めるには効率的な解探索が不可欠であるが、[Wesley81] ではそのような解の探索手法については論じていない。また、多数のソリッドモデルを解として保持する際にはデータ量が多くなるという問題もある。

なお、Wesley は多面体の場合のみについて示したが、Sakurai は三面図には現れない２次曲面のシルエット稜線 (silhouette edge) や接稜線 (tangency edge) を扱う方法を示し、対象を２次曲面にまで拡張した [Sakurai83] 。また、Gu ら は、２次曲面間の干渉で生じる空間曲線を含む立体の三面図に対応する方法を示した [Gu85] 。

その他の手法

また、原田らは二面図に基づき、あらかじめ登録された頂点のパターンを利用して、立体の稜線を算出する方法を示した [Harada87] 。しかし、多面体のみを扱っているおり、さらに登録されていないパターンには対応できないため、生成できるソリッドモデルが限定されているという問題がある。Aldefeld は、図面に現れる線分などのパターンを解釈するためのヒューリスティックな手法を提案した [Aldefeld83] 。しかし、この方法においてもすべての解釈法をあらかじめ用意できないために、適用範囲が比較的単純な図面に限定されるという問題点がある。

また、三面図との対応は扱っていないが、ワイヤフレームからソリッドモデルを生成する研究も多くなされている [Kela89, Agarwal92, Courter]。

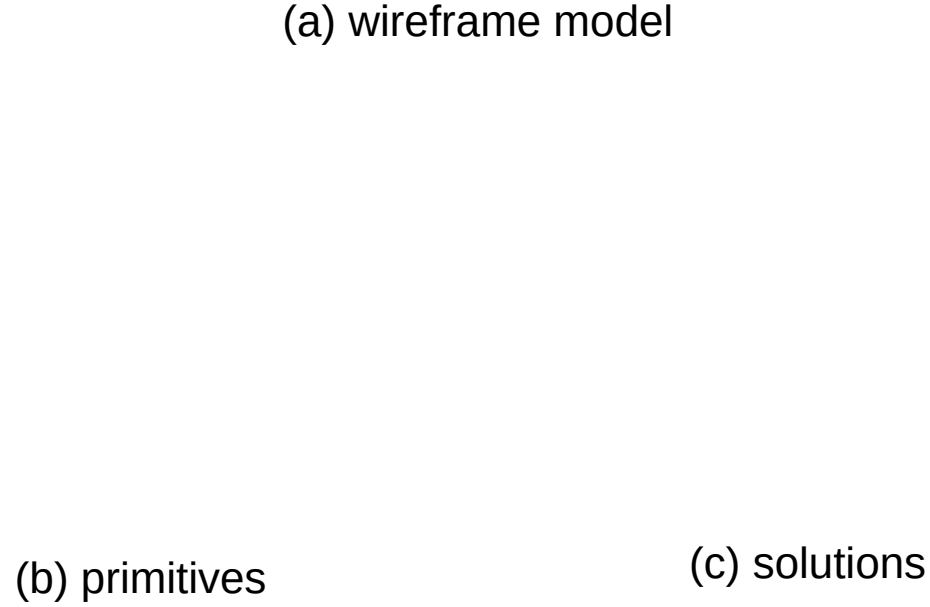


図 6.2: Wesley のソリッド合成法 [Wesley81]

6.3 従来の手法の問題点

これまで提案されてきた三面図からソリッドモデルを作成する手法では、処理効率に問題があったり、対象とする図面に制約があったり、また誤りや省略のない完全な三面図しか扱えないといった問題があり、実用的には十分なものではなかった。

既存の方法の問題点は以下のようにまとめることができる。

1. 三面図からソリッドモデルを作成する問題においては、三面図からワイヤフレームモデルを作成した後に面を生成し、三面図を満たすソリッドに相当する面の集合を算出する必要がある。そのため、処理の過程で、ワイヤフレーム、サーフェス、ソリッド表現が必要となるが、それらを独立に表現した場合、個々の形状モデル間の位相的な関係や、三面図との対応が記述されていなければならないので、データの管理が非常に複雑になる。

2. 三面図はソリッドモデルに比べて曖昧であり、三面図が複数のソリッドに対応することがある。三面図を幾何的に解釈していった場合、しばしば非常に多くのソリッドモデルが生成されるが、その場合、すべてのソリッド解の生成に時間がかかり、保持すべきデータが大きくなる。

3. ソリッド解の探索問題は本質的に NP 完全である。そのため、問題の性質を十分に生かした推論を行なわないとソリッド解の探索に非常に多くの計算時間を要する。この問題を立体を構成する面の探索問題として捉えた [Idesawa72] のような方法の場合、矛盾のないソリッドとなるための幾何的拘束条件と、ソリッドの投影図が与えられた三面図に一致する条件の両方を満たす面の組み合わせを求めるために、多くのルールを用いた複雑な探索が必要で、処理が重くなる。また、[Wesley81] のように基本立体の探索問題として捉えた研究もあるが、有効な解探索方法については示していない。

4. 従来の手法では、誤りがなく、かくれ線がもれなく描かれた完全な三面図が必要で、かくれ線の一部が省略されたり図面に誤りのある場合はソリッドモデルの合成を行なうことができなかった。しかし、現実の三面図はほとんどの場合かくれ線の省略や多少の誤りを含んでおり、完全な三面図はあまり多くないのが実情である。そのため、既存の方法では現実の三面図にはあまり対応できない。

そこで本章では、このような問題に対処するために、非多様体形状モデルを用いた新しいソリッド合成法を考える。

6.4 本研究の方針

本研究では、非多様体形状モデルと ATMS (Assumption-based Truth Maintenance System) に基づく推論方法を組み合わせることで前節の問題を解決することを考える。

以下に前節の各問題に対応させて、問題解決のための基本方針について説明する。

1. 三面図からのソリッド合成問題においては、中間形状としてワイヤフレーム、サーフェスモデルが現れ、本手法ではさらに 2 で述べるようにセル分割モデルが現れる。このように様々な表現が現れる問題においては、非多様体形状モデルが非常に有効であると考えられる。

   本手法では、非多様体形状モデルを用いて統一的な位相表現をしているので、個々の表現間の対応関係を保持する手間は必要なく、従来の方法に比べて処理を単純化することができる。

2. Wesley の方法では、ソリッド解の算出を基本立体の組み合わせ問題に帰着させたが、この方法ではソリッド生成のための集合演算の負荷が重く、また多数のソリッド解の保持に多大なデータ量を必要とした。

   本手法では、基本立体ではなく、複数の volume を持ったセル分割モデルを用いることにより、探索空間のすべてのソリッド候補を単一の形状モデルで表現する。前章で説明した抽出操作を用いれば、セル分割モデルの位相要素の部分集合としてソリッドモデルが簡単に作成できるので、Wesley の方法のように基本立体の組み合わせを評価する手間は必要ない。そのため形状処理が効率良く行なえ、複数のソリッドモデルの解を別々に保持する必要もなくなる。

3. 本手法では、ソリッド解探索をセルの組み合わせ問題に帰着させることができる。セルの組み合わせは、セル構造モデルと三面図との間に成立する拘束式を解くことにより算出する。

   拘束式を解く手段に ATMS を利用することにより、複数解を比較的短時間で求めることができる。ATMS は、仮説を与え、ある命題を成立させるための仮説の集合を管理するシステムであるが、セルの存在を仮説、 三面図の線分とセル分割モデルの稜線をノードとすることによって ATMS が適用でき、 三面図に該当するセルの組合せを探索することができる。一般に ATMS の応用では比較的探索時間がかかる傾向があるが、本問題では、探索時間がかかる大きな原因である nogood 環境の個数がそれほど多くないという特徴があるので探索領域を限定でき、ATMS にとって有利となっている。

4. 三面図に誤りがあり、本来あってはならない線分が含まれているときは、ソリッド解を算出することができない。このような場合、推論システムに与える条件を緩め、三面図の一部を満たすような解を探索することで、ソリッド候補を算出することができる。

   また、足りない線分や省略された隠れ線がある場合は、図面と３次元モデルとの間の不整合が検出できるので、その不整合を解消できるような線分を「あってもなくてもよい」という属性を付けて仮想的に三面図に追加した上でソリッド解探索を行なう。ただし、どのような線分を追加するかには任意性があり、一意に決めることはできない。そこで、本手法においては、足りない線分は主軸に平行であるという制約のもとで誤り修正を行う。

本手法の制限

本手法は、三面図が直線と円弧で構成され、かつ、ソリッドモデルが平面、円筒面、円錐面、球面、トーラス面の５種類の曲面で構成されるという条件のもとで考える。また、各曲面の軸は、主軸に平行であるとする。さらに、すべての稜線は平面曲線とし、曲面が干渉してできる空間曲線は存在しないものとする。（なお、楕円を含めた場合の処理方法は既に知られており [Gu85]、本手法を楕円を含む場合に拡張することは比較的容易である）

また、一般にソリッドモデルは二面図だけからでも合成できる [Wesley81]。ただし、二面図の場合には、省略された投影図には円筒と円錐を投影して現れる円弧は存在しないものとする。

6.5 全体構成

処理の流れ

本論文で提案する三面図からのソリッド合成法について、処理の大まかな流れを図 6.3 に示す。本手法が従来の手法と大きく異なるのは、形状処理が非多様体形状モデルを用いて統一的になされていることと、中間状態としてセル分割モデル (cell decomposition model) を用いていることである。

図 6.3 において、まず、三面図から対応する 2 次元線分の組を探しだし、3 次元のワイヤフレームモデルを作成する。さらに、ワイヤフレームモデルから面のループを探索し、face を生成することによって、サーフェスモデルに変換する。この処理は、非多様体形状モデルを利用してなされる。

次に、非多様体形状モデルを用いたセル分割モデルを中間表現として生成する。セル分割モデルは、volume の集合で構成されているので、volume の組み合わせに応じて様々なソリッドモデルを抽出できる。ここでは、三面図とセル分割モデルの対応関係を拘束式 (constraints) として求めて、推論システム (ATMS) によって適切なセルの組み合わせを算出する。セルの組み合わせが算出されると、セルの集合に対応する幾何形状の境界位相要素をセル分割モデルから抽出することでソリッドモデルを作成する。

セル分割モデル

図 6.4 (a) に空間を 3 つのセル、C_1, C_2, C_3 に分割したセル分割モデルの例を示す。ここから適当なセルを選ぶことによって様々な立体形状を表現することができる。あるセルが立体形状の一部として選択されたとき、 そのセルを active と呼び C_1 のように書く。また active でないセルを$\overline{C_1}$ のように書くものとする。図 6.4(b),(c) はそれぞれ、$\{C_1, C_2, C_3\}, \{C_1、C_3\}$ を選ぶことによって生成される形状である。一般に、 n 個のセルを組み合わせて生成できる立体形状は空集合も含めて 2^n 通りである。セル分割モデルを用いると、多数の立体形状を単一の形状モデルで表現できるので、ソリッド合成の過程で現れる複数の候補立体を表現するのに都合がよい。

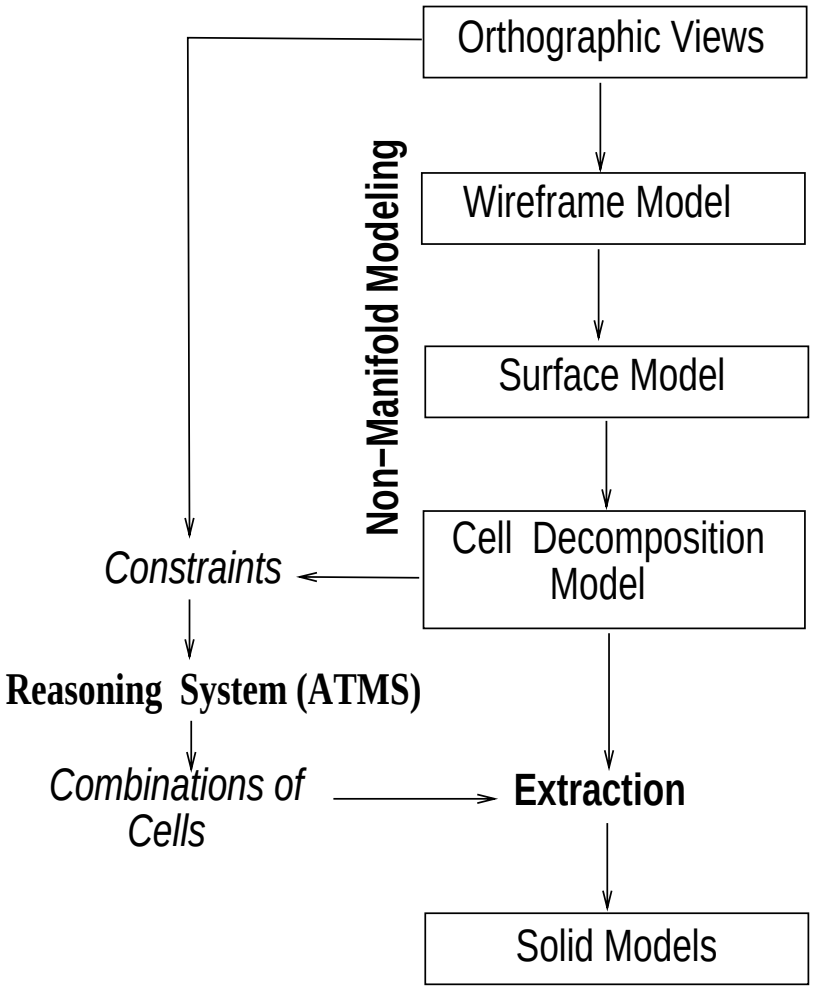


図 6.3: ソリッド合成の手順.

拘束条件とセルの選択

セルを選択して生成されるソリッドモデルのうち、三面図に一致するものが求めるべき解である。図 6.4 (a) の稜線 e_1 について考えてみると、(b) のように $\{C_1, C_2, C_3\}$ の場合には稜線 e_1 はソリッド解には現れないが、(c) のように $\{C_1, \overline{C_2}, C_3\}$ のときには現れる。このような条件を調べることによって、各稜線がソリッドモデルに現れる条件をセルに関する論理式として記述することができる。また、三面図の線分 d_1 に投影されるセル分割モデルの稜線が $e_1, ..., e_n$ であるとき、線分 d_1 を満たすためには $e_1, ..., e_n$ のうち少なくとも一つがソリッドの稜線として現れなければならない。したがって、三面図の各線分の条件もセルの条件で記述することができる。求めるべきソリッドモデルは、すべての三面図の線分の条件を満たすようなセルの組み合わせとし

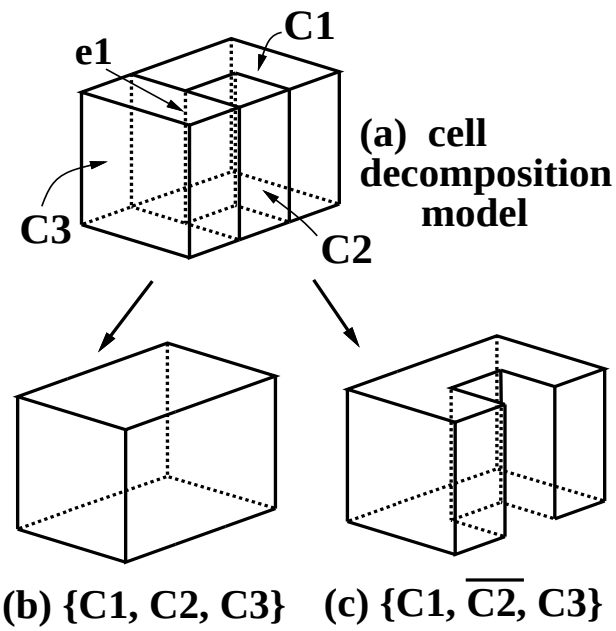


図 6.4: セル分割モデルから取り出されるソリッドモデル.

て得ることができる。

境界位相要素の抽出

セルの集合が得られると、それらの境界位相要素としてソリッドを得ることができる。active なセルの集合から生成される立体形状の境界位相要素は次の条件を満たすものである。ここでは、 取り出される位相要素を active と呼ぶ。

1. 隣接する active なセルがただ一つである face は active である。
2. active な face の境界上の edge において、連結するすべての active な face が同一曲面上にはないとき、その edge は active である。
3. active な edge の端点で、 連結なすべての active な edge が同一曲線上にはない vertex は active である。

図 6.4の (b), (c) はこの条件を満たす位相要素をセル分割モデルから取り出して表示したものである。この処理では、幾何計算は曲面と曲線の同一性の判定だけで、重い処理は一切行なわないので、 非常に高速に立体の境界要素を取り出すことができる。

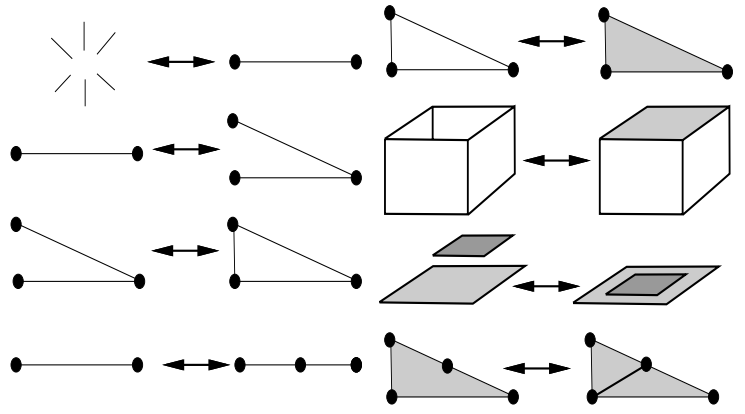
以後の節において、それぞれの処理について詳細に述べていく。

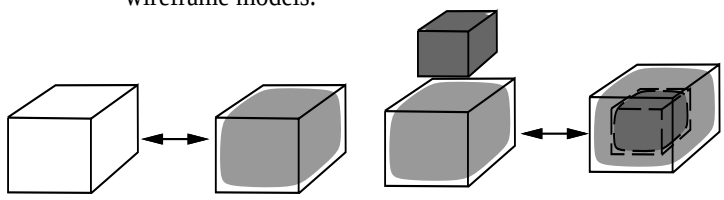
6.6 三面図からのセル分割モデルの生成

本節では、三面図からセル分割モデルを生成する手順について説明する。

6.6.1 非多様体形状モデルを用いた形状処理

三面図からワイヤフレームモデル、サーフェスモデル、セル分割モデルが順次生成されるが、それらに対する形状処理は、非多様体形状モデルのためのオイラー操作を用いることにより実現できる。図 6.5 は、処理の各段階で必要とされる位相変形操作について示している。





(c) Operations for cell decomposition models.

図 6.5: セル分割モデルの生成に必要なオイラー操作.

また、図 6.6 は、三面図からセル分割モデルを作成する手順を示したものである。各段階の形状処理について説明する。

(1) 候補稜線の生成 （ 図 6.6(a)→(b) ）

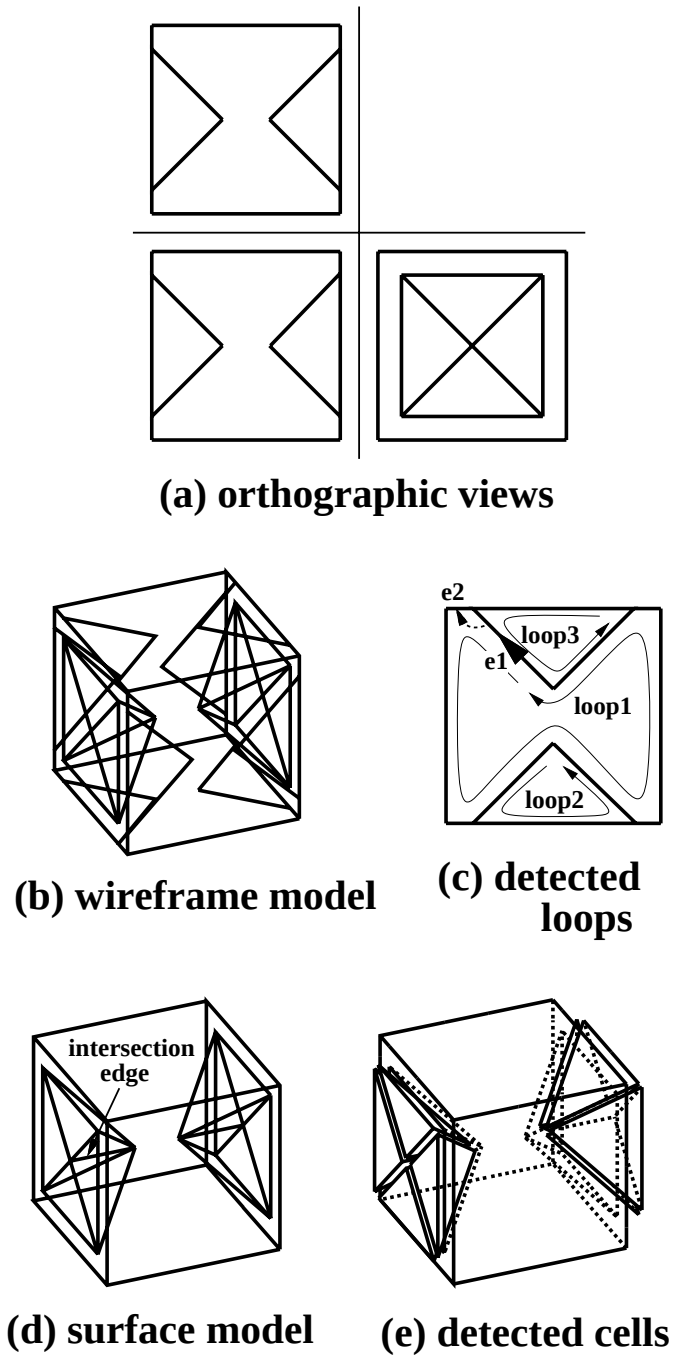
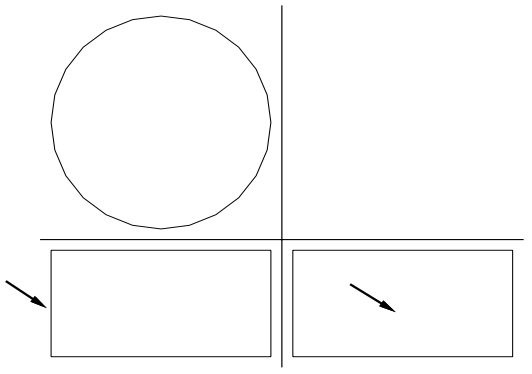


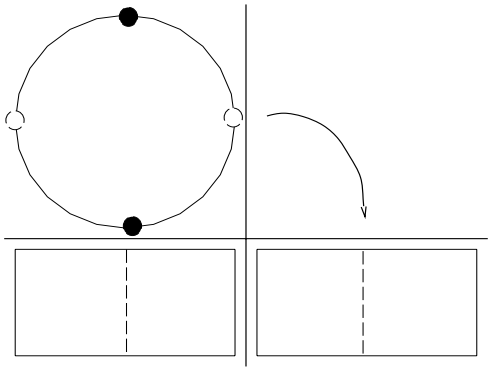
図 6.6: セル分割モデルの生成.

本手法では、直線と円弧で構成される三面図を考え、現れる曲面は、平面、球、及び中心軸が主軸と平行である円筒、円錐、トーラス面とする。

２次曲面を扱うときに問題となるのは、輪郭線として投影図に現れる部分が投影の方向によって異なることである。ここでは、三面図に円弧が現れたときには、その円弧から生成され得る曲面を三面図の線分を検索して調べ、接線連続な２曲面の境界稜線となり得る部分を三面図に付加的な線分として書き加える。書き加えた線分にはシルエットライン候補として属性を付けておき、後に述べる推論では区別して扱う。図 6.7 に例を示す。図 (a) の三面図の正面図において、矢印で示した線分に対応する箇所は側面図では矢印で示した部分になるが、この稜線は描かれていない。そこで、線分の対応が取れるように、図 (b) のように、上面図の白点と黒点で示した極値から破線で示したような補助稜線を正面図と側面図に生成する。図 (b) の破線は、円筒の場合に生成されるシルエットライン候補を示している。



(a)



(b)

図 6.7: 円筒の場合のシルエット稜線.

次に、三面図上の各頂点から、３次元座標上の頂点になる可能性のある組を選びだし、３次元の候補頂点を合成する。正面図、上面図、側面図の頂点の組 v_f, v_t, v_s の座標値をそれぞれ (x_f, y_f), (x_t, z_t), (y_s, z_s) とするとき、

$$x_f = x_t,\ y_f = y_s,\ z_t = z_s$$

が成立すれば、この三点からただ一つの３Ｄ頂点 (x_f, y_f, z_t) が決まる。このような組合せをすべて求めて３Ｄ頂点を決める。

さらに、求められた３Ｄ頂点の任意の２つを結ぶ線分の投影図が、与えられた三面図のすべてに頂点または線分として現れていれば、その線分はワイヤフレームモデルの edge となる。このとき、シルエットライン候補から生成されたワイヤフレームの edge にはシルエット稜線として属性を付けておく。得られたすべての稜線を図 6.5(a) のオイラー操作によって生成すると、図 6.6(b) のようなワイヤフレームモデルとなる。生成された edge には、元となる三面図の線分へのポインタを持たせておく。

なお、得られたワイヤフレームモデルにおいて、ぶら下がり稜線など、ソリッド に成り得ないものは除去しておく。この段階では、三面図のすべての稜線は、少なくとも一つのワイヤフレームの edge に対応しなければならないので、この条件を満たさない線分が三面図に存在すれば図面の誤りが存在することが分かる。

二面図の処理

ここで、二面図が与えられた場合の対応について述べておく。側面図が与えられない場合、正面図と上面図から側面図を合成する。もちろん側面図が一意に決定できるわけではなく、側面図として考えられるすべての稜線の集合を求めることになる。すなわち、合成された側面図には過剰な線分が含まれるが、上面図と側面図のみを満たすための条件で解探索を行なうことで対応できる。

図 6.8の場合を考えてみる。二面図において e_t と e_f は互いに対応する稜線である。正面図と上面図で、対応する頂点間に稜線があるかどうかを調べることによって、e_t と e_f から生成される側面図の稜線は、太線で示したクロスする二稜線であることが分かる。すべての組合せに関して稜線を求めると図のような側面図になる。このような側面図が合成された後は、推論の際の条件が異なる以外は、三面図が与えられた場合と同様の処理が行なえる。

(2) 面の生成 (図 6.6(b) → (d))

ワイヤフレームモデルの稜線を辿り、loop を探索することによって、face を生成する。

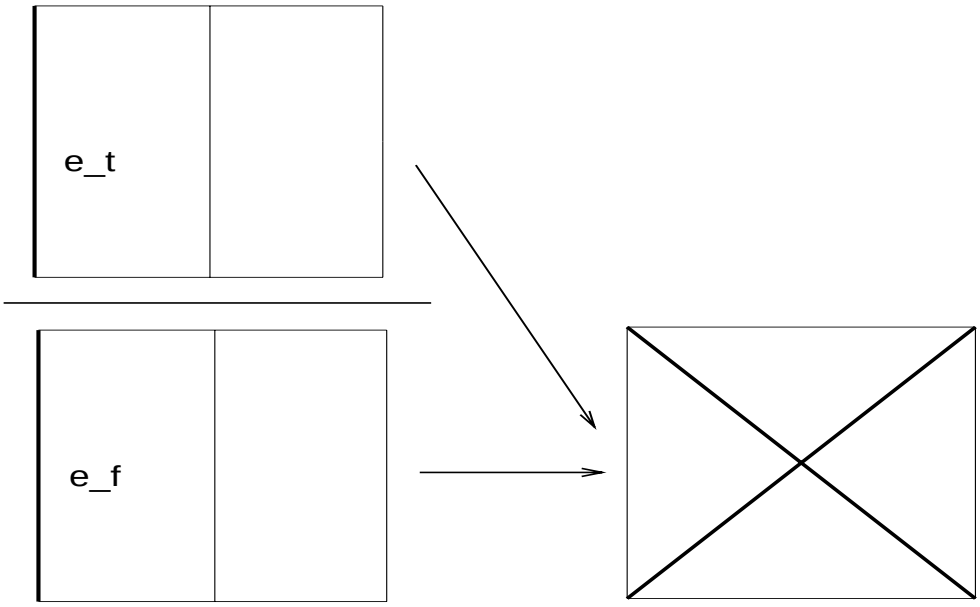


図 6.8: 正面図と上面図からの側面図の合成.(太線の稜線の組合せから側面図の太線の稜線が合成できる.)

ここでは、曲面を５種類に限定しているので、生成し得る曲面の方程式は、同一曲線上にない２つの edge の方程式から決定することができる。次に、得られた曲面上で反時計回りに辿るような edge の集まりを loop として求める。loop を探索するには、同一の平面または曲面に乗っていて、かつ時計回りに見て最も近い edge を順に辿っていく。図 6.6 (c) では、同一平面上に乗っているワイヤフレームモデルの edge を示しているが、この例で edge e_1 を矢印で示した向きに辿る場合、時計回りで最も近い edge は e_2 となる。このようにして e_1 に戻るまで順次 edge を辿っていくと１０本の edge から成る $loop_1$ を得ることができる。$loop_2, loop_3$ も同様にして得られる。

得られた loop には図 6.5(b) のオイラー操作を用いて face を生成する。このとき、loop 間の包含関係を調べ、もしある loop が他の loop を含んでいたら、二つ以上の loop で構成され、穴を持つ face が生成される。図 6.6(d) は、生成されたサーフェスモデルを示している。face の生成の際には他の face との干渉を調べ、もし face の境界以外で交わるならば干渉稜線を生成しておく。

なお、処理 (1) でシルエットライン候補を生成しているために、曲面 Sに関して主軸方向から見て輪郭線となる部分（$\partial S/\partial x = 0$, 又は $\partial S/\partial y = 0$, 又は $\partial S/\partial z = 0$ となる部分）には稜線が生成されることになる。従って、円筒と円錐は４分割、球とトーラスでは８分割され、それぞれ４枚と８枚の face で表現される。

(3) セルの生成 (図 6.6(d) → (e))

次に図 6.6(d) のサーフェスモデルから face に囲まれた閉領域を検出する。得られた face の集合を shell とする。本手法では、非多様体形状位相構造を用いているので、shell を取り出す処理は *radial* ポインタ を辿ることによって容易に行なうことができ、また得られた shell に図 6.5(c) で示す操作を行なうことによって volume として管理することができる。このとき shell 間の包含関係を調べ、もしある shell が他の shell を含んでいたら、二つ以上の shell で構成され、空洞を持つ volume が生成される。ここでは、volume をセルと呼ぶことにする。図 6.6では (e) に示すような９個のセルが検出される。この図では、わかりやすいようにセルを別々のソリッドであるかのように描いているが、これらは単一のセル分割モデルを構成する volume として管理されているものである。

以上の処理によって、三面図からセル構造モデルが生成でき、ソリッド 合成問題をセルの組合せを選ぶ問題に帰着させることができる。

6.7 ATMS を用いた解の算出

本節では、セル分割モデルから三面図に相当するセルの組み合わせを ATMS を用いて求める手法について述べる。

6.7.1 ATMS : Assumption-based Truth Maintenance System

ATMS は、de Kleer によって提案された、命題がどのような文脈で成立するかを管理するためのシステムである [deKleer86]。複数の文脈での推論を可能にするために，推論の際に現れる仮説の全体から、束 (lattice) を作ることによって可能な仮説の組み合わせすべてを扱うことができる。

ATMS では、推論の際に前提にした条件を仮説として表現することによって、ある命題を成立させるための究極の条件を計算する。以下は ATMS で用いられる用語である。

- 命題を導き出すための条件式を正当化式と呼ぶ。たとえば、条件 C のとき命題 P が成立するならば、正当化式は、$C \Rightarrow P$ となる。
- 仮説の組合せを環境と呼び、命題が成立するような環境の集合をラベルと呼ぶ。たとえば、命題を P 、仮説を C_n とし、$C_1 \cap C_2$ または $C_1 \cap C_3$ のときに命題 P が成立するときには、P のラベルは $\{C_1C_2, C_1C_3\}$ となる。C_1C_2、C_1C_3 のそれぞれが環境である。
- 矛盾を生じる仮説の集合を nogood 環境と呼ぶ。たとえば、仮説 C_1 と仮説 C_2 は同時に真となることはありえないとき、$C_1C_2 \Rightarrow nogood$ となる。また、ある環境において、これ以上仮説を追加すると常に矛盾を生じるとき、その環境を極大無矛盾環境と呼ぶ。

ATMS では、正当化式が次々に処理されていくと、データベースの整合性を維持するために命題のラベルを更新し、データに矛盾が生じた場合は nogood 環境に追加する。

本手法では、この機能を用いて三面図に対応するセルの組合せを求める。ここでは、三面図に対応するソリッドを求める問題に、ATMS を次のように当てはめる。

- セルが active であるという仮説を $C_1, C_2, ...$、active でないという仮説を $\overline{C_1}, \overline{C_2}, ...$ のように書く。

- セル分割モデルの edge e_n がソリッドモデルに現れるという命題をノード：$e_1, e_2, ...$ のように書き、e_n が成立するためのセルの条件を正当化式とする。正当化式は、
$$C_1, C_2, ... \Rightarrow e_n$$
のように記述される。

- 三面図の線分 d_n がソリッドモデルの投影図に現れるという命題をノード：$d_1, d_2, ...$ のように書き、d_n を成立させる条件を正当化式とする。

6.7.2 正当化式の算出

ATMS を用いて解の探索を行なうためには、まず、正当化式を求め、それを ATMS に与えてデータを更新することが必要である。そこで、本問題において成立する正当化式を求める。

(1) セル分割モデルの edge がソリッドの稜線となる条件

まず、セル分割モデルの edge が立体形状の稜線として現れる条件を求める。セル分割モデルでは稜線の周りにセルが存在するが、どのセルを active にするかによって稜線が見えるかどうかが決まる。edge がシルエット稜線でないとき、この条件は「稜線周りに active な face が二つ存在し、かつそれらが同一曲面上にない」と記述できる。

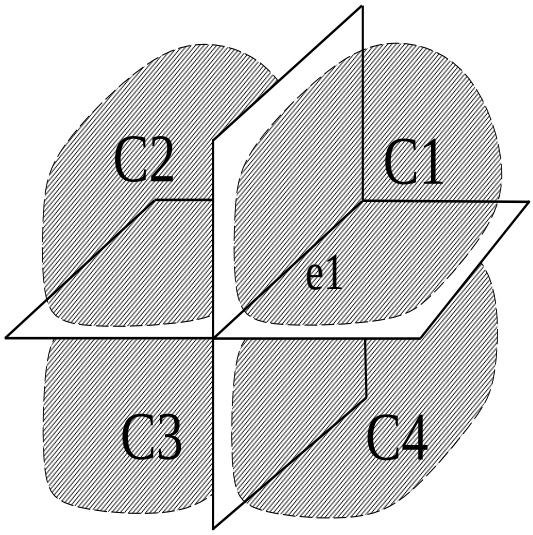


図 6.9: 稜線の回りのセル.

この条件を、図 6.9 に当てはめて考えてみる。この図では、edge e_1 の周りに４つのセル $\{C_1, C_2, C_3, C_4\}$ が存在するが、これらのセルの組み合わせのうち、e_1 がソリッドモデルの稜線として現れ、かつ非多様体稜線を生じないのは、

$\{C_1, \overline{C_2}, \overline{C_3}, \overline{C_4}\}$, $\{\overline{C_1}, C_2, \overline{C_3}, \overline{C_4}\}$,
$\{\overline{C_1}, \overline{C_2}, C_3, \overline{C_4}\}$, $\{\overline{C_1}, \overline{C_2}, \overline{C_3}, C_4\}$,
$\{C_1, C_2, C_3, \overline{C_4}\}$, $\{C_1, C_2, \overline{C_3}, C_4\}$,
$\{C_1, \overline{C_2}, C_3, C_4\}$, $\{\overline{C_1}, C_2, C_3, C_4\}$,

の 8 通りである。そこで、これらを 8 個の正当化式としてそれぞれ

$$C_1\overline{C_2C_3C_4} \Rightarrow e_1.$$

のように表現し、ATMS に与える。

(2) 非多様体稜線を禁じる条件

本手法では、セル分割モデルを表現するために非多様体形状モデルを用いており、解ソリッドとして非多様体となる形状を算出することも可能である。しかし、現実の工業部品が非多様体形状となることはあり得ないので、解ソリッドから非多様体稜線を生じるものを除外することにする。そのために、一本の稜線が三枚以上の active な face に共有されるのを禁じる条件を加える。図 6.9の稜線 e_1の場合、非多様体稜線となるセルの組合せは $\{C_1\overline{C_2}C_3\overline{C_4}\}$ と $\{\overline{C_1}C_2\overline{C_3}C_4\}$ の場合なので、

$$C_1\overline{C_2}C_3\overline{C_4} \Rightarrow nogood.$$
$$\overline{C_1}C_2\overline{C_3}C_4 \Rightarrow nogood.$$

という条件を ATMS に与える。

(3) シルエット稜線のための条件

シルエット稜線 は、ある特定の方向から見た場合のみ輪郭線として投影図に現れる。そのため、(1) とは異なった条件を与える必要がある。edge e_1 が、平面 P に投影したときに輪郭線として投影図に現れるのであれば、edge e_1 が満たすべき条件は「e_1 の周りに active な face が二つ存在し、少なくともその一方が平面でなく、かつ二つの face の e_1 上での接平面が平面 Pに垂直となる」と記述できる。

この条件を三面図に当てはめると、x-y, y-z, z-x 平面の少なくとも一つについてこの条件を満たすような場合を考えればよい。そのようなセルの組み合わせをシルエット稜線のための正当化式として ATMS に与える。

(4) 干渉稜線のための条件

また、セル分割モデルを作成する場合に、面候補同士が交わると図 6.6(d) のように干渉稜線が生じる。干渉稜線は三面図に現れてはならないので、干渉稜線が見える条件は nogood 環境となる。もし、稜線 e_1が干渉稜線のときは、

$$e_1 \Rightarrow nogood.$$

とすることによって，この稜線が見えるためのセルの組合せがすべて nogood 環境となる。

(5) 三面図の線分がソリッドの投影図に現れる条件

次に、セル分割モデルと三面図との関係を考える。三面図に描かれた線分は、ソリッドモデルの稜線として現れることが必要である。いま、三面図の線分 d_1 から生成された edge を $e_1, ..., e_p$ とすると、これらの edge のうち少なくとも一つはソリッドモデルの稜線とならなければ d_1 がソリッドモデルの投影図として現れない。このことは次のような p 個の正当化式として表現できる。

$$e_1 \Rightarrow d_1.$$
$$e_2 \Rightarrow d_1.$$
$$\ldots$$
$$e_p \Rightarrow d_1.$$

(6) すべての線分がソリッドの投影図に現れる条件

また、与えられた三面図の線分のすべてが投影図として現れるという条件を、存在すべき線分の集合によって正当化されるノード $views$ として次のように書く。

$$d_1\, d_2 \ldots d_m \Rightarrow views.$$

この正当化式を ATMS に与えると、$views$ を成立させるための条件として、セルの組み合わせを要素とするラベルが計算される。

6.7.3 極大無矛盾環境の算出

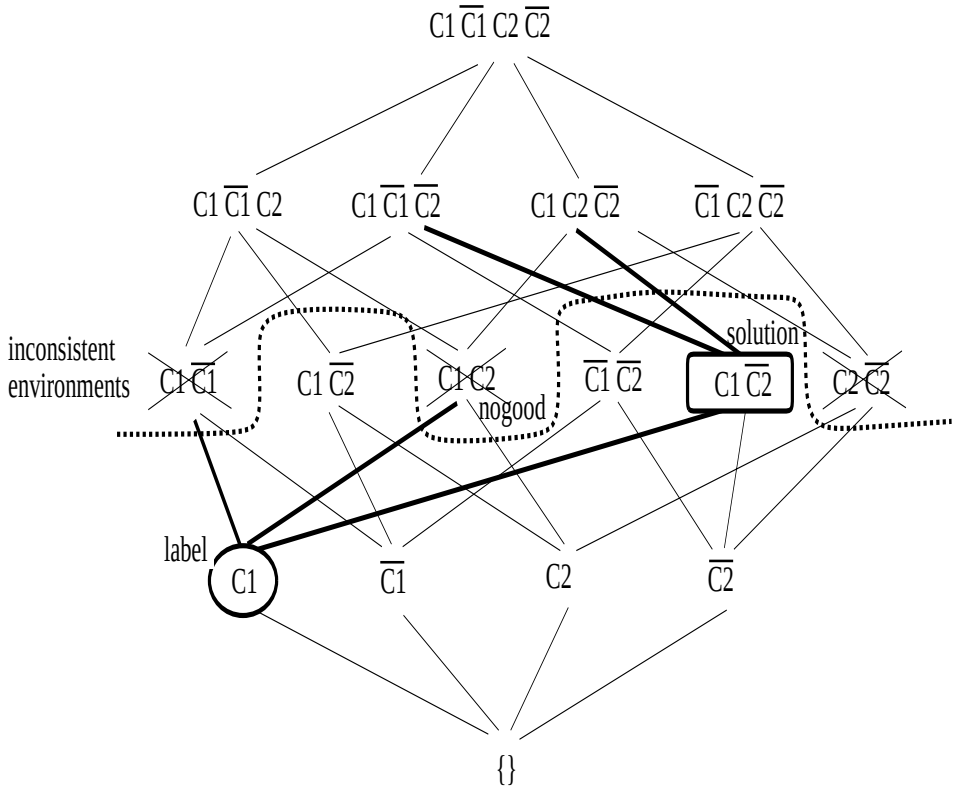


図 6.10: 全体集合の束と極大無矛盾環境.

以上の条件を ATMS に与え、ラベル更新処理を行なうことによって、ノード $views$ のラベルがセルの集合として求まる。次に、三面図と一致するソリッドモデルを求めるために、このラベルと nogood 環境から $views$ を成立させるための極大無矛盾環境を求める。

このことを図 6.10 を用いて説明する。この例では、セルが C_1, C_2 の二つのみで、セルの否定を示す仮説 $\overline{C_1}, \overline{C_2}$ を含む４つの仮説 $C_1, \overline{C_1}$, C_2 , $\overline{C_2}$ が存在する。図 6.10 はこれらから構成される束である。ここで、いま views のラベルが $\{C_1\}$, また、環境 C_1C_2 が干渉稜線か非多様体条件のいずれかの理由で nogood であるとする。また、$C_1\overline{C_1}$, $C_2\overline{C_2}$ が真となることはあり得ないので、これらも nogood である。

このとき、図の破線より上が矛盾した環境となる。一方、ラベル $\{C_1\}$ が真であることがソリッド解の必要条件となっている。したがって解は、C_1 のスーパーセットの中に存在し、このうち図の太線で示した $\{C_1\}$ を含んだ環境で矛盾を生じない $C_1\overline{C_2}$ のみが解となる。このように、ラベルのスーパーセットで矛盾を生じないセルの組み合わ

せは、極大無矛盾環境を求めることによって得ることができる。

ここでの計算量はラベル更新処理と矛盾環境を算出する処理の和になる。図の破線で示したようなすべての矛盾環境を算出する際の計算量は、一般には nogood 環境の数の指数のオーダであり、非常に計算時間がかかる。しかし、本アプリケーションの場合には、極大無矛盾環境算出の処理を $views$ のラベルを含むようなスーパーセットという限られた領域に限定して適用することができる。しかも、仮説の数に比べて nogood 環境の数がそれほど多くないという特徴があるので、計算量の爆発は生じにくく、この方法は有効なものになっている。

6.8 ソリッド合成の例題

本節では、本手法を幾つかの例題に対して適用した結果について述べる。

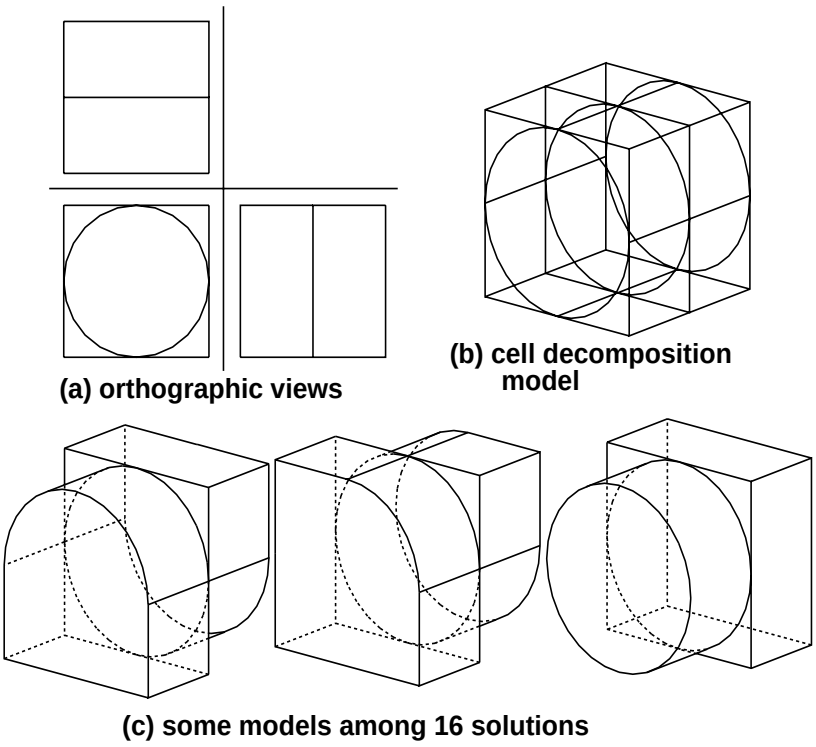


図 6.11: １６個の解を持つ三面図の例題.

本論文で述べた手法を図 6.6、6.11、6.12 の例に適用した。三面図は隣接関係の定義されていない線分の集合として与えられ、２線分の端点がある誤差の範囲内で一致していれば連結であるとみなす。またここでは、三面図の実線、破線の区別はしないものとする。以上の条件で三面図を読み込んでセル構造モデルを生成するまでの処理時間 (elapsed time) と、ATMS を用いてすべての解を求めるまでの処理時間をそれぞれ測定し、表 6.1 に示した。用いた計算機は、IBM RS/6000-980 である。

図 6.6 と図 6.11 は複数の立体形状が解として求まる例である。図 6.6 では、解ソリッドの個数は 13 個、図 6.11 では解ソリッドは全部で 16 個あり、そのうちのいくつかの例を図 6.11 (c) に示した。

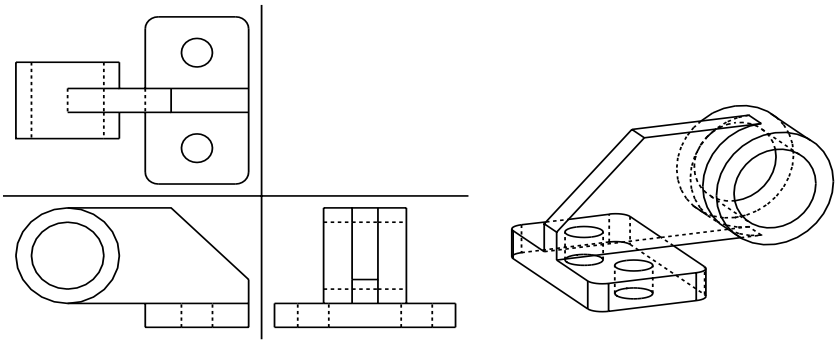
一方、図 6.12 は解が唯一つに決まる例で、三面図とその解ソリッドを示した。生成されるセル分割モデルは曲面を含んでおり、図 6.12 (a) では、平面 (22 枚) と円筒面 (26

例題	個数			処理時間		
	面の個数	セルの数	解の個数	セル生成	推論	合計
図 6.6	26	9	13	0.38 s	0.99 s	1.37 s
図 6.11	39	10	16	0.49 s	1.56 s	2.05 s
図 6.12a	48	5	1	0.95 s	0.43 s	1.38 s
図 6.12b	33	6	1	0.81 s	0.44 s	1.25 s
図 6.12c	51	6	1	0.89 s	0.31 s	1.20 s
図 6.12d	57	9	1	1.22 s	0.62 s	1.84 s

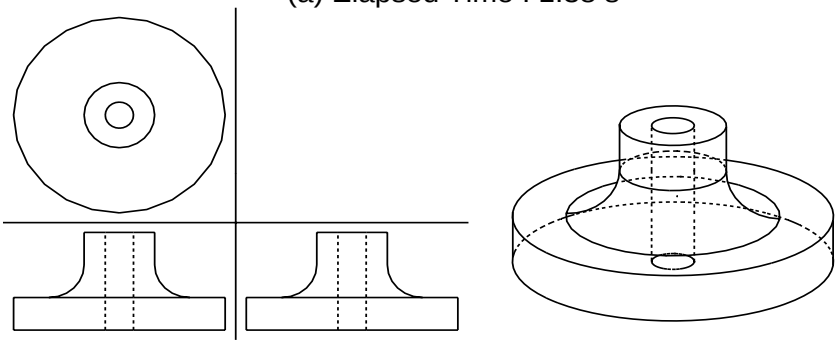
表 6.1: 計算時間 (IBM RS/6000-980).

枚)、図 6.12 (b) は、平面 (9) と 円筒面 (20) とトーラス面 (4)、図 6.12 (c) は、平面 (19) と円筒面 (28) と円錐面 (4)、図 6.12 (d) は、平面 (17)、円筒面 (32)、円錐面 (4)、トーラス面 (4) から構成されている。

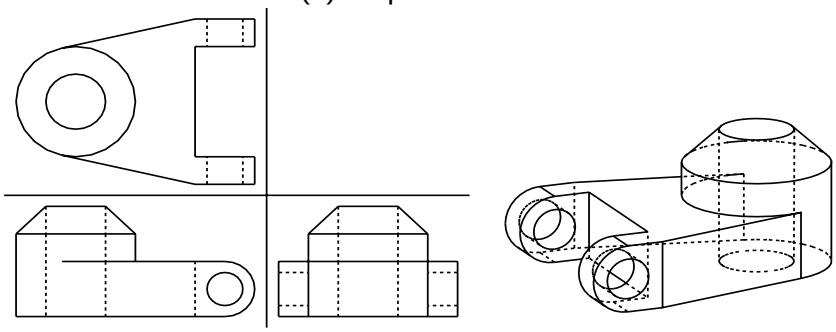
三面図からのソリッド合成の計算時間を表 6.1 に示したように、本手法を用いることによって、複数の解が存在する場合や曲面を含んだ場合でも十分実用的な時間で三面図からのソリッド合成ができることがわかる。



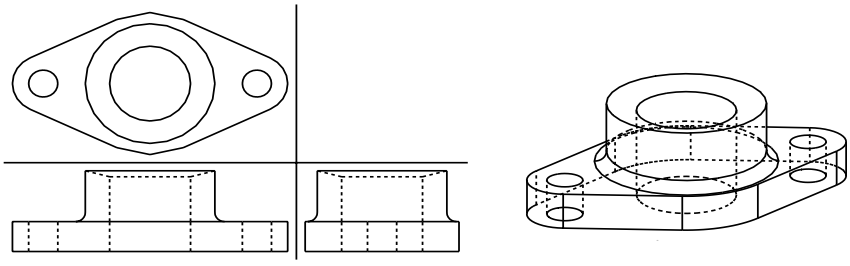
(a) Elapsed Time : 1.38 s



(b) Elapsed Time : 1.25. s



(c) Elapsed Time : 1.20 s



(d) Elapsed Time : 1.84 s

図 6.12: 曲面を含んだ形状の三面図の例題.

6.9 矛盾した線分を含んだ三面図の処理

本節では、三面図に誤りが含まれており、本来あってはならない線分が存在する場合のソリッド合成手法について述べる。そのために、三面図のうち少なくとも一つは正しいという仮定の元に拘束式を算出することを考える。

6.9.1 余分な線分を含んだ三面図

三面図からソリッドモデルを生成しようとする研究は古くから行なわれており、様々な手法が提案されている。しかし従来手法の問題点は、与えられた三面図に誤りがないことを前提としている点である。三面図に誤りのある場合は解が得られず、しかも、誤った箇所が必ずしも検出できないため、三面図修正に非常に労力を要する。

一般的なソリッドモデル合成手法では、まず、三つの投影図から対応する線分を選びだして３次元の稜線を求める。このとき、正しい三面図ならば稜線生成の段階で現れる稜線は過剰であってもよいが、足りないことはありえない。この条件を利用して、対話的に足りないワイヤフレームの稜線を補うことによって、ある程度の誤り修正を行うことは可能である。しかしながら、このような方法を用いたとしても、三面図に過剰な線分がある場合は、誤った部分を自動的に検出することはできない。

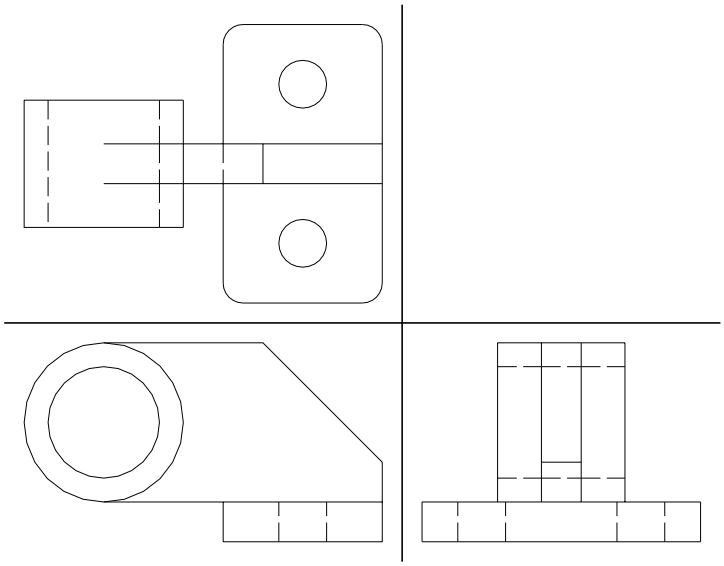
図 6.13 は、誤った線分を含んだ三面図の例である。これらはいずれも本来あってはならない線分を含んでいるが、この例題のように直観的にわかりにくい誤りでは、設計者に検出を期待するのは無理があると考えられる。

そこで、本節で述べたソリッド合成法に基づき、三面図に誤った線分が含まれている場合に対応できるような手法を考える。

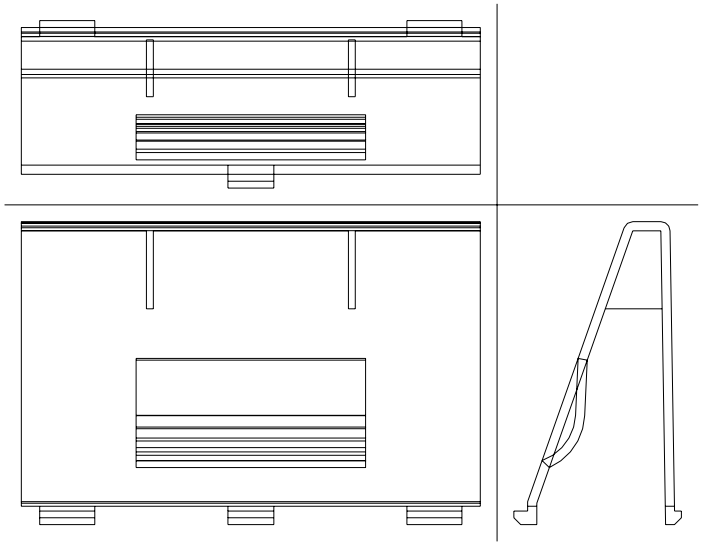
6.9.2 誤り検出法

三面図に誤った線分が含まれている場合、すべての線分が投影図として現れるようなソリッドモデルは得られない。そこで、本手法では、三面図の三つの投影図のうち少なくとも一つ、または二つが正しいという仮定の下で候補ソリッドを求めることにする。

本手法においては、このような条件で解探索することは容易である。ATMS に与える正当化式のうち、views の正当化式を変更するだけでよい。すなわち、上面図 P_t、正面図 P_f、側面図 P_s を構成する線分をそれぞれ、



(a)



(b)

図 6.13: 誤りを含んだ三面図.

$t_1 t_2 ... t_l \Rightarrow P_t.$
$f_1 f_2 ... f_m \Rightarrow P_f.$
$s_1 s_2 ... s_n \Rightarrow P_s.$

のような正当化式として ATMS に与えるとき、少なくとも一つの投影図を満たす条件の場合には、

$P_t \Rightarrow views.$
$P_f \Rightarrow views.$
$P_s \Rightarrow views.$

を正当化式として与え、また、少なくとも二つの投影図を満たす条件で解探索を行ないたい場合には、

$P_t P_f \Rightarrow views.$
$P_f P_s \Rightarrow views.$
$P_s P_t \Rightarrow views.$

を views のための正当化式として与えればよい。あとの処理は同じである。

ここでは、三面図のうち少なくとも一つを満たすソリッドの条件に関して、図 6.14 の例を用いて具体的に説明する。この図において、t_i, f_i, s_i はそれぞれ上面図、正面図、側面図の線分を示している。この三面図を用いて、本章で述べたセル分割モデルを生成したものが (b) であり、e_i は３Ｄ稜線、C_i はセルを表している。 ここで、t_i, f_i, s_i を候補ソリッドの投影図に含まれるとき真となる２値変数であると考える。

このとき、三面図の各投影面を P_t (上面図)、P_f (正面図)、P_s (側面図) とすると、各投影図の満たすべき条件はブール式で、

$P_t = t_1 t_2 ... t_8$
$P_f = f_1 f_2 ... f_{13}$
$P_s = s_1 s_2 ... s_{13}$

と書くことができる。ここで、三面図の線分 t_i, f_i, s_i が真となるのは、対応する３次元の稜線の集合 $\{e_i\}$ の少なくとも一つが真である場合である。たとえば、図 6.14において、投影図が t_6 と一致するセル分割モデルの稜線は、e_1, e_2, e_3 の三本で、このうち少なくとも一つがソリッドの稜線になるとき、t_6 が真となる。そこで、いま e_2 が候補ソリッドの稜線となる条件について考えてみると、e_2 に隣接しているセルは図 6.14 の C_1, C_2, C_3 の三つなので、e_2 が候補ソリッドの稜線として現れるのは、

$\{C_1 C_2 C_3, \overline{C_1} C_2 C_3, \overline{C_1} C_2 \overline{C_3}, C_1 \overline{C_2 C_3}, \}$

の４通りの場合で、e_2 が候補ソリッドの稜線となる条件は、

$e_2 = C_1 C_2 C_3 \cup \overline{C_1} C_2 C_3 \cup \overline{C_1} C_2 \overline{C_3} \cup C_1 \overline{C_2 C_3}$

と書くことができる。このような関係式を e_1, e_3 についても、同様に、

$e_1 = \overline{C_1} C_2 \cup C_1 \overline{C_2}$
$e_3 = C_3$

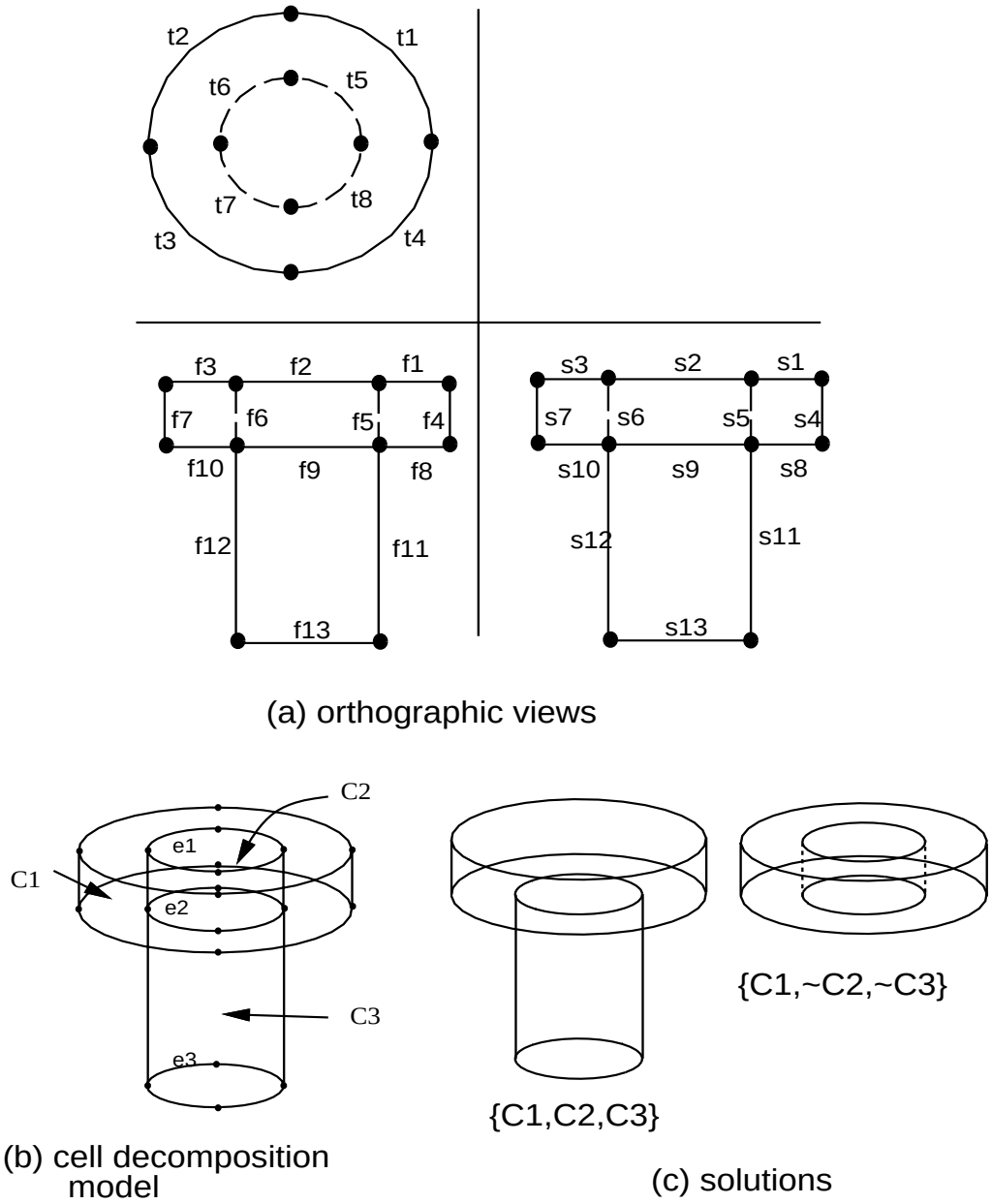


図 6.14: 誤りを含んだ三面図とセル分割モデル.

と求めることができる。

よって、三面図の線分 t_6 が真となる条件は、

$$t_6 = e_1 \cup e_2 \cup e_3 = \overline{C_1} C_2 \cup C_1 \overline{C_2} \cup C_3$$

となる。このような条件をすべての三面図の線分について求めて、その AND をとれば、三面図 P_t, P_f, P_s のそれぞれが真となるための条件がセルの選択条件として記述できる。たとえばこの例では、

$$P_t = C_1 C_2 C_3 \cup C_1 \overline{C_2 C_3}$$
$$P_f = 0$$
$$P_s = 0$$

が得られる。

この例の場合には、どのようなセルの組み合わせも正面図や側面図を満たすことがないことがわかる。少なくとも一つの三面図が正しいという条件は、

$$P_t \cup P_f \cup P_s = C_1 C_2 C_3 \cup C_1 \overline{C_2 C_3}$$

となり、条件を満たす立体は、$C_1 C_2 C_3$ と $C_1 \overline{C_2 C_3}$の二つとなる。それぞれに対応する立体形状は 図 6.14 (c) に示した通りである。設計者がこのうち一方のソリッドを選択すれば、三面図のどの部分が誤っていたかが明らかになる。

6.9.3 余分な線分を含んだ三面図の例題

本手法を用いることにより、三面図に過剰な線分が含まれる場合において、少なくとも一つの投影図が正しければ、ソリッドモデルの候補を得ることができる。

図 6.15、6.16 は、本手法を図 6.13 で示した三面図に適用したものである。ここでは、三面図のうち少なくとも一つを満たすソリッドを求め、ソリッドの稜線として用いられなかった線分を矢印で示した。矢印で示した線分を除去することによって正しい三面図となる。なお、図 6.15 では、この条件のソリッド解は三つあるが、そのうちの一つのみを示している。

この方法では、ATMS に与える三面図の条件を変えるだけで、矛盾した線分の検出ができるので、誤り検出のための計算負荷は軽いものとなっている。したがって、ソリッド解候補作成までの処理時間は、誤りのない場合とほぼ同じ計算時間となる。本手法では、あってはならない線分が含まれていても、三面図のうち少なくとも一つが正しければ効率的にソリッド候補を生成できるので、実際の図面を処理する上で非常に強力である。

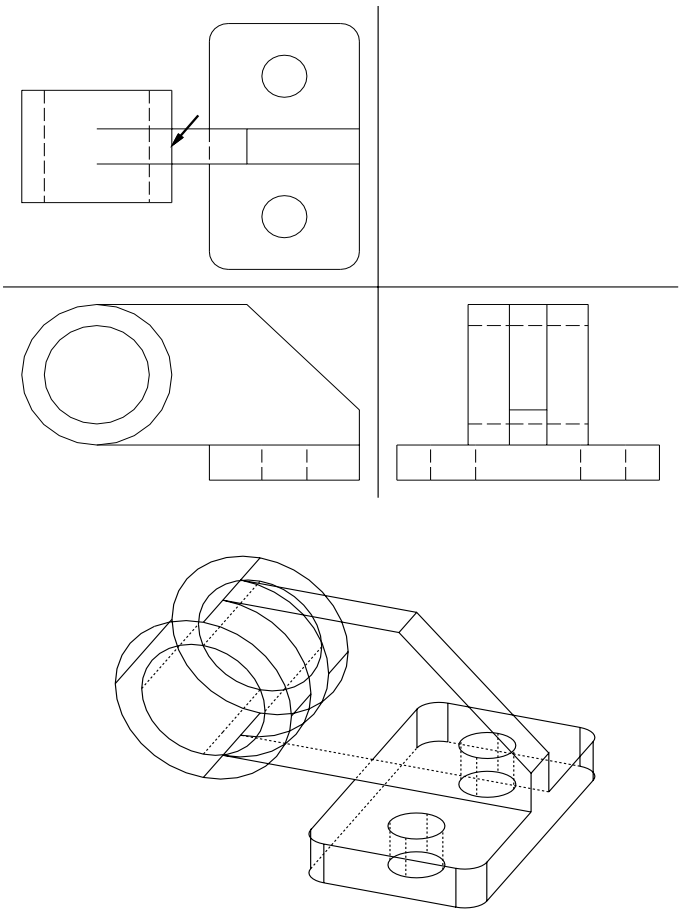


図 6.15: 誤りを含んだ三面図 (1).

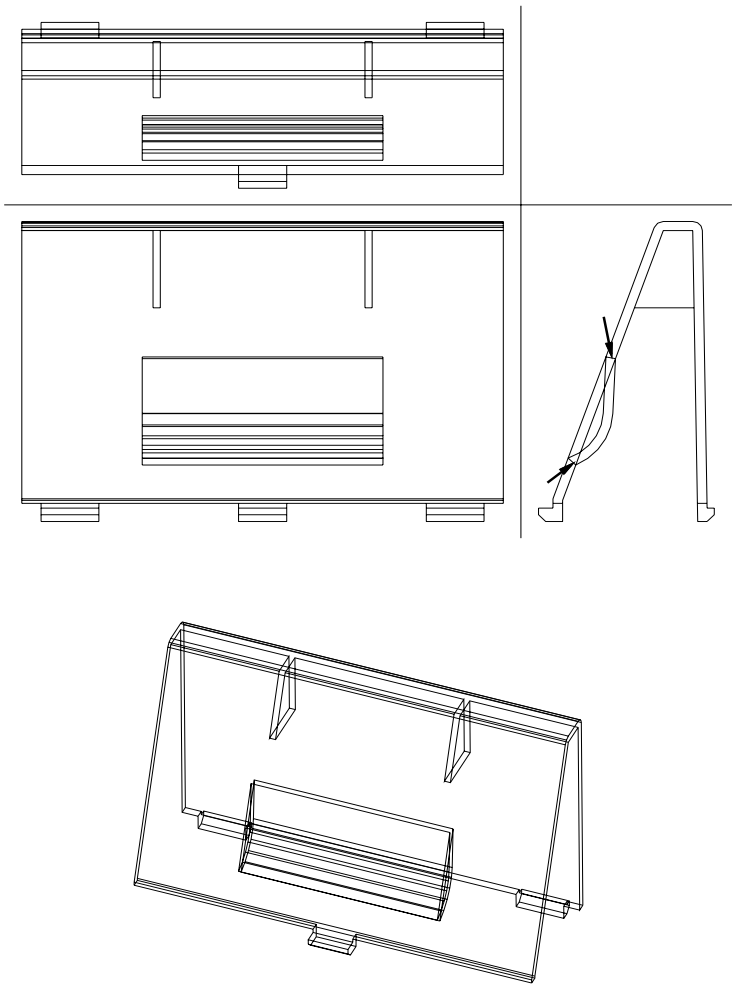


図 6.16: 誤りを含んだ三面図 (2).

6.10 線分の不足する三面図からのソリッド生成

本節では、線分の不足する三面図に対応する方法について述べる。ただし、不足する線分は水平または垂直であるという仮定を設ける。

6.10.1 不足する線分への対応

かくれ線が省略されている場合、十分なワイヤフレームモデルを作成することはできないため、従来のソリッド合成手法では対応することができない。そこでここでは、与えられた三面図からソリッドを生成する過程で三面図の矛盾箇所を検出し、その矛盾を解消するような線分を補うことを考える。ただし、一般に、省略された図形を補うためには対象に関する知識が必要であり、汎用的な手法ですべての場合に対応することは困難である。たとえば、図 6.17 はかくれ線の描かれた三面図 (a) と省略された三面図 (b) であるが、(a) の線分 h_1, h_2, h_3, h_4 がなければ円弧 c が貫通穴であることはわからないので、ソリッド解は有限個に確定しない。

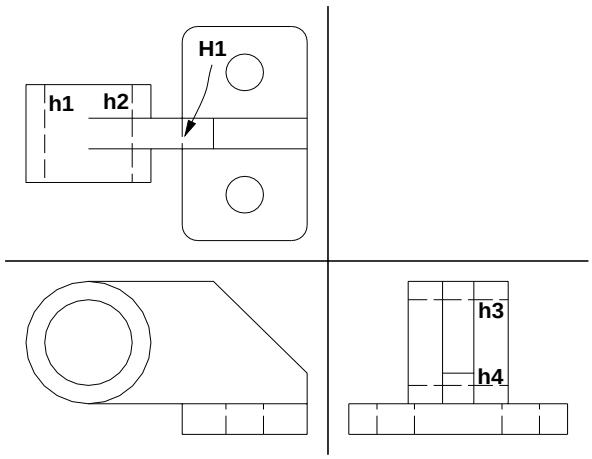
そこで、前提条件を設けて、省略線分の対象を限定することによって、欠けている線分を補うことにする。ここでは、欠けている線分は、垂直または水平な直線である場合に限定する。また、通常、図面でよく用いられる前提知識を用いて、解ソリッドの個数を絞り込む方法をとることにする。

矛盾箇所の検出と線分の追加

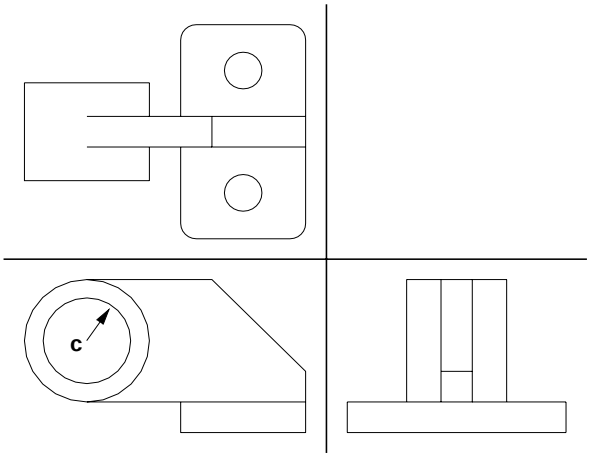
三面図として立体になりえない部分を検出し、その部分に対応した線分を補うための手順は以下の通りである。

1. 三面図に基づいて求められたワイヤフレームを各投影面に投影した図形 W と三面図 V との比較を行ない、$V - W$ を線分の集合 $\{d_i\}$ として求める。このとき、d_i は、ソリッドとして解釈することのできない線分である。そこで、仮想の線分として、元の三面図に d_i の両端点から他の投影図に水平または垂直の線分を加えることにより、線分 d_i をワイヤフレームの edge として解釈できるようにする。その上で、ワイヤフレームを再計算する。

2. さらに、サーフェスモデルで、面の境界とならないワイヤを除去した後、1 の処理を行なう。



(a) Orthographic views with hidden lines.



(b) Orthographic views without hidden lines.

図 6.17: かくれ線の省略された三面図.

3. 同様に、図 6.6 (d) のセルモデルでセルの境界上にない面とワイヤを除去した後、1 の処理を行なう。

この処理で追加した線分には、実際にはソリッドの稜線とならない余分な線分も含まれているが、それらに対応することは容易で、単に付加した線分に関するブール式を ATMS に与えなければよい。その場合、付加した線分はソリッドの稜線として現れても現れなくてもよい、という条件のもとでのソリッド解が得られる。

図面の前提知識の利用

ただし、加えた仮想線分が見えても見えなくてもよいという前提でブール式を解いた場合、解の個数が非常に多くなってしまうことがある。そこで、一般的に製図で用いられている前提知識を用いて解を絞り込むことを考える。ここでは、貫通穴の前提と、２次元パターンの前提の二つを用いる。

図 6.18 はかくれ線の省略された三面図と、生成されたセル分割モデルを示している。これらを用いて説明する。

一般に、穴のかくれ線が省かれた場合は貫通穴と見做される。そこでこの知識をブール式に反映させる。そのために、仮想線分の追加によって、図 6.18 の C_1, C_5 のような面を共有する円筒のセルが生成されたとき、これらに同じ解釈をするための正当化式を追加する。

すなわち、C_1 が active で、C_5 が active でないときには、途中で穴が埋まってしまったことを意味するので、次の条件を ATMS に加える。

$$C_1\overline{C_5} \Rightarrow nogood.$$
$$\overline{C_1}C_5 \Rightarrow nogood.$$

この条件を加えることにより、もし線分の追加によって穴が生成されるならば、必ず貫通穴になる、という条件が記述できたことになる。

また、図面においては、同じ２次元領域には同じ解釈を施すという暗黙の了解がある。図 6.18 の場合は、三面図に合同な円弧が４個現れている。これらに同じ解釈を施すためには、C_1, C_2, C_3, C_4 がすべて真かすべて偽になるというブール式を加えれば良い。

このような条件を付加することにより、常識的なソリッド解に絞り込むことができる。

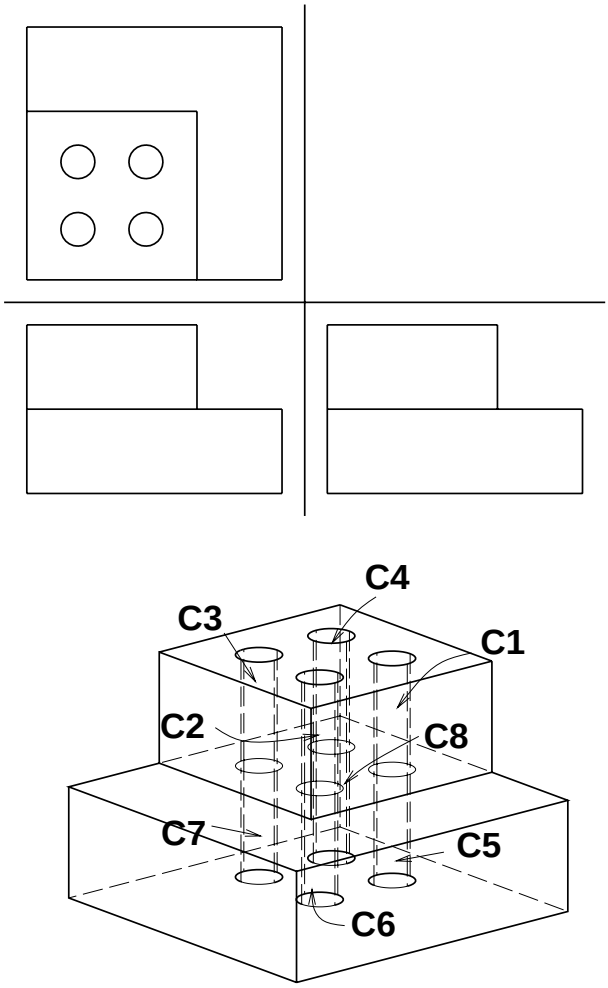
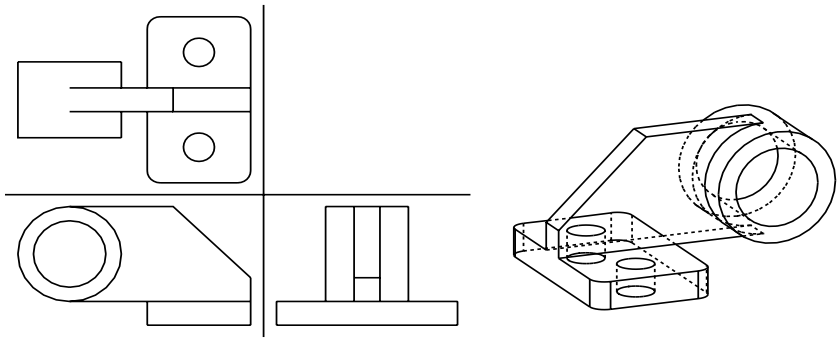


図 6.18: かくれ線の省略された三面図とセル分割モデル.

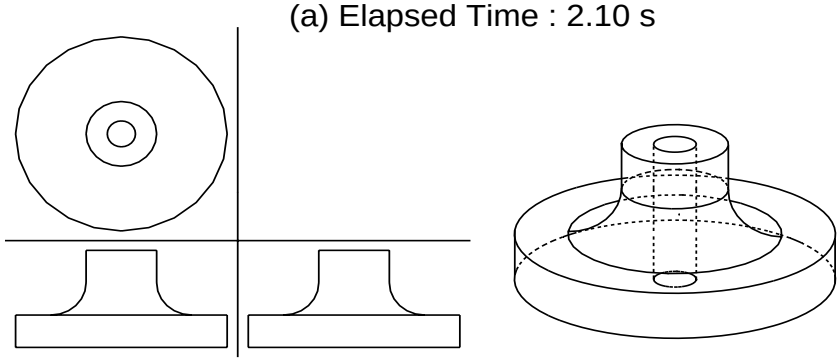
6.10.2 線分の不足した三面図の例題

本手法を幾つかの例題に適用した。図 6.19 に例を示す。この図は、図 6.12 と同じ例題からかくれ線をすべて除去したものである。ここで示した例では、前提条件を用いることにより、すべて唯一つのソリッド解になった。この例から、水平または垂直な線分の不足している三面図からでも、十分実用的な計算時間で適切な解が算出できていることがわかった。

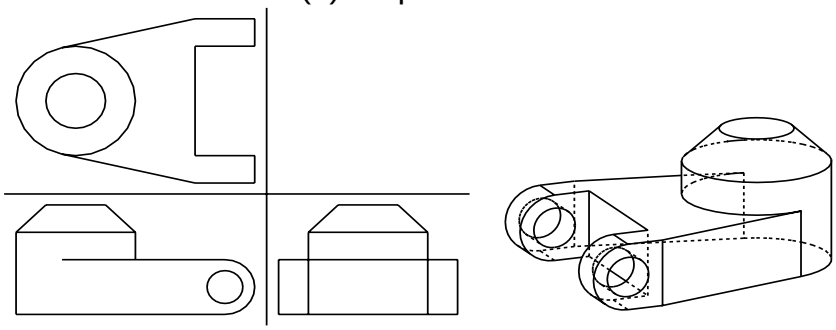
ただし、矛盾を解消する線分を加えた上で再処理を行なうので、正しい三面図に比べてその分計算時間がかかることも明らかになった。この例題では、正しい三面図を処理する場合に比べて、平均で 2 倍、最大で 3 倍の計算時間がかかっている。計算時間の増加に差があるのは、矛盾する線分が処理のどの段階で検出できたかが異なっているためである。



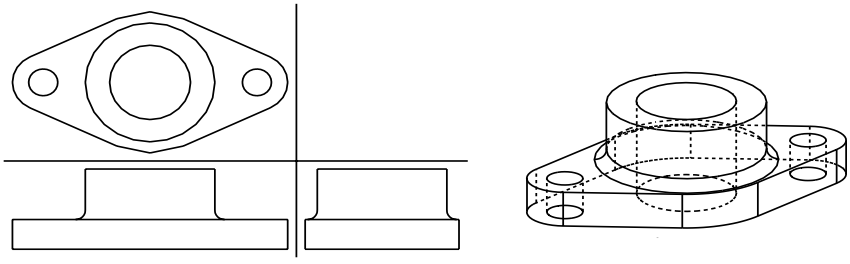
(a) Elapsed Time : 2.10 s



(b) Elapsed Time : 1.25 s



(c) Elapsed Time : 2.78 s



(d) Elapsed Time : 5.54 s

図 6.19: かくれ線の省略された三面図からのソリッド生成.

6.11 結論

本章では、三面図からのソリッドモデルの生成に関して、非多様体形状モデルと ATMS を用いた手法を提案し、十分実用的な処理速度でソリッド解が求まることを示した。また、誤りを含んだ三面図にもある程度対応できることを明らかにした。

要点は以下の通りである。

- 非多様体形状モデルを用いることにより、互いに関連のあるワイヤフレーム、サーフェス、ソリッド、セルモデルを単一のデータ構造によって表現できるので、個々の形状モデルを独立に表現して、関係を記述しておく必要がなく、処理が簡単化できる。
- すべてのソリッド解候補は、セル分割モデルを用いることにより、単一の形状モデルで表現でき、また、任意のセルの組み合わせから生成されるソリッドモデルが簡単に抽出できるので、処理が高速化できる。
- ソリッド解の推論に ATMS を用いる方法を示し、ATMS が本問題において有効であることを示した。また、ATMS に与える正当化式を変更することで、誤りを含んだ場合にも対処できることを示した。
- 余分な線分を含んだ三面図において、三面図のうち少なくとも一つが正しいという制約のもとでソリッドを生成する方法を示した。
- 線分の不足した三面図において、立体としての矛盾を検出し、矛盾を解消するために主軸に平行な線分を加えることによって、不足した線分を補う方法を示した。本手法は、不足した線分が水平または垂直な場合に限り有効である。

Chapter 7

結論

7.1 結論

本論文では、非多様体形状モデリングに関して、

1. 理論的な基礎の検討、
2. 非多様体形状モデリングシステム構築のための基礎技術の確立、
3. 応用システムの構築による有用性の実証、

の問題を扱った。得られた結論は以下の通りである。

１. 理論的な基礎

非多様体形状モデリングの基礎として重要な結論は以下の通りである。

1. 非多様体形状モデルの定義域を複体に準じて定めた。本定義に基づいて非多様体形状モデルのための基礎技術を確立し、また実際に非多様体形状モデリングシステムを実装し、さらに幾つかの応用に適用することを通して、本論文で定めた定義が理論的にも実用的にも非常に有用なものであることを示した。
2. 非多様体形状モデルのための集合演算の定義を、(1) 非多様体形状モデルの定義域 D に関して D×D から D への全射であり、(2) 正則な形状の集合演算が正

則集合演算に一致する、という条件を満たすように決めることができた。また、この定義に基づいた集合演算を実装し、幾つかの応用に適用することを通して、本定義が集合演算として適切なものであることを確認した。

3. 複体のオイラー・ポアンカレの式を用いることによって非多様体形状モデルの位相的な拘束式である

$$v - e + (f - r) - (V - Vh + Vc) = C - Ch + Cc$$

(v, e, f, V はそれぞれ vertex, edge, face, volume の個数、r, Vh は f ace, volume の穴の個数, Vc は volume の空洞の個数, C, Ch, Cc はそれぞれ形状モデルの連結成分, 貫通穴, 空洞の個数) を導出し、この式に基づいてオイラー操作が定義できることを示した。そして、非多様体形状モデルでは９種類の位相操作が独立であり、それらの操作を組み合わせて任意の位相構造が構築できるという定理を導いた。

それ以外の結論は以下に示す通りである。

1. 境界表現の非多様体形状モデルは、vertex, edge, face, volume 間の関係を記述することで表現できることを示した。

2. 非多様体形状モデルのためのオイラー・ポアンカレの式を適用することで、ワイヤフレームモデル、サーフェスモデル、ソリッドモデルなど、非多様体形状モデルの部分集合となっている幾何形状のためのオイラー操作が導出できることを示した。

２. 基礎技術

これらの性質に基づいて、データ構造や形状処理手法に関して基礎技術の検討を行なった。さらに、提案された基礎技術に基づいて非多様体形状モデリングシステムが実装できることを確認した。

重要な結論は以下の通りである。

1. volume, face, edge, vertex を位相要素とするデータ構造を提案し、それによって実現された形状処理システムは十分実用的な速度で動作することが確認できた。

2. 非多様体形状モデルの集合演算では、表現している点集合としては同一であっても、最小限の位相要素により演算結果を表現する方法と、元のプリミティブの位相構造を保持する表現法の二通りがあることを示した。

3. 非多様体形状モデルの集合演算の実現手法として、併合操作、抽出操作、簡略化操作の三つの操作を組み合わせた方法を提案した。そして、この集合演算が非多様体形状モデルの持つ位相的な柔軟性を十分に引き出せるものであることを確認した。

それ以外の結論は以下に示す通りである。

1. オイラー操作を実装するための仕様を示した。また実際に実装を行ない正常に動作することを確認した。

2. 併合操作と抽出操作は、CSG/Brep ハイブリッド構造の実現手段として有効であることを示した。

3. 併合操作と抽出操作を利用することで、優先順位を考慮した集合演算が実現可能であることを示した。

4. 併合操作は幾何計算とオイラー操作によって実装でき、抽出操作は集合の元を比較することで実装できることを示した。

３．応用システムの開発

非多様体形状モデルを利用して以下に示すような応用システムを実装し、非多様体位相構造が設計作業を支援する上で有用であることを確認した。

1. 集合演算の取消や修正を演算順序に依存せず高速に行なえる手法が、非多様体位相構造を利用することで実現できることを示した。そして、この手法を実装することで、多数の基本立体を組み合わせて生成された大規模なソリッドモデルでも高速に形状修正ができ、試行錯誤的な形状生成において非常に有効であることを確認した。

2. 形状特徴表現に必要な位相要素を保持した位相構造が非多様体形状モデルによって生成でき、また形状特徴や製品形状などの幾何形状が適宜抽出操作で取り出せることを利用して、形状特徴の表現、管理に適したシステムが構築できることを示した。

3. 三面図からソリッドモデルを合成する問題に非多様体形状モデルを用いることで、従来よりも簡潔で効率のよりアルゴリズムが実現できることを示した。そして、ワイヤフレーム、サーフェス、ソリッド、セル構造モデルが統一的なデータ構造で表現できることの利点を明らかにした。

全体として

本研究によって、十分な理論的な基礎に基づいた非多様体形状モデリングシステムが実装でき、また実用的に十分な効率で形状処理が行なえることが確認できた。

そして、非多様体形状モデルをいくつかの工学的応用に利用することによって、柔軟な位相構造を持つ、ということが設計や生産を支援するために大変有効であるという結論を得ることができた。

なお、本システムは C++ 言語を用いて実装され、形状処理ルーチンと API からなる約１３万６千行のプログラムコードから構成されている。コーディングはすべて筆者が行なった。

7.2 展望

本研究に関連して、非多様体形状モデリングにおいてさらに研究を要する問題を以下に示す。

1. データ構造の記述法に関する理論的な裏付けはまだ十分ではない。非多様体形状モデルは、vertex, edge, face, volume の組み合わせで表現できることはわかったが、それらの間にどのような順序関係を記述すればよいかについては理論的には十分説明されていない。n 次元多様体に関しては、Brisson の研究があるものの[Brisson90]、非多様体ではさらに数学的な面からの研究を必要とする。

2. 非多様体形状モデルは、設計や生産で利用される広範な形状を定義域に含んではいるが、有界な形状に限定されている。設計では無限直線や無限平面のような有界でない形状もしばしば用いられるが、有界でない幾何形状も統一的に扱うような理論やデータ構造は提案されていない。モデリング空間をユークリッド空間ではなく、射影空間とすることで解決できるかもしれない。

3. 非多様体形状モデルを本研究よりも広い定義域とすることは可能であるが、その場合にはオイラー操作が定義できるかが不明である。一般に、どのような性質を持つ幾何形状の集合が有限個のオイラー操作の組み合わせで記述できるかはわかっていない。

4. 集合演算では数値誤差の扱いが問題になるが、定義域を非多様体にまで広げた場合における、誤差の影響を受けにくい計算法についてはまだ明らかではない。

5. 集合演算を実現する方法として、併合操作と抽出操作を用いた手法を提案したが、最悪の場合ではデータ量が非常に増大してしまう可能性がある。そのような場合には簡略化操作を用いてデータ量を削減することになるが、形状モデルを監視して自動的に簡略化操作を起動させるには、起動条件の設定方法など、まだ研究を要する。

6. 本研究で試作した形状モデラには自由曲面は入れていない。非多様体形状モデルのデータ構造を構築するときには、edge 周りの face をソートすることが不可欠であるが、自由曲面を許した場合に、このソート作業の確実性や計算容易性がどうなるかについてはまだわかっていない。

7. 本論文で提案した集合演算の取消操作のアルゴリズムは、面の張り変えなどの局所変形操作が集合演算に混在した場合には対応できない。局所変形操作には undo

操作が有効であるが、取消操作と undo 操作を共存させる方法はまだわかっていない。

8. 三面図からのソリッド合成問題を示したが、図面には断面図などの場合も多い。断面図を対象とするにはさらに研究を必要とする。

謝辞

本論文は、1987 年に日本アイ・ビー・エムで３次元形状モデリングに関するプロジェクトが開始されて以来行ってきた研究をまとめたものです。このプロジェクトは、川辺真嗣氏、嶋田憲司氏、そして私の三人で始められ、当時は、design-by-feature、パラメトリックモデリング、非多様体形状モデリングなど、様々な新しい形状モデリング手法が現れてきた時期でした。研究を始めた当初は、非多様体位相に関する研究があまりなく手探り状態でしたが、その反面、手の付けられていないことが多く、非常に魅力的な研究テーマでした。テーマに関する示唆を頂いた川辺氏には深く感謝致します。

また、東京大学工学部の木村文彦教授には、本論文を執筆するために非常に多くの助言を頂きました。木村教授は、設計生産システムの分野で世界的に活躍されており、国内外の多くの委員会や会議、大学の講義などで非常に多忙である中、貴重な時間を割いて頂きました。また、木村教授には、修士論文の際にも指導して頂きました。本論文の背景となる多くの基礎知識はその当時に習得されたものです。深く感謝の意を表します。

東京大学工学部名誉教授である佐田登志夫教授には、卒業論文の御指導を頂くと同時に、佐田教授の研究室に入ることで、ＣＡＤの分野に携わるきっかけを与えて頂きました。深く感謝致します。

本論文で述べられているいくつかの重要なアイデアは、日本アイ・ビー・エムの嶋田氏と川辺氏との議論を通じて生み出され、発展してきました。両氏とは連日のように形状処理に関する激しい議論を行ない、様々なアイデアを生み出し、淘汰していく上で非常に大きな影響を受けました。新ためて感謝の意を表したいと思います。

途中からプロジェクトに加わった沼尾雅之氏にも、非多様体形状モデルの応用に関して多くの示唆を頂きました。特に第６章で述べた手法の推論機構の部分については、沼尾氏の強力なサポートにより実現することができました。また、推論エンジンは同じく途中からプロジェクトに加わった清水周一氏の研究開発したものを利用させていただきました。両氏にも感謝致します。

松家氏、香田氏にはマネージメントサポートをして頂きました。香田氏には、商用のＣＡＤベンダとの会議などを通じて、多くの知見を得る機会を与えて頂きました。両氏にも深く感謝致します。また、様々な助言やサポートを頂いた日本アイ・ビー・エム、東京基礎研究所の方々にも感謝の意を表します。特に、徳山豪氏には位相幾何学の問題に関して有益な助言をして頂きました。

最後に、非多様体形状モデリングの研究に関して助言や示唆を与えて頂いた東京大学工学部の鈴木宏正助教授、同教養学部の山口泰助教授、富山県立大学の小林一也助教授を始めとする形状処理の研究に携わってきた先輩諸氏にも感謝の意を表します。

論文リスト

1. S.Kawabe, K.Shimada, and H.Masuda,
'A Framework for 3D Modeling: Constraint-Based Description and Non-Manifold Geometric Modeling,'
Organization of Engineering Knowledge for Product Modeling in Computer Integrated Manufacturing, Elsevier, pp.325–354, Oct. 1988.

2. H.Masuda, K.Shimada, M.Numao, and S.Kawabe,
'A Mathematical Theory and Applications of Non-Manifold Geometric Modeling,'
Advanced Geometric Modeling for Engineering Applications, North-Holland, pp.89-103, Nov. 1989.

3. H.Masuda,
'Form-Feature Representation Based on Non-Manifold Geometric Modeling,'
MICAD 92, Hermes, pp.17–35, Feb. 1992.

4. H. Masuda,
'Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models,'
Computer Aided Design, 25(2):119-129, Feb. 1993.

5. 増田宏,
' 非多様体形状モデルを用いた形状特徴の表現と管理,'
精密工学会誌, 59(6):956-962, Jun. 1993.

H.Masuda,
'Representation and Maintenance of Form-Feature Based on Non-Manifold Geometric Modeling,' *Journal of Advanced Automation Technology (*精密工学会欧文誌*), Fuji Technology Press Ltd.*, 6(2):80-85, Mar, 1994.

6. 増田宏、沼尾雅之、清水周一,
' 非多様体形状モデラと ATMS を用いた三面図からのソリッド合成法,'
情報処理学会論文誌, 35(3):453-460, Mar. 1994.

研究発表リスト

1. 増田宏, 嶋田憲司, 川辺真嗣,
   ’非多様体モデルのためのオイラー・オペレーション,’
   情報処理学会：グラフィックスとＣＡＤシンポジウム論文集, Oct. 1988.

2. 増田宏, 嶋田憲司, 川辺真嗣,
   ’設計作業に適した３次元モデリングシステム 第１報：非多様体幾何に基づく形状モデリング,’
   昭和 63 年度精密工学会秋季大会論文集, Oct. 1988.

3. 増田宏, 嶋田憲司, 川辺真嗣,
   ’非多様体モデルにおける取り消し操作:境界表現における基本立体の保存,’
   元年度精密工学会春季大会論文集, Mar. 1989.

4. 増田宏, 嶋田憲司, 川辺真嗣,
   ’非多様体形状モデルの集合演算,’
   元年度精密工学会秋期大会論文集, Oct. 1989.

5. 増田宏, 嶋田憲司, 川辺真嗣,
   ’非多様体形状モデルの集合演算 第２報：集合演算の順序依存性の解消,’
   1990 年度精密工学会秋期大会論文集, Oct. 1990.

6. 増田宏,
   ’非多様体モデルにおける集合演算アルゴリズム,’
   グラフィックスとＣＡＤ研究会, May 1991.

7. 増田宏、沼尾雅之、清水周一,
   ’非多様体形状モデラを用いた三面図からのソリッド合成,’
   情報処理学会全国大会論文集, Oct. 1992.

8. 増田宏, 沼尾雅之,
’不完全な三面図からのソリッドモデルの合成,’
情報処理学会全国大会論文集, Oct. 1993.

9. 増田宏, 松澤裕史, 沼尾雅之,
’２次元図面からのソリッドモデル合成システム: ２次元パターンの利用とかくれ線の省略,’
情報処理学会全国大会論文集, Sep. 1994.

Bibliography

[Aldefeld83] B.Aldefeld, 'On Automatic Recognition of 3D Structures from 2D Representations,' *Computer Aided Design,* 15(2):59–64, Mar. 1983.

[Arbab90] F.Arbab, 'Set Models and Boolean Operations for Solids and Assemblies,' *IEEE Computer Graphics and Applications,* 10(6):76-86, Nov. 1990.

[Agarwal92] S.C.Agarwal and W.N.Waggenspack, 'Decomposition Method for Extracting Face Topologies from Wireframe Models,' *Computer Aided Design,* 24(3):123–140, Mar 1992.

[Aizawa86] 相沢民王, 大和功一, '幾何モデリングシステムの構造,' 第４回自動化工学講演論文集,' Jul. 1986.

[Aizawa89] T.Aizawa, 'A Cell Complex Chain Model of Data for CAE Databases: A Base for Unification of Geometric Models,' *Bulletin of the Japan Society of Precision Engineering,* 23(2):152–158, Jun. 1989.

[Baumgart74] B.Baumgart, 'Geometric Modeling for Computer Vision,' *PhD Thesis, Stanford University, STAN-CS-320,* 1974.

[Baumgart75] B.Baumgart, 'A Polyhedron Representation for Computer Vision,' *AFIP Conf. Proc.,* 44:589–596, 1975.

[Boyse82] J.W.Boyse and J.E.Gilchrist, 'GM Solid: Interactive Modeling for Design and Analysis of Solids,' *IEEE Computer Graphics and Applications,* 2(2):27-39, Mar. 1982.

[Braid73] I.C.Braid and C.A.Lang, 'Conputer-Aided Design of Mechanical Components with Volume Building Bricks,' *Proc. of PROLAMAT'73,* 1973.

[Braid75] I.C.Braid, 'The Synthesis of Solids Bounded by Many Faces,' *Communications of the ACM,* pp.209–216, Apr. 1975.

[Braid80] I.C.Braid, R.C.Hillyard and I.A.Stroud, 'Stepwise Construction of Polyhedra in Geometric Modelling,' *Mathematical Method in Computer Graphics and Design,* Academic Press, pp.123–141, 1980.

[Braid93] I.C.Braid, 'Boundary Modeling,' *in Fundamental Developments of Computer-Aided Geometric Modeling, Academic Press,* pp.165-183, 1993.

[Brisson90] E.Brisson, 'Representaion of d-Dimensional Geometric Objects,' *PhD. Thesis, University of Washington*, 1990.

[Brun76] J.-M.Brun, 'EUCLID: Equipe Graphique et les Sciences de l Ingenieur,' *LISMA,* 1976.

[Chiyokura83a] 千代倉弘明, '自由局面を持った立体の生成に関する研究,' 東京大学工学部精密機械工学科博士論文, 1983.

[Chiyokura83b] H.Chiyokura and F.Kimura, 'A Method of Representing the Solid Design Process,' *IEEE CG&A,*5(4):32–41, 1983.

[Chiyokura85] 千代倉弘明, 'ソリッドモデリング,' 工業調査会, 1985.

[Chiyokura88] H.Chiyokura, 'Solid Modeling with Design Base,' *Addison-Wesley,* 1988.

[Courter] S.M.Courter, 'Automated Conversion of Curvilinear Wireframe Models to Surface Boundary Model: A Topological Approach,' *ACM SIGGRAPH,* 20(4):171–178, Aug. 1986.

[Crocker91] G.A.Crocker and W.F.Reinke, 'An Editable Nonmanifold Boundary Representation,' *Computer Graphics and Applications,* 11(2):39-51, Mar. 1991.

[deKleer86] J.de Kleer, 'An Assumption-Based TMS,' *Artificial Intelligence,* 28(2):127–162, 1986

[Desaulniers92] H.Desaulniers and N.F.Stewart, 'An Extension of Manifold Boundary Representations to the r-Sets,' *ACM Transactions on Graphics,* 11(1), Jan. 1992.

[Dixon87] J.R.Dixon, et al, 'Expert Systems for Mechanical Design: Examples of Symbolic Representations of Design Geometries,' *Engineering with Computers, Splinger-Verlag,* pp1–10, 1987.

[Eastman86] C.M.Eastman and K.Presiss, 'A Review of Solid Shape Modeling Based on Integrity Verification,' *Comouter Aided Design,* 16(2):66-80, Mar. 1984.

[Gomes91] A.J.P.Gomes and J.C.G.Teixeira, 'Form Feature Modelling in A Hybrid CSG/Brep Scheme,' *Comput. and Graphics,* (2). 1991

[Gu85] K.Gu, Z.Tang, and J.Sun, 'Reconstruction of 3D Objects from Orthographic Projections,' *CG Forum,* (5):807–811, 1985.

[Gursoz88] L.Gursoz, Y.Choi and F.B.Prinz, 'Vertex-Based Representation of Non-Manifold Boundaries,' *Geometric Modeling for Product Engineering*, North-Holland, pp.107–130, Sep. 1988.

[Gursoz91] E.L.Gursoz, Y.Choi and F.B.Prinz, 'Boolean Set Operations on Non-Manifold Boundary Representation,' *Computer Aided Design,* 23(1):33–39, Jan. 1991

[Harada87] 原田毅士, 木村文彦, ' 三面図入力を基本にした三次元入力システムの試作,' グラフィックスと CAD, 28(2):15–22, Aug. 1987.

[Higashi90] 村端普一, 東正毅, ' 立体表現・干渉における位相管理,' 精密工学会論文誌, 56(9):71–76, Sep. 1990.

[Higashi91] 東正毅, 水野誠二, 弥富英樹, ' 非多様体サーフェスモデリングによるフィレット生成,' 精密工学会論文誌, 57(11):81–86, Nov. 1991.

[Higashi94] 東正毅, 水谷好宏, 中倉清, ' 非多様体シェルによる統一的形状モデリング,' 精密工学会論文誌, 60(11):1658–1662, Nov. 1994.

[Hoffman89b] C.M.Hoffmann, J.E.Hopcroft, and M.S.Karasick, 'Robust Set Operations on Polyhedral Solids,' *Computer Graphics and Applications*, 9(6):50-59, Nov. 1989.

[Hoffman89] C.M.Hoffmann, 'Geometric and Solid Modeling: An Introduction,' *Morgan Kaufmann Publishers,* 1989.

[Hosaka74] M.Hosaka, F.Kimura and N.Kakishita, 'A Unified Method for Processing Polyhedra,' *Information Processing '74, North-Holland,* pp.167-172, 1974.

[Idesawa72] 出沢正徳, ' 三面図からの物体生成のためのシステム,' 機械学会論文誌, 38(310):1267-1276, Jun. 1972.

[Idesawa73] M.Idesawa, 'A System to Generate A Solid Figure from A Three View,' *Bulletin of the Japan Society of Mechanical Engineering,* 16:216–225, Feb. 1973.

[Ito90] 伊藤潔, ' 三面図を用いたソリッドモデルの構成,' 情報処理学会誌, 31(8):1095–1106, Aug.1990.

[Kawabe88] S.Kawabe, K.Shimada, and H.Masuda, 'A Framework for 3D Modeling: Constraint-Based Description and Non-Manifold Geometric Modeling,' *Organization of Engineering Knowledge for Product Modeling in Computer Integrated Manufacturing, Elsevier,* pp.325–354, Oct. 1988.

[Kela89] A.Kela, J.E.Davis and P.M.Finnigan, 'Wireframe to Solid Models: A Survay,' *Proc. ASME Computers in Engineering Conference,'* 1:53–61, 1989.

[Kimura84] F.Kimura, 'Geomap-III: Designing Solids with Free-Form Surfaces,' *CG&A,* 5(1):21-40, 1985.

[Kimura89] F.Kimura, Y.Yamaguchi and K.Kobayashi, 'New Features of Geometric Modelling for Product Description,' *Advanced Geometric Modeling for Engineering Applications, North-Holland,* pp.35–52, Nov. 1989.

[Kobayashi87] 小林一也, 山口泰, 'Geomap-III の拡張機能,' 精密工学会産学協同研究分協議会研究協力分科会, 資料 *PR-S-1-1.*

[Kobayashi89] 小林一也, ' 構造に基づく製品モデルの形状生成法,' 東京大学博士論文, 1989.

[Kobayashi92] 小林一也, 山口泰, 木村文彦, ' 形状モデラのための基本インタフェース,' 精密工学会論文誌, 58(2):259–264, Feb. 1992.

[Laidlaw86] D.H.Laidlaw, W.B.Trumbore and J.F.Hughes, 'Constructive Solid Geometry for Polyhedral Objects,' *ACM Computer Graphics,* 20(4):161-170, Aug. 1986.

[Lienhardt89] P.Lienhardt, 'Subdivisions of n-Dimensional Spaces and n-Dimensional Generalized Maps,' in: *5th Symposium on Computational Geometry*, pp.228-236, 1989

[Lienhardt91] P.Lienhardt, 'Topological Models for Boundary Representation: A Comparison with n-Dimensional Generarized Map,' *Computer-Aided Design,* 23(1):59–82, Jan. 1991,

[Mantyla82] M.Mantyla and R.Sulonen, 'GWB: A Solid Modeler with the Euler Operators,' *IEEE Computer Graphics,* 2(7):17–31, Sep. 1982.

[Mantyla86] M.Mantyla, 'Boolean Set Operations of 2-Manifolds through Vertex Neighborhood Classification', *ACM Transactions on Graphics,* 5(1):1-29, Jan. 1986.

[Mantyla88] M.Mantyla, 'An Introduction to Solid Modeling,' *Computer Science Press,* 1988.

[Masuda88a] 増田宏, 嶋田憲司, 川辺真嗣, ' 非多様体モデルのためのオイラー・オペレーション,' 情報処理学会：グラフィックスとＣＡＤシンポジウム論文集, Oct. 1988.

[Masuda88b] 増田宏, 嶋田憲司, 川辺真嗣, ' 設計作業に適した３次元モデリングシステム 第１報：非多様体幾何に基づく形状モデリング,' 昭和 63 年度精密工学会秋季大会論文集, pp.445–456, Oct. 1988.

[Masuda89a] 増田宏, 嶋田憲司, 川辺真嗣, ' 非多様体モデルにおける取り消し操作:境界表現における基本立体の保存,' 元年度精密工学会春季大会論文集, pp.845–846, Mar. 1989.

[Masuda89b] 増田宏, 嶋田憲司, 川辺真嗣, ' 非多様体形状モデルの集合演算,' 元年度精密工学会秋期大会論文集, Oct. 1989.

[Masuda89a] H.Masuda, K.Shimada, M.Numao, and S.Kawabe, 'A Mathematical Theory and Applications of Non-Manifold Geometric Modeling,' *Advanced Geometric Modeling for Engineering Applications, North-Holland,* pp.89-103, Nov. 1989.

[Masuda89b] 増田宏, 嶋田憲司, 川辺真嗣, ' 非多様体形状モデルの集合演算 第２報：集合演算の順序依存性の解消,' 1990 年度精密工学会秋期大会論文集, pp.1219–1220, Oct. 1990.

[Masuda91] 増田宏, ' 非多様体モデルにおける集合演算アルゴリズム,' グラフィックスとＣＡＤ研究会, May 1991.

[Masuda92a] H.Masuda, 'Form-Feature Representation Based on Non-Manifold Geometric Modeling,' MICAD 92, Hermes, pp.17–35, Feb. 1992.

[Masuda92b] 増田宏、沼尾雅之、清水周一, ' 非多様体形状モデラを用いた三面図からのソリッド合成,' 情報処理学会全国大会論文集, Oct. 1992.

[Masuda93a] H. Masuda, 'Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models,' *Computer Aided Design*, 25(2):119-129, Feb. 1993.

[Masuda93b] 増田宏, ' 非多様体形状モデルを用いた形状特徴の表現と管理,' 精密工学会誌, 59(6):956-962, Jun. 1993.

[Masuda93c] 増田宏, 沼尾雅之, ' 不完全な三面図からのソリッドモデルの合成,' 情報処理学会全国大会論文集, Oct. 1993.

[Masuda94a] 増田宏、沼尾雅之、清水周一, ' 非多様体形状モデラと ATMS を用いた三面図からのソリッド合成法,' 情報処理学会論文誌, 35(3):453-460, Mar. 1994.

[Masuda94c] 増田宏, 松澤裕史, 沼尾雅之, ' ２次元図面からのソリッドモデル合成システム: ２次元パターンの利用とかくれ線の省略,' 情報処理学会全国大会論文集, Sep. 1994.

[Masuda94b] H.Masuda, 'Representation and Maintenance of Form-Feature Based on Non-Manifold Geometric Modeling,' Journal of Advanced Automation Technology, Fuji Technology Press Ltd., pp. 80–85, 6(2), Mar, 1994.

[数学 1] 服部晶夫, ' 岩波講座基礎数学 位相 幾何学 I,II,' 岩波書店, 1977.

[数学 2] 日本数学会編集, ' 岩波数学辞典第２版,' 岩波書店, 1968.

[数学 3] I.M.Singer and J.A.Thorpe, 'Lecture Notes on Elementary Topology and Geometry,' 1967. (邦訳: 赤摂也監修,' トポロジーと幾何学入門', 倍風館, 1976)

[数学 4] 日本数学会編集, ' 岩波数学辞典第３版,' 岩波書店, 1987.

[数学 5] 野村泰敏, ' 位相幾何学入門,' サイエンス社, 1977.

[数学 6] 斎藤正彦, ' 線型代数入門,' 東京大学出版会, pp107–112, 1966.

[数学 7] I.M.Singer and J.A.Thorpe, ' トポロジーと幾何学入門,' 倍風館, pp157–165, 1976.

[Nishida91] 西田淳, 張紹星, 西原清一, ' 面の組み合わせ探索による三面図の解釈,' 人工知能学会誌, 6(1):96-103, Jan.1991.

[Okino73] N.Okino, Y.Kakazu, and H.Kubo, 'TIPS-1 : Technical Information Processing System for Computer Aided Design and Manufacturing,' *Computer Language for Numerical Control*, North Holland, pp.141–150, 1973.

[Ostrowski87] M.C.Ostrowski, 'Feature-based Design using Constructive Solid Geometry,' *SIGGRAPH Course Notes: Solid Modeling - Archtectures, Mathematics and Algorithms,* Aug. 1990.

[Paoluzzi93] A. Paoluzzi,et.al., 'Dimension-Independent Modeling with Simplical Complexes,' *ACM Transaction of Graphics*, 12(1):57-102, Jan. 1993.

[Pegal88] Pegal, 'Partitioning Polyhedral Objects into Nonintersecting Parts,' *IEEE Computer Graphics and Applications*, 8(1):53-67, Jan. 1988.

[Pratt88] M.J.Pratt, 'Synthesis of an Optimal Approach to Form Feature Modeling,' *ASME Computers in Engineering Conference,* Aug. 1988.

[Pratt89] M.J.Pratt, 'A Hybrid Feature-Based Modeling System,' *Advanced Geometric Modeling for Engineering Applications, North-Holland,* pp.189–202, Nov. 1989.

[Rainer94] K.Rainer and S.Kou, 'Applying Non-Manifold Geometric Modeling Methods in Integrated Design and FE analysis Environment,' *American Society of Mechanical Engineering, Petroleum Division,* 59::pp.85–89, 1993.

[Requicha80] A.G.Requicha, 'Representations for Rigid Solids: Theory, Methods, and Systems,' *ACM Computing Surveys,* 12(4):437-464, Dec. 1980.

[Requicha83] A.A.G.Requicha and H.B.Voelcker, 'Solid Modeling: Current Status and Research Directions,' *IEEE Computer Graphics and Applications,* 3(5):25–37, Oct. 1983.

[Requicha85] A.A.G.Requicha and H.B.Voelcker, 'Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algothithms,' *Proc. of IEEE,* 73(1):30-44, Jan. 1985.

[Requicha86] A.A.G.Requicha et al, 'Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry,' *IEEE Jornal of Robotics and Automation,* RA-2(3), Sep. 1986.

[Rossignac90] J.R.Rossignac and M.A.O'Connor, 'SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries,' *Geometric Modeling for Product Engineering, North-Holland,*' pp.145-180, 1990.

[Rossignac91] J.R.Rossignac and A.A.G.Requicha, 'Constructive Non-Regularized Geometry,' *Computer-Aided Design,* 23(1), Feb. 1991.

[Sakurai83] H.Sakurai and D.C.Gossard, 'Solid Model Input through Orthographic Views,' *Proc. of SIGGRAPH'83,* 17(3):243–247.

[Sakurai90] H. Sakurai, 'Automatic Setup Planning and Fexture Design for Machining,' PhD thesis, MIT, Jan. 1990.

[Satoh90] 佐藤敏明, 高村禎二, 鳥谷浩志, 千代倉弘明, ' 他種類の曲面を持つ立体に対する集合演算,' 情報処理学会：グラフィックスとＣＡＤシンポジウム論文集, pp.111-120, Oct. 1990.

[Satoh91] T.Satoh and H.Chiyokura, 'Boolean Operations on Sets Using Surface Data,' *Proc. of Symposium on Solid Modeling Foundations and CAD/CAM Applications,* pp.119–128, Jun. 1991.

[Shah88] J.J.Shah et al. 'Current Status of Feature Technology,' *CAM-I Report C-88-GM-01,* Nov. 1988

[Shah91] J.J.Shah, 'Assessment of Features Technology,' *Computer Aided Design,Vol.23, No.5,* pp.331–343, Jun.1991.

[Shimada87] K.Shimada, M.Numao, H.Masuda, and S.Kawabe, 'Constraint-Based Object Description for Product Modeling,' *Computer Applications in Production and Engineering, North-Holland,* pp.95–106, Oct. 1989.

[Shimizu91] S.Shimizu, K.Inoue and M.Numao, 'An ATM-Based Geometric Constraint Solver for 3D CAD,' *Proc. of Tools For Artificial Intelligence,* pp.282–290, Nov. 1991.

[Spur75] G.Spur and J.Gausemeier, 'Processing of Workpiece Information for Production Engineering Drawing,' *Proc. of 16th Int. Machine Tool Design and Research Conf.,* pp.17-21, 1975.

[Spur82] G.Spur, F.-L.Krause and J.J.Harder, 'The Compac Solid Modeller,' *Computers in Mechanical Engineering,* 1(2):58-72, 1984.

[Takahashi91] 高橋正充, 佐々木康仁, 伊藤潔, ' 非線形疑似ブール計画法による曖昧な三面図からの多面体の一意的合成,' 人工知能学会誌, 6(6):904-911, Nov.1991.

[Tilove80] R.B.Tilove, 'Set Membership Classification: A Unified Approach to Geometric Intersection Problems,' *IEEE Transactions on Computers*, C-29(10):874-883, Oct.1980.

[Toriya86] H.Toriya, T.Satoh, K.Umeda, and H.Chiyokura, 'Undo and Redo Operations for Solid Modeling,' *IEEE Computer Graphics and Applications,* 6(4):35-42, Aug. 1986.

[Turner87] J.U.Turner, 'Tolerances in Computer-Aided Geometric Design,' PhD thesis, Rensselaer Polytechnic Institute, May 1987.

[Ferrucci91] V.Ferrucci and A.Paoluzzi, 'Extrusion and Boundary Evaluation for Multidimensional Polyhedra,' *Computer-Aided Design,* 23(1):40–50, Jan. 1991.

[Voelcker77] H.B.Voelcker and A.A.G.Requicha, 'Geometric Modelling of Mechanical Parts and Processes,' *IEEE Computer,* 10(12):48–57, 1977.

[Voelcker93] H.B.Voelcker and A.A.G.Requicha, 'Research in Solid Modeling at the University of Rochester: 1972-87,' *in Fundamental Developments of Computer-Aided Geometric Modeling, Academic Press,* pp.201-254, 1993.

[Weiler85] 'Edge Based Data Structures for Solid Modeling in Curved-Surface Environments,' *IEEE Computer Graphics and Applications,* 5(1):21–40, Jan. 1985.

[Weiler86] K.Weiler, 'Topological Structures for Geometric Modeling,' *PhD. Thesis, Rensselaer Polytechnic Institute,* Aug. 1986.

[Weiler86b] K.Weiler, 'The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Modeling,' *Geometric Modeling for CAD Applications,* North Holland, pp.3–36, May 1986.

[Weiler86c] K.Weiler, 'Boundary Graph Operators for Non-Manifold Geometric Modeling Topology Representations,' *Geometric Modeling for CAD Applications, North Holland,* pp.37–66, May 1986.

[Wesley80] M.A.Wesley.,et al., 'A Geometric Modeling System for Automated Mechanical Assembly,' *IBM Journal of Research and Development,* 24(1):23-30, Jan. 1980.

[Wesley81] M.A.Wesley and G.Markowsky, 'Fleshing Out Projections,' *IBM Journal of Research and Development,* 25(6):938-954, Nov. 1981.

[Wilson85] P.R.Wilson, 'Euler Formulas and Geometric Modeling,' IEEE Computer Graphics and Applications, 5(4):24-36, Aug. 1985.

[Wilson86] P.R.Wilson, 'Multiple Representations of Solid Models,' *Geometric Modeling for CAD Applications*, North-Holland, pp.99–113, May 1986.

[Wilson90] P.R.Wilson, 'Feature Modeling Overview,' *in SIGGRAPH'90 Cource Note,* 1990.

[Wilson88] P.R.Wilson and M.Pratt, 'A Taxonomy of Features for Solid Modeling,' *Geometric Modeling for CAD Applications, North-Holland,* pp125–136, 1988.

[Woo85] T.C.Woo, 'A Combinatorial Analysis of Boundary Data Structure Schemata,' *IEEE Computer Graphics and Applications,* 5(4):19–27, Aug. 1985.

[Wu89] S.T.Wu, 'Towards A Unified Data Scheme for Geometric Representations,' *Computer Applications in Production and Engineering, North-Holland,* pp.259–266, Oct. 1989

[Yamaguchi90b] 山口泰, 木村文彦, ' 非多様体位相の計算機内表現と操作,' グラフィックスとＣＡＤシンポジウム, pp.83–89, Nov. 1990.

[Yamaguchi90] 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, 32(6):731–739, Jun. 1991.

[Yamaguchi91] 山口泰, 木村文彦, ' 非多様体形状モデリングと例外処理,' グラフィックスと CAD シンポジウム論文集, Nov. 1991.