グラフィクスとCAD 50－1
（1991．5．24）

非多様体モデルの集合演算アルゴリズム

増田 宏

日本アイ・ビー・エム（株） 東京基礎研究所

非多様体モデルは、ワイヤフレーム、サーフェス、ソリッドが混在した形状を表現することができる．ここでは、非多様体モデル間の和、差、積の集合演算について述べる．まず、集合演算の定義とその位相表現の方法について解説する．非多様体モデルの位相表現には、結果形状の境界のみを保持するやり方と集合演算で用いられた基本立体の位相をすべて保持したハイブリッド表現があり、特に後者はモデリング環境を向上させるために有効である．これら両方の集合演算は現在の試作システム上で実現されており、十分な計算速度で動くことを確認している．これらの集合演算を実現するための処理手順について説明する．

Definition and Implementation of Boolean Operations for Non-Manifold Geometric Modeling

Hiroshi Masuda

IBM Research, Tokyo Research Laboratory

5-19, Sanbancho, Chiyoda-ku, Tokyo 102, Japan

Boolean operations for non-manifold geometric models are discussed. Non-manifold models are used to represent wireframe, surface, and solid shapes in a single architecture. First, Boolean operations are defined mathematically, and then the topological representations are discussed. It is possible to realize two kinds of topological representations for the Boolean set: one maintains only the boundary of the resultant shape, and the other maintains all primitive shapes that are used for Boolean operations as well as the resultant shape. Both representations were implemented, and the algorithms for them were proven to work adequately.

1．はじめに

設計や生産の場に計算機が利用されるようになって以来、３次元形状モデリング技術は広く利用されてきた．特にソリッドモデルは、さまざまなシミュレーションや加工データの生成など幅広く使えるため、その重要性を増してきている．ソリッドモデルは、一般に処理が重いという難点があったが、最近では強力なワークステーションが安価で手にはいるようになり、以前よりも身近なものになりつつある．今後ソリッドモデルが使われる場はどんどん増えていくであろう．

しかしながら、依然としてソリッドモデルは、サーフェスモデルに比べて扱いにくいという問題がある．実際、ソリッドモデルで複雑な形状を作成することは熟練と根気と時間を要する仕事である．この原因には、現在のソリッドモデルに制約が多く、モデリング環境がよくないことが挙げられる．近年では、形状特徴や制約設計などを用いて、モデリング環境を向上しようとする動きが盛んであり一定の成功を収めているが、モデリング環境を向上させるにはただ単に既存のモデラに賢いインタフェースを付けるだけでは不十分である．形状モデラが本質的にもつ不自由さを取り除き、製品設計に適した形状モデラを開発していくことが必要である．ソリッドモデルの制約の一つに表現力の融通のなさが挙げられるが、位相構造の面から形状モデルの自由度を上げていこうとする試みが非多様体モデリングの研究である．

非多様体形状モデルは、従来のソリッドモデルよりも柔軟な表現力をもった形状モデラである．このモデラの利点は幾つか指摘されているが、まず、第一にワイヤフレーム、サーフェス、ソリッドが単一のデータ構造で表現できることが挙げられる．この利点は、すべての機能がそろった汎用のブラックボックス的なＣＡＤシステムを使っているうちはあまり問題にはならなかった．しかし、ＣＡＤの応用範囲が拡大し、ユーザ自身が特定の業務に適したシステムを構築したり他のシステムと接続したりするようになってくると、これまでのような複数のデータ構造の存在は、データ管理、相互のデータ変換、データベースへの対応など問題を複雑にする可能性がある．この点で、非多様体モデラは、ユーザ構築のＣＡＤシステムの中核システムとして有望である．また、アプリケーションによっては、細部まで完全なソリッドモデルを作る必要のない場合も多い．実際、サーフェスモデルから徐々にソリッドを構成していく場合は多いが、非多様体モデルではソリッドとサーフェスの混在を自然に表現することができるのでより柔軟なモデリングが可能である．また、一方、筆者らが提案してきたように[2,3,4]、非多様体モデルの高い表現力を利用して、境界表現とＣＳＧ表現を混在させたハイブリッド表現も可能である．この表現では、両者の表現が密接に結び付いており、ＣＳＧ Tree を適当に編集した境界表現を短時間で得ることができる．この新しい表現では従来時間のかかる処理であった境界表現の修正が簡単に行うことができるので試行錯誤的な形状モデリングに向いている．

非多様体モデルは、豊かな表現力を持った新しいモデラとして有望であるが、その一方、形状処理として扱うべき範囲もまた増えてくる．形状処理として最も一般的なのは集合演算であるが、非多様体モデルでは、ワイヤフレーム、サーフェス、ソリッドが混在した状況も扱う必要がある．筆者は、非多様体モデル一般に適用可能な集合演算を開発したのでそれについて報告する．本稿では、まず、非多様体モデルの概略について述べたあと、集合演算の定義と位相表現の考え方を述べる．次に処理手順について述べる．

2．非多様体モデルの境界表現

非多様体モデルは、ソリッドモデルよりも広範な範囲の形状を統一的なデータ構造で表現することができる幾何モデルである．具体的には、図１に示すように、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、それらの混合などが含まれる．このような形状の集合は多様体でない部分も含み得るので、慣用的に「非多様体モデル」と呼ばれている．ソリッドモデルには、境界面が閉じた２－多様体でなければならないという制約がある．簡単にいうと、すべてのエッジ(稜線)はちょうど２つの面に共有されなければならない．この制約では、面の境界にならないワイヤエッジや、３つ以上の面に共有される非多様体エッジを含むような形状は、正しい形状とは見做されない．また、一つの頂点の近傍の面は、その頂点を通ることなし

に一周できなければならない．従って、２つの面が点接触するような形状は扱えない．非多様体形状モデリングは、ソリッドモデルで表現できなかったこれらの位相構造を正しく扱えるようにすることによって、形状モデリングの自由度を高めていこうとするものである．

非多様体モデルを境界表現で表わす方法について、階層構造と空間表現の２つの観点から説明する．まず、階層構造から述べる．境界表現は、ＣＳＧ表現と共に、ソリッドモデルの代表的な表現方法であるが、境界表現の基本的な考え方は、形状を面や線、点などに分解してそれらの接続関係によって３次元形状を表現しようとするものである．直感的に考えて、３次元形状は、vertex(点)、edge(稜線)、face(面)、volume(立体) に分解できることがわかると思う．volume は、ソリッドモデルには陽に扱われない概念なので、簡単に説明しておく．例として、さいころ (六面体) をイメージしてみて欲しい．もし、さいころを紙でつくれば中空の六面体になるし、木で作れば中の詰まった六面体になるであろう．これらは、境界面は全く同じ６枚の面であるが、面で囲まれた閉領域に実体があるかないかの違いがある．この３次元の実体 (境界は含まない) を volume と呼ぶ (region と呼ばれることもある)．さて、境界表現においては、３次元形状は、これらの形状要素、volume, face, edge, vertex は、下位の形状要素が上位の形状要素の境界となるように、階層的に組み立てられる(図2)．volume、face に空洞を許す場合は、shell と loop をさらに形状要素として追加することが必要になる．空洞が存在すると境界が非連結となるため、連結している境界要素の集合を表現するために shell と loop が用いられる．非多様体モデルの階層構造では、各階層の形状要素のタイプが必ずしも同じでないので注意が必要である．たとえばソリッドモデルでは edge は常に face の境界であるが、非多様体モデルでは、ワイヤエッジであったり、volume の境界(volume 中のワイヤエッジ)となったりする．従って、形状の自由度が高まる反面、形状処理のアルゴリズムは複雑になる傾向がある．edge がどのような使われ方をしているかを知るには、上位の形状要素が、model なのか volume なのか face なのかを知る必要があり、それに応じた場合分けをして処理することも必要である．

次に、空間の表現方法について述べておく．ソリッドモデルでは、閉じた空間はただ一つしか存在しなかったが、非多様体モデルでは、面で囲まれた閉空間、すなわち volume は何個存在してもよい．幾つか小部屋の集まりから構成されるセル構造も表現可能である．非多様体モデルの形状処理を考えるとき、これらの volume を適切に扱うことは大変重要な問題である．volume 管理の機構として重要なのが 図3 に示す radial edge pointer である．この図において、face の両側の点線は、面の裏表に対応する．radial edge pointer は必ずしも不可欠ではないが、効率的な処理には非常に効果的である．radial edge pointer は、図3 に示すように、edge 回りの面を時計回りに順次辿れるように張られている．radial edge pointer を介して連結なすべての面を辿ると、それが閉空間を構成する面の集合となる．特に集合演算を考えると、volume の分割や再構成が頻繁に起こるため、volume が容易に管理できる構造がないと効率的な処理は困難である．

ところで、radial edge pointer は隣り合う面の関係を edge を介して結び付けているが、同等の関係付けを vertex を介して張ることも可能である．そのような構造を vertex-based 構造と呼ぶ[5]．vertex-based 構造では、vertex 回りの関係を保持しておき、例えば図4 のような２つの形状を、位相構造上区別する．edge-based 構造では、これらは区別しないことを考えると、vertex-based 構造の方が多くの情報を保持しているといえる．ただし、実用上のことを考えると、情報が多ければよいというものでもない．問題は、保持している情報が頻繁に利用されるのかということにある．図4 の２つの形状を区別するためには、データ構造に反映させるために、少なくとも一度は必要な計算をしなければならない．従って、もしその情報があまり使われないのであれば、計算コストが回収できず収支は赤字になるであろう．基本的に、verte

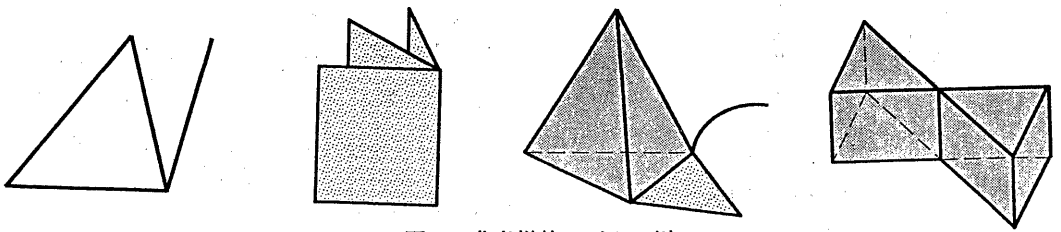


図１．非多様体モデルの例

x-based 構造でできることは、edge-based 構造でもできるので、どちらがよいかの議論は効率やデータ量の観点からなされていくべきである．なお、筆者の試作しているモデラは edge-base である．

３．集合演算の定義と位相構造

集合演算には、和集合、差集合、積集合の３通りの演算がある．非多様体モデルは、ソリッドモデルよりも定義域が広いため、集合演算もより柔軟に考えることができる．ここでは、非多様体モデルの集合演算をどう定義するかを考える．まず、集合論的な観点から述べ、次に位相的な観点から述べる．

３．１　集合演算の定義

非多様体モデルの集合演算は空間的にみてどのような演算なのかを考えてみたい．一般的な集合論においては、和集合∪、差集合－、積集合∩ の定義は明確である．形状を３次元空間の点の集合と考え、その間での演算を考えるのである．たとえば、数直線上の２つの形状Ａ，Ｂを考え、その範囲がそれぞれ、Ａ＝[－１，０]，Ｂ＝[０，１] とすると、和集合 Ａ∪Ｂ＝[－１，１] である．しかしながら、この定義をそのまま非多様体モデルの集合演算に持ち込むと、不都合が起こる．なぜなら、非多様体モデルは和集合∪と積集合∩には閉じているが、差集合－に関しては閉じていないからである．この例では、Ａ－Ｂ＝[－１，０）であり、端点を持たない形状を作り出してしまう．非多様体モデルは閉じた形状であるという制約があるので、この演算結果は正しい形状とは見做されない．このような問題はソリッドモデルでも同様に生じるので、非多様体モデリングにおいてもソリッドモデリングと同様の解決方法をとる．すなわち、差集合を閉じた演算とするために、非多様体モデルＡ，Ｂの差集合を（Ａ－Ｂ）の閉包と定義するのである．閉包をとるとは、要するに、境界のない部分に境界を付け加えることである．この差演算を －* と書くことにする．

一方、実用的な観点からみると、積集合 ∩ にも問題がある．境界を接するような２つの面の積集合を考えると、演算結果は２次元の面が縮退して１次元の線分となる．非多様体モデルではこのような縮退した場合でも矛盾なく扱うことができるのであるが、実用的にみて縮退した形状は残すべきか、という問題がある．通常の形状モデリングの場合には、ソリッドモデリングがそうであるように、縮退した部分は多くの場合不要であり、消去した方が都合がよい．一方、ソリッド間の干渉チェックの場合には、縮退した面・線・点はそれぞれ面接触、線接触、点接触を表わすので、残した方がよいかもしれない．どちらの定義でも可能

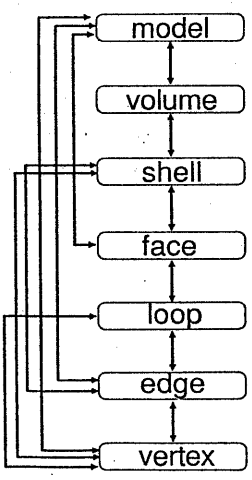


図２．階層構造

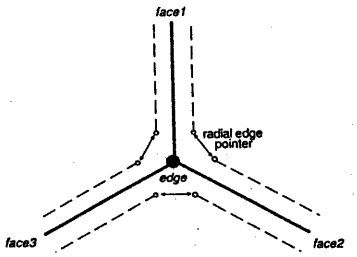


図３．Radial Edge Pointers

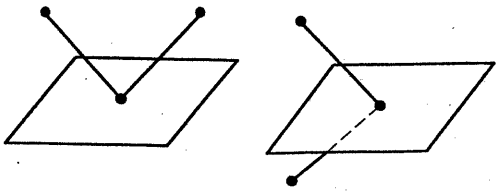


図４．vertex-based 構造で区別する位相

なので、結局、積集合には、２つのバージョンがあると考えるべきであろう．どちらを使うかは、アプリケーションが決めるべきことである．縮退した部分を消去する積集合を∩*、残す演算を∩ と書くことにする．

以上で、空間的な意味での非多様体モデルの集合演算が定義できた．以後、集合演算という用語は、この定義で用いることにする．

３.２　和集合の位相構造

３.２.１ 非多様体モデルによる位相構造

集合演算を位相構造の観点から考えてみる．形状Ａ，Ｂの和集合とは、それぞれの形状の占める空間の和のことであり、この点にあいまいさはない．ただし、この空間の和にどのような位相構造を与えるかには複数の選択肢がある．図5 に示すような和集合を考えると、(a),(b) のどちらの位相構造も可能であるし、位相的に考えて合理性がある．表現(a) は、Ａ，Ｂの占める空間の和を単一の face とその境界で表現したものであり、表現(b) は、和集合を Ａ，Ｂの位相構造の和として表現したものである．ここでは、表現(a) を Ｒ（ ＡＵＢ ）、表現(b) を R^{Δ}（ ＡＵＢ ）と書くことにする．ソリッドモデリングにおいては、このような選択肢は存在しなかった．なぜならば、ソリッドモデルで 表現R^{Δ} のような位相構造を持たせると非多様体エッジを生じてしまい、ソリッドモデルの定義域にはおさまらなくなるからである．表現R^{Δ} はもともとの形状Ａ，Ｂの位相構造を残しているので、表現Ｒ よりも情報量が多く、後述するように、このプラスアルファの部分がいくつかのアプリケーションでは非常に重要となる．一方、表現Ｒの情報で十分なアプリケーションでは、不必要なデータを保持していない点で 表現Ｒが適しているであろう．表現R^{Δ} は非多様体モデル特有の表現であるので、特にこの表現について述べる．

３.２.２ ハイブリッド表現

集合演算に使われる形状Ａ，Ｂを基本立体と呼ぶことにする．表現R^{Δ} では基本立体の位相構造が演算結果に残っている．すなわち、表現R^{Δ} のすべての形状要素は、基本立体の一部であったか、基本立体間の干渉によって生じたかのいずれかである．また、もともとの基本立体のすべての形状要素は、表現R^{Δ} の形状要素の少なくとも一つに対応する．従ってもし、これらの関係を形状要素の属性として付加してやれば、も

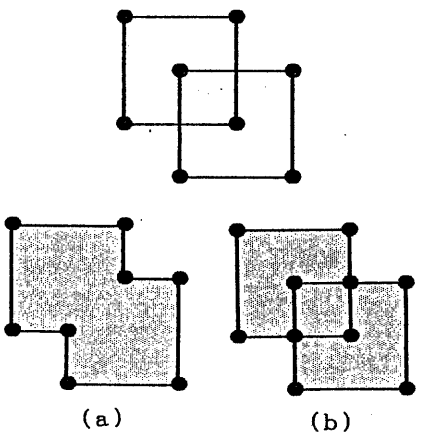


図５．和集合の位相表現

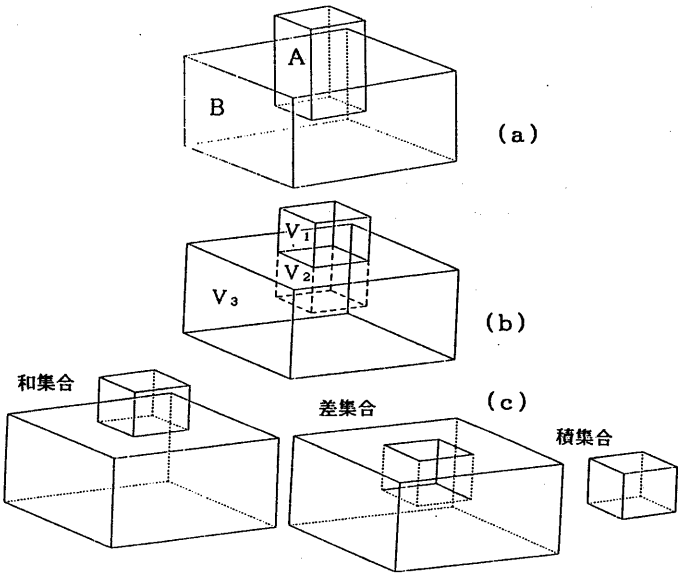


図６．ハイブリッド表現

ともとの基本立体の形状を 表現R^{Δ} のなかに保持できることになる[2,3,4]．この表現では製品形状の境界表現と基本立体の境界表現を同時に表現出来るので、この表現をハイブリッド表現と呼ぶことにする．ハイブリッド表現は、「形状特徴モデリング」のように、個々の基本立体が工学的に意味を持つような場合に大変に有効なものとなる．なぜなら、ハイブリッド表現中に保持された基本立体にパラメータや加工情報などのさまざまな属性を与えて管理することができるからである．また、演算結果の形状要素から、特定の基本立体に由来する部分を自由に取り出して表示したり、消去や変更を行うことも可能なので、形状モデリングの柔軟性を高める上でも重要である．

さて、ここで、ハイブリッド表現を利用した集合演算操作について述べる．集合演算には、和、差、積の３種類があるわけだが、R^{Δ}（A∪B）から、和集合 A∪B、差集合 A−*B、積集合 A∩*B が適宜抽出できるので、表現R^{Δ} さえ用意しておけば一般の集合演算に対応できる．図6 にその考え方を示す．図6(b) は基本立体A，Bの和集合表現 R^{Δ}（A∪B）である．この形状は、３つの volume、$\{V_1, V_2, V_3\}$ からできているが、集合演算の種類に応じて適当な volume を抽出すれば、和、差、積を得ることができる．ハイブリッド表現のそれぞれの形状要素には、どの基本立体から作られたかものなのかが記述されているので、この場合、V_1 と V_2 が A から生成され、V_2 と V_3 が B から生成されたということが得られる．したがって、$A=[\{V_1,V_2\}]$，$B=[\{V_2,V_3\}]$ である．これらを代入することによって、$A\cup B=[\{V_1,V_2,V_3\}]$，$A-^*B=[V_1]$，$A\cap^*B=[V_2]$ と計算できるので、抽出すべき形状要素を知ることができる．$[\{V_i\}]$ は、$\{V_i\}$ および、$\{V_i\}$ のいずれか一つのみに接する face とその境界のことである．なお、縮退した部分を残す積集合∩の場合には、V_1 と V_3 に同時に隣接する形状要素を付け加えればよい．このようにして取り出された形状が、図6(c) である．

ここでは、簡単のためソリッド形状の場合のみ述べたが、「volume」の代わりに「自分自身より次元の高い形状要素の境界にならない形状要素」といいかえればワイヤフレームやサーフェスモデルを含んだ一般の非多様体モデルでも適用できる．一般に、このような集合演算を実現するために必要なデータは、(1) 基本立体 $\{P_i\}$ の和集合表現R^{Δ} $(\cup P_i)$、(2) 表現R^{Δ} の形状要素と基本立体との対応関係、(3) ＣＳＧ表現の３つである．ＣＳＧ表現は、基本立体間の論理演算を保持しているので、ＣＳＧ表現の基本立体の部分に(2)で得られる適当な形状要素の集合を代入してやれば、必要な形状要素を(1)の表現から抽出できる．

４．集合演算のアルゴリズム

これまで、非多様体モデルの集合演算として、和集合∪、差集合−*、積集合∩*、∩ を定義し、保持すべきデータ構造として RとR$^{\Delta}$ があることを述べた．ここでは、これらのすべてに対応した集合演算操作を考える．

４．１ ハイブリッド表現に基づく集合演算

まず、３．２で述べたハイブリッド表現に基づく集合演算について述べる．この場合重要なのは、表現R$^{\Delta}$ の構成方法と、基本立体との関係付けの管理である．

４．１．１ 位相構造の生成

非多様体モデルは、前述したように、volume，face，edge，vertex を構成要素に持つものとし、また volume の境界でない face を surface-face、face の境界でない edge を wire-edge と呼ぶことにする．

２つの非多様体モデルの和集合を求めるプロセスは大きく分けて、(1)ラフチェック、(2)干渉線／干渉点の生成、(3) 一致する face，edge，vertex の併合、(4) volume の再構成 に分けられる．処理の流れは次の通りである．

(1) ラフチェック

A，B から、volume，surface-face，wire-edge(volume の境界でないもの) を取り出す．それらを $\{a_i\}$，$\{b_j\}$ とする．すべての (a_i, b_j) の組み合わせについて干渉を調べるわけであるが、まず、個々の形

状要素を包含する直方体同志の干渉の有無を調べるラフチェックを行う．ラフチェックではじけない場合、もし ai，bi のいずれかが volume であれば、volume を境界である face や wire-edge に分解してさらに干渉チェックを行う．最終的には、干渉チェックではじけない (face, face), (face, wire-edge), (wire-edge, wire-edge) の組みが残る．

(2) 干渉線／干渉点の生成

ラフチェックで残った組みに関して、干渉線／干渉点を求める．干渉線がどの形状要素の上に生じたかも記録する．すべての干渉線／干渉点が求まった後、その干渉線／干渉点を２つの非多様体モデル上にそれぞれ生成する．

(3) 一致する形状要素の併合

互いに一致する形状要素を、vertex，edge，face の順に併合していき、２つの非多様体モデルを合体させる．edge を併合する際、もし edge が３つ以上の face から共有されていれば radial edge pointer を計算し直すことが必要である．edge 回りの face の方程式を調べて、face が edge の回りで時計回りになるように radial edge pointer を張り直す．

(4) ｖｏｌｕｍｅ の再構成

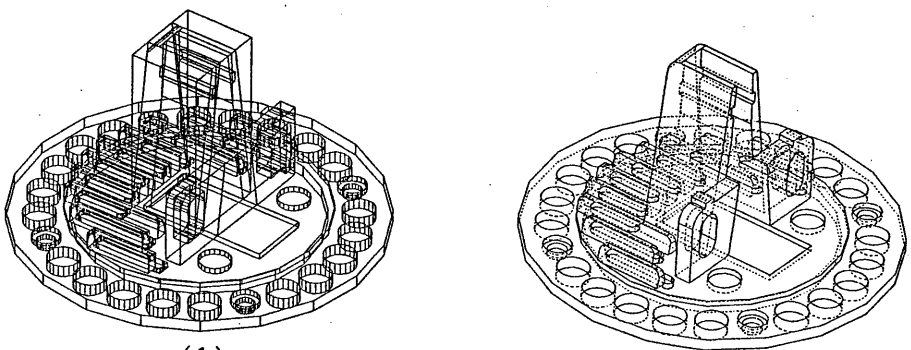
２つの非多様体モデルを合体させることによって、volume が分割されていることがあるので、radial edge pointer を辿ることによって volume を求め直し、必要に応じて volume を分割する．また、この段階で wire-edge が volume の境界になるかどうかチェックして、階層構造を調整する．

図7(1) にこの表現で構成されたプリンタヘッド（基本立体１４９個）の例を示す．計算時間は、RS/6000-730 (IBM) で ８７.５秒であった．

４.１.２ 基本立体との対応関係の管理

ハイブリッド表現において、すべての形状要素はどの基本立体から構成されたかが管理されなければならない．たとえば、あるエッジ e_1 がもともと基本立体 P_1 の一部であったとすれば $e_1 \rightarrow \{P_1\}$ という関係が保持されなければならない．上記の処理手順においては、この関係の管理は比較的簡単である．このプロセスではすべての形状要素は、(1) 干渉によって初めて生じるか、(2) 分割されるか、(3) 併合されるかのいずれかである．もし、e_1 が分割されて e_1 と e_2 になったとすれば、e_2 に関係をコピーして $e_2 \rightarrow \{P_1\}$ とすればよいし、$e_1 \rightarrow \{P_1\}$, $e_2 \rightarrow \{P_2\}$ である２つの edge を併合して e_{12} を生成したとすれば $e_{12} \rightarrow \{P_1, P_2\}$ とすればよい．また、基本立体 P_1 の一部であった face 上に干渉線を e_1 を生成した場合には、P_1 上に新たに生成されたという意味で $e_1 \rightarrow \{(\text{on } P_1)\}$ という属性を付ければよい．干渉で生じた場合を特別に扱う理由は、このエッジが他の基本立体との関係で初めて存在し得るということを記述するためである．

この関係付けとＣＳＧ表現を用いて抽出された形状を図7(2)に示す．抽出に要した時間は、０.１６秒であった．



(1)　　(2)

図７．プリンタヘッドのモデル

４．２　従来型の集合演算の手順

ハイブリッド表現が得られていれば、3.2.2 で述べたように、和集合、差集合、積集合が得られるということを述べた．これは、4.1.1 で述べた位相構造から、適当な形状要素を選び出すことで実現できる．従って、選ばれなかった形状要素を、表現R^ から削除すれば、従来ソリッドモデリングで用いられてきたような位相表現R が得られる．この手順は次の通りである．

(1) (ハイブリッド表現の生成) 個々の非多様体モデルを仮想的に単一の基本立体から構成される形状と考え、形状要素に基本立体との関係を属性として加える．その上で、ハイブリッド表現を作成する．
(2) (形状要素の抽出) 集合演算の種類 (∪、−*, ∩*, ∩) に応じて、適当な形状要素を抽出する．抽出された形状要素にはマークを付けておく．
(3) (不要な形状要素の削除) マークの付いていない形状要素を削除する．削除は、volume, face, edge, vertex の順で行う．

この手順を従来のソリッドモデルの集合演算[6,7]と比較してみると、不要な形状要素の削除に際して、包含関係を調べる手順が省かれている．従来は、face が他方の形状の内部にあるか調べて、削除すべきかどうかを判断していた．本手法では、これと同等な計算は、radial edge pointer を張り直すという処理に含まれている．また、本手法は、集合演算の種類が異なっても、処理手順がほとんど変わらない点も特徴的である．

５．まとめ

非多様体モデルの集合演算の考え方について述べ、その実現方法を述べた．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える他、基本立体の位相構造と製品形状の境界表現を同時に表現したハイブリッド表現も可能である．

[参考文献]
[1] Weiler. "Topological Structure for Geometric Modeling," PhD.Thesys, Rensselaer Polytechnic Institute, Aug.1985
[2] 増田,鳴田,川辺. "非多様体モデルのためのオイラー・オペレーション," ｸﾞﾗﾌｨｯｸｽとCAD ｼﾝﾎﾟｼﾞｳﾑ, Oct.1988
[3] 川辺,鳴田,増田. "A Framework for 3D Modeling: Constraint-Based Description and Non-Manifold Geometric Modeling," Toyota Conference, Oct.1988
[4] 増田,鳴田,川辺. "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," ｸﾞﾗﾌｨｯｸｽとCAD ｼﾝﾎﾟｼﾞｳﾑ, Oct.1988
[5] Gorsoz,Prinz,Choi. "Vertex-Based Representation of Non-Manifold Boundaries," Geometric Modeling for Product Engineering, North Holland,1990
[6] 千代倉. "ソリッドモデリング," 工業調査会, 1985
[7] Mantyla. "Solid Modeling," Computer Science Press, 1988