公差解析のための知識表現言語とそのプログラミング手法

望月雅光[†] 長澤 勲[††] 梅田政信[††]
樋口達治[†††] 小島崇司[††††,*]

近年，設計業務の効率化のために，設計現場に設計検証システムの1つとして公差解析システムが開発され導入されるようになってきた．しかしながらこれらのシステムは，一般によく用いられる公差解析手法だけを組み入れたものである．このため製品ごとに存在する様々な知識を反映できず，公差解析の支援を不十分なものとしている．このような問題を解決するために，本論文では，製品ごとに存在する知識に基づいて，設計者が整理したばらつき発生のモデルや公差解析手法を，容易に表現できる設計者用の知識表現言語を提案した．次に，公差解析システム構築のために必要なプログラミング手法を整理することにより，組立構造，寸法，公差の各表現手法を示し，これらを用いてばらつき発現モデルおよび公差解析手法を表現できる見通しを得た．最後に，本手法の効果を確認するために，実際の設計問題を試作システムを用いて記述し，解析実験を行った．その結果，本手法を用いた公差解析システムは，保守性，拡張性が高いことを示すことでき，設計者による保守や拡張が可能な公差解析システムの実現性を確認できた．

A Knowledge Representation Language for Tolerance Analyses and Its Programming Techniques

Masamitsu Mochizuki,[†] Isao Nagasawa,[††] Masanobu Umeda,[††]
Tatsuji Higuchi[†††] and Takashi Ojima[††††,*]

Recently, tolerance analysis systems have been developed for estimating assemblability, verifying part drawing and so on to improve efficiency of design work. However, since they have only composed of existing tolerance analysis methods and the knowledge is not made explicit, it is difficult for designers to modify or extend these systems, by using to design knowledge about their products. In this paper, we first propose a knowledge representation language for tolerance analysis and variational model. We second propose programming techniques for representing assembly structure, dimension and tolerance. Better tolerance analysis systems can be constructed using this language and these programming techniques. And designers who do not have knowledge about computers can easily put their knowledge into this tolerance analysis system. Finally, we report prototype system that has been developed and applied to real assembly parts. The result was encouraging. We show through real application that this language is easy to write, modify, and extend knowledge by designer. We hope that our approach will have an impact on design verification systems.

† 九州工業大学大学院情報工学研究科博士後期課程
Graduate School of Computer Science and System Engineering (Doctoral's Programs), Kyushu Institute of Technology
†† 九州工業大学情報工学部
Faculty of Computer Science and System Engineering, Kyushu Institute of Technology
††† オリンパス光学工業株式会社第一開発部
Product Development Department, OLYMPUS Optical Co. Ltd.
†††† 九州工業大学大学院情報工学研究科修士課程
Graduate School of Computer Science and System Engineering (Master's Programs), Kyushu Institute of Technology
* 現在，同博士後期課程
Presently with Doctoral's Prgrams

1. はじめに

民生機器等に使用する機械部品の寸法や形状は，部品図[1]に従い，金型や工作機械を用いて製作するために必ず誤差を含む．このため設計では寸法や形状に許容限界を設け，許容値間に実物の寸法や形状が入れば良いという方式を採る．例えば，寸法の許容限界を寸法公差，形状の許容限界を幾何公差と呼び，それぞれ寸法と形状のばらつきを規定する[2]．これらの公差は，部品の詳細設計に際して，部品に必要な精度，部品を製造する工場の工程能力，そして規格を考慮して設定する[3),4]．適切な公差の設定は重要であり，公差の設

定がきびしいと必要以上の加工精度が必要になり，逆に，設定があまいと部品同士の干渉等の不具合が生じる．このため，設計者は寸法や形状のばらつきを評価する解析手法（以下，公差解析手法）を用いて設計検証を行う．

現状の設計現場では，設計者自身が公差解析を手計算で行うことが多く，多数の箇所を検証するためミスを生じやすい．また角度寸法を含むような複雑な解析は，手計算では困難な場合もある．このため公差解析システムが開発され市販されるようになってきた[5),6)]．しかしながら，公差の設定や解析には，過去の製品開発によって蓄積された知識を必要とする．例えばカメラのレンズ系の設計では，その性能がレンズやレンズ枠のばらつきにどのように依存するか（以下，ばらつき発現モデル），あるいは必要な計算精度を得るためには，どのような公差解析手法を用いればよいか，を知る必要がある[7),8)]．ところが，一般に既存のシステムは，このような知識をシステム内に組み入れる機能が整備されておらず，製品開発に必要な公差解析支援を十分に行っているとは言えない．このような問題の解決には，製品ごとに設計者が考案するばらつき発現モデルや公差解析手法を含めて，公差解析に関わる全ての知識を統一的に取り扱う枠組が必要である．

従来の研究では，公差解析の対象である部品や組立品の構造（以下，組立構造）をグラフや数式で表現する方法[9),10)]，機能を担う部品の部位間の関係を座標系間の関係で表現する方法[11)]，また，軸ものを対象とした設計支援システムを提案している[12)~14)]．これらの研究は，公差解析システム構築のための基本概念の幾つかを提案したが，設計実務に関わる知識を体系的に整理する枠組としては十分とは言えない．そこで，筆者らは次のような手順で一連の研究を行っている．

（1）公差解析に関わる知識を統一的に表現できる知識表現言語を考案し，その処理系を試作する．

（2）上記の知識表現言語を用いて，公差解析に関わる知識を表現するための技法☆（以下，プログラミング手法）を開発する．

（3）設計実務を分析することにより，設計者が暗黙に使用しているばらつき発現モデルおよび公差解析手法を上記（2）を用いて系統的に整理する．

（4）公差解析のための知識表現言語を改良し，その実用処理系を開発する．

（5）カメラ等の機構設計を対象とした公差解析システムを開発する．

（6）設計現場に導入することにより，公差解析システムの運用技術を完成させる．

本論文では，これらの一連の研究の基礎となる上記（1）（2）について述べる．以下，2章では，公差解析のために必要な知識表現言語の要件を述べる．3章では，本論文で用いた知識表現言語について述べる．4章では，3章の知識表現言語を用いて公差解析システムを構築するためのプログラミング手法について述べる．5章では，4章のプログラミング手法を用いて試作した公差解析システムについて述べる．6章では，実用規模の設計問題を用いて記述・解析実験を行い，本論文に提案したプログラミング手法が効果的であることを示す．7章では，本論文の総括を行う．

2. 知識表現言語の要件

本論文が提案する知識表現言語の要件は次のとおりである．

（1）**知識表現の範囲**　本言語は，民生機器等の部品や組立品の公差解析を中心とした設計検証システムを構築するための支援環境の一部であり，公差解析に関する知識を統一的に表現できる必要がある．

（2）**組立構造の表現**　本言語は，部品の構造や組立品の構造を体系的に表現することができ，一般の設計者にとって理解性，保守性の良い環境を提供する必要がある．

（3）**公差解析のための推論機構**　本言語は，最悪解析，統計解析，モンテカルロ法等の既存の公差解析手法や製品ごとに存在するばらつき発現モデルを統一的に表現できる推論機構を提供する必要がある．

（4）**利用者環境への配慮**　利用者は設計者であるため，従来の設計作業との連続性を保つ適切な使用環境を必要とし，形状の表現に用いられる通常の3面図や形状モデラーとの結合が必要である．このため本言語は，他の処理系との結合性が良く，設計者に違和感を与えない良好な環境を構築できる必要がある．

3. 知識表現言語

本研究では，2章の要件を満たすために適切と考えられる，オブジェクト指向と拘束条件解法を融合した知識表現言語を用いる．ここで，オブジェクト指向の機能は組立構造の表現や利用者インターフェースの作

☆ 知識ベースシステムを構築するためのプログラミング手法を知識ベースプログラミング[15)]と言う．

成に，拘束条件解法は寸法や組立などの制約の表現に用いる．

ここでは，著者の一人が先に考案した設計システム記述言語 ADL[16),17)]の基本概念を流用し，判読性を向上させるために記述形式を改め，機能の一部* を削除した．なお，理解を容易にするため，既報の内容と一部重複させている．

3.1 オブジェクト指向

この節では，オブジェクト指向の機能について述べる．

通常のオブジェクト指向と同様に，同じ性質を持つ部品や組立品を代表する表現をクラス，クラスから生成される個々の部品や組立品の表現をインスタンスと呼ぶ．

クラス **図1**にクラスの記述形式を示す．図中，〈クラス名〉はそれを識別する一意な名称を記述する．〈上位クラス〉はクラス間の階層関係を定義し，上位のクラス名を指定する．〈属性定義〉はクラスの性質を表す属性名とその型を記述し，必要に応じてデフォルトの式を記述する．〈部分定義〉は，〈部分名〉と〈クラス名〉の対からなり，クラスが持つ部分を記述する．ここで〈部分名〉は部分を識別するためのクラスに一意な名称，〈クラス名〉は部分が属するクラス名を記述する．〈拘束条件〉は属性や部分間の関係を表す拘束条件を記述する．

ここでは，**図2**に示す一対の平歯車[18)]（以下，歯車対）の例を用いて，クラスの記述形式を説明する（**図3**）．図2中，外径は歯車の大きさを表し，ピッチ円直径[18)]は歯の寸法を決める場合に基準になる互いに転がり接触する円筒面の大きさを表す．図3中，gear は平歯車，gear_pair は歯車対のクラスをそれぞれ表す．gear は，属性として歯数 teeth とモジュール** module を持つ．gear_pair は部分として駆動歯車 driver，従動歯車 follower を，属性として入力回転数 rpm_in と出力回転数 rpm_out を持つ．また拘束条件として，歯車対の歯数と回転数の関係を表す拘束条件 gear_ratio を持つ．この gear_ratio の引数 driver!teeth は駆動歯車の歯数を，follower!teeth は従動歯車の歯数を，!rpm_in は歯車対の入力回転数を，!rpm_out は出力回転数を表す．

メソッド クラスは，そのインスタンスを処理するメソッドを持つことができる．**図4**にメソッドの記述形

```
<クラス定義> ::=
    defclass: <クラス名>,
      super: <上位クラス名>,
      attributes: <属性定義>, …, <属性定義>,
      sub_parts: <部分定義>, …, <部分定義>,
      constraints: <拘束条件>, …, <拘束条件>.

<クラス名>        ::= <文字列>
<上位クラス名>    ::= <文字列>
<属性定義>        ::= <属性名>:<型> |
                      <属性名>:<型> := <式>
<属性名>          ::= <文字列>
<式>              ::= <値> | <計算式>
<型>              ::= <システムが提供する
                       任意の型>
<計算式>          ::= <システムが提供する
                       任意の算術演算や関数>
<値>              ::= <システムが提供する
                       任意のデータ>
<部分定義>        ::= <部分名>: <クラス名>
<部分名>          ::= <文字列>
<拘束条件>        ::= <拘束条件名>(<引数>, …,
                                  <引数>)
<拘束条件名>      ::= <文字列>
<引数>            ::= <値> | <属性参照>
<属性参照>        ::= !<属性名> |
                      <部分名>!<属性参照>
<部分参照>        ::= !<部分名> |
                      <部分名>!<部分参照>
```

図1 クラスの記述形式
Fig. 1 Syntax of classes.

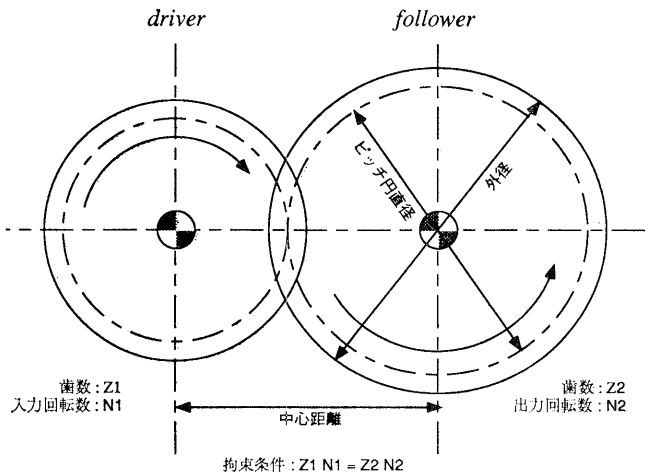


図2 一対の平歯車
Fig. 2 A pair of spur gears.

* 拘束条件リダクション過程においてオブジェクトを動的に生成する機能．

** ピッチ円直径を歯数で割った値をモジュールといい，一般に歯の大きさを表す．

式を示す．図中，〈クラス名〉はメソッドが定義されるクラスの名称，〈セレクタ〉はメソッドを識別するためのクラスに一意な名称，〈メソッド本体〉はシステムが用意する述語の呼び出しや代入文からなる手続きである．**表1**に，クラスのインスタンスを操作するメソッドの例を示す．

インスタンスの生成　クラスの記述からインスタンスを生成するには，次の述語を用いる．

make_instance(*Ins*, *Class*,
　　attributes($a_1(v_1), \cdots, a_i(v_i), \cdots, a_n(v_n)$),
　　sub_parts($p_1(i_1), \cdots, p_j(i_j), \cdots, p_m(i_m)$))

ここで，*Ins* は変数，*Class* は生成するインスタンスが所属するクラス名を指定する．$a_i(v_i)$ $(i=1, \cdots, n)$ は属性 a_i の初期値が v_i であることを，$p_j(i_j)$ $(j=1, \cdots, m)$ は既に生成されているインスタンス i_j をインスタンス *Ins* の部分 p_j として組入れることを表す．この述語を実行すると変数 *Ins* に生成されたクラス *Class* のインスタンスの識別名が代入される．

メソッドの実行　クラスに定義されたメソッドを実行するには，次の述語を評価すれば良い．

send(〈インスタンス名〉, 〈セレクタ〉(〈引数〉…〈引数〉))

ここで〈インスタンス名〉はインスタンスの指定に，〈セレクタ〉はクラスに定義されたメソッドの識別に用いられる．

簡単な例を用いてオブジェクト指向の機能を説明する．図3に示した歯車対において，中心距離をピッチ円直径の和の 1/2 として計算するメソッドは次のようになる（図2参照）．

```
defmethod(gear, diameter(D)) :-
    D is !teeth * !module.
defmethod(gear_pair, shaft_distance(L)) :-
    send(driver, diameter(D1)),
    send(follower, diameter(D2)),
    L is (D1+D2)/2.
```

ここで，クラス gear のメソッド diameter は歯数とモジュールからピッチ円直径を求め，クラス gear_pair のメソッド shaft_distance は，中心距離を求める．

今，歯車のモジュールが3，歯数が 19 および 64 枚の歯車対 Gp の中心距離Lを求めるには，次のようにすれば良い．

```
?- make_instance(G1, gear,
                 attributes(teeth(19),
                            module(3))),
   make_instance(G2, gear,
                 attributes(teeth(64),
                            module(3))),
```

```
defclass: gear,
  super: object,
  attributes: teeth:int := unknown,
              module:real := unknown.
```

（a）　平歯車のクラス
（a）　The class spur gear

```
defclass: gear_pair,
  super: object,
  attributes: rpm_in:real := unknown,
              rpm_out:real := unknown,
  sub_parts: driver: gear,
             follower: gear,
  constraints:
    gear_ratio(driver!teeth,
               follower!teeth,
               !rpm_in, !rpm_out).
```

（b）　歯車対のクラス
（b）　The class gear pair

図 3　平歯車と歯車対のクラス
Fig. 3　The class spur gear and gear pair.

```
defmethod(<クラス名>,
        <セレクタ>(<引数>,…,<引数>)) :-
    <メソッド本体>.
<セレクタ>      ::= <文字列>
<メソッド本体>  ::= <文>,…,<文>
<文>            ::= <述語呼び出し> | <代入文>
<述語呼び出し>  ::= <述語名>(<引数>,…,<引数>))
<代入文>        ::= <変数> is <式>
```

図 4　メソッドの記述形式
Fig. 4　Syntax of methods.

表 1　インスタンスを操作するメソッドの例
Table 1　Examples of methods for manipulating instances.

メ ソ ッ ド	機　　能
add_sub_part(Self, <部分名>, <クラス名>)	インスタンスに部分を追加
add_constraint(Self, <拘束条件>)	インスタンスに拘束条件を追加
get_constraints(Self, <拘束条件集合>)	インスタンスから拘束条件を集める
get(Self, <属性名>, <変数名>)	インスタンスから属性値を取り出す
put(Self, <属性名>, <値>)	属性値をインスタンスへ戻す

```
make_instance(Gp, gear_pair,
                sub_parts(driver(G1),
                          follower(G2))),
send(Gp, shaft_distance(L)).
```

3.2 拘束条件解法

この節では，クラスに定義された拘束条件集合を解くための推論機構について述べる．拘束条件解法には，素論理式の集合で表した拘束条件集合をリダクション手続で解く拘束条件リダクション法[16),17)]を用いる．

リダクションルールの記述形式 プログラムは，次の形式で記述されるリダクションルール（以下，ルール）の集合である．

$H :- G_1, \cdots, G_n \mid B_1, \cdots, B_m.$

Hをルールの頭部（head），$G_1, \cdots, G_n$ をガード（guard），$B_1, \cdots, B_m$ を本体（body）と呼び，いずれも素論理式で表す．ルールの意味は，ガード $G_1, \cdots, G_n$ が成立するとき拘束条件Hが拘束条件集合 $B_1, \cdots, B_m$ に分解可能であることを表す．また頭部の引数には，入力引数を示す修飾子?，出力引数を示す修飾子@を付与することができる．例えば乗算のルールは，次のようになる．

```
mult(?X, ?Y, @Z) :-  true      | call(Z is X*Y).
mult(@X, ?Y, ?Z) :-Y=\=0.0 | call(X is Z/Y).
mult(?X, @Y, ?Z) :-X=\=0.0 | call(Y is Z/X).
```

このルールは，3つの引数のうち2つの引数に値が与えられた場合に，残りの引数に値が求まるように記述している．またルール本体の call は，その引数をシステムが用意する述語や代入文として実行することを意味する．

リダクション手続き **図5**は，リダクション手続きをprolog を用いて説明したものである． reduce 述語は，第1引数に解こうとする拘束条件集合，第2引数に待ち状態の拘束条件集合を持つ．①はリダクション手続きの終了条件を示す．②は call を処理する．③は拘束条件にルールを適用しその本体 Bs に展開する．ここで rule(H, G, Bs) は，ルールの頭部H，ガードG，拘束条件 Bs を取り出す述語である．④は遅延処理である．

```
① | reduce([ ], W).
② | reduce([call(R)|Rs], Ws):-
       !,call(R),
       append(Rs, Ws, Rs1),
       reduce(Rs1, [ ]).
③ | reduce([H|Rs], Ws):-
       rule(H, G, Bs),
       call(G),!,
       append(Bs, Rs, Rs1),
       append(Rs1, Ws, Rs2),
       reduce(Rs2, [ ]).
④ | reduce([R|Rs], Ws):-
       reduce(Rs, [R|Ws]).
```

図5 リダクション手続き
Fig. 5 Reduction procedure.

図3で与えたクラス gear_pair の拘束条件 gear_ratio を例にとり，リダクション過程を説明する．gear_ratio のルールは次のようになる．

```
gear_ratio(Z1, Z2, N1, N2) :-
    true |
    mult(Z1, N1, A),
    mult(Z2, N2, A).
```

ここで Z1, Z2, N1, N2 は，それぞれ駆動歯車の歯数，従動歯車の歯数，入力回転数，出力回転数を表す．またそれぞれの歯数と回転数の積が一定であることを表す．**図6**は，このルールに Z1=19, Z2=64, N1=1280 を与えた場合のリダクション過程である．図中，①は初期状態を，②は gear_ratio を2個の mult に分解した状態を，③は mult を実行した状態を，④は解が求まった状態を示す．また _ は値が未定であることを示す．

リダクション手続きのオブジェクト指向への拡張 リダクション手続きをオブジェクト指向へ拡張するために，拘束条件 get を導入する．get はリダクション手続きの中でインスタンスの属性値を記号表を用いて管

	拘束条件集合	変数値
①	gear_ratio(19, 64, 1280, N2)	Z1 = 19 N1 = 1280 Z2 = 64 N2 = _
②	mult(64, N2, A) mult(19, 1280, A)	Z1 = 19 N1 = 1280 Z2 = 64 N2 = _ A = _
③	mult(64, N2, 24320)	Z1 = 19 N1 = 1280 Z2 = 64 N2 = _ A = 24320
④		Z1 = 19 N1 = 1280 Z2 = 64 N2 = 380 A = 24320

図6 gear_ratio のリダクション過程
Fig. 6 The reduction process of gear_ratio.

理するためのものである．get は，

get(〈インスタンス名〉, 〈属性名〉, 〈変数〉)

のように記述し，〈インスタンス名〉と〈属性名〉により識別される属性値が〈変数〉に単一化される．

図7は，図5に示したリダクション手続きにオブジェクト指向の機能を拡張したものである．同図において，reduce の第3引数 Oblist は拘束条件 get から参照するインスタンスの属性値を管理する記号表である．また属性値参照のための手続き①を追加している．図中，lookup は記号表を参照するための述語である．

```
reduce([ ], W, Oblist).

reduce([call(R)|Rs], Ws, Oblist):-
    !,call(R),
    append(Rs, Ws, Rs1),
    reduce(Rs1, [ ], Oblist).

① reduce([get(Ins, Iv, V)|Rs], Ws, Oblist):-
    !,lookup(Ins, Iv, V, Oblist),
    append(Ws, Rs, Rs1),
    reduce(Rs1, [ ], Oblist).

reduce([H|Rs], Ws, Oblist):-
    rule(H, G, Bs),
    call(G),!,
    append(Bs, Rs, Rs1),
    append(Rs1, Ws, Rs2),
    reduce(Rs2, [ ], Oblist).

reduce([R|Rs], Ws, Oblist):-
    reduce(Rs, [R|Ws], Oblist).
```

図 7　オブジェクト指向のリダクション手続き
Fig. 7　Reduction procedure with object oriented.

	拘束条件集合	変数値	記号表
①	get(gear_pair!driver, teeth, Z1)	Z1 = _	
	get(gear_pair, rpm_in, N1)	N1 = _	
	get(gear_pair!follower, teeth, Z2)	Z2 = _	
	get(gear_pair, rpm_out, N2)	N2 = _	
	gear_ratio(Z1, Z2, N1, N2)		
②	get(gear_pair, rpm_in, N1)	Z1 = 19	[gear_pair!driver, teeth, 19]
	get(gear_pair!follower, teeth, Z2)	N1 = _	
	get(gear_pair, rpm_out, N2)	Z2 = _	
	gear_ratio(Z1, Z2, N1, N2)	N2 = _	
③	get(gear_pair!follower, teeth, Z2)	Z1 = 19	[gear_pair!driver, teeth, 19]
	get(gear_pair, rpm_out, N2)	N1 = 1280	[gear_pair, rpm_in, 1280]
	gear_ratio(Z1, Z2, N1, N2)	Z2 = _	
		N2 = _	
④	get(gear_pair, rpm_out, N2)	Z1 = 19	[gear_pair!driver, teeth, 19]
	gear_ratio(Z1, Z2, N1, N2)	N1 = 1280	[gear_pair, rpm_in, 1280]
		Z2 = 64	[gear_pair!follower, teeth, 64]
		N2 = _	
⑤	gear_ratio(Z1, Z2, N1, N2)	Z1 = 19	[gear_pair!driver, teeth, 19]
		N1 = 1280	[gear_pair, rpm_in, 1280]
		Z2 = 64	[gear_pair!follower, teeth, 64]
		N2 = _	[gear_pair, rpm_out, _]

図 8　get のリダクション過程
Fig. 8　The reduction process of 'get'.

今，クラス gear_pair に，次のようなリダクション手続きを実行するメソッドを用意する．

```
defmethod(gear_pair, solve(Self)) :-
    send(Self, get_constraints(Cs, Oblist)),    ……(1)
    send(Self, reduce(Cs, [], Oblist)),         ……(2)
    send(Self, puts(Oblist)).                   ……(3)
```

ここで(1)は，インスタンス Self から拘束条件 Cs を集め，Cs に含まれる属性値の参照を拘束条件 get へ置き換える．その後に，(2)によって図5に示したリダクション手続きを用いて拘束条件集合を解き，(3)のメソッド put によって実行結果をインスタンスに戻す．**図8**に，このメソッドを実行した際のリダクション過程を示す．同図は，拘束条件集合，変数値，および記号表の変化を表している．①はインスタンスから拘束条件集合を集めた直後の状態を，②〜④は get の実行により各変数の参照と記号表への追加が行われた状態を表す．また⑤以降は図6と同様に実行される．

4. 公差解析システム構築のためのプログラミング

この章では，3章の知識表現言語を用いて公差解析システムを構築するためのプログラミング手法について述べる．

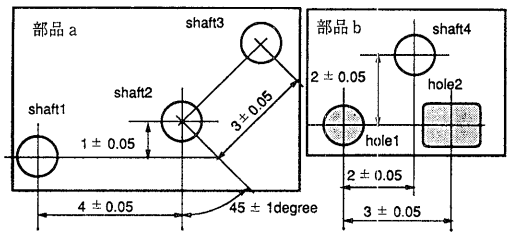
4.1 組立構造

公差解析の対象となる組立構造の表現について述べる．部品は，軸や穴のように他の部品と相互作用することにより機能を発現する部分を持つ[11),19)]．これを機能素* と呼び組立構造を特徴づける構成要素とする．機能素の例を**表2**に示す．部品は機能素から構成され，組立品は部品または下位の組立品から構成される．これら組立品，部品，機能素を総称して組立オブジェクトと呼ぶ．組立オブジェクトはその特徴を表す寸法，形状，材質等の属性を持つ．部品を構成する機能素の配置関係は，長さ寸法や角度寸法[1)] を指定することにより表現する．同様に，組立品の構造は

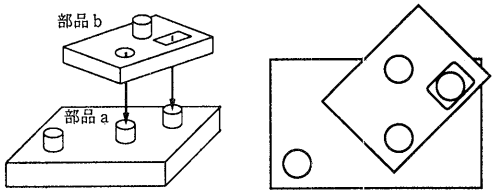
* 機能素という用語を用いたのは，文献19) が最初である．本論文の機能素は，文献11) に用いられている部品素の概念に近いものである．

表 2 機能素と属性の例
Table 2 An example of functional elements and attributes.

機能素	属性
point (点)	center (配置座標系)
shaft (軸)	center, diameter (直径)
hole (穴)	center, diameter
long_hole (長穴)	center, diameter, length (長さ), width (幅)
plane (面)	center



(a) 部品の例
(a) Example of parts



(b) 組立の様子
(b) The process of assembly

(c) 組立品 c
(c) Assembly C

図 9 組立品の例
Fig. 9 Examples of parts and an assembly.

表 3 寸法制約と組立制約
Table 3 Dimension constraints and assembly constraints.

拘束条件の種類	拘束条件名	目的
寸法制約	transfer	長さ寸法の指定
寸法制約	rotate	角度寸法の指定
組立制約	unify_point	位置の一致
組立制約	unify_orientation	姿勢の一致

部品を構成する機能素間の接続関係を指定することにより表現する.

ここで, 図 9 (a) に示す部品 a と部品 b の組立を例に組立構造を説明する. 部品 a は 3 つの軸を, 部品 b は穴, 長穴, 軸を持ち, それぞれ長さ寸法や角度寸法により位置を拘束されている. この 2 部品は, 図 9 (b) に示すように, 部品 a の shaft 2 と部品 b の hole 1, 部品 a の shaft 3 と部品 b の hole 2 のそれぞれがはめあい, 図 9 (c) に示す組立品 c を作る.

図 10 に, 組立品 c の組立構造を示す. 各部品を構成する機能素の属性間には, その配置関係を表すために長さ寸法や角度寸法に関する制約 (以下, 寸法制約) が付与され, また機能素間には部品の接続関係を表すはめあいや角度規制などの組立に関する制約 (以下, 組立制約) が付与されている. 図 10 中, 太線円は組立オブジェクトを, 細線円は属性を, 太線は全体・部分関係を, 矢印は寸法・角度制約および組立制約を, 破線は測定の対象とする組立オブジェクト間の関係を表している. ①〜⑤は寸法制約, ⑥⑦は組立制約を表す. また⑧は測定 measure_length である. 表 3 に寸法制約, 組立制約の例を示す.

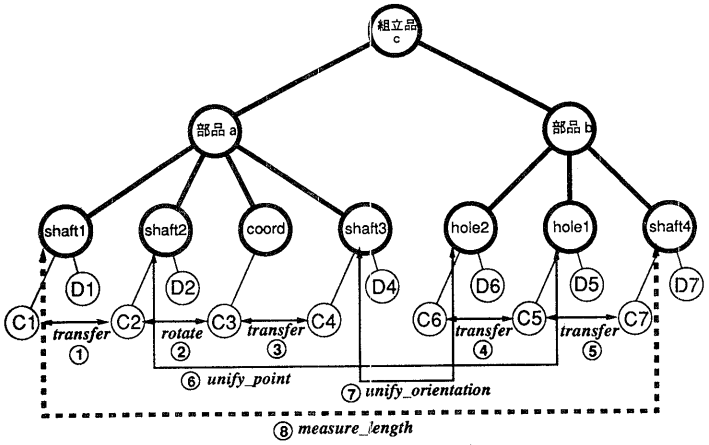


図 10 組立品を表現する組立オブジェクト
Fig. 10 An assembly object representating the assembly.

組立品の組立構造は一般に図 11 に示すようなクラス階層を用いて表現できる. 図中, object は組立オブジェクトを, functional_element は機能素を, part/assembly は部品や組立品の総称を, part は個々の部品を, assembly は組立品を表す.

4.2 長さ寸法と角度寸法

組立オブジェクトの配置位置を基本的なデータ構造として用いる. これを配置座標系 (以下, 混乱の恐れがないときは座標系と略す) と呼ぶ. 簡単のため 2 次元の座標系を例にする. 座標系の原点

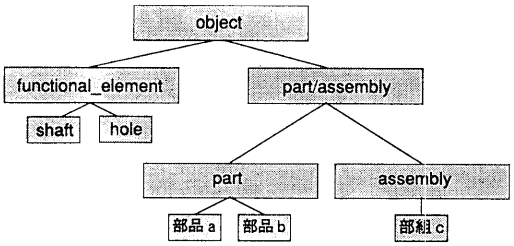


図 11 クラス階層
Fig. 11 Class hierarchy.

位置を Ox, Oy, 配置角度をRとすると（図 12(a) 参照), 座標系は {Ox, Oy, R} のように表現できる. ここで {} は, データの組を表す.

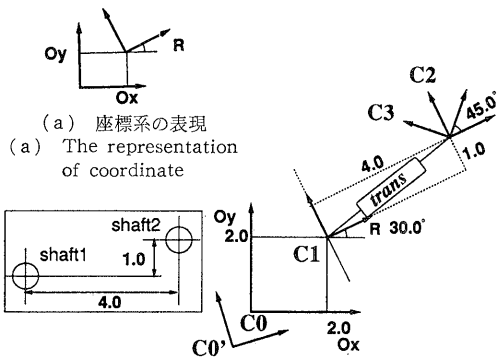
長さ寸法や角度寸法は, 座標系間の関係であると考えて良い. 例えば, 図 12(b) に示す2つの軸を持つ部品において, shaft 1 と shaft 2 の配置位置をそれぞれ座標系 C1, 座標系 C2 とし, この部品を配置する座標系を C0 とするとき, 座標系間の関係を模式的に表すと図 12 (c)のようになる. 図中, 2つの shaft の寸法は, 座標系 C2 が座標系 C1 からみてX方向に 4.0, Y方向に 1.0 平行移動したものである. この座標系間の関係を trans(C1, C2, {4.0, 1.0}) のように表現する. また座標系 C3 が座標系 C2 からみて 45.0° 回転した座標系であるとき, rot(C2, C3, 45.0) のように表現する. この2つのルールを図 13 に示す. 図中, ==は単一化を意味し, 数式は mult 等の拘束条件集合に分解する. また rotate_point は座標 {X0, Y0} を角度 RD 回転すると座標 {X1, Y1} になる関係を表すルールである.

今, 部品の座標系 C0 からみた C1 の位置姿勢が {Ox, Oy, R} = {2.0, 2.0, 30.0} とすると, C1 = {2.0, 2.0, 30.0}, C2≒{4.96, 4.87, 30.0}, C3≒{4.96, 4.87, 75.0} となる. これら寸法や角度の関係は, 部品の座標系 C0 とは独立であり, 他の座標系 C0' から見ても保存される.

4.3 公差領域と生成子

一般に公差付きの寸法は, 4±0.05 のように記述し, 称呼値, 最小公差, 最大公差の3組で与えられる. これを {4.0, −0.05, 0.05} のようなデータ構造で表現する.

寸法や角度に公差を指定した場合, 配置座標系の原点位置は, ある領域（以下, 公差領域）を形成する. 寸法公差は方形, 角度公差は扇形の公差領域となる. 図 9 (a)に与えた部品 a の公差領域を図 14 に示す. 例えば, shaft 2 の中心は shaft 1 の中心から, 長さ寸法によって横方向へ 4±0.05 と縦方向へ 1±0.05 の位置に



(a) 座標系の表現
(a) The representation of coordinate

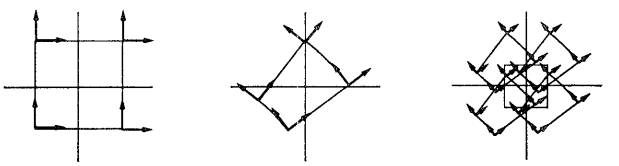
(b) 部品の例
(b) An example of part

(c) 寸法を座標系間の関係として表現
(c) The representation of dimension using relation of coordinates

図 12 寸法表現
Fig. 12 The representation of dimension.

```
trans({Ox1, Oy1, R1}, {Ox2, Oy2, R2}, {Dx, Dy}) :-
  true | rotate_point({Dx, Dy}, {DimXR, DimYR}, R1),
         {Ox2, Oy2} = {Ox1, Oy1} + {DimXR, DimYR},
         R1 == R2.
rot({Ox1, Oy1, R1}, {Ox2, Oy, R2}, R) :-
  true | R2 = R1 + R,
         {Ox2, Oy2} == {Ox1, Oy1}.
rotate_point({?X0, ?Y0}, {@X1, @Y1}, ?RD) :-
  true | X1 = (X0 * cos(RD)) - (Y0 * sin(RD)),
         Y1 = (X0 * sin(RD)) + (Y0 * cos(RD)).
rotate_point({@X0, @Y0}, {?X1, ?Y1}, ?RD):-
  true | X1 = (X0 * cos(RD)) - (Y0 * sin(RD)),
         Y1 = (X0 * sin(RD)) + (Y0 * cos(RD)).
rotate_point({?X0, ?Y0}, {?X1, ?Y1}, @RD) :-
  true | RD = acos((X0 * X1 + Y0 * Y1) /
                   (sqrt(X0^2 + Y0^2) *
                    sqrt(X1^2 + Y1^2))).
```

図 13 寸法制約のリダクションルール
Fig. 13 Reduction rules of dimension constraints.



shaft1 から見た shaft2 の公差領域
shaft2 から見た shaft3 の公差領域
shaft1 から見た shaft3 の公差領域

図 14 代表集合を用いて公差領域を表現
Fig. 14 A set of coordinates representating tolerance zones.

拘束されているため，方形の公差領域を形成し，shaft 2 の中心は，shaft 1 の中心から見て公差領域内の任意の位置姿勢に分布することを意味する．この公差領域を座標系の集合（以下，代表集合）を用いて公差領域を近似的に表現する．

この代表集合を表現するために非決定的な動作を行う拘束条件（以下，生成子）を用いる．

ここで，Dnom は称呼値，Dmin は公差の最小値，Dmax は公差の最大値とすると，寸法の最大値と最小値のみを考慮する最悪解析に用いるルールは次のようになる．

```
gen({?Dnom, ?Dmin, ?Dmax}, D) :-
    true |
    minmax({Dnom, Dmin, Dmax}, D).
```

このルールは，寸法の最大値と最小値を生成する minmax を用いて寸法Dを生成する．

モンテカルロ法に用いるルールは次のようになる．

```
gen({?Dnom, ?Dmin, ?Dmax}, D) :-
    true |
    call(Min is Dnom+Dmin),
    call(Max is Dnom+Dmax),
    random(Min, Max, D).
```

このルールは，寸法の最小値から最大値の範囲において，一様分布や正規分布等の乱数を生成する random を使用して，任意の分布に従う寸法Dを生成する．

一般に gen ルールを変更することにより，任意の分布の代表集合を生成することができ，各種の公差解析手法を容易に表現することができる．

4.4 寸法公差と角度公差

公差付き寸法を扱うために，4.2 節の trans を拡張し，transfer(C 1, C 2, {{4.0, −0.05, 0.05}, {1.0, −0.05, 0.05}}) のように表現する．このルールの定義は，公差解析手法ごとに異なる．最悪解析の場合，最大値，最小値の組合せを生成する gen と称呼値を計算する trans を用いると，次のようなルールになる．

```
transfer(C 1, C 2, {Dxt, Dyt}) :-
    true |
    gen(Dxt, Dx),
    gen(Dyt, Dy),
    trans(C 1, C 2, {Dx, Dy}).
```

例えば，C 1＝{0.0, 0.0, 0.0} とすると，C 2 は {3.95, 0.95, 0.0}，{4.05, 0.95, 0.0}，{3.95, 1.05, 0.0} {4.05, 1.05, 0.0} の 4 通りの値をとる．角度公差についても同様の議論が成立し，ルールは次のようになる．

```
rotate(C 1, C 2, Rt) :-
    true |
    gen(Rt, R),
    rot(C 1, C 2, R).
```

今，図 9（a）に示した部品 a のばらつき発現モデルをクラスを用いて記述すると次のようになる．ここで，transfer や rotate を実現するルールを交換することにより，任意の公差解析手法を選択することができる．

```
defclass : part_a,
  super : part,
  attributes :
    dim 1 :dimension := {4.0, −0.05, 0.05},
    dim 2 :dimension := {1.0, −0.05, 0.05},
    deg 1 :dimension := {45.0, −1.0, 1.0},
    dim 3 :dimension := {3.0, −0.05, 0.05},
    dim 4 :dimension := {0.0, 0.0, 0.0},
  sub_parts :
    coord : coordinate,
    shaft 1 : shaft,
    shaft 2 : shaft,
    shaft 3 : shaft,
  constraints :
    transfer(shaft 1!center,shaft 2!center,
            {!dim1, !dim 2}),
    rotate(shaft 2!center, coord!center, !deg 1),
    transfer(coord!center, shaft 3!center,
            {!dim3, !dim 4}).
```

ここで，クラス中の dimension は公差付き寸法を表す型である．

4.5 座標変換

個々の部品が持つ座標系をローカル座標系，部品の外部に任意にとった座標系をグローバル座標系と呼ぶ．図 9 で与えた組立品を例にすると，shaft や hole 等に張り付けられた座標系の位置姿勢にはローカル座標系から見た位置と，グローバル座標系から見た位置とがある．これらローカル座標とグローバル座標との変換を行うルールは次のようになる．

```
local_to_global({OxL, OyL, RL}, {OxG, OyG,
  RG}, {Ox, Oy, R}) :-
  true |
  RG=RL+R,
```

{OxG, OyG} = {Ox, Oy} + {X, Y},
rotate({OxL, OyL}, {X, Y}, R).

このルールは，ローカル座標系がグローバル座標系からみた位置姿勢 {Ox, Oy, R} に配置されているとき，ローカル座標値 {OxL, OyL, RL} とグローバル座標値 {OxG, OyG, RG} を相互に変換する．この座標変換を用いて，部品の結合する位置や姿勢の拘束関係を表現すると**図 15** のようになる．図中，破線枠は，ローカル座標系とグローバル座標系の変換を表す．すなわち位置姿勢を，枠の内側ではローカル座標に変換し，枠の外側ではグローバル座標に変換して読む．

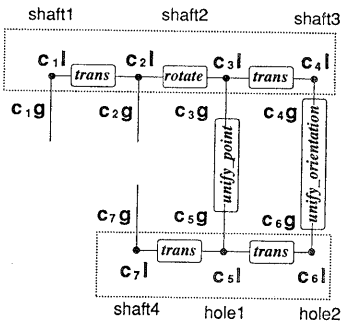


図 15 座標変換
Fig. 15 Coordinate transformation.

4.6 座標系の拘束

一般に，座標系の位置や姿勢を拘束する方法は様々であり，部品の組立方法によって異なる．例えば，座標系を拘束するルールは，次のように記述することができる．

unify_point({Ox1, Oy1, R1}, {Ox2, Oy2, R2} :-
  true |
  {Ox1, Oy1} == {Ox2, Oy2}.
unify_orientation({Ox1, Oy1, R1}, {Ox2, Oy2, R2}) :-
  true |
  R1==R2.

ここで unify_point は座標系の位置を，unify_orientation は座標系の姿勢を拘束する．図 9 (c) に与えた組立品 c をこのルールを用いて記述すると次のようになる．

```
defclass : asse_c,
    super : assembly,
        sub_parts : a : part_a,
                    b : part_b,
    constraints :
        unify_point (a!shaft2!coordniate,
                     b!hole1!coordinate),
        unify_orientation (a!shaft3!coordinate,
                           b!hole2!coordinate).
```

ここでは，部品 a の shaft 2 が持つ座標系の位置と部品 b の hole 1 が持つ座標系の位置を，部品 a の shaft 3 が持つ座標系の姿勢と部品 b の hole 2 が持つ座標系の姿勢を一致させることにより部品 b の位置姿勢を拘束した．

5. 公差解析システムの試作

本論文で提案した知識表現言語を用いて試作した公差解析システムについて簡単に述べる．

システムの概要を**図 16** に示す．システムは編集や表示のためのツール群，組立構造にルールを作用させ公差解析を行う推論機構，知識ベース，およびデータベースから構成される．ここで知識ベースには，公差積算のためのルールや位置姿勢のばらつき発現モデルを表現したルールが格納され，データベースには組立構造が格納される．

システムの試作は次のように行った．オブジェクト指向の機能は，prolog の述語を用いてクラス，インスタンス間の関係を表現することにより実現した．拘束条件解法は，局所伝搬法[20]を実現した．また，効率を改善するために拘束条件集合をC言語のプログラムへ翻訳する機能を開発した．

6. 解析実験と評価

本論文で提案した知識表現言語やプログラミング手法の有効性を確認するために，試作システムを用いて，**表 4** に示す解析実験を行った．表中，例 1 は図 9 に示した組立品 c の shaft 1 から見た shaft 4 の位置を最悪解析を用いて検証したもの，例 2 は公差解析を

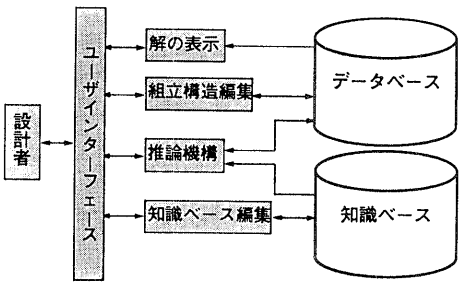


図 16 システムの概要
Fig. 16 The system overview.

表 4 解析実験
Table 4 Analysis experiment.

例	内　　容	部品	公差付き寸法	処理時間
1	図9に与えた例	6個	6個	25秒
2	レリーズボタン	5個	13個	48秒
3	レンズの姿勢	2個	10個	110秒

組立性検証へ適用し[21]，干渉の検証を行ったもの（付録A(1)参照），例3はレンズケースの性能を決めるレンズ受け面のばらつき* を表現し，ばらつきを推論し，設計仕様を満たすかどうか検証した（付録A(2)参照）ものである．これらの例題の記述はカメラ設計者が行った．

解析実験の結果は，次のように整理できる．

(1) 本知識表現言語は公差解析業務に関わる知識の記述に十分な表現能力を備えていることが確認できた．また，例3のような製品の性能に依存するばらつき発現モデルは，4.3 節で示した生成子を変えることにより表現できた．

(2) 本言語を用いて，設計者が知識の記述が行え，設計者が書いたプログラムを第3者の設計者が判読できた．このことから，知識の隠蔽化の防止や情報処理技術者が行う知識獲得作業の軽減に役立つものと思われる．

(3) 一般に，機械系の組立品の干渉問題を解くような公差解析では，多くの公差付き寸法を積算することはなく，高々，表4に示した程度である[8]．試作システムを用いて記述実験を行い，本手法の実現性を確認できた．

7. むすび

本論文をまとめると次のようになる．

(1) 公差解析システムのために必要な知識表現言語を提案した．

(2) 公差解析システム構築のために必要なプログラミング手法を整理することにより，組立構造，寸法，公差の各表現手法を示し，これらを用いてばらつき発現モデルおよび公差解析手法を表現できる見通しを得た．

(3) 試作システムを用いて，実際の公差解析問題を記述することで，本論文の接近法が効果的であることを示した．

(4) 本手法を用いることにより，ばらつき発現モデルや公差解析手法に関する知識の隠蔽化の防止や判読性を向上させた．これにより設計者による公差解析システムの保守や拡張が可能であることを示した．

本論文で述べた手法により設計者による知識の記述や保守が可能な公差解析システムを実現できる見通しを得た．しかしながら公差解析システムを実用化し実際の設計業務で運用するには，次のような課題が残されている．

(1) 公差解析システムを構築するための支援環境を整備する必要がある．例えば，従来の設計作業との連続性を保つために設計対象を一元的に管理するデータベースの構築や既存の処理系との結合，操作性を向上させるためのユーザインターフェースやデバッギング環境の構築などがある．

(2) 公差解析システムを設計現場で運用するためには，実際の設計業務を分析し，公差解析に必要な知識や解析結果の評価に関する知識の体系化を十分に行う必要がある．特に，製品の品質やコストに影響する製品の特性とばらつき発現の機構をモデル化することは重要である．

謝辞 日頃，議論して頂く九州工業大学情報工学部機械システム工学科 林朗 助教授（現，広島市立大学情報科学部情報機械システム工学科教授），周能法 助教授，ならびに本研究を行う上で御協力頂きました皆様に深謝します．

* 本論文の範囲を越えるので，詳細は別稿で述べる．

参考文献

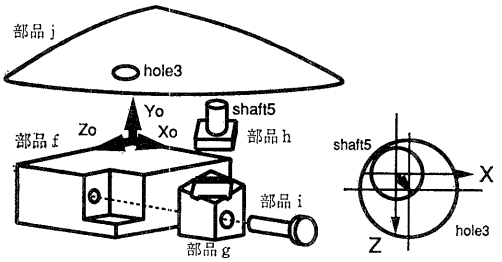
1) 製図用語 JIS Z 8114-1984, 日本規格協会(1984).
2) 機械システム設計便覧編集委員会（編）: JIS に基づく機械システム設計便覧, pp. 49-77, 日本規格協会 (1986).
3) 畑村洋太郎: 実際の設計, 日刊工業新聞社 (1988).
4) JIS ハンドブック 5 機械要素, pp. 403-522, 日本規格協会 (1992).
5) Turner, J. U. and Gangoiti, A. B.: Tolerance Analysis Approaches in Commercial Software, *Concurrent Engineering*, pp. 11-23 (1991).
6) 高橋 究, 鈴木宏正, 木村文彦: 公差の計算機処理に関する研究動向, 第8回設計シンポジウム講演論文集, pp. 82-87 (1990).
7) 松居吉哉: 光学系公差の合理的決定方法について, 光学技術コンタクト, Vol. 4, No. 2, pp. 13-17 (1966).
8) 樋口達治, 長澤 勲, 望月雅光, 梅田政信, 小島崇司: 知識表現言語を用いたばらつき解析のための一手法, 信学技報 KBSE 93-37, pp. 33-40 (1994).

9) 吉川弘之：機械のトポロジ，精密機械，Vol. 38, No. 12, pp. 1018-1023 (1972).
10) 北嶋克寛，吉川弘之：階層的ネットワークモデルに基づく対話型機械設計システム HIMADES-1の開発，精密機械，Vol. 47, No. 12, pp. 1490-1497 (1981).
11) Ito, M. and Kono, M.: CONMOTO A Machine Part Description System Based on Designers' Mental Processes, *Proc. of IFIP W. G. 5. 2. Working Conference* (1985).
12) 関口 博，小島俊男，井上久仁子，本多庸悟：回転機能部品の部品展開手法に関する研究，精密工学，Vol. 51, No. 2, pp. 359-365 (1985).
13) 関口 博，今村 聡，小島俊男，井上久仁子：回転機能部品の部品展開手法に関する研究（第2報），精密工学，Vol. 53, No. 8, pp. 1183-1188 (1987).
14) 関口 博，今村 聡，小島俊男，井上久仁子：回転機能部品の部品展開手法に関する研究（第3報），精密工学，Vol. 54, No. 9, pp. 1782-1786 (1988).
15) 渡辺正信：知識ベースプログラミング，海文堂，東京 (1982).
16) 長澤 勲，古川由美子，荒牧重登：論理プログラミングを基礎とした設計システム言語 ADL，情報処理，Vol. 25, No. 4, pp. 606-613 (1985).
17) 長澤 勲，古川由美子：拘束条件リダクション法を用いた機械設計計算支援システム，情報処理 Vol. 27, No. 1, pp. 112-120 (1986).
18) 歯車用語 JIS B 0102-1983, 日本規格協会(1983).
19) 富山哲男，吉川弘之：設計過程モデル論，精密機械，Vol. 49, No. 4, pp. 441-446 (1982).
20) Steele, G. L., Jr.: *The Definition and Implementation of a Computer Programming Language Based on Constraints Technical Report*, MIT Artificial Intelligence Laboratory (1980).
21) 望月雅光，長澤 勲，梅田政信，樋口達治，小島崇司：製品開発における組立性評価のための知識表現と推論機構，PROLOG 産業応用シンポジウム論文集 1992, pp. 35-43 (1992).
22) 幾何公差の図示方法 JIS B 0021-1984，日本規格協会 (1984).
23) 幾何偏差の定義及び表示 JIS B 0621-1984，日本規格協会 (1984).

付録A

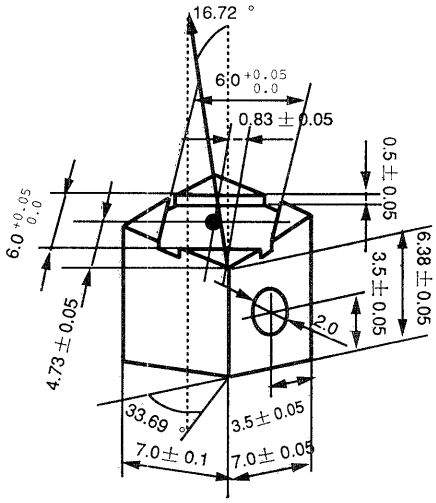
（1） 組立性へ適用した実用規模の例を示す．

図 A1 に 2.5 次元の公差解析問題を示す．同図はカメラのレリーズボタンの部分を模式的に描いたものである．図 A 1（a）は組立図を示し，部品 j が外観部品，shaft 5 がレリーズボタンである．もし公差の設定が悪いと，外観部品の穴 hole 3 とレリーズボタン



（a） 組立図 （b） 測定場所
（a） Assembly drawing （b） The point of measurement

図 A1 公差解析の例題
Fig. A1 An example of tolerance analyses.



（a） 部品 g の部品図
（a） The part drawing of part_g

```
defclass: part_g,
  super: part,
  attributes:
    dim1:dimension3 :=
      {{0.0, 0.0, 0.0},{3.0, -0.05, 0.05},{3.5, -0.05, 0.05}},
    dim2:dimension3 :=
      {{7.0, -0.1, 0.1},{0.0, 0.0, 0.0},{0.0, 0.0, 0.0}},
    dim3:dimension3 :=
      {{7.0, -0.1, 0.1},{6.38, -0.05, 0.05},{7.0, -0.05, 0.05}},
    dim4:dimension := {16.72,33.69,0.0},
    dim5:dimension3 :=
      {{-0.83,-0.05,0.05},{0.0,0.0,0.0},{-4.73,-0.05,0.05}},
    dim6:dimension3 :=
      {{0.0,0.0,0.0},{-0.5,-0.05,0.05},{0.0,0.0,0.0}},
  sub_parts: origin_g: coordinate,
             hole4: hole,
             coord1: coordinate,
             coord2: coordinate,
             hole5: hole,
  constraints:
    transfer(origin_g!center,hole4!b!center,dim1),
    transfer(hole4!b!center,hole4!t!center,dim2),
    transfer(origin_g!center,coord1!center,dim3),
    rotate(coord1!center,coord2!center,dim4),
    transfer(coord1!center,hole5!b!center,dim5),
    transfer(hole5!b!center,hole5!t!center,dim6).
```

（b） part_g クラス
（b） The class part_g

図 A2 部品 g
Fig. A2 part_g.

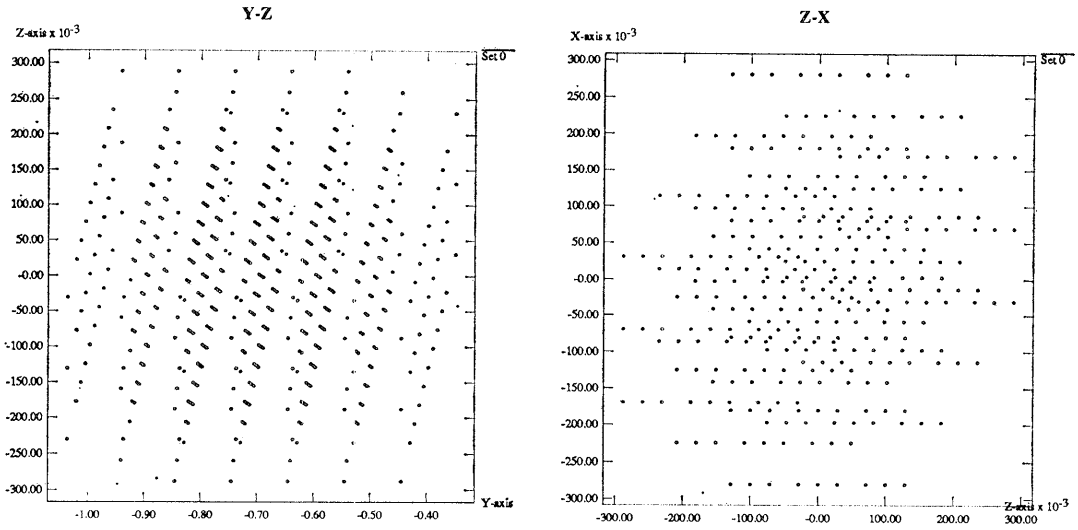


図 A3 実行結果
Fig. A3 Execution result.

表 A1 測定した座標値の最大値と最小値
Table A1 Coordinate obtained from the experiment.

	X	Y	Z
nominal	0.000000	−0.692726	0.000000
max	0.280278	−0.349070	0.288147
min	−0.280278	−1.036382	−0.288147

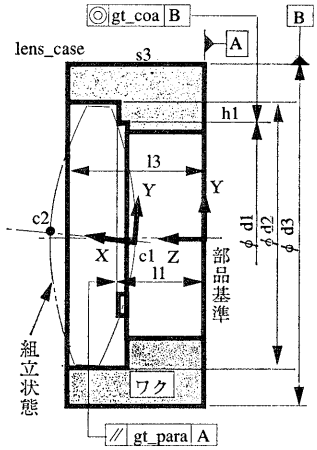


図 A4 レンズ組立の例
Fig. A4 An example of lens assembly.

の位置がずれ，商品の美観を損ねる．また穴からレリーズボタンが上に突き出ていなければ，機能を果たさない．そこで，図 A1（b）に示すようにレリーズボタンの中心を測定座標系 XYZ の原点とし，穴の中心までの距離を測定し，穴と軸の偏心が許容できるかどうかの，また部品間の干渉が無いかどうかの検証を行う．

ここで部品の記述例として，**図 A2**（a）に示す部品 g の部品図をクラスを用いて記述したものを図A 2（b）に示す．

測定の一例として最悪解析を用いた場合の測定結果を**図 A3** に示す．これは2方向から見た公差領域であり，その座標値を**表 A1** に示す．この結果から，レリーズボタンの直径は 3 mm，穴の直径は 4 mm なので偏心による干渉はなく，レリーズボタンは穴から約 0.7±0.35 mm 上に突き出ていることがわかる．

（2）**図 A4** に位置公差や姿勢公差[1]を含む，レンズ組立の例を示す．

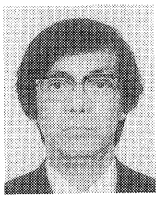
同図は，レンズとレンズケースを模式的に描いたものである．レンズの位置姿勢のばらつきは，レンズの性能に大きく関わっている．この設計では，レンズケースにレンズを受ける3点を設け，その3点からなる面の位置や姿勢のばらつきを平行度公差[22),23)]と同軸度公差[22),23)]を用いて規制している．

ここではレンズケースの基準を表す配置座標系から見たレンズ受け面の配置座標系のばらつきを本論文のプログラミング手法を用いて表現し，レンズの位置姿勢が設計仕様を満たすかどうかを検証した．

（平成5年12月20日受付）
（平成6年 6 月20日採録）

望月　雅光（学生会員）

昭和43年生まれ．平成3年近畿大学九州工学部経営工学科卒業．平成5年九州工業大学大学院情報工学研究科修士課程修了．現在，同大学院情報工学研究科博士後期課程在学中．知識情報処理の立場から設計支援システムの研究に従事．人工知能学会学生会員．日本経営工学会会員．

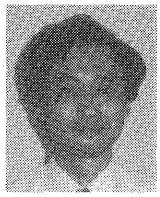


長澤　　勲（正会員）

昭和19年生まれ．昭和42年九州大学工学部電子工学科卒業．昭和47年同大学院工学研究科博士課程単位取得退学．昭和47年九州大学中央計数施設講師．現在，九州工業大学情報工学部教授（機械システム工学科）．工学博士．知識情報処理の立場からCAD/CAM，ロボット，医療システム等の研究開発に従事．人工知能学会，日本建築学会，精密工学会，電子情報通信学会，日本機械学会，日本設計工学会，日本ロボット学会各会員．

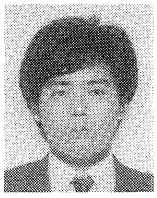
梅田　政信（正会員）

昭和34年生まれ．昭和57年九州大学理学部物理学科卒業．昭和59年同大学院工学研究科修士課程修了．昭和59年富士通株式会社．平成元年長崎県北工業技術センタ．現在，九州工業大学情報工学部助手（機械システム工学科）．知識情報処理の立場から設計支援システム等の研究開発に従事．精密工学会会員．



樋口　達治

昭和29年生まれ．昭和52年九州大学工学部生産機械工学科卒業．昭和54年同大学院修士課程修了．同年オリンパス光学工業(株)入社．平成4，5年九州工業大学情報工学部機械システム工学科非常勤講師．現在，オリンパス光学工業(株)第1開発部勤務．カメラ開発，および開発設計現場で実用可能な設計モデルに関する研究に従事．精密工学会会員．



小島　崇司

昭和43年生まれ．平成4年九州工業大学情報工学部機械システム工学科卒業．平成6年同大学院情報工学研究科修士課程修了．現在，同大学院情報工学研究科博士後期課程在学中．知識情報処理の立場から設計支援システムの研究に従事．人工知能学会学生会員．