TR-424

機械設計支援システム構築ツール MECHANICOT

寺崎　智、永井　保夫、横山　孝典、
井上　克己、堀内　英一、滝　寛和

October, 1988

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

機械設計支援システム構築ツール
MECHANICOT

寺崎　智　永井　保夫　横山　孝典
井上　克巳　堀内　英一　滝　寛和

（財）新世代コンピュータ技術開発機構　第５研究室

＜概　要＞

個別性の高い設計作業を支援するために，設計者自らが設計システムを構築し維持管理して行くことのできる構築ツールの提案を行った。

このツールは，宣言的に表現された設計対象の構造や設計手法に関する知識，および設計者のヒューリスティクスを入力とし，これらの設計知識を制約と捉えて制約間の依存関係を解析することにより設計手順を生成し，設計知識を問題解決機構と結び付ける。出力されたシステムは設計者のヒューリスティクスを含む専用設計システムとなる。さらに，このアーキテクチャに基づき現在開発中の，機械のパラメトリック設計を対象としたツールMECHANICOTについて述べる。

機械設計支援システム構築ツール
MECHANICOT

Mechanical Design System Building Tool: MECHANICOT

寺崎　智　　永井　保夫　　横山　孝典
Satoshi TERASAKI　Yasuo NAGAI　Takanori YOKOYAMA
井上　克巳　　堀内　英一　　滝　寛和
Katsumi INOUE　Eiichi HORIUCHI　Hirokazu TAKI

（財）新世代コンピュータ技術開発機構　第５研究室
Fifth Research Laboratory, Institute for New Generation Computer Technology

Abstract

Different groups of designers use different kinds of standard such as design methods, parts, and component units. Thus, building tools that an expert system can be built and maintained by designers themselves are required.

The purpose of this report is to clarify the architecture of the tool. The structure of a design object, knowledge about problem solving, heuristics, and design requirements are input to the tool. Viewing knowledge and design requirements as constraints, the tool analyzes dependencies among constraints, generates a design plan, and provides an interface between design knowledge and the constraint solver. The output of the tool is a specialized expert system including desiners' heuristics.

As a specific example, MECHANICOT which is a tool for a mechanical parametric design is described.

１．はじめに

設計は個別性の高い作業であり，設計における標準的な部品や手法は設計対象が異なれば全く違ったものとなる。また，過去の設計事例を参照して設計する場合でも，納期やコストといった設計要求により異なってくる。さらには，設計者の属する集団によりそれぞれの標準が存在する。従って，設計者自らが設計システムを構築し維持管理して行くことのできる環境の実現が強く望まれている。このためには，従来のように設計知識を手続き的プログラムとして表現したり，ある特定の問題解決機構と合うように記述するのを強いることを避けなくてはならない。これは，設計知識を支援システム内部に埋め込んで設計者から見えなくしてしまい設計知識の再利用や設計者間での継承をできなくしてしまう，ことを避ける上でも必要である[ICOT編 88]。

すなわち，設計知識を手続きとしてではなく，宣言的に，かつ問題解決機構と独立に表現しただけで設計エキスパートシステムを構築できるようになっている必要がある。

ここでは，このような問題を解決するための設計支援システム構築ツールのアーキテクチャの提案と，このアーキテクチャに基づいた，機械のパラメトリック設計を対象とした構築ツールMECHANICOTについて報告する。

まず，次章で設計型問題の考察を設計知識と問題解決機構の観点から行う。次に，宣言的に記述された設計知識を制約とみなし，制約解析を行うことにより設計手順を自動生成する構築ツールのアーキテクチャについて述べる。最後に現在開発中の構築ツールについて報告する。

２．設計型問題の検討

設計作業は，大きく概念設計と詳細設計に分けられるが，ここでは対象の構造がある程

度定まった後の詳細設計段階についてのみ扱うとする。本章では，まず設計知識について考察した後，問題解決機構について述べる。

2．1　設計知識

設計知識は，設計対象そのものに関する知識と，設計対象を解析し詳細化するための知識に大別される。設計対象に関する構造，形状，属性などの対象そのものに関する知識は，従来から対象モデルと呼ばれている[Ohsuga 85],[上野 85]。対象モデルでは，構造や形状に関する制約を表現する。これらの制約は，数値に関するものもあれば記号的なものもある。これに対し，対象モデルの解析方法や設計手順に関する知識をここでは設計手法に関する知識と呼ぶことにする。すなわち，設計手法知識は，対象モデルの解析，修正，評価に関する知識，解の探索手順に関する知識等から成り，今まで設計者のヒューリスティクスといわれていたものから，対象に関する知識を除いたものを指す。これらは，非線形の関数，等式や不等式等で表され，また，解に代替案がある場合やない場合がある。

このように設計知識にはさまざまな表現があるが，これらの表現は設計者の意図や設計対象の付加価値といった，陽には表現されない要求を反映して選択されたものとみなすことができる。ところが，従来のエキスパートシステム[Brown 86],[Mittal 86]等では，これらの知識を手続化して表現することに重点を置きすぎており，設計者の意図などの重要な情報を見失ってしまっていた。しかし，対象モデルと設計手法知識が与えられ，さらに設計者の意図を反映するようこれらの知識の実行順位に優先度が付けられたとすれば，設計手順に関する知識はこれらの知識の依存関係を解析する[Nagai 88a] ことにより生成できると考えられる。

一方，従来のシステムでは，対象モデルと設計手法知識が明確に分離されていないものが多かった。しかしこれでは対象に関する知識と設計手法に関する知識が未分化になり，システムの柔軟性を損ない，設計システムの高度化や一般性のあるシステムの構築を阻害してしまう。さらに，知識の再利用の観点からも，対象モデルと設計手法知識を分離して明示的に扱うことが必要である。これは詳細設計段階における設計過程を，①まず対象の構造を対象モデルを設定することで規定し，②ついでその対象モデルの解析方法（解析モデルあるいは工学モデル）を設計手法知識を選択することにより規定し，③これらのモデルの基で対象モデルの属性を詳細化する，とみなすことと同等である。

また，設計知識は全てが特殊なヒューリスティクスから成り立っているのではなく，標準的な部品を用いたり，よく知られた公式や実験式を用いたりすることが多い。従って，公式や実験式などの一般的な知識の記述量を減らすことが要求される。

2．2　問題解決機構

一般に設計手順が陽に与えられれば，それに従って解を得れば良い。ところが設計問題では，制約条件だけが与えられて，設計手法が陽に与えられない場合がある。この様な場合には，設計過程を制約充足過程と捉え，制約問題解決の枠組みを用いることが有効である。さらに，対象モデルは設計対象の構造の制約を表し，要求仕様及び設計手法に関する知識も制約とみなすと，設計過程全体を制約問題解決の観点から統一的に捉えることができる。これらの制約条件は，性能と納期・コストといった条件間のトレードオフや，設計者の意図・好みにより，優先度が付けられたり動的に変更されたりする。一方，従来のエキスパートシステムでは制約の概念が弱く，与えられた設計知識を有効に利用しているとはいえない。従って，設計問題に向いた制約問題解決機構が必要となる。

3．構築ツールのアーキテクチャ

以上のように我々は設計問題における知識を設計対象に関するもの（対象モデル）と設計方法に関するものに分離する構成を採用する。この方式は知識管理が容易となり，知識の修正にも柔軟に対応することができる。またこれらの知識や与えられる要求仕様を制約と捉え，設計問題を制約問題解決という統一的な枠組みで解くこととする。さらに問題に適した設計システムを設計者自らが容易に作成可能とするため，これらの設計知識を入力とし，制約の依存関係を基に設計手順を自動生成して，設計知識と問題解決機構とを結び付ける設計支援システム構築ツールを提案する。

3．1　知識表現

設計知識には，汎用的な知識と，ある特定目的に対してのみ有効な知識が混在している。例えば，設計手法知識では公式，実験式，カタログ検索などが汎用的な知識に当たる。また，対象モデルでは基本的な部品や機能ブロックが汎用的な知識に当たる。

そこで，設計者自身がこれらの知識を容易に表現できるようにするために，以下のような方法をとる。設計手法の知識のうち，公式やカタログなどの汎用的な知識は予めシステム側でライブラリとして用意しておく。同様に対象モデルも基本的な部品や機能ブロックを予めシステム側でライブラリとして用意しておく。なお，これらの知識表現にオブジェクト指向の枠組みを用いることにする。これは，対象モデルに関しては部品とその基本的な属性をオブジェクトとして自然に表現でき，設計手法知識に関してはその知識をメソッドとして容易に表現できる利点を持つことによる。従ってオブジェクト指向のクラス・メソッド・継承の概念を利用することにより，設計者はシステムのライブラリを参照したり，変更を加えるだけで，設計者自身のヒューリスティクスやある特定の設計のみに用いられる知識を宣言的に入力することができる。

さらに，このような機能を提供することにより，ある設計でどのような知識を用いたが明確になり，設計知識の明示化，標準化などに役立てることができる。

3．2　問題解決機構（制約ソルバ）

従来のエキスパートシステム構築支援ツールには制約という概念を表現できるものはなく，システム構築者自身がツールの開発言語を駆使して適用対象に依存した制約表現の適用メカニズムの実現を図っているのが現状である。本節では，設計問題を対象にした場合の制約の性質について述べ，必要な機能について考察する[Nagai 88b]。

多くの制約解決機構では，制約とは絶えず一定であり不変（静的）であるとして扱っている。ところが，設計問題では全ての制約条件が予め明らかになっているわけではなく，設計が進むにつれて追加・削除される場合も多い。また，すべての制約が対等なものとして選択・実行されるわけではない。つまり，制約には優先度があり，その重要性は設計要求や設計者の意図に依存する。義務的な制約はすべてが満足されねばならぬもので，一般的には陽に与えられるものである。一方，示唆的な制約は，選択枝より最良の枝を選択するためのガイドとして使用され，義務的な制約と比べて優先度は低い。従って，義務的な制約を満足できない時には，示唆的な制約を変更して義務的な制約が満たされるようにする場合もありうる。さらに，従来の制約問題解決機構は，制約の適用範囲は全て大局的なものとして扱っている。しかし，設計では問題をいくつかの部分問題に分割して解いていくことが多い。従って，制約の適用範囲が問題全体に及ぶ大局的なものか，部分問題の中だけに適用される局所的なものかを区別する必要がある。さらに，これらの部分問題の局所的な制約間のインタラクション，および局所的な制約と大局的な制約のインタラクションについても考慮する必要がある。最後に，設計問題を考えた場合，制約条件が不等式と

して表現される場合もあるので，値を伝播する制約だけではなく，値の取り得る領域を区間として伝播する制約に対しても考慮する必要がある。

従って，設計問題向けの制約ソルバには，制約を動的に追加・削除する機能，優先順位を考慮して解く機能，制約の有効範囲を指定できる機能，および，区間等の伝播情報に対する指定ができる機能が要求される。

3．3　アーキテクチャ

宣言的に表現された設計知識から設計手順を生成し，設計知識と問題解決機構とを結合するために，制約解析機構を用いる。制約解析機構は，知識コンパイラ[Araya 87]や制約コンパイラ[Feldman 88]と同様に，制約の依存関係を解析し，問題解決機構に対してそれらの制約を効率良く解く手順を与えるものである。いいかえれば，制約解析機構は汎用的な知識をノウハウやヒューリスティクスと組合わせて，ある特定目的のための知識に特化させるものと見ることができる。図3.3 に構築ツールのアーキテクチャを示す。

構築ツールへの入力としては，要求仕様，および対象モデルと設計手法に関する知識を与える。対象モデルと設計手法に関する知識は，設計者がシステムに付属しているライブラリを指定したり，参照したり，継承して変更を加えることにより与える。過去の設計事例を参照することや設計解の探索に関する設計者の持つヒューリスティクスなども設計手法として表現できる。

この様な入力に対し，構築ツールは，変数および制約条件間の依存関係を解析して設計手順を生成し，設計知識と問題解決機構とを結び付ける。構築ツールの出力は，設計者のヒューリスティクスを含んだある特定の設計対象のための専用設計システムとなる。

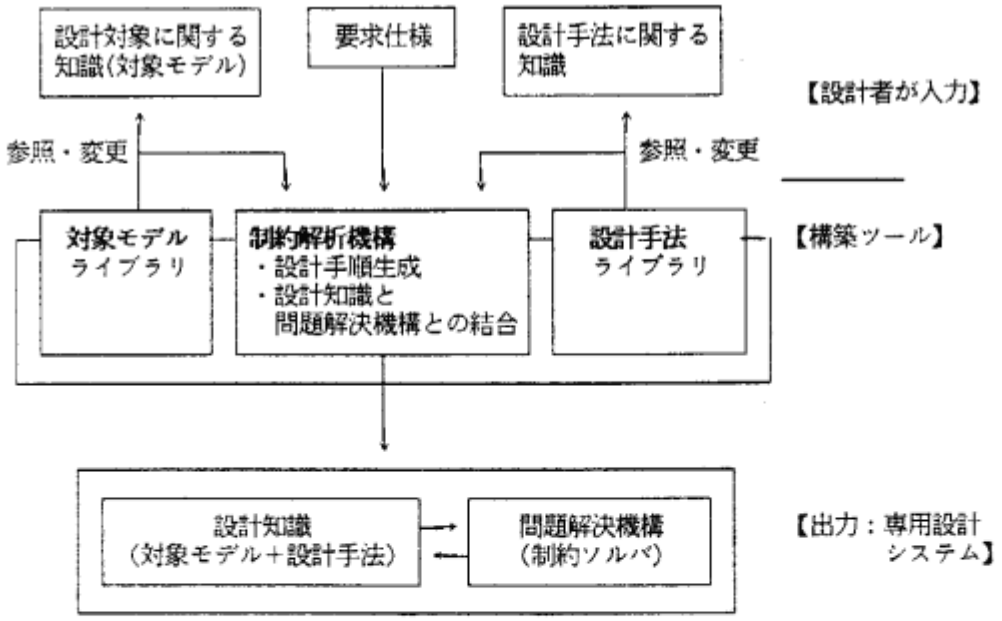

図 3.3　設計支援システム構築ツール

4．構築ツールの具体例：MECHANICOT

設計エキスパートシステム構築支援ツールの実現へ向けてのアプローチとして，第2，3章で上げた全ての機能を一度に実現するのではなく，段階的に実現する方針をとった。すなわち，第一段階としては予め設計対象の構造や設計手法知識がよく分かっているルーチン設計を対象に，制約を静的なものとして扱う制約解析機構の実現から開始した。この制約解析機構を中心に順次，対象モデルの改良，設計途中に動的に制約の付加・削除を許す制約ソルバの開発，制約への優先順位の付与等の改良を行っていく予定である。

この方針に基づいた構築支援ツールの具体例として，現在開発中の旋盤の主軸部分の設計を対象にした MECHANICOT について述べる。MECHANICOTは機械のパラメトリック設計を対象に，設計対象の構成やパラメータ間の依存関係に着目して設計手順を生成し，専用の設計システムを出力するツールである。MECHANICOTは，逐次型推論マシンPSI上のESPで記述されている。MECHANICOTは，入力として4.2 節で述べる情報をテキスト型式で与えると設計システムをESP のソースの形で出力する。また，問題解決機構として PSIの推論機構をそのまま使用することとした。

4．1　設計対象

設計対象は図4.1 に示すような工作機械の旋盤の主軸の部分である[井上 88]。ワーク（被切削物）を取り付けて回転させる主軸，動力源であるモータ，モータからの回転を伝えるためのVベルト，プーリおよびプーリ軸，回転軸を支えるためのベアリング，主軸の回転数を高・低の二段階に変速するギア等からなる。主軸設計問題は，ワークの材質，最大ワーク径，主軸の最高回転数，切削時の切込み深さ・ワークの送り速度等の切削能力に関する仕様と評価基準としてベアリングの寿命を与え，これらの仕様を満足するように各部の寸法を求めたり，パーツの品番をカタログから検索して決定する問題である。すなわち，この問題はルーチン設計で，かつ要求仕様を満足するパラメータを求めるパラメトリック設計問題と見なすことができる。なお，設計パラメータは規格値の利用やカタログからの選択等により離散値として扱うこととする。表4.1 に入力仕様と設計パラメータの例を示す。

表 4.1　入力仕様と設計パラメータの例

入力パラメータ		設計パラメータ
切削能力	対象ワーク材質	主軸径，プーリ軸径
	工具材質	ギア比，プーリ比
	最大ワーク径	ギア歯数，ピッチ円径
	最高回転数	ベアリングスパン
	最大切込み深さ	ベアリングマウント
	最大送り	ベアリング型式，品番
	ドリル径	モータ型式，品番
	ドリル切削速度	Vベルト型式，品番
		プーリ型式，品番
評価	ベアリング寿命	等々

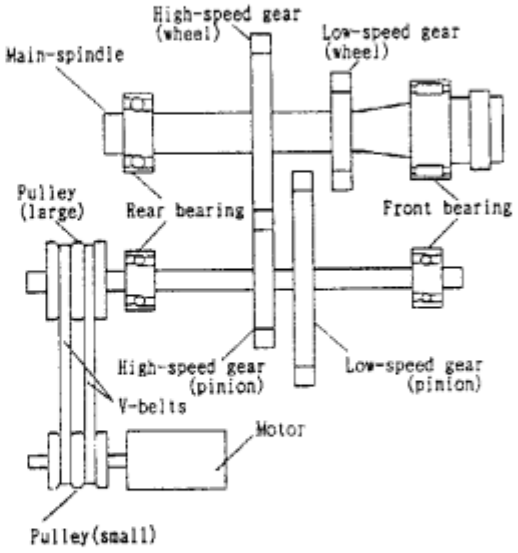

図4.1　設計対象の概略図

４．２　入力

（１）対象モデル

対象モデルは，設計対象の構造と構造から生じる制約，および対象の設計パラメータを表現する。設計対象の構造は，設計対象全体，機能ブロック，部品をそれぞれクラスとして階層的に表現する。クラス階層表現の例を図4.2.1 に示した。対象モデルは，対象の構成要素をconsist_of，構造から生じる制約をconstraint，さらに設計パラメータをparameterとして記述する。図4.2.2に，『プーリ軸システム』の対象モデル定義を示した。

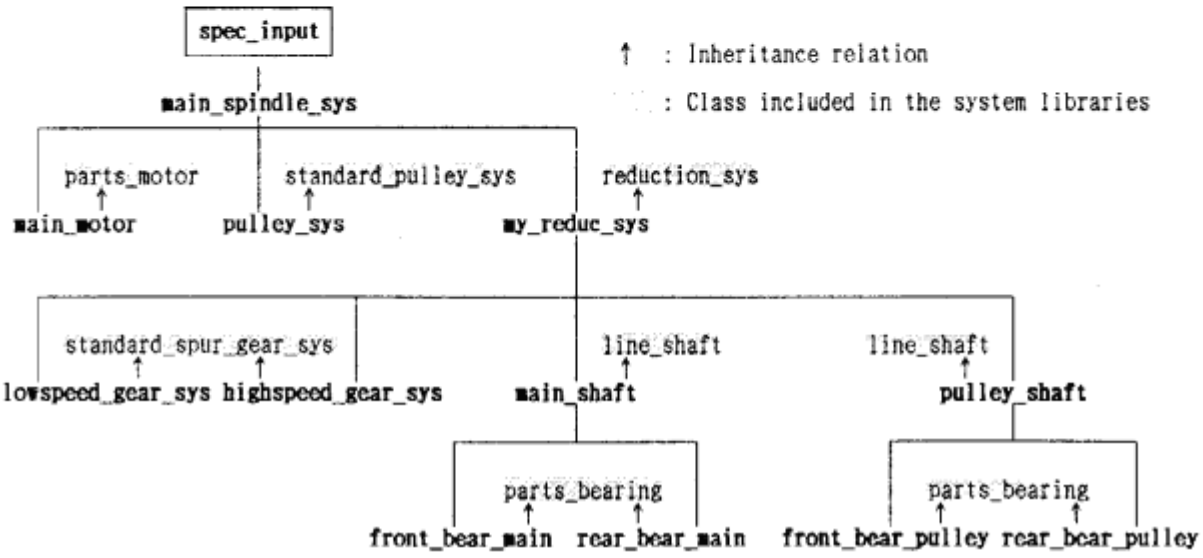

図4.2.1　主軸のクラス階層表現

```
class_name
    pulley_shaft;

inherit_from
    line_shaft;

consist_of
    front_bear_pulley,
    rear_bear_pulley;

parameter
    front_bearing_type,
    rear_bearing_type;

constraint
    #front_bear_pulley!shaft_dia := shaft_dia,
% 'shaft_dia' が設計されたら，その値をクラス
% 'front_bear_pulley' の変数 'shaft_dia' に伝播する。
    #front_bear_pulley!type := front_bearing_type,
    #front_bear_pulley!shaft_rpm := rpm_max,
    #rear_bear_pulley!shaft_dia := shaft_dia,
    #rear_bear_pulley!type := rear_bearing_type,
    #rear_bear_pulley!shaft_rpm := rpm_max;
end.
```

```
class_name
    line_shaft;

parameter
    shaft_dia,
    hole_dia,
    material,
    rpm_max,
    twisting_moment,
    shearing_strength,
    torsion_angle;
end.
```

図4.2.2　クラス『pulley_shaft』に対する対象モデルの定義例

```
class_name
    pulley_shaft;

inherit_from
    line_shaft;

consist_of
    front_bear_pulley,
    rear_bear_pulley;

parameter
    front_bearing_type,
    rear_bearing_type;

constraint
    #front_bear_pulley!type := front_bearing_type,
                    :
    #rear_bear_pulley!shaft_rpm := rpm_max;

design_method
    {[front_bearing_type, rear_bearing_type],
      bearing_mount_search(#result_of_previous_design,
                front_bearing_type, rear_bearing_type)
    };
% 変数 'front_bearing_type' 及び 'rear_bearing_type'
% はクラス 'result_of_previous_design' のメソッド
% 'bearing_mount_serch' により求める
end.


class_name
    line_shaft;

parameter
    shaft_dia,
    hole_dia,
    material,
    rpm_max,
    twisting_moment,
    shearing_strength,
    torsion_angle;

design_method
    {[shaft_dia],
     shaft_dia(#shaft_dia_calc, twisting_moment,
               torsion_angle, shearing_strength,
                          hole_dia, shaft_dia)
    },
% 変数 'shaft_dia' はクラス'shaft_dia_calc'の
% メソッド 'shaft_dia' により求める。ここで,
% 'twisting_moment','torsion_angle',
% 'shearing_strength', 'hole_dia' はこのメソッド
% の入力である。
    {[shearing_strength],
     shearing_strength_search(#material_data_base,
                    material, shearing_strength)
    };

end.
```

図4.2.3　クラス『pulley_shaft』の定義に設計手法に関する知識を付加した例

表4.2　設計手法知識の表現形式の例

表現形式	タイプ
関数，等式 （過去の設計事例の検索） （カタログ，図表検索）	**design_method** パラメータを求める（代替案なし）
連立不等式 （生成範囲の規定） カタログ，図表検索 過去の設計事例の検索	**generator** パラメータを求める（代替案あり）
等式，不等式	**tester** （評価・検証）
数表検索	**adjust_by** （規格化）

（２）設計手法に関する知識

設計手法の制約条件としての表現形式を表4.2 に示した。表現形式としては，関数や等式の他に，不等式や，過去の設計事例およびカタログ・数表の検索といったものを含んでおり，これらの扱いを考慮しなくてはならない。不等式には，生成範囲を規定するものと，解の検証として用いるものがある。過去の設計事例，カタログ，図表の検索は代替案を持つことがあり，生成範囲の規定として用いる不等式と同様に複数の解を生成する可能性があることに注意しなくてはならない。これらの制約条件を扱うために，MECHANICOTでは制約条件をdesign_method，generator，tester，adjust_by の４つのタイプに分類した。

design_method は代替案を持たない関数や図表の検索などを表す。generatorはカタログや図表の検索で代替案を持つ場合や，連立不等式で示された範囲の中から適当な値を生成する設計手法に対して与えられる。testerは，generator で生成した値の検証や，設計パラメータの評価のために用いる不等式などを表す。adjust_by は求めた値を規格値に揃えるためのフィルタである。

設計手法知識は，ある対象モデル内の設計パラメータに対して，それを求めるためのメソッド名と上記のタイプを指定することにより与えられる。制約に付加されたタイプは，制約を解析する際に，設計者の意図を反映しかつ効率良い設計手順を生成するための情報として用いる。図4.2.3 は，『プーリ軸システム』の対象モデルに対して設計手法を定義した例である。

（３）要求仕様

設計要求として，設計対象，入力として与えるパラメータ名（値ではない）と，入力パラメータがどのクラスの設計パラメータに対応するかを入力する。設計対象を示すのにdesign_object，入力パラメータ名を示すのにparameterを用いる。また，入力パラメータが，どのクラスの変数に対応しているのかを示すのにconstraintを用いて表現する。入力パラメータの値は，構築ツールの出力である設計システムが実際に起動される時に具体的な値が入力される。

４．３　設計手順の生成

以上の設計知識を与えられたとして，本節では，制約解析機構による設計手順の生成に関して述べる。設計手順の生成は，コンパイラのデータフロー解析とよく似ており，以下のようにしてなされる。

1) クラス毎にconstraint，generator，tester，adjust_by，design_method にそれぞれサブゴールを割り当てる。constraintの場合には，制約条件１つに対し１つのサブゴールを割り当てる。その他の場合には，メソッド単位で１つのサブゴールとなる。なお，サブゴール名は，ユニークでなければならない。サブゴールの割当ての例を図4.3(a)に示した。
2) クラス毎にサブゴールを入出力パラメータの依存関係を基に，いくつかのゴールにまとめ上げる。ゴール名の付け方はサブゴールの場合と同様である。図4.3(b)にサブゴールからゴールへのまとめあげの例を示した。
3) 次にゴールに対する入出力パラメータの依存関係から，ゴールの呼びだし順を決定する。ゴールの呼びだし順は，階層関係で１つ上のクラスが管理する。例えば，クラスinput_shaft内のゴールの呼びだし順はクラスpulley_shaftが管理する。また，クラスmain_spindle_sysを管理するのは仕様入力クラスである。

```
class_name
    pullley_shaft;
    :
constraint
    #front_bear_pulley!shaft_dia := shaft_dia,
                    ⇒  pulley_shaft_subgoal1
                    :
    #rear_bear_pulley!shaft_rpm := rpm_max;
                    ⇒  pulley_shaft_subgoal6

design_method
    {[front_bearing_type, rear_bearing_type],
     bearing_mount_search(#result_of_privious_design,
                                          . . . . . )
    };              ⇒  pulley_shaft_subgoal7
end.


class_name
    line_shaft;
    :
design_method
    {[shaft_dia],
     shaft_dia(#shaft_dia_calc, twisting_moment,
                                        . . . . . )
    },              ⇒  line_shaft_subgoal1
    {[shearing_strength],
     shearing_strength_search(#material_data_base,
                                        . . . . . )
    };              ⇒  line_shaft_subgoal2
end.
```

図4.3(a)　サブゴールの割当て

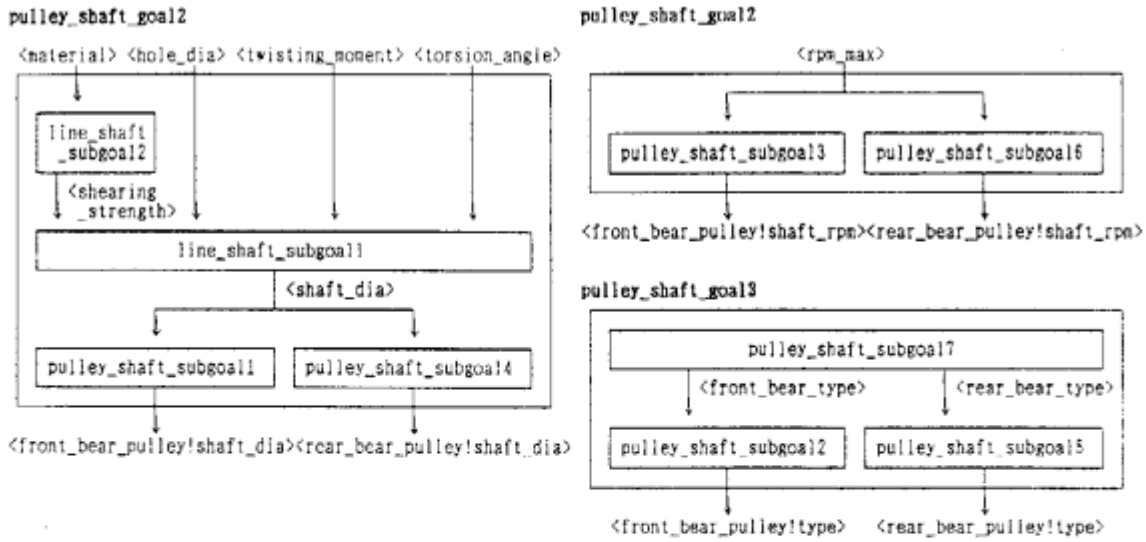

図4.3(b)　サブゴールからゴールへのまとめあげ

なお，制約の依存関係の解析は，一番下のクラスから行われる。すなわち，**図4.2** でいえばクラス**input_shaft** やクラス**front_bear_pulley** から解析が行われる。また，継承関係が存在した場合は，継承関係のある全てのクラスを含んだ制約を解析の対象としている。すなわち，継承がある場合には制約を親子クラス間で階層的に扱うのではなく，フラットに扱うことになる。

4．4　考察

本章の冒頭で述べたように，MECAHNICOTにはいくかの制限事項，検討課題がある。これらの制限は，本例題のような設計対象が不変で設計手法に関する知識も予め良く分かっているルーチン設計の場合にはあまり問題とはならない。しかし，設計対象の構造や設計手法の動的な変更が必要となる場合には，現状のMECHANICOTでは能力が不足する。そこでこれらの課題について今後の見通しを述べておく。

対象モデルについては，設計対象の構造の動的な変更をサポートする対象モデル表現システムFREEDOM[横山 88]を検討中であり，FREEDOM の構築ツールへの応用を考えている。制約解析機構については，MECHANICOTのようなコンパイル型式ではなく，インタプリタ型式として設計手順を動的に生成する方法を検討中である。ただし，制約の動的な変更を許した場合には，設計の進み具合によっては変更されていない制約の再評価を行う必要が生じる。従って，再評価を行う必要のある制約集合をいかに見出すかが問題となる。制約ソルバについては，CAL[坂井 88]，Sup-Inf法に基づく制約ソルバ[大木 88]などの利用を検討中である。

6．おわりに

本報告では，個別性が高い設計作業を支援するために，設計者自らが設計システムを構築できるツールを提供することを目的とし，

- 設計知識を制約とみなすと，設計過程は対象モデルに対する制約充足過程として統一的に捉えられる，
- 設計知識を明示化して管理を容易にし，知識の再利用を図るには，設計知識を対象モデルと設計手法に関する知識に分割して扱うことが必要である，
- 設計者自身が対象モデル，および設計手法知識を容易に入力できるためには，これらの知識が宣言的に入力できることが必要である，

ことから，制約解析機構を利用して宣言的に記述された設計知識から設計手順を生成し，これらの設計知識と問題解決機構である制約ソルバを自動的に結び付けてくれる構築ツールのアーキテクチャを提案した。また，このアーキテクチャに基づいた具体例として，機械のパラメトリック設計を対象としたMECHANICOTについてその概要をのべた。

今後は，4.5 節で述べた課題を中心に，知識獲得や仮説推論の技術の応用とも関連付けながら構築ツールの機能に必要となる要素技術の研究を行って行く予定である。

7．謝辞

本研究を行うにあたり，多くの御助言を頂いた九州大学の長澤講師，東京工業大学の伊藤助手，ならびに，ICOTの設計型エキスパートシステムサブワーキンググループの委員の方々に深く感謝致します。また，設計問題の具体例の提供，問題の解析にあたり多大な御援助を頂いた工業技術院機械技術研究所の小島氏，今村氏に本論文をかりて厚く御礼申し上げます。また，本研究の機会を与えて下さり常に御指導頂いたICOTの淵所長，第５研究室の藤井室長に深く感謝致します。

＜参考文献＞

[Araya 87] : Araya, A., A. and Mittal, S. "Compiling Design Plans from Descriptions of Artifacts and Problem Solving Heuristics." Proc. of IJCAI pp. 552-558(1987)

[Brown 86] : Brown, D., C. and Chandrasekaran, B. "Knowledge and Control for a Mechanical Design Expert System", IEEE Computer pp. 92-100, July(1986)

[Feldman 88] : Feldman, R. " Design of a Dependency-Directed Compiler for Constarint Propagation" , Proc. of 1st International Conference on International & Engineering Application of Artificial Intelligence and Expert Systems, IEA/AIE-88, (1988)

[ICOT編 88] : 昭和62年度 KSS-WG, DES-SWG 報告書, " 設計型エキスパートシステム研究に対するイメージ及び技術課題について" , ICOT Technical Memorandum (To Appear)

[Mittal 86] : Mittal, S. and Dym, C., L. and Morjaria, M. "PRIDE - An Expert System for the Design of Paper Handling System, IEEE Computer, pp. 102-114, July(1986)

[Nagai 88a] : Nagai, Y. "Towards Design Plan Generation for Routine Design Using Knowledge Compilation -Focusing on Constraint Representation and Its Application Mechanism for Mechnical Design-.", ICOT Technical Memorandum, TM-504, (1988)

[Nagai 88b] : Nagai, Y. "Towards an Expert System Architecture for Routine Design -Focusing on Constraint Representation and an Application Mechanism for Mechnical Design-.", ICOT Technical Memorandom, TM-456, (1988)

[Ohsuga 85] : Ohsuga, S. "Conceptual Design of CAD Systems Involving Knowledge Bases." Knowledge Engineering in Computer-Aided Design (Gero, J. S. (ed)), North-Holland, pp. 29-50(1985)

[井上 88] : 井上 克巳, 永井 保夫, 藤井 裕一, 今村 聡, 小島 俊雄, " 工作機械の設計手法の解析－旋盤の回転機能部品の設計, " ICOT Technical Memorandum, TM-494, ICOT(1988)

[上野 85] : 上野 晴樹, " 知識工学入門" , オーム社, pp. 160-187(1985)

[大木 88] : 大木 優, 澤本 潤, 坂根 清和, 藤井 裕一, " Sup-Inf法に基づいた制約論理型プログラミング言語", 日本ソフトウェア科学会第５回大会論文集, C1-1, (1988)

[坂井 88] : 坂井 公, 相場 亮, " CAL: 制約論理プログラミングの理論と実例" , 電子情報通信学会研究会資料, SS87-22, (1988)

[横山 88] : 横山 孝典, " 設計対象記述のための知識表現システム FREEDOMの提案" , 情報処理学会, 知識工学と人工知能研究会資料, 88-60, (1988)