設計問題向け制約指向知識コンパイラにおける制約解析および手順生成について

Towards Constraint Analysis and Plan Generation of Constraint Compiler for Design Problems

永井 保夫*
Yasuo NAGAI
（株）東芝
TOSHIBA Corp.

寺崎 智
Satoshi TERASAKI
（財）新世代コンピュータ技術開発機構
ICOT

This paper deals with constraint analysis and plan generation of constraint compiler for design problems. Constraint compilation is a technique by which knowledge about the domain (constraints, facts, and theories) is stored in declarative form and is applied by constraint-based problem solving mechanisms. A constraint compiler makes existing paths of constraint processing more efficient rather than enabling new paths according to this technique. Considering a design problem, it transforms the input design specifications into the design plan, assuming that the structure of the design object has been determined.

We propose methods of constraint analysis and plan generation of the constraint compiler. First, we describe an actual constraint compiler that applies the concept of constraint compilation to mechanical design, MECHANICOT. In particular, we describe an overall flow of the constraint compiler, concentrating on algorithms that analyze constraints and generate design plan, and the process of design plan generation through constraint compilation. Finally, we give the details of design plan generation using this compiler, giving as an example the problem of gear unit design.

1 はじめに

設計向けの知識ベースシステムを開発していくためには設計対象に依存した知識が大量に必要とされる．従って，設計問題固有の知識表現と，その知識を用いた効率の良い問題解決の方法が設計システム研究におけるキーポイントとなっている．制約という概念を導入することにより，設計知識を宣言的な形式で容易に表現でき，解空間を規定することで余分な探索を減らし問題解決能力を向上させることが期待される．我々は，このような制約表現の利点に注目し，設計問題向け制約問題解決機構の実現を検討している[1]．そのなかで，特に制約を有効利用して問題解決能力を向上させるために必要となる技術課題として，知識変換技術を用いた知識の有効な活用法の一つである知識コンパイラを考えている．

我々は特に制約に着目した知識コンパイラを実現し，これを制約指向知識コンパイラと呼んでいる．制約指向知識コンパイラは，宣言的な知識表現である「制約」を手続き的な知識表現に変換し，変換された知識を問題解決機構が効率的に解釈・実行（解を導出）することを可能にする．設計問題を対象とした場合には，設計タスク用の制約指向知識コンパイラが必要となる．このコンパイラは，宣言的に表現された設計知識から，あらかじめ設定された設計過程モデルに基づいて問題解決機構が効率よく実行可能な設計手順を生成する．

我々は，このような制約指向知識コンパイラを適用した設計支援システム構築ツールMECHANICOTを開発した[2]．MECHANICOTは機械のパラメトリック設計を対象に，設計対象の構造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り，そのアーキテクチャはFig.1に示すような構成となっている．

本稿では，MECHANICOTの制約指向知識コンパイラにおける制約解析とその解析結果に基づいた手順の生成について説明する．まず，制約指向知識コンパイラの処理フローの概要について述べる．さらに，設計過程モデルに基づいた問題解決機構があらかじめ設定されている場合の手順生成について述べる．例として，工作機械のギヤ・ユニット設計[3]を取り上げて制約解析と手順生成について説明する．最後に，制約解析の現在の問題点について考察し，今後の方針についても述べる．

2 制約指向知識コンパイラ

制約指向知識コンパイラは，Fig.2に示される処理フローに従って制約を解析し，設計手順を生成する．ここでは，設計知識を全て制約とみなすことにより，さまざまな形式で表現された知識を統一的に扱うことができる．言い替えれば，ノウハウやヒューリスティックスといった領域に依存した知識も制約と見なし，これらを汎用的に用いられる知識と組合わせて，ある特定目的のために特化した知識に変換する操作を行っているといえる．具体的には，宣言的に表現された設計知識（制約）とその使い方に関する知識を入力として与え，データフロー情報を抽出して，それに基づいて制約間の依存関係を解析し，手順を生成する．これにより，設計知識と問題解決機構との結合機能を提供できるので，設計知識を問題解決機構とは独立に表現できる．

2.1 コンパイラの処理フロー

ソース入力部では，ユーザにより与えられた設計対象定義ならびにライブラリ化された設計対象を構成する部品や機能ブロックが読み込まれる．継承関係解析部では，設計対象を構成する機能ブロックや部品間での継承関係が解析される．制約解析順序決定部では，設計対象の構成要素間の関係が解析され，部分・全体関係ならびに抽象・具体関係により表現された有向非循環グラフが生成される．この有向グラフに従い，解析順序が決定される．制約解析部では，決定された順序に基づき，設計対象を構成する部品および機能ブロック内での制約間の依存関係が解析される．

* 昭和64年1月4日までICOT第5研究室に所属

さらに，機能ブロックと部品間，機能ブロック間ならびに部品間にまたがる制約間の依存関係が解析され，最終的にはシステム全体の制約間の依存関係が求められる．手順生成部では，Fig.3 に示された問題解決機構が設定されているという仮定のもとで，解析された制約間の依存関係に基づき，効率的な問題解決が可能となる設計手順が生成される．

2.2 制約解析および手順生成

制約解析は設計対象の制約間の依存関係を解析し，手順生成ではその結果に基づいて適した制約問題解決（解導出）機構を割り付ける．

ところで，設計問題では取り扱う対象システムが大規模・複雑化し，多量の知識を取り扱うため，なんらかの手段を用いて問題の複雑さを軽減することが必要である．たとえば，対象システムの有する抽象性を利用して，問題をより規模が小さくかつ易しい問題に分解して解くことが考えられる．また，制約ネットワークを考える場合には，対象システムの大規模化に伴い，ネットワーク構造も複雑化し，その取り扱いも困難なものとなる．従って，ネットワーク全体をフラットに取り扱うかわりに階層化し，サブ・ネットワークの組み合せとして取り扱うことが重要になる．この制約ネットワークの階層のレベルには，機能ブロックや部品などが対応する．

我々はこのような階層化された制約ネットワークにおける制約解析に対処するために，解析フェーズを二つに分割し，それぞれをフェーズ１とフェーズ２とした．解析は，対象システムの部分・全体関係ならびに継承関係を示したトリー表現に基づき実行される．フェーズ１はボトム・アップなアプローチをとり，トリー表現の葉から始まって，根に向かって処理が行われる．このフェーズでは，各部品および機能ブロックにおいてデータフロー解析ならびに解析情報の簡略化（縮退）が行われる．フェーズ２は，フェーズ１とは逆にトリーの根から葉に向かって処理される．このフェーズでは，トップ・ダウンに，機能ブロック間，機能ブロックと部品間および部品間にまたがる依存関係を再解析することにより，制約ネットワーク全体の依存関係が求められる．

2.2.1 制約解析

以下では，制約解析アルゴリズムについて説明する．まず，全体の解析アルゴリズムについて述べ，さらに，部品ならびに機能ブロックのデータフロー解析と簡略化アルゴリズムについて示す．

[全体の解析アルゴリズム]

制約ネットワークの依存関係解析の処理効率は，問題を部分問題に分割するとき，各部分問題での処理に要する計算量の和を最小にすることが必要となる．ここでは，全体の依存関係解析は，対象システムの階層構造に基づき分割されている部分問題（構成要素）の解析とみなし，このような問題間の階層関係を崩さずに，上記の計算量について考慮した形で，ボトム・アップな処理とトップ・ダウンな処理を融合することにより実現される．3-7 行目では，ボトム・アップに構成要素としての部品および機能ブロックのデータフロー解析ならびに解析結果の簡略化（縮退）が行われる．具体的には，3-5 行目は，構成要素がアトミックな（構成要素の組み合せとして実現不可能）部品やアトミックでかつ継承される部品の解析処理を示している．6-7 行目は，構成要素が機能ブロックの解析処理である．8-9 行目は，ボトム・アップに解析された依存関係を，あらたにトップ・ダウンに解析することで，制約ネットワーク

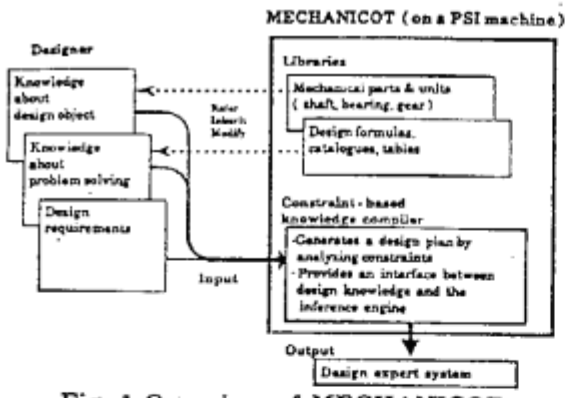


Fig. 1 Overview of MECHANICOT

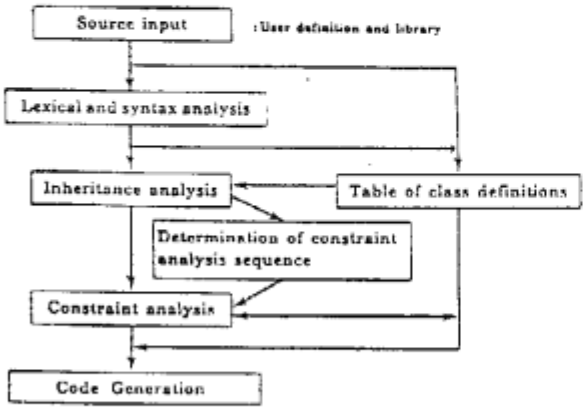


Fig. 2 General Flow of Constraint Compiler

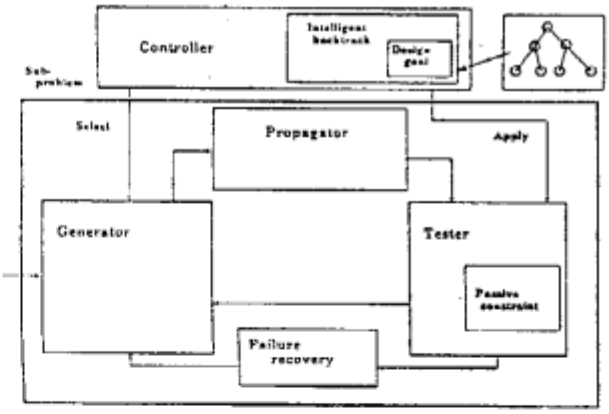


Fig. 3 Problem Solving Model for Constraint Reasoning

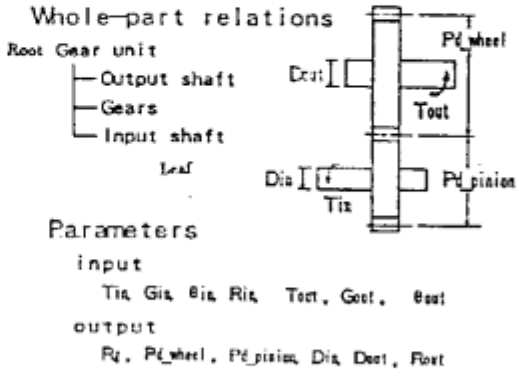


Fig 4 Schematic description of gear unit

全体の依存関係を求めている．

```
1 procedure main
2   begin
3     while （構成要素が終端である）
4       data_flow_analysis_and_merge ;
5     end;
6     for V ∈ 順序付けされた機能ブロックのリスト do
7       data_flow_analysis_for_block (V) ;
8     for V ∈ 設計対象構造のトリー表現 do
9       top_down_analysis(V) ;
10  end;
```

[部品における解析アルゴリズム]
部品のデータフロー解析と簡略化を行なうアルゴリズムは，コンパイラのデータフロー解析アルゴリズムと同様な処理を行なう [5]．まず，部品に定義された制約と実行文を解析して，三つ組表現に変換する．変換された三つ組表現で用いられるオペランド・テーブルの初期化を行なう．この三つ組表現中のオペランドに対して値を付与する値番号法を実行している．三つ組表現中のオペレータにレベルを割り付け，解析によって求められたデータフロー情報に基づき三つ組表現をマージする．
[機能ブロックにおける解析アルゴリズム]
部品のデータフロー解析と簡略化を行なうアルゴリズムも，部品と同様な処理を行なう．まず，継承関係木によって示されているすべての部品及び機能ブロックにおいて簡略化済みのデータフロー表現をあらたに解析し，三つ組表現に変換する．次に，機能ブロック及び継承のある部品における制約記述及び実行文のデータフロー解析を行なう．さらに，継承関係木によって示されているすべての部品及び機能ブロックにおいて簡略化済みのデータフロー表現と自分自身で定義されてデータフロー表現とを併合して，あらたに，データフロー解析を行なう．最後に，この解析に基づいて簡略化された演算要素を三つ組として表現する．

2.2.2 手順生成

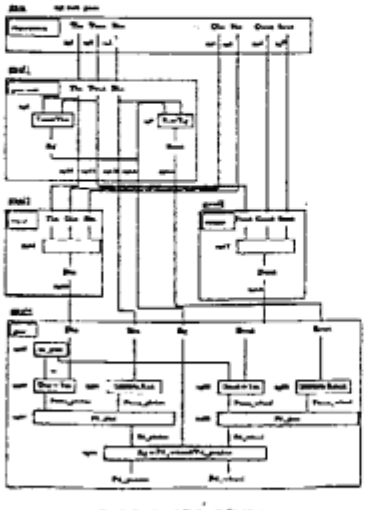
制約解析によって得られた制約間の依存関係は構成要素である部品や機能ブロックでの処理を対象システムの階層構造に従って，トップ・ダウンに行うことにより，手順へと変換される．この手順は問題解決機構を仮定して，効率的な問題解決が実行できるように生成される必要がある．つまり，生成された手順が効率化な問題解決を行なうためには，先程得られた依存関係から，問題を解くのに必要となるタスクを同定・仮定し，問題に適し問題解決戦略に基づいた問題解決器を生成することが必要である [4]．問題解決戦略としては，制約伝播とその制御，失敗回復処理（制約の緩和，アドバイス機構など），最小拘束原理，生成・検査法（階層的生成・検査法），問題分割法（分割統治法）などを考慮しており，特にここでは生成・検査法を前提とした手順生成が行なわれる．

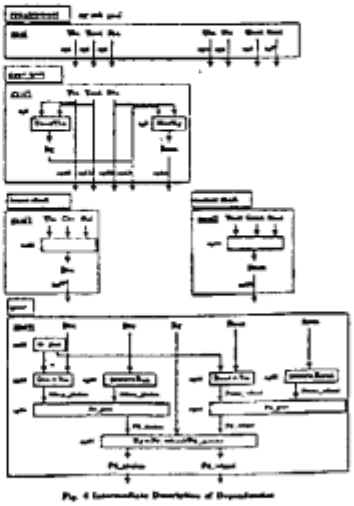
3 ギア・ユニット設計での制約解析および手順生成

パラメトリック設計は，設計仕様として機能および性能仕様が与えられ，ユーザの要求や制約を満足するように構成要素の属性値を決定する問題とみることができる．この問題は設計対象の構造表現（基本的な構成要素集合およびそれらの接続関係）が複数の設計案として与えられ，さらに構成要素についても属性表現されることが前提となっている．さらにこの設計案に基づき対象構造を解析・評価する方法も既知であることが必要となる．我々はギア・ユニット設計をこのようなパラメトリック設計の一例とみなし，以下では，制約指向知識コンパイラにおける制約解析および手順生成を説明する．

3.1 ギア・ユニット設計

ギア・ユニットは，Fig. 4 に示される入力シャフト，出力シャフト，一組のギアから構成される．ギア・ユニット設計では，このような対象の構造の仮定により決定される対象の解析・評価方法に基づき，入力パラメータを満足するように，出力パラメータを決定する．Fig.5 は，対象の構造を仮定することにより，解析・評価用知識として与えられた評価式および設計公式や構成要素間の（接続）関係などを示している．さらに，対象を実現するためには，規格や標準部品といった点にも考慮する必要がある．ギア・ユニットに代表されるパラメトリック設計は，設計において定型化された部分を対象としたルーチン設計であり，設計問題としては十分に定式化可能である．この問題は，「制約」という視点から眺めると制約ネットワークとして定式化可能であるが，制約中の変数に対する値の生成方法に選択肢が存在するとみなすことができる．つまり，問題を解く方法（問題解決戦略）がいくつか存在しているため，その方法により問題解決が効率的であったり，非効率であったりする場合がある．また，問題が制約ネットワークとしては定式化して与えられているにもかかわらず，適した問題解決戦略が設定されていないため，十分な問題解決が行





なわれないことがある．従って，実際の問題に対処するためには，与えられた問題に適した問題解決戦略を提示し，効率的な問題解決を支援する機能が重要である．

3.2 制約解析および手順生成の具体例

まず，Fig. 4 のギア・ユニットを構成する部品の関係が解析され，部分・全体関係を示すトリー表現が生成され，このトリー表現に従って解析順序が決定される．設計対象を構成する部品および機能ブロック内での制約間の依存関係は，決定された順序に基づいて解析される．フェーズ1では，対象構造を示したトリー表現の葉の部分にあたる入力シャフト，出力シャフト，一組のギャ（入力側と出力側）内のデータフロー解析ならびに解析情報の簡略化が行なわれる．次に，トリー表現を根の方向にたどり上方にある機能ブロックであるギア・ユニット内のデータフロー解析ならびに解析情報の簡略化が行なわれる．さらに，ギア・ユニット自身とその構成要素，つまり入力シャフト，出力シャフト，一組のギア（入力側と出力側）間のデータフロー解析ならびにフロー情報の簡略化を行う．このようにして，処理がトリー表現の根に達して，機能ブロックがなくなったところで解析を終了する．

フェーズ2は，フェーズ1とは逆にトリーの根から葉に向かって解析が実行される．このパスでは，トップ・ダウンに，機能ブロックであるシャフトと部品である入力シャフト，出力シャフト，一組のギア（入力側と出力側）間の依存関係ならびにこのような部品間にまたがる依存関係を再解析し，最終的には対象システム全体の依存関係を求める．

Fig. 6 は工作機械のギア・ユニット設計における制約解析の中間段階の依存関係を示している．Fig. 7 はその解析結果に基づき生成された手順を示し，Fig. 8 は最終的に得られる ESP コードを表している．

3.3 考察

現在の制約解析および手順生成処理には，以下のような問題点がある．

- 各制約は対等であり，制約間に優先度は与えられていない．
- 制約に対する役割が予め与えられている．
- 制約の解析が静的であり，問題解決時に起こる動的な制約の処理には対処できない．
- 制約間の依存解析処理ではループ検出をおこなっていない．
- 制約解析時の探索コストの粗い先読み（予測）を用いたゴール・サブゴールのスケジューリング（順序付け）はおこなわれていない．

今後の方針としては、上記の問題点に対処していくとともに，実問題によるコンパイラの性能評価もおこなっていく予定である．

4 むすび

以上，設計問題向き制約指向知識コンパイラにおける制約解析およびその解析結果に基づいた手順の生成について論じた．さらに，現時点での問題点について考察し，今後の方針についても述べた．

謝辞

本研究を行うにあたり有益な助言をいただいたＩＣＯＴ第五研究室の横山孝典氏，井上克巳氏，滝寛和氏に感謝いたします．また，制約解析部のインプリメンテーションにあたりご支援いただいた星氏をはじめとしたＪＩＰＤＥＣの方々に感謝いたします．また，本研究の機会を与えてくださり，常にご指導いただいたＩＣＯＴの淵所長，第五研究室長生駒憲治氏に深く感謝いたします．

参考文献

[1] 永井保夫，滝 寛和，寺崎智，横山孝典，井上克巳：設計問題向けツール・アーキテクチャ，人工知能学会誌，Vol.4 No.3, (1989)

[2] 寺崎 智，永井保夫，横山孝典，井上克巳，堀内英一，滝 寛和：機械設計支援システム構築ツール－ MECHANICOT －，人工知能学会，知識ベースシステム研究会資料，SIG-KBS-8803, (1988)

[3] 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 Memorandumn, TM-494, ICOT, (1988)

[4] 溝口理一郎，山口高平，角所 収：エキスパートシステム構築方法論について，人工知能学会研究会資料，SIG-KBS-8801-2, (1988)

[5] A. V. Aho and J. D. Ullman, Principles of Compiler Design, Addison-Wesley Publishing Company, (1977)

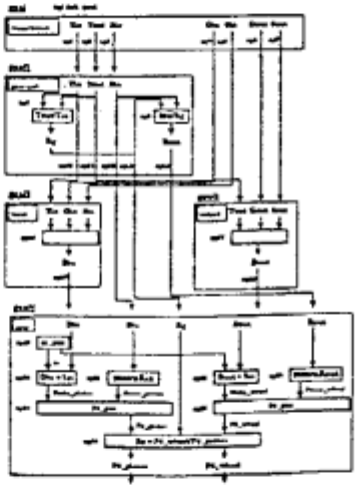


Fig. 7 Generated Design Plan

Fig.8 Design Plan written in the ESP code