設計における思考支援システムの研究*

——知識を用いた制約コンパイラ——

岸 義 樹**

A Study on Thought-Aids System for Conceptual Design
—Constraint Compiler Using Knowledge—

Yoshiki KISHI

This paper describes a method of constraint compiler from functional constraints expressed by linguistic terms to mathematical constraints in order to get a model of physical behavior and rough values of design variables. The functional constraints are composed of two types of knowledge. One is functional knowledge which represents functional relations among design objects. The other is mathematical knowledge which represents physical formula to give relations between function and behavior. The method derives relations of physical behavior from functional knowledge by making reference to mathematical knowledge, and makes mathematical constraints consistent in all design objects. The mathematical constraints are compiled to constraint networks, which are data to Network Specification Solver (NeSS) using the first order approximation. NeSS is developed by the author, and infers values of design variables from the networks gives some specifications. The method is applied to the design simulation of Laser Graphic Device. As a result, the validity of this method is certified.

Key words: constraint compiler, compilation knowledge, functional knowledge, mathematical constraint, constraint network, specification satisfaction

1. 緒 論

設計の知的支援では、設計対象の機能を言語的記号で構成する機能構成形支援¹⁾と、設計対象の挙動を数式や数値で記述する挙動記述形支援²⁾が典型としてある。しかし、前者は言葉の操作が主であるため物理的な挙動を扱うのが不得意であり、後者は数式の操作が主であるため意図的な機能を扱うのが不得意である。

そのため、両者は併用が望ましい。例えば、定性推論によって、挙動を表す数式間の相互関係を定性的に把握し、簡単な機能を生成する試み³⁾などがある。確かに、物理公式などを組み合わせて挙動を表す数式の意味は、公式が自然現象の理解を基にするため、明確である。しかし、設計される人工物は実現すべき意図的な機能を最初から考慮する。しかも、機能には数式に変換できるものと情報の取捨選択にかかわるものがあり、数式の機能的妥当性は後者による。

上述の試みでは、事前に情報の取捨選択を行って機能を定性的な式に変換し、その式を目標に数式の定性

操作を行う。しかし、機能の定性式への還元性⁴⁾(複合機能が数式に変換可能か)、導いた数式の機能に対する適合性(機能と意味的に一致した数式の準備、操作が可能か)、数式が導く設計解の妥当性(与えた仕様値を充たす解が得られるか)などに問題が残る。

そのため、機能構成、挙動記述形支援の併用は別の方法を必要とする。その一つとして、(1)所与の機能的情報の取捨選択により挙動を表す数式を導出して、(2)その数式を因果関係と解探索に適した形式に再編し、(3)仕様値を充足する解を確認する方法がある。これは設計における思考にも適うので、本報はこの方法を採用⁵⁾し、設計向き制約処理の思考支援を目指す。その際、情報の取捨選択法は所与の機能的情報として、数式生成に必要な情報は設計公式として与えるとする。

上に述べた「挙動を表す数式」は設計対象への数式による制約と考えて数式制約と呼び、数式制約の基になる機能的情報は機能制約と呼ぶ。本報で採用した方法は、機能制約を数式制約に翻訳し、解を求める方法と考えて、制約コンパイラ(**constraint compiler, compiler**)と命名し、(1)は数式制約導出、(2)は数

* 原稿受付 平成3年9月3日

** 正 会 員 製品科学研究所(つくば市東1-1-4)

式制約再編, (3)は数式制約充足と呼ぶ。

数式制約充足は仕様充足ソルバ (NeSS)⁶⁾として開発済みである。したがって, 本報は, 数式制約導出 (Constraint Instantiation for Network, CoIN), および数式制約再編 (Constraint Network by Transformation, CoNT) の開発で conspiler を実現し, その性能をレーザグラフィックデバイス (LGD)⁷⁾ の設計シミュレーションで検証する。なお, CoIN と CoNT は数式制約生成 (Constraint Network Generator, CoNG) と総称し, SUN 4 上の K-Prolog で約 3 000 行である。

conspiler の類似法として, 再設計向きの構造と機能の記述に関する研究⁸⁾がある。この方法は固定した数式を陽に記述し, 事前に与えた一定の因果関係で推論するため, 前述の (2), (3) に対応する操作は実現されない。また, conspiler は制約プログラミングの一つでもある。これには, 制約ネットワークを直接解く方法⁹⁾¹⁰⁾, 知識形問題として解く方法¹¹⁾, 非線形式も部分的に扱える方法¹²⁾¹³⁾などがある。しかし, いずれも基本的には値が決まる変数から順に解くため, ある仕様値を満たすように設計変数の値を求める, 変数が仕様に与える影響を知る, 数式制約を生成する等は困難である。これに対し, 本方法は, 機能情報からの数式制約生成, 数式制約再編, 仕様充足による非線形制約の処理など, 上述の困難さに対応可能である。

以下, 2章で conspiler の方法, 3章で機能制約の表現, 4章で数式制約導出, 5章で数式制約再編, 6章でシミュレーション例, 7章で結論を述べる。

2. 制約コンパイラの方法

2.1 機能制約について

機能制約には主に言語で記述した部品に関する情報 (情報を機能知識, 記述する部品を機能部品と呼び, 機能部品は単に部品とも省略) と, 機能知識を数式制約に変換するテンプレートとして使う情報 (数式知識と呼ぶ) がある。以下, 各々の扱いを述べる。

2.1.1 機能知識の扱い

機能知識の表現は, 設計対象の言語的表記の意味を保持するため, 格文法を利用した frame 表現を提案している¹⁴⁾¹⁵⁾。本報もこの表現を使うが, 詳細は文献に譲り, ここでは本報に必要な範囲で概略を述べる。図1に示す LGD のビームスポット結像レンズの機能知識から分かるように, 知識表現は次の5部からなる。

```
(@=var,      [Theme=spot--lens,
              ...,
              Source=light--modulator,
              Goal=display--waist--spot]). } % variable part

(@=theme,    Theme). } % name part

(@=construction, Theme
  ...
  @=input,    [... @=focus, @?data, Theme ...]
  @=output,   ... ). } % nominal-data part

(@=connection, fixture ... ). } % connection part

(@=action,    transmit--path ... ).
(@=action,    transmit--power ... ).
(@=action,    size--beam--spot ... ).
(@=action,    locate--beam--spot ... ). } % action part
...

@=source,     Source, @?at, center
@=goal,       Goal,  @?at, center
@=situation,  [... @=condition, ? ...] ).
```

Fig. 1 Example knowledge for beam spot lens

変数管理部: 記述上の重要語や頻繁に使う用語を変数として管理する特別な frame である。

部品名称部: 記述する部品の名称を示す。

公称知識部: 部品の入出力や寸法, 構成要素などの公称的な情報を記述する。

接続知識部: 部品を構成する要素間の静的な接続関係を記述する (本報では使用しない)。

作用知識部: 部品が動的状態におかれた時に発揮する作用を記述する。

各部は, 丸括弧が囲み, 「@=」で始まる slot を持つ frame で表す。「@=」直後の記号は格と同時に slot 標識も表し, その後に続く「,」が区切る情報は単語と修飾子からなる slot 内容を示す。slot 内容には slot リストも許される。単語には K-Prolog の許す任意記号列が, 修飾子には @?of, @?about, @?at が使える。

slot 内容の先頭には, 機能部品やその作用, 接続関係などの主題である機能部品の一部, 機能部品の動作状態で生じるビームスポットなどの実体 (両者を部品細部と呼ぶ) を主題とする場合には, 「部品細部, @?of, 機能部品」(4.1 節に例がある) と表す。

部品部位などの詳細な記述には @?about や @?at を使う。@?about に続く単語 (power, diameter 等) は主題のタイプを示し, @?at に続く単語は主題が直接使う部品部位を示す。部位を示す単語は, inlet (入力部), center (中央部), outlet (出力部) である。

各部 frame の概略を述べるが, connection slot で始まる接続知識部は使用しないので説明は略す。変数管理部は var slot のみで, 主題は変数情報である。部品名称部は theme slot のみで, 主題は部品である。

公称知識部は部品を主題とする construction slot

を先頭に持つ。本報で使う他の slot としては、焦点距離を示す focus slot などのレンズ特有の slot を総称する input (入力), output (出力) slot がある。focus slot 中の「@?data, xxx」は、「xxx」用データベースの focus データ参照を指示する。

作用知識部は作用名を主題とする action slot を先頭に持つ。本報で使うその他の slot には、拘束条件を示す condition (この slot の用途は 2.1.2 項で述べる。また, situation slot は slot リストを総称する), レーザビームの入力元部品および出力先部品を各々示す source, goal がある。以降, frame は先頭の slot 標識やその主題で指示する場合もある。なお, transmit-path frame は部品の前後関係を示し, 機能知識ごとの作用知識部に必ず記述する必要がある。

2.1.2 数式知識の扱い

機能知識から数式制約を導く方法は二つあると考えられる。一つは機能知識から数式制約を発見する方法であり, 他は機能知識に数式制約指示の情報を書き込む方法である。両者ともに, 機能知識から数式制約を導く設計公式などの数式知識を必要とする。前者は数式制約の一般的発見法という困難な問題がある。それに反し, 後者は数式制約の指示があるので, 指示の煩わしさと制約探索の限定はあるが, 数式知識への当てはめ法を考えればよい。そのため, 本報は数式制約指示法を採用する。

実際の数式制約指示は, 数式制約を部品ごとの拘束条件と考えて, 機能知識の作用知識部にある condition slot に記述する。そして, その指示を解釈し数式制約を導く数式知識を新たに導入する。数式知識は設計公式のほかに, 数式制約再編で数式制約を変形する際に参照される式なども含む。この詳細は 3 章で述べる。

2.2 conspiler の構成

NeSS に対応して conspiler を構成するための, CoNG の入力及び出力情報, 推論プロセスを以下に述べる。

2.2.1 入力および出力情報

CoNG の入力情報は以下の機能制約である。

- ④ 設計対象の機能知識 (ベース)
- ⑤ 数式制約を当てはめる数式知識 (ベース)

CoNG の出力情報は NeSS に必要な情報であるが, NeSS では数式制約の変数を以下の 3 種に分類する。

仕様変数: 設計要求として仕様や性能値を与える変数
設計変数: 設計解を求めるべき寸法や位置などの変数
中間変数: 設計変数から仕様変数を段階的に導く過程で表れる, 上記以外の変数

数学的には, 仕様変数と中間変数は設計変数や中間変数の関数である。上の 3 種の変数は, 次の数式制約,

$$\begin{cases} A = X^2 + c, & c = Y \cdot \tan(Z) + 4 \\ c > Y, & 5 \geq X > 0, & 20 \geq Y > 0, & \pi \geq Z \geq 0 \end{cases}$$

において, 仕様変数を A , 設計変数を X, Y, Z とするなら, 中間変数は c に当たる。なお, 等式による数式制約を「等式を表す数式制約」, 不等式による数式制約を「不等式を表す数式制約」と呼ぶことにする。

そして, NeSS では, 仕様変数から中間変数を経て設計変数へ至る, 数式制約の一次近似に基づく制約ネットワーク上における値修正量の伝播により, 与えた仕様値へ仕様変数の値を許容誤差内で充足させる設計変数の値が, 設計変数の初期値から推論される。その際に必要な情報を次に示す。

- ④ 設計変数の初期値
- ⑤ 仕様変数の仕様値
- ⑥ 変数間の数式制約とその一次近似式
- ⑦ 仕様変数から設計変数に至るネットワーク情報

この中には適当な知識ベースで扱える情報もある。それを決定するため, 推論プロセスを次項で考察する。

2.2.2 推論プロセスの設定

機能知識から数式制約を導く具体的思考プロセスとして, 本報は次の ①～④ を conspiler で実現する。

① まず, 数式知識にある設計公式を機能知識ごとに当てはめて数式制約と変数の候補を抽出した後, 重複や類似によって複数の数式制約に共通な変数を整理し, 全体に一貫する数式制約を導出する (CoIN が推論)。② 次に, 仕様変数と設計変数の候補となる変数を設定し, 仕様変数から設計変数へ至るネットワーク生成を目的に, 数式知識にある数式制約を変形する式を参照しながら数式制約を再編する (CoNT が推論)。③ 最後に, 仕様変数に与える仕様値が設定された後, 仕様充足が実行される (NeSS が推論)。

以下, 各推論の持つモジュール群を示すが, 円内の番号は CoIN, CoNT, NeSS を通した処理手順を示す。

CoIN の推論モジュール群:

- ① 機能制約の定義 (人手によってファイル化)
- ② 機能制約の読み込み
- ③ 数式知識を当てはめた機能知識ごとの数式制約の導出

④ 重複, 類似変数の整理による数式制約の一貫化

CoNT の推論モジュール群:

- ⑤ 仕様変数と設計変数の設定

Fig. 2 Outlook of constraint compiler (conspiler)

- ⑥ 仕様変数から設計変数へ至る数式制約の再編成
- ⑦ ネットワーク情報のファイルへの書出し
ファイル書出しは NeSS が既開発のためである。

NeSS の推論モジュール群：

- ⑧ ネットワーク情報の読み込み
- ⑨ 仕様変数に対する適当な仕様値の付与
- ⑩ ネットワークに従った仕様充足の実行

この推論部は、文献 6) の方法に ⑨ のモジュールを付け加え、拡張している。

2.2.3 入出力情報と推論プロセス

前述したごとく、④ 機能知識は知識ベースとして事前に定義する。⑤ 数式知識は定義に柔軟性を持たせるため、機能知識ごとの特有な情報を持つ数式（固有数式）と汎用的に利用可能な数式（汎用数式）を区別し、固有数式は数式知識部として機能知識に新たに導入し、汎用数式は独立な知識ベースとしてファイル化する。

③ 設計変数の初期値は CoNT が計算するが、この計算は機能知識ごとの初期データを使うため、初期データを定義する初期情報部を機能知識に新しく導入する。④ 仕様変数の仕様値は NeSS で与える。③の一部、変数間関係性は CoNT が最終的に導くが、一次近似式は固有、汎用数式と対応するので数式知識に組み込む。① ネットワークは CoNT が決定する。

以上に述べた conspiler の処理の概要を図 2 に示す。

3. 機能制約の表現

3.1 機能知識の拡張

機能知識のデータ構造は、2.1 節の各部のほか、新しく初期情報部と数式知識部が加わる。既存の知識部では、action frame の condition slot に、数式制約指示が新しい情報として定義される（図 3）。数式知識部の固有知識は汎用数式と共に 3.2 節で述べる。

図 4 に示す初期情報部は database slot のみであり、公称知識部と作用知識部に必要な値を定義する。公称知識部に対しては対応する slot ごとに値を定義し、2.2.1 項に述べた「@?data」で値を参照する。作用知識部では、LGD 設計が部品間距離を必要とするため、記述（Theme）部品と入力元（Source）、出力先（Goal）部品の部位間距離の初期値が定義される。初期値が設定されるのは以下の部品部位間である。

- ① 入力元部品の出力部位—記述部品の入力部位
- ② 記述部品の入力部位—記述部品の中心部位
- ③ 記述部品の中心部位—記述部品の出力部位
- ④ 記述部品の出力部位—出力先部品の入力部位

3.2 数式知識の表現

固有、汎用数式共に表現が二つある。一つは機能知識に当てはめ、数式制約を導くプロトタイプの式（公式と呼ぶ）である。ほかは NeSS に必要な制約ネットワーク生成用の式（メタ式）で、これは公式を変数ごとに解いた式（変形式）と、変形式に対応し制約ネットワーク生成で参照する一次近似式（近似式）からなる。

図 5 はレンズ公式の汎用数式例を示す。図から明らかなように、公式は 2 つのリストを変項とする述語、formula_database で表す。変項の最初のリストは公

Fig. 3 Action frame indicating constraint name

Fig. 4 Example of database frame

Fig. 5 Example of formula knowledge

```

approx_database(      % the 1st definition
  lens--formula.      % formula name
  1.0/F=1.0/A*1.0/B,  % formula expression
  [F=A*B/(A*B)],      % the 1st transformation form
  [A, (F/B)**2],       %  $\partial F/\partial A$ 
  [B, (F/A)**2] ]      %  $\partial F/\partial B$ 
  ).
...
approx_database(      % the n-th definition
  lens--formula.      % formula name
  1.0/F=1.0/A*1.0/B,  % formula expression
  [B=A*F/(A-F)],      % the n-th transformation form
  A>F,                % inequality constraint
  [A, -(B/A)**2],      %  $\partial F/\partial A$ 
  [F, (B/F)**2] ] ).

```

Fig. 6 Example of meta-formula knowledge

式の名前を示す。2番目のリストは、変数を含む公式表現と、3項からなる変数定義に分かれる。

変数定義の第1項は変数名、第2項は数式名が属するframeか公称知識部にある、変数が参照するslot名である。slotの探索はaction frameを優先し、数式知識に合致する変数は機能知識のslotのみで決定する。第3項は変数のタイプである。タイプには、記述部品と第2項のslot中にある隣接部品間の相対距離を示すinterval、値の直接参照を指示するabsolute-* (*はdiameter, length等)がある。

次に、メタ式の汎用数式表現を述べる。この表現は図6に示すように変形式と近似式が一体であり、変項が3個の述語approx_databaseで表す。第1変項は公式名、第2変項は公式そのものである。第3変項はリストをなし、その要素は変形式、不等式条件(省略可能)、近似式の変数ごとの係数である。図6の下部に示す式の第3変項の意味は、変形式が「 $B=A*F/(A-F)$ 」、不等式条件が「 $A>F$ 」、変数Aの係数が「 $-(B/A)**2$ 」、変数Bの係数が「 $(B/F)**2$ 」である。

固有数式では、汎用数式の「formula_database(」をformula slot「(@ = formula, 」, 「approx_database(」をapprox slot「(@ = approx, 」に各々変えて、この順で一つのframeに格納し、数式知識部とする。

4. 数式制約導出

数式制約導出はCoINが推論を行う。

4.1 機能知識ごとの数式制約の具体化

まず、各機能知識のcondition slotの内容である数式制約指示に対応する数式知識を探索する。そして、指示の属する機能知識から数式知識の変数定義に合うframe, slotを探索して、変数の候補となる部品情報を得る。同時に、変数の候補がすべて見つかった公式は数式制約の一つとなる。変数は自動的に変数名を割

```

name = v_70
type = interval
content = (spot, @?of, light--modulator, @?about, diameter, @?at, center),
          (condenser--lens, @?at, center)
detail = (associative, v_36)

name = v_72
type = power
content = (spot--lens, @?about, power), beam--pitch--adapter
detail = (same, v_1)

```

Fig. 7 Example of instantiated variables

り当てられる。変数名は、nnを変数生成順の番号とする「v_nn」である。これが数式制約の具体化である。

図7に示すごとく、生成された変数は4個の属性を持つ。nameは変数を特定する変数名である。typeは公式の説明で言及した変数タイプである。contentは公式の各変数と合致したslot内容で、変数の意味を表す。detailにはオプション情報が書き込まれる。

さて、変数の候補がすべて見つかった公式の左辺の変数以外にも、ほかの変数の関数となり得る変数(NeSSにおける中間変数になる可能性が大きい)、この段階での変数の区別はできないので、単に従属変数と呼んでおく)があれば、従属変数化の推論を実行し、新しい数式制約を生成する。この推論は、実際には中間変数となる変数を設計変数として扱うことで生じる矛盾を排除するためである。

従属変数化は、変数のtype属性がintervalの場合、content属性に次の推論を行う。まず、content属性からslot内容を2個取り出して、transmit--path frameのsource, goal slotの内容を探索し、その2個の間をつなぐ部品を見つける。つなぐ部品が両端を含め3個以上あれば、隣接部品同士のslot内容をcontent属性、intervalをtype属性とする新しい変数群を隣接部品ごとに生成する。そして、それらの変数群のリストを従属変数のdetail属性に設定する。

4.2 重複、類似する変数の整理

変数の重複とは、content属性の内容が変数間で一致することである。変数の類似とは、content属性の内容にある機能部品の変数間で一致することである。type属性がintervalであるために相対値の必要な変数と、powerなどであるための絶対値が必要な変数では、重複、類似する変数の扱いが異なる。絶対値が必要な変数はcontent属性の最初のリストのみを対象とする。相対値が必要な変数はcontent属性の二つのリストを対象とし、その順序は交換できる。

重複、類似変数の整理は2段階で行う。第1段階では、公式の当てはめ中に生成される新変数が生成済み変数とその古い順に比較され、重複や類似を示す情報

```

name = v_70
type = interval
content = (spot, @?of, light--modulator, @?about, diameter, @?at, center),
           (condenser--lens, @?at, center)
           ↓ associative
name = v_36
type = interval
content = (spot, @?of, condenser--lens, @?about, diameter, @?at, center),
           (light--modulator, @?at, center)

name = v_72
type = power
content = (spot--lens, @?about, power), beam--pitch--adapter
           ↓ same
name = v_1
type = power
content = (spot--lens, @?about, power), spot--lens

```

Fig. 8 Variable identification

が新変数の detail 属性に設定される。設定情報は、類似が「associative, v_*」、重複が「same, v_*」である。v_* は一致した生成済み変数で、整理変数と呼ぶ。

機能知識ごとの変数をすべて得た後、整理の第2段階が始まる。重複変数の整理はまず、detail 属性に整理変数を持つ変数の探索（整理変数探索と呼ぶ）を行う。次に、数式制約に含まれる、探索した変数すべてに整理変数による書換え（整理変数書換えと呼ぶ）を行う。

類似変数の整理も重複変数と同様に整理変数探索を行う。そして、探索した変数の type 属性が interval の時は相対値を扱うので、content 属性の修飾子 @?at とこれに続く単語を整理変数が含むなら、整理変数書換えを行う。type 属性が interval 以外では絶対値を扱うので、content 属性の修飾子 @?about とこれに続く単語が整理変数にも含まれるなら、整理変数書換えを行う。

図8に変数の整理例を示す。この結果、設計対象全体に一貫する数式制約（初期数式制約と呼ぶが、NeSS 向きの数式制約に再編されていない）が決定される。

5. 数式制約再編

初期数式制約の NeSS 向きの再編は CoNT が行う。

5.1 仕様変数と設計変数の設定

定数は初期数式制約では変数として扱うが、変数の属性値に database frame を参照する修飾子 @?data を持つので識別できる。一方、仕様変数と設計変数はユーザの任意な指定も可能であるが、本報は以下の仮定を導入し、それらの自動設定を行う。

通常、公式は仕様変数の可能性が高い変数を左辺に

instantiated constraint applying formula form
 $1.0/v_{67}=1.0/v_{65}+1.0/v_{66}$

constraint after variable identification
 $1.0/v_{67}=1.0/v_{65}+1.0/v_{36}$

reconfigured constraint using transformation form
 $v_{36}=v_{65} \cdot v_{67} / (v_{65}-v_{67})$

Fig. 9 Example of constraint generation and change about lens--formula

設定するので、仕様変数は初期数式制約に1回、かつ等式を表す初期数式制約の左辺に表れる変数とする。設計変数は2種類ある。一つは初期数式制約の右辺に1回のみ表れる変数で、これを基底設計変数と呼ぶ。他の変数は中間変数となるはずである。しかし、初期数式制約の再編（再編で得た数式制約を再編数式制約と呼ぶ）の結果、中間変数にもかかわらず、等式を表す再編数式制約の右辺にのみ存在する変数は、派生設計変数と呼ぶ。派生設計変数の決定法は次節に述べる。

5.2 ネットワーク生成のための数式制約再編

数式制約再編は、初期数式制約を参照し、仕様変数から設計変数に至るネットワークのアーキを生成しながら推論が進む。以下、その推論法を述べる。

まず、ネットワークの初期アーキを作る。初期アーキの始端は仕様変数で、仕様変数が左辺にある、等式を表す初期数式制約の右辺の変数が終端となる。初期アーキの生成は全仕様変数に行う。次に、生成済みネットワークを拡張するアーキ（拡張アーキ）を作る。拡張アーキの始端は生成済みアーキの終端の変数で、終端は、始端の変数が左辺にある、等式を表す初期数式制約と合致する変形式の右辺の変数である。

拡張アーキは、合致する変形式の右辺の変数ごとに生成された後、生成済みネットワークに追加されネットワークが拡張する。同時に初期数式制約が変形式で置換されて、以降の置換を受けない数式制約（再編数式制約と呼ぶ）の一つとなる。あるアーキ終端の変数が拡張アーキを生成不能ならば、そのアーキは目的アーキと呼び、以降の拡張アーキの探索対象としない。

全初期数式制約が再編数式制約に変わり、全基底設計変数が目的アーキ終端となれば、ネットワークの生成と数式制約再編は終わる。この時、目的アーキ終端となる中間変数が派生設計変数である。もし、目的アーキ終端とならない基底設計変数があれば、推論のバックトラックが生じる。公式の当てはめで得た数式制約、初期数式制約、再編数式制約の例を図9に示す。

```

cell_name(v_68).
...
original_form(4.0*v_13*v_36/(v_12*v_35)).
approximation_form([v_36, 4.0*v_13/(v_12*v_35)],
    [v_35, -4.0*v_13*v_36/(v_12*v_35**2)], -
    [v_12, 0],
    [v_13, 0]). } % constant
parent_arc([]).
child_arc([v_36, Rv_36, Wv_36, [ ]],
    [v_35, Rv_35, Wv_35, [ ]],
    [v_12, Rv_12, Wv_12, [ ]],
    [v_13, Rv_13, Wv_13, [ ]]).
...
comment_list(variable_type(spec_variable)).

```

$v_{21}=2*v_{22}*\tan(v_{23})$
 $v_{22}=v_{11}+v_{24}$
 $v_{57}=v_{15}+v_{24}/v_6$
 $v_6=v_{10}+v_{11}+v_{76}+v_{77}+v_8+v_9$

v_6 : distance from S1 to Ws
 v_{24} : distance from Ws to Ds
 v_{21} : max display width on Ds
 v_{23} : scanned angle by Bs
 v_{57} : diameter of beam spot
 v_{15} : diameter of beam on S1
spec variable: v_21, v_57

v_8 : distance from S1 to inlet of Ba
 v_9 : distance from inlet to outlet of Ba
 v_{76} : distance from outlet of Ba to Ms
 v_{77} : distance from Ms to inlet of Bs
 v_{10} : distance from inlet to outlet of Bs
 v_{11} : distance from outlet of Bs to Ws
design variable: v_11, 23, 24, 76, 77, 8, 9

Fig. 12 Example of reconfigured constraint

Fig. 10 Example object of spec variable

Fig. 11 Outline of LGD

5.3 NeSSに必要な情報のファイルへの書き出し

ネットワークが確定した後で、database frameを使って設計変数の初期値と定数値を求め、初期値ファイルに書き出す。次に、再編数式制約に対応する近似式を基に、述語形式ノード情報を生成しノードファイルへ、また、不等式を表す数式制約は不等式ファイルへ書き出す。図10は仕様変数のノード情報例を示す。図の述語は、cell_nameが変数名、original_formが再編数式制約、approximation_formが近似式、parent_arcが親ノードへのアーク、child_arcが子ノードへのアーク、comment_listが種々のコメント情報を表す。

6. シミュレーション

LGDは概要を図11に示すように、ビームを発生するレーザ管(Lt)、コンデンサレンズ(Cl)、変調器(Bm)、スポット結像レンズ(Sl)、光路調整器(Ba)、入射ミラー(Ms)、走査器(Bs)、ディジタル化付き画面(Ds)を主な部品とし、ビームは列挙部品順に通過する。その際に、ビームはコンデンサレンズで変調器中心に集光される。更に、スポット結像レンズが走査器と画面の間の適当な位置(Ws)にビームを収束させ、走査器でラスタ走査されるビームスポットを

画面上に作る。

CoNGがLGDの機能制約から導いた等式を表す数式制約は21個、不等式を表す数式制約は2個で、手計算結果と等しい。等式を表す数式制約の例を図12に示す。更に、NeSSにより、ビームスポット径(v_57, 初期値は0.6892 mm)と表示幅(v_21, 初期値は407.28 mm)の仕様値を各々0.7000 mm, 370.00 mm, 許容誤差を0.2%とし、数式制約の仕様充足を試みた結果、5回の充足繰返しで許容誤差内に仕様変数の値が収束している。

7. 結 論

本報は、設計対象の機能制約から、仕様充足ソルバ(NeSS)に必要な数式制約を自動生成する数式制約生成法(CoNG)の提案とシステム開発を行い、その所定動作をレーザグラフィックデバイスの設計シミュレーションで確認すると共に、機能制約から設計解を導く、CoNGとNeSSからなる制約コンパイラの動作も確認した。この結果は、設計対象の機能と挙動を関係づける新しい方法を提供するものである。

今後の課題としては、数式制約の近似式を利用して定性的推論⁴⁾を行い、NeSSの計算制御に有効な情報を提供する方法、機能知識に対する知識操作¹⁴⁾¹⁵⁾の結果を数式制約に反映させる方法などが挙げられる。

参 考 文 献

- 1) 例えば、岸 義樹：知識依存形概念設計支援システムの研究——機械構造の知識表現、精密機械、51, 11 (1985) 2077.
- 2) 例えば、L. Joskowicz：Shape and Function in Mechanical Devices, AAAI-87, (1987) 611.
- 3) B. C. Williams：Interaction-based Invention：Designing Novel Devices from First Principles, AAAI-90, (1990) 349.
- 4) 西田豊明：機能の認識、精密工学会誌、56, 6 (1990) 969.
- 5) 岸 義樹：機能知識を用いた制約式自動生成の一方、1991年度精密工学会春季大会学術講演会講演論文集、

- (1991) 27.
- 6) 岸 義樹：ネットワークによる仕様充足問題解決の一方
法，精密工学会誌，57，4 (1991) 711.
 - 7) 岸 義樹：入力と出力を一体化したレーザグラフィック
デバイスの試作，精密工学会誌，57，7 (1991) 1289.
 - 8) A. Goel & B. Chandrasekaran：Functional Representa-
tion of Design and Redesign Problem Solving, IJCAI-
89, (1989) 1388.
 - 9) G. J. Sussman & G. L. Steele, Jr.：CONSTRAINTS - A
Language for Expressing Almost-Hierarchical Descrip-
tion, Artificial Intelligence, 14, (1980) 1.
 - 10) N. Murtagh & M. Shimura：Constraint-Based Reason-
ing Applied to Engineering Analysis, 人工知能学会誌，
6，5 (1991) 710.
 - 11) 長澤 勲ほか：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，情報処理学会論文誌，27，1 (1986)
112.
 - 12) N. Heintze, S. Michaylov & P. Struckey：CLP(R) and
Some Electrical Engineering Problems, in J.-L. Lassez
(ed.)：Logic Programming, The MIT Press, (1987)
675.
 - 13) D. C. Gossard & D. Serrano：MATHPAK：An Inter-
active Preliminary Design Package, Computer Eng., 3,
(1985) 221.
 - 14) 岸 義樹ほか：機能の理解と生成における思考支援モデ
ル，人工知能学会知識ベースシステム研究会，SIG-KBS-
8803-7 (1988) 59.
 - 15) Y. Kishi：An Architecture to Model Design Intelli-
gence for Creative Conceptual Design, Intelligent CAD,
III, North-Holland, Amsterdam, (1991) 163.

極微粒ダイヤモンドホイールによる超精密研削に関する研究

(財)工作機械技術振興財団(東京都港区芝公園3-5-22, 電話 03-3433-2042)では標記の件について東北大学機械電子工学科庄司克雄教授ほか2名に研究依頼していたが、このほどその成果がまとまったので発表された。

OA 機器や AV 機器などに代表されるように、製品に要求される機能はますます高度化し、それに伴って、部品加工の高精度化、高効率低コスト化に対する要求が重要な課題となっている。しかもこれらいわゆるメカトロニクス製品に使われる材料は、セラミックスなど高硬度、高ぜい性、高融点の難加工材料が多く、レーザ加工や放電加工など物理的、化学的加工が不向きであるため、専ら砥粒による機械加工に依存しているのが現状である。

砥粒加工には、ラッピングやポリシングなど遊離砥粒を用いる方法と砥石のような固定砥粒による研削加工法の2種類がある。遊離砥粒法は、

(1) 比較的簡単な機械で高精度の加工が行える

(2) 工具の管理が容易である

などの利点が多い。しかし一方では、

(1) 作業性が悪い

(2) 加工能率が低い

(3) 自動化に不向きである

(4) 形状精度が劣る

など、固定砥粒法に比べて欠点も多く指摘される。特に円筒面の鏡面加工においては、形状加工精度の点で遊離砥粒法には問題が多く、砥石による研削加工に頼らざるを得ない。

一般に機械加工においては、加工精度を優先させれば加工能率は犠牲にせざるを得ない。この互いに背反する

要求を同時に満たす一つの解決策はラッピングやポリシングなどの遊離砥粒法から、微粒ないし極微粒ホイールを用いた研削やホーニング、超仕上げなどの固定砥粒法への転換であろう。最近では、製造技術の進歩に伴い、サブミクロン粒径の超微粒ホイールの製造も可能になり、極微粒もしくは超微粒ホイールに対する加工技術者の期待はますます大きくなっている。

ところでこのような硬ぜい材料の鏡面研削加工においては、粒径10 μm 以下の極微粒ダイヤモンドホイールが使われる。しかし極微粒ホイールでは、1 μm のオーダーで砥粒突出し量をコントロールするツルージングとドレッシングが必要であり、現在ではその技術はまだ確立されていない。そこで砥石メーカー側では、結合剤中に固定潤滑剤を(フィラー)充てん剤として混入し、工作物と結合剤のラッピング作用による研削焼けやホイールの損傷を極力防止する方策が取られているが、完全に成功しているとはいえない。

また鏡面研削では、1回あたりの切込み量を十分に小さくしなければならない。特に微粒ホイールを用いた場合は、その砥粒突出し量が μm オーダーであるため、切込み量が大きくなると、ホイールのボンド剤と工作物が接触して、研削抵抗や研削温度の上昇を招く。したがって鏡面研削加工において、極微粒ホイールに対応した工作機械技術の研究開発を早急に行う必要がある。

さらに、切込みがサブミクロンオーダーの研削では、これまでの粗粒ホイールによる研削とは砥粒の摩耗形態、目詰まりの進行などの相違が予想される。安定した切れ味を持続して保つには、研削過程におけるホイール作業面の変化を解明し、ホイールの切れ味を維持する研削条件を確立しなければならない。(p. 1022 へ続く)