オブジェクト指向設計知識・データ表現システム†

中島裕生††* 馬場富男†† 加藤亨††

近年，機械設計の分野では，より柔軟かつ高効率な知識表現，および大容量データを利用する知識ベースシステムが求められている．これまで知識表現手段として，特に固有のモデルベースとなる知識表現に向いていることから，オブジェクト指向技術がよく利用されている．一方，機械設計支援を対象としたエキスパートシステムを構築する際に新しく有用な概念として，「属性モデル」がある．本論文では，この機械設計における属性モデル表現のための要件を明らかにし，それを満足するよう開発した，設計対象の属性モデル表現，および設計アクティビティの属性表現のためのオブジェクト指向知識・データ表現システムについて述べている．本システムは，オブジェクト指向言語の上に知識表現レベルを構成しているスーパフレーム言語を持つ．本言語で記述すると，オブジェクトクラスタと呼ばれる一群のオブジェクトを生成し，それらはフレーム・オブジェクト，アトリビュート・オブジェクト，アイテム・オブジェクトから構成される．これに加え，アトリビュートタイプ・オブジェクトをシステムは定義しており，これらによって属性記述，属性値タイプの共通利用，属性の可搬性，属性間の関係などをオブジェクトレベルで実現している．

1. はじめに

近年，機械設計の分野では，より柔軟かつ高効率な知識表現，および大容量データを利用する知識ベースシステムが求められている．これまで知識表現手段として，特に固有のモデルベースとなる知識表現に向いていることから，オブジェクト指向技術がよく利用されている．ここで言うオブジェクト指向技術とは，人工知能で使われるフレームと，ソフトウェア工学でのオブジェクトについて総称しており，これらは過去においてそれぞれの領域で独立に概念や言語が作成され応用システムに使われてきたものである[1]．ここでは，フレームについて作成されている言語をフレーム表現言語，ソフトウェア工学でのオブジェクトについて作成されている言語をオブジェクト指向言語と呼ぶ．フレーム表現言語の例として，CRL[2]，KEE[3] があり，オブジェクト指向言語の例として，Flavors[4]，C++[5] がある．現在では，知識表現単位としてのフレームと，ソフトウェア工学でのオブジェクトを統一的に捕え，オブジェクト指向でソフトウェアを開発する姿勢が取られている[1] が，現実のエキスパートシステム開発においては，採用する言語が，AI ツールが提供する，CRL，KEE などの知識表現能力が比較的豊富なフレーム表現言語か，あるいは一般の言語として提供されている，知識表現能力として比較的劣る Flavors，C++などのオブジェクト指向言語かによって実際の開発方法が異なる．オブジェクト指向言語を使用する理由として，実行速度，汎用性，コストでの優位性と，さらに最近では二次記憶装置での永続オブジェクトを実現するオブジェクト指向データベース利用可能などが挙げられる．本論文が対象としている機械設計の分野において，このオブジェクト指向言語を使用した知識表現について，これまで必ずしも十分研究されているとは言いがたい．ここでは，知識表現手段として Flavors というオブジェクト指向言語を使用した，機械設計に必要な知識表現方法について述べる．

一方，機械設計支援を対象としたエキスパートシステムを構築する際に新しく有用な概念として，「属性モデル」「属性モデリング」が提唱されている[6],[7]．

この概念は，設計対象にはそれを表現するのに本質的かつ必要十分な属性集合が存在し，設計はその属性の操作によって実現されるというものである．ここで言う属性とは単に値だけでなく幾何形状情報も含んでいる．さらに，機械設計において，設計対象に本質的な属性集合と対応する幾何実体を洗い出すことが可能であり，これを行うことにより設計支援システムの貢献度合いを上げるというものである．このような概念にしたがった対象のモデル化を属性モデリング，そのモデルを属性モデルと呼ぶ．筆者らも，すでに実用化されている油圧回路設計支援エキスパートシステム[8] の中でこうした概念を使用した知識処理を実現している．この開発経験から，一般の機械設計支援エキスパートシステム構築を考える場合，現実の設計対象を

† An Object-Oriented Design Knowledge and Data Representation System by YUSEI NAKASHIMA, TOMIO BABA and TOHRU KATO (Development Team, Technology & Research Center, Kayaba Industry Co., Ltd.).

†† カヤバ工業(株)技術研究所開発チーム
* 現在 ニチメンデータシステム(株)知能情報システム部
Artificial Intelligence & Information Engineering Department, Nichimen Data Systems Corp.

属性集合と捕える際には，属性の種類と量の多さ，属性値の種類の多さという量的な側面と，共通属性，固有属性に基づいた属性モデリングの質的な側面を満足する言語が必要であるという認識が生まれた．

属性モデリングという用途にオブジェクト指向言語が提供するオブジェクト，クラスラチス，継承，インスタンス変数，メソッドといった簡潔な機能を単純に適用すると，属性モデルという知識を分解してそのフレームワークに当てはめることになる．すると，大規模な知識ベースになったときに，その理解および保守が困難になる．そこでオブジェクト指向言語レベルの上に属性モデルを表現する知識表現レベルを設け，その知識表現にオブジェクト・クラスタおよびクラスタ間での汎化階層構造という考えに基づくスーパフレーム言語と呼ぶものを開発した．この言語は機械設計支援エキスパートシステム構築シェル：MAGIC[9] の設計対象知識表現，および設計アクティビティ知識表現のための言語となっている．ここでは，属性モデル表現に求められる要件を明らかにし，それを実現するオブジェクト指向言語上に作られた知識表現言語スーパフレーム言語について述べる．

2. オブジェクト指向言語と属性モデル

2.1 オブジェクト指向言語

オブジェクト指向言語の特徴を整理すると以下のようになる．オブジェクト指向言語は，オブジェクト，クラス，継承，およびメッセージパッシングに基づいている．オブジェクトは，データと手続きとを組み合わせた実体であり，オブジェクトはローカルメモリー（変数）と手続き（メソッド）とを持っている．オブジェクト間の相互作用は，メッセージパッシングによってなされる．オブジェクトは，複数のクラスの形で組織化され，クラスに属するオブジェクトは，クラス・インスタンスと呼ばれる．複数のクラスは継承関係によって結ぶことができる．ある種のオブジェクト指向言語は，メタクラス機能を与える[10]．メタクラスは複数のクラスのクラスである．

従来の伝統的なアプローチと比べると，オブジェクト指向言語は，データの抽象化を提供し，情報の構造化，およびプログラムのモジュラリティを増し，継承機能によって冗長な部分を廃し，よりロバスト性の高いソフトウェアという利点を持つ．

次に Symbolics 社の Lisp 上のオブジェクト指向言語 Flavors の概要について述べる[11]．詳細は，文献 4) を参照されたい．Flavors では，Flavor と呼ばれるユーザ定義データタイプに属するデータ（これを Flavor のインスタンスと呼ぶ）をオブジェクトとみなす．Flavor そのものは DEFFLAVOR というマクロで定義され，一般に言われるクラスに相当する．Flavor に対して，メソッドと呼ばれる手続きを DEFMETHOD というマクロで定義する．クラスの階層構造について述べるとき，より上位のクラスをスーパクラス，より下位のクラスをサブクラスと呼ぶ．継承というのは，一般的に，スーパクラスに相当する上位概念が持つ性質，機能を，下位のサブクラスに相当する概念によって受け継がれることをさす．また，多重継承というのは，スーパクラスを複数個持つことが許されるようなクラス間の継承方法である．Flavors では，スーパクラスに対して上位概念という見方でなく，部品 (component) という見方をする．したがって，新しい Flavor を作るときには，他の Flavor を利用し混合 (mixin) するという考え方をする．この Flavor の混合が上に述べた継承に相当する．継承の対象になるのは，インスタンス変数とメソッドである．この多重継承によるモジュール化が Flavors の特徴の一つになっている．

2.2 属性モデル表現に求められる要件

機械設計において，製品開発の企画段階で作成される品質-機能展開の品質特性，機能特性という形で設計対象を表現する属性を得ることができる．また，幾何形状の属性抽出に関して，面分と面分，面分と稜線などの接続関係，つまり位相構造に注目した属性抽出技法が提案されている[12]．こうした属性を分類すると，要求仕様である目的属性，品質属性，機能属性，実現構造属性，幾何形状属性などになる[9]．また，設計計算式（例えば，機械構造の理論解析式，実験式）などは，属性変数間の拘束関係であり，属性変数には述語表現（例えば，near-member，これは属性変数が数値列に含まれる数値に近いものを取るという述語）や不等式表現などの定性的・定量的制限を加える属性変数修飾子がある．こうしたことに加え，フィールドで実績のある製品と属性変数値との表，属性変数および設計式の使い方に関するヒューリスティックスを表現する属性制約などが，実際，設計ノウハウと現場で言われる知識である．

設計対象の属性例として，動力伝達用平歯車の歯形属性を**表1**に示す．これは幾何形状属性という分類の“動力伝達用平歯車歯形諸元”という属性種類であり，

表 1 動力伝達用平歯車の歯形属性
Table 1 A feature of tooth profile for spur gear.

モジュール
歯 数
工具圧力角
歯先円直径
歯底円直径
転位係数
工具歯形歯先丸味半径
内 径
トッピング歯先円方向
トッピング半径方向

属性変数がモジュール，歯数，工具圧力角，歯先円直径，歯底円直径，転位係数，工具歯形歯先丸味半径，内径，トッピング歯先円方向，トッピング半径方向などから構成されている．これらは具体的な属性値を持つが，個々の属性の値という見方と同時に，特定のモジュール属性値での歯数属性値，工具圧力角属性値という属性変数間および属性値間の関係という見方も重要である．さらに，歯先円直径，歯底円直径などは異なる属性名でありながら，単位は mm，データタイプは数値タイプ，さらに双方とも中心と半径で決まる円の式による軌跡を持つという共通の属性値のタイプを持つ．歯プロフィールは，インボリュート曲線によりユニークに決定されるが，この際のインボリュート曲線式は，属性変数間の拘束関係である．また，使用できる工具から工具歯形歯先丸味半径が決まることから，属性変数修飾子が存在する．歯車の属性は表1の歯形属性だけでなく，目的属性としての歯車性能，機能属性としての面圧・曲げ強度，品質属性としての製作誤差・精度，および干渉などがある．一方，歯形を製品の一部として持つスプラインシャフト，歯車を部品として持つ遊星歯車減速機などの設計属性は，これまで述べた歯車の属性の幾つかを再利用することができる．

こうした属性モデルを表現するための要件を整理すると次のようになる．

（1） 属性種類，属性変数，属性値，属性変数修飾子，属性制約条件，属性変数間拘束関係などが記述可能なこと．以下，単に属性という場合には，属性記述に必要なこれらを含む．

（2） ある属性変数が持つ属性値のタイプを規定できることと，それらを任意の属性変数が利用できること．

（3） 設計対象を表現する個々の属性は，共通に利用される共通属性と，対象にユニークな個別属性がある．共通利用の属性とは，属性の可搬性を意味している．

（4） 分類された属性の内容，さらに属性間の関係も重要である．

（5） 多量の属性を効率良く管理できること．

これらに加え，属性モデル表現のフレームワークに次の項目，

（6） 属性の操作を行う設計アクティビティ自体の属性表現と，それへの設計対象を表現する属性集合の設計アクティビティの組み込み．

が可能になると，設計対象，設計アクティビティに関する知識ベースフレームワークとして条件が完備するものと考えられる．

2.3 オブジェクト指向言語による知識表現例

オブジェクト指向言語は，比較的低いレベルの抽象化機能を与える．この低いレベルの抽象化は，現実世界を反映した知識とその表現との間でセマンティック・ギャップが存在することを意味し，このことがオブジェクト指向プログラミング言語を直接，知識表現に使用する際の問題点として指摘されている[13]．直接にオブジェクトを使って，現実の知識表現を行おうとすると，オブジェクト指向プログラミング言語が提供する，オブジェクト，クラス階層，継承等のフレームワークに当てはめるために知識の分解を行わなければならない．分解された知識から構成される知識ベースは，知識のグラニュラリティが小さいことにより，理解するのが困難で，保守性が悪くなるという知識工学上の問題点がある．こうした傾向は大規模知識ベースになるほど著しい．

こうした例を**図1**に示す．図1では，機械設計での設計アクティビティに関する知識をオブジェクト表現したものである．設計アクティビティに関する知識をオブジェクト指向言語 Flavors で直接表現してみる．まず，“シリンダブロック設計”というアクティビティを一つのオブジェクトで表現し，そのアクティビティの属性変数である has-design-sub-activity をインスタンス変数にして，“シリンダブロック設計”アクティビティのさらに詳細な設計アクティビティを値として記述している．この値は，さらに構造体を持ったオブジェクト名を指しており，その例が“シリンダブロックボア間最大応力検討”というオブジェクトである．このオブジェクトは，“シリンダブロックボア間

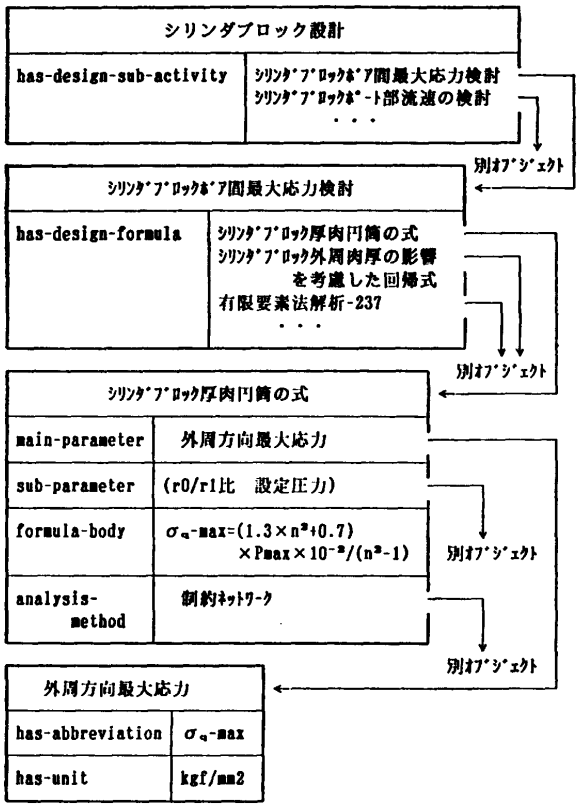

図 1 オブジェクトによる設計知識表現例
Fig. 1 An example of design knowledge representation by objects.

最大応力検討”というアクティビティを一つのオブジェクトで表現し，それは属性変数 has-design-formula をインスタンス変数にして，その値には設計式名が記述してある．この設計式名も，実はその下に示すオブジェクトである．このように“シリンダブロック設計”というアクティビティ表現のために，一連のオブジェクト群には，オブジェクト間の関係としての変数解釈手続きが要求される．例えば has-design-sub-activity，has-design-formula という二つのインスタンス変数が，それぞれが所属するオブジェクトと他のオブジェクトとの関係を示していることを，所属しているオブジェクトに手続きとして解釈できるよう記述しておく必要がある．こうしたオブジェクト間の関係だけでなく，インスタンス変数に対する手続きの付加，インスタンス変数が持つ値に対するメタ表現（例えば，数値データに対する単位など）に対して，Flavor というオブジェクト単体では低いレベルにある．知識ベースの規模が大きくなると，こうした表現に対する手続きの散在は，保守性を大きく損なう．

3. オブジェクト指向設計知識・データ表現システム

3.1 オブジェクトクラスタの考え

知識表現に際したオブジェクト指向言語の限界を先に述べたが，これはオブジェクト指向言語を放棄することではなく，オブジェクト指向言語の持つ2.1節で述べた特徴を知識表現の基礎として，その上に知識表現レベルを設けることで，健全な基礎を持った設計知識表現言語の構築が可能と思われる．知識表現レベルは，2.2節で述べた属性モデルを表現するための要件を満足すると同時に，こうした知識に関連した処理に関する手続きを自動的に完備する必要がある．この知識表現レベルは，その下部に存在するオブジェクト指向言語の抽象レイヤを構成していることになる．

オブジェクトクラスタは，知識表現レベルでの知識表現単位であり，これをスーパフレームと呼ぶ．オブジェクトクラスタは，設計対象，設計アクティビティに関する知識をモデリングするクラス群をグループ化し，さらにこれらを統括するクラスオブジェクト（ルートオブジェクトであり，フレームと呼ぶ）を定義し，クラスの具体化（インスタンシエーション）を行うという特徴を持っている．このオブジェクトクラスタの概念図を**図2**に示す．図に示されるように，オブジェクトクラスタを構成するオブジェクトは，次の3種類からなる．

（1） フレーム・オブジェクト
（2） アトリビュート・オブジェクト
（3） アイテム・オブジェクト

また，オブジェクトクラスタとして組み込まれないが，（2），（3）のオブジェクトに影響を及ぼすものと

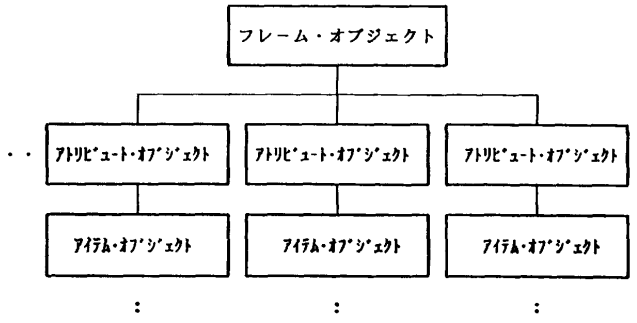

図 2 オブジェクトクラスタの構成図
Fig. 2 Configuration of Object Cluster.

して，アトリビュートタイプ・オブジェクトがある．(2)，(3)の各オブジェクトはオブジェクトクラスタの構成員となることで，次のような制約を課される．

(a) フレーム・オブジェクトは独立オブジェクトであり，アトリビュート・オブジェクト，およびアイテム・オブジェクトは，フレーム・オブジェクトに対する従属オブジェクトである．

(b) 従属オブジェクトとは，その存在が他のオブジェクトの存在に依存するものをいい，ある一つのオブジェクトによって所有されているものである．

(c) したがって，従属オブジェクトはその所有者であるオブジェクトが存在しなければ，作られることができない．

(d) さらに，フレーム・オブジェクトが消されると，その従属オブジェクトであるアトリビュート・オブジェクト，アイテム・オブジェクトも消去されねばならない．

オブジェクトクラスタの各オブジェクトは，次のように BNF で定義される．

```
<Frame Object>
    ::=<Frame Object Root>
       (<Linked Dependent>*),
<Linked Dependent>
    ::=<Attribute Object>
       <Sub-Linked Dependent>,
<Sub-Linked Dependent>
    ::=(<Item Object>*).
```

上の定義で，＊は任意の数だけ続けられることを示すメタシンボルである．フレーム・オブジェクトは，特別なインスタンス・オブジェクトを持ち，ルートオブジェクトと呼ばれる．ルートオブジェクトは，いくつもの従属オブジェクトを結合している．ルートオブジェクトに直接結ばれている従属オブジェクトは，アトリビュート・オブジェクトと呼ばれ，さらにこれは別の従属オブジェクトと結ばれている．このアトリビュート・オブジェクトに結ばれている従属オブジェクトをアイテム・オブジェクトと呼ぶ．一つのアトリビュート・オブジェクトには，いくつものアイテム・オブジェクトを定義できる．アトリビュート・オブジェクト，アイテム・オブジェクトの数は，一つのオブジェクトクラスタ内で事実上無制限である．こうした一つのフレーム・オブジェクトに結合された複数のアトリビュート・オブジェクト，および複数のアイテム・オブジェクトを総称してスーパフレームと呼ぶ．また，スーパフレームと他のスーパフレームの間においては，多重継承を許した階層構造を取ることができる．これを**図3**に示す．さらに，図3の⇒印で示すように，あるアトリビュートのアイテムのならびをキーにみなした検索をスーパフレーム内で行うことや，フレームからアトリビュートを経て直接アイテムにアクセスする手続きを完備する．

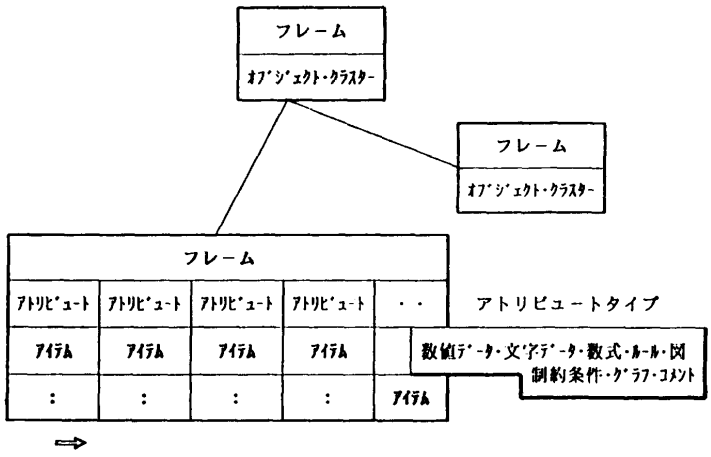

図 3 スーパフレーム言語で定義されるクラスタ構造とその階層化
Fig. 3 Object clusters and its hierarchy defined by super-frame language.

こうしたオブジェクトクラスタによる属性モデル表現には，**表2**のような対応が可能である．

表 2 設計属性とオブジェクトクラスタの対応
Table 2 Objects corresponding to design features.

属性種類	フレーム・オブジェクト
属性変数	アトリビュート・オブジェクト
属性変数修飾子	アトリビュート・オブジェクト
属性値	アイテム・オブジェクト
属性値タイプ	アトリビュート・オブジェクト アトリビュートタイプ・オブジェクト
属性制約条件	アトリビュート・オブジェクト
属性変数間拘束関係	アトリビュート・オブジェクト
属性間の関係	アイテム・オブジェクトをキーにした検索 フレーム，アトリビュートをキーにした検索

3.2 オブジェクトクラスタの考えを実現するスーパフレーム言語

オブジェクトクラスタつまりスーパフレームをスーパフレーム言語で記述すると次のようになる.

```
(defframe<frame-name>
        (<attribute-object>*)
        (<component-frame>*)
        <data-order-identifier>::=
          :attribute-order|:item-order
          (<item-object>*)
        <:init-plist>)
```

frame-name で与えられた名前のオブジェクトを,<attribute-object>*で与えられた定義済みのアトリビュート・オブジェクトを生成し,そのインスタンス変数に設定する. defframe で生成されるフレーム・オブジェクトは,アトリビュート・オブジェクトのほかに,<component-frame>*で与えられる他のスーパフレームを継承し,さらに基本フレームというオブジェクトを継承する. 基本フレームは,フレーム自体の補助的な情報を保持するためのオブジェクトである. 基本フレームに含まれる情報は,別名称,フレーム・グループ,コメント,アイテム・オブジェクトをキーにした検索機構などがある. <:init-plist>は,この補助的な情報をフレーム生成時に与える.

アトリビュート・オブジェクトをスーパフレーム言語で記述すると,次のようになる.

```
(defattribute<attribute-name>
            <attribute-type>
            <:init-plist>)
```

各フレームに設定されるアトリビュートを定義する. これによって,別途定義するアトリビュートタイプ指定子を含んだ<attribute-name>のオブジェクトとメソッドが生成される. <:init-plist>は,アトリビュート・オブジェクトに対し与えたい補助のデータを記述する.

アトリビュートタイプ指定子は,スーパフレームが提供,あるいはユーザが定義するアトリビュートタイプに関する情報を含んだアトリビュートタイプ・オブジェクトの存在をもとに記述する. そのアトリビュートタイプは,次のように記述する.

```
(defattribute-flavor<attribute-type>
                    {<instance-variable>*})
```

また,アトリビュートタイプ・オブジェクトに対して,次のようなメソッドが定義できる.

```
(defattribute-method<method-name>
                    <attribute-type>
                    {<arg-list>}<body>)
                    <body>::=<lisp-form>.
```

アトリビュートタイプ・オブジェクトに定義されたメソッドと同種のメソッドが,そのアトリビュートタイプの利用を宣言したアトリビュート・オブジェクト生成時に,アトリビュート・オブジェクトに対し生成される. 各アトリビュートタイプの種類に応じて固有のメソッドがユーザにより定義されるが,言語が提供する基本的なメソッドは,データタイプのチェックである.

アイテム・オブジェクトは,データをアイテム単位で扱うために,次のようなオブジェクト構造をしている.

```
<item-object>::=<index-slot><link-slot>
                <data-slot>,
<index-slot>::=<index-instance-variable>
                <number>,
<link-slot>::=<link-instance-variable>
              (<item-object>*),
<data-slot>::=<data-instance-variable>
              <data>.
```

インデックス・スロットはアトリビュート・オブジェクトに結合されるアイテム・オブジェクト群の中で何番目に位置するのかを示している. リンクスロットは,そのオブジェクトと他のオブジェクトの参照に使用される. データスロットは,蓄えられるべきデータを保存する. アイテム・オブジェクトは,一度生成されると,検索パフォーマンス向上のためハッシュテーブルに登録される.

アイテム・オブジェクトが扱うデータは,アイテム・オブジェクトが所属するアトリビュートのアトリビュートタイプでなければならない. スーパフレーム言語がデフォルトで提供する基本的なタイプを以下に示す.

```
<attribute-type>
  ::=<string>|; 文字列
    <multi-string>|; 文字列のリスト
    <multi-string-list>|; 文字列のリストのリスト
    <combination-string>|; 組み合わせ可能文字列
    <lisp-form>|; リスプ形式に従った構造
    <formula-form>|; 代数式形式に従った構造
```

<rule-form>|; ルール形式に従った構造
<constraint-form>|; 制約形式に従った構造
<number>|; 数値
<multi-number>|; 数値のリスト
<range-number>|; 長さ2の数値リスト
<multi-range-number>|; 長さ2の数値リストのリスト
<graph>|; 長さ2の数値リストのリストおよびグラフ表現に必要な補助変数
<string-index-multi-graph>|; 文字インデックスを持つ複数グラフ
<value-index-multi-graph>.; 数値インデックスを持つ複数グラフ

上記のアトリビュートタイプ群は，それぞれに特有の手続きを有している．例えば，文字列のリストなどは，データタイプチェック以外に，リスト内文字列検索手続きが設けられている．また，グラフタイプの場合には，データタイプチェック，グラフ表示，離散値データの補間，補間曲線の計算などの手続きが備わっている．

4. スーパフレーム言語による知識表現およびデータ表現

ここではスーパフレーム言語によって，設計アクティビティ自体の属性表現と，設計対象を表現する属性モデルの表現がどのように行われるかを図1の設計アクティビティについて見てみる．まず，アトリビュートタイプ・オブジェクトの定義を**図4**のように行う．この際，そのアトリビュートタイプに持たせたい手続きを図4の(b)，(g)，(i)に示すように，アトリビュートタイプ・オブジェクトに対するメソッドとして定義する．次に，**図5**に示すように上記アトリビュートタイプをつかった図1に関連したアトリビュート・オブジェクトの定義を行っておく．すると，**図6**(a)に示す二つのフレーム定義（シリンダブロック設計，シリンダブロック設計式）によって，(b)の構造のスーパフレームが得られる．つまり図1のオブジェクト群が二つのスーパフレームによって表現された．また，フィールドで実績のある製品の属性変数とその値という情報，あるいは現在設計対象となっている製品の属性変数とその値という情報は，アトリビュート・オブジェクトに属性変数を，アイテム・オブジェクトに属性値を対応する形で，図6(c)のようなスーパフレームで示される．

実際には，エンドユーザがスーパフレーム言語で直接記述する場合は少なく，システムが提供する専用の開発環境のもとでウィンドウ上のメニューにしたがってスーパフレーム内容を構築していく．

アトリビュートタイプとその手続きの付加例として，図4(f)，(g)，図5の(g)，(h)にある「制約フォーム」について詳しく見ると以下のようになる．図4(f)の「制約フォーム」を定義するアトリビュートタイプ・オブジェクトは，属性変数間の拘束関係を制約ネットワークの形に展開し，属性変数値がネットワーク上を伝播し，未決定属性変数を決定するために導入される．このため「制約フォーム」を定義するアトリビュートタイプ・オブジェクトに付加される手続

```
(defattribute-flavor :string)
(a)ｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄの定義.

(defattribute-method type-check
                     :string () (type-check-body))
(b)ｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄに対して、そのﾀｲﾌﾟをﾁｪｯｸするﾒｿｯﾄﾞを定義.
   同種のﾀｲﾌﾟﾁｪｯｸのﾒｿｯﾄﾞは、以下のすべてのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟについても定義する（略）.

(defattribute-flavor :multi-string)
(c)ﾏﾙﾁｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄの定義.

(defattribute-flavor :multi-string-list)
(d)ﾏﾙﾁｽﾄﾘﾝｸﾞﾘｽﾄのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄの定義.

(defattribute-flavor :number)
(e)数値のｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄの定義.

(defattribute-flavor :constraint-form)
(f)制約ﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄ定義.

(defattribute-method execute-constraints-compiler
                     :constraint-form () (execute-body))
(g)制約ﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄに対し、制約ｺﾝﾊﾟｲﾗｰを起動し、
   制約ﾈｯﾄﾜｰｸ上へのﾃﾞｰﾀ展開をさせるﾒｿｯﾄﾞを定義.

(defattribute-flavor :rule-form)
(h)ﾙｰﾙﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄ定義.

(defattribute-method execute-rule-compiler :rule-form ()
                     (execute-body))
(i)ﾙｰﾙﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟｵﾌﾞｼﾞｪｸﾄに対しﾙｰﾙｺﾝﾊﾟｲﾗｰを起動し、前向き推論
   ﾙｰﾙの場合にﾘｰﾄﾈｯﾄﾜｰｸ上にﾙｰﾙﾄﾘｶﾞｰを展開させるﾒｿｯﾄﾞを定義.
```

図4　アトリビュートタイプオブジェクトの例
Fig. 4　An example of attribute type object.

```
(defattribute 'has-design-sub-activity :string
      '(:has-design-sub-activity-japanese-attribute-name "検討項目"))
(a)ｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"検討項目"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'has-design-formula :multi-string
      '(:has-design-formula-japanese-attribute-name "関係設計式"))
(b)ﾏﾙﾁｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"関係設計式"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'has-reference-data :string
      '(:has-reference-data-japanese-attribute-name "参照(実績ﾃﾞｰﾀ"))
(c)ｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"参照(実績ﾃﾞｰﾀ"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'formula-name-for-ｼﾘﾝﾀﾞﾌﾞﾛｯｸ設計 :string
      '(:formula-name-for-ｼﾘﾝﾀﾞﾌﾞﾛｯｸ設計-japanese-attribute-name "設計式名"))
(d)ｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"設計式名"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'main-parameter :multi-string
      '(:main-parameter-japanese-attribute-name "主ﾊﾟﾗﾒｰﾀ"))
(e)ﾏﾙﾁｽﾄﾘﾝｸﾞのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"主ﾊﾟﾗﾒｰﾀ"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'sub-parameter :multi-string-list
      '(:sub-parameter-japanese-attribute-name "従ﾊﾟﾗﾒｰﾀ"))
(f)ﾏﾙﾁｽﾄﾘﾝｸﾞﾘｽﾄのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"従ﾊﾟﾗﾒｰﾀ"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'has-formula-body :constraint-form
      '(:has-formula-body-japanese-attribute-name "設計式本体"))
(g)制約ﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"設計式本体"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'has-constraint-for-formula-body :constraint-form
      '(:has-constraint-for-formula-body-japanese-attribute-name
            "設計式に対する制約条件"))
(h)制約ﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"設計式に対する制約条件"の
   ｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義

(defattribute 'has-application :rule-form
      '(:has-application-japanese-attribute-name "適用種別"))
(i)ﾙｰﾙﾌｫｰﾑのｱﾄﾘﾋﾞｭｰﾄﾀｲﾌﾟ指定子を持つ"適用種別"のｱﾄﾘﾋﾞｭｰﾄｵﾌﾞｼﾞｪｸﾄの定義
```

図 5 層性表現に対するアトリビュートオブジェクトの例
Fig. 5 An example of attribute object for feature representation.

きは，データとして属性変数間拘束関係をチェックし，そのデータをアクセスして制約コンパイラを起動し，制約ネットワーク上に展開するメソッドであり，これを図 4 (g)のように定義する．

機械設計支援エキスパートシステム[4]に，このスーパフレーム言語を使用した例として，一つのスーパフレーム内に，ヒューリスティックスを表現するルール，数式で代表される設計式，設計式を構成する設計パラメータに関した制約，設計式の適用範囲，設計プロセスで参照する数値データ・グラフ，それに対するコメント，関連した図などが表現できている．また，こうしたスーパフレームを，設計アクティビティというより抽象度の高い概念記述をしたスーパフレーム，さらには特定製品の設計という現実のタスクを記述したスーパフレームに多重継承させることによって，機械設計における設計アクティビティの属性表現，設計対象の属性モデル表現がスーパフレーム言語によって達成されることを確認した．さらに，実際の設計プロセスに対応した形で，大量の設計属性型知識，属性のデータが，一つのスーパフレーム内に表現でき，それを視覚的にテーブルのイメージで捕えられるシステムの開発環境によって開発生産性，および保守性が向上している．

5. 考　　察

本論文の目標である，機械設計における設計対象の属性モデル表現，設計アクティビティの属性表現の観点から，本システムと他のシステムとの関連を述べる．設計対象の属性表現において，概念上これ以上分解できない属性（ここではプリミティブ属性と呼ぶ）というものを考えてみる．“表面粗さ”という機械設計ではよく頻出する属性がある．この属性はプリミティブ属性と考えられるが，この属性自体，加工により与えられ，計測によりその値が決められるものである以上知識として表現するにはそう単純ではなく，(属性，属性値) のペアで表現の完結はしない．“疲労強度”，“平行度”，“同軸度”，“硬さ”等々ほとんどすべての機械設計上の属性はプリミティブと考えられても，同様のことが言える．こうした知識をオブジェクト指向言語で表現する場合，Flavors であれば（インスタンス変数，変数値）のインスタンス変数に機械設計上の属性を直接マッピングしようとすると，それだけでは表現記述が完結せず，他のオブジェクトによる記述追加を行い，2.3 節で述べたようなクラスの数に比べオブジェクト間の関係記述コード量が多くなり保守性に影響を与える．このことは Flavors 以外のオブジェクト指向言語，例えば CLOS (Common Lisp Object System)[14] で提供される（スロット，スロット値）を使用しても，機械設計上の属性をスロットに直接マッピングしようとすると同じことになる (CLOS を使用すれば，値に対する制約記述，データタイプに対するメソッド記述は可能になる)．またインスタンス変数，スロットでなく，機械設計属性をオブジェクトそのものへマッピングしようとしても，属性をマッピングしたオブジェクトを構造化するなど，属性モデル表現のための要件をオブジェクト指向言語が陽に提供しているわけでないので

```
(defframe ｼﾘﾝﾀﾞﾌﾞﾛｯｸ設計
        (検討項目 関係設計式 参照実績ﾃﾞｰﾀ)
        (ｼﾘﾝﾀﾞﾌﾞﾛｯｸ設計式)
        :attribute-order
        ((ｼﾘﾝﾀﾞﾌﾞﾛｯｸﾎﾞｱ間最大応力検討 ｼﾘﾝﾀﾞﾌﾞﾛｯｸﾎﾟｰﾄ部流速の検討 ・・)
         (ｼﾘﾝﾀﾞﾌﾞﾛｯｸ厚肉円筒の式 ｼﾘﾝﾀﾞﾌﾞﾛｯｸ外周肉厚の影響を考慮した回帰式
          ・・))
(defframe ｼﾘﾝﾀﾞﾌﾞﾛｯｸ設計式
        (設計式名 主ﾊﾟﾗﾒｰﾀ 従ﾊﾟﾗﾒｰﾀ 設計式本体 設計式に対する制約条件
         適用種別) ()
        :attribute-order
        ((ｼﾘﾝﾀﾞﾌﾞﾛｯｸ厚肉円筒の式 ・・) (外周方向最大応力 σq-max kgf/mm2)
         (((r0/r1比 n) (設定圧力 Pmax kgf/cm2)))
          (σq-max=(1.3×n²+0.7)×Pmax×10⁻²/(n²-1)) (・・) (・・)))
```

(a)

シリンダブロック設計		
検討項目	関係設計式	参照実績ﾃﾞｰﾀ
ｼﾘﾝﾀﾞﾌﾞﾛｯｸ ﾎﾞｱ間最大応力検討	ｼﾘﾝﾀﾞﾌﾞﾛｯｸ厚肉円筒の式 ｼﾘﾝﾀﾞﾌﾞﾛｯｸ外周肉厚の影響を考慮した回帰式 有限要素法解析-237	製品実績表
ｼﾘﾝﾀﾞﾌﾞﾛｯｸ ﾎﾟｰﾄ部流速の検討	・・・	・・・
・・・		

多重継承関係

シリンダブロック設計式					
設計式名	主ﾊﾟﾗﾒｰﾀ	従ﾊﾟﾗﾒｰﾀ	設計式本体	設計式に対する制約条件	適用種別
ｼﾘﾝﾀﾞﾌﾞﾛｯｸ厚肉円筒の式 ・・	外周方向最大応力 σq-max kgf/mm2	・・	・・ ↓ σq-max=(1.3×n²+0.7)×Pmax×10⁻²/(n²-1) →((r0/r1比 n) (設定圧力 Pmax kgf/cm2))	σq-max < 材料耐久限	適用ﾙｰﾙ1

(b)

シリンダブロック諸元					
製品形式	外周方向最大応力	ﾎﾟｰﾄ専有率	ﾎﾟｰﾄ幅比	ﾋﾟｽﾄﾝﾎﾞｱ深さ	(属性変数)
(形式名)	(応力値)	(専有率値)	(幅比値)	(深さ)	(属性値)

(c)

図 6 スーパフレーム言語による設計知識表現例
Fig. 6 An example of design knowledge representation defined by super-frame language.

困難である.

それに対し，本システムでは基本的に機械設計上の一つの属性をフレーム・オブジェクト，アトリビュート・オブジェクト，アトリビュートタイプ・オブジェクトの三つのオブジェクトで表現しようとするものである．この方が実際の機械設計上の属性を表現する能力がインスタンス変数，スロットで記述するより豊富である．さらに本システムは一つの属性を表現する複数のオブジェクトを構造化し，さらに複数の属性についてもオブジェクトクラスタの考えで構造化することができる．また，比較的表現能力が劣るが，パフォーマンスの良いC++オブジェクト指向言語を使用する場合にも，本システムで提案した考えで機械設計上の属性表現が可能と思われる．

6. おわりに

本論文では，機械設計における設計対象の属性モデル表現，および設計アクティビティの属性表現のためのオブジェクト指向設計知識・データ表現システムについて述べた．本システムは，オブジェクト指向言語の上に知識表現レベルを構成しているスーパフレーム言語を持つ．本言語で記述された場合，実際に生成されるオブジェクトの構成は，オブジェクトクラスタの考えに沿ったものである．提示した属性モデル表現のための要件のうち，「多量の属性を効率良く管理できること」を除く要件について，スーパフレーム言語は満足した．「多量の属性を効率良く管理できること」について，オブジェクト指向設計知識・データ表現システムは，オブジェクト指向データベースの利用を念頭においている．これによって，多量の属性管理の問題解決を意図するものである．

本システムは Symbolics 3600 シリーズ，XL シリーズ上で稼働する．実際に本システムを実用してみて，設計属性の操作が容易なため，設計知識，データの追加・維持管理を設計者自身に実施してもらうなどの保守性向上が見られた．一方，問題点として，属性モデリングの際，物理的な物の特徴に対応する属性名を付けるのが困難な場合があること，および設計者間で属性名に対するコンセンサスを取る必要があることがわかった．

謝辞 九州工業大学の長澤教授，および東京工業大学の伊藤氏には属性モデリングについて御教授いただいた．これらの方々に深くお礼申し上げる．

参考文献

1) 溝口：オブジェクト指向概念による知識表現言語，情報処理，Vol. 29, No. 4, pp. 382-389 (1988).
2) Wright, J. M. and Fox, M. S.: SRL: Schema Representation Language, Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA (1983).
3) KEE Software Development System, Core Reference Manual, Intellicorp., CA (1986).
4) Symbolics Common Lisp-Language Concepts Manual, Symbolics, Cambridge, MA (1988).
5) Stroustrup, B.: *THE C++ Programming Language*, Addison-Wesley, Reading, MA (1986).
6) 高度技術化に対応する機械製図システムの標準化のための調査研究報告書，日本設計工学会 (1989).
7) 伊藤：CAD システムの機能評価とこれからを見通す技術，PIXEL，10月～12月号，図形情報処理センター (1989).
8) Nakashima, Y. and Baba, T.: OHCS: Hydraulic Circuit Design Assistant, *Proc. of IAAI-89*, AAAI, pp. 124-130 (1989).
9) 中島，田原，加藤：機械設計支援エキスパートシステム構築シェル：MAGIC，情報処理学会論文誌，Vol. 31, No. 11, pp. 1572-1580 (1990).
10) Moon, D. A.: The Common Lisp Object-Oriented Programming Language Standard, in Kim, W. (ed.), *Object-Oriented Concepts, Databases, and Applications*, pp. 49-78, ACM Press, USA (1989).
11) 梅村，大里：Lisp 上のオブジェクト指向プログラミング，情報処理，Vol. 29, No. 4, pp. 303-309 (1988).
12) 福田：溶接構造設計エキスパートシステム，コンピュートロール，Vol. 25, pp. 112-117, コロナ社，東京 (1989).
13) Goffaux, L. and Mathonet, R.: A Technique for Customizing Object-Oriented Knowledge Representation Systems, with an Application to Network Problem Management, *Proc. of IJCAI-89*, pp. 97-103 (1989).
14) Keene, S. E.: *Object-Oriented Programming in Common Lisp*, Addison-Wesley, Reading, MA (1988).

(平成2年3月 6 日受付)
(平成3年2月 12日採録)

中島 裕生（正会員）

1951年生．1976年東京工業大学大学院材料科学専攻修士課程修了．同年カヤバ工業(株)技術研究所勤務．1990年よりニチメンデータシステム(株)勤務．現在，知能情報システム部長．知識工学応用システム，オブジェクト指向データベース応用開発に従事．著書「計算力学とAI」(養賢堂，共著)．AAAI，人工知能学会，OR学会各会員．

馬場 富男（正会員）

1952年生．1975年東京農工大学機械工学科卒業．同年カヤバ工業(株)入社．現在，カヤバ工業(株)技術研究所勤務．エキスパートシステム開発に従事．

加藤 亨（正会員）

1963年生．1987年日本電子専門学校国際コンピュータ技術研究科卒業．同年カヤバ工業(株)入社．現在，カヤバ工業(株)技術研究所勤務．エキスパートシステム開発に従事．