制約に基づくオブジェクト指向知識表現システム†

横山孝典†† 佐塚秀人†††

近年，エキスパートシステムによる設計型問題への対応が盛んになっている．設計型問題は要求仕様を満足する対象モデルを生成する問題と見なすことができ，単純なフレームやオブジェクト指向による対象モデル表現では，効率的な問題解決が困難である．この問題に対応するため，本論文ではオブジェクト指向に制約概念を導入した知識表現システムを提案する．本システムの知識表現はオブジェクト指向を基本に，設計型問題を解く場合に重要な知識である，属性や構造，オブジェクトの型などに関する制約を宣言的に記述可能とするとともに，抽象・具体関係と単なる機能の包含関係とを明確に区別したクラス階層表現や，問題解決過程で動的に変更可能なクラス・インスタンス関係等の拡張を行う．そして，オブジェクトを常に制約充足状態に保つ機能を提供することにより，制約を満足する属性値の決定や，制約充足可能なクラスの探索，トップダウン詳細化支援などの，設計型問題に有効な問題解決機能を実現する．そして，以上の機能を有する設計対象モデル表現システム FREEDOM のプロトタイプを作成し，その有効性を確認した．

1. はじめに

近年様々な分野でエキスパートシステムが開発されている．エキスパートシステムで扱う知識は問題の対象に関する知識と問題解決の方法に関する知識に大別できるが，従来のシステムではこれらのうち問題解決に関する知識のみを重視していた．ところが最近では，対象に関する知識の有効活用がシステムの能力向上に不可欠なことが認識され，これを対象モデルとして表現することが重要視されている[1),2)]．対象モデルの表現には，構造や属性の表現に優れたフレームやオブジェクト指向が適していると言われている．

対象モデルの観点からとらえると，診断，制御等の解析型問題はあらかじめ固定的な対象モデルが与えられ，それを解析する問題である．したがって対象モデルは構造や属性を静的に表現すればよく，従来のフレームやオブジェクト指向による表現で特に問題はなかった．

これに対して最近エキスパートシステムによる対応が期待されている設計型問題は，設計解が対象モデルで表現され，要求仕様を満足する対象モデルを生成することがその目的である[1)]．このため解の候補となるモデルの取捨選択や，修正，詳細化等，動的に対象モデルを生成するための機能が重要視される．また効率的な対象モデル生成を実現するには，対象に関する知識，特に構造や部品の種類，属性などに関する制約条件を有効に活用するための機能が必要である．

制約を問題解決に利用する場合，従来のフレームやオブジェクト指向に基づくシステムでは，付加手続きやメソッドにより手続き的に記述しなければならず，理解のしやすさや柔軟性などの点で問題が残る．そこで，制約を宣言的に記述でき，問題解決に利用できるシステムが求められている．

本研究の目的は，オブジェクト指向と制約を有機的に統合化することにより，設計型問題における対象モデル表現に適した知識表現システムを実現することである．

そのためここでは，オブジェクトに関する制約を宣言的に表現可能とするとともに，オブジェクトが制約充足状態を保持する機能を提供する．ここでオブジェクトは制約充足のため，属性値のみでなく，それが属するクラスやその構造を変更することができる．これにより，知識を制約表現するのみで，オブジェクト自身が能動的に働き，問題解決を進めることが可能となる．

以下では，まずオブジェクト指向と制約の融合の必要性について述べ，次に制約の導入によりオブジェクト指向を拡張した知識表現と，制約に基づく問題解決を提案する．そして，提案した知識表現および問題解決機能を有する，対象モデル表現システムFREEDOMについて説明し，最後に今後の課題について述べる．

2. オブジェクト指向と制約の融合

2.1 オブジェクト指向に基づく対象モデル表現

対象モデルには対象の構造や部品の種類，属性，それらの関係や動作など，問題解決上必要な知識を記述する．従来，対象の構造や属性の記述に適した知識表

† A Constraint-Based and Object-Oriented Knowledge Representation System by TAKANORI YOKOYAMA (Institute for New Generation Computer Technology) and HIDETO SAZUKA (NTT Data Communication Systems Corp.).

†† (財)新世代コンピュータ技術開発機構
††† NTT データ通信(株)

現としてフレーム[3]が広く用いられ，最近ではプログラミング言語の分野で発達してきたオブジェクト指向プログラミング[4),5)]と融合，統合化されてきた．

オブジェクト指向表現を利用すれば，部品をそれぞれオブジェクトに割り当て，全体・部分関係などの構造をオブジェクト間の関係で，属性を変数（スロット）で表すことができ，対象モデルを自然に表現できる．また，クラスの継承機能を利用して概念間の抽象・具体関係を表す“is_a”関係（“a_kind_of”関係）を定義することができる．

設計型問題では対象に関する一般的な知識の利用が重要になる．例えば電子回路を設計するには，基本回路の構成や，回路定数と特性の関係式などの知識を有効に活用する必要がある．従来このような知識は設計手続きに埋め込まれた形で表現されることが多かったが，これでは知識の有効活用ができず，システムの柔軟性を損なうことになる．

そこでここでは，具体的な対象を表すもののみでなく，対象一般に関する知識を含めて対象モデルとして表現する．そして前者をオブジェクト指向におけるインスタンス，後者をクラスに対応させる．

2.2 制約の導入

以上のようにオブジェクト指向知識表現は対象モデルの記述に適しているが，従来の枠組みでは設計型問題に対応するには機能不足の点がある．

一般のオブジェクト指向言語では，オブジェクトのインスタンス変数（あるいはスロット）に他のオブジェクトへのポイントを格納することにより，複数のオブジェクトを関係付ける．しかし，その関係の意味を直接宣言的に表すことはできず，メソッドを用いて手続き的に表現しなければならない．これはオブジェクトの属性間の関係についても同様である．しかし手続き的表現は，ユーザが直感的に理解しにくく，特定の利用法に依存した形でしか関係を表現できないため柔軟性に欠けるという問題がある．

そこで，知識の利用法や処理手順に依存しない宣言的記述法である，制約表現を導入することが考えられる．人工知能の分野では以前より，対象の属性間の制約に着目した知識表現が研究されてきた[6)~8)]．これらは制約を宣言的に記述できる点で優れているが，構造的な記述ができないため，大規模な対象の記述は難しい．

このため，構造記述に優れたオブジェクト指向に制約表現を導入することにより，両者の長所を生かすことが考えられ，属性（インスタンス変数）間の関係記述に制約を導入したシステムが報告されている[9)~12)]．しかしこれらは制約を属性値の決定にしか利用できず，処理の多くは手続き的に記述する必要がある．

本論文では，問題解決全体を制約概念でとらえた知識表現システムを提案し，この問題の解決を図る．そのため，単に従来のオブジェクト指向の枠内に制約を導入するのではなく，制約に基づく立場からオブジェクト指向を見直し，自然な両者の融合を目指す．

3. オブジェクト指向表現の拡張

3.1 オブジェクトと制約

従来の多くのシステムでは制約はオブジェクトとは独立の概念として扱われてきた．しかし，複数のオブジェクト間に何らかの関係（制約）が存在する時は，その関係を生ずるための上位の概念が存在すると考えられ，それをオブジェクトで表現できる．したがってここではすべての制約はいずれかのオブジェクトに属するものとして扱う．

例えば図1の例で，オブジェクト「初段増幅器」の属性「出力抵抗」とオブジェクト「次段増幅器」の属性「入力抵抗」が等しいという制約は，それらを含む上位のオブジェクト「2段増幅器」に記述する．

オブジェクトに関する制約は属性値に関するもののみでなく，オブジェクトの型に関する制約や構造上の制約も存在する．ここではクラスを指定することで型に関する制約を表現し，指定したクラスおよび“is_a”関係においてその下位クラスであれば制約を満たすものとする．

オブジェクトの構造の表現においては部分・全体関係，いわゆる“part_of”関係が重要であるが，この関係には四角形における4つの辺のように，部分が全体にとって必要不可欠な構成要素である場合と，本棚におけるその中の本のように必ずしもそうでない場合がある．ここでは前者を“consists_of”関係と呼んで，構造上の制約として扱うことにする．

また，エキスパートシステムで扱うヒューリスティ

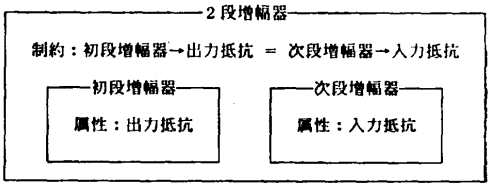

図1 オブジェクト間の制約の記述
Fig. 1 Constraint among objects.

ックスの中には制約表現が適したものが少なくない．そこでインスタンスに動的に制約を追加，削除する機能を提供し，ヒューリスティックスとしての制約をオブジェクトに付加するのみで，制約充足処理により問題解決を進めることを可能とする．

3.2 クラス・インスタンス関係

一般にオブジェクト指向言語では，クラスは特定の種類の対象物に共通な性質を記述したもの，インスタンスは具体的な個々の対象物の構造や属性値を表現したものとして扱うが，制約の観点からはクラスは特定の種類の対象物に共通な制約をモジュール化して表現したものととらえることができる．

従来のオブジェクト指向言語ではクラスとインスタンスの関係，いわゆる“instance_of”関係は固定的であった．しかし上記の考え方に従うと，その必然性はない．むしろ設計型問題では両者の関係を動的に変更可能とすることが問題解決に役立つ．

すなわち，設計型問題では要求仕様を満足する部品の種類と属性値の決定，すなわちクラスの探索とインスタンスの属性値の決定を並行して行うが，クラス・インスタンス関係が固定されていると，クラスを変更する場合，新しいクラスのインスタンスを生成し，既に決定した構成要素や属性値を再設定する必要がある．クラス・インスタンス関係を動的変更可能とすればそのような処理は不要になる．

ただし，クラスの変更が有効なのは抽象的なクラスから具体的なクラスへの詳細化（具体化）など，同一種類のクラス間で変更する場合がほとんどであることから，ここでは“is_a”の階層に沿った形でのみクラスの動的変更を可能とする．

インスタンスの属するクラスを“is_a”関係に従って動的に変更することにより，大規模な問題における標準的な手法である，トップダウン詳細化[13]を効率化できる．

図2はクラスの“is_a”階層を示したものである．一般にひとつのインスタンスはそれが属するクラスおよびその上位のクラスに記述された属性を持ち，制約を満足する必要がある．ここで，最初に抽象的なクラス0のインスタンスを生成し，その属性値を決定しながら，より具体的な下位のクラスに変更していくことにより，自然な形での詳細化処理が実現できる．

また，クラス・インスタンス関係を動的に変更可能とすれば，分類問題や画像理解のように，ある個体のデータすなわちインスタンスの属性が最初に与えられ，その型すなわちクラスが何かを推論するような問題にも自然な形で対応できると考えられる．

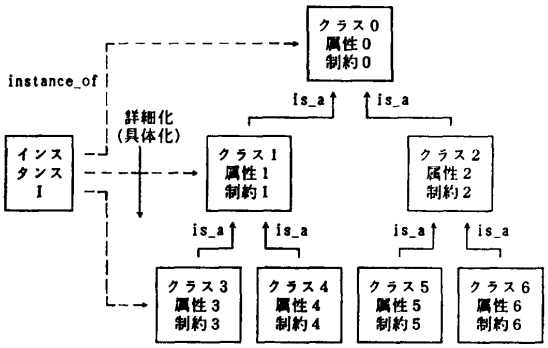

図2 クラスの“is_a”階層を利用した詳細化
Fig. 2 Refinement using “is_a” relations of classes.

なお，ここではクラスをオブジェクトとして扱わず，以後，オブジェクトといった場合インスタンスを指すものとする．

3.3 クラス階層の表現

知識処理ではクラスの抽象・具体関係，すなわち“is_a”関係を問題解決に利用することが多い．ところが一般には機能の包含関係を表すためにクラスの継承機能が用いられ，必ずしもクラス階層をそのまま“is_a”関係とみなすことはできない．

例えば多重継承の説明によく用いられるウィンドウの例を考えると，ラベルの付いたウィンドウのクラス「ラベル付きウィンドウ」を，クラス「ウィンドウ」とクラス「ラベル」を多重継承して定義する．この場合「ラベル付きウィンドウ」is_a「ウィンドウ」は意味があるが，「ラベル付きウィンドウ」is_a「ラベル」は抽象・具体関係を表すものとは言いがたい．

そこでここでは抽象・具体関係を表すものと，単に機能の包含関係を表すものとを明確に区別してクラス階層を表現する．前者を“is_a”関係，後者を“includes”関係と呼ぶ．また“is_a”関係では多重継承を許さないこととする．

一般に設計型問題では対象に関する広い知識を必要とし，多数のクラス定義が必要になるが，“includes”関係における包含されるクラスを指定クラスのみに限定せず，そのサブクラスをも許すことにより効率的なクラス構成が実現できる．そして，特定の包含クラスのみに着目した詳細化を可能とする．

本方式による金属板のクラス階層の例を**図3**に示す．ここで金属板に関するクラス構成は，「金属板」is_a「板」，「金属板」includes「金属」であり，金属については「鉄」is_a「金属」，「アルミ」is_a「金属」が

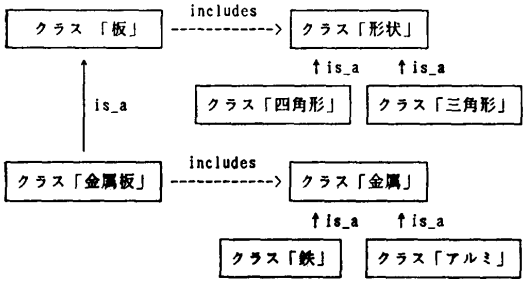

図 3 "is_a" と "includes" 関係によるクラス階層
Fig. 3 Class hierarchy with "is_a" and "includes" relations.

定義されている．金属板に使用する金属の種類を決定する問題では「金属板」の包含するクラス「金属」にのみ着目し，これを "is_a" 階層を下にたどって詳細化し「鉄」あるいは「アルミ」とすればよい．また "includes" 関係を用いることにより，「鉄板」，「アルミ板」等金属の種類ごとに金属板のクラスを定義する必要がなくなる．

4. 制約充足に基づく問題解決

4.1 制約充足状態の保持

問題解決過程においてオブジェクトはクラスで定義された制約と，外部から付加された制約を常に満足するように状態を変化させる．そして，複数の関連あるオブジェクト間で制約を伝播しながら，全体として問題解決を進めていく．

制約充足は属性値の決定に有効な手法である．すなわち，オブジェクトの属性値の設定や変更，制約式の付加等を行った場合に制約充足処理を実行することにより，具体的処理を記述することなく，関連する属性値を決定したり，修正できる．

例えば2段増幅器において，全増幅率，初段増幅率，次段増幅率の3つの属性間に

全増幅率＝初段増幅率＊次段増幅率

次段増幅率＝2＊初段増幅率

という制約が存在する場合，いずれか1つの属性値が与えられれば，制約式を評価することにより，残りの2つの属性値を決定できる．

4.2 クラスの探索

実際の問題では制約充足のためには，オブジェクトの属性値を変更するのみでなく，オブジェクトの型（クラス）や構造を変化させる必要があることが多い．しかし従来のシステムではそのための機能を提供していない．

ここでは，制約充足処理によりオブジェクトの属するクラスを変更可能とすることを提案する．すなわち，インスタンスの属性値や構成要素を変更した場合，そのインスタンスが属するクラスでは制約充足不可能であれば，自動的に制約充足可能なクラスを探索し，そのクラスに変更する機能を提供する．ただし，クラスの型に関する制約により，クラスの変更は "is_a" 階層において指定されたクラス（インスタンス生成時のクラス）とその下位クラスの範囲に限られる．

この機能を先程の図2のトップダウン詳細化の例を用いて説明する．最上位のクラス0のインスタンスIを生成し，その構成要素や属性値を決定しながら，クラス1まで詳細化が進んだとする．この時インスタンスIは制約0と制約1を満足しているが，ここでさらに詳細化を進める．まず，Iの属するクラスをクラス3に変更し，制約0，制約1，制約3の充足を試みる．これが可能であればIのクラスはクラス3となり，詳細化は終了するが，失敗すればバックトラックが発生し，次のクラス4に変更し，制約0，制約1，制約4の充足を試みる．成功すればIはクラス4のインスタンスとして終了し，再び失敗であれば詳細化処理自体が失敗したものとみなす．

この機能を利用して，処理手続きを記述することなく制約を満足するクラスを探索する機能を提供できる．これを制約に基づく分類法的推論 (Constraint-based taxonomic reasoning) と呼ぶことにする．

制約に基づく分類法的推論の非常に簡単な例として，増幅器の設計を考える．**図4**は増幅器のクラス階層であり，これに従って要求を満足する増幅器の種類を決定する．最初クラス「増幅器」のインスタンスと

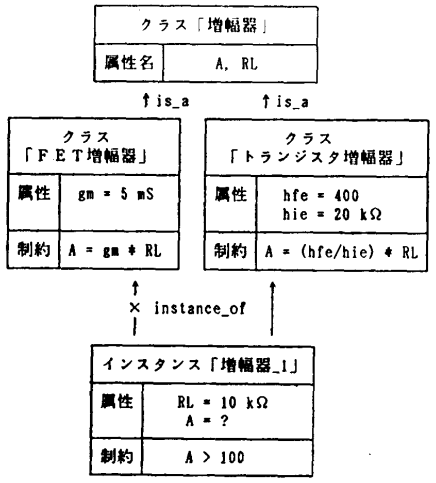

図 4 制約に基づくクラスの探索
Fig. 4 Class search by constraint satisfaction.

して生成したオブジェクト「増幅器_1」を，「FET 増幅器」か「トランジスタ増幅器」に具体化したい．増幅器_1 は増幅率Aと負荷抵抗 RL という属性を持ち，増幅率に関する制約条件が付加されている．RL の値が未定の状態では FET 増幅器とトランジスタ増幅器のいずれの具体化も可能であるが，RL の値を図のように決定した段階で，FET 増幅器は制約条件を満足せず，そのクラスはトランジスタ増幅器に決定される．

なお，クラス変更時にはその包含クラスや構成要素のクラスの型をチェックし，クラスの型に関する制約を満足していなければ満足するクラスに変更する処理を行う．

4.3 トップダウン詳細化

大規模な問題を解く場合には，これまで述べた「抽象」から「具体」へという形の詳細化（具体化）のみでなく，「全体」から「部分」へという形での詳細化（細分化）も重要である．前者は “is_a” 関係を利用したクラス探索を行うが，後者は “includes” および “consists_of” 関係を利用して部分問題に分割する．

一般に詳細化の手順は，まず，抽象的なクラスのインスタンスを生成し，これを “is_a” 階層に沿って具体化するとともに “includes” と “consists_of” 関係を利用して部分に分解し，分解したものをさらに具体化するという処理の繰り返しとなる．

図5 にギアシステムを例に3種の関係を利用したトップダウン詳細化のイメージを示す．最上位のクラス「ギアシステム」のインスタンスを生成し，クラスの “is_a” 階層に沿って要求仕様を満足する構造を持つギアシステムに具体化する．この場合「2段ギア」を選択したものとすると，これは “consists_of” 関係

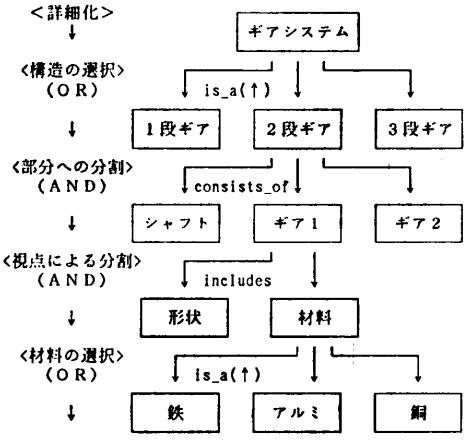

図5 対象モデルの構造を利用したトップダウン詳細化
Fig. 5 Top-down refinement of an object model.

により「シャフト」，「ギア1」，「ギア2」等の部品から構成されることが定義されており，次にこれらの部品ごとに詳細化を実行する．そして例えばギア1を “includes” 関係を利用して「形状」と「材料」のそれぞれの観点から具体化する．材料の場合，金属の種類を “is_a” の階層を利用して決定する．

詳細化処理において “is_a” では下位のクラスのいずれかを選択することになるが，“includes” および “consists_of” では，包含クラスや，構成要素はすべて不可欠の要素である．したがって，前者は OR 関係の，後者は AND 関係の木構造を表す．後者の場合，一般に分割したものの間には何らかの依存関係が存在するため，それらの間で制約伝播を行う必要がある．

5. 対象モデル表現システム FREEDOM

5.1 位置付け

以上述べた知識表現および問題解決機能を有する設計対象モデル表現システム FREEDOM (a Framework for REpresenting and Elaborating Design Object Model) を開発中である[14]．

図6 に FREEDOM を用いた設計システムの基本的な構成を示す．対象モデルは FREEDOM 上に表現され，設計手続きを実行する部分はその外部に存在する．一般にエキスパートシステムでは後者は設計方法に関する知識ベースである．

設計過程では，設計手続きが設計データを表す FREEDOM 上のインスタンスに対し，構成要素や属性値の変更，制約の追加削除等の処理を実行する．FREEDOM はクラスに記述された制約と，設計手続きがインスタンスに付加した制約を評価して，制約充足処理や詳細化処理を実行する．したがって FREEDOM はそれ自身が設計処理を実行するのでなく，インスタンスを常に制約充足状態に保持することにより，設計解の生成を支援するもので，TMS[15] に似た

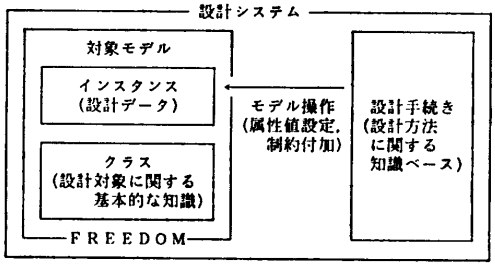

図6 FREEDOM を用いた設計システムの基本構成
Fig. 6 Basic structure of a design system with FREEDOM.

```
fr_class クラス名 has
    is_a 上位クラス名 ;
    includes
        [ キー名 :: ] 包含クラス名,
                  ・・・          ;
    consists_of
        構成要素名 [ ::= 構成要素クラス名 ],
                  ・・・                ;
    attributes
        属性名 [ ::= デフォルト値 ],
                  ・・・              ;
    constraints
        制約式,
        ・・・;
    meth(Obj, メッセージ) :- メソッド本体;
                  ・・・                ;
fr_end.
```

(a) クラス定義形式

```
        % メッセージ形式 %
    セレクタ(引数1, ・・・, 引数n)

        % メッセージ送信の記述形式 %
    :send(送信先のオブジェクト, メッセージ)
```

(b) メッセージの形式

図 7 FREEDOM におけるクラス定義形式
Fig. 7 Syntax of class definition in FREEDOM.

形態のシステムである.

今回, FREEDOM 開発の第一段階として小規模な組み合わせ設計, パラメトリック設計を対象としたプロトタイプを, 逐次推論マシン PSI 上に ESP[16] を用いて試作したので, 以下それについて述べる.

5.2 クラス定義

FREEDOM のクラスの記述形式は ESP ユーザにとって違和感のないことを重視し, 図7(a)のような ESP に近い文法を採用した.

クラス定義では必要な項目のみを記述すればよく, 各項目で [ ] で囲まれた部分は省略できる. また, 包含するクラス名にはキーを付けることができ, キー名を指定することによりオブジェクトの特定の側面のみに着目した処理が可能である. 構成要素の定義で指定したクラス名は構成要素の型に関する制約として扱われる. 現在制約式は数式表現可能なものに限定し, 制約式中で参照できるのは当該オブジェクトとその構成要素の属性に限る.

図8に本システムの評価用の例題として用いた電子回路のクラス定義の一部を示す. ここで制約式中の "x :-> y" は構成要素 x の属性 y を表す.

メソッドやメッセージの記述形式は図7(a), (b)のような形式である. メソッドは対象モデルを記述するためというより, 必要により一般のオブジェクト指向言語と同様な手続き的処理を記述したり, 外部から対象モデルを操作する場合のインタフェースを提供するためのものである.

```
%%%  電圧増幅器  %%%

fr_class  voltage_amplifier  has
    attributes
        gain,  input_imp,  output_imp,
        max_output,  vcc,  vee,  i,  power;
    constraints
        power = (vcc + vee) * i;
fr_end.

%%%  2段差動増幅器  %%%

fr_class  two_stage_differential_amplifier  has
    is_a
        voltage_amplifier;
    consists_of
        stage_1 ::= differential_amplifier_unit,
        stage_2 ::= differential_amplifier_unit;
    attributes
        beta,  k,  open_gain,  amp_input_imp,
        r_nfb_1,  r_nfb_2,  r_input;
    constraints
        2 * open_gain * (stage_1 :-> output_imp
          + stage_2 :-> input_imp) = stage_2 :-> input_imp
                  * stage_1 :-> gain * stage_2 :-> gain,
        beta * (r_nfb_1 + r_nfb_2) = r_nfb_2,
        k * (1 + open_gain * beta) = 1,
        gain = k * open_gain,
        k * amp_input_imp = stage_1 :-> input_imp,
        input_imp * (amp_input_imp + r_input)
                          = amp_input_imp * r_input,
        output_imp = k * stage_2 :-> output_imp,
        max_output = stage_2 :-> max_output,
            ・  ・  ・  ( 一部略 )  ・  ・  ・  ;
fr_end.
```

図 8 電子回路のクラス定義の例
Fig. 8 An example of class definition.

また, インスタンスの生成, クラスの明示的変更, 詳細化, 属性値の変更, 構成要素の変更, 制約の追加および削除等の処理を行うメソッドをシステム提供しており, これらを用いて対象モデルを操作することができる.

5.3 実現方式

前述の文法に従って記述されたクラス定義のソースプログラムはトランスレータにより ESP 形式の内部表現に変換されて実行可能となる.

インスタンスの属するクラスの動的変更を効率よく行うため, 内部表現では, 1つのインスタンスを, 上位クラスとの差分を表す ESP のインスタンス (以下部分インスタンスと呼ぶ) の組み合わせで表現する. これによりクラスの動的変更処理を部分インスタンスの追加, 削除で実現できる.

制約充足は制約論理プログラミングと制約伝播を融合した方式を採用した[17]. 1つのオブジェクト内の制約充足は Buchberger アルゴリズムを用いた制約論理プログラミング言語 CAL[18] の制約評価系を利用している. これにより連立方程式を解くことができる. 不等式については, 現在はその不等式を満足するかどうかの検査のみを行っている.

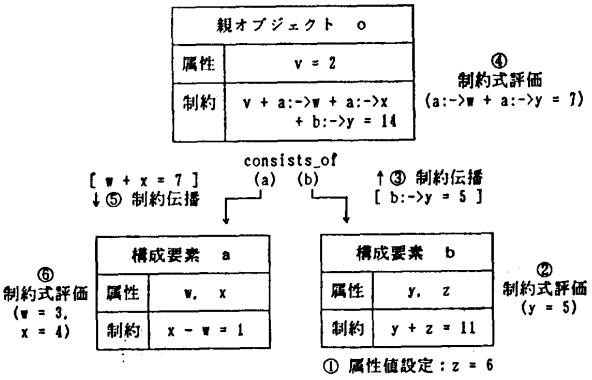

図 9 オブジェクト間の制約伝播の例
Fig. 9 An example of constraint propagation.

複数のオブジェクトにわたる制約充足は各オブジェクトの独立性に優れた制約伝播による解法を採用している．FREEDOM では全体を表すオブジェクト（以下，親と呼ぶ）はその構成要素の属性を参照できるが，構成要素は親の属性を参照できない．したがって属性の相互参照はなく，制約伝播により効率のよい制約充足が可能である．

複数のオブジェクト間での制約式の依存関係が単純な場合は，属性値が決定した時その値を関連するオブジェクトに伝播することにより解くことができる．しかし制約式における属性の参照関係が複数のオブジェクトにわたってループを生じる時には，単純な属性値の伝播では解けないことがある．そこで FREEDOM では制約式を伝播し，1つのオブジェクト上で関連のある制約式を同時に評価することにより解く方式を採用した．

図 9 は制約伝播の例である．親オブジェクト o は構成要素 a と b を持ち，各オブジェクトは図示したような属性および制約を有する．今，構成要素 b の属性 z の値が6に設定された（①）とすると，構成要素 b で制約評価を実行し（②）属性 y の値が決まり，それを親オブジェクト o に伝播する（③）．o での制約評価（④）により構成要素 a の属性のみに関する制約式が得られるので，これを a に伝播し（⑤），a がもともと持っていた制約式とともに評価することにより属性 w と x の値が得られる（⑥）．

なお，制約充足のため，既に設定されている属性値の変更が必要となる場合には，できる限り少数の属性値を変更する．変更したくない属性値は明示的に制約として表現すればよい．

6. 今後の課題

本論文で提案した知識表現および問題解決の基本機能の有効性については，試作した FREEDOM のプロトタイプを簡単な電子回路などの問題に適用することにより確認できた．しかし，本プロトタイプは基本機能の評価のための実験的システムで，機能的，性能的に制限されており，実用化にはいくつかの課題がある．

まず，制約充足機構の機能強化が必要である．現在のプロトタイプでは数式表現された制約のみを扱っているが，実用化には記号的な制約を扱う機能を提供することが必要である．

また，ここではトップダウン詳細化に重点を置いているが，実際の問題解決ではボトムアップ的な手法が用いられることも多い．したがって今後トップダウン処理とボトムアップ処理の協調的な問題解決を実現したいと考えている．

本論文で述べた知識表現は対象モデル記述に重点を置いており，問題解決に関する知識の表現および利用のための機構については外部にあることを仮定してきた．したがってヒューリスティックスやメタ知識の制約表現については今後の課題である．

大規模な問題に対しては全体を部分問題に分割して並列に解くことにより，効率のよい問題解決を実現することが期待される．そこで現在，対象モデルの構造を利用して分割する方式を検討中である．すなわち 4.3 節で述べたように，“is_a”，“consists_of”，“includes” 関係を利用して部分問題に分割し，OR 並列あるいは AND 並列に処理を進める．ただし，部分問題間の制約伝播が並列度を低下させることのない方式を考案する必要がある．

7. おわりに

制約に基づくオブジェクト指向知識表現および問題解決機構を提案し，それに基づく設計対象モデル表現システム FREEDOM のプロトタイプを試作した．

ここで提案した知識表現はオブジェクト指向と制約指向の融合を図ったもので，制約の宣言的記述，動的変更可能なクラス・インスタンス関係，“is_a と “includes” 関係に基づくクラス階層の表現等の特徴を有する．そして，属性値の決定やクラス探索機能を持つ制約充足機構により，オブジェクトを常に制約を満足する状態に保ち，問題解決を行う．

今後，汎用的で実用的なシステムを目指して機能強

化を図っていく．また，オブジェクト単位の並行処理機能や並列制約充足機能，並列探索機能などを取り込み，並列推論機能を有する知識表現システムへと発展させる予定である．

謝辞 本研究の機会を与えていただくとともにご指導いただいた，ICOT の淵所長，第五研究室の藤井前室長(現在 NTT)および生駒室長，有益な御討論をいただいた永井氏（現在東芝），寺崎氏はじめ，第五研究室の皆さんに感謝する．

参 考 文 献

1) 上野晴樹：知識工学入門，6.1 対象モデル，オーム社 (1985).
2) Ohsuga, S.: Conceptual Design of CAD Systems Involving Knowledge Bases, in *Knowledge Engineering in Computer-Aided Design* (Gero, J. ed.), pp. 29-88, North-Holland (1985).
3) Minsky, M.: A Framework for Representing Knowledge, in *The Psycology of Computer Vision* (Winston, P.H. ed.), McGraw-Hill (1975).
4) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley (1983).
5) Stefik, M. and Bobrow, D.G.: Object-Oriented Programming: Themes and Variations, *AI Magazine*, Vol. 6, No. 4, pp. 40-62 (1986).
6) Stallman, R.M. and Sussman, G.J.: Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, *Artif. Intell.*, Vol. 9, pp. 135-196 (1977).
7) Sussman, G.J. and Steel, G.L. Jr.: Constraints—A Language for Expressing Almost-Hierarchical Descriptions, *Artif. Intell.*, Vol. 14, pp. 1-39 (1980).
8) Heintze, N., Michaylov, S. and Stuckey, P.: CLP (R) and Some Engineering Problems, in *Logic Programming, Proc. of 4th Int. Conf.* (Lassez, J.L. ed.), pp. 675-703, MIT Press (1987).
9) Borning, A.: The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory, *ACM Trans. Prog. Lang. Syst.*, Vol. 3, No. 4, pp. 353-387 (1981).
10) Borning, A. and Duisberg, R.: Constraint-Based Tools for Building User Interfaces, *ACM Trans. Graphics*, Vol. 5, No. 4, pp. 345-374 (1986).
11) Harris, D.R.: A Hybrid Structured Object and Constraint Representation Language, *Proc. of AAAI-86*, pp. 986-990 (1986).
12) Struss, P.: Multiple Representation of Structure and Function, in *Expert Systems in Computer-Aided Design* (Gero, J. ed.), pp. 57-92, North-Holland (1987).
13) Hays-Roth, F. et al.: *Building Expert Systems*, Chapter 4, Addison-Wesley (1983).
14) 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-AI-60 (1988).
15) Doyle, J.: A Truth Maintenance System, *Artif. Intell.*, Vol. 12, pp. 231-272 (1979).
16) Chikayama, T.: ESP Reference Manual, ICOT Technical Report, TR-044 (1984).
17) 横山孝典，佐塚秀人：制約に基づくオブジェクト指向知識表現システム FREEDOM における制約充足方式，第 38 回情報処理学会全国大会論文集，7D-5 (1989).
18) 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，SS 87-22 (1988).

(平成元年 5 月 11 日受付)
(平成元年 9 月 12 日採録)

横山 孝典 (正会員)

1959 年生．1981 年東北大学工学部通信工学科卒業．1983 年同大学院工学研究科電気及通信工学専攻修士課程修了．同年(株)日立製作所入社．日立研究所にて図形認識，ユーザインタフェースの研究開発に従事．1987 年より(財)新世代コンピュータ技術開発機構に出向．知識表現の研究に従事．電子情報通信学会会員．

佐塚 秀人 (正会員)

1962 年生．1985 年静岡大学工学部情報工学科卒業．1987 年同大学院修士課程修了．同年日本電信電話(株)に入社．1988 年 NTT データ通信(株)に転籍，現在に至る．エキスパート・システム構築ツールの研究開発に従事．日本ソフトウェア科学会会員．