情報処理学会第37回全国大会
(論理型言語・エキスパートシステム関係)

September, 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

設計型問題向き対象モデル表現方式

－ オブジェクト指向パラダイムの拡張 －

横山　孝典

新世代コンピュータ技術開発機構

1．はじめに

最近エキスパートシステムにおいて，問題の対象に関する知識と問題の解決法に関する知識とを分離し，前者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]。対象モデルでは属性や形状，構造，その他問題解決上必要な対象に関する情報や知識を表現する必要があるが，その表現方式としてはオブジェクト指向言語（またはフレーム）が適していると言われる。

ところで一般に分析型問題における対象モデルが固定的であるのに対して，合成型問題における対象モデルは問題解決過程で動的に変化するという特徴がある。特に設計は，様々な制約条件のもとで要求仕様を満足するモデルを生成する問題と定式化でき，モデルの取捨選択，修正，詳細化等の処理を効率的に行える表現方式が望まれる。しかし従来のオブジェクト指向言語は，オブジェクトに関する処理を全てメソッドとして手続き的に記述する必要があり，そのままでは設計対象モデルの表現方式として適切とは言い難い。

本報告ではオブジェクト指向を基本に，制約充足機構を積極的に取り込むことにより，設計における効率的な問題解決処理を支援できる対象モデル表現方式を提案する。これにより対象モデルは単なる静的なデータ構造ではなく，それ自身ある程度の問題解決能力を持つことができ，エキスパートシステムの高度化に役立つものとなる。

2．オブジェクト指向への制約の導入

設計，あるいはより一般に合成型問題では，制約条件が問題解決上重要な役割を演ずる。すなわち，制約は合成問題の基本的問題である組み合わせの爆発を防ぎ，属性値や構造の決定に利用できる。そこで設計型問題向きの対象モデル表現の枠組みとしてはオブジェクト指向に制約を導入する方式が有力と考えられる。

これまでにもオブジェクト指向と制約指向を融合したシステムが報告されている[2][3]。しかしこれらはインスタンス変数（属性値）間の数値的な制約しか扱っていない。ところが，設計型問題においては，対象の構造上の制約や，記号的な制約を扱えなければならない。

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，この関係には部分が全体にとって必要不可欠な場合と，必ずしも必要でない場合がある。前者の例は四角形における４つの辺で，後者の例は本棚におけるその中の本である。ここでは前者を全体に対する「構成要素」，その関係を“consists-of”と呼んで，オブジェクトの構造上の制約として扱うことにする。

また，数値的な制約と記号的な制約を同じ形で取り扱い可能とするため，制約条件は基本的に述語表現する。

ところで，制約条件は静的に与えられるものとは限らず，問題解決の過程で動的に生成されるものもある。このためインスタンスに動的に制約を追加，削除する機能も必要となる。

3．柔軟なクラス・インスタンス関係

設計における対象モデルでは設計対象一般に関する知識を表現するものと，設計過程で生成される設計解（あるいはその中間結果）を表現するものとを区別する必要がある。いわば前者は対象モデルの「クラス」，後者は「インスタンス」である。

この観点でみると設計過程は要求仕様を満足するクラスを探索し，インスタンスの属性値を決定することと定式化できる。ところが，実際の設計では両者は並行して行われ，インスタンスの属性値の決定後にクラスを変更することが頻繁に発生する。また，一般に構造の異なるオブジェクトは別のクラスで表現するため，構造上の修正はクラスの変更を招くことになる。そこでここでは，クラス・インスタンス関係，いわゆる“instance-of” 関係を従来のように固定的なものとせず，問題解決過程で動的に変更可能とする方式を提案する。

ところで，クラスを特定の範疇の対象に関する制約を記述したものととらえれば，インスタンスの属するクラスの変更を制約充足処理のひとつとして実現することが考えられる。すなわち，インスタンスの属性値の変更や，構成要素の追加，削除を行った場合，それが属するクラス内での制約充足処理が不可能であれば，自動的に制約充足が可能なクラスに変更する機能を提供する。この機能により，インスタンスに対する操作のみで，制約を満たすクラスの探索が可能となる。

例えば金属板の種類を決定する問題を考える。この時のクラス階層を図１に示す。ここで，最初インスタンス「金属板Ａ」は鉄板を使用することを想定していたが，面積と厚さを決定した時点で，重量を一定値以下とする制約により金属板Ａの属するクラスは「鉄板」から「アルミ板」に変更される。

4．“is-a” と “includes” によるクラス階層

一般に設計はトップダウンになされるが，このことは対象モデルを抽象的なレベルから具体的なレベルへと詳細化していくことと見なせる。そこで効率的な詳細化処理を実現するため，「抽象－具体」を表わすクラス階層，いわゆ

A Framework for Representing Object Model in Design　- An Extension of Object-Oriented Paradigm -

Takanori YOKOYAMA

ICOT

る“is-a”関係を利用することが考えられる。ところが従来の多重継承を許すオブジェクト指向言語では，機能の包含関係を表わすためにクラスの継承機能を利用することが多く，必ずしも意味的な“is-a”関係の表現とはなっていない。また多重継承を多用するとクラス階層の理解が困難になるという問題もある。

そこで本方式では“is-a”関係を意味的に同一次元に属するクラス間の「抽象－具体」関係を示すものに限定し，多重継承は許さないことにした。これにより“is-a”関係は単純な木構造となる。従って設計過程における詳細化処理は一般に“is-a”の階層に沿って，要求仕様を満足するクラスを探索することになる。そして，前述の制約充足によるクラスの動的変更についても“is-a”の同一木構造に属するクラス間でのみ許すことにする。

ただし機能の包含関係という意味での多重継承機能も知識のモジュール化という点で重要である。そこで“is-a”関係とは別に，クラス間の機能の包含関係を表わすものとして“includes”関係を定義可能とした。ここで，包含するクラスは指定したクラスのみに限定せず，そのサブクラスをも許す。そして包含するクラスのうち特定のものに着目して詳細化を行うことができるようにする。

本方式による金属板のクラス階層の例を図２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」が定義されている。金属板を鉄板あるいはアルミ板に決定する問題では「金属板」の含むクラス「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”関係を用いれば，「鉄板」，「アルミ板」等金属の種類全てのクラスを定義する必要がなくなり，クラス構成をすっきりしたものにできる。

なお全体を部分に分割して解くという意味での詳細化処理もあり，これは“consists-of”関係の階層を利用することになるが，その詳細についてはここでは触れない。

５．ＥＳＰによる実現

現在，ここで提案した機能を有する知識表現言語をＰＳＩ上のＥＳＰによりインプリメントしている。数値的な制約充足機能は制約論理プログラミング言語ＣＡＬ[4]を利用することを考えている。

ここではひとつのインスタンスを継承するクラス数だけのＥＳＰのインスタンスの組み合わせとし，クラスの動的変更処理をクラス間の差分を表わすＥＳＰインスタンスの追加，削除で実現している。この場合，継承機能はいわゆる“delegation”[5]によって実現する。

また処理のオーバヘッドを考慮すると，大規模な対象モデルを扱う場合には常時制約充足を行うのではなく，明示的に指定された時のみ実行するのが実用的と思われる。

６．おわりに

以上設計型問題における対象モデル記述のための知識表現方式の基本機能について述べた。本方式はオブジェクト指向を基本に，制約の導入，柔軟なクラス・インスタンス関係，“is-a”と“includes”によるクラス階層の表現等の機能拡張を図ったものである。この拡張においては制約が重要な働きをするのが特徴である。この点で本方式は単

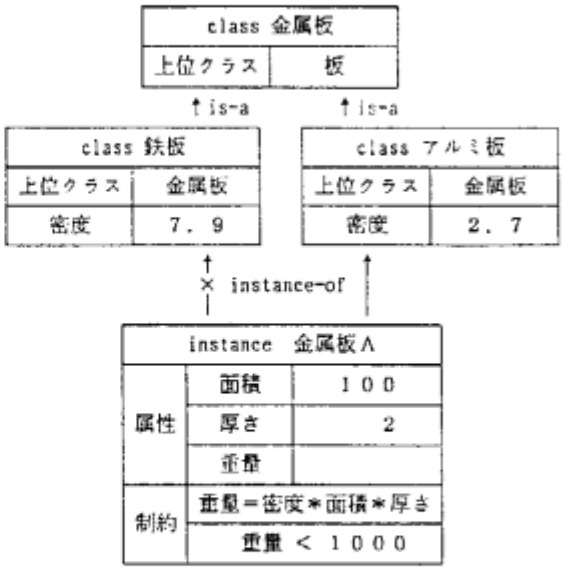

図１ 制約充足によるクラスの変更

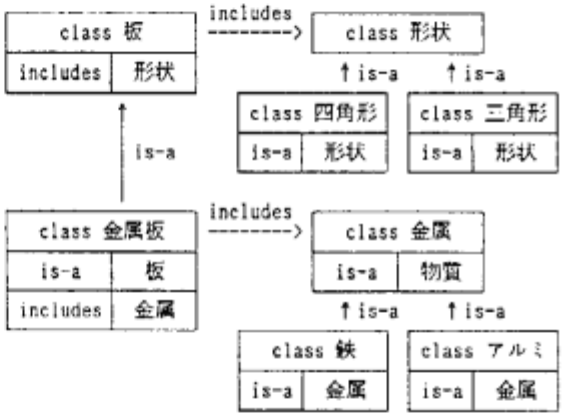

図２ “is-a”と“includes”関係によるクラス階層

なるオブジェクト指向への制約の導入というより，オブジェクト指向パラダイムを制約の観点で新たにとらえ直そうという試みである。

今後の課題としてはクラスの動的生成や動的変更機能の実現や，制約伝播を利用した並列協調問題解決処理への発展がある。

参考文献

[1] 上野 “知識工学入門”，６章 対象モデル，オーム社，(1885)

[2] Borning, A. “The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory”, ACM Trans. Program. Lang. Syst., vol.3, no.4, pp353-387, (1981)

[3] Harris, D. R. “A Hybrid Structured Object and Constraint Representation Language”, Proc. of AAAI-86, pp986-990, (1986)

[4] 坂井，相場 “ＣＡＬ：制約論理プログラミングの理論と実例”，信学会研究会資料，SS87-22，(1988)

[5] Lieberman, H. “Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems”, OOPSLA'86 Proc., pp214-223, (1986)