並行オブジェクト指向言語 y におけるクラス階層の定義と継承

6J－4

小島一人＊ 崔梗泳 ＊＊ 野呂昌満＊ 原田賢一＊＊

＊南山大学情報管理学科 ＊＊慶應義塾大学理工学部

要旨

並行オブジェクト指向プログラミング言語 y は，ネットワークソフトウェアの記述を目的として設計したプログラミング言語である．これまでの多くのオブジェクト指向言語では，クラスの論理的関係である IS-A 関係と，コード共有のための継承とを区別せずに扱っている．クラスの論理的関係を壊すことなく継承によってコード共有するために，y では多重継承をコード共有を記述するもの，多重 IS-A 関係をクラスの論理的関係を記述するものとして定義した．実現だけを継承することで，クラスの論理的階層構造を壊さずにコードを共有できる．

1　はじめに

これまでのオブジェクト指向プログラミング言語の多くは，クラスの論理的関係である IS-A 関係とクラスのコード共有のための継承が区別されていない．クラスが IS-A 関係にある場合に，継承が起こると考えるのは不自然ではない．しかし，コード共有のためにサブクラスがスーパークラスの属性を継承している場合，必ずしもスーパークラスとサブクラスが IS-A 関係にあるとは限らない．例えば，Smalltalk-80[Gol89] におけるスーパークラス `Dictionary` とサブクラス `Set` の関係は，論理的なスーパークラス / サブクラスの関係にない．コード共有のためだけにクラス階層を定義した結果，この例ではクラスの論理的階層構造が壊されている．

我々はプログラミング言語 y[Koj93] において，コード共有を記述するための多重継承と，クラス間の論理的関係を記述するための多重 IS-A 関係とを分離して定義することを提案した．その結果，クラスの論理的階層構造を壊さずにコード共有を行えるようになった．

Definition of Class Hierarchy and Inheritance in the Concurrent Object-Oriented Programming Language y

Kazuhito Kojima*, Kyong-Rok Choi**, Masami Noro*, Ken'ichi Harada**

*Dept. of Information Systems and Quantitative Science, Nanzan University, 18 Yamazato-cho Showa-ku Nagoya 466, Japan

**Dept. of Computer Science, Keio University, 3-14-1 Hiyoshi Kouhoku-ku Yokohama 223, Japan

2　y におけるクラス定義

y では，クラスが抽象データ型を記述するための構文要素である．y でクラスを定義する場合，仕様部と実現部に分けて記述する．クラスの仕様部には，抽象データ型をアクセスするためのメソッドの仕様を記述し，実現部には，データ構造の定義とメソッドの実現を記述する．クラスの実現部は公開部と非公開部に分割でき，公開部にサブクラスに対して公開するメソッドを記述する．仕様部と実現部を，C++[Str91] のクラス記述と対比させると図 1のようになる．

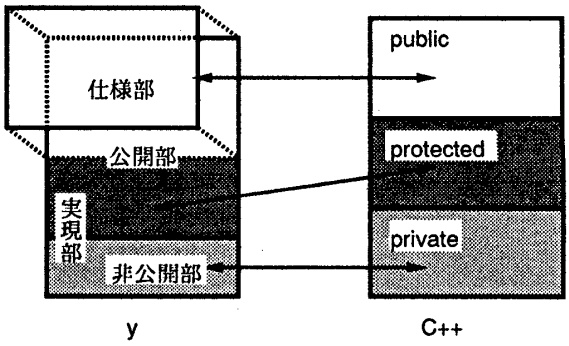

図 1: クラス記述

クラスインスタンスを使用する立場から見ると，仕様部に記述されたメソッドは C++ の公開 (public) メンバに相当する．サブクラスだけに公開される C++ の限定公開 (protected) メンバは，実現部の公開部のうち仕様部に記述されないデータに相当する．C++ の非公開 (private) メンバは，仕様部に記述されない，実現部の非公開部のメソッドと局所変数に相当する．

3　クラス階層の定義と継承

y ではクラスの論理的関係は仕様部の IS-A 関係として記述し，クラスの実現部の多重継承によってコード共有できるようにした．前述の Smalltalk-80 の例は，y では図 2のように定義できる．クラスの論理的関係からいえば，クラス `Dictionary` は `Collection` のサブクラスで

ある．コード共有のために，クラス Dictionary は，論理的なスーパークラスではない Set の属性を継承する．

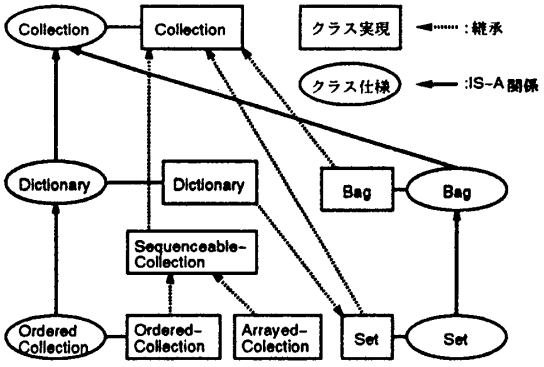

図 2: 多重継承と多重 IS-A 関係

多重 IS-A 関係

単一の IS-A 関係よりも多重の IS-A 関係のほうが現実世界を自然な形式で記述できるので，*y*では多重 IS-A 関係によってクラスの論理的関係が記述できるようにしている．例えば，馬というクラスを考えた場合，哺乳類と乗物のサブクラスであると考えられることは，一般にいわれている通りである．

*y*では，クラスが名前互換 (name compatible)[Weg90] であれば IS-A 関係が成り立つと考え，サブクラスでのメソッドの追加と再定義だけを許し，メソッドの削除は許さない．スーパークラスのメソッドと同じ名前のメソッドを持つ限り，サブクラスは名前互換であるので，サブクラスでのメソッドの追加と再定義は名前互換である．しかしスーパークラスで定義されているメソッドがサブクラスで削除された場合，スーパークラスと名前互換ではない．

名前互換であることがクラスが IS-A 関係であることの十分条件ではないが，振舞互換 (behavior compatible)，署名互換 (signature compatible) で IS-A 関係を定義する場合には，以下の問題がある．

- 振舞互換ではスーパークラスの振舞を明確かつ完全に定義することが困難である．
- 署名互換ではメソッドの実現の異なったコードに対して意味的に同一であることを保証しなければならないが，これを言語で支援することは困難である．

これらの問題点に対する一般的に受け入れられている解決方法がないため，現時点では，クラスが名前互換であれば IS-A 関係であると定義する．しかし今後の研究によって，振舞の明確な記述の方法について解決がなされれば，IS-A 関係の条件として振舞互換の導入を検討する．

多重継承

複数クラスから実現を継承し，クラス定義の記述の重複をできるかぎり少なくするために，*y*に多重継承を導入した．例えば，入出力ストリーム IOstream はクラス InputStream と，クラス OutputStream の両方を継承することによって，コードを共有し実現できる．

メソッドの動的結合

*y*のすべてのメソッドは C++ の仮想関数に相当する．あるインスタンスのメソッドを動的に結合する場合，実現部の階層の最下部からメソッドを結合する．

メソッドの名前衝突の解決

多重継承によってメソッドの名前の衝突が起こった場合，*y*では C++ と同様の方法を用いてスコープ解決演算子によって解決する．

4 おわりに

*y*ではコード共有を記述するための多重継承と，クラス間の論理的関係を記述するための多重 IS-A 関係を分離して定義した．図 2に示したように，*y*では，クラスの論理的階層構造を破壊することなく，コードを共有できる．

*y*では，クラスが名前互換であれば IS-A 関係が成り立つと定義する．しかし厳密な議論をすると，名前互換では必ずしもクラスが IS-A 関係であることを保証することはできない．今後，振舞互換における明確な振舞の定義方法について解決し，振舞互換を IS-A 関係の条件とすることを検討している．

参考文献

[Gol89] Goldberg, A.: *Smalltalk-80: the language*, Addison-Wesley, 1989.

[Koj93] Kojima, K. and M. Noro: The Design of the Object-Oriented Concurrent Programming Language y, in *Proceedings of the Joint Conference on Soft. Eng. '93*, IPSJ, pp. 59–66.

[Str91] Stroustrup, B.: *The C++ Programming Language*, Addison-Wesley, second edition, 1991.

[Weg90] Wegner, P.: Concepts and Paradigm of Object-Oriented Programming, *OOPS Messenger*, Vol. 1, No. 1, 1990, pp. 7–87.