

1T-10

クラスの意味記述を利用した オブジェクト指向プログラミング支援

工藤 智広
東京大学

濱田 喬
学術情報センター

1 はじめに

オブジェクト指向プログラミング (Object-Oriented Programming; 以下 OOP) では、独立性の高いモジュールであるオブジェクトがプログラムの構成要素として用いられる。このため、OOP ではオブジェクトを単位としてプログラム部品の再利用を行いやすくなっている。多くの OOP 言語では、オブジェクトの性質を記述するのにクラスを用いており、実際には、このクラスの再利用を行うことになる。

一般に、プログラム部品の再利用においては、既存部品の集合の中から、現在取り扱っている状況に合った部品をいかに選び出すか、が重要なポイントとなっている。このことは OOP においては、クラスの集合であるクラスライブラリの中からいかにクラスを検索するか、になる。

本研究では、効果的な OOP のクラス検索方法の提案を目的として、クラスの構文的情報にかわって意味的情報を検索の手がかりとして用いることを考えた。そして、このためのクラスの各手続きの動作を格フレーム形式で記述する方法として OSD(Object Semantic Description) を提案する。

2 OSD によるクラス記述方法

2.1 OSD の記述内容

まず、OSD により記述される内容について解説する。OSD で記述するクラスの意味とは、クラスから生成されるオブジェクトがどのような動作をするか、ということである。オブジェクトの動作は、

Object-Oriented Programming Assistance using Class Semantic Description

Tomohiro KUDO (University of Tokyo), Takashi HAMADA (National Center for Science Information Systems)

通常、外部からのメッセージ送信によって内部の手続き (メソッド) が起動されることによって実現される。したがって、各メソッドの動作を記述することで、オブジェクトが行うことができる動作を記述することができる。そこで、OSD での記述内容は該当クラスのメソッドの動作とした。

この記述方針はクラス検索の立場からいっても適切である。なぜなら、クラスを検索する場合、クラス全体としてよりも、ある機能を満たすクラスを探す、ということが自然だからである。

2.2 OSD の記述形式

前節で述べた内容を実際に記述する方式であるが、OSD では、操作的方法や公理的方法などの形式的意味記述方法でなく、格フレームを用いたより自然言語形式に近い形をとった。このことから、OSD では、形式的な方法に比べ、記述内容の正確さや厳密さは失われることになるが、クラス検索を行なう際の記述の理解しやすさや、実際に記述を行なう時の記述しやすさといった点が活かされる。現実のクラス選択の場面では、ある程度対象を絞り込んで、その後、より詳しい仕様などの情報を見ることがにより最終的に再利用部品を決定する、ということがおこなわれるので、その初期段階の選択を目的とすると、正確さよりも扱いやすさが重視されるのが現実的といえる。

OSD で用いる格フレーム形式とは、Fillmore によって提唱された格文法に基づくものである。格文法では、構文上の表層的な格 (通常の文法における主格、目的格など) に代わり、文の意味に即した深層的な格を用い、これによって自然言語文の表す意味を記述しようとする。格フレーム形式は、述語とそれに応じて決まる深層格 (行為者格、対象格など) との対応を記述したものである。

これら深層格の種類を決定することによって、目

的に応じた格システムを構築することが可能であるが、OSD では、メソッドの動作を記述するという目的から用いる深層格を選択した。なお、動作の主体は常にメソッド自身であるとして、行為者格は自明なものとして省略する。

以下に OSD での深層格を示す。各格の名前の後に示したのは、それぞれの格を表すキーワードである。

対象格 (OBJ) メソッドの動作が何を対象として行われたかを表す。

始点格 (FROM) メソッドの動作の出発点を表す。

終点格 (TO) メソッドの動作の終点を表す。

道具格 (WITH) メソッドの動作が何を用いて行われたかを表す。

場所格 (LOC) メソッドの動作が行われていた場所を示す。

状態格 (COND) オブジェクトの状態を尋ねるメソッドの記述に用いる。

例えば、ファイルを転送するメソッドの動作を記述するときには、対象格となるのはファイルであり、始点格は転送元、終点格は転送先となる。また、画面に線を描画するメソッドの動作であれば、対象格となるのは線、場所格となるのは画面ということになる。最後の状態格は、オブジェクトが「ある状態」であるかどうかを調べる、といった記述を行うときに、その「ある状態」を表現するために導入した(図1の empty の記述参照)。

2.3 OSD の記述例

OSD による記述例を図1に示す。この例は、整数を扱うスタック (LIFO バッファ) クラスの一部であり、各メソッドの宣言の後に対応する OSD 記述を行っている。ここで、対象 OOP 言語としては C++ を用いている、他の OOP 言語においても同様に記述することができる。

2.4 OSD の特徴

今までに述べてきたことも含めて、OSD の特徴についてまとめる。

```
class intStack {
    void push(int);
        (push down (OBJ:integer, int) (TO:stack))
    int pop();
        (pop up (OBJ:integer, int) (FROM:stack))
    int top();
        (return (OBJ:integer, int)
         (FROM:top of stack))
    int empty();
        (check (OBJ:stack) (COND:empty))
};
```

図 1: OSD の記述例

理解しやすさ、記述しやすさ 自然言語に近い形式を用いているため、理解しやすく記述がしやすい。

汎言語性 特定の OOP 言語に依存しない。また、記述に用いる言語も特定の言語には依存せず、例えば、日本語を用いた記述も可能である。

半面、正確で厳密な記述は不可能であり、また、自由な自然言語形式に比べると、記述能力はかなり制限される、などといったマイナス面もある。ただし、大量のクラスの中からの候補の絞り込みという目的であれば、OSD は有効であると考えられる。

3 おわりに

今後は、実際の検索アルゴリズムについても検討し、具体的な検索システムを試作して OSD の有効性を検証するとともに、クラス検索以外の再利用支援方法への適用も検討していく予定である。

参考文献

- [1] 工藤, 濱田: “クラス情報データベースを用いたオブジェクト指向プログラミング支援システム”, 1992 年電子情報通信学会春期全国大会
- [2] 乙部, 濱田: “オブジェクト指向プログラミングのためのクラス検索システム”, 1990 年電子情報通信学会秋期全国大会