

# 객체지향 모델에서 사용범위 기능 도입에 관한 연구

권기향<sup>†</sup> · 김지승<sup>††</sup>

## 요 약

기존의 객체 지향 모델들은 공유와 재사용 측면에서 효율적이지만, 어렵게도 객체의 사용 범위를 정의할 수 있는 방법이 결여되어 있다. 본 논문에서는 객체의 사용 범위를 정의할 수 있도록 하는 새로운 객체 지향 모델을 정의한다. 즉 객체는 필요할 동안만 프로그램 내에 존재하게 된다. 따라서 이 모델에서 현재 사용중인 객체의 집합은 동적으로 변할 수 있고, 필요한 객체들만이 관리된다. 우리는 예제들을 통하여 이 모델의 유용성을 보인다.

## Extending Object-Oriented Models with Scoping Constructs

Keehang Kwon and Jiseung Kim

## ABSTRACT

While object-oriented models are effective in achieving sharing and code reusability, they unfortunately lack a mechanism for giving scope to objects. We propose an object-oriented model in which each object can be given a scope, i.e., an object becomes available only when it is needed. Thus, the set of currently available objects is dynamically changing and only the needed set of objects is maintained in this model. We illustrate the usefulness of this model through some examples.

## 1. 서 론

*obj.f*

현재 다양한 객체 지향 모델들이 제시되어 있으며, 이를 기반으로 하는 여러 시스템들이 사용되고 있다[1-8]. 하지만 이와 같은 객체 지향 모델들은 뛰어난 재 사용성을 가지고 있음에도 불구하고 객체가 사용되는 범위를 지정할 수 있는 구문이 결여되어 있다는 점에서 비효율적이다. 본 논문에서 제시하고자 하는 모델은 사용자가 각 객체의 사용범위를 지정 할 수 있게 한다. 따라서 프로그램은 항상 필요한 객체들에 제한하여 관리하게 되므로 프로그램의 크기가 줄어든다.

기존의 객체 지향 모델들과 본 논문에서 제시하는 객체 지향 모델을 비교하기 위하여 다음과 같은 구문을 살펴보자:

이 논문은 정보통신부 대학기초연구지원사업(과제번호 96038-IT2-I1)의 연구결과임.

<sup>†</sup> 현재 동아대학교 컴퓨터공학과 조교수

<sup>††</sup> 경일대학교 산업시스템공학과 부교수

기존의 객체 지향 모델에서 이 구문은 “*obj*라는 객체에 메시지 *f*를 전송한다”는 의미로 해석된다. 해석은 메시지 *f*를 수행하기 전에 *obj*가 프로그램 내에 이미 존재한다는 것을 전제로 한다. 하지만 여기서 중요한 점은 *obj*는 메시지 *f*를 전송할 때만 필요하고 그 이전에는 프로그램내에 필요하지 않다는 것이다.

이런 관점에서 본 논문에서 제안한 객체 지향 모델에서 위 구문의 해석은 다음과 같이 바뀐다: “**obj**를 현재의 프로그램(즉, 객체의 리스트)에 동적으로 추가하고 메시지 *f*를 전송한다.”

두 해석의 차이점을 이해하기 위하여, 그림 1과 같은 객체들의 집합이 있다고 가정하자. 이 경우 A.f(a, b)의 값을 구하기 위해서는 기존의 객체지향 모델에서는 프로그램 내에 객체 A, B, C가 필요하고 다음과 같이 (1)-(4)의 순서를 거쳐 값을 구할 수 있다.(여기서 PI-E는 프로그램 P를 사용하여

expression E의 값을 구한다는 의미이다.)

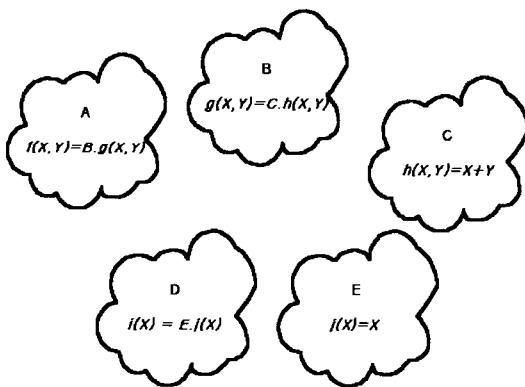


그림 1. 객체들의 집합

- (1) A, B, C  $\vdash A.f(a, b)$
- (2) A, B, C  $\vdash B.g(a, b)$
- (3) A, B, C  $\vdash C.h(a, b)$
- (4) A, B, C  $\vdash a + b$

여기서 주목할 점은 계산을 수행하는 동안 프로그램 내에 객체 A, B, C가 계속 존재한다는 것이다. 하지만 객체 A, B, C가 계속 존재할 필요는 없다. (1), (2), (3) 각각의 경우에 대해서 프로그램에 B와 C, A와 C, A와 B가 불필요하게 포함되어 있으며, (4)의 경우  $a + b$ 의 값을 구한 후, 프로그램에 A, B, C가 존재할 필요가 없다.

우리가 제안한 해석에서는 다음과 같이 프로그램 nil로부터  $A.f(a, b)$ 의 값을 구할 수 있으며, 필요한 객체는 계산과정에서 프로그램에 점진적으로 도입된다.

- (1) nil  $\vdash A.f(a, b)$
- (2) A  $\vdash f(a, b)$
- (3) B, A  $\vdash g(a, b)$
- (4) C, B, A  $\vdash h(a, b)$
- (5) C, B, A  $\vdash a + b$

이와 같이 본 논문에서 제시하는 객체 지향 모델은 항상 필요한 객체만을 프로그램에 동적으로 추가하는 방법을 사용한다. 방법의 장점은 프로그램이 실행 중 필요한 객체만을 관리하면 되므로 프로그램의 크기를 줄일 수 있다. 본 논문에서는 이와 같은 새로

운 객체 지향 모델을 정의하고 이 모델 내에서 객체에 사용범위를 부여하는 기능이 어떻게 구현되고 있는지 설명한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 논문에서 제시하는 새로운 객체 지향 모델에 대해서 정의한다. 3장에서는 2장에서 설명한 모델을 사용하여 실제 예를 보인다. 마지막으로 4장에서는 결론과 앞으로 연구해야 할 과제를 제시한다.

## 2. 객체 지향 모델 정의

이 장에서는 본 논문에서 제시하는 객체 지향 모델을 정의한다. 본 논문에서 제시되는 객체지향 모델은 기존 객체지향 모델의 핵심부분으로서 객체와 메시지 전달부분만을 사용하고 상속이나 클래스 개념은 고려하지 않는다. 본 논문에서는 모델 정의에 대한 복잡성을 줄이기 위해서 객체의 내부 펄드는 제외하고 정의한다.

다음은 본 논문에서 제시하는 객체 지향 모델을 정의하기 위한 몇 가지 기본적인 용어이다.

**정의 2.1** *nil* 은 널(null) 리스트를 의미하고,  $:::$  리스트 생성자를 나타낸다. 또  $a_1 :: a_2 :: \dots :: a_n :: nil$  은  $a_1$  부터  $a_n$  까지  $n$ 개의 원소를 갖는 리스트를 나타낸다.

**정의 2.2** 본 논문에서 사용할 구문은 변수 *Obj*와 *Expression*을 사용해서 다음과 같이 정의된다.

$$\begin{aligned} Obj &= (f(X_1, X_2, \dots, X_n) = Expression) :: nil \\ &\quad | (f(X_1, X_2, \dots, X_n) = Expression) :: Obj \end{aligned}$$

$$\begin{aligned} Expression &:= obj.Expression \\ &\quad | f(Expression, \dots, Expression) \\ &\quad | Constant \\ &\quad | Variable \\ &\quad | if\ Expression\ then\ Expression \\ &\quad \quad else\ Expression \end{aligned}$$

정의 2.2에서 볼 수 있는 것처럼 본 논문에서 제시한 객체 지향 모델에서 하나의 객체는 메소드들의 리스트로 구성되어 있으며, 객체에 전송하는 메시지인 *Expression*은 메소드, 상수, 제어문 등의 여러 구문을 사용할 수 있다. 주목할 점은 *Expression*에서

객체가 반복하여 나올 수 있다는 점이다. 즉  $a, b, c$ 가 객체일 때  $ab.c.f(d)$ 와 같은 표현이 허용된다.

**정의 2.3** 프로그램은  $obj$ 의 리스트로서 구성되어 있다.

$$\begin{aligned} \text{Program} &:= \text{nil} \\ &\mid obj :: \text{Program} \end{aligned}$$

지금까지 본 논문에서 제시한 객체 지향 모델의 문법을 정의하였다. 이번에는 본 논문에서 제시한 객체 지향 모델에 사용되는 언어의 의미에 대해 정의한다. 이를 위해서 프로그램  $P$ 와 Expression  $E$ 가 주어졌을 경우, 값을 구하는 함수  $P \vdash E$ 를 다음과 같이 정의한다.

**정의 2.4** 프로그램  $P$ 와 Expression  $E$ 가 주어질 때  $P$ 로부터  $E$ 의 값을 구하는 함수  $P \vdash E$ 는 다음과 같이 정의된다.

가) 만일  $E$ 가  $obj.E_I$ 이면

$$P \vdash E = obj :: P \vdash E_I$$

나) 만일  $E$ 가  $f(E_1, E_2, \dots, E_n)$ 이면

여기서  $P$ 는  $obj_1 :: obj_2 :: \dots :: obj_n :: nil$  이라고 하고  $obj_1$ 부터  $obj_{k-1}$ 에  $f$ 가 정의되어 있지 않고  $obj_k$ 에  $f$ 가 처음으로 다음과 같이 정의되어 있다고 한다.

$$f(X_1, X_2, \dots, X_n) = E_k$$

그러면

$$P \vdash E = P \vdash [R_1/X_1, R_2/X_2, \dots, R_n/X_n]E_k$$

여기서  $R_m = P \vdash E_m, 1 \leq m \leq n, [R_1/X_1, R_2/X_2, \dots, R_n/X_n]E_k$ 는  $E_k$ 에서  $X_1$  대신  $R_1$ 을 대입,  $X_2$  대신  $R_2$ 을 대입, ...,  $X_n$  대신  $R_n$ 을 대입한 결과를 의미한다.

다) 만일  $E$ 가 Constant  $C$ 이면

$$P \vdash E = C$$

라) 만일  $E$ 가  $\text{if } E_1 \text{ then } E_2 \text{ else } E_3$ 이면

$$P \vdash E = P \vdash E_2 \text{ if } P \vdash E_1 > 0$$

$$P \vdash E = P \vdash E_3 \text{ if } P \vdash E_1 = 0$$

정의 2.4의 가)는  $P$ 로부터  $obj.E_I$ 의 값을 구하기 위해서는,  $P$ 의 맨 앞에 객체  $obj$ 를 추가하여 생성된 객체의 리스트  $obj :: P$ 에서  $E_I$ 의 값을 구한다는 것을 의미한다.

정의 2.4의 나)는  $E$ 가 임의의 메소드  $f$ 일 경우 객체의 리스트  $P$ 의 가장 왼쪽 객체(가장 마지막으로 객체의 리스트에 추가된 객체)부터 오른쪽 객체의 순으로 검색하여  $f$ 가 처음으로 정의되어 있는  $obj_k$ 를 찾고,  $obj_k$ 의 메소드  $f$ 를 사용하여 Expression의 값을 구한다. 여기서 메소드  $f$ 의 값을 구하기 위해 필요한 인자  $E_1, E_2, \dots, E_n$ 는  $R_k = eval(P, E_k), 1 \leq k \leq n$ 와 같은 계산 방법을 사용하여 값을 구하고, 이 값을 메소드  $f$ 에  $R_1/X_1, R_2/X_2, \dots, R_n/X_n$ 와 같이 인자로 대입하여  $f(E_1, E_2, \dots, E_n)$ 의 값을 구할 수 있다.

정의 2.4의 다)는  $E$ 가 상수일 경우 계산 과정이 필요하지 않으므로 상수 자체가 계산 결과임을 보여주고 있다.

정의 2.4의 라)는 조건문의 경우  $E_I$ 의 값을 구하여 0보다 큰 경우  $E_2$ 의 값을 구하고, 0일 경우는  $E_3$ 의 구문을 계산한다는 것을 보여준다.

이와 같이 본 논문에서 제시한 객체 지향 모델은 필요한 객체를 동적으로 추가하여 사용할 수 있다. 또 동적 상속[5]을 쉽게 구현할 수 있다. 가령, 객체 A, B가 있을 때, A.B.expression은 B가 A를 상속한 후 expression의 값을 구하는 효과를 가진다. 즉, A와 B에 있는 모든 메소드를 사용할 수 있으나 같은 메소드일 경우 B의 메소드가 우선순위를 갖게 된다.

지금까지 본 논문에서 제안한 객체 지향 모델을 정의하였다. 다음 장에서는 본 논문에서 제시한 객체 지향 모델의 사용 예를 보인다.

### 3. 객체 지향 모델 사용의 예

이 장에서는 2장에서 정의한 객체 지향 모델을 사용한 실제 예를 보인다.

**예 3.1** 그림 1과 같은 객체들의 집합이 있다고 가정 할 때 expression C.h(a, D.i(b))의 값은 다음과 같은 단계를 거쳐 구한다.

1)  $\text{nil} \vdash C.h(a, D.i(b))$

처음에 프로그램의 객체 리스트는 nil이다. 그리고

현재 실행하려 하는 expression은  $C.h(a, D.i(b))$ 이다.

2)  $C \vdash h(a, D.i(b))$

expression에 있던 객체 C가 프로그램의 객체 리스트에 추가되었다. 여기서 메소드 h의 두번째 인자 D.i(b)의 값이 구해지지 않았으므로, 메소드 h의 값을 구하기 전에 인자인 D.i(b)의 값을 먼저 구한다.

3)  $D, C \vdash i(b)$

$D.i(b)$  인자의 값을 구하기 위해서 객체 D를 프로그램 객체의 리스트에 추가한다. 여기서 객체 D에서 메소드 i(b)는  $E.j(b)$ 를 계산한다.

4)  $E, D, C \vdash j(b)$

$E.j(b)$  인자의 값을 구하기 위해서 객체 E를 프로그램 객체의 리스트에 추가한다. 객체 E에서  $j(b)=b$ 이므로  $D.i(b)$ 는 b가 되고, 객체의 집합은 (2)단계와 같이 구성된다.

5)  $C \vdash h(a, b)$

객체 C에서  $h(a, b) = a + b$ 이므로, 결과는  $a + b$ 가 된다.

6)  $C \vdash a + b$

예 3.2 그림 2와 같은 객체들의 집합이 있다고 가정할 때 다음 expression을 살펴보자.

`polygon.rectangle.fillcolor(moveto(create(4, 6, 10, 20), 25, 37), Blue)`

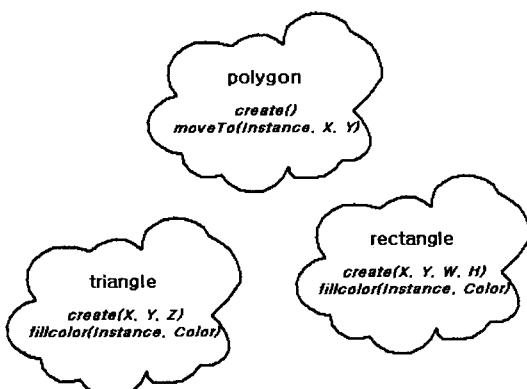


그림 2. Polygon, Triangle, Rectangle 객체

이 expression은 화면에 rectangle을 생성하고 rectangle의 좌표를 이동한 후 rectangle 내부를 푸른 색 색으로 칠한다. 이 expression은 rectangle 객체의 메소드 create, fillcolor를 사용하고, polygon 객체의 moveto 메소드를 사용한다.

본 논문에서 제시한 객체 지향 모델은 객체의 사용 범위를 정의할 수 있으므로 예제 3.2에 사용되는 Expression에서 객체의 집합을 polygon과 rectangle으로 정의하여 triangle 객체가 프로그램에 포함되지 않게 할 수 있다. 그리고 polygon.rectangle이란 표현은 polygon을 상속받은 rectangle 객체와 같은 효과를 가지게 된다. 이와 같은 본 논문에서 제시한 모델은 객체들 사이의 상속 관계가 메시지 전송 구문에서 지정될 수 있으므로, 자연스러운 동적 상속을 할 수 있다.

#### 4. 결 론

본 논문에서 제시한 객체 지향 모델은 프로그램에 사용되는 객체의 집합을 동적으로 변경할 수 있도록 하였다. 그러므로 프로그램에 항상 필요한 객체의 집합만이 유지할 수 있으므로 프로그램의 크기를 줄일 수 있다는 장점이 있다.

전통적으로 객체지향 모델에서는 재사용성을 얻는 기법에 관해 많은 연구가 이루어 졌으나 객체의 사용범위를 지정할 수 있는 기능에 관한 체계적인 연구는 거의 알려지지 않고 있다. 한편 논리언어와 함수형 언어에서는 모듈의 사용범위를 지정해 주는 기능에 대하여 활발히 연구되고 있다[9, 10, 11]. 본 논문에서 제시한 객체 지향 모델은 논리언어인 λ prolog[9]와 함수형 언어인 ML[10]의 let-endlet 구문에서 주된 아이디어를 얻었다.

본 논문에서는 내부 필드가 존재할 경우에 관해서는 고려하지 않았는데 앞으로 이런 점을 고려하여 여기 제시된 모델을 확장할 예정이다. 또 현재 논문에서 제시한 모델의 실제 구현 문제에 대해서 연구중이다. 동적으로 변하는 프로그램의 관리는 스택을 사용하여 구현할 예정이다.

#### 참 고 문 현

- Ungar, "A Shared View of Sharing : The Treaty of Orlando" In Kim W., and Lochovsky, F., editors, Object-Oriented Concepts, Applications, and Databases, Addison-Wesley, 1988.
- [ 2 ] H.Lieberman, "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems," *Proceedings of the First ACM Conference on Object-Oriented Programming Systems, Languages and Applications, SIGPLAN Notices*, vol. 21, no. 9, pp. 214-223, 1986.
- [ 3 ] D.Ungar and R.B. Smith "Self: The Power of Simplicity", *Proceedings of the Second ACM Conference on Object-Oriented Programming Systems, Languages and Applications, SIGPLAN Notices*, vol.22, no. 10, pp.227-242, 1987.
- [ 4 ] L.A. Stein "Delegation Is Inheritance", *Proceedings of the Second ACM Conference on Object-Oriented Programming Systems, Languages and Applications, SIGPLAN Notices*, vol. 22, no. 10, pp.138-146, 1987.
- [ 5 ] G.A. Agha, "ACTORS : A Model of Concurrent Computation in Distributed Systems", The MIT Press, Cambridge, MA, 1987.
- [ 6 ] Craig Chambers, David Ungar, Elgin Lee "An Efficient Implementation of SELF, a Dynamically-Typed Object-Oriented Language Based on Prototypes", *OOPSLA 89, ACM SIGPLAN*, 1989.
- [ 7 ] A.Mercaado Jr., "Hybrid : Implementing Classes with Prototypes, Masters's Thesis", *Technical Report CS-88-12, Brown University Department of Computer Science*, Province, RI, July, 1988.
- [ 8 ] David Ungar, Craig Chambers, Bay-Wei Chang, Urs-Holzle, "Organizing Program without Classes", *LISP AND SYMBOLIC COMPUTATION*, 1991.
- [ 9 ] Gopalan Nadathur and Dale Miller. "An overview of  $\lambda$  Prolog", In fifth international logic programming conference, page 810-827, August 1988
- [10] R.Harper, "Introduction to Standard ML", *School of Computer Science Carnegie Mellon University*, 1986.
- [11] Luis Monteiro and Antomio Porto, "Contextual Logic Programming", Sixth International Logic programming Conference, page 248-299, portugal 1989.

### 권 기 향



교수

관심분야 : 프로그래밍 언어, 소프트웨어 공학

### 김 지 승



학과 부교수

관심분야 : 프로그래밍 언어, 시스템 공학

1983년 서울대학교 컴퓨터공학과(학사)

1985년 미국 Georgia Tech Computer Science(석사)

1991년 미국 Duke University Computer Science(박사)

현재 동아대학교 컴퓨터공학과 조

1983년 서울대학교 산업공학과(학사)

1985년 서울대학교 산업공학과(석사)

1991년 서울대학교 산업공학과(박사)

현재 - 경일대학교 산업시스템공