Galois-격자 구조를 이용한 객체지향 분석 모델 구축과 유지에 관한 갱신 알고리즘

안 희 석[†] 전 문 석[††] 류 성 열[†††]

요 약

본 논문에서는 이산수학에서 많이 다루고 있는 Galois-격자를 이용하여 객체지향분석 모델을 구축하고 이를 유지 보수하기 위한 방법들을 제시하고, 예를 통해 객체지향분석 모델을 구축하는 것에 대해 분석하였다. 클래스 객체들과 그들 사이에 형성된 속성을 이항관계로 표현한 Galois-격자를 이용하여 관계(relation)를 정립하고, 분석단계의 클래스 계층구조에 새로운 클래스 노드를 추가할 때마다 Galois-격자구조를 점증적으로 갱신하는 알고리즘을 제안하였다. 이러한 제안은 실험을 통하여 새로운 클래스 노드의 추가는 일정한 시간내에 수행됨을 알았으며, 격자구조의 성장 속도는 클래스 노드수와 비례하며, 클래스 노드에 관련된 속성의 수가 상한치를 갖을때 점증적 알고리즘의 최악경우 복잡도는 객체수에 따라 선형적으로 증가함을 보였다. 이 결과는 객체지향 분석 모델의 이해도와 모델의 유지보수하는 추적도를 높이고, 객체지향 시스템의 장점인 클래스의 재사용 가능성을 향상시키고 클래스 계층 유지보수를 실질적으로 지원한다.

Updating Algorithms using a Galois-Lattice Structure for Building and Maintaining Object-Oriented Analysis Models

Hisuck Ahn,[†] Moonseog Jun,[††] and Sungyul Rhew[†††]

ABSTRACT

This paper describes and constructs object-oriented analysis models using Galois-lattices that we are always studying in discrete mathematics, shows fundamental approaches to maintain the models, analyzes the construction of object-oriented analysis models through good examples. Also, we define several properties of Galois-lattices that have binary relations between class objects, propose the incremental updating algorithms that can update the Galois-lattice whenever new classes are added. This proposal shows that in case of adding new class nodes the results from simulations can implement in constant time and have linearly the incremental structures in worst cases, and in that the growth rate of lattices is proportioned to class nodes in time complexity. This results can achieve the high understandability of object- oriented analysis models and the high traceability of maintenance models. Furthermore, it is possible to make more efficient performances of class reusability in advantages of object-oriented systems and support truly the class hierarchical maintenances.

1. 서 론

80년대 소프트웨어의 위기에 대한 관심이 대두되면서 소프트웨어공학의 발전이 가속화되었다. 구조적 프로그래밍의 발전은 소프트웨어의 재사용이나 문서화에 대한 관심을 유발하였다. 개발의 생산성과 품질인자는 소프트웨어 개발의 성패 요인으로 인식되기 시작하였으며, 객체지향개발 방법론은 이러한 인식을 바탕으로 급격한 발전을 가져왔다[2]. 특히, 객체기반(Object-based)의

† 정 회 원 : 숭실전산원 부원장
†† 정 회 원 : 숭실대학교 컴퓨터학부 부교수
††† 정 회 원 : 숭실대학교 컴퓨터학부 교수

논문접수 : 1995년 2월 6일, 심사완료 : 1995년 5월 18일.

소프트웨어 개발에 비하여 객체지향화는 상속성을 이용한 소프트웨어 모듈의 재사용성을 극대화할 수 있기 때문에 매우 관심 있는 분야로 대두되었다. Smalltalk-80은 객체지향의 개념을 바탕으로 클래스 라이브러리의 재구축과 상속성을 지원하는 초창기의 프로그래밍 언어로써 인정받았다. 하지만, 실제 클래스간의 행위 접속 관계를 바탕으로 계층화를 시켰을 때, 상속성 계층 구조와의 이질성이 문제시되었다[8]. 이러한 연구는 비단 Smalltalk-80이라는 한정된 프로그래밍 언어에만 국한된 것이 아니라, 객체지향 프로그래밍에 대한 문제점 제기 및 포괄적인 상속구조의 처리를 위한 방안 연구로 인식되어질 수 있다.

본 논문에서 행위접속관계 계층화란 클래스간의 접속관계를 부분정렬의 형태로 계층화한 것을 의미하며, 상속성이 공급자의 메카니즘을 기본으로 한 개념이라면 접속관계는 소비자의 메카니즘을 기본으로 한 개념이다[3]. 본 논문에서 제안하는 Galois-격자의 기본 개념[6, 7]은 객체지향 모델 구축과 유지를 위한 공통클래스를 추출하는 최적화 단계를 수행하는데 있어서 간단한 구조체로 이용할 수 있음을 보여주고 있다. 인터페이스 일치에 중점을 둔 클래스 인터페이스 계층구조의 구축은 자연스럽게 구성된 Smalltalk-80 클래스[3, 8, 13]를 갖고서 설명하였으며, 클래스 인터페이스의 특성[3, 8]에 기반을 둔 점증적 알고리즘을 제안하였다.

2장에서는 이진 관계에 의한 Galois-격자구조 구축을 이론적으로 다루었으며, 3, 4, 5장에서는 실제적인 예제로서 Smalltalk-80[3, 8]에 대한 클래스 구조를 격자 구조로 작성하였다 이러한 구조는 객체지향 분석 모델 유지보수에 응용될 수 있는 갱신 알고리즘의 배경이 되었다. 마지막 장에서는 갱신 알고리즘의 시간 복잡도를 측정하는 실험 결과를 보였다.

2. 이진관계를 갖는 Galois-격자 구축

객체지향 분석 모델[14]의 유지보수를 하는데 Galois-격자구조를 응용하기 위하여 이진관계 R의 행렬 표현을 Galois-격자 구조로 나타내고 그 특성을 살펴보기로 하자. Galois-격자는 그래프 구조 형태와 복잡한 이진 관계를 갖고 있다.

〈표 1〉 이진관계 R의 행렬 표현
〈Table 1〉 Matrix representation of binary relation R

	at	atPut	atAllPut	first	last	addFirst	addLast	keys	values
Collection	1	0	1	0	0	1	0	1	0
Set	1	0	1	0	0	0	1	0	1
Bag	1	0	0	1	0	0	1	0	1
Linked List	0	1	1	0	0	1	0	1	0
Array	0	1	0	0	1	0	1	0	0

〈표 1〉의 이진관계 R은 행렬을 객체 지향 이진관계로 표현할 수 있다. 이러한 이진관계는 실체들의 유한집합 V와 그것에 대한 속성들의 유한집합 W로 구성된다. 이후부터는 간략히 표현하기 위하여 Collection, Set, Bag, Linked List Array는 1, 2, 3, 4, 5로 대치하며, at, atPut, atAllPut… 은 a, b, c … h, i로 대치한다. 〈표 1〉은 V={1, 2, 3, 4, 5}이고 W={a, b, c, d, e, f, g, h, i}이라 할 때, 이들의 이진관계 R을 행렬로 표현한 것이다. (그림 1)은 〈표 1〉을 Galois-격자로 구축한 것을 나타낸 것이다. (그림 1)의 Galois-격자내의 각 노드는 객체 지향 모델에서 클래스(Class)를 나타내며[3, 6], 각 노드들 사이의 간선(edge)은 클래스의 속성이 하향적으로 상속되는 것을 나타낸다.

두 개의 유한집합 V, W 상에 관계 R이 주어져 있으면 이 관계 R에 대해 Galois-격자가 유일하게 존재한다. 격자(lattice) L의 원소는 순서쌍들이며 다음과 같이 나타낼 수 있다.

$L=\{(X, X') \mid W \in \mathbf{P}\{V), W' \in \mathbf{P}(W)\}$ 격자 L의 원소들은 모두 완전한 순서쌍(complete couple)이어야 한다. 어떤 순서쌍 (x, x')이 다음 두 가지 조건을 만족하면 관계R에 대하여 완전하다(complete)라고 정의한다.

① $X'=f(X)$ $where$ $f(X)$
$=\{x' \in W \mid \forall, \in X, xRx'\}$

② $X=f'(X')$ $where$ $f'(X')$
$=\{x \in V \mid \forall,' \in X', xRx'\}$

함수들의 순서쌍 (f,f')을 $\mathbf{P}(V)$와 $\mathbf{P}(W)$ 사이의 Galois 연결(connection)이라 한다.

이런 이항 관계들에 대한 Galois-격자 G는 다음과 같은 반순서를 갖는 완전한 순서쌍(complete couple)들의 집합이다.

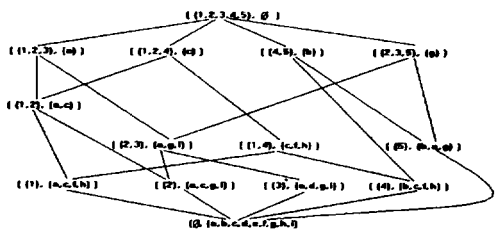
$C_1 = (X_1, X_1')$, $C_2 = (X_2, X_2')$이라 할 때,
$C_1 < C_2 \Leftrightarrow X_1' \subset X_2'$

결국, Galois-격자에서는 집합 X와 X' 사이에 쌍대(dual) 관계가 존재한다.

즉 $X_1' \subset X_2' \Leftrightarrow X_2 \subset X_1$ 이다. 그러므로 $C_1 < C_2 \Leftrightarrow X_2 \subset X_1$ 이다. 반순서(partial order)는 다음과 같은 방법으로 그래프를 생성하는데 이용된다. 만약 $C_1 < C_2$ 이고 $C_1 < C_3 < C_2$ 와 같은 원소 C_3 가 속안에 존재하지 않으면, 간선(edge) (C_1, C_2)가 존재한다. 이 때, C_1 을 C_2 의 부모(parent), C_2 를 C_1 의 자식(child)라 부른다. 이런 그래프를 일반적으로 Hasse diagram 이라 한다. Hasse Diagram을 그릴 때, 일반적으로 간선의 방향은 하향이다.

각 클래스 노드들을 나타내는 $C_i \in \mathbf{P}(V) \times \mathbf{P}(W) \subseteq R$ 은 Galois-격자 위상에서 반순서관계[15]로 정의된다.

이러한 격자구조의 형태는 객체지향 분석 모델에서 가장 간단하고 중요한 이론으로 정립될 수 있으며, 객체지향 분석 모델 구조에 대한 유지보수 관계로 해석될 수 있다.



(그림 1) 〈표 1〉의 관계에 대한 Galois-격자 구조
(Fig. 1) Galois-lattice structure of Table 1

Galois-격자 G의 각 원소인 클래스 노드 C에 대하여, Sup(C)는 최소상한으로서 최상위 클래스 노드라고 정의하고, inf(G)는 최대하한으로서 최하위 클래스 노드로 정의하자. 그때, 주어진 Galois-격자 구조에서 예로서 주어진 클래스들의 범위 C={ C1, C2, C3, }이며, C1 < C3 < C2인 격자구조에서 다음과 같은 결과를 쉽게 얻을 수 있다.

sup(C)={C1} : 최상위 클래스 노드
inf(C)={C2} : 최하위 클래스 노드

3. 상속관계의 Galois-격자 관계

실제적인 예제로서 다음 (그림 2)와 같은 Smalltalk 80[3, 8]에 이용되는 구조형태 중심으로 Galois-격자 구조를 이용하여 객체지향형태의 구조로 해석하고, 분석하는 함으로서 객체지향 분석 모델에 대한 상속관계를 설명하여 보자.

```
Object
    Behavior
        Class
        MetaClass
    BitBlt
        CharacterScanner
        Pen
            Animation
            Commander
    Boolean
        False
        True
    ClassBrowser
    ClassHierarchyBrowser
    ClassReader
    Collection
        Bag
        IndexedCollection
            FixedSizeCollection
                Array
                Bitmap
                ByteArray
                    CompiledMethod
                    FileControlBlock
                Inteval
                String
                    Symbol
            OrderedCollection
                Process
                SortedCollection
        Set
            Dictionary
                IdentityDictionary
                    MethodDictionary
                SystemDictionary
            SymbolSet
    Compile
    Context
    CursorManager
        NoMouseCursor
    DemoClass
    Dictionary
    DiskBrowser
    Dispatcher
        GraphDispatcher
        PointDispatcher
        ScreenDispatcher
        ScrollDispatcher
            FormEditor
            ListSelector
            TextEditor
                PromptEditor
            TopDispatcher
    DispatchManager
    DisplayObject
        DisplayMedium
            Form
                DisplayScreen
                SelectForm
        DisplayString
        InfiniteForm
    File
    Inspector
        Debugger
        DictionaryInspector
    Magnitude
        Association
        Character
        Date
        Number
            Float
            Fraction
            Integer
                LargeNegativeInteger
                LargePositiveInteger
                SmallInteger
        Time
    Menu
    Message
    Pane
        SubPane
            GraphPane
            ListPane
            TextPane
        TopPane
    Pattern
        WildPattern
    Point
    Prompter
    Rectangle
    Stream
        ReadStream
        WriteStream
            ReadWriteStream
                FileStream
                TerminalStream
    StringModel
    TextSelector
    UndefinedObject
SwappedOutObject
```

(그림 2) Smalltalk-80 의 클래스 객체에 대한 계층 구조.
(Fig. 2) A hierarchical structure of class objects for Smalltalk-80.

(그림 2)의 기본적 구조에서 추출된 이진관계 행렬을 이용하여 다음과 같은 〈표 2〉의 예제를 갖고서 Galois-격자구조를 구축하여 설명하여 보자.

〈표 2〉 클래스 객체에 대한 행렬의 간단한 예.
〈Table 2〉 An Example for matrix representative of a binary relation R.

	Collection	Set	Bag	Sequenceable Collection	Dictionary	Linked List	Array
isEmpty	1	1	1	1	1	1	1
size	1	1	1	1	1	1	1
include	1	1	1	1	1	1	1
add	0	1	1	0	1	1	0
remove	0	1	1	0	0	1	0
minus	0	1	0	0	1	0	0
addWith Occurrences	0	0	1	0	0	0	0
at	0	0	0	0	1	0	1
atPut	0	0	0	0	1	0	1
atAllPut	0	0	0	0	0	0	1
first	0	0	0	1	0	1	1
last	0	0	0	1	0	1	1
addFirst	0	0	0	0	0	1	0
addLast	0	0	0	0	0	1	0
keys	0	0	0	0	1	0	0
values	0	0	0	0	1	0	0

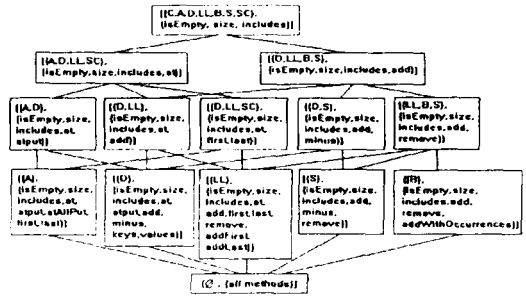
이러한 행렬은 실체의 원소 V′와 속성의 원소 W′로 구성된 클래스 노드 C=(V′, W′)로 각각의 클래스 노드를 표현할 수 있다. 새로운 클래스 노드가 추가됨으로서 다른 노드에 영향을 주게 되고, 삭제하게 되는 경우 영향받는 노드와 영향받지 않은 노드의 격자 구조를 이론적으로 다루어 보자. 클래스 노드 C=(V′, W′)에 있어서 공유되는 실체와 속성에 대한 상속관계는 다음과 같다.

$$V'' \le V' \le V,\ W'' \le W' \le W$$
$$C_1 = (V'', W'')$$

상속받는 클래스 노드들은 C=(V′, W′)로 부터 새로 생성되는 노드 C1=(V″, W″)로 변환되며, 위상이 높은 것과 낮은 것과의 유지보수 관계를 정의할 수 있으며, 생성되는 노드 클래스를 이론적인 정의에 의해서 정립할 수 있다. 클래스 노드 C=(V′, W″)에서 생성되는 클래스의 새로운 실체원소 V″와 속성원소 W″로 구성되며, 다음과 같이 정의할 수 있다.

$$V'' = \{v \in V \mid R(v) = V'\}$$
$$W'' = \{w \in W \mid R^{-1}(w) = W'\}$$

(그림 3)은 추상적인 이름을 갖는 상속관계의 Galois-격자 구조를 설명하고 있다. Indexed Collections의 클래스 노드(Φ, {at})는 position/index에 의해 상속관계를 가질 수 있는 가능성을 갖고 있다. 클래스 노드(Φ, W″)를 가진 클래스 노드의 경우와 일치하는 노드이며, W″와 노드의 정확하게 선행자에서 의미하는 것을 구현하는 클래스는 존재하지 않는다. 즉, 추상적인 이름을 클래스에 부여함으로서 상속관계 구조를 쉽게 분석할 수 있으며, 유지보수해야 할 영역을 알게 된다. 이러한 관계는 격자의 성격을 클래스 개념에서 쉽게 얻을 수 있는 결과이며, 실제적으로 C++언어에서 많이 이용되고 있다. 다음은 점증적인 갱신 알고리즘을 설계하기 위하여 관련된 이진관계와 Galois-격자 구조를 이용하여 클래스 노드가 추가되는 경우 유지 보수에 대하여 고려한다.



(그림 3) 표 2에 대한 Galois-격자 구조 변환
(Fig. 3) Galois-lattice for the relation in Table 2

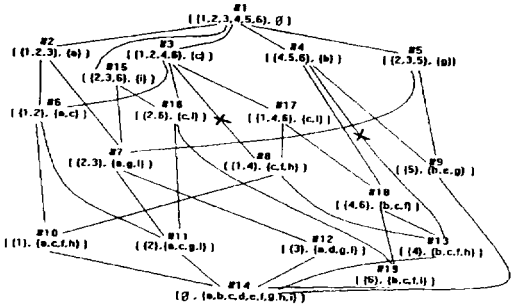
4. 점증적인 갱신 격자 설계

4.1. 점증적 갱신 격자구축

객체지향분석에서 이용되는 것은 격자구조의 Hasse 다이어그램이다. 이런 상속관계를 이용하여 점증적인 갱신 알고리즘을 설계하기 위한 기본적인 이론을 작성하고자 한다.

점증적인 갱신 알고리즘을 작성하기 위하여 그래프 이론을 이용하여 설명하여 보자. 그래프 [15]는 정점들의 집합과 간선들의 집합으로 구

성되어진다. 유사한 방법으로 격자의 구조는 클래스를 원소로 하는 정점들과 이것들을 연결하여 생성되는 속성들로 이루어지는 간선들의 집합으로 고려할 수 있다. 클래스의 집합 E에 추가함으로서 생성되는 새로운 클래스 x*를 체계적으로 작성하고, 유지보수할 수 있는 구조를 갖게 된다. 이러한 구조는 전체 구조를 하나하나 재 생성하는 방법보다는 부분적이며, 효율적으로 갱신하는데 중점을 두고 고려할 수 있다.



(그림 4) 멤버 함수 {b, c, f, i}을 구성하고 있는 클래스 6을 추가한 후 수정된 Galois 격자 구조.

(Fig. 4) A modified Galois-lattice resulting from adding elements 6 described by the set {b, c, f, i}

새로운 실체 원소 x*=6를 추가함으로서 얻을 수 있는 속성은 다음과 같다고 가정하고 수행 구조를 고려해 보자.

$$f*(\{x*\}) = \{b, c, f, i\}$$

새로운 실체원소 x*를 추가하는 경우, 속성 f*({x*})가 구조에서 작성된다. 새로운 클래스 노드 ({x*}, f*({x*}))=({6}, {b, c, f, i})가 추가된 후의 갱신된 Galois-격자 구조는 (그림 4)와 같다.

Galois-격자의 구조 (그림 4)에서 의미하는 내용은 추가된 클래스 노드에서 영향을 받은 클래스는 새로운 멤버함수 속성들을 포함하는 격자를 부분적으로 작성하는 결과를 볼 수 있다. 여기서, 얻어진 Hasse 다이어그램의 노드는 객체지향 분석 모델에서 클래스와 그것에 포함된 멤버함수 속성들로 구성된 객체로서 표현되고 있음을 알 수 있다. 다시 말하면, 주어진 격자 그래프 G에서 새로운 노드와 간선을 추가하고 영향받은 간선들을 삭제하고, 부분적으로 수정함으로서 새로운 격자 구조 G*를 얻을 수 있다. (그림 4)에서 삭제된 간선은 "×"로서 표현하였다. Galois-격자를 이해하기 쉽게 표현하고, 연관된 객체지향 분석 모델 유지보수 이론에 관련된 새로운 이론을 얻기 위하여 점증적인 갱신 알고리즘을 연속해서 다음 절에서 구현하였다.

5. 갱신 알고리즘 설계

객체지향 분석 모델 유지보수를 위한 Galois-격자 구조에 대한 갱신 알고리즘을 작성하여 보자. 우선, 격자 구조에 대한 성격을 정의하고, 추가되는 클래스에 대하여 갱신되는 격자의 구조를 알고리즘으로 전환할 수 있다. 격자 구조에 대한 기본적인 갱신 수행과정은 집합 V의 원소를 갱신하고, 집합 W에 대한 새로운 노드를 생성함으로서 격자 그래프 G에 존재하는 노드들을 재구성하는 것이다. 이러한 수행과정에서 다음과 같은 중요한 노드 변수명을 이용하여 갱신 알고리즘을 이해하기 쉽게 작성할 수 있다.

구체적으로, 클래스 노드 (Y, Y′)는 주어진 원래 격자 그래프 G의 클래스 노드라고 정의하고, (X, X′)를 격자 생성 그래프 G*의 클래스 노드라고 정의하면 갱신시 발생하는 다음 몇 가지 노드를 분류할 수 있다.

① 새로운 노드(New) : X′의 집합이 격자 그래프 G에 존재하지 않은 새로운 노드

② 원래의 노드(Old): X와 X′가 G와 같은 노드 존재하는 노드

③ 수정된 노드(Mod): X′는 G에 존재하지만, X는 수정되어 G*에 존재하는 노드

(Y, Y′)이 G에 있고, X′=Y′인 조건에서, 수정된 노드의 X는 항상 Y∪{x*}와 같다. 그리고 원래의 노드 (Y, Y′)는 Y′⊆f*({x*}) 인 조건에서 항상 성립함을 알 수 있다. 이것에 대한 예제는 (그림 4)을 통해서 쉽게 이해할 수 있다.

클래스 노드 15에서 19까지의 새로운 노드 x*를 포함하는 X가 존재하며, (Y, Y′) ∈ G인 모든 노드들은 G*의 수정된 노드를 수정 생성하게 된다. 그래프 격자 G에 속해 있는 다른 클

래스 노드들은 G*의 원래 노드들을 그대로 이용하게 된다.

새로운 노드을 생성하는 것은 더욱 복잡하게 된다. G*의 새로 구성된 X′는 격자 그래프 G에서 아직 표현되지 않은 Y′의 집합과 연관된 f*({x*})으로부터 추론된 집합이 된다. 이러한 방법으로 추론해 나가면 특별한 새로운 형태의 노드 클래스가 형성되는 Galois-격자로 유지보수 된다.

(그림 4)의 예제처럼 새로운 노드 15에서 X′의 { i }는 노드 19의 f*({x*}) = {b, c, f, i}에서 영향을 받아서 생성된 노드임을 알 수 있다. 이러한 노드들 중 유일하게 하나 존재한다. G*안의 새로운 집합인 X′는 (W≠W*일 때, W와 f*({x*})는 제외) G안에 이미 존재하는 어떤 집합 Y′와 f*({x*})와의 교집합이 된다. 예를 들면 X′={i}인 노드 15는 7노드의 집합 Y′={a,g,i}와 f*({x*})={b,c,f,i}와의 교집합에 의해 만들어진다. (그림 4)에서 New 노드 15, 16, 18, 19에 대해서 생성된 노드들은 각각 7, 11, 8, 13, 14으로 객체지향 분석 모델에서 상속되는 관계를 유지한다.

초기에 주어진 클래스 노드를 Old 노드라고 정의되면, 이들중 어느 노드가 생성자 역할을 해서 새로운 New 노드를 생성하게 된다. Old 노드 중에는 두 가지 형태로 분류할 수 있는데 생성자 역할을 도와주는 클래스 노드가 있는 반면, 원래 그대로 존재하는 클래스 노드가 존재한다. 생성자 노드 (Y, Y′)는 G*에서 New 노드 (X, X′)의 생성자 역할을 OldGen 노드라고 하며, 다음과 같이 간결하게 정의할 수 있다.

$$(Y, Y') = \inf\{(Z, Z') \in G \mid X' = Z' \cap f^*(\{x^*\})\}$$

격자 그래프 G에서 OldGen 노드와 New 노드 사이에 일대일 대응 관계가 성립한다. 이것의 의미는 생성자 노드의 성질을 이용하여 Old 노드를 발견함으로서 새로운 노드를 생성할 수 있다. 이것을 이용하여 갱신 알고리즘을 작성할 수 있다. 이러한 알고리즘에서 객체지향 분석 모델 유지보수를 위한 Galois-격자 구조에서 실체와 속성으로 구성된 클래스 노드를 적절한 위상에 추가하고, 삭제하는 갱신 알고리즘이 체계적으로

```
{ 갱신 알고리즘 }
Begin
1  If sup(G) = ∅,  then  (sup(G)에 갱신된 격자구조인 sup(G′)을 구축)
                   else  (새로운 노드를 생성)
2  C[i] ← { H: ‖ X(H) ‖ = i }
          //* X′에서 동등한 속성의 수를 갖는 클래스 노드 *//
3  C′[i] ← ∅
4          //* 오름차순의 노드수에 의해서 순서적으로 C′[i]를 수행*//
5  For i = 0  to  ‖V‖ do
6     For ( C[i]안에 있는 각각의 H 노드 )
7        If X′(H)⊆f′(f(x′))  then ( 수정된 Mod 노드 수행 )
8                     X(H) ← x* ∪ X(H)
9                     C′[i] 에 H를 추가한다.
10                    IF X′(H)=f′(f(x′))  Then exit
11          else ( 수행된 노드 Old을 수행 )
12                Int ←  X′(H) ∩  f′(f(x′))
13                   If [int가 C′([‖Int‖])안에 존재하지 않은 경우]
                        Then  ( H는 OldGen 노드임 )
14                           새로운 노드 Hn = (X(H) ∪ {x′})를 생성하고,
                      C′[‖Int ‖]에 Hn 를 추가
15                    Hi 에서 H에로의 간선 추가연결
16                    Hi 보다 적은 cardinality를 갖는 C′내의 노드들에
                       대해서 Hn에 대한 간선들을 수정함
17       If int = f′(f(x′))  then exit
18       End_if
19    End_if
20 End_For
21 End_For
22 End
```

성립되어야 한다.

초기 생성된 격자의 각 노드(클래스)들은 내포하고 있는 속성들을 모두 다 새로이 생성하는 것이 아니고 각 노드의 parent node(super class)에 있는 것을 상속받고 없는 것만 새로이 생성한다. 따라서 이러한 상속관계를 고려하지 않고 초기 생성된 격자에 새로운 노드를 추가할 경우, 중복된 속성생성을 유발할 수밖에 없고, 이렇게 될 경우, 소프트웨어 유지보수에 악영향을 미칠 수밖에 없다.

먼저 초기 생성 격자에서 각 속성들이 어떤 클래스에서 생성되는 지를 살펴보자. a는 2번, b는 4번, c는 3번, d는 12번, e는 9번, f는 8번, g는 5번, h는 8번, i는 7번에서 각각 생성됨을 알 수 있다. 따라서 나머지 노드에 표현된 속성은 이들 클래스에서 속성을 상속받은 것이다.

이러한 상속관계에 대한 고려없이 초기 생성 격자에 ({6}, {b,c,f,i})를 추가한다고 가정해보자. 이것은 상한과 하한사이에 다른 노드들과의 연관관계와 상관없이 표현할 수 있다. 이런 관계에서 속성 b,c,f,i는 구현상으로는 가장 간단하지만 새로이 추가되는 노드와 함께 초기 생성 격자의 4번, 3번, 8번, 7번에서 중복됨을 알 수 있다. 만약 새로운 노드를 추가할 때마다 이와 같이 중복된 속성을 양산한다면, 클래스의 상속관계를 명확히 알 수 없음은 물론, 유지보수를 하는데 막대한 어려움을 초래하게 될 것이다. 이와 같은 중복된 속성의 생성을 방지하기 위해서는

기존의 클래스 계층구조에서 새로이 추가되는 클래스 노드가 클래스 상속구조를 적절히 반영시키도록 하는 것이 필요하다. 점증적 갱신 알고리즘은 ({6}, {b,c,f,i})이 추가될 때, 단순히 기존속성의 중복생성만 이루어지지 않게 한 것이다. 즉, b의 경우는 4번, c와 f의 경우는 7번, i의 경우는 6번에서 상속받게 하면 된다. 그러나 7번 노드의 경우에는 c,f이외에 h를 포함하고 있고, 6번 노드의 경우 i이외에 a,g를 더 포함하고 있어서 결과적으로 새로운 노드는 b,c,f,i이외에 a, g,h를 더 상속받는 결과를 초래한다. 그리고 1,2, 3,4,6,7번 노드의 실체에 6을 더 추가해야 됨은 물론이다.

Galois격자 구조는 그 정의에 따라 최적의 클래스 상속구조를 나타낸다. 따라서 Galois격자구조를 이용하면 위와 같은 불필요한 속성상속을 배제한 최적의 클래스 상속구조를 나타낼 수 있다. 새로운 노드의 속성이 기존 클래스에 존재한다면 Galois격자 원리에 입각해서 필요에 따라 중간 노드를 생성해서라도 이를 최적의 상태로 상속받을 수 있도록 하고, 만약 기존 클래스에 속성이 존재하지 않는다면 그 속성만 새로이 추가되는 클래스에서 생성되도록 한다. 그리고 중간 노드와 연관된 edge도 Galois격자 원리에 의해 연결시킨다.

이러한 Galois-격자 구조를 증대시키는 갱신 알고리즘은 O(n) 시간 복잡도 속에서 수행한다. 초기 격자는 어떤 내용도 갖고 있지 않은 (∅, ∅)인 노드만을 갖고 있다고 가정하고, 제안된 갱신 알고리즘을 수행할 수 있다. 이것은 E=E′=∅의 의미를 갖으며, 새로운 원소를 추가함으로서 갱신 알고리즘은 다른 형태의 격자를 작성할 수 있음을 의미한다. 이진관계에서 한쪽의 위상은 모든 원소들을 포함하고 있고, 다른 위상은 원소가 하나도 없는 형태는 두 가지 노드 (E, ∅)와 (∅, E′)를 생각할 수 있다. 갱신 알고리즘의 기본은 주어진 격자 그래프 G에서 갱신된 그래프 G*를 생성하는 것으로, 추가된 새로운 W와 그것에 연결된 노드들 상에서 변화되는 결과를 이진관계 형태를 새로운 구조를 갱신하는데 있다. 새로운 노드는 수행하는 과정에서 추가되고자 하는 노드을 의미하며, 격자 구조에서 생성자의 역할을 하고 있다. 이것에 대한 시간 복잡도 분석은 다음 절에서 이어진다.

6. 알고리즘 분석

갱신 알고리즘은 격자 그래프 G의 모든 노드 클래스를 생성하는 데 영향을 줄 수 있으며, 부분 그래프를 갱신하는데 거의 대부분의 시간을 수행하고 있다. 중간 노드의 추가인 경우 제한적인 범위 안에서 갱신되고 있음을 알 수 있다. W와 f*({x*}) 사이에 공통부분인 노드가 존재하지 않은 경우 깊이우선 탐색 알고리즘과 경로 절단 알고리즘에 의하여 갱신된다.

알고리즘의 수행은 x′∈W를 포함하는 가장 작은 노드상에서 반복적으로 수행되며, 모든 x′∈f*({x*})으로 시작되는 노드를 시작점으로 위에서 아래로 수행하며, 깊이 우선 탐색 알고리즘에 의하여 수행한다. 분명한 사실은 수행되는 노드에서 f*({x*})와 공통되는 노드는 적어도 x′를 포함하고 있음을 알 수 있다. 이러한 격자에 대한 유지 보수 관계는 소프트웨어 공학에서 해석하기 어려운 점들을 격자를 이용하여 쉽게 표현할 수 있으며, 발생되는 모든 종류의 경우를 고려하고 있다는 점에서 본 연구에 중점을 두고 있다.

모의 실험은 변화되는 격자 형태와 처리 시간을 중심으로 수행하였다. 새로운 실체 x*을 6으로 하고, 속성은 f({x*})은 V={a, b, c, d, e, f, g, h, i}에 대한 멱집합 2^9=512개중 공집합과 V 자신을 제외한 나머지 510개로 하여 ({x*}, f({x*}))쌍 510개의 노드에 대해 다음과 같은 환경에서 실험을 했다.

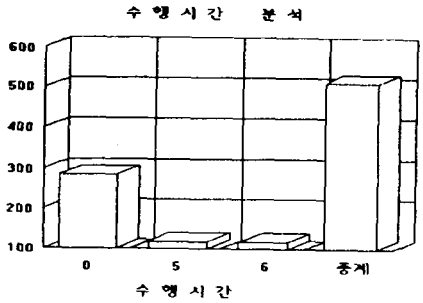
실험환경
- 하드웨어 : IBM 호환 PC (CPU=>intel 80486 DX2 66Mhz)
- 운영체제 : MS-DOS 6.0
- 컴파일러 : Turbo C V2.0

실험은 수행시간과 새로이 작성되는 노드수에 사항을 분석하는 것을 목표로 수행했다.

먼저 수행시간은 510개의 노드수가 다음과 같이 1/100, 5/100, 6/100초의 결과를 갖는 세 가

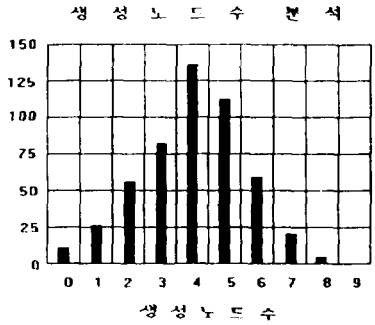
지 유형으로 분류되고 따라서 이들은 일정한 시간 내에 갱신이 이루어진다는 것을 알 수 있다.

수행시간	실험노드수
1	288
5	110
6	120
총 계	510



(그림 5) 수행시간 분석
(Fig. 5) Analysis of execution time

생성노드수	실험노드수
0	12
1	26
2	57
3	82
4	136
5	112
6	59
7	22
8	4
9	0
총 계	510

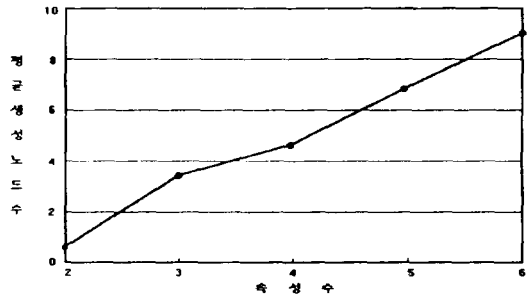


(그림 6) 생성노드수 분석
(Fig. 6) Analysis of creation nodes

새로이 생성되는 노드의 수는 최대 8개 이었으며 위와 같은 결과를 얻을 수 있었다.

또한 실체 x*를 6으로 고정시키지 않고 6, 7, 8, 9, A, B, C ……와 같이 변화시키면서 각각에 대응되는 다른 속성 f({x*})를 적용하여 나올 수 있는 클래스를 Galois-격자에 계속 추가시켜서 각 상태별로 새로이 생성되는 노드의 수를 살펴 본 결과 다음 표와 같이 속성 f({x*})의 갯수가 2개인 경우 평균 1.333개 3개인 경우 평균 2.75 , 6개인 경우 평균 8.5개의 새로운 노드가 생성되었다. 이를 통해서 아래 그래프처럼 속성의 갯수에 대한 평균 생성노드수가 선형적으로 증가됨을 알 수 있었다. 이와 같은 실험을 통하여 새로운 객체의 추가는 시간적 측면에서 객체수와 비례하며 객체에 관련된 특성의 수가 상한치를 가질 때 점증적 알고리즘의 최악경우분석은 객체 수에 따라 선형적으로 증가함을 알 수 있다.

속성수	평균 생성 노드수
2개	1.333333
3개	2.75
4개	4.333333
5개	4.416667
5개	8.5



(그림 7) 속성변화에 따른 생성노드수
(fig. 7) The number of Creation nodes for each attributes.

7. 결 론

본 논문은 Galois-격자 갱신을 위한 점증적 알고리즘을 제안하고 분석하였다. 이에 따른 방식 설계는 이진 관계와 Galois-격자 개념에 기반을 두고 연구되었고 이를 통해서 격자에 새로

운 객체를 추가할 때 Galois-격자의 이론에 기초한 상속관계를 유지하지 않고 구현의 편리성만을 고려하여 클래스 계층구조를 생성하게되면, 중복된 속성을 갖게 되거나 불필요한 속성을 상속받게 되는 결과를 초래하게 된다는 것을 알 수 있었다. 이 논문의 점증적 알고리즘은 실험을 통하여 격자에 새로운 객체를 추가할 때 전체 객체 수에 비례하여 시간이 소요됨을 알 수 있었다. 실험을 통하여 격자에 새로운 객체를 추가할 때 전체 객체 수와 비례하여 시간이 걸림을 알 수 있었다. 뿐만 아니라 실체를 고정시키지 않고 속성과 함께 변화시켜서 나올 수 있는 클래스를 Galois-격자에 연속적으로 추가 시킬 때 속성 수에 대한 평균 생성 노드 수가 선형적으로 증가함을 알 수 있었다. 이 결과는 객체지향분석모델의 이해도와 모델의 유지보수하는 추적도를 높이고, 객체지향시스템의 장점인 클래스의 재사용 가능성을 향상시키고 클래스계층유지보수를 실질적으로 지원한다. 그리고 추가되는 클래스들을 효율적인 클래스 계층구조로 구축될 수 있음을 보여주었다. 그러나 클래스 추가에 따라 새롭게 생성되는 클래스 노드들과 이와 관련된 하위레벨의 클래스 노드들과의 선분(edge)을 연결하는 알고리즘이 매우 복잡하여 이에 따른 개선이 필요하다. 또한 점증적으로 구축된 모델이 기존 설계 모델과 구현 모델에 미치는 추가적인 파급효과를 분석하여 일관성 있는 모델을 효율적으로 구축하는 방법에 관한 추가적인 연구가 요구된다.

참 고 문 헌

[1] P. Bergstein and K. J. Lieberherr. "Incremental class dictionary learning and optimization", in proceedings of ECOOP '91, ed. Sprin -ger Verlag, Geneva, Switzerland, pp. 377-395, 1991.

[2] Peter Coad and Edward Yourdon, in Object-Orentied Analysis, Prentice Hall, 2nd. edition, 1991.

[3] William R. Cook, "Interfaces and Specification for the Smalltalk80 Collection Classes", in Proceedings of OOPSLA '92, pp. 1-15, ACM Press, Vancouver, B.C., Canada, pp. 18-22, 1992.

[4] Brad J. Cox, "Planning the Software Revolution", IEEE Software, Vol. 7(6), pp. 25-35, November 1990.

[5] J. H. Gennari, P. Langley and D. Fisher, "Models of Incremental Concept Formation",in Machine Learning: Paradigms & Methods, ed. J. Carbonell, MIT Press, Amsterdam, Netherlands. pp. 11-62, 1990.

[6] R. Godin, E. Saunders and J. Gecsei, "Lattice Models of Brows-able Data Spaces", Journal of Information Sciences, Vol. 40, pp. 89-116, 1986.

[7] R. Godin, R. Missaoui and H. Alaoui, "Learning Algorithms Using a Galois Lattice Structure", in Proceedings of the Third International Conference on Tools for Artificial Intelligence, IEEE Computer Society Press, San Jose, CA, pp. 22-29, 1991.

[8] Glenn Krasner, Smalltalk-80, Bits of History, Words of Advice, Addison-Wesley Publishing Company, 1983.

[9] Bertrand Meyer, in Object-Oriented Software Construction, ed. Prentice Hall International, 1988.

[10] Ali Mili, Noureddine Boudrigua and Fathi Elloumi, "The Lattice of Specifications: Applications to a Specification Methodology", Formal Aspects of Computing, Springer-Verlag, 1992.

[11] Hafedh Mili, John Sibert and Yoav Intrator, "An Object-Oriented Model Based on Relations", Journal of Systems and Software, Vol. 12, pp. 139-155, 1990.

[12] Harold Ossher and William Harrison, "Combination of inheritance Hierarchies", SIGPLAN Notices, Vol. 27, No. 10,

Vancouve B.C. (Canada), 1992. Proceedings of OOPSLA '92, pp. 25-40, 1992.

[13] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy and William Lorensen, in Object-Oriented Modeling and Design, Prentice Hall, 1991.

[14] R. Wille, "Restructuring Lattice Theory: A Approach Based on Hierarchies of Concepts", in Ordered Sets, ed. I. Rival, Reidel, Dordrecht-Boston, pp. 445-475, 1982.

[15] 전문석, "이산 수학", 홍릉과학출판사, 1992.

안 희 석

1978년 숭실대학교 전산학(학사)
1988년 숭실대학교 전산학(석사)
1994년 숭실대학교 전산학(박사)
1981년 숭실전산원 전임교수
1985년 숭실전산원 전산실장
1990년 일본 모리 정보대학 교환교수
1995년 ~현재 숭실전산원 부원장
관심분야 : S/W 리엔지니어링, 객체지향시스템, 컴퓨터 알고리즘

전 문 석

1980년 숭실대학교 전산학(학사)
1986년 Univ. of Maryland 전산학(석사)
1988년 Univ. of Maryland 전산학(박사)
1989년 Morgan State Univ. 전산수학과 조교수
1991년 New Maxico State Univ. 부설 Physical Science Lab. 책임연구원
1991년~현재 숭실대학교 컴퓨터학부 부교수
관심분야 : 병렬 알고리즘, 병렬컴퓨터 구조, 대규모 집적회로, 병렬처리 이론, 폴트톨로런스

류 성 열

1977년 숭실대학교 전산학(학사)
1980년 연세대학교 산업대학원(석사)
1991년 아주대학교 전산학(박사수료)
1981년 전국 금융협회 고문
1986년 숭실대학교 전자계산연구 소장
1994년 한국정보과학회 총무이사
1981년~현재 숭실대학교 컴퓨터학부 교수
관심분야 : S/W 재사용, S/W 리엔지니어링, 객체지향시스템