시각화된 환경에서 다차원 관점을 지원하는 객체기반 패싯 시소러스 관리 시스템 모델의 정형화 및 구현

(A Data Model for an Object-based Faceted Thesaurus System Supporting Multiple Dimensions of View in a Visualized Environment)

김 원 중 † 양 재 동 ††
(Won-Jung Kim) (Jae-Dong Yang)

요 약 본 논문에서는 패싯 분류에 기반하여 다차원 사용자 관점의 도메인 시소러스를 체계적으로 구축하고, 각 관점에 따라 시각화된 환경에서 시소러스를 브라우징, 항해 그리고 검색할 수 있는 다차원 관점 객체기반 패싯 시소러스 시스템을 제안한다. 기존의 패싯 시소러스 시스템들과 달리, 본 시스템은 개념들간의 다각적인 상관 관계를 객체 지향 패러다임에 따라 자동으로 설정 구축할 수 있고, 브라우징과 항해를 통해 다차원 시소러스의 복잡한 개념 구조를 적절히 시각화할 수 있기 때문에, 시소러스의 유지 보수 관리가 용이하다. 다차원 브라우징 및 항해는 계층화된 패싯 용어들간의 조합으로 이루어진 패싯화된 시소러스를 필요시 동적으로 생성함으로써 이루어진다. 패싯화된 시소러스는 일종의 다차원 가상 시소러스 계층들로 볼 수 있다. 제안하는 방식에 의한 시소러스 자동 구축은 패싯들의 조합으로 새로운 차원의 시소러스를 용이하게 추가할 수 있기 때문에, 융통성 있는 시소러스 확장이 가능하고 대량의 시소러스 인스턴스들을 관점에 따라 적절한 해석을 부여하는 방식으로 구조화하기 때문에 개별적 사용자 관점에 부합되는 인스턴스들을 참조 질의에 의해 효율적으로 검색할 수 있다. 본 논문에서는 먼저 제안 시스템을 체계화하기 위한 모델을 정형화하고 이를 바탕으로 모델의 실용성을 입증하기 위해 그 프로토타입을 구현하였다.

키워드 : 패싯 시소러스, 패싯 질의, 시소러스 시스템, 그래픽 편집기

Abstract In this paper we propose a formal data model of an object-based thesaurus system supporting multi-dimensional facets. According to facets reflecting on respective user perspectives, it supports systematic construction, browsing, navigating and referencing of thesauri. Unlike other faceted thesaurus systems, it systematically manages its complexity by appropriately abstracting sophisticated conceptual structure through visualized browsing and navigation as well as construction. The browsing and navigation is performed by dynamically generating multi-dimensional virtual thesaurus hierarchies called "faceted thesaurus hierarchies." The hierarchies are automatically constructed by combining facets, each representing a dimension of view. Such automatic construction may make it possible the flexible extension of thesauri for they can be easily upgraded by pure insertion or deletion of facets. With a well defined set of self-referential queries, the thesauri can also be effectively referenced from multiple view points since they are structured by appropriately interpreting the semantics of instances based on facets. In this paper, we first formalize the underlying model and then implement its prototype to demonstrate its feasibility.

Key words : Faceted Thesaurus, Facet Query, Thesaurus System, Graphic Editor

· 본 논문은 전북대학교 영상정보신기술연구소의 지원으로 수행되었음
† 학생회원 : 전북대학교 컴퓨터통계정보
kimwj@chonbuk.ac.kr
†† 정 회 원 : 전북대학교 전자정보공학부 교수
jdyang@chonbuk.ac.kr
논문접수 : 2006년 10월 10일
심사완료 : 2007년 6월 9일

1. 서 론

시소러스는 사용자 질의를 검색에 적합한 형태로 변형하거나 확장함으로써 검색 결과의 재현률(recall)을 향상시키고자 할 경우 주로 사용된다[1]. 즉 질의내의 검

색어들만으로 얻어진 결과가 너무 포괄적이거나 제한적일 때, 시소러스는 검색어들의 의미를 개념적으로 제한하거나 확대함으로써 검색 효율을 높일 수 있다.

지난 십여년간 시소러스를 효율적으로 구축하고 이용할 수 있는 시소러스 시스템에 대한 많은 연구들이 수행되었다. 상용화된 시소러스 구축 시스템의 문제점은 일반적으로 전문가들이 일관된 관점에서 목표로 하는 도메인 시소러스를 구축하기 어렵기 때문에, 여러 관점이 한 계층내에 필연적으로 혼재하게 된다는 점이다.

이러한 문제점을 해결하기 위해, 관점을 나타내는 속성인 패싯을 이용한 시소러스 구축기법이 제안되었다. 이 기법은 시소러스 구축시 개념들이 가지고 있는 공통된 속성에 따라 패싯 분류를 생성하고, 이 분류에 따라 개념들이 다각적 관점에서 개별적으로 구조화될 수 있도록 지원한다. 패싯 분류는 주제를 분류 체계에 삽입하는 것이 아니라 패싯내의 개념으로 조합하기 때문에 새로운 주제를 쉽게 수용할 수 있다는 장점이 있다[2]. 또한 더 많은 복합 주제를 표현하기 위해 필요한 만큼 조합할 수 있기 때문에, 패싯 분류를 정의하여 사용한다[3].

패싯 기법을 시소러스 구축에 적용하는 많은 상용 시스템들[4-11]이 개발되었다. 예를 들어, [4-6]은 웹과 온라인 상의 검색에 패싯을 적용하였고, [7]은 패싯 분류 구축 및 패싯 분류의 필터링에, [8-11]은 카테고리와 개념들간의 계층 구조에 대한 브라우징 및 항해에 각각 패싯 적용을 시도하였다. 그러나 이 시스템들은 패싯들의 유효한 조합 검색을 지원하지 않기 때문에 각 사용자 관점에 맞는 검색 결과를 제공하지 못하거나 패싯 조합에 의해 새롭게 생성된 개념들의 유기적인 상관 관계를 적절히 추상화하여 시각화하지 못하는 공통적인 단점을 보이고 있다.

본 논문에서는 패싯 분류에 기반하여 다차원 사용자 관점의 도메인 시소러스를 체계적으로 구축하고 개별적 관점에 따라 시각화된 환경에서 시소러스를 브라우징, 항해 그리고 검색할 수 있는 다차원 관점 객체기반 패싯 시소러스 시스템 모델을 제안하고 그 프로토타입을 구현한다.

제안 시스템은 패싯으로 표현되는 개념들간의 다각적인 상관 관계를 자동으로 설정 구축할 수 있고, 브라우징과 항해를 통해 다차원 시소러스의 복잡한 개념 구조를 적절히 추상화할 수 있기 때문에, 시소러스의 유지보수 관리가 용이하다. 다차원 브라우징 및 항해는 계층화된 패싯 용어들의 조합 색인으로 이루어진 패싯화된 시소러스를 필요시 동적으로 생성함으로써 이루어진다. 제안하는 방식에 의한 시소러스 자동 구축은 패싯들의 조합으로 새로운 차원의 시소러스를 용이하게 추가할 수 있기 때문에, 융통성 있는 시소러스 확장이 가능하고 대량의 시소러스 인스턴스들을 관점에 따라 적절한 해석을 부여하는 방식으로 구조화하기 때문에 개별적 사용자 관점에 부합되는 인스턴스들을 참조 질의에 의해 효율적으로 검색할 수 있다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 기존 시소러스 시스템에 관한 연구들을 살펴보고, 3장에서는 본 논문에서 제안한 다양한 관점을 지원하는 객체기반 패싯 시소러스 시스템 모델을 기술한다. 4장에서는 제안 모델에 따라 설계된 시스템의 구현 내용을 기술하고, 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

시소러스를 효율적으로 구축하고 이용할 수 있는 상용화된 시소러스 구축 시스템으로는 Term Tree[12], BrainEKP[13], Thesaurus Master[14], MultiTest[15], Taxonomy Editor[16], TermChoir[17] 등이 있다. 이들은 공통적으로 시소러스를 관리함에 있어 개별적인 관점을 표현하는 패싯 개념을 지원하지 못할 뿐 아니라 적절히 시소러스를 추상화하여 시각화하지 못하는 공통적인 단점을 가지고 있다. 이 단점은 특히 여러 관점이 혼재함으로 인해 시소러스 구조가 급격히 복잡해 질 경우에 시소러스 구조를 개념적으로 이해하는데 큰 장애요소가 될 수 있다.

이를 해결하기 위해 패싯 기법을 시소러스 구축에 적용하는 많은 상용 시스템들이 개발되었다. 예를 들어, FacetMap[7]은 패싯 분류의 필터링에 의한 개념 검색은 패싯 구조화에 의한 개념 기반 색인에 의해 이루어지기 때문에 정확하게 일치하는 검색 결과만을 제공한다. HiBrowse[5]는 사용자에게 동시에 다른 패싯에 속하는 개념들에 대한 정보도 제공하는데, 예를 들어, 사용자가 패싯 'Disease'의 값 'phisical disease'을 선택하면, 한 단계 아래에 나타나는 개념들과 각 개념들을 포함한 문헌의 수가 함께 보여지며, 사용자가 'Therapy'과 'Groups'와 같은 다른 패싯을 선택하면, 세 가지 패싯 형태를 모두 포함하는 문헌의 수가 제시된다.

이 시스템들의 문제점은 사용자가 질의를 직접 명시하는 것이 아니라 패싯 계층을 통해서만 항해하도록 했다는 점이다. 이는 질의 명세가 패싯에 속하는 데이타를 보여주는 것과 밀접하게 연결되어 있지 않음을 나타내고 있다.

에이치슨(J. Aitchison)에 의해 개발된 전기공학분야의 Thesaurofacet[11], Facet Project[8], 그리고 Art and Architecture Thesaurus[9]는 모두 분류 부분은 종속 관계와 같은 완전한 계층을 묘사하고 시소러스 부분은 상위 개념, 하위 개념, 관련 개념과 같은 계층 부분들 간의 개념 관계를 보여주고 있다. 즉 카테고리와 개념들 간의 계층 구조에 대한 브라우징 및 항해를 제

공하고, 검색어의 조합을 통해 일치하는 개념 검색을 지원하며 개념의 계층 구조로 항해 또한 가능하다. 그러나 이 연구들 또한, 시소러스 다차원 패싯 계층 관계를 시각화하지 못할 뿐 아니라 다차원 패싯 조합에 따른 검색은 지원하지 못하는 한계점을 가지고 있다. 예를 들어, 두 패싯 form과 design에 대해, "(chairs by form) and(chairs by design)"과 같은 패싯 조합을 통한 검색은 제공하지 못한다.

패싯 기반 구축 기법으로, [18]은 패싯 용어들의 유효한 조합을 통한 검색을 지원하는 연역적 추론 메커니즘을 제공하고, 용어 조합을 통한 항해 트리의 동적 생성도 제공한다. 그러나 이 방식은 시소러스가 아닌 웹 카탈로그에 패싯 분류 방식을 적용하였다는 점과 연역 논리에 기반을 둔 이론적인 연구라는 점에서 본 논문이 제안하는 시스템과는 다르다.

최근의 연구사례로 원자력분야 용어를 대상으로 특정 용어의 관련어(Related Term)에 대해 개념 패싯을 도입하여, 시소러스를 온톨로지화 하기 위한 작업[19]과 한국과학기술정보연구원의 상·하위어 간 조어적(형태적) 유사성에 의한 기준에 패싯을 적용한 과학기술분야 시소러스 구축 사업[20]을 들 수 있다. 그러나, [19]는 패싯 기법을 관련어에만 도입함으로써, 활용에 있어 큰 제약을 갖는다는 단점이 있고, [20]은 패싯 시소러스가 전적으로 수작업에 의해 구축되었기 때문에, 반자동으로 패싯화된 시소러스를 구축하는 본 제안 방식과는 근본적으로 차이가 있다.

다음 절에서는 지금까지 언급한 문제점들을 해결하기 위한 패싯 시소러스 시스템의 제안 모델을 구체적으로 기술한다.

3. 다차원 관점을 지원하는 객체기반 패싯 시소러스 시스템 모델

이 절에서는 먼저, 제안 시스템이 기초로 하고 있는 객체기반 시소러스 시스템을 간단히 소개하고, 필요한 기본 개념들을 설명한 뒤, 이 시스템의 문제점을 알아보도록 한다.

3.1 객체기반 시소러스 시스템[21]

기존의 시소러스 시스템과는 달리 객체기반 시소러스에서 한 노드는 고유 식별자를 가진 객체로서 용어명 이외 다양한 관련 부가 정보를 가진다. 일반적으로, 객체기반 시소러스는 계층 관계와 둘 또는 그 이상의 객체들간의 관계들로 표현된다. 계층 관계는 일반화, 인스턴스화 관계가 있으며 객체들간의 종속 또는 포함 관계에 의해 이루어진다. 예를 들어, 그림 1은 계층 관계를 적용하여 구축한 'Software' 시소러스이다. 여기서 'Software'와 'Script Language,' 'Script Language'와 'Perl'은 일반화 관계이며, 인스턴스화 관계는 'Perl'과 'Fink,' 'C#'과 'VitePad' 등이다. 또한 'VitePad'는 'C#,' 'Windows' 그리고 'TextProcessing' 세 객체들의 인스턴스 객체이다. 객체기반 시소러스 시스템의 차별화되는 다른 특징들은 [21]을 참조하기 바란다.

패싯을 지원하지 않는 객체기반 시소러스 시스템에 다음의 세 패싯을 최상위 개념 객체인 'Software' 에 적용해서 시소러스를 구축할 경우를 고려해보자.

- PL(Programming Language)
- OS(Operating System)
- Function

패싯은 구체적으로 표현되는 객체들의 공통된 속성으

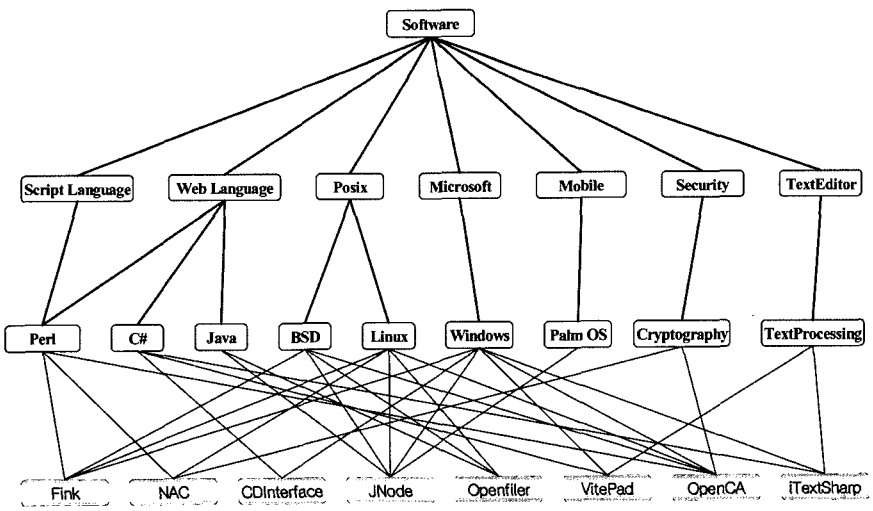

그림 1 'Software' 시소러스

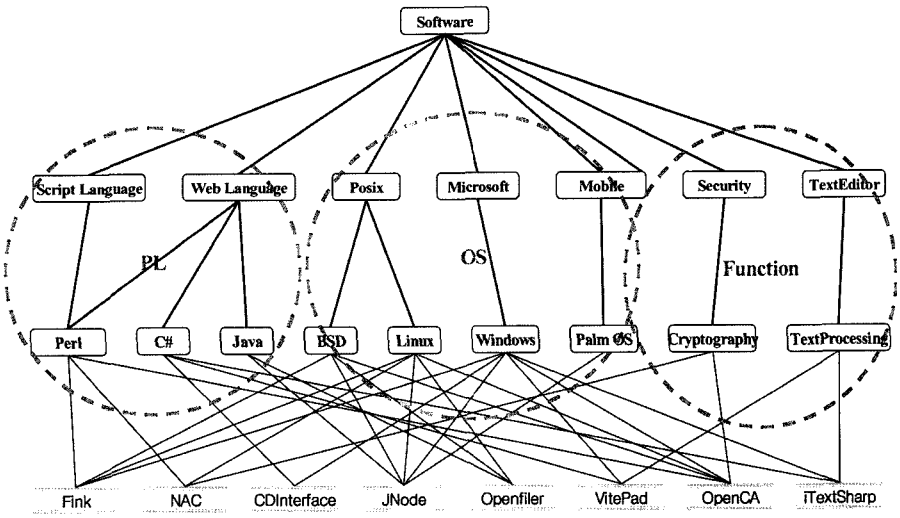

그림 2 패싯으로 구분한 'Software' 시소러스

로 도메인에 대한 특별한 관점을 나타낸다. 그림 2는 'Software' 시소러스를 패싯에 따라 구분한 화면이다.

다른 기존의 시소러스 시스템과 마찬가지로 객체기반 시소러스도 한 시소러스 계층내에서 여러 관점이 혼합되면서 시소러스의 가독성이 현저히 떨어진다는 것을 알 수 있다. 다음 절에서는 객체기반 시소러스에 패싯 분류를 체계적으로 적용함으로써 이를 해결할 수 있는 본 제안 시소러스 모델을 기술한다.

3.2 객체기반 패싯 시소러스 관리 시스템 모델

제안 모델은 인스턴스 객체, 개념 객체, 패싯, 패싯 개념 용어들로 구성된다. 이들은 다음과 같이 정의된다.

정의 1. 인스턴스 객체 $o \in O$는 자신의 하위 개념을 갖지 못하는 시소러스 객체이다. 개념 객체 $c \in C$는 직간접적으로 인스턴스 객체와 instanceOf 관계를 맺고 있는 시소러스 객체이다. 패싯 $f \in F$는 인스턴스 객체들의 특정 관점을 표현하는 속성이며, 패싯 개념 용어, 또는 패싯 용어는 패싯 $f \in F$에 실례화될 수 있는 개념 객체이다. 인스턴스 객체들의 집합은 I로 표기한다.

예 1. 'Windows'는 패싯 os$\in F$의 한 실례이므로 패싯 개념 용어이다.

각 패싯에 속하는 패싯 개념 용어들은 고유의 관계성을 통해 독립적인 시소러스 구조를 가진다. 이를 위한 정의는 다음과 같다.

정의 2. 패싯 개념 용어간에 존재하는 관계성들의 집합을 $R_f = \{\text{subConceptOf}_f, \text{synonymOf}_f, \text{associationOf}_f\}$라고 할 때, 패싯 f는 다음과 같이 R_f에 의해 구조화되는 패싯 시소러스로 정의할 수 있다.

$f = \{t \mid t = < o_1, o_2, r >$, 여기서 $r \in R_f$이고 $o_1, o_2 \in O\}$

여기서 패싯 시소러스 f내에 포함된 패싯 개념 용어들의 집합은 Of로 표기한다. 패싯 개념 용어도 개념 객체이므로 고유의 인스턴스들을 가질 수 있다. 그러나 본 모델이 복잡해지는 것을 피하기 위해 여기서는 패싯 시소러스내의 패싯 개념 용어만을 고려하도록 한다. 따라서 C_f는 C_f로 표기가 가능하다.

예 2. 패싯 시소러스 $pl \in F$는 R_{pl}에 의해 구조화된다. 이 패싯 시소러스에 속하는 튜플들은 $< \text{Perl, Script Language, subConceptOf}_{pl} >, < \text{Java, 자바, synonymOf}_{pl} > \in pl$ 등이 있다.

R_f에 의해 구조화된 패싯 시소러스들은 각 패싯 시소러스내의 패싯 개념 용어들간의 관계로만 이루어진다. 즉 각 관점에 따라 생성되는 패싯 개념 용어들은 각 패싯내에서 독립적으로 구조화된다. 이를 위한 정의는 다음과 같다.

정의 3. 두 패싯 시소러스 f_1과 f_2에 대해서 $r_1 \in R_{f_1}$이고 $r_2 \in R_{f_2}$일 때, $r_1 = r_2$인 경우는 없다.

예 3. $t_1 = < c_1, c', \text{subConceptOf}_{f_1} > \in f_1$이고 $t_2 = < c', c_2, \text{subConceptOf}_{f_2} > \in f_2$일 때, subConceptOf의 전이성에 의해 $t = < c_1, c_2, \text{subConceptOf}_{f_1} > \in f_1$ 또는 $t' = < c_1, c_2, \text{subConceptOf}_{f_2} > \in f_2$는 정의 3에 의해 성립하지 않는다. 즉 $r \in R_f$은 한 패싯 시소러스 f의 개별적 관점에 따라 $c, c' \in C_f$간에만 성립하는 관계성이다.

정의 4. $\{f_1, f_2, ..., f_m\} \subseteq F$일 때, 패싯화된 개념 용

어 $c_{12...m} \in C$ 는 $\forall c_i \in C_{f_i}, i=1...m$ 와 facetedConceptOf 관계성을 맺는 개념 객체로 정의하며 다음과 같이 표기한다.

$$c_{12...m} = c_1 // c_2 // ... // c_m // c_{top}$$

여기서 c_{top} 은 facetedConceptOf 관계성이 적용되는 개념 용어로 일관성을 유지하기 위해 최상위 패싯화된 개념 용어로 정의한다. 더하여, 패싯 시소러스들의 집합 $\{f_1, f_2, ..., f_m\} \subseteq F$ 일 때, $c_1 \in C_{f_1}, c_2 \in C_{f_2}, ..., c_m \in C_{f_m}$ 인 모든 $\{c_1, c_2, ..., c_m\}$ 들을 원소로 하는 멱집합은 C_F 로 표기한다. 즉 $\{c_1, c_2, ..., c_m\} \in C_F$ 이다.

패싯화된 개념 용어는 다음의 함수 GenFacetedConcept 을 통해 생성되는데, 이 함수에서 $c_{12...m}$ 내 $c_1, c_2, ..., c_m$ 의 순서는 패싯 조합 후 복합용어 사전을 참조함으로써 정해진다.

```
GenFacetedConcept({c1,c2,....,cm},ctop){
   CandidateList ={};
  CandidateList = CandidateList ∪CombinationOfFacetConcept
  ({c1,c2,....,cm},ctop);
   For each c∈CandidateList
    if(ExistInDictionary(c)) //복합용어사전 참조
          return (c);
   end
   return (c12...m);// 사전에 없을 경우 원래순서대로 생성
}
```

그림 3 GenFacetedConcept 함수

예 4. c_{top} =Software라고 하자. 만약 c_1 =C#$\in C_{pl}$, c_2 =Windows$\in C_{os}$ 일 경우, GenFacetedConcept 함수는 c_{12} =C#//Windows//Software$\in C$ 를 반환하고, c_{12} 는 각 패싯 개념 용어와 facetedConceptOf 관계성을 맺는다. 그러나 만약 c_{21} =Windows//C#//Software가 복합용어 사전에 있다면 GenFacetedConcept는 c_{21} 을 반환하게 된다.

편의상, 한 패싯 용어 $c_i \in C_{f_i}, i=1,...,m$ 가 패싯화된 개념용어 $c_{12...m}$ 의 i 번째 조합에 사용된다면 $c_i = comp_i(c_{12...m})$ 으로 표기하기로 한다. 위치가 중요하지 않을 경우에 i 는 생략될 수 있다.

패싯화된 개념 용어들로 구성되는 패싯화된 시소러스의 계층 관계는 다음과 같이 정의된다.

정의 5. $\{f_1, f_2, ..., f_m\} \subseteq F$ 이고, $i_1, i_2, ..., i_m \in \{1,2,...,m\}$ 일 때, c, c' 이 각각 c_{top} 아래의 패싯화된 개념 용어라고 하자. 만약 $comp_{i_1}(c) \geq_{f_{i1}} comp_{i_1}(c'), ..., comp_{i_j}(c) \geq_{f_{im}} comp_{i_m}(c')$ 라면 $c \geq c'$ 이다. 여기서 $\geq_{f_i}$ 은 패싯 시소러스 f_i 내의 패싯 개념 용어간 직간접 상하위 관계 또는 같은 레벨의 패싯 개념 용어를 의미한다. $c \geq c'$ 일 때, c 는 c' 의 상위 패싯화된 개념 용어, c' 는 c 의 하위 패싯화된 개념 용어라고 부르고 각각 c_{super}, c_{sub} 로 표기한다.

$c_{12...m} = c_1 // c_2 // ... // c_m // c_{top}$ 이 모두 의미있는 패싯화된 개념 용어들이 될 수 없음을 유의해 보자. 각 패싯 개념 용어가 facetedConceptOf 관계성을 통해 조합되어 하나의 패싯화된 개념 용어를 생성할 때, 그 개념 용어를 구성하는 패싯 개념 용어들간에 하나 이상의 양립할 수 없는 쌍이 존재하면 그 조합은 유효한 패싯화

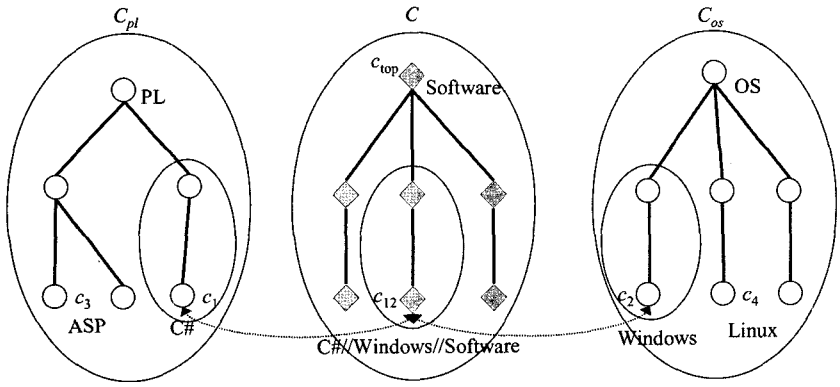

그림 4 패싯화된 개념 용어 생성

된 개념 용어를 생성할 수 없게 된다. 예를 들어, c_{top} =Software 일 때, c =ASP//Linux//Software 또는 c =C#//Linux//Software는 패싯화된 개념 용어가 될 수 없다. 즉 패싯 개념 용어 'ASP', 'C#'은 'Linux'와 비양립쌍(Incompatible Pair)을 이루기 때문에 이들간의 조합은 유효하지 않다. 개념 객체의 상속 성질에 따라 비양립 쌍에 속하는 각 패싯 개념 용어의 하위 패싯 개념 용어들간의 쌍 또한 비양립 쌍이 된다. 예를 들어, c_{top} =Software, C# $\in C_{pl}$, Posix $\in C_{os}$ 일 때, 패싯 개념 용어 쌍인 <C#, Posix>가 비양립 쌍을 이루기 때문에 'C#'은 'Posix'의 하위 패싯 개념 용어인 BSD, Linux $\in C_{os}$ 들과도 비양립 쌍이 된다. 따라서 이러한 용어들은 적절한 판단 알고리즘에 의해 모두 제거될 필요가 있다. 다음은 이 알고리즘의 기반이 되는 명제를 증명없이 기술한다.

명제 1. $c_i \in C_{f_i}, c_j \in C_{f_j}$ 일 때, $< c_i, c_j >$가 비양립쌍이라고 하자. 그러면 $c_i \geq_{f_i} c_i', c_j \geq_{f_j} c_j'$ 인 $\forall c_i' \in C_{f_i}, \forall c_j' \in C_{f_j}$ 에 대해, $< c_i', c_j' >$는 비양립 쌍이며 다음을 만족하는 패싯화된 개념 용어 $c'_{12...m}$ 는 무효한 패싯화된 개념 용어이다.

$$\{c_i', c_j'\} \subseteq \{comp_l(c'_{12...m}), l = 1,..., m\}.$$

다음의 GenFacetedConceptHierarchy 는 자동으로 패싯화된 개념 용어들간의 계층화된 시소러스를 생성하는 프로시저이다.

```
GenFacetedConceptHierarchy({c1,c2,...,cm},c,ctop)
{
  For each {c1',c2',...,cm'} = CombinationOfSubFacetConcepts({c1,c2,...,cm})
          // 각 ci,1,...,m의 하위 패싯 개념 용어들간의서로다른조합들에 대해
     c' = GenFacetedConcept({c1',c2',...,cm'},ctop);
     if(HasIncompatiblePair({c1',c2',...,cm'}){
        AddInvalidFacetedConceptList(c');
        GenInvalidFacetedConceptHierarchy({c1',c2',...,cm'},c',ctop);
     }else if(c ≥ c'){     // c,c' ∈ C,[정의5]로 부터판별
        set(c' ∈ subConceptOf(c,ctop));
        set(c' ∈ facetedConceptOf(ci,ctop),i=1,...,m);
        GenFacetedConceptHierarchy({c1',c2',...,cm'},c',ctop);
     }
  end
}
```

그림 5 GenFacetedConceptHierarchy 함수

계속해서, 그림 6은 유효하지 않은 패싯화된 개념 용어를 최상위 개념으로 하는 계층을 생성하기 위한 프로시저 GenInvalidFacetedConceptHierarchy를 보이고 있다.

```
GenInvalidFacetedConceptHierarchy({c1,c2,...,cm},c,ctop)
{
   For each {c1',c2',...,cm'} = CombinationOfSubFacetConcepts({c1,c2,...,cm})
      c' = GenFacetedConcept({c1',c2',...,cm'},ctop);
      AddInvalidFacetedConceptList(c');
      if(c ≥ c') set(c' ∈ subConceptOf(c,ctop));
      GenInvalidFacetedConceptHiearchy({c1',c2',...,cm'},c',ctop);
   end
}
```

그림 6 유효하지 않은 패싯화된 개념 용어 계층 생성

예 5. c_{top} =Software, C# $\in C_{pl}$, Posix $\in C_{os}$ 일 때, HasIncompatiblePair(C#, Posix)에 의해 비양립 쌍임이 확인되면 생성된 "C#//Posix//Software"는 유효하지 않은 패싯화된 개념 용어 리스트에 추가된다. 다음으로 GenInvalidFacetedConceptHierarchy는 CombinationOfSubFacetConcepts을 사용해 C#과 'Posix'의 하위 개념 용어인 BSD, Linux $\in C_{os}$ 들간 서로 다른 가능한 조합을 구한 뒤 GenFacetedConcept 함수를 통해 각각 패싯화된 개념 용어, "C#//BSD//Software"와 "C#//Linux//Software" 등을 생성한다. 이들 또한 유효하지 않은 패싯화된 개념 용어 리스트에 추가된다. 마지막으로, 'BSD'와 'Linux'는 재귀적으로 GenInvalidFacetedConceptHierarchy의 입력 값이 된다. 그 외, 유효한 패싯화된 개념 용어들의 생성 과정에 대해서는 예 6에서 자세히 설명하도록 한다.

본 논문에서는 시소러스 구축시만을 고려하여 상속에 의한 비양립 쌍만을 검사하였으며 데이타 임포트시에 비양립 쌍을 포함하는 패싯화된 개념 용어의 판별은 고려하지 않았다.

마지막으로, GenFacetedConceptHierarchy 프로시저에서 패싯화된 개념 용어 c'와 조합에 참여한 각 패싯 용어 $c_i, i = 1,..., m$ 간 facetedConceptOf 링크를 만드는 다음 실행문을 고려해 보자.

$$\text{set}(c' \in \text{facetedConceptOf}(c_i, c_{top}), i = 1,..., \text{m}); \quad (1)$$

일반적으로 패싯 용어들간의 조합은 방대한 양의 패싯화된 개념 용어들을 만들어 내기 때문에 해당 링크들을 모두 생성해 낸다면, 링크 정보가 중복될 뿐 아니라, 삭제 갱신 등으로 인한 불일치등 시소러스 관리에 많은

어려움을 야기하게 된다. 예를 들어, 5개의 패싯 시소러스가 각각 10개의 패싯 용어들로 구성된다면, 패싯화된 개념 용어들의 수는 최대 10만개가 될 수 있다. 이 문제는 다음과 같이 위 프로시저에서 (1)을 다음 (2)와 같이 수정함으로써 간단히 해결할 수 있다.

$$\text{if } (\text{comp}(c) \neq c_i', i = 1,...,m) \text{ then}$$

$$\text{set}(c' \in \text{facetedConceptOf}(c_i, c_{top}), i = 1,...,\text{m}); \quad (2)$$

즉 c'의 조합에 사용되는 어떤 패싯 용어 c_i'가 $c \geq c'$인 c의 조합에도 사용되었다면 c'는 자신의 facetedConceptOf 링크를 만들지 않고 c의 해당 링크를 상속받음으로써 불필요한 링크의 남용을 막을 수 있다.

이제, 여러 패싯 시소러스들에 속하는 패싯 개념 용어들의 조합으로 생성되는 패싯화된 시소러스는 아래와 같이 정의된다.

정의 6. $R = \{\text{subConceptOf}, \text{synonymOf}, \text{instanceOf}\}$, $\{f_1, f_2, ..., f_m\} \subseteq F$라고 할 때, 패싯화된 시소러스 T_F^{top}는 다음과 같이 정의된다.

$T_F^{\text{top}} = \{t \mid t = < c_1, c_2, r > \text{ 여기서 } r \in R \text{ 이고 } c_1, c_2 \in C_F^{valid} \cup I\}$

C_F^{valid}는 유효한 패싯화된 개념들의 집합이고 I는 인스턴스들의 집합이다. 임의의 개별적 $f_i \in F$가 적용되어 구성되는 패싯화된 시소러스를 명시하고자 할 때에는 $T_{f_i}^{\text{top}}$로 표기한다.

다음 예의 그림 7은 패싯 시소러스 'pl'과 패싯 시소러스 'os'로부터 패싯화된 시소러스 $T_{\{\text{pl,os}\}}^{\text{Software}}$를 구축하는 과정을 보이고 있다.

예 6. Script Language $\in C_{pl}$과 Microsoft, Posix $\in C_{os}$는 각각 생성된 패싯화된 개념 용어인 "Script Language//Microsoft//Software" c_1, "Script Language //Posix//Software" c_2와 facetedConceptOf 관계를 통해 링크된다.

그림 8은 'Script Language'의 하위 패싯 개념 용어인 'Perl'과 'Microsoft'가 "Perl//Microsoft//Software" c_3를 생성하는 과정을 보여준다. 이때, 패싯화된 시소러스의 개념 용어들간의 계층 구조는 정의 5에 의해 자동 유지된다. 즉 $c_1 \geq c_3$이다. c_3는 'Perl'과 $\text{facetedConceptOf}_{pl}$ 관계성 링크는 유지하지만 'Microsoft'와의 해당 관계성, 즉 $\text{facetedConceptOf}_{os}$는 불필요한 링크의 남용을 피하기 위해 c_1의 관계성을 상속받는다.

계속해서, 그림 9는 'Perl'과 'Windows'가 "Perl// Windows//Software" c_4를 생성하는 화면이다. 여기서 "Microsoft $\geq$ Windows"이므로 'Windows'와 c_4간 링크는 c_3로부터 상속받는 링크를 대체해서 새로이 생성되고 나머지 링크는 c_3로부터 상속받는다.

동일한 방식으로 그림 10의 패싯화된 시소러스가 구축될 수 있다.

다음으로 그림 11은 'C#'이 'Microsoft'와 "C#// Microsoft//Software"를 생성하고 'Windows'와 "C#//

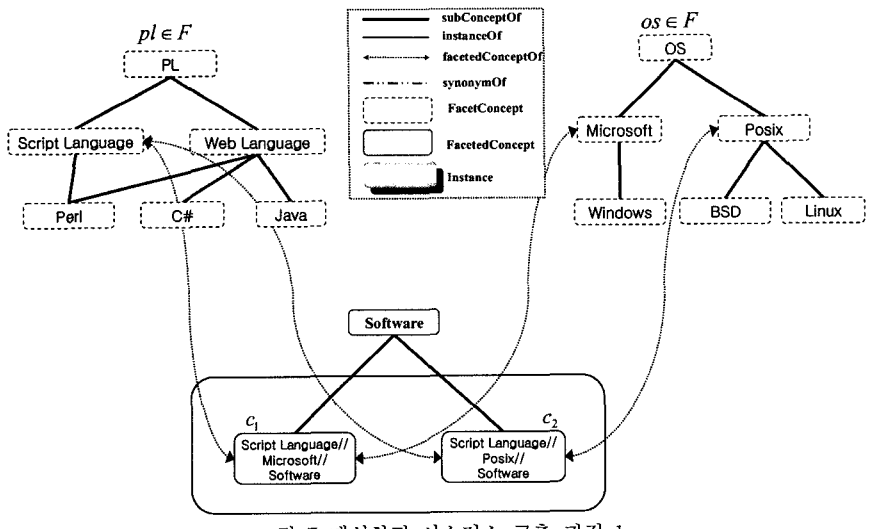

그림 7 패싯화된 시소러스 구축 과정 1

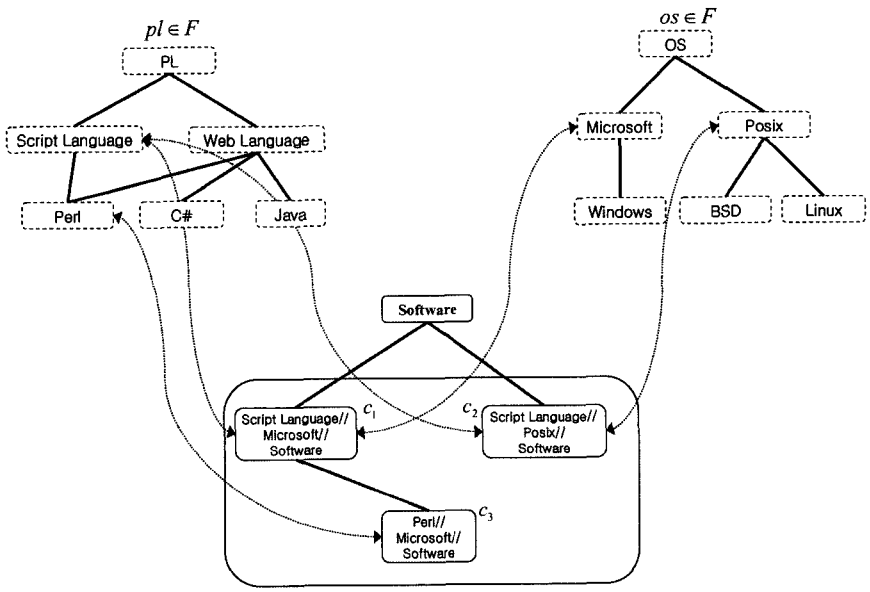

그림 8 패싯화된 시소러스 구축 과정 2

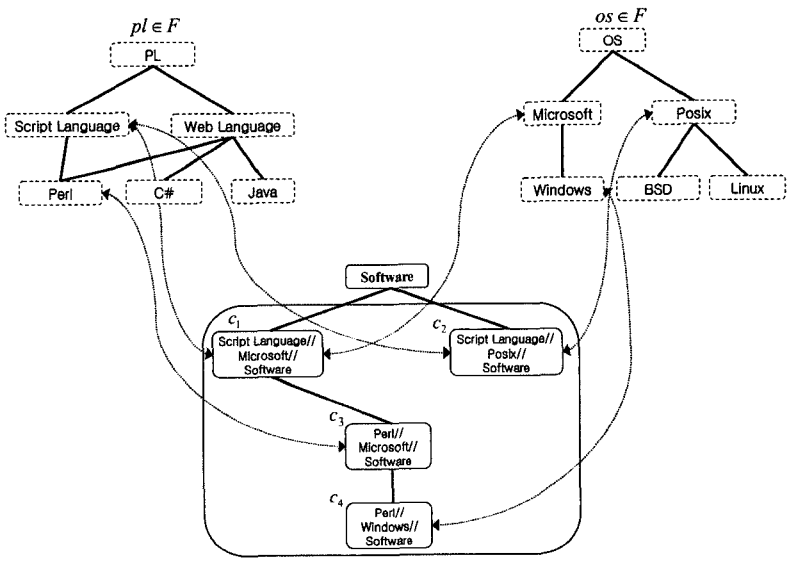

그림 9 패싯화된 시소러스 구축 과정 3

Windows//Software"를 생성하는 과정을 보이고 있다.

그런데 이 경우는 'C#'이 'Posix'와는 "C#//Posix// Software"를 생성할 수 없는데, 그 이유는 <C#, Posix>가 비양립 쌍이기 때문이다. 또한 'Posix'의 하위 패싯 개념 용어들도 상위 패싯 개념 용어가 갖는 비양립 속성을 상속받기 때문에 'C#'과의 조합에 의한 유효한 패싯화된 개념 용어를 생성하지 못한다. 그림 12는 하위 패싯 개념 용어들에게도 적용되는 비양립 쌍에 의해 유효하지 못한 패싯화된 개념 용어들로 구성된 계층을 보여준다.

지금까지 패싯 시소러스들간의 조합으로 자동으로 패싯화된 시소러스를 생성하는 과정을 보였다. 이를 위한 처리 절차는 그림 13과 같다.

다음으로, 예 7의 그림 14는 패싯 시소러스의 한 패싯 개념 용어가 삭제될 경우 패싯화된 시소러스가 재구성되는 과정을 패싯 개념 용어 'Perl'을 예로 들어 보여주고 있다.

예 7. 먼저, 'Perl'과 facetedConceptOf 관계를 갖는

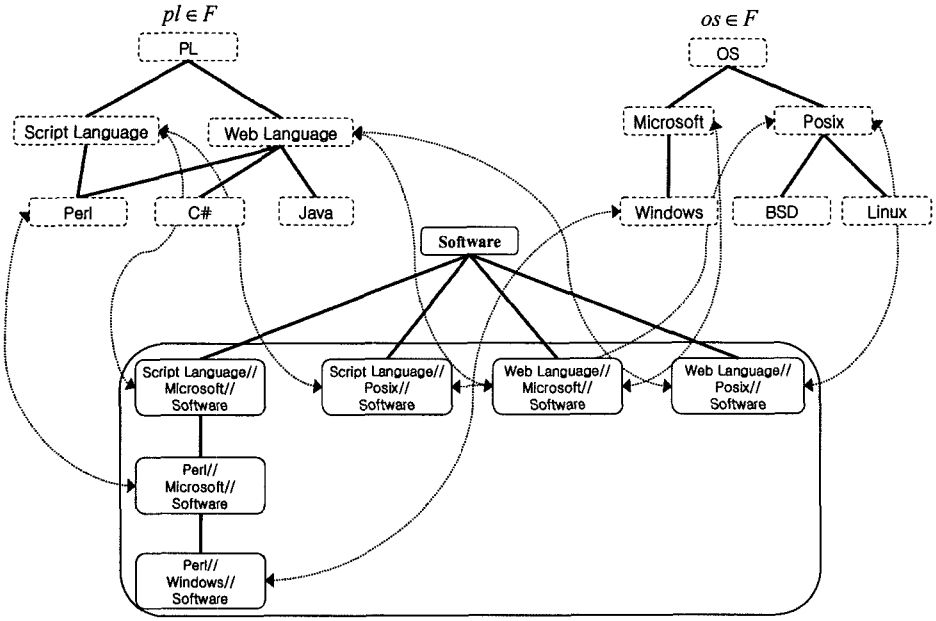

그림 10 패싯화된 시소러스 구축 과정 4

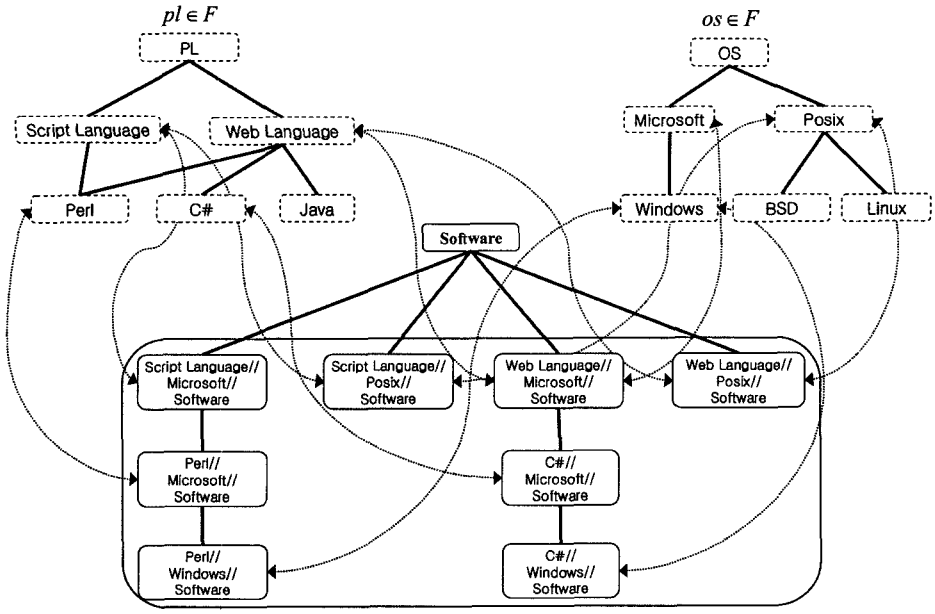

그림 11 패싯화된 시소러스 구축 과정 5

각 패싯화된 개념 용어, 예를 들어, "Perl//Posix//Software" c_1을 검색한 후, 검색된 c_1의 하위로 순회하며 $c_1 \geq c$ 인 패싯화된 개념 용어 c, 즉 c_2, c_3를 검색한다. 다음으로 c_2의 각 facetedConceptOf 관계를 검색하여 삭제하고 c_2를 삭제한다. 결과적으로 $c_2 -> c_3 -> c_1$ 순으로 모든 관계들이 삭제되고 패싯화된 개념 용어들 c_2, c_3, c_1이 삭제된다. 마지막으로 'Perl'이 삭제되어 그림 15와 같이 재구성된다.

본 논문에서 제시한 객체기반 패싯 시소러스의 또 다른 장점으로는 패싯 개념 용어들의 관계성을 이용해 시소러스가 확장될 수도 있다는 점이다. 즉 한 패싯 시소

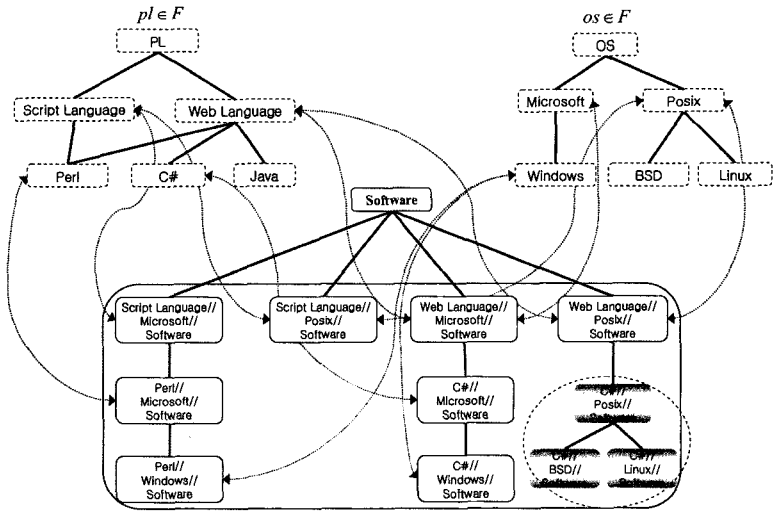

그림 12 유효하지 못한 패싯화된 개념 용어 생성

```
GenFacetedThesaurus(C_F, c_top)
{
    For each {c_1, c_2, ..., c_m} ∈ C_F
        c = GenFacetedConcept({c_1, c_2, ..., c_m}, c_top);
        set(c ∈ facetedConceptOf(c_i, c_top), i = 1, ..., m);
        GenFacetedConceptHierarchy({c_1, c_2, ..., c_m}, c, c_top);
    end
}
```

그림 13 패싯화된 시소러스 구축 절차

러스 $f \in F$ 내 패싯 개념 용어들이 갖는 유의어들, 또는 부분어들을 이용하여 변환된 패싯 시소러스 f' 를 $T_f^{c_{top}}$ 에 적용함으로써 새로운 패싯화된 시소러스 $T_{f'}^{c_{top}}$ 를 자동으로 생성할 수 있다. 예를 들어, 그림 16은 유의어와 부분어 관계를 이용하여 패싯화된 시소러스 $T_{\{pl,os\}}^{Software}$ 를 확장한 화면이다. 여기서 $\sigma_{유의어}: F \rightarrow F$ 와 $\sigma_{부분어}: F \rightarrow F$ 는 $f \in F$ 내 각 패싯 개념 용어를 해당 유의어, 그리고

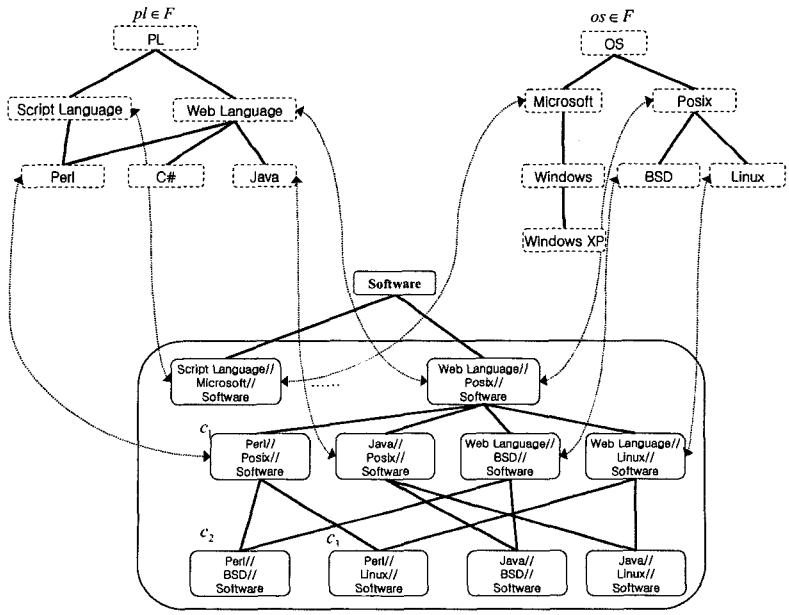

그림 14 'Perl' 삭제에 의한 패싯화된 시소러스 재구성 1

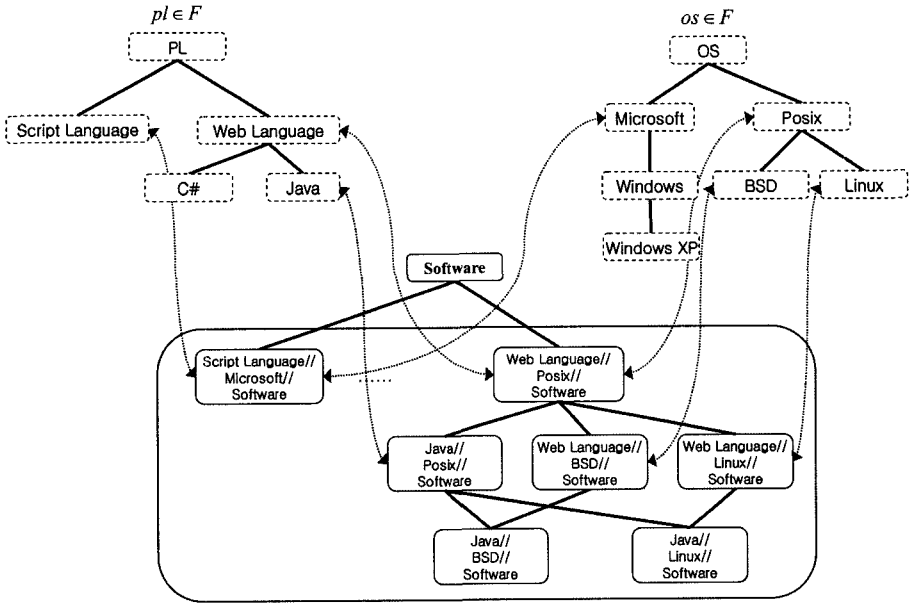

그림 15 'Perl' 삭제에 의한 패싯화된 시소러스 재구성 2

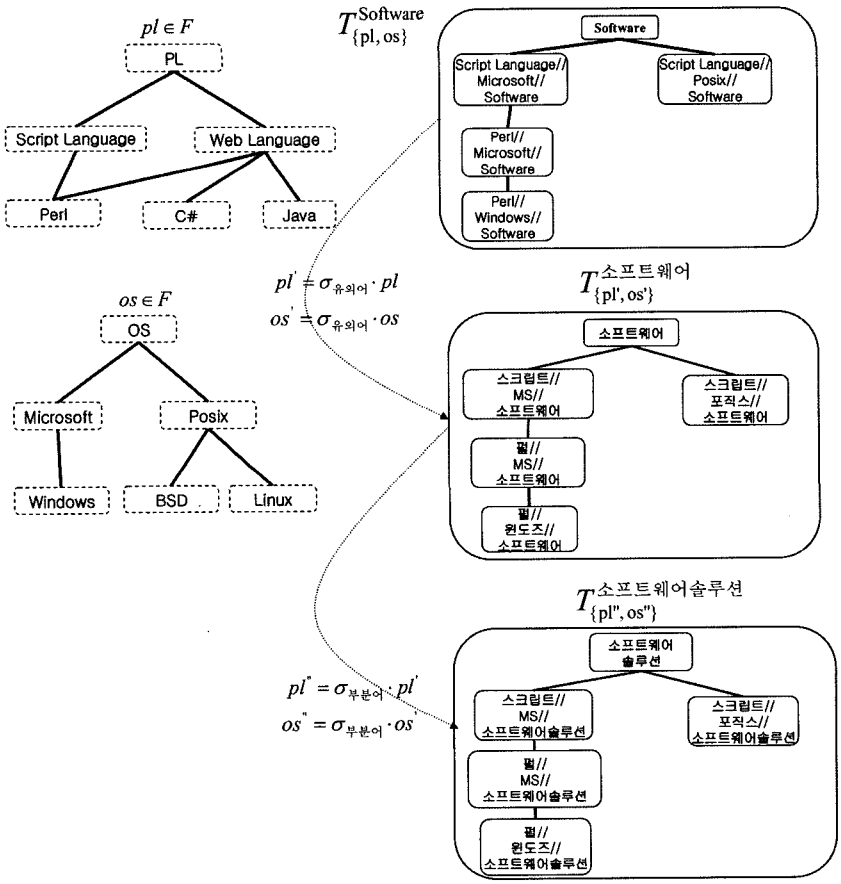

그림 16 유의어, 부분어 패싯들에 의한 패싯화된 시소러스 확장

해당 부분어로 각각 바꾸어 주는 함수들이다.

지금까지 패싯 시소러스 집합과 패싯화된 시소러스 집합에 대해 설명했다. 이를 포함한 패싯 시소러스 시스템의 전체 정의는 다음과 같다.

정의 7. 모든 $c_{top} \in C$ 에 대해 생성된 패싯화된 시소러스들의 집합이 T_F일 때, 시소러스 T는 다음과 같이 정의된다.

$$T =< F \cup T_F, OP, Q_T >$$

여기서, F는 패싯 시소러스 집합, T_F는 패싯화된 시소러스 집합, OP는 연산자 집합 그리고 Q_T는 참조 질의 집합이다.

다음 절에서는 시소러스 T의 연산자 집합 OP와 참조 질의 집합 Q_T에 대해 기술한다.

3.3 패싯 기반 참조 질의 기법

시소러스 참조 질의, 또는 간단히 질의 $q_T \in Q_T$는 패싯화된 개념 용어와 인스턴스 객체들을 내부적으로 검색할 때 사용되며, 다음의 정의들에 따라 작성된다.

정의 8. 패싯 시소러스 $f \in F$에 대한 패싯 질의 요소(FacetQueryFactor) 집합 Q_f는 다음과 같이 정의된다.

1) $f \in F$에 대해, $c \in C_f$, $C \in 2^{C_f}$ 일 때,
   c, $GLBS(C)/C_f$, $LUBS(C)/C_f \in Q_f$.
2) $q_{f_1}, q_{f_2}, \ldots, q_{f_i} \in Q_f$ 라면, $GLBS(\{q_{f_1}, \ldots, q_{f_i}\})/C_f$, $LUBS(\{q_{f_1}, \ldots, q_{f_i}\})/C_f \in Q_f$.
3) 다른 어떤 것도 Q_f 가 될 수 없다.

Q_T는 연산자 집합, OP ={GLBS,LUBS,&,|}을 사용하는 데, Q_T를 정의하기 위해서는 Q_f에서 사용하는 GLBS(Greatest Lower Bound Set, 최대 하계 집합)와 LUBS(Least Upper Bound Set, 최소 상계 집합) 연산자들을 먼저 정의할 필요가 있다. 이 두 연산자들은 각각 다음의 LBS(Lower Bound Set)와 UBS(Upper Bound Set)를 이용하여 정의된다.

정의 9. 패싯 시소러스 $f \in F$에 대한 임의의 패싯 개념 용어 집합 $C \in 2^{C_f}$ 의 LBS와 UBS는 다음과 같이 정의된다.

$$LBS(C) = \bigcup_{\forall c \in C} sub(c),\ UBS(C) = \bigcup_{\forall c \in C} sup(c)$$

여기서

$$sub(c) = \{c' \in C_f \mid c = c' \vee < c', c, \text{subConceptOf} > \in f\},$$

$$sup(c) = \{c' \in C_f \mid c = c' \vee < c, c', \text{subConceptOf} > \in f\}.$$

정의 10. $C \in 2^{C_f}$ 라고 할 때,

$$GLBS(C) = LBS(C) - \{c \mid c, c' \in LBS(C), c \le c'\},$$

$$LUBS(C) = UBS(C) - \{c \mid c, c' \in UBS(C), c \ge c'\}.$$

정의 11. 시소러스 질의 $q_T \in Q_T$는 q_T : [*Qualification query*, *Scope*]로 표현되며, 한정 질의(*Qualification query*) q 와 영역(*Scope*)은 다음과 같이 주어진다.

1) $q = \&_{i=1}^{m} q_i$ 또는 $q = |_{i=1}^{m} q_i$
2) $Scope = c$

여기서 q 에서의 각 q_i는 패싯 질의 요소, *Scope* 내 c는 c_{top} 또는 유효한 패싯화된 개념 용어이다.

한정 질의는 패싯 개념 용어들의 조합을 통해 패싯화된 시소러스에 대한 검색 조건을 지정하며, 영역은 패싯화된 시소러스의 여러 패싯화된 개념 용어 계층들 중 하나를 명시함으로써 한정 질의의 적용 영역을 설정한다. &는 한정 질의에서 명시된 서로 다른 패싯 시소러스들 내 패싯 개념 용어들을 영역에 동시 모두 적용, |는 영역에 선택적으로 적용함을 의미한다.

예 8. 시소러스 질의 q_T = [(Script Language GLBS Web Language)/PL & Posix/OS, Software]는 사용자가 $T^{\text{Software}}_{\{pl, os\}}$ 계층 내의 모든 패싯화된 개념 용어들 중에서 Script Language GLBS Web Language = {Perl} $\in C_{pl}$과 Posix $\in C_{os}$로 조합된 패싯화된 개념 용어들의 모든 인스턴스 객체 집합을 검색하고자 할 때 사용되며, q_T = [(Perl LUBS Java)/PL & Linux/OS, Software]는 사용자가 $T^{\text{Software}}_{\{pl, os\}}$ 계층 내의 모든 패싯화된 개념 용어들 중에서 Perl LUBS Java ={Web Language} $\in C_{pl}$과 Linux $\in C_{os}$로 조합된 패싯화된 개념 용어들의 모든 인스턴스 객체 집합을 검색하고자 할 때 사용된다.

위의 예에서, *Scope*가 항상 c_{top}이 될 필요는 없음을 유념해 보자. 예를 들어, 질의 q_T = [(Perl LUBS Java)/PL & Linux/OS, Web Language//Posix//Software]라면, 검색의 대상이 되는 시소러스 계층은 $T^{\text{Web Language//Posix//Software}}_{\{pl, os\}}$ 이 될 것이다.

한정 질의 q가 $q = \&_{i=1}^{m} q_i$로 주어질 때, 각 패싯 질의 요소 q_i의 질의 결과로 얻어지는 개념들의 집합은 $\| q_i \|$로 표기하며 $\| q_i \| \in 2^{C_f}$ 이다. $\| q_i \|$를 얻기 위한 함수는 GetFacetQueryFactorConcepts(q_i) 인데, 예를

들어, GetFacetQueryFactorConcepts ((Script Language GLBS Web Language))= {Perl}이 된다. 이 함수를 구현하기 위한 자세한 알고리즘은 [21]를 참조하기 바란다.

정의 11에 따르면, 각 $\| q_i \|$는 관점별로 q_T의 영역 c를 한정짓게 된다. 예를 들어, 예 8의 q_T = [Perl/PL & Posix/OS, Software]에서 Perl은 $\text{facetedConceptOf}_{pl}$ 링크를 통해서, Posix는 $\text{facetedConceptOf}_{os}$ 링크를 통하여 $T^{\text{Software}}_{\{pl,os\}}$ 내 해당 패싯화된 개념 용어들을 각각 한정짓는다(관련 함수는 GetFactededConcepts). q_T의 질의 처리 결과 얻어지는 객체들은 각 $c_i \in \| q_i \|$에 의해 한정되는 패싯화된 개념 용어들의 인스턴스 집합이 되는데, 이 결과는 $\| q_T \| \in 2^I$로 표기한다. 다음은 $\| q_T \| \in 2^I$를 구하기 위한 프로시저 ReferentialQuery-Evaluation 을 보이고 있다.

```
ReferentialQuery Evaluation(q_T :[q, scope])
{
 q = q_1 & q_2 & .. & q_m;
 For each C = GetFacetQueryFactorConcepts(q_i), i = 1,..., m
     For each facet concept c in C
          C' = GetFacetedConcepts(c, scope) ∈ C_F^valid
          if (C' = φ or C' ⊆ InvalidFacetedConceptList) then return ("invalid query");
          FacetedConceptList_i = FacetedConceptList_i ∪ C';
     end
 end
 FacetedConceptList = FacetedConceptList_1 ∩...∩ FacetedConceptList_m
 For each faceted concept c' in FacetedConceptList
     I = InstancesOf(c');
     InstanceList = InstanceList ∪ I;
 end
 return(InstanceList);
}

GetFacetedConcepts(c, scope)
{
 FacetedConceptList = {};
 C' = facetedConceptOf(c, scope);
 FacetedConceptList = FacetedConceptList ∪ C';
 For each c'_sub where c' ∈ C'
    if c = comp(c'_sub) then FacetedConceptList = FacetedConceptList ∪ {c'_sub};
       // c 의 facetedConceptOf 링크를상속받는패싯화된 개념용어들을검색
 end
 return(FacetedConceptList);
}
```

그림 17 패싯 기반 검색 프로시저

정의 8에 따르면 패싯 질의 요소 q는 $q = \|_{i=1}^{m} q_i$가 될 수도 있다. 이 경우, 질의 처리 과정은 $q = \&_{i=1}^{m} q_i$와 모두 유사하고 ReferentialQueryEvaluation에서 FacetedConceptList = FacetedConceptList$_1$ ∪...∪ FacetedConceptList$_m$ 부분만 다르다. 중복을 피하기 위해 이 부분의 기술은 생략한다.

예 9. 그림 18은 시소러스 질의 q_T = [Perl/PL & Posix/OS, Software]의 검색 결과로 얻는 $\| q_T \|$={ Fink, OpenCA, NAC }을 보이고 있다.

먼저, GetFacetedConcepts(Perl,Software)={Perl//Microsoft//Software(c_3), Perl//Posix//Software(c_5)}, GetFacetedConcepts(Posix,Software) = {Script Language//Posix//Software(c_2),c_5} 이므로, FacetedConceptList= {c_5}이다. 따라서 ReferentialQueryEvaluation(q_T) = InstanceOf(Perl//Posix//Software) = $\| q_T \|$. 여기서 facetedConceptOf (Posix,Software)= {c_2} 인데, c_5 ∈ GetFacetedConcepts(Posix,Software)인 이유는 c_5가 c_2의 링크를 상속받아 사용하고 있기 때문이다. 이 상속 관계는 comp함수에 의해 감지될 수 있는데, 즉 GetFacetedConcepts 프로시저에서 Posix =comp(c_5) 이므로 {c_5} ∈ FacetedConceptList이다.

계속해서, 그림 19는 시소러스 질의 q_T = [Web Langauge/PL & Linux/OS, Web Language//Posix//Software] 일 때, $\| q_T \|$={ OpenCA, NAC, JNode, Openfiler }을 보이고 있다.

4. 객체기반 패싯 시소러스 시스템의 구현

이 절에서는 다차원 관점을 지원하는 객체기반 패싯 시소러스 시스템의 구현에 대해 기술한다. 구현된 시스템은 다차원으로 패싯화된 개념 용어들간의 복잡한 관련성을 시각적으로 적절히 추상화함으로써 효과적으로 시소러스의 복잡한 구조를 추적 관리할 수 있도록 지원한다. 시소러스 저장을 위해서는 관계형 데이타베이스 MySql 을 사용하였으며, 사용자 인터페이스는 Visual C++을 이용하였다.

4.1 시스템 스키마 설계

패싯 시소러스의 구축 및 검색을 위한 스키마 구조는 그림 20과 같다. 테이블은 패싯 개념 용어, 패싯화된 개념 용어, 인스턴스 그리고 이들간의 관계성을 저장하는 테이블로 구성된다. 패싯 개념 용어 테이블은 Facet-Concept, 패싯화된 개념 용어 테이블은 FacetedConcept

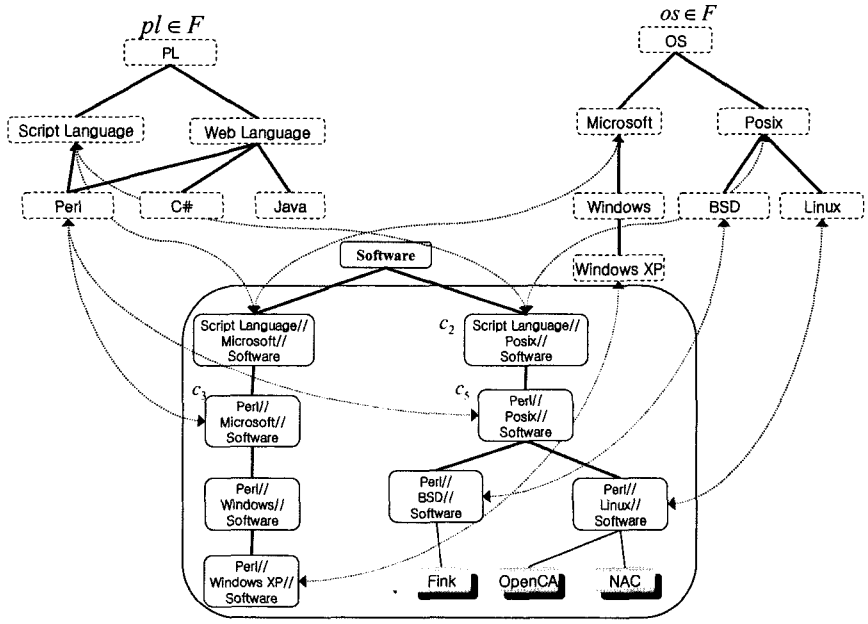

그림 18 질의 처리 결과

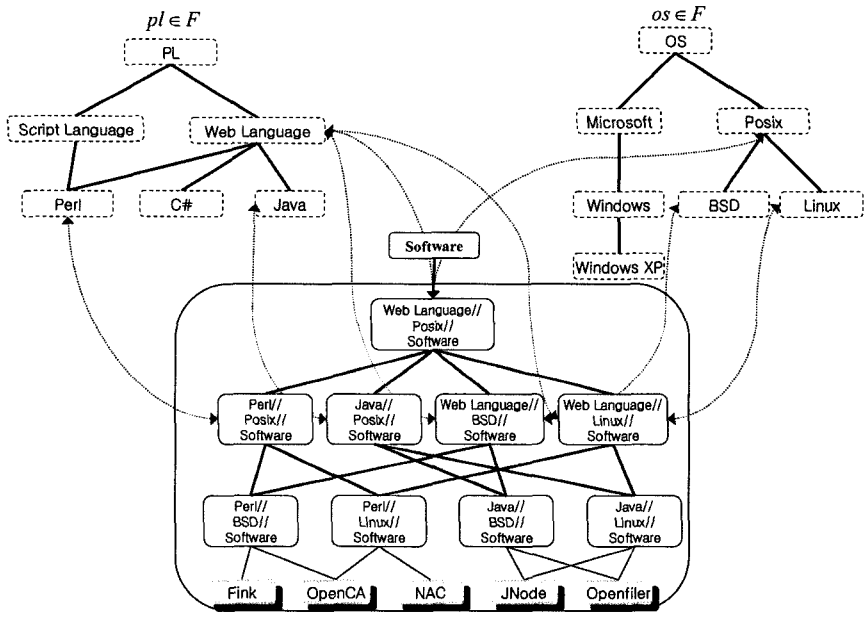

그림 19 질의 처리 결과

그리고 인스턴스 테이블은 Instance이다. 관계성 테이블은 패싯 시소러스와 패싯화된 시소러스 구조를 저장하는 테이블과 패싯 시소러스와 패싯화된 시소러스간의 관계성을 저장하는 테이블로 구분된다. 전자는 각각 FacetThesaurus와 FacetedThesaurus, 후자는 Faceted-Relationship이다.

예를 들어, q_T = [Perl/PL & Posix/OS, Software] 질의 처리는 다음과 같은 순서로 처리된다. 먼저, 패싯 개념 용어 Perl$\in C_{pl}$(ID='5')는 FacetedRelationship 테이블의 FacetConcpetID로부터 facetedConceptOf 링크 값(FacetedRelationshipID="23,32")을 얻는다. 다음으로 $T^{\text{Software}}_{\{pl,os\}}$ 의 FacetedConcept 테이블로부터 패싯화된 개

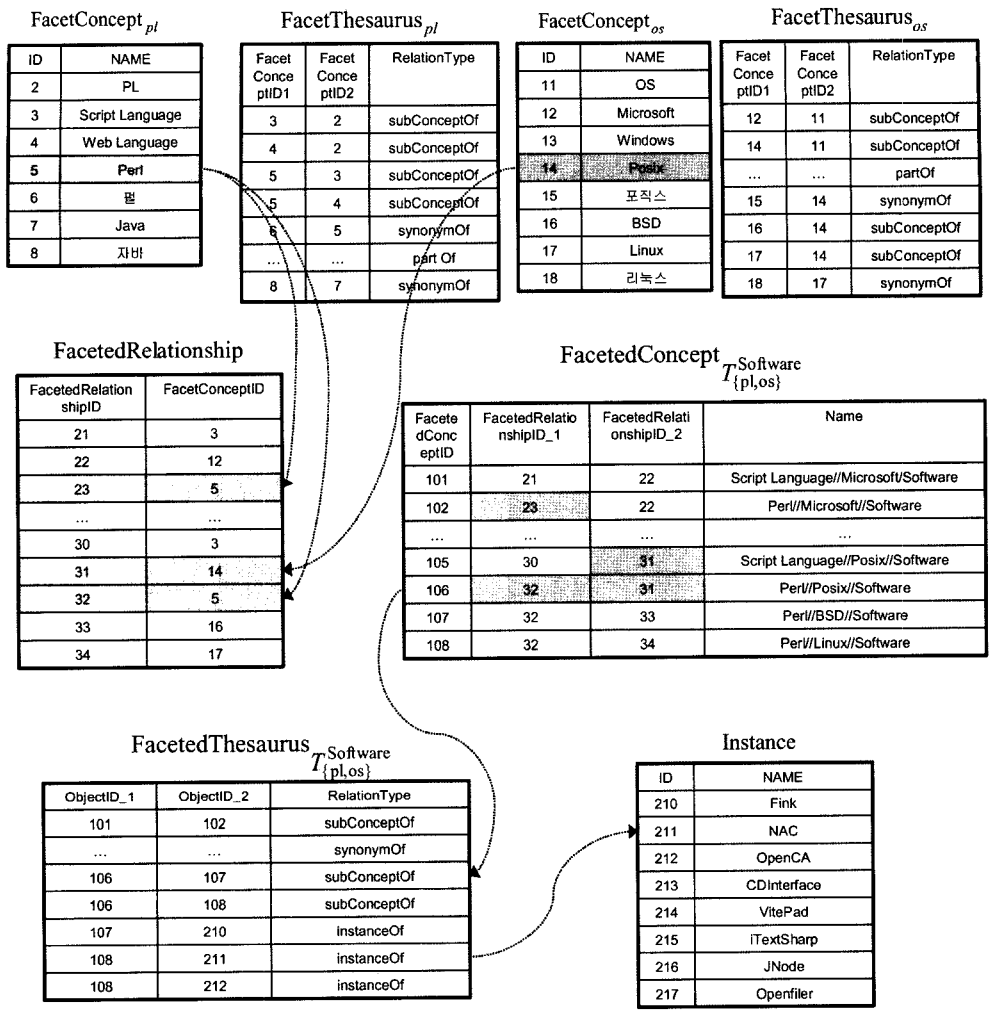

그림 20 스키마 구조

념 용어 값(FacetedConceptID= "102,106")을 얻는다. 같은 방식으로, 패싯 개념 용어 Posix$\in C_{os}$ (ID='14')는 facetedConceptOf 링크 값(FacetedRelationshipID='31')을 얻고, 계속해서, FacetedConcept 테이블로부터 패싯화된 개념 용어 값(FacetedConceptID= '105')를 얻는다. 여기서 프로시저 ReferentialQueryEvaluation의 Get-FacetedConcepts 함수에 의해 링크를 상속받는 패싯화된 개념 용어 값(FacetedConceptID= '106')이 감지되기 때문에 (예 9 참조) 두 facetedConceptOf 링크에 의해 한정되는 패싯화된 개념 용어 값(FacetedConceptID= '106')을 얻을 수 있다. 마지막으로, FacetedThesaurus 테이블로부터 FacetedConceptID '106'의 하위 패싯화된 개념 용어(ObjectID="107,108")와 InstanceOf 관계를 맺는 인스턴스 값 '210,' '211' 그리고 '212'를 얻는다. 즉 $\| q_T \|$={ Fink, NAC, OpenCA }.

4.2 시스템 인터페이스

4.2.1 다차원 패싯 시소러스 구축

패싯 시소러스 생성을 위한 패싯 개념 용어 생성, 인스턴스 생성등을 포함한 다차원 시소러스 관계성의 구축 및 유지 기능은 모두 시각화된 환경에서 지원되는데, 예를 들어, 'Software'에 속하는 인스턴스들이 'PL,' 'OS' 그리고 'Function' 관점을 갖는 경우, 각 관점에 따라 독립된 화면에서 패싯 개념 용어들을 구축할 수 있다. 그림 21은 'OS' 관점의 패싯 개념 용어를 구축하는 화면이다.

'Linux'는 'Posix'의 하위 패싯 개념 용어로 추가된다. 하위 패싯 개념 용어 필드는 패싯 개념 용어 이름을 입력하고 비고 필드는 같은 이름을 갖는 패싯 개념 용어

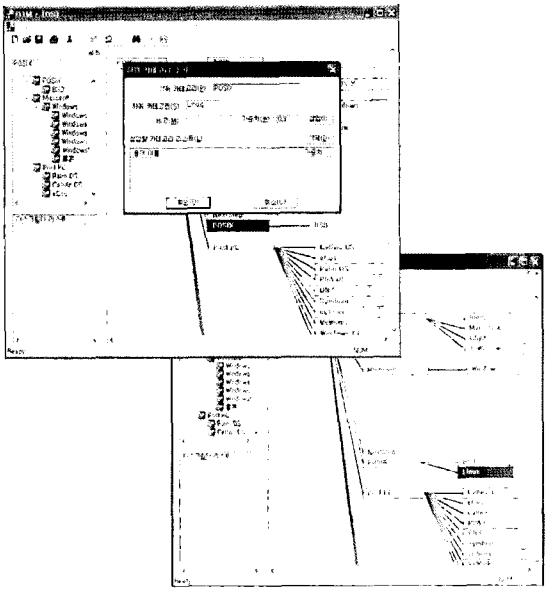

그림 21 패싯 개념 용어 추가

나 인스턴스가 존재할 때 구분하기 위한 구분자를 입력하면 된다. 이와 같이 구축된 시소러스 관계성(BT/NT)에지들은 각 패싯 관점에 따라 spreading activation 방식으로 확장/축소되므로 패싯 시소러스가 적절히 추상화된다.

그림 22는 최상위 패싯 개념 용어 'Software'의 인스턴스로 'CDInterface'를 추가하는 화면이다.

다음은 관점에 따라 계층화된 패싯 개념 용어들의 조합 색인에 의한 다차원 패싯화된 시소러스를 동적으로 구축하는 예이다. 즉 일련의 패싯들을 대상 인스턴스들에 적용 색인함으로써 특정 관점의 시소러스를 자동으로 구축한다. 예를 들어, 'Software'에 속하는 인스턴스들을 대상으로 'PL'과 'OS' 관점을 적용하면 그림 23과 같이 패싯화된 시소러스가 자동으로 구축된다.

4.2.2 패싯화된 시소러스 브라우징

패싯 조합에 의해 자동으로 구축된 패싯화된 시소러스의 브라우징은 다차원으로 패싯화된 개념 용어 간의 복잡한 관계를 적절히 시각화함으로써 효과적으로 시소

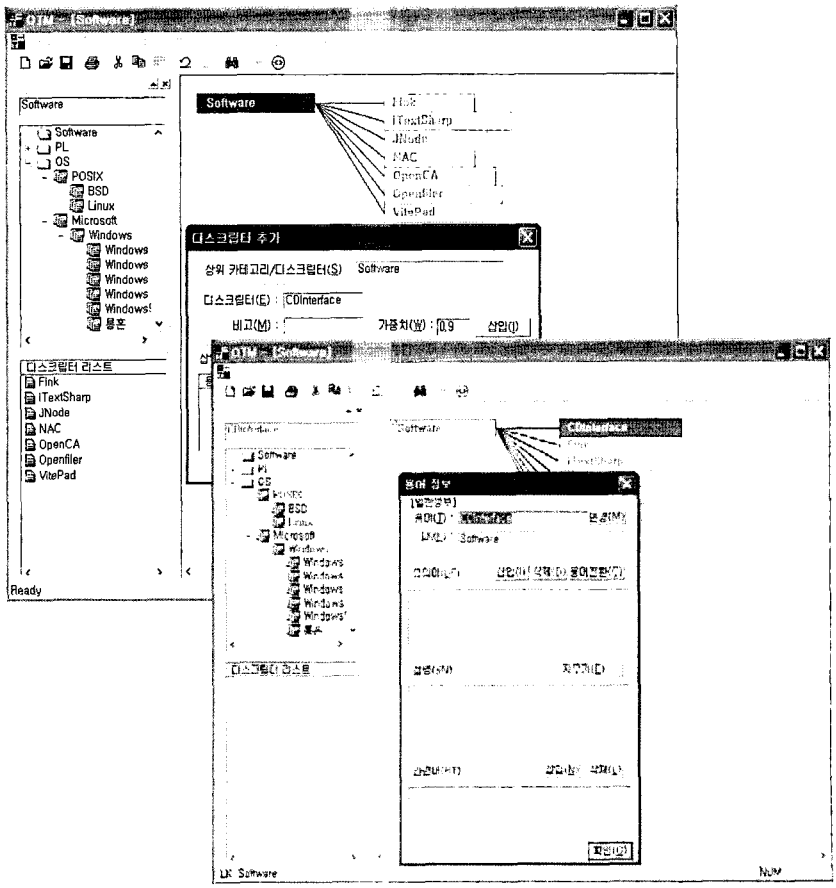

그림 22 인스턴스 추가

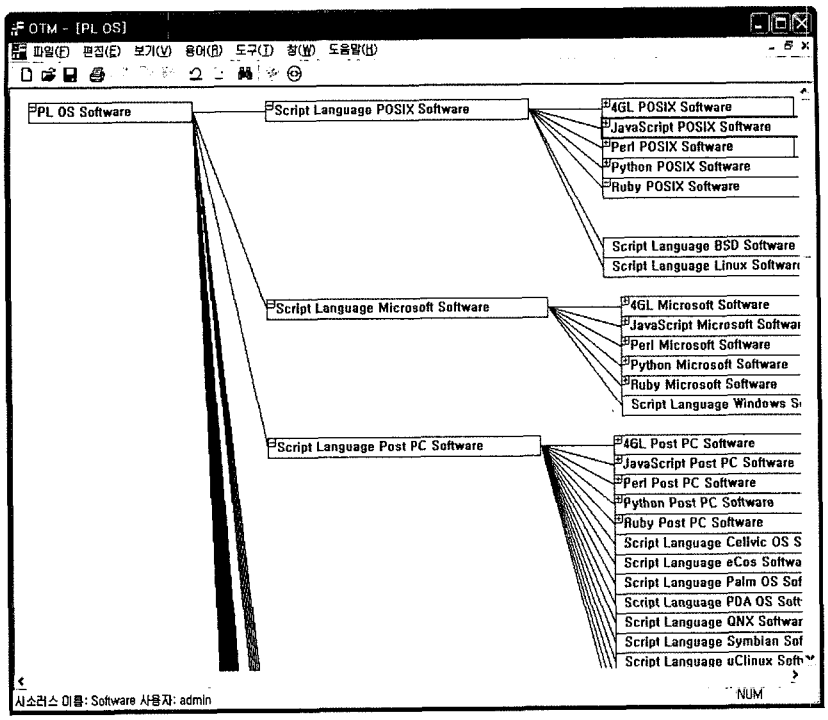

그림 23 자동 구축된 패싯화된 시소러스

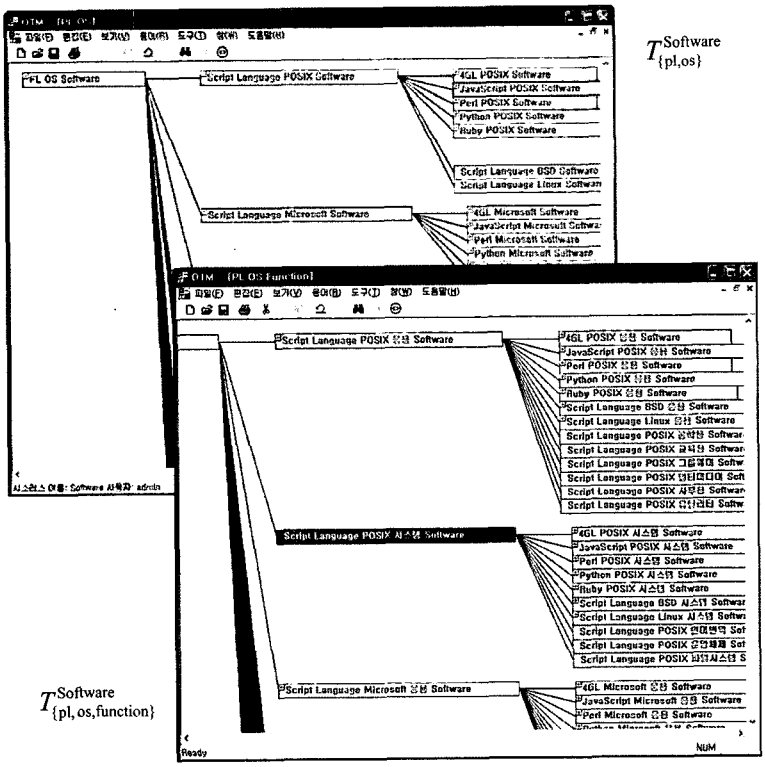

그림 24 패싯화된 시소러스 브라우징

러스 구조를 추적 관리할 수 있다. 그림 24는 $T^{Software}_{\{pl,os\}}$ 과 $T^{Software}_{\{pl,os,function\}}$ 를 각각 브라우징할 때 보여지는 결과이다.

4.3 성능평가 및 고찰

4.3.1 실험 집단

본 연구의 성능 평가를 위해 실험에서는 소프트웨어의 세 관점(PL, OS, Function)에 따라 패싯 개념 용어를 생성하고, 이들의 조합을 통해 자동으로 패싯화된 시소러스를 구축한 결과를 보이고 있다. 4번의 실험에 쓰인 각 패싯 개념 용어들의 합은 100, 150, 200, 그리고 250개이며, 각 관점에 할당된 패싯 개념 용어들은 표 1과 같다. 실험에 사용된 패싯 개념 용어들은 그림 25와 같이 최대 5 단계까지의 계층 구조로 이루어졌다.

표 1 실험 집단

	100	150	200	250
PL	40	50	60	80
OS	30	50	70	80
Function	30	50	70	90

4.3.2 성능평가 결과

표 2는 세 관점의 패싯 용어들간 무조건적인 조합 결과와 비양립 쌍 제거를 통해 유효한 패싯화된 개념 용어들만을 생성하는 본 방식의 조합 결과를 보여 준다. 예를 들어, 250개 무조건 조합의 경우 57만6000개의 개념 용어가 생성된 반면, 본 방식에서는 10만 4930개의 개념 용어들만이 생성되었다.

표 2 조합 결과

	100	150	200	250
무조건 조합	36000	125000	294000	576000
본 방식 조합	9715	33670	64620	104930

그림 26은 각 방식에서 개념 용어들을 생성하는 데, 소요되는 시간을 비교한 것으로, 본 방식이 생성되는 용어수가 많아질수록 상대적으로 효율적임을 보여준다. 즉, 무조건적인 조합의 경우 용어수가 증가할수록 본 방식과의 생성 시간 격차가 현저히 커지는 것을 알 수 있다.

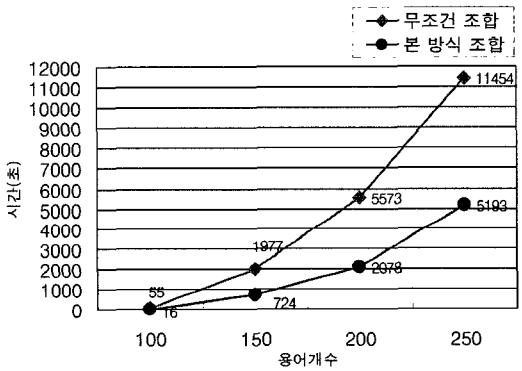

그림 26 패싯화된 시소러스 생성 시간

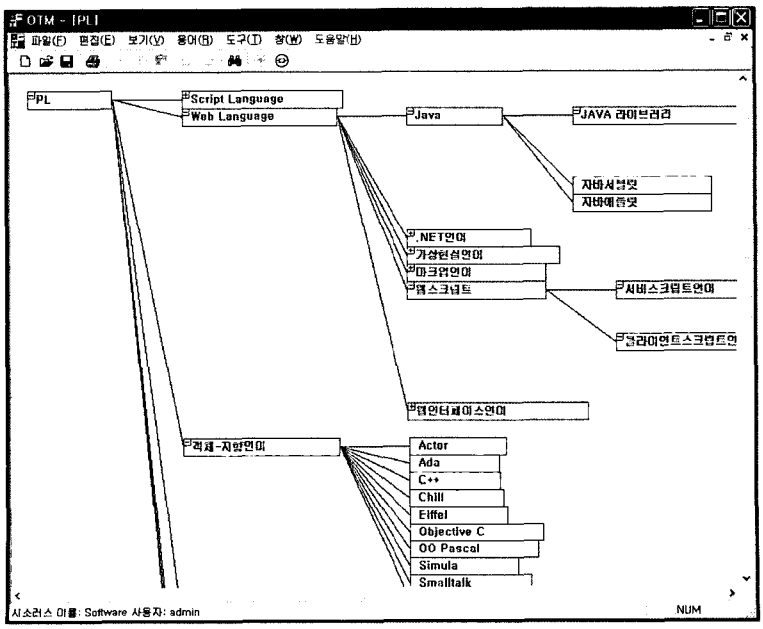

그림 25 PL 관점의 패싯 개념 용어들

다음으로, 그림 27은 두 방식에 의해 생성된 시소러스들의 정확률을 비교한 것으로, 여기서 시소러스의 정확률은 생성된 패싯화된 개념 용어들 대비 유효한 개념 용어들의 비율로 정의한다.

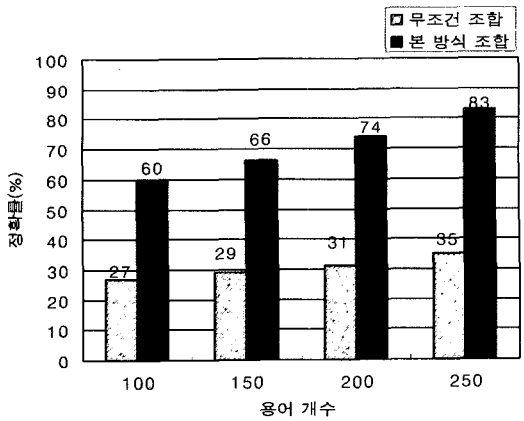

그림 27 패싯화된 시소러스에 대한 정확률

유효한 개념 용어들을 판단하는 과정에서 복합용어사전을 사용한 결과는 성능평가에 반영하지 않았는데, 그 이유는 성능평가가 이용하는 사전의 용어 수에 지나치게 종속적이기 때문이다. 따라서 사전을 사용할 경우, 그림 27에 보인 본 방식의 정확률은 무조건 조합에 비해 보다 현격한 차이를 보일 것으로 예상된다. 예를 들어, 사전에 의해 유효한 개념 용어들이 20% 정도 선택된다고 가정할 경우, 두 방식의 정확률은 대략 다섯 배의 차이를 보이게 될 것이다.

5. 결론 및 향후 연구 과제

본 논문에서는 다차원 관점을 지원하는 객체기반 패싯 시소러스 시스템을 설계 및 구현하였다. 이 시스템은 다차원으로 패싯화된 개념간의 복잡한 관계를 적절히 시각화함으로써 효과적으로 구조를 추적 관리할 수 있고 패싯 조합으로 새로운 차원의 시소러스를 용이하게 추가함으로써 융통성 있는 시소러스 확장이 가능하다. 계층화된 패싯 개념 용어들의 조합 색인에 의한 다차원 가상 시소러스 트리의 동적 생성이 가능하며 다차원 가상 시소러스의 차원에 따른 브라우징 및 항해를 지원한다. 본 연구는 시각화된 환경에서 사용자가 원하는 관점에 따라 다차원 패싯 시소러스를 구축하고 항해 검색할 수 있는 진보된 시소러스 시스템을 체계화하기 위한 모델의 정형화를 처음으로 시도하였고 이를 바탕으로 모델의 실용성을 입증하기 위한 프로토타입을 구현했다는 점에서 큰 의미가 있다고 본다.

계속적으로 이루어져야 할 향후 연구로, 패싯화된 시소러스 자동 구축을 통해 생성된 패싯화된 개념 용어의 유효성 평가가 필요하다. 또한 패싯 시소러스를 이용한 웹 검색 서비스를 위해 패싯 개념 용어를 사용자에게 제시해 줄 수 있는 시각화 인터페이스 모델이 필요하며 패싯 개념 용어들간 다양한 관계를 지원하여 질의시 검색 결과의 재현율과 정확률을 높일 수 있는 방안이 필요하다.

참 고 문 헌

[1] M. Hancock-Beaulieu, M. Fieldhouse and T. Do, "An Evaluation of Interactive Query Expansion in an Online Library Catalogue with a Graphical User Interface," Journal of Documentation, Vol. 5, No. 3, pp. 225-245, 1995.

[2] 이재윤, 김태수, "WordNet과 시소러스", 언어 정보의 탐구, pp. 232-269, 1999.

[3] Bowker, Lynne and Lethbridge, Timothy C, "Terminology and Faceted Classification: Applications Using CODE4," Advances in Knowledge Organization, Vol. 4. (Proceedings of Third International ISKO Conference, Copenhagen, 20-24 June 1994.

[4] Chan L., Childress E., Dean R., O'Neill. & Vizine-Goetz D, A faceted approach to subject data in the Dublin Core metadata record. Journal of Internet Cataloging, pp. 35-47, 2001.

[5] Pollitt A., Interactive information retrieval based on facetted classification using views, Proc. 6th International Study Conference on Classification, London:51-56, 1997.

[6] Prisss. Uta, Mustafa el Hadi, Maniez and Pollitt, "A Graphical Interface for Conceptually Navigating Faceted Thesauri," Structures and Relations in Knowledge Organization. Proceedings of 5th Int. ISKO Conf., Advances in Knowledge Organization, Vol. 6, 1998, pp. 184-190.

[7] A production of Complete Information Architecture., FacetMap, "http://facetmap.com," 2003.

[8] FACET Research Project, University of Glamorgan, "http:web.glam.ac.uk/schools/soc/research/hypermedia/facet_proj/index.php," 2002.

[9] J.Paul, "Art and Architecture Thesaurus," Getty Trust, "http://www.getty.edu/research/tools/vocabulary/aat/," 2004.

[10] E. J. Hunter, "Classification Made Simple," Aldershot: Gower, 1988.

[11] J. Aitchison and Patricia, "Thesaurofacet: a thesaurus & faceted classification for engineering & related subjects," English Electric Co. Ltd., 1969.

[12] The A.C.S. Active Classification Solutions, "http://www.termtree.com.au," 2005.

[13] The Brain Technologies Corp., "http://www.thebrain.com," 2006.

[14] The Data Harmony, Inc., "http://www.dataharmony.com/products/tm.htm," 2006.

[15] The Hector Echeverria, Technical Marketing Director, Multisystems, "http://www.multites.com," 2006.
[16] Alan Gilchrist and Barry Mahon, Information Architecture: Designing Information Environments for Purpose, Facet Publishing, 2004.
[17] The WebChoir, Inc., "http://www.webchoir.com," 2005.
[18] Yannis T., Nicolas S., Panos C. & Anastasia A., Extended Faceted Taxonomies for Web Catalogs, 2002.
[19] 정영미, 김명옥, 이재윤, 한승희, 유재복, "과학기술 분야 통합 개념체계의 구축 방안연구", 한국정보관리학회지, 1013-0799, 19(1): 135-161, 2002.
[20] 황순희, 정한민, 성원경, "패싯(Facet)을 이용한 과학기술분야 시소러스 구축과 활용방안", 정보관리연구, Vol. 37, No. 3, pp. 61-84, 2006.
[21] Jae Hun Choi, Jae Dong Yang, Dong Gil Lee, "An Object-Based Approach to Managing Domain Specific Thesauri: Semiautomatic Thesaurus Construction and Query-Based Browsing," Int'l Journal of Software Engineering & Knowledge Engineering, Vol.10, No.4, pp 1-27, 2002.

김 원 중

1999년 전북대학교 전자계산학과 졸업(학사). 2001년 전북대학교 대학원 전산통계학과(석사). 2004년~현재 전북대학교 대학원 컴퓨터통계정보학과 박사수료. 관심분야는 정보검색, 온톨로지, 시맨틱 웹, CASE

양 재 동

1983년 서울대학교 컴퓨터공학과(학사) 1985년 한국과학기술원 전산학과(석사) 1991년 한국과학기술원 전산학과(박사) 1995년~1996년 Univer.of Florida, Visiting Scholar. 현재 전북대학교 전자정보공학부 교수, 영상·정보 신기술연구소 연구원. 관심분야는 OODBs, Expert System, CASE, 온톨로지, 시맨틱 웹