계층적 속성문법을 위한 효율적인 점진적 속성평가

Effictive incremental attribute evaluation for a hierarchical attribute grammar

장 재 춘* 김 태 훈**
Jae-Chun Jang Tae-Hoon Kim

요 약

점진적 속성 평가 알고리즘에서는 새로운 입력 트리가 기존 입력 트리와 정확히 비교되어서 새로운 트리를 구성할 때 기존 속성 트리의 어떤 서브 트리를 사용해야 하는가를 결정한다.

이 논문에서는 계층적 속성 문법의 점진적 평가를 효율적으로 하기 위해 점진적 속성 평가 알고리즘을 이용하였으며 Carle과 Pollock의 알고리즘을 분석하여 점진적 속성 평가 알고리즘으로 재구성하고, 속성 트리 dcopy의 구성요소를 새로운 속성 트리 d'copy에 적용하여 최적화된 속성 트리 d'copy의 점진적 속성 평가 알고리즘을 구성하였다. 또한 점진적 속성 평가 알고리즘을 이용하여 실제적인 입력 프로그램에서 재사용된 노드의 표현과 정의된 변수 형(type)이 어떻게 점진적인 속성 평가를 수행하는가를 나타내었다.

Abstract

In incremental attribute evaluation algorithm, a new input attribute is exactly compared with a previous input attribute tree, and then determine which subtrees from the old should be used in constructing the new one.

In this paper incremental attribute evaluation algorithm was used to make incremental evlauation of hierarchical attribute grammar more efficiently, and reconstructing the incremental attribute evaluation algorithm by analyzing that of Carle and Pollock, finally the incremental attribute evaluation algorithm for optimalized attribute tree d'copy was constructed by applying element of attribute tree dcopy to a new attribute tree d'copy. Also proving that how the reused nod and type of defined parameter in input program carried out the incremental attribute evaluation by using that algorithm.

1. 서 론

속성 문법은 모든 방향의 자료전달을 표현할 수 있으며, 비 단말간의 자료전달이 명확히 기술된다. 이러한 자료를 속성이라 하고 자료의 흐름은 속성의 전달에 의해서 이루어진다. 그러나 모든 방향으로의 속성 전달이 가능하기 때문에 상황에 따라 순환구조를 이루기도 하고 속성 전달이 파싱 방향과 관계없이 전달되므로 한번 이상의 파싱이 필요한 경우도 발생한다. 이러한 문제 때문에 속성 평가라는 방식이 도입되었고 여러 가지의 속성 평가 방법들이 제안되었다[1,2,3].

최근 몇 가지 새로운 속성 평가 방법이 개발되었는데 모두 크고 복잡한 형태가 아니라 보다 작은 속성 문법의 모듈 조합을 통해 평가하는 것을 일반적인 목표로 하고 있다.

특히, 이들 언어들은 복잡한 시스템들을 상대적으로 작은 모듈로 세분화 될 수 있도록 한다. 각각의 모듈 명세는 함수를 정의하게 되는데 이 함수는 트리나 대그(dag)로 구조화된 구문을 만들어내기 위해 입력 언어에 속성 문법 평가기를 적용함으로써 계산된다[4,5].

따라서 이러한 모듈 접근을 지원하는 새로운 속성 언어를 계층적 속성 문법(hierarchical attribute grammar)이라 하며, 모듈 접근을 지원하지 않는 속성 문법 언어를 비 계층적 속성 문법(nonhierarchical

* 정회원 : 영동전문대학 전자계산과 조교수
jcjang@yeongdong.ac.kr
** 비회원 : 영동전문대학 전자계산과 부교수
aristocrat@yeongdong.ac.kr
*** 본 논문은 2000년도 영동전문대학 산업기술연구소 학술연구비 지원을 받아 연구 되었음.

attribute grammar)으로 나타낸다. 이들 시스템들이 프로그래밍 환경에서 통합되어질 때 속성의 점진적 평가 방법이 중요해진다[6,7].

점진적 속성 평가 알고리즘은 속성 트리를 계층적 시스템의 각 모듈 인스턴스를 위해, 점진적 연산자를 데이터 베이스처럼 관리하여 필요에 따라 각 모듈 인스턴스를 갱신한다. 이미 산출한 속성 값의 재사용은 이전 속성 트리의 요소로부터 각 모듈 인스턴스를 위한 새로운 속성 트리를 생성하게 된다.

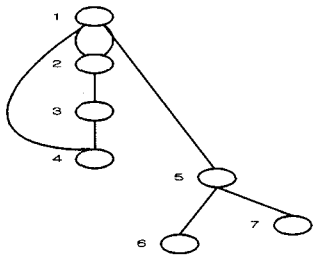
이 알고리즘에서는 새로운 입력 트리가 기존 입력 트리와 정확히 비교되어서 새로운 트리를 구성할 때 기존 속성 트리의 어떤 서브 트리를 사용해야 하는가를 결정한다. 또한, 계층적 속성 문법의 점진적 평가를 효율적으로 하기 위해서는 모듈 루트에 의해 평가되어질 구문항(syntax term)에 대해서 수정된 구문 항, 모듈 루트 바로 전에 평가된 속성 트리 그리고 바로 이전에 속성화된 구문 항에 대한 정보를 갖고 있어야 한다. 따라서 계층적 속성 문법을 위한 점진적 평가는 이전의 속성화된 트리로부터 새로이 속성화된 트리를 구축하는 과정을 참조해야 하는데 이것은 새로운 속성 트리를 구성하기 위해 재사용되는 기존 속성 트리의 서브 트리를 결정하는 역할을 담당하기 때문이다[8,9,10].

이 논문에서는 Carle과 Pollock의 속성 평가 알고리즘을 분석하여, 속성 트리 dcopy의 구성요소를 새로운 속성 트리 d'copy에 적용하여 최적의 d'copy를 구성하기 위한 점진적 평가 알고리즘으로 재구성하였다. 또한 재구성된 점진적 속성 평가 알고리즘에서 실제의 입력 변수가 어떻게 점진적인 속성 평가 알고리즘에 적용되고 재사용될 수 있는지를 보이고자 한다.

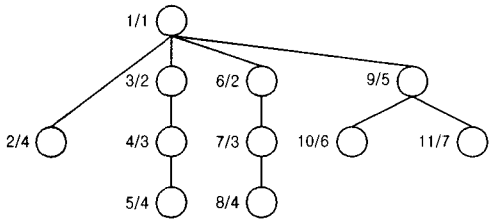
2. 관련 연구

2.1 대그 d와 속성 트리 dcopy

이 알고리즘은 하나의 모듈 인스턴스에 의해 속성화된 가장 효율적인 조건 값을 트리 형태가 아닌 대그 형태로 표현한다.



(그림 1) 대그 d



(그림 2) dcopy 트리

그래프 형태의 표현 중 대그는 공통되는 부분식들이 식별되기 때문에 같은 내용을 나타내지만 속성 트리 dcopy는 원시 프로그램의 자연스러운 계층 구조를 묘사하고, 대그는 속성 트리 dcopy보다 간결한 표현 방법이다.

대그 d의 구성은 그림 1과 같으며 대그 d는 노드 1, 2, 3, 4, 5, 6, 7과 노드간의 연결 가지로 구성된다. 대그 d의 구성요소들은 대그 d'를 구성할 때 사용한다.

대그 d를 dcopy 트리로 구성하면 그림 2와 같으며, 대그 노드는 각각 하나의 고유한 이름이 부여된다. dcopy 트리의 각 노드들은 X/Y의 라벨이 정해지며 X는 속성 트리 노드 자신을 나타내는 고유 식별자이며 Y는 트리 노드에 의해 표현되는 대그 d의 노드와 연계된 고유한 식별자이다.

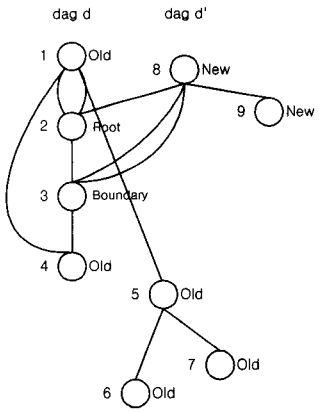
2.2 속성트리 d'copy의 구성

점진적 평가 알고리즘은 대그 d를 새로운 대그 d'에 연결한다. 대그 d와 d'의 연결은 그림 3과 같으며, 대그 d'는 새롭게 생성된 노드 8, 9를

New로 표시하고, 대그 d의 구성요소 중에서 (d'-d) 노드의 직접상속자는 Boundary로 표시하며, 이미 속성화된 모든 노드는 Old로 표시한다.

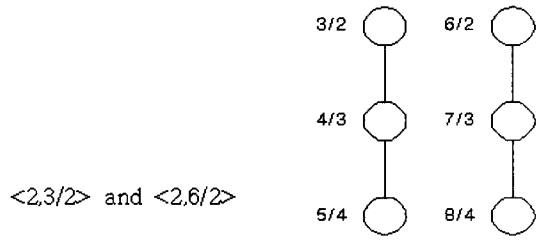
(d∩d')는 대그 d에서 나온 d와 d'가 공유하는 서브 트리를 제거하면 불완전한 대그가 구성된다. 여기서 (d'-d)는 대그 d에서 생성되지 않는 대그 d'의 노드를 나타내기 위해 사용되고, (d-d')는 대그 d'에서 생성되지 않는 대그 d의 노드를 나타내기 위해 사용된다.

속성 d에서 공유하는 서브 트리 평가에 적합한 새로운 d'copy트리를 생성한다. 따라서 대그 d를 깊이 우선 순서로 모든 경계노드를 그림 3과 같이 루트 노드로 표시하고, 공유 서브 트리에 의해 연관 사상 엔트리와 dcopy의 서브 트리를 표현한다.

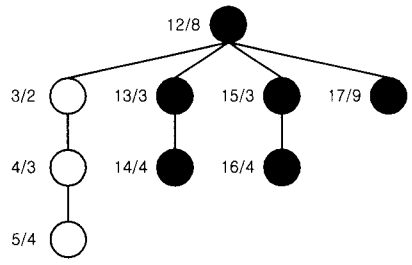


(그림 3) 대그 d와 d'

연관 사상 엔트리와 dcopy의 서브 트리를 나타내면 그림 4와 같다.



(그림 4) 연관 사상 엔트리와 dcopy의 서브 트리



(그림 5) d'copy 트리

대그 d를 새로운 대그 d'에 연결하고 속성 트리 dcopy를 새로운 속성 트리 d'copy에 연결하여 속성 평가를 구성하기 위해 연관 사상 엔트리와 dcopy의 서브 트리 적용한다.

속성 트리 dcopy의 연관 사상 엔트리와 dcopy의 서브 트리를 적용하면 그림 5와 같다.

속성 트리 dcopy의 서브 트리를 사용해 속성 트리 d'copy를 구성한다. 채워진 원은 새로운 속성 트리 노드이고, 채워지지 않은 노드는 기존의 속성 트리 노드이다.

3. 점진적 속성평가 알고리즘

점진적 속성평가 알고리즘은 기존 대그 d와 새로운 대그 d'와 대그 d에 대한 속성 트리 dcopy를 입력받아 속성 트리 d'copy를 구성한다.

속성 트리 d'copy를 구성하기 위해 각 대그 노드에 대한 PROVIDED 집합과 NEEDED 집합을 이용한다. PROVIDED 집합은 속성 트리 dcopy에서 사용된 대그 노드의 집합이며, NEEDED 집합은 속성 트리 d'copy를 구성할 때 새로이 추가되는 노드의 직접 상속자 집합이다. PROVIDED는 속성 트리 dcopy를 구성하는 각 서브 트리들의 루트 노드이며, 이것은 속성 트리 d'copy를 구성할 때 속성 트리 dcopy의 구성요소를 서브 트리 단위로 사용할 수 있게 한다.

NEEDED는 속성 트리 d'copy를 구성하는 새로운 노드에 속성 트리 dcopy의 기존 서브 트리를 삽입하거나, 새로 삽입되어야 할 노드의 레벨과

위치를 결정한다.

그림 6은 Carle과 Pollock의 속성 평가 알고리즘을 분석하여, 속성 트리 dcopy의 구성요소를 새로운 속성 트리 d'copy에 적용하여 최적의 d'copy를 구성하기 위한 점진적 평가 알고리즘으로 대그 d와 d' 속성 트리 dcopy를 입력받아 새로운 최적의 속성 트리 d'copy를 생성한다. 알고리즘에서 대그 d와 d'의 모든 대그 노드와 대그 d에 대한 속성 트리 dcopy를 입력받은 대그 노드는 스택에 의해 모든 노드들을 선형적인 순서로 위상 정렬

```
Procedure Initialize(dagnode, Tree_Dcopy)
begin
        Dag_Buffer[] ← Topol(dagnode);
        ROOT_PROV[] ← Tree_Dcopy;
        Tree_D'copy[] ← Create ⊤();
        ROOT_NEED ← ⊤ { Temporary root node }
end;

Procedure Com_Prov_Need(Dag_Buffer[])
  begin
        for dag ← Dag_Buffer[Min] to Dag_Buffer[Max] do
          begin
                Need[dag] ← NEEDED(dag);
                    {NEEDED is immediate inherited set}
                Prov[dag] ← ROOT_PROV[dag]
                    {ROOT_PROV is set of dagnode used in dcopy}
          end
        if  Need not EMPTY then
                for dag ← Dag_Buffer[Min] to Dag_Buffer[Max] do
                  Dag_NEED[dag] ← <Need[dag],i>;
        if Prov not EMPTY then
                for dag ← Dag_Buffer[Min] to Dag_Buffer[Max] do
                  Dag_PROV[dag] ← Prov[dag]
  end;

Procedure Make_D'copy(Dag_Buffer[], Dag_NEED[], Dag_PROV[],  Tree_D'copy[])
  begin
        Dag_Buffer[] ← Topol(d'_root(dagnode));
            {Call function that new dagnode is sorted topological
             algorithm}
        for dag ← Dag_Buffer[Min] to Dag_Buffer[Max] do
          begin
                if Dag_PROV[dag] is EMPTY then
                  begin
                        New_Dag ← NEW(dag);
                        Tree_D'copy[Dag_NEED[dag]] ← New_Dag
                  end
                if Dag_NEED[dag] not EMPTY then
                  Tree_D'copy[Dag_NEED[dag]] ← Dag_PROV[dag];
          end;
              Remove ⊤(Tree_D'copy[])
              {Construction of optimized attribute tree d'copy that eliminating the temporary root node ⊤}
  end;
```

(그림 6) 최적화된 속성 트리 d'copy 알고리즘

한다. 기존 속성 트리 dcopy는 속성 트리 d'copy를 구성하는데 유용하므로 속성 트리 dcopy를 PROVIDED 집합에 삽입한다.

또한, 속성 트리 d'copy를 구성하는 임시 루트 노드 ⊤를 생성하고, 이 루트 노드를 NEEDED 집합의 초기 값으로 설정한다.

위상 탐색 순서대로 정렬된 대그 노드를 입력받아서 각 대그 노드에 대한 NEEDED 집합과 PROVIDED 집합을 계산하여 계산된 NEEDED와 PROVIDED가 공집합이 아니면, 즉 대그 d와 d'가 서로 연관되어 있고 속성 트리 dcopy의 기존 노드가 속성 트리 d'copy를 구성하는데 유용하다면 Need와 Prov를 각 대그 노드에 대한 NEEDED 집합과 PROVIDED 집합에 추가한다.

입력받은 대그 노드는 대그 d'의 루트 노드를 시작으로 위상 탐색 순서대로 정렬되며, 각 대그 노드에 대한 Dag_PROV가 공집합이면 새로운 노드를 생성하여 Tree_D'copy의 Dag_NEED에 추가하고, 공집합이 아니면 해당 대그 노드에 대한 Dag_PROV를 Tree_D'copy의 Dag_NEED에 삽입한다.

이 결과에서 Dag_NEED의 값이 공집합인 경우는 해당 대그 노드가 속성 트리 d'copy를 구성하는데 필요 없는 노드임을 나타내며, Dag_PROV의 값이 공집합인 경우는 해당 대그 노드가 속성 트리 d'copy에 새로운 노드로 삽입되어야 함을 의미한다.

4. 알고리즘 적용

이 논문에서는 재구성된 속성평가 알고리즘을 이용하여 입력 프로그램에서 정의된 변수의 형(type)이 어떻게 점진적인 속성평가를 수행하는가와 재사용된 노드를 나타낸다.

그림 7은 블록으로 구성되는 프로그램 구조를 나타내는 문맥 자유 문법이다. 이 문법에서 프로그램은 <decl list>와 <stmt list> 그리고 <expr>의 각 선언부와 문장의 확장을 위한 <decl list branch>와 <stmt list branch> 그리고 <add expr>로 구성된다.

```
(1) <Program> ::= Program id <block>
(2) <decl_list> ::= <decl>|<decl_list_branch>
(3) <decl_list_branch> ::= <decl> <decl_list>
(4) <stmt_list> ::= <stmt>|<stmt_list_branch>
(5) <stmt_list_branch> ::= <stmt> <stmt_list>
(6) <decl> ::= id : TYPE
(7) <stmt> ::= <assign>|<block>
(8) <assign> ::= <id> := <expr>
(9) <block> ::= begin <decl_list> <stmt_list> end
(10) <expr> ::= <add expr>|<id>
(11) <add_expr> ::= <expr> + <expr>
(12) <id> ::= id
```

(그림 7) 입력 프로그램을 위한 문맥 자유 문법

```
Program A                    Program B

begin                        begin
  i : integer                  i : integer
  j : integer                  j : integer
      begin                        begin
        k : real                       k : real
        i := j+k                       begin
      end                                  l : real
end                                        i := j+k
                                           j := i+l
                                       end
                             end
                         end
```

(그림 8) 두 가지 입력 프로그램

그림 7의 문법에 의해 생성되는 두 가지 입력 프로그램을 나타내면 그림 8과 같다. 프로그램 A는 외부 블록에서 두 변수 i와 j를 선언하고, 내부 블록에서는 변수 k를 선언한다. 내부 블록에서는 <add_expr>에 의해 j와 k의 더하기를 수행한다. 프로그램 B는 프로그램 A보다 내부 블록을 하나 더 포함하고 있으며, 변수 l을 사용하여 변수들 사이의 더하기를 수행한다.

그림 8의 두 입력 프로그램을 대그 d와 d'로 나타내면 그림 9과 같다. 대그 d는 두개의 블록을 포함하는 프로그램을 나타내고, 대그 d'는 세 개의 블록을 포함하는 프로그램이다. 대그 d'는 세

부분의 변수 선언부를 대그 d의 노드에서 재사용한다.

대그 d에 대한 속성 트리 dcopy를 나타내면 그림 10과 같다.

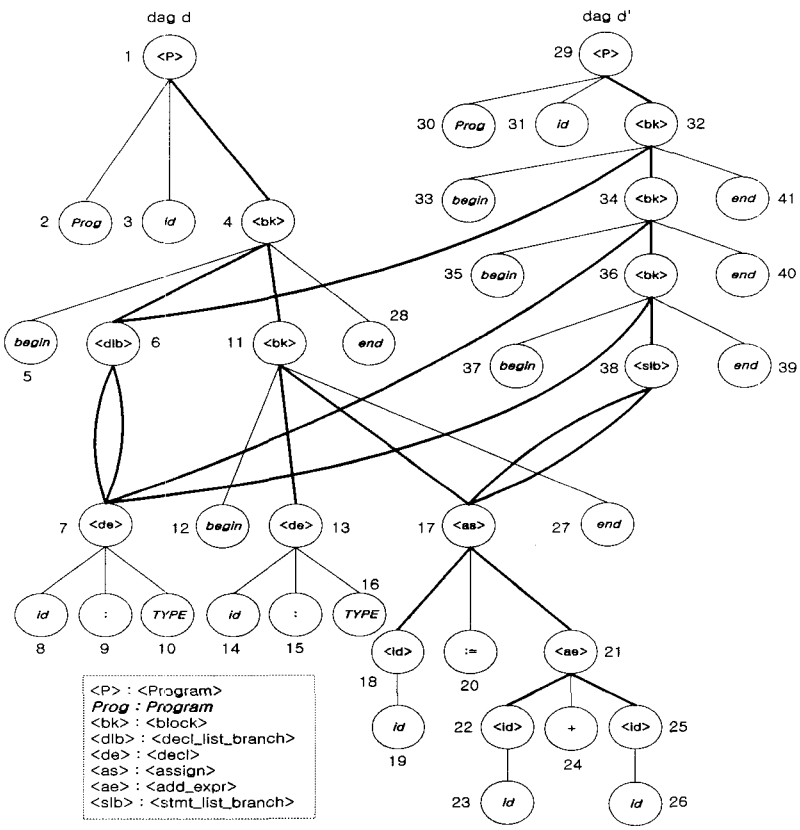
그림 10의 트리 dcopy를 구조적 속성 트리로 재구성하면 그림 11와 같다.

대그 d와 d' 그리고 트리 dcopy를 최적화된 점진적 속성평가 알고리즘에 의해 수행한 결과를 나타내면 그림 12과 같다. 채워진 원은 점진적 속성평가 알고리즘에 의해 속성을 평가하지 않고 재사용 된 노드를 나타내며, 채워지지 않은 원은 속성을 새롭게 평가한 것을 나타낸다.

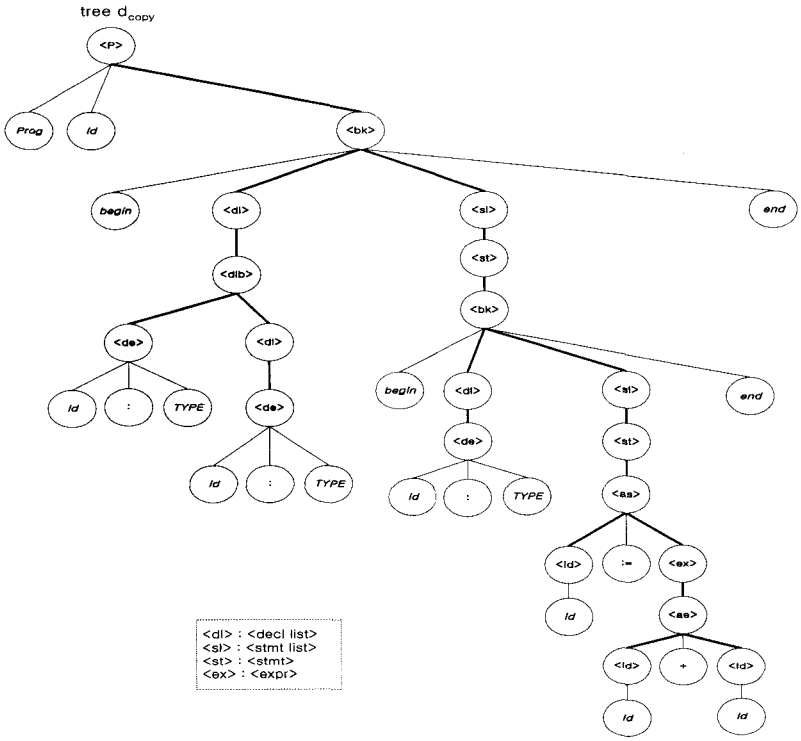
따라서, 실제 프로그램과 같이 동일한 구조를 가지는 속성 문법의 경우에서도 더 많은 서브 트리를 재사용 할 수 있음을 알 수 있다.

5. 결 론

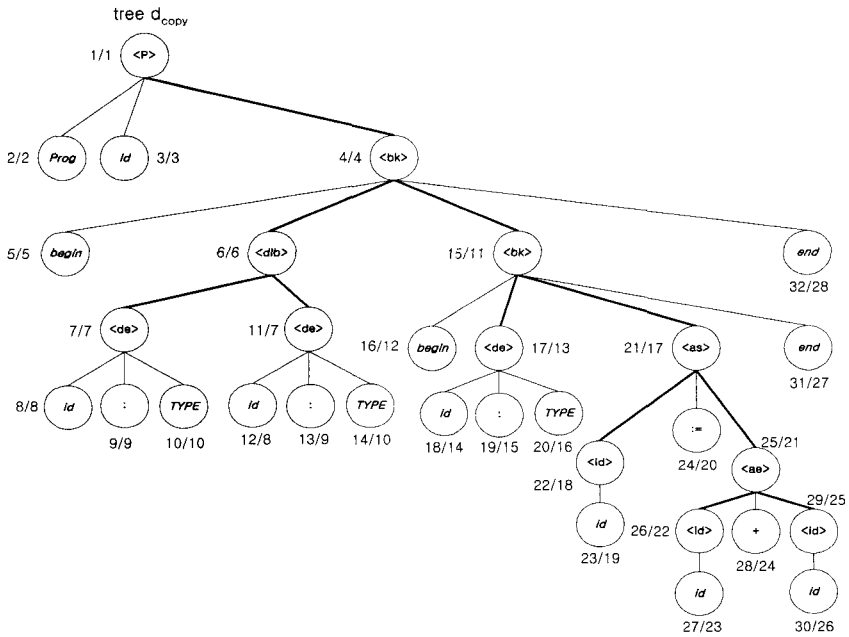
이 논문에서는 계층적 속성 문법의 점진적 평가를 효율적으로 하기 위해 점진적 속성평가 알고리즘을 이용하였으며, 점진적 속성평가 알고리즘에서는 새로운 입력 트리가 기존 입력 트리와 정확히 비교되어서 새로운 트리를 구성할 때, 기존 속성 트리의 어떤 서브 트리를 사용해야 하는가를 결정한다. 따라서 이 논문에서 제안한 속성평가 알고리즘의 동작 과정에서도 새로운 속성 트리의 생성 과정을 나타냈으며, Carle과 Pollock의 알고리즘을 분석하여 점진적 속성 평가 알고리즘으로 재구성하고, 속성 트리 dcopy의 구성요소를 새로운 속성 트리 d'copy에 적용하여 최적화된 속성 트리 d'copy의 점진적 속성평가 알고리즘



(그림 9) 입력 대그 d와 d'



(그림 10) 트리 dcopy



(그림 11) 구조적 트리 dcopy

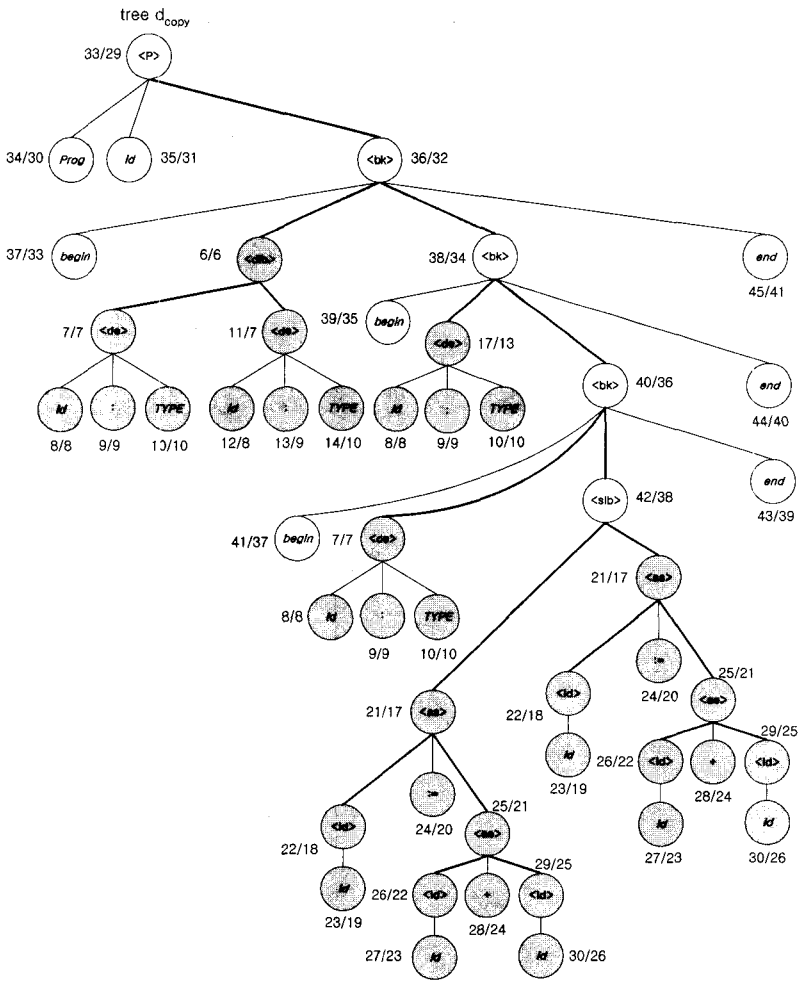
을 구성하였다.

속성 트리 dcopy의 구성요소를 새로운 속성 트리 d'copy에 적용하여 최적의 d'copy를 구성하기 위한 점진적 평가 알고리즘을 이용하여 실제적인 입력 프로그램에서 정의된 변수 형(type)이 어떻게 점진적인 속성평가를 수행하는가와 점진적 속성평가 알고리즘에 의해 재사용된 노드를 나타내었다.

점진적 속성평가 알고리즘에 대한 최적화는 제거, 단순화, 프로그램의 실행 시간과 공간을 줄이기 위한 시도로써 생성된 속성들의 재정렬을 포함하는 다양한 변환들을 수행한다. 따라서 이들 환경 내에서 변환된 속성과 복잡한 재계산을 피하고 앞서 수행된 재사용 정보를 효율적으로 대체하기 위한 다양한 점진적 기법의 최적화와 웹 데이터베이스 구축에 많은 도움이 될 것이다.

참 고 문 헌

[1] Horowitz, S. and Teitelbaum, T. "Generating Editing Environments Based On Relations and Attributes", ACM TOPLAS, Vol. 8, No. 4, pp. 577-608, Oct 1986.



(그림 12) 점진적 속성평가 알고리즘의 적용 결과

[2] Hudson, S. E. "Incremental Attribute Evaluation: A Flexible Algorithm for Lazy Update", ACM TOPLAS, Vol. 13, No. 3, pp. 315-341, July 1991.

[3] Walz, J. A. and Johnson, G. F. "Incremental evaluation for a general class of circular attribute grammars", In Proceedings of the 1988 Conference on Programming Language Design and Implementation. SIGPLAN Notices, Vol. 23, No. 7, pp. 308-320, 1988.

[4] Carle, A. and Pollock, L. "A context-based incremental evaluator for hierarchical attribute grammars", J. Program. Lang. Vol. 3, pp. 1-29, 1995a.

[5] Carle, A. "Hierarchical attribute grammars: Dialects, applications and evaluation algorithms", Ph.D. thesis, Dept. of Computer Science, Rice Univ, Houston, Tex. 1992.

[6] Carle, A. and Pollock, L. "Modular specification of incremental program transformation systems", In Proceedings of the 11th International Conference on Software Engineering. IEEE, pp. 178-187, 1989.

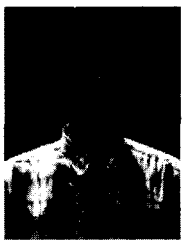
[7] Teitelbaum, T. and Chapman, R. "Higherorder attribute grammars. and editing environments", In Proceedings of the SIGPLAN '90 Symposium on Programming Language Design and Implementation. ACM, New York, pp. 197-208, 1990.

[8] Carle, A. and Pollock, L. "Matching-based incremental evaluators for hierarchical attribute grammar dialects", ACM TOPLAS. Vol. 17, No. 2, pp. 394-429, 1995b.

[9] Carle, A. and Pollock, L. "Incremental evaluation for modular attribute grammars", Tech. Rep. TR90=103, Rice Univ., Houston, Tex. Jan. 1990.

[10] Carle, A. and Pollock, L. "On the Optimality of Change Propagation for Incremental Evaluation of Hierarchical Attribute Grammars", ACM TOPLAS, Vol. 18, No. 1, pp. 16-29, 1996.

◐ 저 자 소 개 ◑



장 재 춘

1992년 관동대학교 정보처리학과 졸업(학사)
1994년 관동대학교 교육대학원 전자계산학과 졸업(석사)
2001년 관동대학교 대학원 전자계산공학과 졸업(박사)
1994~현재 영동전문대학 전자계산과 조교수
관심분야 : 병렬 컴파일러, 웹 프로그래밍, 알고리즘, 코드 최적화.
E-mail : jcjang@yeongdong.ac.kr

김 태 훈

1983년 중앙대학교 통계학과 졸업(학사)
1986년 중앙대학교 대학원 통계학과 졸업(석사)
1996년 중앙대학교 대학원 통계학과 졸업(박사)
1992~현재 영동전문대학 전자계산과 부교수
관심분야 : 경제통계, 알고리즘
E-mail : aristocrat@yeongdong.ac.kr