2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

능동형 데이터베이스 시스템에서의 규칙 종료 분석기

김홍기, 박인석 현순주
한국정보통신대학원대학교(ICU)
e-mail : hkkim@icu.ac.kr, ispark@icu.ac.kr, shyun@icu.ac.kr

A Rule Termination Analyser in Active DBMS

Hong K. Kim, In S. Park, Soon J. Hyun
Dept. of Engeering, Information and Communications University (ICU)

요 약

능동형 데이터베이스 시스템은 사용자가 정의한 rule 의 집합이 해당 event 가 발생하는 순간 능동적으로 일련의 행위를 수행하도록 정의된 시스템이다. 그러나, 이처럼 서로 긴밀한 관계를 갖는 rule 들이 능동적으로 수행되는 과정에 종료되지 않고 무한히 순환하여 수행하는 경우가 발생할 수 있다. 이처럼 무한히 순환하여 수행할 수 있는 가능성을 분석하는 것이 Termination Analysis 이다. 본 논문은 compile time 에 rule 의 termination 을 예측하는 방법에 대한 연구로, Java Language 를 rule definition language 로 사용며, composite evnet 의 경우도 지원하도록 기존의 Termination analysis 방법을 확장하였다.

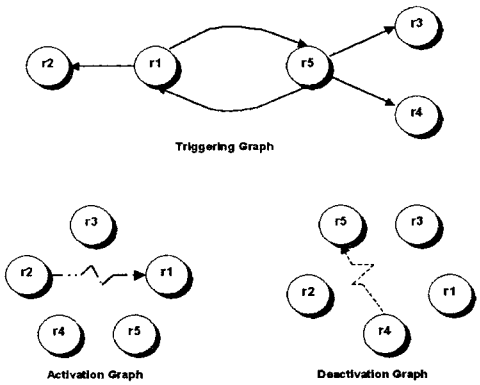
1. Introduction

방대한 용량의 데이터를 여러 사용자가 데이터의 일관성을 유지하면서 저장, 삭제, 변경의 기능을 제공하던 기존의 데이터베이스 관리 시스템(이하 DBMS)의 기능이 사용자의 요구에 의해서 수행되는 passive 한 형태였다면, 능동형 데이터베이스 관리 시스템(이하 Active DBMS)는 사용자의 요구가 없어도 일정한 조건을 만족하는 상황이 발생하면 스스로 특정 작업을 수행하는 기능을 제공한다. 이러한 기능의 제공을 위해 Active DBMS 에서는 각각의 능동적인 작업의 단위를 rule 을 통해 표현하고 있으며, 이러한 rule 들을 실행시키기 위한 동기의 부여를 event 의 발생을 통함으로써 능동적인 기능을 제공하고 있는 것이다[1]. Rule 의 동작 원리는 해당 event 가 발생하는 경우, rule 의 condition part 에 정의된 조건이 만족하게 되면 rule 의 action part 의 작업을 수행하는 것이다. Evnet 의 발생에 의해 rule 이 수행되는 과정에서 다른 rule 의 event 를 발생할 수 있다. 이러한 경우, 능동적으로 다음 단계의 rule 이 수행되며, 최악의 경우 rule 의 동작이 연속적으로 다른 event 를 발생시키므로 결과적으로 무한한 rule 의 수행이 이루어지는 경우가 발생할 수 있다. 이와 같은 문제의 발생을 예측하고 처리하기 위한 연구가 활발히 이루어지고 있는데, 일부 관계형 데이터베이스 관리 시스템(이하 RDBMS)에서는 단위 rule 의 수행에 있어서 일정 단계까지의 연속적인 수행만을 인정하도록 되어 있다. 즉, 하나의 rule 의 action part 의 수행 중에 다른 rule 의 수행할 수 있는 단계의 수를 미리 정의해 놓고 해당 단계를 넘어서 수행하는 프로세스를 제거하는 방식을 취하고 있는 것이다[2]. 또 다른 접근 방법으로는 rule 들의 능동적인 행위들을 예측하는 것으로 Static analysis 가 있다. 이것은 rule 의 집합을 등록하는 과정, 즉 compile time 에 분석하므로써 예상되는 rule 의 연속적인 실행을 프로그래머에게 알려주는 역할을 하는 것이다. 이처럼 Active DBMS 에서 rule 의 종료를 보장해주는 분석 방법은 매우 중요한 의미를 제공하는 것이다. 본 논문에서는 한국정보통신대학원대학교에서 개발된 AIMS(Active Information Management System)를 기반으로 rule 의 termination 을 보장하는 방법에 대해서 설명하고자 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로 본 연구에서 취한 Static Analysis 의 사례에 대해 간단히 알아보고, 3 장에서는 본 논문에서의 접근 방법에 대해 설명한다. 4 장에서는 본 연구의 수행을 위한 구조에 대해 간략히 설명하며, 5 장에서는 향후 연구 방향과 함께 결론을 맺도록 한다.

2. Overview : Rule Termination Analysis

Rule Termination analysis 에 관한 연구는 지금까지 많은 연구가 있었다. 여러 연구을 분류하는 방식으로는 크게 분석을 수행하는 시점에 의한 분류로 compile time 에 수행하는 Static Analysis 와 rule 의 실행 시점에 분석하는 Runtime Analysis 로 크게 나눌 수 있다. 일반적으로 Runtime Analysis 보다는 Static Analysis 에 대한 연구가 활발히 진행되고 있는데, 이것은 시스템의 수행 시점에 있어서 termination analysis 에 의한 시스템의 부담을 줄일 수 있기 때문이다.



그림[1] TG, AG and DG

Static Analysis 는 근본적으로 Triggering Graph[3]에 기반을 두고 이루어진다. Triggering Graph 란 rule 의 condition, action part 의 수행 중에 rule set 에 등록되어 있는 다른 rule 을 수행하도록 event 를 발생시키는 관계를 그래프의 형식으로 나타낸 것이다. 그러나, 이것은 서로 관련되어 있는 rule 들의 condition part 와 action part 에 대한 세밀한 고찰이 없이, event 발생에 의한 연결의 의미만을 지니고 있으므로, 정확한 의미의 termination 에 대한 증명을 나타낸다고 보기에는 부족한 점이 있다. 이러한 문제점으로 인하여 [4, 5]에서는 보다 정확한 terminaion 의 분석을 위하여 Activation Graph 의 개념을 사용하게 되었다. Activation Graph 란 rule 의 condition 과 action part 에서의 수행이 다른 rule 의 condition 값을 true 로 만드는 관계를 그래프로 나타낸 것이다. 즉, Activation Graph 란 rule 의 수행의 순서에서 반드시 실행이 되는 것을 의미하는 것이다. 또한, AG 의 반대 개념으로 Deactivation Graph 의 개념이 사용되게 되었는데, 이것은 다른 rule 의 condition part 를 false 로 만드는 것을 의미한다. 이와 같이 TG 를 기본으로 하여, AG 와 DG 의 정보를 이용, TG 에서 각각의 node(여기에서 node 는 개별적인 rule 을 의미한다.)의 연결을 유지 시키거나, 삭제하며 cycle 의 발생을 확인하여 rule 의 termination 을 증명할 수 있게 된다. 그림[1]은 TG, AG, DG 를 나타낸 것이다.

그러나, 앞에서 언급된 방식은 composite event 의 경우가 고려되지 않은 접근 방식이다. Composite event 는 단일 event, 즉 primitive event 들이 Disjunction 또는 Conjunction 과 같이 primitive event 의 조합에 의해서 새로운 하나의 event 를 이루는 것이다. 본 시스템과 같이 Object-oriented DBMS 에서 Active DBMS 의 기능을 지원하며, primitive event 와 composite event 를 지원하는 시스템은 Hipac, Ode, SAMOS[6]가 있다.

3. Our Approach

기존의 Termination analysis 에 비하여 본 연구에서 수행하는 Termination analysis 의 차이점은 다음과 같다.

3.1 Rule Definition Language

본 연구의 기반이 된 AIMS(Active Information Management System)는 Java 로 개발되어 Active DBMS 의 기능을 지원하고 있다. 이에 대한 자세한 내용은 [7]에 언급되어 있다. AIMS 를 기반으로 하여 rule 을 정의하기 위한 언어로 시스템에서 의존적인 특수한 definition language 가 아닌 Java language 를 사용하고 있다. 이와 같이 rule 의 정의 언어로 범용 프로그래밍 언어를 사용할 때, 이것을 Expressive Language[6]라 한다. 본 연구와 같이 Expressive Language 를 사용하는 Active DBMS 로는 HiPAC, Ode, SAMOS 가 있다. 이중 Hipac 의 경우 smalltalk 를, Ode 의 경우 O++, SAMOS 의 경우 C++를 사용한다. 아래의 그림[2]는 AIMS 에서 사용되는 rule 의 실제 사용 모습을 표현하고 있다.

```
DEFINE RULE RU_org_notify_candidate
EVENT EUB_member_delete
CONDITION
        DEC
            Org org = new Org(aof);
        EXP
           org.getNumberOfMember() < 20;
ACTION
        CandidateList clist = org.getCandidateList();
        ListIterator mailList
             = clist.getMailList().listIterator(0);

        while (mailList.hasNext()) {
           String mailID = (String)mailList.next();
           Runtime.getRuntime().exec("mailto "+mailID);
        }
        return true;
MODE IMMEDIATE, IMMEDIATE
AFTER RU_org_add_exmember
```

그림[2] Example rule in AIMS

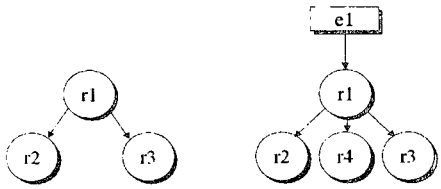
3.2 Relationship between Rules

본 연구에서도 일반적으로 다른 termination analysis 에서 취했듯이 TG 를 근반으로한다. 또한, TG 에서 cycle 이 발생하는 경우, 보다 정확한 termination 을 보장하도록, AG 와 DG 의 정보를 이용하도록 한다.

앞에서 언급했듯이, 본 시스템에서는 rule definition language 로 범용 프로그래밍 언어인 Java 를 사용한다. 이러한 접근 방법에서 가장 문제가 되는 것은 data modification 이 명확히 드러나지 않는다는 것이다. 즉, 변수의 사용을 통한 modification 일 경우, 실행 과정일 경우 데이터의 값을 알 수 있지만, 일반적인 static time 에서는 어떠한 값을 가지고 있는지 알수 없게 된다. 본 연구에서는 이렇게 원시 코드, 또는 rule 의 action part 에서 변수를 통한 data modification 이 발생하는 경우를 PG (Potential Graph)라 하여 AG 의 의미로 사용하고자 한다. 이것은 rule 의 실행과정에서 termination 을 보장하기 위해 최악의 경우인 not-termination 의 상황을 고려하기 위해 PG 를 AG 와 같은 의미로 해석하는 것이다. 이러한 접근 방법은 Expressive Language 를 사용하여 rule 정의를 하는 경우, static time 에 termination 을 분석함에 있어서, 기존의 TG, AG, DG 만을 사용하는 경우보다 안정적인 termination 을 보장할 수 있다.

3.3 Events based Termination Analysis

기존의 termination analysis 방식을 살펴보면, TG 에 참가하는 rule 들을 시작점으로 cycle 의 발생여부를 확인하고, cycle 이 발생한 경우 AG 와 DG 의 정보를 이용하여 TG 에서의 infinite 한 실행을 보다 명확하게 보장하거나 제거하도록(building or pruning) 하고 있다. 이러한 방식은 primitive event 의 경우 의미적으로 적절한 방식이지만, composite event 의 지원에 있어서 부적절한 방법이다.



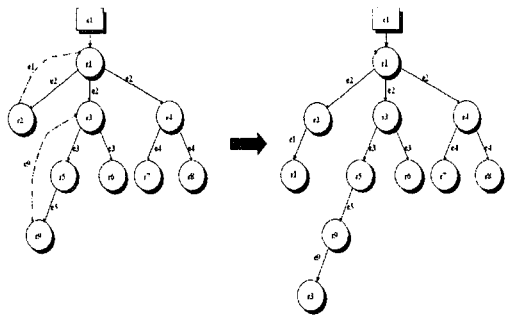
그림[3] Difference of Start point between rule and event

예를 들어, composite event 에서 sequance 의 경우를 살펴보자. Sequance e1, e2 란 primitive event e1 이 발생된 후 연속적으로 발생하는 rule 들의 실행이 종료 되기 이전에 primitive event e2 가 발생되면 sequance e1, e2 를 event 로 가지고 있는 rule 을 실행시키는 의미이다. 이럴 경우, 기존의 방법과 같이 TG 를 중심으로 하면 최초의 시작 evnet 를 고려하지 않고 rule 을 termination check 의 시작점으로 보기 때문에, composite event 의 고려점에는 문제가 발생하는 것이다. 그림[3]에서 r1 의 경우 event e1 에 의해서 수행이 되고, 내부적으로 e2 를 포함한 다른 event 를 발생시킬 때, 만일 기존의 방식을 사용한다면, sequance e1, e2 를 event 로 가지고 있는 r4 의 경우는 실행되지 않는다. 하지만, event 를 시작점으로 할 경우, 그림[3]의 우측의 TG 처럼 r4 의 정상적인 실행을 볼 수 있다. 따라서, 본 연구에서는 composite event 를 고려하기 위해 rule 을 시작점이 아닌 해당 rule set 에 참여하는 event 를 각각의 초기 시작점으로 termination analysis 를 수행하도록 하였다.

4. Architecure of Rule Analysis in AIMS

기본적으로 analysis 의 방법은 먼저 TG 의 구성으로부터 이루어진다. 만일 TG 를 통해 cycle 이 발견되지 않는다면 해당 rule 의 집합의 수행은 일정 단계 이후, 종료되는 것을 보장하는 것이다. 만일 TG 에 cycle 이 발견되면, 3 장에서 설명한 AG, DG, PG 를 구성하여 보다 세밀한 termination analysis 를 수행한다.

본 연구에서는 기존의 graph 에서 cycle 이 이루어지는 것을 통해 rule set 의 Non-termination 을 예측하던 방법을 사용하지 않고, rule 의 execution model 을 고려하여 rule 이 수행 될 순서에 의해 termination 의 여부를 분석하도록 tree 를 통해 cycle 의 발견하는 방법을 취한다. 또한, 기존의 시스템에서 cycle 을 확인하기 위해 rule 을 기반으로 했던 것에 비해, 본 연구에서는 rule set 에 참여하는 primitive event 를 기반으로하여 termination 을 보장하도록 하였다. 이와 같은 접근 방법은 primitive event 뿐만이 아니라, event 의 순서가 중요한 composite event 가 포함된 rule set 의 경우에 보다 정확한 termination 을 보장할 수 있다. 아래의 그림[4]은 TG 를 tree 형태로 표시한 것이다.



그림[4] Representation of TG as a Tree

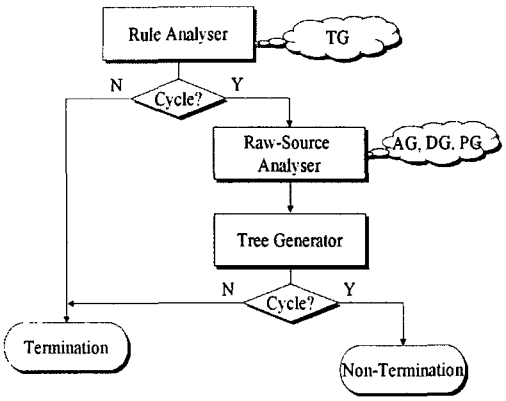
물론 위의 그림에서 cycle 의 발생을 나타내는 점선은 실제 tree 로 변환되는 과정에서 cycle 을 나타내는 새로운 node 로 표시되어, tree 의 성질을 유지한다.

아래의 그림[5]는 본 시스템의 간략한 구조도이다.

Rule Analyser 는 rule 상에 정의된 event 를 기반으로 하여 rule 의 condition, action part 에 정의된 행위들이 다른 rule 의 evnet 를 발생시키는 여부에 따라서 TG 를 생성하게 된다. 물론, 이때 composite evnet 에 대한 고려와 특정한 event 가 발생될 경우, 해당 실행 순서도 고려되어진다. 생성된 TG 를 통해서, cycle 이 발생하지 않는 경우 rule set 의 실행은 termination 이 이루어지는 것을 증명하게 되는 것이다.

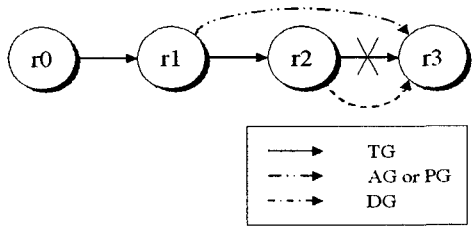
Raw-Source Analyser 는 원시 소스 코드 즉, Java Class file 에 대해 data 의 modification 활동이 일어나는지 확인하는 역할을 수행한다. Rule set 내에 정의된 rule 의 condition part 의 내용과 비교하여 condition part

의 실행 값을 true 로 변화시키면 AG 로, false 로 변화시키면 DG 로, 변수의 사용으로 modification 의 행위가 이루어지면 PG 로 분류하게 된다.



그림[5] Architecture of Rule Termination Analysis

Tree Generator 에서는 앞의 과정을 거쳐 생성된 TG, AG, DG, PG 를 이용하여 tree 를 생성하게 된다. 이러한 방식을 취한 이유는 tree 를 통하여 rule 들의 실행 순서가 결정되며, 이 결정 순서와 TG, AG, DG, PG 의 정보를 이용하여 보다 정확한 rule 의 실행을 예상할 수 있기 때문이다.



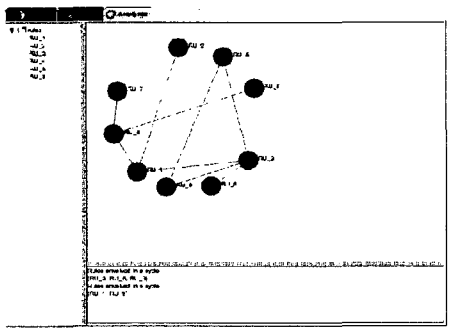
그림[6] Usage of AG, PG and DG in analysis

AG,DG 그리고 PG 의 적용 방법은 그림[6]을 참고한다. 이것은 net dffect 효과를 적용한 것이다. AG 의 내부에 <r1, r3>이 포함되어 있을 경우와 DG 의 내부에 <r2, r3>이 포함되어 있다고 가정한다면, tree 의 traverse 순서에 의해서 r1 > r2 > r3 의 순으로 rule 이 실행될 수 있음을 알수 있다. 따라서, AG,PG,DG 내에서 동일한 목적지를 가지고 있는 리스트가 있다면, 실행 순서를 우선 순위로 하여 목적지에 가까운 rule 을 포함한 그래프의 리스트, 즉 예에서 DG 의 리스트만 영향을 미칠 수 있다. 따라서, 전체의 실행 과정을 살펴볼 때, r2 는 r3 를 triggering 시킬 수 없게 된다. 만일, r3 가 r0 를 triggering 하여 cycle 이 이루어지는 경우, TG 상에서는 cycle 이 발생하여 termination 이 되지 않지만, 확장된 의미의 검색을 통해서 <r2, r3>가 TG 로 부터 제거(pruning)되므로, 결과적으로는 r2 에서 termination 이 발생함을 알 수 있는 것이다.

5. Summary

본 논문에서는 rule 의 무한한 수행을 방지하기 위해서 static time 에 rule 의 종료에 대한 가능한 예측을 할 수 있도록 termination analysis 와 동시에 개발자에게 예상되는 rule 의 수행 순서를 제공하도록 tree 를 이용하는 방식을 택하여 기존의 방법을 확장하도록 하였다.

그림[7]은 위에서 제시한 방법을 이용하여 rule 의 triggering 과정에서 발생하는 cycle 을 표시하도록 개발된 GUI tool 이다. 추가적으로 cycle 정보 뿐만이 아닌, rule 개발자에게 도움이 되도록, 전체 rule 의 실행 순서의 표현 등 개발자에게 필수적인 simulator 의 기능을 가진 tool 로의 지속적인 개발이 필요할 것이다.



그림[7] Terminiation analysis tool for AIMS

References

[1] O. Diaz, N. Paton and P. Gray. "Rule Namagement in Object Orented Databases: A Uniform Approach", Proceedings of the 17th International Conference on VLDB, 1991. 317-326

[2] Danilo Montesi, Maria Bagnato and Cristina Dallera. "Termination Analysis in Active Databases", Database Engineering and Applications, 1999. IDEAS '99. International Symposium Proceedings , 1999. 288-297

[3] A. Aiken, J. Widom and J. M. Hellerstein. "Behavior of database production rules: Termination, confluence and observable determinism". SIGMOD 21, 2 (Jun.), 1992. 59-68

[4] E. Baralis, S. Ceri and J. Widom. "Better termination analysis for active databases". Proceedings of the First International Workshop on Rules in Database Systems, (Sep.). 1993. 163-179.

[5] E. Baralis and J. Widom. "An algebraic approach to rule analysis in expert database systems". In Proceedings of the Twentieth International Conference on VLDB, 1994. 475-486

[6] Anca Vaduva, Stella Gatziu, Klaus R. Dittrich. "Investigating Termination in Active Database Systems with Expressive Rule Languages". RIDS, 1997. 149-164

[7] 이종희, 현순주. "객체관계형 데이터베이스와 능동 규칙 관리시스템의 설계와 구현". 제 6 회 학술발표대회 및 총회 논문집. 1(Jul.), 2000. 169-174