규칙 컴파일러를 위한 종료 분석 연구

강 병 극[†]·황 정 희[†]·신 예 호[†]·류 근 호[††]

요 약

능동 데이터베이스에서의 규칙(rule)은 트리거하는 사건이 감지되면 자동으로 규칙을 트리거하여 데이터베이스의 상태변화에 대응하는 조치를 자동으로 수행한다. 이러한 능동 규칙은 자신을 포함한 서로 다른 규칙을 트리거 할 수 있으므로 종료하지 못하고 연속적으로 실행될 가능성이 있다. 이와 같은 문제는 규칙의 종료 분석을 통해 차단할 수 있으며 규칙의 종료 분석은 규칙의 컴파일 시간에 수행하는 것이 가장 효과적이다. 따라서 이 논문에서는 규칙 종료분석기를 내장한 규칙 컴파일러를 설계하고 그 수행 모델 및 알고리즘을 제안한다. 아울러 제안 모델의 핵심이라 할 수 있는 규칙 종료 분석 알고리즘의 정형화를 통해 제안 모델의 완전성을 검증한다.

A Study on Termination Analysis for Rule Compiler

Bing Ji Jiang[†]·Jeong Hee Hwang[†]·Ye Ho Shin[†]·Keun Ho Ryu[††]

ABSTRACT

In the active databases, whenever an event occurs, active rules with the matching event specifications are triggered automatically, its action will be executed. Because these rules may in turn trigger other rules including themselves, the set of rules may be executing each other indefinitely. These problem can be solved by rule termination analysis, and it is efficient for the rule termination to execute in compile time of rule. In this paper we not only design rule compiler with rule termination analyzer, but also propose its execution model and algorithm. The completeness of proposed model is verified by algorithm formalization of rule termination analysis.

키워드 : 능동 데이터베이스(Active Database), 능동규칙(Active Rule), 규칙 종료분석 (Rule Termination Analysis), 규칙 실행의미(Rule Execution Semantics), 규칙 컴파일러(Rule Compiler)

1. 서 론

능동데이터베이스는 능동규칙을 통해 사용자 또는 응용프로그램의 명시적 요구 없이도 특정 연산에 대응해서 스스로 조건부 조치를 자동으로 수행할 수 있다. 특히 능동규칙은 사용자 수준에서 정의, 수정 및 삭제가 가능하여 데이터베이스 시스템의 활용 영역을 극대화 할 수 있는 특성이 있다[1].

사용자에 의해 정의된 능동규칙은 규칙 컴파일러에 의해 실행 가능한 규칙으로 컴파일되어 규칙시스템에 등록되며 동시에 규칙 베이스에 저장된다. 그러므로 정의된 규칙의 사건범주에 대응하는 사건이 사용자 연산에 의해 발생하게 되면 자동으로 규칙이 트리거되고 조건 평가 후 조치를 수행하는 일련의 규칙 실행이 개시된다.

사용자에 의해 능동규칙이 정의되고 수행되는 과정에서 가장 중요한 부분 중의 하나는 규칙 수행의 종료를 보장할 수 있는냐하는 점이다[2]. 왜냐하면 능동규칙의 특성상 능동규칙이 또 다른 능동규칙을 트리거 시키는 순환적 실행이 발생할 수 있으며 이는 비종료 상태를 유발할 수 있는 가능성을 내포하므로 규칙의 수행비용을 증가시켜 시스템 성능에 치명적 영향을 끼칠 수 있을 뿐만 아니라 바람직하지 못한 데이터베이스 상태가 발생할 수 있기 때문이다. 따라서 사용자에 의한 규칙 정의시 규칙의 종료를 보장할 수 있는 방안에 대한 연구가 능동규칙의 활용 범위를 확대시키는데 있어 중요한 문제이다.

능동 규칙의 종료를 분석하기 위해 정적분석(static analysis)방법과 동적 분석(dynamic analysis) 방법이 다양하게 연구되고 있다[3-5]. 정적 분석방법[3]은 컴파일 시간(compile time)에 규칙 실행의 잠재적인 비종료 여부를 탐색하는 방법이고, 동적 분석방법[4,5]은 미리 정의된 규칙을 이용하여 능동 데이터베이스가 동작되는 런타임(run-time)에 규칙의 동작을 조사하는 방법이다. 동적 분석 방법은 실행시간 분석에 따른 성능 저하의 문제를 수반하므로 일반적으로 능동 규칙의 일부 특성, 즉 규칙 관계만을 이용하는 정적 분석 방법을 사용한다.

이와 같은 정적 규칙 종료 분석을 수행하기 위해서는

※ 이 연구는 과학재단 특정기초연구(R01-1999-00243)의 연구비 지원으로 수행되었음.
† 준 회 원 : 충북대학교 대학원 전자계산학과
†† 종신회원 : 충북대학교 전기전자 컴퓨터공학부 교수
논문접수 : 2001년 6월 28일, 심사완료 : 2001년 8월 23일

규칙 컴파일러가 능동규칙을 컴파일 하는 과정에서 규칙의 종료 분석을 수행할 수 있는 기능을 내장해야 한다. 따라서 이 논문에서는 규칙 종료 분석이 가능한 규칙 컴파일러에 대한 연구의 일환으로 규칙 종료분석기를 내장한 규칙 컴파일러를 설계하고 그 수행 모델 및 알고리즘을 제안한다. 아울러 제안 모델의 핵심이라 할 수 있는 규칙 종료 분석 알고리즘의 정형화를 통해 제안 모델의 완전성을 검증한다.

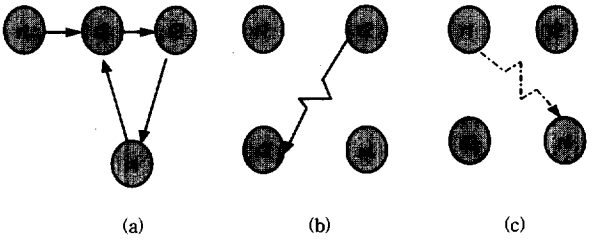
이 논문의 구성은 다음과 같다. 2장에서는 관련연구로 기존에 제시되었던 종료 분석방법에 대해 알아보고 3장에서는 종료 분석기를 포함하는 규칙 컴파일러 시스템 구조를 제시하고 4장에서는 실행의미를 반영하는 종료 분석 단계를 설명한다. 그리고 5장에서는 종료 분석기의 규칙 종료 분석 방법을 검증하기 위해 규칙 관계 정의를 이용하여 증명하고 6장에서는 기존 연구와의 차이점을 비교 분석한다. 끝으로 7장에서는 결론 및 향후 연구 방향을 기술한다.

2. 종료 분석

능동 규칙의 비종료 문제는 능동 데이터베이스를 이용하는 모든 응용 시스템에서 발생될 수 있다. 그러므로 규칙 설계자는 규칙의 동작을 미리 예측하기 위해 트리거 관계의 규칙에 대한 종료 분석을 수행해야 한다[6, 9, 10].

2.1 전개 그래프

[11]에서는 규칙간에 형성되는 트리거, 활성화, 그리고 비활성화 관계를 모두 고려하여 규칙의 처리를 정적으로 시뮬레이션하는 전개 그래프(evolution graph)를 제시하였다. 전개 그래프는 규칙간에 형성되는 각각의 트리거 그래프(Trigger Graph : TG)[5, 6], 활성화 그래프(Activation Graph : AG)[5, 18, 19], 비활성화 그래프(Deactivation Graph : DG)를 참조하여 임의의 규칙 실행으로 영향을 받는 다른 규칙과의 관계를 분석하여 형성된다. 즉, 규칙들의 추상적인 실행 상태를 예측하여 표현한 것이다. (그림 1)은 각 규칙간에 형성되는 관계 그래프로써 (a)는 트리거 관계를 나타내는 트리거 그래프, (b)는 활성화 관계를 나타내는 활성화 그래프, 그리고 (c)는 비활성화 관계를 표현하는 비활성화 그래프이다.



(a) (b) (c)

(그림 1) TG(a)와 AG(b) 그리고 DG(c)

이 방법에서는 각각의 규칙 실행 상태를 다음과 같이 네 가지로 나누어 기술하였다.

규칙 ri가 트리거되고 활성화 되는 상태를 s_i로, 트리거되고 비활성화되는 상태를 s_i^t로, 활성화되고, 트리거되지 않는 상태는 s_i^a로, 그리고 비활성화되고, 트리거 되지 않는 상태는 s_i^n으로 표기한다.

이러한 네 가지 상태로써 트리거 그래프에서 사이클을 형성하는 각 규칙의 실행으로 인해 발생 가능한 다른 규칙과의 변화 상태를 함께 기술하여 같은 실행 상태가 반복되면 규칙 실행이 종료하지 않을 가능성이 있으므로 그 규칙 집합은 종료하지 않는다는 결정을 한다.

2.2 기존 종료 분석 방법

위에서 기술된 종료 분석 방법에 대한 연구 외에도 능동 데이터베이스의 종료 문제를 해결하기 위한 많은 연구가 발표되었고 이 논문과 관련된 몇 가지 방법을 간단하게 요약하면 다음과 같다.

[10]에서는 규칙 관계를 나타내는 트리거 그래프에서 사이클이 형성되더라도 한정적으로 반복 실행되는 사이클은 종료할 수 있다는 결정을 하는 종료 분석 방법을 제안하였고 이러한 사이클은 그래프를 변형하여 사이클을 형성하지 않도록 하는 그래프 변형 알고리즘을 제시하였다. 그리고 [12]에서는 종료 분석을 위해 규칙을 작은 단위의 그룹으로 묶어 모듈화하는 방법을 제안하였다. 즉 같은 결과를 위한 규칙들을 한 계층으로 구분하여 그 모듈에 대한 종료분석을 하는 방식으로의, 각각 구분된 작은 단위의 모듈에 대한 종료 분석은 전체의 규칙 집합에 대한 종료 여부의 예측도 가능하다는 것에 의한다.

한편 [13]에서는 규칙의 조치 수행으로 인해 다른 규칙의 조건에 영향을 미치는 것을 결정하기 위한 방법으로 관계 대수를 이용한 전파(propagation) 알고리즘을 사용하였다. 전파 알고리즘은 입력으로 질의(query)와 수정(modification)을 받아 insert, delete, update의 연산자를 결과로 출력한다. 그리고 이러한 결과를 이용한 종료 분석에서는 활성화 그래프에 있는 두 규칙의 조치와 조건을 전파 알고리즘에 적용하여 결과로써 insert, update operation이 나오면 활성화 할 수 있고, delete operation 또는 아무런 operation도 출력되지 않으면 이 규칙 관계의 간선은 그래프에 포함되지 않는다는 결정을 한다.

위에 소개된 분석 방법과는 다른 접근 방식으로 [3]에서는 종료 분석을 하기 위해 능동 데이터베이스 시스템에서 규칙 실행으로 발생되는 종료에 대한 개념을 자세하게 설명하였고 복합사건의 규칙을 포함하는 종료 분석방법에 대해 언급하였다.

그러나 기존의 연구에서는 단순한 규칙언어를 사용하는 규칙에 대해서만 종료 분석의 수행 방법을 제시하였을 뿐

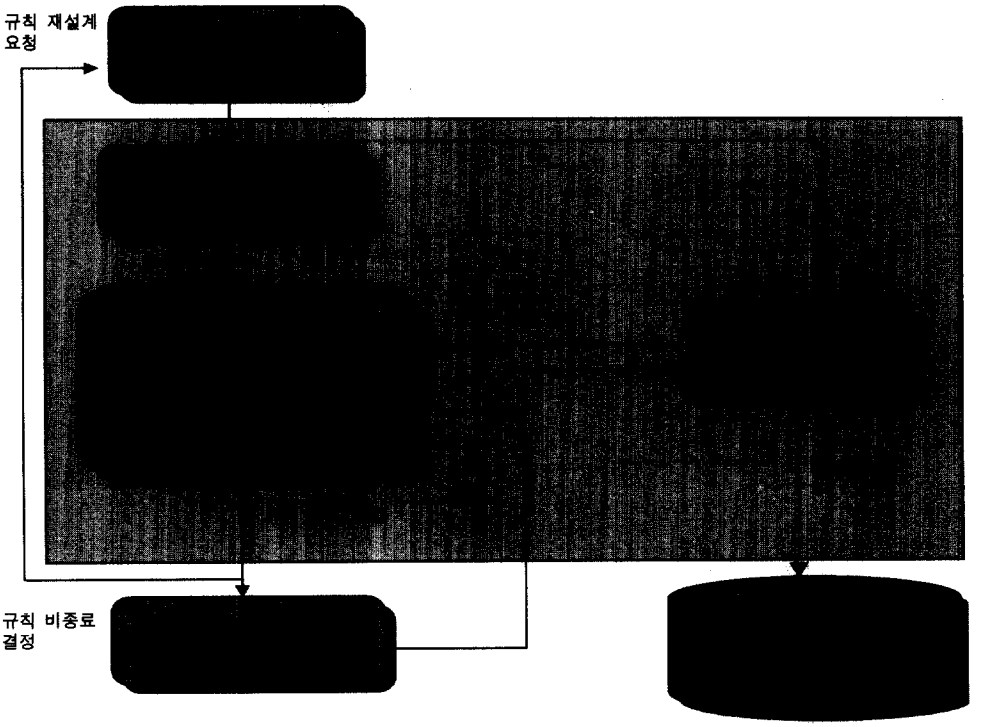
사건절에 명세되는 다양한 요소 즉, 복합사건과 규칙 실행 시점을 나타내는 before, after[7, 8]를 고려하고 있지 않다. 그러므로 이 논문에서는 이러한 기존의 종료 분석 방법의 문제점을 개선하여 규칙의 실행의미를 반영한 종료 분석 방법의 종료 분석기를 제시하고, 이를 위해 기존의 트리거 그래프를 변형한다. 아울러 규칙의 조건절을 이용하여 형성되는 비활성화 그래프를 결합함으로써 분석 과정의 복잡성을 단순화시키며, 이 분석 방법에 의한 분석기를 내장한 규칙 컴파일러 구조를 제안한다.

3. 규칙 컴파일러

이 장에서는 이 논문에서 제안하는 종료 분석기를 내장하는 규칙 컴파일러 시스템 구조를 제시하고 각 구성요소의 역할을 설명한다.

규칙 컴파일러는 규칙언어를 이용하여 정의된 능동 규칙을 실행 가능한 규칙으로 컴파일 해 주는 규칙 처리기의 일부이다. 규칙 컴파일러에는 종료분석을 마친 결과의 규칙을 규칙 베이스에 저장하고 분석 대상이 되는 기존의 규칙을 규칙 베이스로부터 가져오는 역할을 하는 분석 관리기(analysis manager), 새롭게 정의된 규칙에 대한 구문 분석과 어휘분석을 담당하는 규칙 전처리기(rule preprocessor), 정의되는 각 규칙에 대해 기존의 규칙 베이스에 저장된 규칙과의 관계를 분석하여 규칙 실행의 종료 여부를 분석해 주는 규칙 종료 분석기(rule analyzer) 그리고 분석이 완료된 규칙들에 대한 정보를 저장하고 있는 규칙 베이스(rule base) 등으로 구성되어 있다. (그림 2)는 종료 분석기를 내장한 규칙 컴파일러 시스템 구조이며 이들 각 구성요소는 규칙의 생성이나 분석 처리 및 저장을 위해 밀접한 관계를 유지한다. 다음은 각 구성요소의 역할에 대한 설명이다.

- **분석 관리자** : 규칙 정의 언어를 이용하여 정의된 능동 규칙이 분석기를 통해 종료 결정되면 규칙 베이스에 저장 및 분석을 위해 규칙 베이스의 규칙을 관리하고 분석에 관계된 규칙을 연결 및 제어한다.
- **규칙 전처리기** : 새롭게 정의된 규칙에 대해 구문분석(syntax analysis)과 어휘분석 (lexical analysis)을 담당하며 종료 분석 대상이 되는 규칙을 추출하는 과정으로, 분석 결과는 규칙 베이스에 저장될 수 있는 각 요소의 형태로 규칙이 상세 구분된다.
- **규칙 종료 분석기** : 올바르게 정의된 규칙 구문이 분석기에 입력되면 기존의 규칙 베이스에 저장되어 있던 규칙중에서 분석 대상이 되는 규칙을 제공받아 종료 분석을 수행한다. 그러므로 정의된 규칙의 수행을 효율적으로 하기 위한 컴파일과정의 최적화 단계이다.



(그림 2) 규칙 컴파일러 시스템

● **규칙 베이스** : 규칙의 정보를 저장하고 있는 기억장소로써 종료 분석 단계를 거친 규칙들에 관한 각각의 정보와 규칙간의 관계를 저장한다.

규칙 컴파일러는 올바르게 정의된 규칙 구문을 규칙 베이스에 저장할 수 있는 각 요소의 규칙 정보 형태와 구문 트리를 형성하며 규칙집합의 종료여부를 결정하기 위한 종료 분석을 수행한다. 그리고 규칙집합의 종료가 결정되면 구문 분석의 결과로써 형성된 규칙의 각 요소 즉, 규칙의 사건, 조건, 조치절에 대한 실행 계획도 함께 저장되어 관리된다.

3.1 규칙 정보와 관리

사용자에 의해 규칙이 정의되면 규칙의 종료 분석을 위해 규칙베이스에 있는 기존의 규칙중에서 정의된 규칙과 같은 테이블을 목표로 하는 규칙들에 대한 정보를 가지고 종료 분석을 하게 된다. 종료분석에 사용되는 정보에는 사건 명세, 조건 명세, 조치 명세, 트리거하는 관계를 표현하는 트리거 규칙(TR : Triggered Rule), 비활성화 관계를 나타내는 비활성화 규칙(DR : Deactivated Rule), 활성화 관계를 나타내는 활성화 규칙(AR : Activated Rule), 규칙 적용에 관계된 목표 테이블(TT : Target Table), 규칙 실행 시점을 표현하는 Before(BE)와 After(AF), 기본사건(PE : Primitive event)과 복합사건(CE : Compositive event)를 구별하는 사건의 형태(Event Type)가 있다. 또한 규칙의 상태(rule status)를 나타내는 Enable(EN)와 Disable(DIS)은 규칙의 실행 가능성 여부를 제공해 주므로 각 규칙에 대한 상태를 규칙에 대한 정보로 저장하고 있어야 한다. 이렇게 각 규칙에 대한 규칙 정보를 규칙 베이스에 저장해야 새로운 규칙과의 종료분석이 가능하다. (그림 3)은 규칙 베이스에 저장되는 규칙 정보의 내용이다

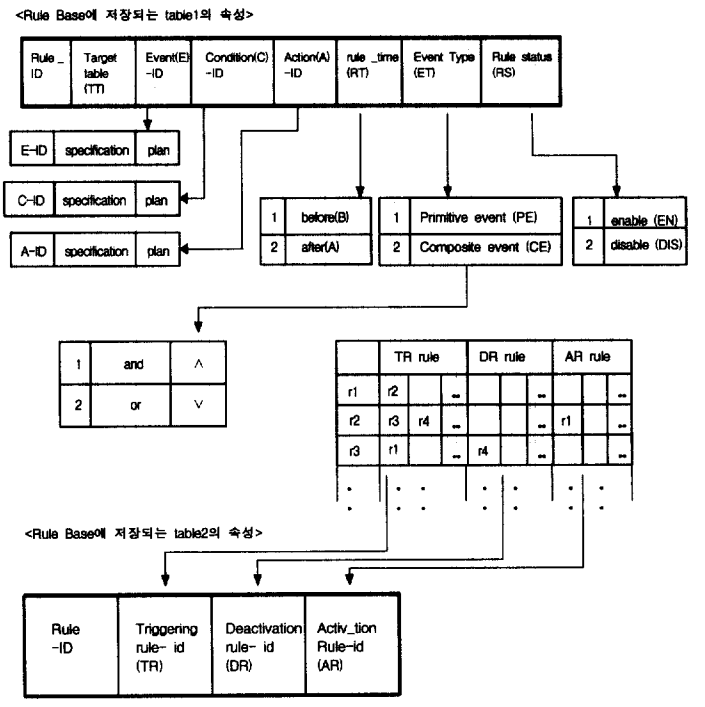
규칙 베이스에 저장되어 있는 규칙이 수정 및 삭제될 때에도, 새로운 규칙이 정의되어 규칙 베이스에 저장되기 전에 종료 분석을 통한 결과의 규칙이 저장되듯이 재분석이 이루어져야 규칙의 동작을 정확하게 예측할 수 있고 무한하게 규칙이 트리거되어 실행되는 것을 피할 수 있다.

3.2 규칙의 어휘 분석 및 구문 분석

규칙 컴파일 과정의 첫 번째 단계인 어휘분석(lexical analysis)은 어휘분석기(lexical analyzer) 또는 스캐너(scanner)에 의해 원시 프로그램을 하나의 긴 문자열로 보고 차례대로 문자를 검조(sacanning)하여 문법적으로 의미있는 최소의 단위로 분할해 내는 것을 말하는데 이 문법적인 단위(syntactic entity)를 토큰(token)이라 하고, 토큰 번호와 토큰 값의 순서쌍으로 구성되는 토큰 정보는 그의 속성과 함께 심벌 테이블(symbol table)에 보관된다[14].

다음 규칙 정의의 예는 토큰 번호와 토큰 값의 구성, 그리고 심벌 테이블에 저장되어 있는 형태를 설명하기 위한 것이다.

【예 3.1】 CREATE TRIGGER r1



(그림 3) 규칙 베이스에 저장되는 규칙 정보

AFTER UPDATE OF **A_attr1** ON **A_table**
WHEN **A_attr1** > 1000
THEN UPDATE **A_attr1** = **A_attr2** + 10

위 규칙 정의에서 조치절에 대해 설명하면, 토큰 번호로 명칭은 1, 상수는 2 그리고 특수 형태의 키워드와 연산자는 자신을 나타내는 고유의 정수를 갖는다고 가정하고, 토큰 값은 일반 형태일 때는 자신의 값이 되며 특수 형태는 토큰 값을 갖지 않지만 편의상 0으로 표시하도록 할 때 입력 스트림

THEN UPDATE **A_attr1** = **A_attr2** + 10 ;
은 (토큰 번호, 토큰 값)의 쌍
(22, 0) (24, 0) (1, **A_attr1**) (39, 0) (1, **A_attr2**) (33, 0)
(2, 10)(7, 0)
으로 변환되어 심벌테이블에 저장된다.

구문분석 단계는 어휘 분석 단계의 출력인 일련의 토큰을 입력으로 하여 주어진 입력이 올바른 규칙 정의인가를 검사하고 다음 단계에서 필요한 정보를 구성하는 과정이다.

구문 분석기는 간단히 파서(parser)라고 부르며, 스캐너에 의해 생성된 토큰을 입력으로 받아 문법적인 검사를 한 후에 다음 단계인 중간 코드 생성기가 효과적으로 중간 코드를 생성할 수 있는 형태의 정보를 구성하여 출력한다.

구문 분석을 수행하는 구문 분석기는 입력 스트링을 받아 그것이 올바로 정의된 문법의 문장이라면 구문 분석 정보의 유도 과정을 나타내는 트리 형태의 구문 트리(abstract syntax tree)로 결과를 출력하고, 정의된 문법의 문장이 아니면 오류 메시지를 출력한다.

구문 분석의 결과로써 출력되는 구문 트리는 입력된 문장의 구조를 나타낸다. 구문 트리는 파스 트리의 변형된 형태로 코드 생성 단계에서 꼭 필요한 의미 정보(semantic information)만을 갖는 트리이다.

앞의 예 3.1에서 표현한 규칙정의를 (그림 4)에서 구문 트리를 이용하여 구문정보를 효율적으로 나타내고 있다.

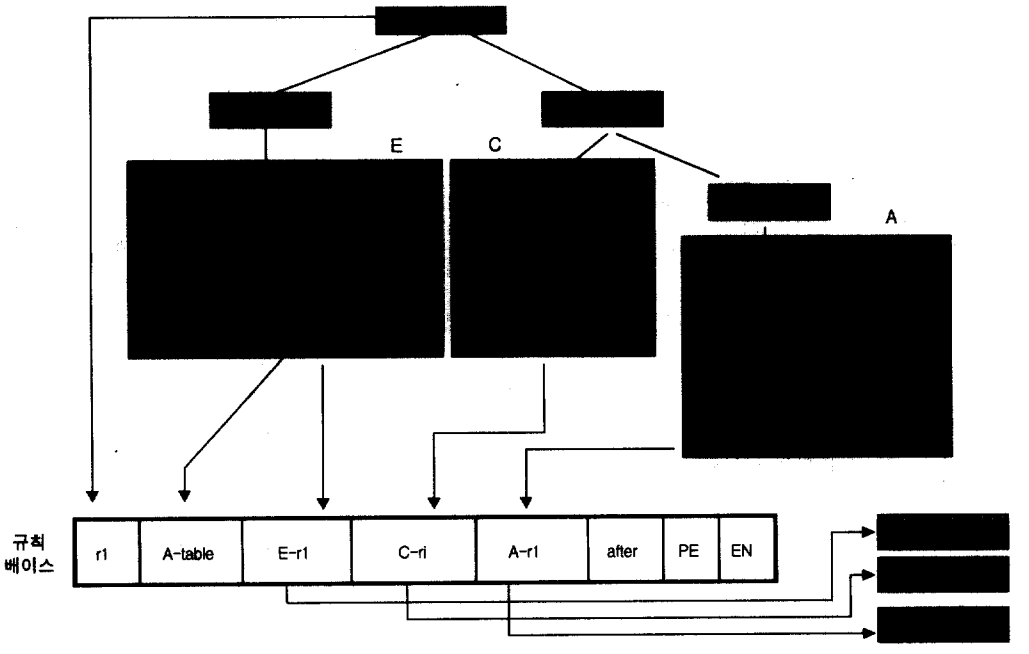
정의된 규칙을 입력으로 받아 어휘 분석과 구문 분석을 거친 올바른 구문은 규칙 베이스에 저장될 수 있는 형태로 나뉘어지고, 규칙의 종료분석을 위해 분석기에 입력되어 규칙 구문의 최적화 단계인 종료 분석을 수행한 결과에 의해 규칙의 실행 계획이 형성된다. 그리고 생성된 실행 계획은 규칙 베이스에 실행 될 수 있는 최적화된 형태의 사건절, 조건절, 조치절의 계획으로 나뉘어 저장된다.

규칙정의의 최적화(optimization)란 입력된 규칙 집합이 실제로 수행되었을 때 무한하게 반복적으로 실행되지 않고 종료할 수 있는가를 분석하는 것으로 정의된 규칙의 올바른 수행 여부를 예측하고 수정하는 과정이다. 따라서 이러한 종료 분석 과정을 거친 결과의 규칙집합은 종료한다는 보장을 할 수 있다.

4. 규칙 종료 분석 단계

이 장에서는 앞에서 제시된 규칙 컴파일러의 핵심 구성요소인 규칙 종료 분석기의 분석단계를 상세히 설명한다. 이 종료 분석 단계는 그래프 형성과 사이클 분석의 2단계로 구성된다.

규칙의 사건절에는 기본사건외에 and, or의 논리연산자를



(그림 4) 구문 트리와 규칙 베이스와의 연관성

포함하는 복합사건과 규칙의 실행시점을 나타내는before, after를 사용할 수 있고, 이것은 규칙의 실행 관계를 바탕으로 하는 종료 분석에 중요한 요소로 작용한다.

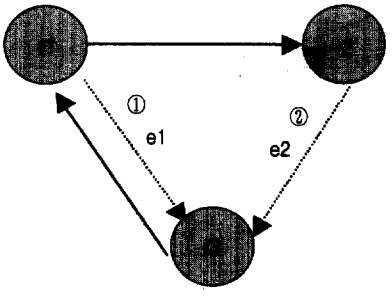
4.1 그래프 생성 단계

기존의 트리거 그래프[5, 6]에서 실선은 기본 사건의 트리거 관계를 나타낸다. 그러나 복합사건은 기본사건의 조합으로 형성되기 때문에 복합사건 규칙의 트리거는 기본사건과 구별하기 위해 점선을 나타내어 트리거관계를 표시한다. 이것은 복합사건을 트리거하는 잠정적인 트리거 관계를 표현하는 것이다. 그러므로 복합사건의 규칙을 트리거하는 규칙 관계에서 복합사건의 조합이 and 연산자로 이루어진 경우에는 트리거 관계의 모든 잠정적 기본사건이 발생해야 규칙이 트리거 될 수 있고, or인 경우에는 적어도 하나의 잠정적 기본사건이 발생해야 복합사건의 규칙이 트리거되므로 이와 같은 트리거 조건의 만족 여부가 종료분석에서 고려되어야 한다.

그리고 복합사건의 규칙에 대한 트리거관계를 나타낼 때는 조치절의 실행으로 트리거되는 규칙의 사건을 간선에 표시하여, 어떠한 사건으로 규칙이 트리거 되는지 원인이 되는 사건을 구분하고, 사건의 발생순서를 중요시하는 복합사건일 경우에는 사건의 발생순서를 참조한다. 이것은 복합사건의 트리거 여부를 정확하게 결정할 수 있도록 한다. (그림 5)는 그래프에서 복합사건의 트리거 관계를 표현한 예이고 다음 (1), (2)은 이를 이용하여 복합사건의 트리거 조건을 표현한 것이다.

(1) 복합사건의 규칙 r3이 and 연산자에 의한 경우
r1(e1) and r2(e2) = r3(e1 ∧ e2)

(2) 복합사건의 규칙 r3이 or 연산자에 의한 경우
r1(e1) or r2(e2) = r3(e1 ∨ e2)



(그림 5) 잠정적 사건 e1과 e2에 의한 복합사건의 트리거 표현

규칙의 실행 시점을 의미하는 before, after를 고려한 종료 분석에서는 기존의 트리거 그래프에서 노드에 규칙 이름만을 나타내었던 것과는 다르게 규칙 이름에 before(b), after(a)를 구분하여 그래프에 표시한다. 이것은 규칙의 실행시점을 명시한 before 또는 after에 의해서 규칙의 실행 순서가 달라지므로 종료 분석 과정에서 고려되어야 하기 때문이다.

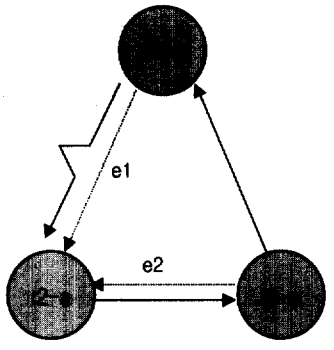
규칙이 실행되는 시점이 명시된 규칙의 실행의미를 반영할 경우 실제로 규칙이 실행되는 순서를 알아보기 위한 예로써, 규칙{r1-b,r2-a,r3-b}의 우선순위는 규칙의 순서에 의하고, 규칙이 트리거되어 실행된다고 할 때 실제적으로 규칙이 실행되어 완료되는 순서는 규칙 실행 시점을 나타내는 before와 after의 사용으로 인해 예측과는 다른 순서로 실행될 수 있다. 즉, r1의 실행에 이어 r2가 트리거 되고 조치가 실행될 때 before 규칙의 r3가 트리거되어 실행된 후 r2의 조치실행이 완료된다.

그러므로 규칙 실행 순서에 따라 생성되는 사건과 생성되는 순서가 달라질 수 있고 그로 인해 트리거 되는 규칙과 규칙의 실행여부 결정에 많은 영향을 준다. 즉, 규칙 실행 시점을 나타내는 before, after는 규칙의 우선순위와 더불어 종료분석 결정에 중요한 요소로 작용한다.

종료 분석은 규칙의 실행 가능성을 표현한 트리거 그래프를 이용한다. 그러나 임의의 규칙 실행으로 인해 트리거된 규칙은 조건을 만족하지 않으면 조치까지의 완전한 실행은 불가능하다. 그러므로 규칙간에 형성되는 비활성화 관계[15]는 규칙 종료 분석에 정확성을 제공하는 요소가 된다. 따라서 이 논문에서는 기존의 트리거 그래프에 비활성화 관계를 나타내는 비활성화 그래프를 결합하여 종료 분석한다. 비활성화 관계의 규칙은 다음의 조건에 의해 그래프에 표현된다.

- 비활성화 그래프(DG) : 임의의 두 규칙 ri, rj (i≠j)에 대해 ri의 조치가 rj의 조건을 거짓으로 변경시킬 때 규칙 ri와 rj를 간선으로 표현하여 DG 그래프에 포함시킨다[11].

(그림 6)은 복합사건의 규칙과 before, after 규칙을 포함하여 TG와 DG 결합그래프를 나타낸 예이다.



(그림 6) 복합사건 및 before, after 규칙을 포함한 트리거 그래프(TG : 실선 및 점선)와 비활성화 그래프(DG : 각진 실선)

규칙의 조건절에 기준을 두는 비활성화 관계는 트리거되는 시점에 따라 실행여부가 달라질 수 있다. 즉, 규칙이 비

활성화 되었다고 해서 무조건 규칙이 실행되지 않는 것은 아니며 트리거 되는 시점과의 간격사이에 또 다른 임의의 규칙에 의해 활성화상태로 변경될 수 있다는 것을 고려해야 정확한 종료 분석이 가능하다. 그러므로 사이클의 실행여부를 결정하는 규칙의 비활성화 관계는 규칙 베이스에 저장되어 있는 활성화 관계를 참조하여 결정한다.

4.2 사이클 분석 단계

규칙종료분석은 규칙관계를 나타내는 그래프에서 규칙이 종료하지 못하는 원인이 되는 사이클을 분석하는 것이다. 그러므로 (그림 7)의 알고리즘에 의해 그래프에 존재하는 사이클을 제거하여 종료분석을 수행한다. 1차 분석에서는 복합사건의 실행여부를 고려하여 사이클을 제거하고 2차 분석에서는 사이클을 형성하는 규칙간에 서로 다른 규칙을 비활성화 시키는 관계가 존재하는지를 고려하고 사이클을 제거하여 종료 분석한다.

사이클 분석 과정을 효율적으로 기술하기 위하여 규칙 ri와 rj가 트리거 관계일 때는 TR(ri, rj)로 표기하고, 비활성화 관계일 때는 DR(ri, rj)로 표기한다.

```
// 트리거 관계 TR(ri, rj), TR(rk, rj) 일 때
  if rj is composite event
     if composed by 'and'
       if(P(rj) > P(ri)) or (Pr(rj) > Pr(rk)) or (Exe-time(rj) = before)
            delete cycle including rj
       else
     else if composed by 'or'
            if (Pr(rj) > Pr(ri) and Pr(rj) > Pr(rk))
                 delete cycle including rj
            else
     else //1차 분석 종료

  if cycle exist
     if selectdeact(rn) = false then // 사이클에 있는 규칙을 반복
                          수행 (rn ∈ CR)
        determine termination //
     else determine non-termination // 2차 분석 종료

  // 사이클에서 비활성화 관계 탐색 알고리즘
   selectdeact(rn) : boolean
     { result = true
       foreach rn ∈ CR
         { if rn.cn = false // 규칙 rn의 조건 cn
              then return = false
           else result = selectdeact(rn) // 조건값이 false가 아니면
                          사이클에 있는
                return = result ∧ return  규칙에 대해 반복 수행 //
         }
     }
```

(그림 7) 사이클 제거 알고리즘

위에 제시된 알고리즘에 의해 (그림 6)의 그래프를 이용하여 종료분석을 하기 위한 가정으로, 규칙 r2-b는 or 연산자에 의한 복합사건이고 규칙의 우선순위가 r1-a, r2-b, r3-a의 순서로 실행된다고 할 때, 규칙의 실행은 우선순위에 의해 r1-a가 먼저 실행되고 그의 조치 실행전에 r2-b가 실행되고 r1-a의 조치실행 완료와 r3-a가 실행되는 과정을 통해 사이클이 형성된다. 그러나 사이클 형성의 규칙간에 비활성화 관계의 간선(r1-a, r2-b)이 존재하며, 이것은 r2-b의 조건값이 거짓이 되어 더 이상 실행이 불가능한 규칙관계를 의미하므로 이러한 규칙 관계를 포함하고 있는 사이클은 제거할 수 있고 종료를 보장한다.

규칙 종료 분석방법에서의 효율성이란 종료를 결정하는 과정의 간결성과 분석결과의 정확성을 의미한다. 이 논문에서 제시하는 트리거 그래프와 비활성화 그래프를 이용한 종료 분석 분석 절차는 다음과 같다.

- 1차 분석 : 트리거 그래프를 구성하는 기초 분석과정으로써 동일한 목표 테이블에 대해 정의된 규칙들의 트리거 관계를 그래프로 표현하여 사이클의 존재여부와 실행여부를 점검하여 종료여부를 결정한다. 즉, 1차 분석은 규칙의 조치수행으로 자신을 포함한 임의의 규칙을 트리거하는 규칙 관계에 대해 분석하는 것이다. 분석 결과로써 사이클의 제거조건에 의해 제거되는 사이클을 제외한, 사이클 관계의 규칙 들에 대해서는 비활성화 관계를 이용하는 2차 분석을 통해 규칙 집합의 종료 여부를 결정한다.
- 2차 분석 : 비활성화 그래프는 규칙의 조치실행으로 자신을 제외한 임의의 규칙의 조건 값을 참에서 거짓으로 변화시킬 때 생성되는 규칙 관계의 간선을 그래프에 포함시켜 형성한다. 1차 분석의 트리거 그래프에서 사이클을 형성하는 규칙들 중에서 비활성화 관계에 의해 제거될 수 있는 사이클 즉, 거짓(False) 사이클의 규칙들은 제거된다. 단, 비활성화 관계의 규칙은 다시 활성화되지 않는 관계이어야 한다.

1차 분석은 규칙간의 조치절과 사건절의 관계를 분석하는 것이며 2차 분석은 1차 분석 결과를 가지고 조치절과 조건절을 상세하게 분석하는 것이다.

1, 2차의 분석과정을 마친 결과로써 규칙들의 관계를 표현한 그래프에 사이클이 존재하지 않는다면 규칙집합은 종료를 보장한다고 결정할 수 있다. (그림 8)은 규칙 종료 분석을 수행하는 알고리즘이다.

복합사건과 before, after 규칙의 종료 분석 방법을 설명하였다. 그러나 가장 최종적인 분석의 결정은 비활성화 그래프를 이용한 2차 분석에서 이루어진다.

임의의 규칙 실행으로 인해 비활성화된 규칙은 트리거가 되더라도 조건을 만족하지 않으면 조치까지의 완전한 규칙 실행은 불가능하다. 그러므로 규칙의 조건에 기준을 두는 비

활성화 관계는 사건에 기준을 두는 트리거 관계보다 더 상세한 규칙 정보를 제공한다. 단, 비활성화 된 규칙이 실행되기 전에 다시 활성화되지 않는다는 규칙 의미를 포함해야 한다. 따라서 규칙 베이스에 규칙 관계를 저장할 때 정확한 종료분석의 결과를 위해 활성화 관계(AR)를 저장한다.

```
• Triggering Step
  fetch rules related in rulebase
  search TR(ri[ai] = rj[ej])
  generation trigger graph
  if non-execution rule ∈ cycle CR
    delete CR
  if not exist cycle then
    stop
  else execute deactivation step.

• Deactivation Step
  search DR(ri[ai] = rj[cj])
  generation deactivation graph
  if deactivation relation ⊂ cycle CR
    delete false cycle CR
  if not exist cycle then
    stop
  else request rule redesign to user
```

(그림 8) 규칙 종료 분석 수행 알고리즘

5. 규칙 관계 증명 및 분석

앞에서 규칙의 실행의미를 고려하고 규칙간의 트리거 관계와 비활성화 관계의 결합 그래프를 이용하는 종료 분석하는 방법을 제시하였다. 이 장에서는 규칙 종료분석 결과의 정확성을 입증하기 위해 앞에서 제시한 종료 분석 방법에 대한 개념을 정의하여 증명한다.

5.1 규칙 관계

규칙 종료는 규칙의 조치 실행으로 발생되는 사건에 의해 트리거되는 모든 규칙의 실행이 반복적으로 지속되지 않고 트리거된 모든 규칙의 실행이 완료된 상태를 말한다. 그러므로 이러한 규칙집합은 종료한다는 보장을 할 수 있다.

규칙 집합 R = {r1, r2, r3, ···, rn}이라고 하면 각 규칙의 사건절은 r.E, 조건절은 r.C , 조치절은 r.A로 표현한다. 규칙 관계의 기본 정의를 위해 규칙 ri의 실행은 Out(ri)로 표기하며, 규칙 rj을 트리거하는 규칙은 TR-by(rj)로 표기한다. 그리고 Trans-into는 규칙의 조건 값을 변화시키는 규칙을 말한다.

[정의 1] 규칙 ri의 조치실행으로 인해 발생하는 사건이 rj를 트리거하는 관계는 다음과 같이 정의한다[9].

$$TR(ri, rj) = \{ri, rj \in R | Out(ri) \cap TR\text{-}by(rj) \neq \phi\}$$

[정의 2] ri의 조치 실행으로 rj의 조건 값을 거짓으로 변화시키면 규칙 ri, rj 는 비활성화 관계(DR)이다[9].

$$DR(ri, rj) = \{ri, rj \in R | Out(ri) \cap Trans\text{-}into(r_j.Ce = False) \neq \phi\}$$

규칙 r의 조건 평가의 값(Condition Evaluation)은 r.Ce으로 표기하며 값의 범위는{True, False}이다.

활성화 관계는 정의 2의 비활성화 관계와 반대되는 규칙 관계 의미를 갖으며 정의 3에서 정의한다.

[정의 3] ri의 조치 실행으로 rj의 조건 값을 참으로 변화시키면 규칙 ri, rj 는 활성화 관계(AR)이다.

$$AR(ri, rj) = \{ri, rj \in R | Out(ri) \cap Trans\text{-}into(r_j.Ce = True) \neq \phi\}$$

이러한 규칙 관계의 기본 정의를 바탕으로 다음에서는 종료 분석을 위한 사이클 관계를 정의 4로 표현한다.

[정의 4] 정의 1의 트리거 관계에 의해 규칙의 실행 순서가 Rs<r1, r2, ···, r1>이면 사이클(CR)이다.

즉, 이러한 규칙 순서의 실행은 {r1.A→ r2.E, r2.A→ r3.E, r3.A→ r4.E, ···, rn.A→ r1.E} 임을 말한다. 여기서 트리거 관계는 →으로 표현한 것이다. rn.A→ r1.E 는 규칙 rn의 조치로 인해 r1을 다시 트리거 하는 것을 말하므로 반복적으로 순환하는 사이클 형태이다. 사이클 CR을 정의 1의 트리거 관계를 이용하여 다음과 같이 나타낼 수 있다.

$$Rs <r1, r2, \cdots r_n, r_1> = CR(r_1, r_2, \cdots r_n) \text{이고}$$
$$CR(r_1, r_2, \cdots r_n) = \{r1, r2, \cdots rn \in R \mid (Out(r1) \cap TR\text{-}by(r2) \neq \phi) \text{ and } Out(r2 \cap TR\text{-}by(r3) \neq \phi) \text{ and } \cdots \text{and } (Out(rn) \cap TR\text{-}by(r1) \neq \phi)\}$$

이 논문에서 제안하는 복합사건의 규칙을 종료 분석에 고려하기 위해 복합사건의 규칙은 정의 5와 같다.

[정의 5] 복합사건의 규칙 r_j는 규칙 r_i, r_k에 의해 트리거되는 and, or 연산자에 의한 규칙이라고 할 때 트리거 되는 경우를 기술한 것이다[9].

$$TR(r_i, r_k \rightarrow r_{j(and)}) = \{r_i, r_j, r_k \in R \mid E(Out(r_i) \cap TR\text{-}by(r_j) \neq \phi) \cup E(Out(r_k) \cap TR\text{-}by(r_j) \neq \phi) = r_j.E\}$$
$$TR(r_i, r_k \rightarrow r_{j(or)}) = \{r_i, r_j, r_k \in R \mid E(Out(r_i) \cap TR\text{-}by(r_j) \neq \phi) \cup E(Out(r_k) \cap TR\text{-}by(r_j) \neq \phi) \subseteq r_j.E\}$$

$E(Out(r_i) \cap TR\text{-}by(r_j) \neq \phi)$은 $TR(r_i,r_j)$관계에서 발생하는 사건을 의미한다.

and 연산자에 의한 복합사건의 규칙 r_j은 잠정적으로 트리거하는 규칙관계 TR(r_i, r_j), TR(r_k, r_j) 관계에서 발생하는 사건이 규칙의 사건절에 명세되어 있는 모든 사건과 일치해야 규칙 r_j가 트리거 가능하다는 것을 의미하고, or 연산자에 의한 복합사건일 때는 규칙 r_j에 명세되어 있는 사건 중 어느 하나만 TR(r_i, r_j), TR(r_k, r_j) 관계에서 발생하면 트리거 될 수 있다는 것을 말한다.

5.2 사이클 분석

지금까지는 종료 분석을 위해 규칙 관계의 타당성을 증명하기 위하여 정의하였고 이 절에서는 종료 분석에 직접적으로 적용되는 사이클 분석의 정의와 정리 증명으로서 규칙 관계 증명을 정리한다.

[정의 6] 트리거 관계의 사이클을 형성한다 할지라도, 사이클 형성의 규칙집합에서 실행이 불가능한 규칙을 포함하고 있는 사이클은 거짓 사이클(False Cycle)이다[9]. 이러한 사이클의 조건을 식으로 표현하면 다음과 같다.

$$\text{False Cycle}(CR) = \{r_i, r_j, r_k \in CR \mid (r_i.Ce = \text{False}) \text{ or } (r_j.Ce = \text{False}) \text{ or } (r_k.Ce = \text{False})\}$$

규칙 집합에 속하는 규칙 중 어느 하나라도 조건 값이 거짓이 되는 규칙을 포함하는 사이클, 즉 규칙의 실행에 있어 조건 값이 거짓이 되는 규칙과 비활성화 관계에 의해 조건 값이 거짓으로 변화되는 규칙(일정한 횟수의 반복 수행후 비활성화 되는 규칙 포함)을 포함하는 사이클은 연속적인 순환적 실행이 불가능하므로 참(True) 사이클이 아니다.

다음은 사이클을 형성하는 규칙 집합의 사이클 관계에 대해 정의한다. 규칙간에는 여러 개의 사이클 즉, CR = {CR_1, CR_2 , , ⋯, CR_m}이 존재할 수 있으며 각각의 사이클 CR_i = {r1, r2, r3, ⋯, rn}으로 표현되는 규칙들로 구성된다.

[정의 7] 사이클 CR1, CR2가 공통된 간선, 즉 CR2 ∩ CR1 = TR(ri, rj) 을 적어도 하나 포함하고, CR2 ⊂ CR1 관계가 성립할 때, 사이클 CR2이 제거되면 사이클 CR1도 제거된다. 이것을 식으로 나타내면 다음과 같다.

$CR_i \subset CR_j = \{CR_i, CR_j \in CR, r_k, r_t \in CR_i, CR_j \mid CR_i \cap CR_j = TR(r_k, r_t)\}$ 일 때,

$TR(r_k, r_t)$관계가 성립하지 않으면 CR_i이 제거되므로 CR_i을 포함하는 CR_j 도 제거된다.

사이클을 형성하는 규칙 집합(R)에 있는 임의의 규칙 ri, rj, rk ∈ R에 대해 TR(ri, rj), TR(ri, rk) 관계가 존재 할 때, 1차 분석에서 사이클 제거 조건의 정의는 다음과 같다.

[정의 8] and 연산자에 의한 복합사건의 규칙 rj 가 규칙 ri, rk의 실행으로 트리거될 때 규칙 ri, rk의 실행으로 발생하는 잠정적 트리거 사건의 어느 하나라도 rj보다 더 나중에 발생하는 사건이 있으면 rj의 실행이 불가능하므로 이러한 규칙을 포함하는 사이클은 제거한다. 즉, 규칙 rj가 after 규칙의 경우에는 우선순위가 {Pr(rj) > Pr(ri) or Pr(rj) > Pr(rk)}일 때 실행 불가능하고, rj가 before 규칙인 경우에는, 발생된 사건의 처리보다 더 먼저 규칙을 실행해야 하는 before 규칙의 실행의미에 의해 실행 불가능하다. 이러한 제거 조건을 정의 5의 복합사건 정의를 이용하여 식으로 기술하면 다음과 같다.

$$TR(r_i, r_k \rightarrow r_{j(\text{and})}) = \{r_i, r_j, r_k \in R \mid E(\text{Out}(r_i) \cap \text{TR-by}(r_j) \neq \phi) \cup E(\text{Out}(r_k) \cap \text{TR-by}(r_j) \neq \phi) = r_j, E\}$$

일 때, 다음의 조건을 갖는 규칙 r_j은 실행 불가능하므로 이러한 규칙을 포함하고 있는 사이클 CR은 제거한다.

$$\text{Rid}(r_j \in CR) = \{r_i, r_j, r_k \in R \mid (Pr(r_j) > Pr(r_i)) \text{ or } (Pr(r_j) > Pr(r_k)) \text{ or } (\text{R-TIME}(r_j) = \text{before})\}$$

여기서, R-TIME(r_j)은 규칙 r_j 의 실행 실행 시점{before, after}를 의미한다.

[정의 9] or 연산자에 의한 복합사건의 규칙 rj 가 규칙 ri, rk에 의해 트리거될 때 규칙 ri, rk 중에 어떤 규칙도 rj보다 먼저 실행되어 잠정적으로 트리거하는 사건을 발생하지 않는 경우에는 rj의 실행이 불가능하므로 이러한 규칙을 포함하고 있는 사이클은 제거한다. 즉, 우선순위가 {Pr(rj) > Pr(ri) and Pr(rj) > Pr(rk)}일 때 규칙 rj의 실행은 불가능하다.

$$TR(r_i, r_k \rightarrow r_{j(\text{or})}) = \{r_i, r_j, r_k \in R \mid (E(\text{Out}(r_i) \cap \text{TR-by}(r_j) \neq \phi) \cap r_j, E \neq \phi) \text{ or } (E(\text{Out}(r_k) \cap \text{TR-by}(r_j) \neq \phi) \cap r_j, E \neq \phi)\}$$ 일 때,

다음의 조건을 갖는 규칙 r_j은 실행 불가능하므로 이러한 규칙을 포함하고 있는 사이클 CR은 제거한다.

$$\text{Rid}(r_j \in CR) = \{r_i, r_j, r_k \in R \mid (Pr(r_j) > Pr(r_i) \text{ and } Pr(r_j) > Pr(r_k))\}$$

다음은 제 4장에서 설명한 2차 분석에서 사이클을 제거하기 위한 조건 정의이다.

[정의 10] 사이클에서, 일정한 횟수의 실행 또는 조건을 거짓으로 변경시키는 비활성화 관계에 의해 DR(r_i, r_j)가 존재하는 사이클은 제거한다[9](단, 활성화 관계 AR(r_k, r_j)의 존

재여부로 결정한다). 이것은 다음과 같이 표현한다.

$$\exists\{DE(r_i, r_j)\{\neg AR(r_k, r_j)\}$$

다음의 조건을 만족하는 사이클(CR)은 비활성화 관계를 이용하는 2차 분석에서 제거된다.

$$Rid(CR) = \{r_i, r_j, r_k \in CR \mid (Out(r_i) \cap Trans\text{-}into(r_j.Ce = False) \neq \phi) \text{ and } (Out(r_k) \cap Trans\text{-}into(r_j.Ce = True) = \phi)\}$$

[정의 11] 규칙 집합에 사이클이 존재하지 않으면 규칙 집합은 종료를 보장한다[16, 17].

이러한 규칙을 바탕으로 종료 분석은 다음과 같은 정리한다.

[정리 1] 실제로 실행이 불가능한 트리거 관계의 규칙(1차 분석)과 비활성화 관계의 규칙(2차 분석)을 포함하고 있는 사이클은 종료를 보장 할 수 있다.

증명 : 1차 분석 과정에서, 실행이 불가능한 규칙을 포함하고 있는 사이클은 정의 6에 의해 거짓 사이클이므로 이러한 규칙을 포함하고 있는 사이클은 정의 8과 정의 9에 의해 제거된다. 그리고 2차 분석에서의 예로써, 사이클을 형성하는 트리거 관계의 규칙집합 S = {r1, r2, r3}에 비활성화 관계DR(r1, r2)관계가 존재하면, 정의 2의 비활성화 관계에 의해 규칙 r1의 조치 실행으로 규칙 r2의 조건 값이 거짓으로 변경되어 조치절의 실행이 불가능하다는 것을 말한다. 그러므로 이러한 사이클은 규칙 r1에 의해 실행이 불가능한 r2를 포함하는 거짓 사이클이고, 정의 10에 의해 제거된다. 따라서 규칙집합 S는 종료를 보장할 수 있다. ■

[정리 2] 한정된 횟수만 실행 가능한 사이클은 일정 횟수의 수행 후 비활성화 되므로 이러한 규칙 관계를 포함하고 있는 사이클은 종료를 보장 할 수 있다.

증명 : 규칙의 종료 분석은 무한하게 반복 실행되는 사이클을 분석하는 것이다. 그러므로 일정한 횟수의 실행 후에는 규칙 집합에 포함되어 있는 임의의 규칙이 비활성화 되는, 즉 조건 값이 거짓이 되어 더 이상 실행이 불가능하다는 것을 의미하므로 이러한 사이클은 정의 6에 의해 거짓사이클이고, 정의 10에 의해 제거된다. 그러므로 이러한 규칙 집합은 종료를 보장할 수 있다. ■

6. 비교 분석

능동 규칙의 종료 분석을 위해, 컴파일 시간에 이루어지는 정적 분석과 실행 시간에 이루어지는 동적 분석 방법이 많이 발표되었다. 그러나 이 논문은 기존의 연구[10, 11, 19]와 다음과 같은 차이점이 있다.

- 규칙의 트리거 관계를 기반으로 하여 규칙의 실행 동작을 예측하는 종료 분석에서 규칙 의 사건 유형과 규칙의 실행 시점은 규칙의 실행 여부와 실행 순서에 중요한 요소로 작용한다. 그러나 [10, 11, 19]에서는 기본 사건만을 대상으로 할 뿐 규칙의 사건절에 명세되는 복합사건과 before, after 규칙의 실행 의미를 반영하는 연구가 수행되지 않았다.
- [19]에서는 트리거 그래프와 활성화 그래프를 이용하였고 [11]에서는 트리거 그래프와 활성화 그래프 그리고 비활성화 그래프를 이용하였다. 그러나 이 논문에서는, 전자의 규칙 실행 가능성만을 표현하여 종료 결정이 명료하지 않은 문제점과 후자의 규칙 관계 정 보를 결합해야 하는 복잡성을 개선한 트리거 그래프와 비활성화 결합그래프를 사용하여 빠른 분석 결정을 유도한다.
- 이 논문에서는 규칙 컴파일러를 위한 종료 분석기를 제시하여, 능동 데이터 베이스의 핵 심 요소인 능동 규칙의 종료 보장을 위한 규칙의 저장, 관리 그리고 정확한 종료 분석 방법을 포함하는 컴파일 과정을 제시하였다. 따라서 기존의 연구에서 단지 규칙의 종료 분석 방법만을 제시하였던 것과는 차별성을 갖는다.

7. 결 론

능동 데이터베이스에서의 능동 규칙은 자신을 포함한 서로 다른 규칙을 트리거할 수 있다는 특징이 있고 그로 인해 무한하게 실행될 가능성이 있다. 그러므로 규칙 실행의 비종료 문제를 해결하기 위해 이 논문에서는 규칙의 종료 분석을 수행할 수 있는 기능을 내장한 규칙 컴파일러를 설계하고 이 규칙 컴파일러의 수행 모델 및 알고리즘을 제안하였다. 아울러 알고리즘의 정형화를 통해 제안 모델의 완전성을 검증하였다.

규칙 컴파일러는 규칙 언어로 정의된 능동 규칙을 실행 가능한 규칙으로 컴파일해 주는 규칙 처리기의 일부로서, 규칙 집합의 종료 여부를 분석하는 종료 분석기를 핵심 요소로 한다. 이 분석기는 그래프 형성과 사이클 분석의 2단계로 구성되며, 입력된 규칙 집합이 실제로 실행되었을 때 무한하게 반복적으로 실행되지 않고 종료할 수 있는가를 분석하고 정의된 규칙의 올바른 수행 여부를 예측, 수정하는 최적화 과정을 수행하도록 하였다. 그러므로 규칙 컴파일러를 거친 규칙 집합은 종료를 보장할 수 있다.

이 논문의 종료 분석기는 기존의 분석방법에서의 문제점인 분석 대상의 규칙범위를 복합사건을 고려함으로써 확장하였고 실행 시점을 의미하는 before, after 규칙의 특성을

반영함으로써 분석 결과에 정확성을 기할 수 있었다. 또한 효율적인 분석과정을 위해 기존의 트리거 그래프 변형 및 비활성화 그래프를 결합하여 규칙의 실행관계를 쉽게 식별할 수 있도록 하였고, 연속적으로 실행 가능한 사이클 분석을 간결하게 할 수 있도록 하였다.

마지막으로, 제시된 분석 방법의 정확성을 종료 분석 방법의 개념을 정의하여 증명하였고, 기존의 연구와의 비교를 통해 이 논문의 특성을 설명하였다.

향후 연구방향으로는 이 논문에서 제안한 종료 분석기를 내장한 규칙 컴파일러를 구현하여 능동 데이터베이스 시스템의 응용 분야에 적용 및 활용할 계획이다.

참 고 문 헌

[1] Jeong-Seok Park, Ye Ho Shin, Kwang Woo Nam, Keun Ho Ryu, "Incremental Condition Evaluation for Active Temporal Rules," Journal of KISS(B), Vol.26, No.4, April, pp. 462-472, 1999.

[2] C. Zaniold, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, R. Zicari. "Design Principles for Active Rules," Chapter 4, Advanced Database Systems, Morgan Kaufman Pub, 1997.

[3] A. Vaduva, S. Gatziu, Klaus R. Dittrich, "Investigating Termination in Active Database Systems with Expressive Rule Languages," RIDS, pp.149-164, 1997.

[4] Baralis E., Ceri S., Paraboschi S ; "Run-Time Detection of Non-Terminating Active Rule System," Proc. of the 4th Intl. Conf. on Deductive and Object-Oriented Databases, DOOD'95, Singapore, December, 1995.

[5] Baralis E., Ceri S., Paraboschi S ; "Improved Rule Analysis by Means of Triggering and Activation Graphs," Proc.of 2nd intl. Workshop on Rules in Database Systems, RIDS '95, Athens, Greece, September, 1995.

[6] S. Ceri, J. Widom. "Application of Active Databases," Active Database Systems-Triggers and Rules for Advanced Databases Processing, Morgan Kaufmann pub, 1996.

[7] Mattos, Nelson M., An Overview of the SQL3 Standard, Database Technology Institute IBM-Santa Teresa Lab., Jul. 1996.

[8] ANSI/ISO/IEC International Standard (IS), Database Language SQL-Part 2 : Foundation (SQL/Foundation), ISO/IEC 9075-2 : 1999 (E), September, 1999.

[9] Ye Ho Shin, Jeong Hee Hwang, Keun Ho Ryu, "Termination Analyzer including Rule Execution Semantics," 정보처리학회논문지 D, 제8-D권 제5호, pp.513-522.

[10] S. Yeung, T. Wang LING "Unrolling Cycle to Decide Trigger Termination," Proc 25th VLDB Conf. Edinburgh, pp.483-493, 1999.

[11] D. Montesi, M, Bagnato, C. Dallera. "Termination Analysis in Active Database," Database Engineering and Applications, 1999.IDEAS '99 International Symposium Proceedings, pp.288-297, 1999.

[12] E. Baralis, S. Ceri and S. Paraboschi, "Modularization Techniques for Active Rules Design," ACM Trasaction on Database Systems, pp.1-29, March, 1996.

[13] E. Baralis and J. Widom, "An Algebric approach to rule analysis in expert Database Systems," Proc 20th VLDB Con, Santiago, Chile, September, 1994.

[14] Se Man O, "Introduction of Compiler," Jungiksa pub, 1994.

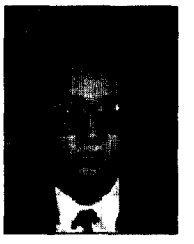
[15] J. Bailey, L. Crnogorac, K. Ramamohanarao, H. Sondergaard. "Abstract Inter - pretation of Active Rules and Its Use in Termination Analysis," ICDT'97, Leeture Notes in Computer Science, 99. pp.199-202, 1997.

[16] A. Aiken, J. M. Hellerstein, "Behavior of database production rules : Termination, Confluence, and Observable determinism," In Proceeding of the ACM SIGMOD conf. pp.59-68, San Diego, California, June, 1992.

[17] A. Aiken, J. M. Hellerstein, J. Widom, "Static Analysis Techniques for Predicting the Behavior of Active Database Rules," ACM Transaction on Database System, Vol.20, No.1, pp.3-41, March, 1995.

[18] S. Ceri and J. Widom, "Deriving Production Rules for Constraint Maintenance. In Dennis McLeod, Ron SacksDavid, and Hans Schek," editors, Proc. Sixteenth Int'l Conf. on Very Large Data Bases, pp.566-577, Brisbane, Australia, August, 1990.

[19] E. Baralis. "Rule Analysis," Chapter 3, Active Rules in Database Systems, Springer-Verlag Pub, 1999.



강 병 극

e-mail : bjjiang@dblab.chungbuk.ac.kr

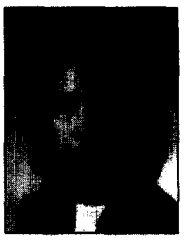
1986년 중국 연변대학교 물리학과 졸업 (이학학사)

1998년 한국 충북대학교 대학원 전자계산학과(이학석사)

2000년 한국 충북대학교 대학원 전자계산학과 박사수료

1986년~1995년 중국 연변전자계산기연구소 연구원

관심분야 : 시간 데이터베이스, 시공간 데이터베이스, 객체지향 데이터베이스, GIS



황 정 희

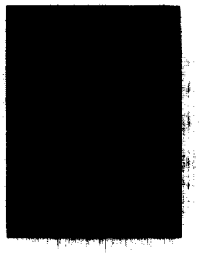
e-mail : jhhwang@dblab.chungbuk.ac.kr

1991년 충북대학교 전산통계학과 졸업 (학사)

2001년 충북대학교 대학원 전자계산학과 졸업(이학석사)

2001년 현재 충북대학교 대학원 전자계산학과 박사과정

관심분야 : 능동 데이터베이스, 시공간 데이터베이스, 데이터 마이닝



신 예 호

e-mail : snowman@dblab.chungbuk.ac.kr

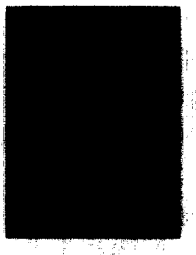
1996년 군산대학교 컴퓨터과학과 졸업 (학사)

1998년 충북대학교 대학원 전자계산학과 졸업(석사)

1998년 충북대학교 대학원 전자계산학과 입학(박사과정)

2000년~현재 충북대학교 대학원 전자계산학과 박사수료

관심분야 : 능동 데이터베이스, 시간 데이터베이스, 공간 데이터베이스, 데이터 마이닝



류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr

1976년 숭실대학교 전산학과(이학사)

1980년 연세대학교 산업대학원 전산전공 (공학석사)

1988년 연세대학교 대학원 전산전공 (공학박사)

1976~1986년 육군군수 지원사 전산실(ROTC 장교), 한국전자통신 연구원(연구원), 한국방송통신대 전산학과(조교수) 근무

1989년~1991년 Univ. of Arizona Research Staff(TempIS 연구원, Temporal DB)

1986년~현재 충북대학교 전기전자컴퓨터공학부 교수

관심분야 : 시간 데이터베이스, 시공간 데이터베이스, Temporal GIS, 객체 및 지식베이스, 지식기반 정보검색 시스템, 데이터 마이닝 및 데이터베이스 보안, 바이오 인포메틱스