그래프 기반의 비즈니스 프로세스 이상현상 검증 방법*

박찬희°, 탁경현, 김재형, 손진현
한양대학교 컴퓨터공학과
{pchlucky,takkh97}@database.hanyang.ac.kr,
jhkim@cse.hanyang.ac.kr, jhson@hanynag.ac.kr

Graph Based Detection Method Of Business Process Anomalies

Chan-hee Park°, Kyung hyun Tak, Jae Hyung Kim, Jin Hyun Son
Department of Computer Science and Engineering, Hanyang University in Ansan

요 약

최근 비즈니스 프로세스가 복잡해짐에 따라, 비즈니스 프로세스를 설계 한 후 실행하는 과정에서 예상치 못한 문제들이 발생할 가능성 또한 높아지고 있다. 예상치 못한 문제의 발생은 막대한 비용 및 인적 자원의 손실을 초래하기 때문에, 설계 된 비즈니스 프로세스는 엔진에 의해 실행되기 전에 다양한 방법으로 검증을 받아야 한다.

본 논문은, BPMN으로 설계된 비즈니스 프로세스의 컨트롤 플로우(Control Flow)에 대한 이상현상 중 데드락과 동기화의 부족에 대한 검증을 목적으로 한다. 데드락과 동기화의 부족은 Split 게이트웨이와 Join 게이트웨이의 종류에 따른 의미적 차이에 의해 발생한다. 본 논문에서는 비즈니스 프로세스를 방향성 그래프로 변환함으로써 Split 게이트웨이와 쌍을 이루는 Join 게이트웨이를 찾을 것을 제안하며, 찾아낸 (Split,Join)쌍은 컨트롤 플로우 검증 테이블을 이용하여 이상현상을 검증한다.

1. 서론

오늘날 기업의 업무가 다양해지고 복잡해지면서 비즈니스 프로세스 관리 시스템(Business Process Management System)을 이용한 비즈니스 프로세스의 설계 또한 복잡해지고 있다. 복잡한 프로세스가 타당성이 검증되지 않은 상태에서 프로세스 엔진에 의한 실행된다면, 잠재되어 있던 오류들이 프로세스의 실행 중에 발생할 수 있으며 이는 막대한 인적 손실 및 비용적인 손실을 초래한다. 때문에 설계된 비즈니스 프로세스의 유효성을 프로세스의 실행 이전에 검증하기 위한 다양한 검증 기법들에 대한 연구가 활발히 진행되고 있다[1,2,4].

그 중, 컨트롤 플로우에 대한 검증은[3] 수행될 태스크(Task)가 결정된 후 올바른 순서로 태스크가 설계되었는가에 대한 검증이며 비즈니스 프로세스의 이상현상에 대한 검증을 목적으로 한다. 비즈니스 프로세스의 이상현상으로는, 데드락(Dead lock), 동기화의 부족(Lack of Synchronization), 시작과 종료가 없는 노드, 무한루프가 있다.

본 논문에서는 BPMN으로 설계된 비즈니스 프로세스에 대하여 데드락과 동기화의 부족 이상현상을 기계적으로 검증하는 것을 그 목적으로 하며, 이를 위해 그래프 기반의 이상현상 검증 기법을 제안한다. 그래프 기반의 비즈니스 프로세스 이상현상 검증 기법의 순서는 그림 1과 같다. BPMN으로 설계된 비즈니스 프로세스를 정형화된 비즈니스 프로세스로 변환하고(단계 1), 정형화된 비즈니스 프로세스는 BPMN의 게이트웨이를 중심으로 레벨을 갖는 방향성 그래프로 변환한다(단계 2). 방향성 그래프를 통해 (split,join)쌍을 이루는 게이트웨이들을 찾은 후, 컨트롤 플로우 이상현상 검증 테이블을 이용하여 이상현상을 검증한다(단계 3). 이러한 검증기법은 시작 이벤트와 종료 이벤트가 각 1개씩 존재하는 정상적인 비즈니스 프로세스를 대상으로 한다.

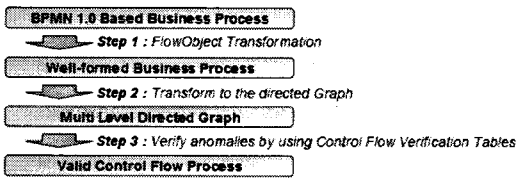

그림 1 그래프 기반의 이상 현상 검증 절차

본 논문의 구성은 다음과 같다. 2장에서 비즈니스 프로세스를 설계하기 위한 그래픽 표기법인 BPMN과 비즈니스 이상현상을 검증하기 위한 기존 연구에 대하여 기술한다. 그리고 3장에서는 본 논문에서 제안하는 그래프 기반의 검증을 위해, 설계된 비즈니스 프로세스를 정형화된 비즈니스 프로세스로 변환하는 방법을 기술한다. 4

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원 사업(IITA-2006-C1090-0603-0031)의 연구결과로 수행되었음.

장에서는 split 게이트웨이와 쌍을 이루는 join 게이트웨이를 찾기 위해 비즈니스 프로세스를 그래프로 변환하는 방법을 보이고, 5장에서 컨트롤 플로우 이상현상 검증 테이블을 이용한 검증을 기술한다. 마지막 6장에서는 본 논문의 결론을 맺는다.

2. 관련 연구

본 장에서는 비즈니스 프로세스를 설계하기위한 그래픽 표기법인 BPMN 및 설계된 비즈니스 프로세스의 컨트롤 플로우를 검증하기 위해 기존에 제안된 방법들에 대하여 간단히 기술한다.

2.1 BPMN

BPMN은 비즈니스 프로세스 다이어그램(Business Process Diagram)을 서식화하기 위한 프로세스 모델링 기호들로써, 비즈니스 프로세스와 관련된 실무자나 비즈니스 프로세스를 위한 기술자들의 원활한 의사소통을 위해서 만들어진 국제 표준이다. 복잡한 비즈니스 프로세스라도 BPMN에서 제공되는 다양한 모델링 기호들의 조합을 통해 충분히 표현 될 수 있으며 그 구성은 그림 2와 같다.

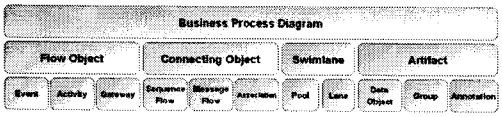

그림 2 BPMN의 구성요소

BPMN의 최상위 구조로 비즈니스 프로세스 다이어그램(Business Process Diagram)이 존재하며, 1개 이상의 프로세스로 구성이 된다. 프로세스는 하나의 업무를 수행하기 위한 규칙들과 참여자들의 정보 전달을 정의한다. 프로세스는 크게 플로우 오브젝트(Flow Object), 연결형 오브젝트(Connecting Object), 스윔레인(Swimlane), 아티팩트(Artifact)로 구분 할 수 있으며 각 항목은 여러 개의 세부항목으로 구분할 수 있다. 플로우 오브젝트는 프로세스 실행 중에 발생하는 사건의 행동을 정의하는 이벤트(Event)와 실제 작업을 의미하는 액티비티(Activity), 그리고 프로세스의 흐름을 결정하는 게이트웨이(Gateway)로 구성된다. 특히 게이트웨이는 프로세스의 흐름을 결정하는 형태에 따라서, AND 게이트웨이, OR 게이트웨이, XOR 게이트웨이 마지막으로 복합 게이트웨이(Complex Gateway)로 구분된다. 각각의 게이트웨이는 프로세스를 분할하거나, 결합하기위한 용도 중의 하나의 목적으로 사용된다. 연결형 오브젝트는 프로세스 내부에 존재하는 기호들의 흐름을 표현하는 순차 플로우(Sequence Flow)와 프로세스 간의 흐름을 표현하는 메시지 플로우(Message Flow) 그리고, 관계성을 가지는 객체들을 묶어주는 연합 플로우(Association)로 구성되며, 스윔레인은 프로세스를 대표하는 참가자의 역할을 표현하는 풀(Pool)과 프로세스를 사용하는 참가자의 역할을 나타내는 레인(Lane)으로 구성된다. 마지막으로 실제 프로세스에 영향을 미치지는 않지만 프로세스와 관련된 정보를 표현하기 위한 아티팩트는 데이터 오브젝트(Data Object), 그룹(Group), 주석(Annotation)으로 구성된다.

2.2 비즈니스 프로세스 이상 현상

비즈니스 프로세스를 디자인 할 때 발생 할 수 있는 대

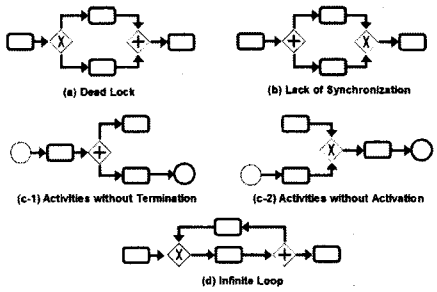

그림 3 비즈니스 프로세스 이상현상의 예

표적인 이상현상은 그림 3과 같다. BPMN에서는 '토큰' 개념을 사용하고 있다. 토큰은 순차 플로우를 따라서 플로우 오브젝트간에 이동하며, 토큰을 가진 플로우 오브젝트는 엔진에 의해서 실행된다. 데드락과 동기화의 부족은 join 게이트웨이와 split 게이트웨이가 가지는 의미적인 차이에 의해 토큰의 진행에 문제가 발생함으로 생기는 이상현상들이다.

① 데드락 : 특정 단계에서 다음 단계로 진행할 수 없는 경우
② 동기화의 부족 : join 게이트웨이로 입력되어지는 흐름들의 동기화를 맞추지 못하게 되어, 의도치 않게 특정 태스크가 여러 번 수행 되는 경우

비즈니스 프로세스가 복잡 해 질수록, 이상현상의 발생 가능성 또한 높아진다. 따라서 이상현상을 빠르게 검출하는 능력이 비즈니스 프로세스 관리 시스템에게 요구된다.

2.3 그래프 축소와 페트리 넷

기존에 연구된 비즈니스 프로세스 이상현상 검증 기법으로는 그래프 축소 방법과, 페트리 넷을 이용하는 방법이 있다. 그래프 축소[5,6]는 데드락 과 동기화의 부족 현상에 대하여 검증이 가능하며 터미널 축소(Terminal Reduction), 순차 축소(Sequential Reduction), 인접 축소(Adjacent Reduction), 종료 축소(Closed Reduction) 그리고 중첩 축소(Overlaping Reduction)이상 다섯 가지의 축소규칙을 반복적으로 적용하면서, 이상현상이 포함되지 않은 부분을 축소해 나가는 방법이다. 그래프 축소 방식은, 최악의 경우에도 복잡도가 $O(n^2)$으로 상당히 좋다는 장점이 있다.

페트리 넷은 비즈니스 프로세스를 표현하고 검증하기위해 오래전부터 사용된 방법이다. [11]에서는 페트리 넷을 기반으로 하는 워크플로우 넷(WF-net)을 구성하고 있다. WF-net은 정형화된 이론에 기반을 두고 있으며, 토큰 기반의 워크플로우 상태를 표현 할 수 있고, 상세한 분석을 할 수 있다는 장점이 있다. 하지만, 현재 대부분의 워크플로우 관리 시스템에서는 페트리 넷의 자체가 이해하기 힘들다는 단점과 표기에 대한 한계 때문에 WF-net을 사용하지 않는다. 또한 [7]에서는 과거에 비해 다양하고 복잡해진 비즈니스 프로세스의 기능들을 상당 부분에서 표현하지 못하고 있기 때문에 그에 대한 타당성 검증이 제약적임을 볼 수 있다. 예를 들어, 프로세스 간의 통신, 이벤트 호출, OR 게이트웨이 등에 대한 표현 방식이 언급되어 있지 않다.

3. 정형화된 비즈니스 프로세스

본 논문에서 제안하는 검증 방법은 게이트웨이를 기반으로 하기 때문에, 분할 및 결합의 기능을 모두 수행하는 것으로 표현된 게이트웨이는 split 게이트웨이와 join 게이트웨이로 분리할 필요가 있다. 또한 일부 상황의 경우 비즈니스 프로세스를 설계함에 있어서 게이트웨이를 생략하고 액티비티와 순차 플로우만으로 표현을 하기도 하는데, 이는 [8]의 "플로우 오브젝트 연결 규칙"을 준수하는 사항이라 할지라도 위와 동일한 이유로 게이트웨이를 삽입할 필요가 있다.

이번 장에서는 [8]을 준수하는 비즈니스 프로세스를 정형화된 비즈니스 프로세스로 변환하는 과정을 기술한다.

3.1 게이트웨이 분리

게이트웨이가 분할 및 결합의 기능 중 하나의 역할을 수행하도록 하기 위해서, 1개의 게이트웨이를 동일한 타입의 2개의 게이트웨이로 나누어야 한다. 이는 그림 4와 같이 수행된다.

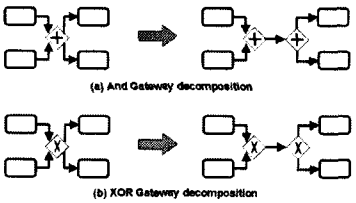

그림 4 게이트웨이의 분리

3.2 게이트웨이 삽입

[8]에 준수하여 게이트웨이가 생략된 경우, 그림 5와 같이 게이트웨이를 삽입할 수 있다. (a)와 같이 하나의 액티비티에 다수의 출력 순차 플로우가 관계하는 것은 이후의 액티비티들이 동시에 진행한다는 의미이므로 분할 AND 게이트웨이가 삽입된 것과 같다. (b)와같이 하나의 액티비티에 다수의 입력 순차 플로우가 관계하고 있는 비즈니스 프로세스는, 결합 XOR 게이트웨이를 삽입함으로써 동일한 의미를 나타낼 수 있다.

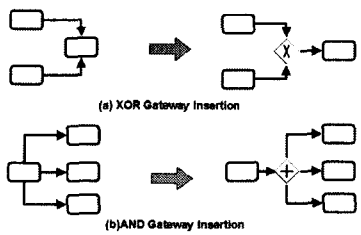

그림 5 게이트웨이의 삽입

4. 게이트웨이 기반의 방향성 그래프

split 게이트웨이와 쌍을 이루는 join 게이트웨이의 의미적인 차이는 데드락과 동기화의 부족과 같은 이상현상을 발생시키는 원인이 된다[9]. 본 논문에서는 검증의 첫 단계로서, 정형화된 비즈니스 프로세스를 레벨을 갖는 방향성 그래프로 변환함으로써 split 게이트웨이와 쌍을 이루는 join 게이트웨이를 찾을 것을 제안한다. 이를 위해, 비즈니스 프로세스를 구성하는 게이트웨이는 노드로 표현하고 게이트웨이와 연결된 순차 플로우는 간선으로 표현함으로써 비즈니스 프로세스의 흐름을 그래프로 표현한다. 2개 이상의 입력을 받는 노드는 join 게이트웨이에 해당하는 노드가 되며, 2개 이상의 출력을 갖는 노드는 split 게이트웨이에 해당하는 노드이다.

비즈니스 프로세스는 단일 스트림과 복합 스트림으로 구분할 수 있다. 단일 스트림은 그림 6과 같이 프로세스의 흐름이 한 방향으로만 진행되는 프로세스, 즉 루프를 포함하지 않는 프로세스를 의미한다. 복합 스트림은 그림 9와 같이 프로세스의 흐름이 한 방향으로 진행되다가 역류를 하는 상황이 존재하는 프로세스, 즉 루프를 포함하고 있는 프로세스를 의미한다. 본 장에서는 단일 스트림 및 복합 스트림 상황에서 비즈니스 프로세스를 그래프로 변환하는 방법에 대하여 기술한다.

4.1 단일 스트림의 그래프 구성

정리 1. 한 개의 split 게이트웨이는 한 개 이상의 join 게이트웨이와 쌍을 이룰 수 있으며, 역으로 한 개의 join 게이트웨이는 한 개 이상의 split 게이트웨이와 쌍을 이룰 수 있다.

정리 2 (split 게이트웨이와 쌍을 이루는 join 게이트웨이 탐색).

① 데드락, 동기화의 부족현상의 발생에 태스크는 영향을 미치지 않으므로, 태스크를 제외하고 게이트웨이와 순차 플로우만으로 그래프를 구성한다.

② 그래프 구성의 시작은 split 게이트웨이를 시작 노드로 구성하고, 이후 존재하는 게이트웨이들이 그래프의 노드가 된다.

③ 플로우 오브젝트 사이의 순차 플로우는 그래프의 간선으로 변환되며, 각 노드를 지날 때마다 그래프의 레벨은 1씩 증가한다.
④ 게이트웨이와 관계하는 각각의 입력 순차 플로우는 -1, 출력 순차 플로우는 1의 가중치를 가지며, 이 값은 그래프를 구성하는 동안 계속 누적한다.
⑤ 임의의 레벨 n에 해당하는 게이트웨이가 join 게이트웨이이고, 레벨 n까지의 가중치의 합이 0이라면 그래프의 구성을 종료한다. 이 때 레벨 n의 노드는 그래프의 시작 노드와 쌍을 이루는 노드이다.

정리 2의 4와 5는 다음과 같이 정리 할 수 있다. 그래프의 레벨을 n이라 하고 각 레벨에 입력되는 간선의 가중치의 합을 in, 각 레벨에서 출력되는 간선의 가중치의 합을 out이라할 때, 레벨 n에 위치하는 게이트웨이가 join 게이트웨이이고, $\sum_{i=0}^{n}(\text{in } i + \text{out } i)$ 의 값이 0 이라면 이는 그래프의 시작 노드와 쌍을 이루는 노드이다.

이러한 방법으로 구성된 그래프는 또 다른 split 게이트웨이를 포함 할 수도 있다. 만약, 그래프 안에 또 다른 split 게이트웨이가 존재한다면, 서브 그래프를 구성하기 위해서 모든 분할 게이트웨이에 정리 2의 2,3,4,5를 반복한다. 정리 2의 반복적인 적용을 통하여, 설계된 비즈니스 프로세스 내에서 존재하는 모든 split 게이트웨이들과 쌍을 이루는 join 게이트웨이를 찾을 수 있다.

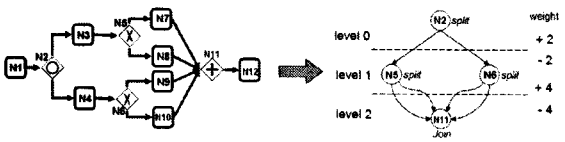

그림 6 단일 스트림에서의 그래프 구성

그림 6은 단일 스트림의 비즈니스 프로세스에 정리 2를 적용함으로써 구성된 그래프를 나타내고 있다. N2는 split 게이트웨이이므로 그래프의 시작 노드가 되며, 출력되는 간선의 수는 2개 이므로 레벨 0에서의 가중치는 +2가 된다. N3과 N4는 태스크로써 컨트롤 플로우에 영향을 끼치는 요소가 아니므로, 그래프 작성에서 제외한다. N5,N6은 split 게이트웨이이므로 그래프의 노드로 형성이 되며, 정리 2의 3에 따라 레벨은 1이 된다. 레벨 2로 입력되는 간선의 수는 2개 이므로 가중치는 -2가 부여된다. 이 시점에서 $\sum_{i=0}^{n}(\text{in } i + \text{out } i)$ 의 값은 0이 되지만 join 게이트웨이가 아니므로, 가중치 4를 부여하고 그래프 의 생성을 계속한다. N7, N8, N9, N10역시 태스크이므로 그래프 구성에서 제외되며, N11은 join 게이트웨이로써 레벨 2에 존재하는 노드가 된다. 이 시점에서 $\sum_{i=0}^{n}(\text{in } i + \text{out } i)$의 값도 0이고, N11도 join 게이트웨이로써 정리 2의 5를 만족하므로, 그래프의 구성을 종료한다. 이러한 연산을 통하여, N2는 N11과 쌍을 이루는 게이트웨이임을 알 수 있다. 완성된 그래프는 추가적으로 split 게이트웨이 N5와 N6을 포함하고 있다. 따라서 정리 2를 N5와 N6에 반복적으로 적용을 할 수 있으며, 이를 통해 N5와 N11, N6과 N11 역시 쌍을 이루고 있음을 알 수 있다. 이를 통해 join 게이트웨이 N11은 복수 개의 split 게이트웨이와 쌍을 이루고 있음을 확인 할 수 있다(정리1).

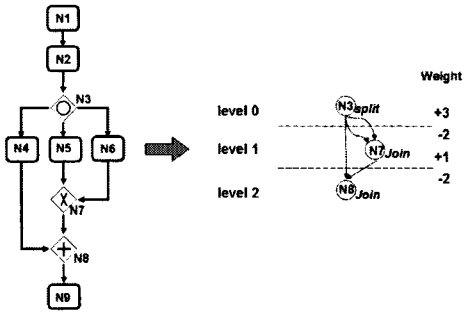

그림 7 복수개의 join 게이트웨이가 하나의 split 게이트웨이와 쌍을 이루는 비즈니스 프로세스

하지만 정리 2와 같이 split 게이트웨이를 중심으로 (split,join)쌍을 탐색하는 방법으로는 비즈니스 프로세스에 존재하는 모든 쌍을 찾을 수 없다. 그림 7은 정리 2에 의하여 만들어진 방향성 그래프를 보이고 있다. 여기서 N3와 N8이 쌍을 이루고 있음은 정리 2를 통하여 알 수 있지만, join 게이트웨이 N7 역시 N3와 쌍을 이루고 있음을 정리 2로는 찾아내지 못한다. 즉, 하나의 split 게이트웨이가 복수개의 join 게이트웨이와 쌍을 이루고 있다면(정리1), 비즈니스 프로세스의 모든 게이트웨이 쌍을 찾기 위해서 join 게이트웨이 관점에서 추가적으로 쌍을 찾을 필요가 있다. 이를 위해서 join 게이트웨이 관점으로 그래프를 구성하고 (split,join)쌍을 찾아내기 위한 정리 3을 제안한다.

정리 3 (join 게이트웨이와 쌍을 이루는 split 게이트웨이 탐색).

① 정리 2를 통해 완성된 그래프를 바탕으로 방향성을 역으로 갖는 그래프를 작성한다.
② 정리 2와 마찬가지로, 게이트웨이와 관계하는 각각의 입력 순차 플로우는 -1, 출력 순차 플로우는 1의 가중치를 가진다.
③ 각 join 게이트웨이를 시작으로 가중치를 누적하여 그 합이 0인 split 게이트웨이가 join 게이트웨이와 쌍을 이루는 split 게이트웨이다.

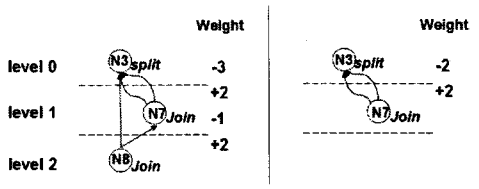

그림 8. join 게이트웨이 관점으로 구성한 방향성 그래프

그림 8은 그림 7에 정리 3을 적용한 그래프이다. N8과 N7에 정리 3을 적용함으로써 (N3,N8)쌍 과 (N3,N7)쌍이 비즈니스 프로세스내에 존재함을 알 수 있다.

이렇듯 split 게이트웨이의 관점과 join 게이트웨이 관점으로 쌍을 찾아냄으로써, 비즈니스 프로세스에 존재하는 모든 (split,join)쌍을 조사할 수 있다.

4.2 복합 스트림의 그래프 구성

복합 스트림은 루프를 포함하고 있는 비즈니스 프로세스를 의미하며, 이는 크게 세 가지로 구분 할 수 있다. 첫째 는, 쌍을 이루는 게이트웨이 안에서 루프가 형성되는 경우이며, 둘째 는, 쌍을 이루는 게이트웨이 안에 위치하는 split 게이트웨이와 밖에 위치하는 join 게이트웨이가 루프를 형성하는 경우이다. 마지막 셋째는, 쌍을 이루는 게이트웨이 안에 위치하는 join 게이트웨이와 밖에 위치하는 split 게이트웨이가 루프를 형성하는 경우이다.

그림 9는 루프가 발생할 수 있는 모든 경우를 표현 하였으며, 복합 스트림과 단일 스트림의 구분 및 복합 스트림의 형태에 따른 구분은 순차 플로우의 출처(source)와 목적지(target)정보를 유지함으로써 기계적으로 할 수 있다.

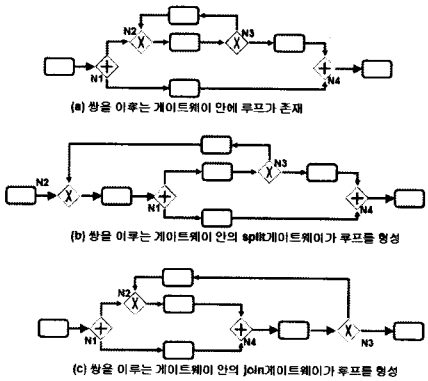

그림 9 복합 스트림의 예

그림 9의 (a)는 (N1,N4)의 쌍안에서 (N3,N2)쌍이 루프를 형성하고 있으며 비즈니스 프로세스의 흐름을 역행하고 있다. 이러한 모습에서는 N2에서 비즈니스 프로세스의 역행하는 흐름이 일반적인 흐름과 Join을 이루기 때문에 정리 2와 정리 3을 적용하여 정상적인 방향성 그래프를 작성할 수 없다. 이러한 문제의 해결을 위해 정리 4을 제안한다.

정리 4 (복합 스트림의 그래프 구성).
하나의 비즈니스 프로세스에서 흐름이 역행하는 부분이 존재한다면, 역행하는 부분을 서브 프로세스로 구성한다.

정리 4에 의하여, 그림 9의 (a)는 두 개의 프로세스로 표현된다. 첫 번째 프로세스는 정상적인 흐름을 갖는 프로세스로써, 정리 2를 통해 방향성 그래프를 구하고 (N1,N4)쌍을 찾을 수 있다. 두 번째 프로세스는 역행하는 흐름을 갖는 프로세스로써 이를 서브 프로세스로 간주하고, 서브 프로세스에 정리 2를 적용함으로써 (N3,N2)쌍을 찾을 수 있다. 그림 10은 그림 9의 (a)에 정리 2와 정리 4를 적용함으로써 구해진 그래프를 나타낸다. 그림 9의 (b),(c) 같은 복합 스트림도 이와 같은 방법으로 그래프로 변환하고 (split,join) 게이트웨이 쌍을 구할 수 있다.

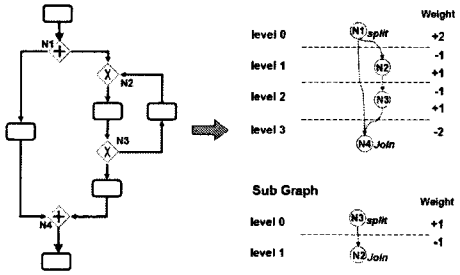

그림 10 복합 스트림의 그래프 변환

4.3 그래프 기반 (Split,Join)쌍 탐지 기법의 타당성

하나의 시작과 끝을 갖는 비즈니스 프로세스에서, split 게이트웨이로 나누어진 비즈니스 프로세스의 흐름은 프로세스가 끝나기 전에 join 게이트웨이를 통하여 하나의 흐름으로 통합되어야 한다. 이러한 불변의 진리는 보조 정리 1로 표현할 수 있다.

보조정리 1. join 게이트웨이는 자신과 쌍을 이루는 split 게이트웨이에서 분할한 흐름의 수를 1개의 흐름으로 되돌린다.

보조 정리 1을 확인하기 위해서는 게이트웨이의 역할, 비즈니스 프로세스의 흐름의 수에 대한 정보가 필요하다. 본 논문에서 제안하는 게이트웨이와 순차 플로우를 이용한 방향성 그래프는 보조 정리 1을 확인하기 위한 모든 정보가 포함된 그래프이다. 때문에, 그래프와 보조 정리 1을 바탕으로 시작 노드(split 게이트웨이)와 쌍을 이루는 종료 노드(join 게이트웨이)를 찾을 수 있다.

5. 데드락 및 동기화의 부족에 대한 검증

레벨을 갖는 방향성 그래프를 통하여 split 게이트웨이와 쌍을 이루는 join 게이트웨이를 찾을 수 있다. 본 장에서는 쌍을 이루는 게이트웨이에 대하여 데드락 및 동기화의 부족 이상현상을 검증하기 위해 컨트롤 플로우 이상현상 검증 테이블을 이용한다. 컨트롤 플로우 이상현상 검증 테이블은 BPMN에서 제안하는 AND,OR,XOR 게이트웨이들을 조합이 가능한 split/join 게이트웨이의 쌍으로 표현한 테이블로써, [10]을 바탕으로 상황별로 발생 가능한 비즈니스 프로세스 이상현상을 구성하였다. BPMN의 복합 게이트웨이의 경우 사용자가 기술하는 표현(expression)에 의하여 흐름이 결정됨으로 본 논문에서는 제외시킨다.

표 1 컨트롤 플로우 이상현상 검증 테이블

	AND	OR	XOR
AND	OK	OK	lack of sync
OR	deadlock	OK	lack of sync
XOR	deadlock	OK	OK

4절에 기술한 방법으로 구한 모든 (split,join)쌍 중에서, 두 노드 사이에 존재하는 간선의 수가 적은 (split,join) 쌍을 우선으로 컨트롤 플로우 이상현상 검증 테이블과 비교한다.

그림 6에서는 4절의 방법을 이용하여 (N2,N11), (N5,N11) 그리고 (N6,N11)의 쌍을 구할 수 있다. 세 개의 (split,join)쌍 중에서 간선의 수가 적은 (N5,N11) 혹은 (N6,N11)을 비즈니스 프로세스 검증 테이블에 비교를 하면, XOR split 게이트웨이와 AND join 게이트웨이이므로 데드락이 발생함을 알 수 있다.

(N5,N11)로 구성된 서브 프로세스가 이상현상을 가지고 있다면 이를 포함한 모든 프로세스는 이상현상을 내포하게 된다. 그러므로 복잡한 비즈니스 프로세스를 효율적으로 검증하기 위해서는 분할정복(Divide and Conquer)방식으로 검증을 수행해야 한다. 동일한 방법으로 그림 7의 그래프를 검증한다면, (N3,N7)에서 동기화의 부족 이상현상이 발생함을 알 수 있다.

비즈니스 프로세스 전 과정에 본 논문에서 제안한 검증기법을 적용하였을 때, 이상현상이 발생하는 서브 프로세스가 존재하지 않는다면 설계된 비즈니스 프로세스는 컨트롤 플로우 측면에서는 올바르게 설계되었다고 할 수 있다.

6. 결론

본 논문에서는 비즈니스 프로세스를 게이트웨이와 간선으로 구성된 레벨을 갖는 방향성 그래프로 변환하였다. 그래프를 이용하여 split 게이트웨이와 쌍을 이루는 join 게이트웨이를 찾아내고, 게이트웨이 간의 의미를 비교한 컨트롤 플로우 이상현상 검증 테이블을 이용하여 데드락과 동기화의 결함에 대한 이상현상을 검증하였다.

이러한 방법은 OR 게이트웨이에 관한 검증의 어려움을 간단하게 해결할 수 있으며, 비즈니스 프로세스가 설계되는 과정에서 이상현상을 기계적으로 검증할 수 있다. 향후, 비즈니스 프로세스 관리 시스템에 그래프 기반의 이상현상 검증방법을 적용함으로써 빠르고 정확하게 이상현상을 판단할 수 있을 것이다.

7. 참고문헌

[1] W.Sadiq and M.E. Orlowska, Analyzing process models using graph reduction techniques, Information System 25(2), 2000.

[2] W.M.P. van der Aalst and Arthur H.M. ter Hofstede, Verification of workflow task structures: A Petrinet-based approach, Information Systems 25(1) (2000) 43-69.

[3] W.M.P van der Aalst. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. Business Process Management, Models, Techniques, and Emirical Studies, 2000.

[4] Shazia Sadiq. Data Flow and Validation in Workflow Modeling. Proceedings of the fifteenth conference on Australasian database - Volume 27 (ACM), 2004.

[5] H. Lin, Z. Zhao, H. Li, and Z. Chen. A novel graph reduction algorithm to identify structural conflicts. In HICSS, page 289, 2002.

[6] W. Sadiq and M. E. Orlowska. Applying graph reduction techniques for identifying structural conflicts in process models. In Jarke and Oberweis , pages 195~209.

[7] W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. I n W. M. P. van der Aalst, J. Desel, and A. Oberweis, e ditors, Business Process Management, volume 1806 of Lecture Notes in Computer Science, pages 161~183. S pringer, 2000.

[8] Businss Process Modeling Notation(BPMN) (version 1.0 - May 3, 2004)

[9] HENRY H. BI. Applying Propositional Logic to Workflo w Verification. Information Technology and Managemen t, Volume 5, 2004.

[10] T. Maruta, S. Onoda, Y. Ikkai, T. Kobayashi and N. Ko moda, A deadlock detection algorithm for business pro cesses workflow models, in: IEEE International Confere nce on Systems, Man, and Cybernetics, Vol. 1 (11-14 October 1998) pp. 611-616.

[11] W.M.P. van der Aalst, Verification of workflow nets, in: The 18th International Conference on Application and Theory of Petri Nets, Lecture Notes in Computer Scie nce, Vol. 1248 (1997) pp. 407~426.