페트리넷을 이용한 워크플로우 명세의 완료가능성 및 무결성 검증

정 희 택† · 이 도 헌††

요 약

최근에 자동화된 업무처리 시스템으로써, 워크플로우 시스템에 대한 연구가 활발히 이루어지고 있다. 본 연구는 병행 수행되는 과업간에 다양한 종속성을 포함한 워크플로우 명세에 대해, 페트리넷을 이용한 완료 가능성 및 완료 무결성 검증 방안을 제안한다. 이를 위해 첫째, 기존에 연구된 워크플로우 그래프를 이용한 워크플로우 명세 방안을 간략히 기술한다. 둘째, 워크플로우 명세의 검증을 위해, 워크플로우 그래프를 페트리넷으로 변환하기 위한 방안을 제안한다. 셋째, 페트리넷을 이용하여 워크플로우 명세가 완료 가능한지를 검증하는 완료 가능성 검증과 원하지 않는 워크플로우 상태가 존재하는지 검증하는 무결성 검증방안을 제안한다. 넷째, 제안된 검증방안의 구현 및 모순 추출 방안을 제안한다.

Commitability and Integrity Verification of Workflow Specification with Petri Nets

Hee-Taek Ceong† · Doheon Lee††

ABSTRACT

A study on workflow as an automated business processing system is done recently. This paper proposes a method to verify commitability and integrity of workflow specifications, which include various dependencies between parallel tasks, using petri nets. For this, firstly, we describe briefly workflow specification method using workflow graphs. Secondly, we propose a method to convert a workflow graph into a petri net to verify workflow specifications. Thirdly, we suggest the commitability verification method using petri nets to check whether workflow specifications can commit or not and the integrity verification method to check whether invalid components exist or not. Lastly, we implement the proposed verification method and propose a defect extraction method.

1. 서 론

최근 기업의 업무 구조가 복잡해짐에 따라 워크플로우 시스템에 대한 관심이 크게 높아지고 있다[1, 2]. 워크플로우는 자동화된 업무 흐름을 말한다. 워크플로우 시스템은 컴퓨터와 통신망을 이용하여 체계적으로 워크플로우를 정의, 관리, 그리고 수행하는 시스템이다[3-8].

워크플로우 시스템에서 업무 흐름은 업무들 각각을 의미하는 태스크들과 그들간의 종속성으로 기술된다. 워크플로우 시스템에서 고려하는 태스크간 종속성의 예로는 순차, 분기, 반복, 병행 등이 있다. 특히 병행은 둘 이상의 태스크를 동시에 수행할 수 있다는 것을 명시하는 것으로 워크플로우 환경에서 빈번히 발생한다. 병행

※ 본 논문은 한국과학재단의 특정기초연구(과제번호 : 98-0102-11-01-3) 연구비 지원에 의한 것임
† 정 회 원 : 여수대학교 전자계산학과 교수
†† 정 회 원 : 전남대학교 컴퓨터정보학부 교수
논문접수 : 1999년 3월 4일, 심사완료 : 2000년 6월 19일

수행하는 태스크들은 다양한 요구사항이 존재할 수 있다. 예를 들면, 보험 워크플로우에서 병행 수행되는 신규 접수 태스크 및 종합검진 태스크간의 요구사항을 고려할 수 있다. 즉, 신규 접수 태스크가 시작된 후 종합검진 태스크가 시작될 수 있으며 종합검진 태스크가 종료되었을 때에만 신규 접수 태스크가 종료되어야 하는 요구사항이 존재한다. 한편, 여행 예약 워크플로우에서 특정 호텔예약에 대해 특정 회사의 차량 대여를 선호할 때, 병행 수행되는 차량대여 태스크와 호텔예약 태스크에 있어 차량 대여가 완료되었을 때만 호텔 예약이 완료되어야 하는 요구가 존재한다. 이러한 요구는 기존에 워크플로우 모델링 방안으로는 모델링 할 수 없다. 이러한 요구를 모델링하기 위해 [9, 10] 연구에서, 병행 수행하는 태스크들간에 순차시작 종속성, 순차종료 종속성, 동시시작 종속성, 동시종료 종속성, 중첩 종속성, 간섭된 종속성, 동시시작 순차종료 종속성, 순차시작 동시종료 종속성, 그리고 동시시작 및 종료 종속성을 제안하였다. 또한, [10]에서는 다양한 종속성을 포함한 워크플로우 명세의 모순 검증 방안을 제안하였다. 그러나, 제안된 연구는 워크플로우 명세에 대해 발생해서는 안될 성질만을 제시하고 이를 검증하는 방안만을 제안하였다. 즉, 태스크간의 존재해서는 안될 자가종속 모순 및 경로간 종속 모순을 제시하고 이를 발견할 수 있는 방안만을 제시하였다. 그러나, 워크플로우 명세의 검증은, 워크플로우 명세가 최종 목적(즉, 완료)에 도달 가능한지 검증할 수 있어야 하고, 원하지 않는 워크플로우 상태의 발생 여부를 검증하는 무결성 검증을 수행해야 한다. 워크플로우 명세의 완료 가능성 및 무결성 검증을 수행하기 위한 수단으로써 본 연구에서는 페트리넷을 이용한다. 이는 페트리넷[11]이 갖는 정형화된 명세의 특성과 다양한 분석 기법을 이용하여 완료 가능성과 무결성을 검증할 수 있기 때문이다.

[12-16]에서는 페트리넷을 이용한 워크플로우 모델링 및 검증 방안을 연구하였다. 먼저, [12-15]에서는 페트리넷의 플레이스(place) 및 트랜지션(transition)을 이용하여 워크플로우를 모델링 하였다. 즉, 태스크를 트랜지션으로, 워크플로우의 상태변화를 플레이스로 모델링 하였다. 워크플로우의 검증은 기존에 연구된 페트리넷 분석방법을 통해 달성하였다. 즉, 도달성(reachability) 및 지속성(liveness)을 기반으로 이루어졌다. 한편, [16]에서는 앞선 연구와 달리 태스크의 상태를 플레이스로, 시작, 완료준비, 완료 그리고 철회의 사건을 트랜지션으로 각각 모델링 하였다. 워크플로우 명세를 검증하기 위해 먼저, 페트리넷 상에서 해당 플레이스로의 전이를 보장할 수 없는 흡수관(siphon) 존재여부를 조사하거나 페트리넷이 최종 상태에 도달 가능한지 여부를 이용하여 검증하였다. 또한, 트랜지션의 시간제약에 대해 최소/최대 분석을 통해 트랜지션의 수행가능 여부를 검증하였다. 그러나, [12-16] 연구들은 페트리넷을 이용하여 모델링함으로써 본 연구의 그래프를 이용한 명세보다 설계자에게 복잡성을 제공하게 되며 다양한 병행 종속성 요소를 고려하지 않았다. 더욱이, [12-15]에서 제안된 방안으로는 본 연구에서 고려하고 있는 시작 및 완료 사건간의 종속성 요구를 표현할 수 없다. 한편, [12-16]에서는 페트리넷 상에서 모순의 존재여부만 검증할 뿐, 모순의 원인이 되는 경로간 종속 모순과 자가 종속 모순을 발견할 수 없다. 마지막으로, WFMC(WorkFlow Management Coalition)에서 제안한 참고 모델은 태스크간의 제어 흐름으로 순차 수행(sequential routing) 및 병행 수행(parallel routing)만을 제안하였다[5, 7]. 제시된 연구에서도 본 연구에서 고려하고 있는 다양한 병행 종속성을 고려하고 있지 않으며 그에 따른 모순 검증을 고려하고 있지 않다.

본 연구의 2장에서는, 다양한 병행 종속성을 포함한 워크플로우 모델링 방안을 간략히 기술한다. 3장에서는, 먼저 워크플로우 그래프로 기술된 워크플로우 명세의 페트리넷 변환방안을 제안한다. 다음으로 변환된 페트리넷에서 완료 가능성 및 무결성 검증 방안을 제안한다. 4장에서는 제안된 요소의 구현 방안과 특정 태스크들간의 모순 존재를 추출하는 모순 추출 방안을 제안한다. 5장에서는 본 연구에 대한 결론을 기술한다.

2. 다양한 병행 종속성을 포함한 워크플로우 모델링

본 장에서는 워크플로우 명세에 대해 완료 가능성 및 무결성 검증을 수행하기 위해, [10]에서 제안한 워크플로우 모델링 방안을 간략히 기술한다. 워크플로우를 구성하는 태스크는 노드로 표현되며 그들간의 종속성은 간선으로 표현된다. 그래프에는 초기 시작 태스크를 의미하는 노드와 마지막으로 수행될 종료 태스크가 하나씩 존재한다. 다양한 병행 종속성을 포함한 워

크플로우 그래프는 아래와 같이 정의한다.

정의 1. 워크플로우 그래프

워크플로우 그래프는 $WFG=(V,E)$는 다음과 같이 정의되는 방향 그래프이다. V는 태스크의 집합이고 E는 제어 연결자(control connector)의 집합이다.

각 태스크 $t \in V$는 태스크 식별자, AND 연결 형태를 갖는 제어 연결자의 식별자 집합, XOR 연결형태를 갖는 제어 연결자의 식별자 집합, 대안태스크 집합과 같은 네 개의 속성으로 구성된다. 여기에서, 태스크 식별자는 태스크 t를 유일하게 구별하게 한다. AND 연결 형태를 갖는 제어 연결자의 식별자 집합은 선행 태스크 모두가 완료될 때 태스크 t로의 전이를 가능하게 하는 제어 연결자들의 집합이다. XOR 연결 형태를 갖는 제어 연결자의 식별자 집합은 선행 태스크들 중 *하나*의 완료로부터 태스크 t로의 전이를 가능하게 하는 제어 연결자들의 집합이다. 대안 태스크 집합은 태스크 t가 철회되었을 때 수행될 수 있는 대안 태스크의 집합이다.

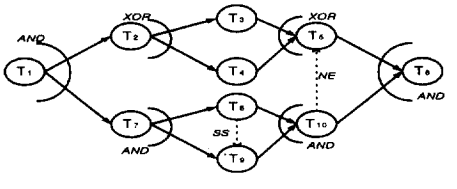
각 제어 연결자 $e \in E$는 식별자, 시작 태스크 식별자, 종료 태스크 식별자, 가능 조건, 종속성 형태로 구성된다. 식별자는 제어연결자 e를 유일하게 구별하게 한다. 시작 태스크 식별자와 종료 태스크 식별자는 제어 연결자 e가 연결하는 태스크의 식별자이다. 가능 조건(enable condition)은 제어 연결자에 부가된 조건식으로 워크플로우 데이터, 시간 데이터, 태스크 수행상태를 인수로 갖는 논리식이다. 종속성 형태는 순차 종속성이나 병행 종속성(즉, SS, PS, SC, PC, IE, SSPC, NE, PSSC, 그리고 PSPC) 중하나이다. □

병행 종속성은 동시에 수행되는 태스크의 시작 및 완료 사건의 관련성에 의해 순차시작(Sequential Starts : SS) 종속성, 동시시작(Parallel Starts : PS) 종속성, 순차종료(Sequential Commits : SC) 종속성, 동시종료(Parallel Commits : PC) 종속성, 간섭된(Interleaved Execution : IE) 종속성, 순차시작 동시종료(Sequential Starts Parallel Commits : SSPC) 종속성, 중첩된(Nested Execution : NE) 종속성, 동시시작 및 순차종료(Parallel Starts and Sequential Commits : PSSC) 종속성, 그리고 동시시작 및 동시종료(Parallel Starts and Parallel Commits : PSPC) 종속성으로 구분된다[1]. 또한 종속성들간에 다음을 만족한다. 다음 정리는 [10]에서 증명하였다. 본 논문에서는 정리 1의 특성에 의해 SS, PS, SC, 그리고 PC를 기본 병행 종속성이라 하고 IE, SSPC, NE, 그리고 PSSC를 고급 병행 종속성이라 한다.

정리 1. 병행 수행하는 임의의 태스크 T_i, T_j는 다음을 만족한다.

$$IE(T_i,T_j) \Rightarrow SS(T_i,T_j) \wedge SC(T_i,T_j)$$
$$SSPC(T_i,T_j) \Leftrightarrow SS(T_i,T_j) \wedge PC(T_i,T_j)$$
$$NE(T_i,T_j) \Leftrightarrow SS(T_i,T_j) \wedge SC(T_j,T_i)$$
$$PSSC(T_i,T_j) \Leftrightarrow PS(T_i,T_j) \wedge SC(T_i,T_j)$$
$$PSPC(T_i,T_j) \Leftrightarrow PS(T_i,T_j) \wedge PC(T_i,T_j)$$ □

다양한 병행 종속성은 워크플로우 그래프에서 노드들간에 레이블을 갖는 제어 연결자로 간단히 표현된다. 즉, 해당 종속성 형태를 제어 연결자의 레이블로 기술한다. 또한, 종속성 표현에 있어 $Start_{Ti} < Start_{Tj}$ (또는 $Commit_{Ti} < Commit_{Tj}$)와 $Start_{Ti} > Start_{Tj}$($Commit_{Ti} > Commit_{Tj}$)의 순서관계를 구분하기 위해 전자의 종속성은 T_i에서 T_j로 방향을 갖는 제어 연결자로 표현하며 후자는 반대 방향의 제어 연결자로 표현한다. 정의된 워크플로우 그래프를 기반으로 간단한 워크플로우 명세는 (그림 1)과 같다. 순차 종속성은 간단한 표현을 위해 레이블이 없는 제어 연결자로 표현하였고, AND 및 XOR 연결형태를 갖는 제어 연결자의 식별자 집합은 그래프의 판독을 쉽게 하기 위해 반호와 AND 및 XOR 레이블로 표현하였다. 또한, 다양한 병행 종속성을 갖는 제어식별자는 구별하기 쉽게 점선으로 표현하였다. (그림 1)은 두 가지 병행 종속성 요구를 모델링 하였다. 첫째, 태스크 T8과 T9간에 SS 종속성은 T8이 시작된 후 T9가 시작되어 병행 수행되어야 함을 모델링 하였다. 둘째, 태스크 T5는 T10 수행 동안에 수행되어야 함을 모델링 하였다. 한편, T5로 들어오는 제어 연결자 상에 XOR는 T3나 T4 중 한 태스크가 완료되었을 때 T5가 수행될 수 있음을 간략히 표현하는 연결 형태이다.



(그림 1) 다양한 병행 종속성을 포함한 워크플로우 그래프

1) [10]에서 제안된 병행 종속성 표현과 본 논문의 종속성 표현이 상이하나, 이는 다음과 같이 의미적으로 동일하다. SSPC⇔ST (Simultaneous Termination), PSSC⇔SS(Simultaneous Starts), PSPC⇔PE(Parallel Execution). 그리고 본 연구에서 고려하고 있는 SS, PS, SC, PC는 두 태스크간에 시작(또는 종료) 사건들간의 관련성만을 고려한 것이다. 이들의 의미는 정리 1에 표현하였다.

3. 워크플로우 명세에 대한 완료 가능성 및 무결성 검증

워크플로우 그래프로 기술된 워크플로우 명세는, 두 태스크간에 요구된 제약 조건을 종속성으로 정확하게 표현할 수 있다. 그러나, 다양한 종속성을 포함한 전체 워크플로우 명세가 완료 가능하고 논리적 모순이 없음을 가정할 수 없다. 본 장에서는 다양한 병행 종속성을 포함한 워크플로우 명세에 대해 완료 가능성 및 무결성 검증 방안을 제안한다. 완료 가능성은, 워크플로우 명세가 초기 태스크에서 마지막 태스크로의 제어흐름의 전이를 통해 완료될 수 있는지를 검증한다. 다음으로 무결성 검증은, 원하지 않는 워크플로우 상태의 존재여부를 검증한다. 완료 가능성과 무결성을 검증하기 위해 정형화된 명세와 다양한 분석 기법을 갖는 페트리넷을 이용한다. 또한, 본 연구에서 고려하고 있는 다양한 병행 종속성을 모델링하기 위해, 병행 수행을 표현할 수 있는 페트리넷을 이용한다. 이를 위해, 첫째, 그래프로 기술된 워크플로우 명세를 페트리넷으로 변환한다. 둘째, 변환된 페트리넷을 이용하여 완료 가능성 및 무결성 검증 방안을 제안한다. 또한, 기존연구[10]에서 발견된 경로간 모순 및 자가 종속 모순이 완료 가능성 및 무결성 검증을 통해 발견될 수 있음을 제시한다.

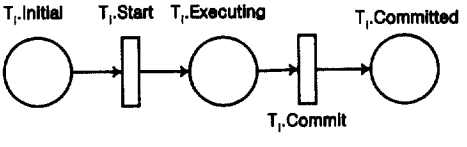
3.1 워크플로우 그래프를 페트리넷으로 변환

워크플로우 그래프를 페트리넷으로 변환은, 워크플로우를 구성하는 각 요소들을 페트리넷을 구성하는 트랜지션, 플레이스, 그리고 간선으로 변환함으로써 달성한다. 워크플로우 그래프를 페트리넷으로 변환하기 위한 알고리즘은 다음과 같다.

알고리즘. 워크플로우 그래프를 페트리넷으로 변환

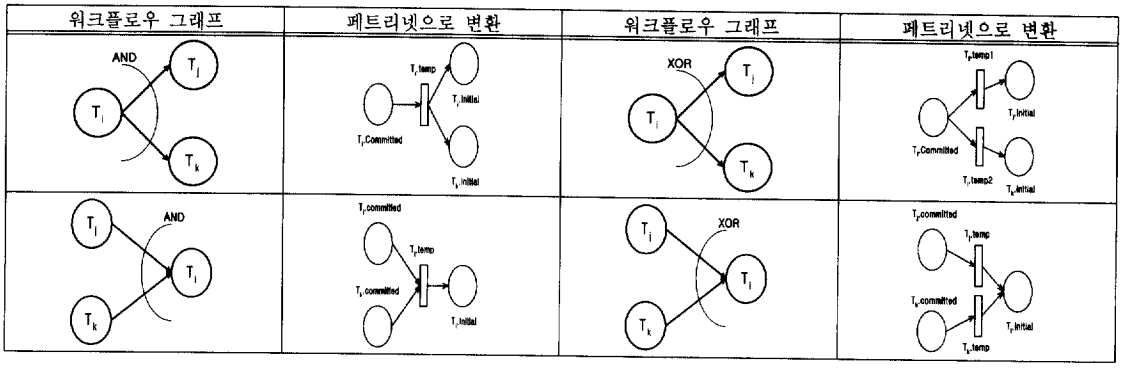
1. 각 태스크를 페트리넷의 플레이스와 트랜지션으로 변환
   - 태스크의 상태를 플레이스로 상태변이를 발생하는 사건을 트랜지션으로 변환
2. AND 및 XOR 연결자를 페트리넷으로 변환
   - AND 연결자를 갖는 태스크들은 동일한 트랜지션을 설정.
   - XOR 연결자를 갖는 태스크들은 각각 트랜지션을 설정
3. 태스크간 종속성을 페트리넷으로 변환
   3-1. 순차 종속성을 페트리넷으로 변환
   - 선행 태스크의 완료 플레이스와 후행 태스크의 초기화 플레이스간에 트랜지션을 이용 연결
   3-2. 다양한 병행 종속성을 페트리넷으로 변환
   - 정리 1에 의해 고급 병행 종속성을 기본 병행 종속성으로 변환
   - 기본 병행 종속성을 페트리넷으로 변환
4. 워크플로우 초기화, 완료 플레이스의 설정
   - 워크플로우 초기화 플레이스를 설정하고, 워크플로우 시작 태스크의 초기화 플레이스와 트랜지션을 이용 연결
   - 워크플로우 완료 플레이스를 설정하고, 워크플로우 종료 태스크의 완료 플레이스와 트랜지션을 이용 연결

단계 1에서는, 워크플로우를 구성하는 태스크는 다음과 같은 페트리넷으로 변환한다. 즉, 태스크의 상태를 플레이스로 표현하며 상태변이를 가능하게 하는 사건을 트랜지션으로 표현한다. 각 플레이스는 태스크의 시작할 수 있는 초기상태, 수행중인 상태, 그리고 완료된 상태를 의미한다. 시작 트랜지션은 초기 상태에서 수행 상태로의 전이를 가능하게 하는 사건을 표현하며, 완료 트랜지션은 수행 상태에서 완료된 상태로의 전이를 가능하게 하는 사건이다. 이는 [12-15]와 달리 태스크의 상태 및 사건을 플레이스와 트랜지션으로 표현함으로써 다양한 병행종속성을 표현할 수 있다. 한편, [16]에서 제안된 방안과 달리 완료준비 상태를 제외한 이유는, 지역 시스템이 완료준비상태를 지원하지 않을 수 있고 일반화된 표현을 위해 제외하였다. 또한 워크플로우 명세가 완료될 수 있음을 검증하기 위해 각 태스크의 철회 및 워크플로우 철회의 표현을 페트리넷 상에서 제외하였다.

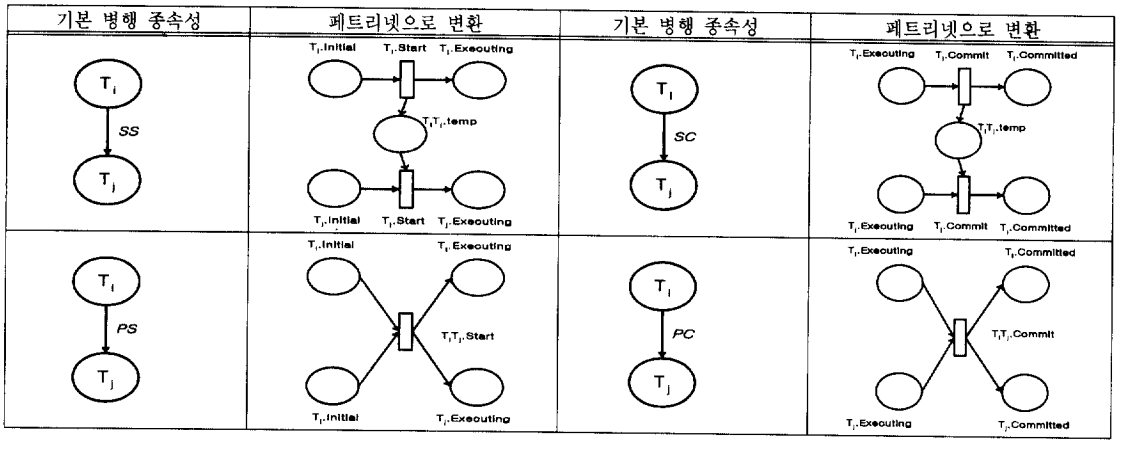


(그림 2) 태스크 T_i를 페트리넷으로 변환

단계 2에서는, 워크플로우 그래프에서 AND 및 XOR 연결형태를 페트리넷으로 변환한다. AND(XOR)연결형태는 (그림 3)과 같이 변환한다. 이는 [16]에서 제안된 것과 의미적으로 동일한 것으로 일반화된 변환방안이다.



(그림 3) AND 및 XOR 연결형태를 페트리넷으로 변환

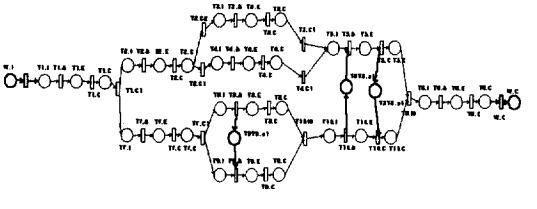


(그림 4) 기본 병행 종속성을 페트리넷으로 변환

단계 3에서는, 먼저, 워크플로우 그래프에서 순차 종속성을 페트리넷으로 변환한다. 순차 종속성은 페트리넷의 이분할 그래프(bipartite graph) 특성에 근거하여 플레이스나 트랜지션을 추가하여 그들간의 간선으로 표현한다. 다음으로, 병행 종속성을 페트리넷으로 변환은 두 단계로 구성된다. 먼저, 고급 병행 종속성(즉, IE, SSPC, NE, PSSC, 그리고 PSPC)을 기본 병행 종속성(즉, SS, PS, SC, PC)으로 변환한다. 다음으로 기본 병행 종속성 각각의 의미적 특성에 의해 페트리넷으로 변환한다. 특히 변환과정에서 PS와 PC를 선행 변환하고 SS, SC를 변환함으로써 페트리넷으로의 변환을 간략화 한다. 이는 PS를 갖는 태스크간에 SS 종속성을 요구시, PS에 의해 단일화된 트랜지션에 SS를 표현함으로써 변환을 단순화 할 수 있기 때문이다. (그림 4)에서 간략한 표현을 위해 관련된 트랜지션 및 플레이스만을 기술하였다. 표현되지 않는 태스크의 부분은 (그림 2)와 동일하다.

단계 4에서는, 전체 워크플로우의 초기화, 그리고 완료를 표현하기 위해 워크플로우 초기화 플레이스, 완료 플레이스를 생성한다. 워크플로우 시작 플레이스는 시작 태스크의 입력 플레이스가 되도록 정의하며, 완료 플레이스는 최종 태스크의 완료 플레이스의 출력 플레이스가 되도록 트랜지션을 추가 변환한다.

제시된 알고리즘에 의해 (그림 1)을 페트리넷으로 변환하면 (그림 5)와 같다. 간략한 표현을 위해, 각 태스크의 상태 및 사건 명을 초기 영문자로 기술한다.



(그림 5) 페트리넷으로 표현된 워크플로우 명세

3.2 워크플로우 명세에서 완료 가능성 및 무결성 검증

페트리넷에서 워크플로우 명세의 완료 가능성을 검증하기 위해, 먼저, 워크플로우 명세에서 완료 가능성을 정의하고, 이를 검증할 수 있는 방안을 제안한다. 워크플로우 명세에서 완료 가능성은 수행가능 경로상의 태스크 수행을 통해 워크플로우가 완료할 수 있음을 의미한다. 즉, 완료될 수 없는 워크플로우 명세는 워크플로우 목적을 달성할 수 없다. 이러한 특성에 기인한 워크플로우 명세의 완료 가능성은 다음과 같이 정의한다.

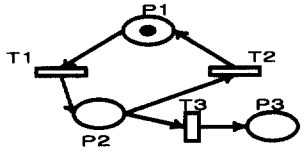
정의 2. 워크플로우 명세에서 수행가능 경로

워크플로우 명세에서, 초기 태스크로부터 최종 태스크까지의 수행될 수 있는 태스크들의 리스트(list)들를 수행가능 경로라 한다. □

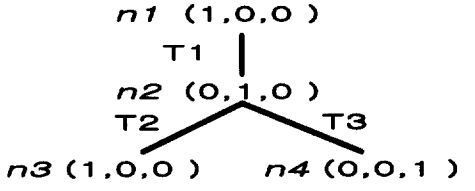
정의 3. 워크플로우 명세에서 완료 가능성

워크플로우 명세가 수행가능 경로상의 태스크들의 수행을 통해 완료할 수 있다면 이를 완료 가능성이라 한다. □

워크플로우 명세의 완료 가능성을 검증하기 위해 도달성 트리를 이용한다. 도달성 트리는 페트리넷을 분석하기 위한 수단으로 페트리넷 상에서 도달 가능한 모든 상태 집합을 표현한다. [11]에서 제안된 도달성 트리 생성방안에 의해 생성된 노드들은 내부(interior) 노드, 중복(duplicate) 노드, 그리고 단말(terminal) 노드를 생성한다. 내부 노드는 다른 상태로의 전이가 가능함을 표현하고, 중복 노드는 해당 상태를 다른 노드에서 표현하고 있음을 의미하고, 그리고 단말 노드는 더 이상 상태 변이가 불가능함(즉, 점화 가능 트랜지션이 존재하지 않는 상태)을 표현한다. 이를 예로 보이면 (그림 6)의 페트리넷을 가정 할 때, (그림 6)의 도달성 트리는 (그림 7)과 같다. 내부노드는 초기 노드 n1와 트랜지션 T1에 의해 생성된 상태를 표현하는 n2를 의미한다. n2 상태에서 T2의 점화에 의해 생성된 n3는 중복 노드이다. 이는 n3이 n1에 의해 이미 표현되고 있기 때문이다. 단말노드는 n4를 의미한다. 이는, n4 상태에서는 더 이상 점화 가능한 트랜지션이 존재하지 않기 때문이다.



(그림 6) 페트리넷 예



(그림 7) 페트리넷에 대한 도달성 트리

도달성 트리에서 완료 가능성 검증은 단말 노드를 기반으로 수행한다. 이는 더 이상 상태 변이가 없는 노드 정보를 기반으로, 발생해야 할 상태가 존재하는지를 검증한다. 즉, 완료 가능성 검증은 워크플로우의 완료 상태를 나타내는 플레이스에 토큰이 존재하는지 검사한다. 해당 플레이스에 토큰이 존재하지 않는다면 워크플로우 명세는 완료 될 수 없는 기술이다. 즉, 워크플로우 명세에는 수행할 수 없는 모순이 존재한다. 워크플로우 명세의 페트리넷 표현에서 완료 가능성은 다음과 같다. 완료 가능성 검증의 예는 예제 1에 기술하였다.

정리 2. 워크플로우 명세의 페트리넷 표현에서 완료 가능성

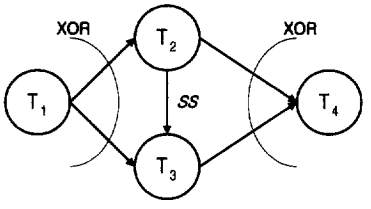
워크플로우 그래프 G를 표현한 페트리넷 C에 대해, C에서 도달 가능한 상태 집합을 표현하는 도달성 트리 R을 가정하자. 이때, 도달성 트리 R에서 워크플로우 초기 플레이스에만 토큰이 존재하는 초기 마킹 M0에 대해, 워크플로우 완료 플레이스에 토큰이 존재하는 마킹 Mc가 존재하면 워크플로우 명세는 완료 가능성을 보장한다.

증명 도달성 트리의 단말노드 특성과 토큰이 존재하면 해당 플레이스를 만족함을 의미하는 페트리넷 특성에 의해 자명하다. □

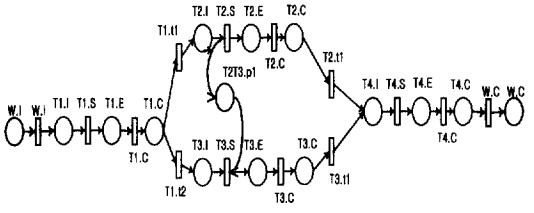
[예제 1] 완료 가능성 검증

(그림 8)과 같은 워크플로우 그래프를 페트리넷으로 변환(그림 9) 후 도달성 트리를 구성하면 (그림 10)과 같다. 도달성 트리를 구성하는 노드는 구현된 도달성 트리 생성 프로그램에 의해 14개의 노드가 생성되나 지면상 이유로 단말 노드를 중심으로 표현한다. 변환된 페트리넷의 플레이스 리스트가 (W.I,T1.I, T1.E,T1.C,T2.I,T2.E,T2.C,T3.I,T3.E,T2T3.p1,T3.C,T4.I, T4.E,T4.C,W.C)라고 가정하고, 하나의 워크플로우 인스턴스가 수행되기 때문에, 워크플로우 초기 플레

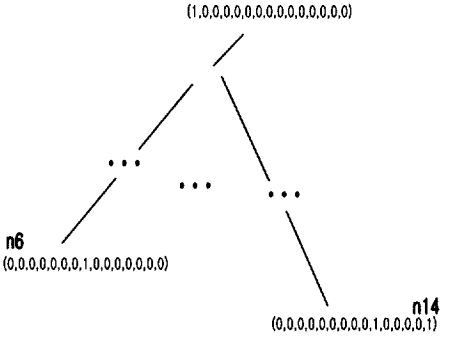
이스에만 토큰이 있는 (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)을 초기 마킹으로 가정한다.



(그림 8) 워크플로우 그래프의 예



(그림 9) 워크플로우 그래프를 페트리넷으로 변환 결과



(그림 10) 도달성 트리의 일부

(그림 10)에서 완료 가능성 검증은 단말 노드 중 워크플로우 완료를 의미하는 플레이스 W.C에 토큰이 존재함을 파악함으로써 달성한다. 단말 노드 n6에서는 W.C 플레이스에 토큰이 존재하지 않으나, 단말노드 n14에서는 W.C 플레이스에 토큰이 존재하기 때문에 (그림 8)의 워크플로우 명세는 완료 가능하다. 즉, 초기 상태에서 n14로 가는 상태 전이에 의해 (그림 8)의 워크플로우 명세는 완료 가능성을 보장한다. □

다음으로, 워크플로우 명세의 무결성 검증을 위해, 원하지 않는 워크플로우 상태를 정의하고 이를 검증할 수 있는 방안을 제안한다. 워크플로우 명세에서 원하지 않는 워크플로우 상태로는 워크플로우가 완료될 수 없는 임의의 상태와 워크플로우의 완료 후에 구성된 태스크들이 임의의 상태에 존재하는 경우를 의미한다. 전자는 해당 워크플로우 명세를 수행할 때, 워크플로우 완료 상태로 전이될 수 없는 명세의 기술을 의미한다. 후자는 워크플로우가 완료 상태로 전이될 수 있으나 구성된 과업이 수행 상태에서 다른 사건의 발생을 기다리는 상태를 의미한다. 이러한 특성에 기인한 워크플로우 명세의 무결성은 다음과 같이 정의한다.

정의 4. 워크플로우 명세에서 무결성

워크플로우 명세가 수행가능 경로상의 태스크들의 수행을 통해 다음이 발생하지 않는다면 이를 무결성이라 정의한다.

(1) 워크플로우가 완료될 수 없는 임의의 상태에 존재
(2) 워크플로우가 완료된 후 구성된 태스크들이 수행 상태에 존재 □

도달성 트리에서 무결성의 검증은 단말 노드를 기반으로 수행한다. 먼저, 워크플로우가 완료될 수 없는 임의의 상태는, 도달성 트리에서 워크플로우 완료 플래이스 이외의 플레이스에 토큰이 존재하는 단말 노드의 존재를 의미한다. 즉, 더 이상 상태변이가 없음을 의미하는 단말노드의 특성과 그 노드의 워크플로우 완료 플레이스에 토큰이 존재하지 않음으로써, 원하지 않는 상태를 검증할 수 있다. 이러한 무결성 검증은 앞서 제시한 완료 가능성과 구별된다, 완료 가능성은 워크플로우 명세에서 발생해야 할 요소로써 오직 완료 가능한가만을 검증한다. 즉, 예제 1에서 처럼 단말노드 n14를 이용 완료 가능한지 여부만을 검증하는데 반해, 완료될 수 없는 상태(즉 단말노드 n6)의 존재를 검증하는 무결성 검증은 발생해서는 안될 요소의 존재를 검증하기 위한 방안이다.

다음으로, 워크플로우가 완료된 후 구성된 태스크들이 수행상태에 존재는, 단말 노드 상에 워크플로우 완료 플레이스와 워크플로우 완료 플레이스를 제외한 플레이스에 토큰이 존재함을 의미한다. 워크플로우 완료 플레이스이외의 플레이스에 토큰의 존재는 해당 태스크가 워크플로우 완료 후에도 특정 상태에 존재함으로써 원하지 않는 상태의 존재를 의미한다. 워크플로우 명세의 페트리넷 표현에서 무결성은 다음과 같고, 워

크플로우 명세에서 무결성 검증의 예는 예제 2에 제시한다.

정리 3. 워크플로우 명세의 페트리넷 표현에서 무결성 워크플로우 그래프 G를 표현한 페트리넷 C에 대해, C에서 도달 가능한 상태 집합을 표현하는 도달성 트리 R을 가정하자. 이때, 도달성 트리 R에서 워크플로우 초기 플레이스에만 토큰이 존재하는 초기 마킹 M0에 대해, 단말 노드 중 다음을 만족하는 마킹 Mc이 존재하지 않으면 워크플로우 명세는 무결성을 보장한다.

(1) 워크플로우 완료 플레이스 이외의 플레이스에 토큰이 존재하는 마킹 Mc
(2) 워크플로우 완료 플레이스와 완료 플레이스 이외의 플레이스에 토큰이 존재하는 마킹 Mc □

증명 도달성 트리의 단말노드 특성과 토큰이 존재하면 해당 플레이스를 만족함을 의미하는 페트리넷 특성에 의해 자명하다. □

[예제 2] 무결성 검증

무결성 검증의 예를 보이기 위해, 예제 1의 워크플로우 명세로써 (그림 8)을 고려한다. (그림 8)에 대한 도달성 트리 (그림 10)에서, 노드 n6는 워크플로우 완료 플레이스 이외의 플레이스 T3.I에 토큰을 갖음으로써 무결성을 위배한다. 즉, (그림 8)의 워크플로우 명세가 T3의 초기상태로 전이 이후 더 이상 제어 흐름의 전이를 보장할 수 없음을 의미한다. 다음으로, n14는 워크플로우 완료 플레이스 이외의 플레이스에 토큰이 존재함으로써 무결성을 위배한다. 즉, 단말노드 n14는 W.C와 T2T3.p1 플레이스에 토큰이 존재한다. 이는 워크플로우가 완료되었으나, 임의의 태스크는 발생할 수 없는 상태 변이를 기다리고 있음을 의미한다. 정리하면, (그림 8)의 워크플로우 명세는 무결성을 만족하지 않는다. □

워크플로우 명세에서 존재하지 말아야 할 요소로써 [10]에서 경로간 종속모순과 자가 종속 모순을 제시하였다. 먼저, 경로간 종속 모순은 동시에 수행 불가능한 태스크간에 병행 종속성이 존재함으로써 발생하는 모순이다. 이는 서로 다른 수행경로 상의 태스크간에 종속성이 존재함으로써, 한쪽 수행경로에 포함된 태스크의 수행이 수행될 수 없는 다른 경로의 태스크 수행을 요구하게된다. 즉, 서로 다른 수행경로에 속하는 태스크의 수행을 기다리는 교착상태가 존재한다. 이러한 경로간 종속 모순을 포함한 워크플로우 명세를 페트리넷으로 변환할 때, 점화 가능한 모든 트랜지션을 점화시키는 도달성 트리의 특성에 의해 두 가지 상태로 표현된다. 첫째는, 경로간 종속성을 갖는 태스크중 점화 가능한 트랜지션의 점화에 의해 워크플로우 완료 플레이스에 도달하는 경우이다. 이 경우는 경로간 종속성을 표현하는 임시 플레이스에 토큰을 위치하게 한다. 둘째는, 경로간 종속성을 갖는 태스크중 점화 불가능한 트랜지션에 의해 도달성 트리에서 단말노드가 되는 경우다. 이 경우는 워크플로우 완료 플레이스에 토큰이 존재할 수 없다. 이러한 페트리넷 점화 특성에 의해 경로간 종속 모순의 검증은, 앞서 제시한 완료 가능성과 무결성 검증을 수행함으로써 검증한다. 즉, 전자의 상태는 완료 가능성 및 무결성 검증에 의해 달성되며 후자는 무결성에 의해 달성된다. 이를 예로 보이면 예제 3과 같다.

[예제 3] 경로간 종속 모순 검증

예제 1의 T2와 T3 태스크는 XOR 연결자에 의해 서로 다른 수행 경로에 포함되지만 그들간에 SS 종속성을 기술하고 있다. 즉, 동시에 수행될 수 없는 T1과 T3이 SS 종속성에 의해, T2의 시작은 T3의 시작을 또는 T3이 T2의 선행 시작을 요구한다. 이러한 경로간 종속성을 페트리넷으로 변환하면, (그림 9)와 같이 임시 플레이스(즉, T2T3.p1)와 간선으로 표현된다. 페트리넷의 점화 특성에 의해 한쪽 경로에 속한 태스크 T2의 수행이 임시 플레이스에 토큰을 위치하게 하거나, T3의 초기 상태(즉, T3.I 플레이스에 토큰이 위치)에서 T2의 시작(즉, T2.S의 점화)을 기다리게 된다. 전자는 도달성 트리에서 노드 n14로 표현되고 후자는 노드 n6으로 표현된다. 정리하면, T2와 T3은 XOR 연결자에 의해 한쪽 태스크만 수행될 수 있음으로 인해, 발생할 수 없는 서로의 수행을 기다린다. 즉, 교착 상태가 발생한다. 이러한 모순을 표현하고 있는 단말노드 n6과 n14에서 n6은 완료 가능성을 만족하고, n6과 n14는 무결성을 위반함으로써 발견된다. 즉, 경로간 종속 모순은 완료 가능성 및 무결성 검증에 의해 검증된다. □

다음으로, 자가 종속 모순은 동시에 수행될 수 있는 태스크간에 수행 불가능한 종속성 요구의 기술을 의미한다. 즉, 동일 수행가능 경로에 속하는 태스크들이 발

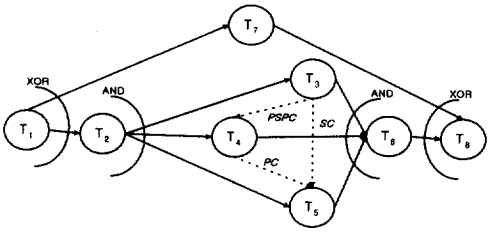
생할 수 없는 서로 다른 태스크의 수행을 요구하는 교착상태를 의미한다. 페트리넷으로 변환된 워크플로우 명세에서 교착상태는 경로간 종속 모순에서의 교착상태의 부분적 특성을 갖는다. 즉, 도달성 트리의 단말노드는 워크플로우 완료 플레이스 이외의 플레이스에만 토큰을 갖는다. 이러한 특성에 의해 자가 종속 모순의 검증은 무결성 검증에 의해 달성할 수 있다. 경로간 종속 모순 검증 방안과 달리 무결성 검증으로만 자가 종속 모순을 검증할 수 있는 이유는, 두 모순사이의 구조적 특성에 기인한다. 경로간 종속 모순은 동시에 수행될 수 없는 태스크간 즉, 최소 하나의 태스크는 수행되어 완료될 수 있는 구조에서 모순이다. 그러나, 자가 종속 모순은 동시에 수행될 수 있는 태스크간 즉, 모두가 수행되어야 완료될 수 있는 구조에서 모순이다. 전자는 모순을 갖는 태스크 중 한 태스크의 수행을 통해 상태 전이를 보장하면서 모순이 존재할 수 있지만, 후자는 모순을 갖는 태스크들의 수행을 통해 상태 전이를 보장할 수 없는 모순이다. 자가 종속 모순의 예를 보이면 예제 4와 같다.

[예제 4] 자가 종속 모순 검증

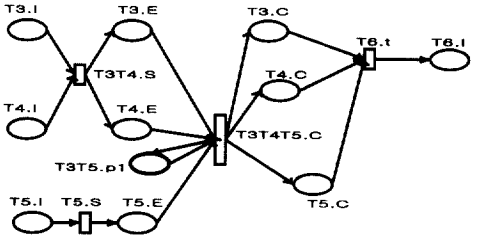
자가 종속 모순을 포함한 워크플로우 명세로 (그림 11)를 가정하자. (그림 11)에서 T3과 T5는 순차 완료 종속성(즉, SC), T3과 T4의 동시 완료 종속성(즉, PC) 그리고 T4와 T5의 동시 완료 종속성(즉, PC)에 의해, 다음과 같이 T3과 T5간에 동시 완료를 요구하면서 순차 완료를 요구하는 자가 종속 모순이 존재한다.

$$SC(T3,T5) \wedge \underline{PC(T3,T4) \wedge PC(T4,T5)} \rightarrow \underline{SC(T3,T4) \wedge PC(T3,T5)}$$

지면상 이유로 종속성 중심의 페트리넷 일부를 표현하면 (그림 12)과 같다. (그림 12)의 페트리넷에 대한 도달성 트리를 생성하면 워크플로우 완료 플레이스에 토큰이 하나 존재하는 단말노드와 T3.E, T4.E, 그리고 T5.E에 각각 토큰이 존재하는 단말노드가 존재한다. 전자는 T1,T7,T8의 수행 경로에 의한 워크플로우 완료를 의미하며, 후자는 T3과 T5간에 SS 종속성을 강요할 수 없음을 의미한다. 이는 T3T5.p1 플레이스에 토큰이 존재할 수 없음으로써 발생한다. 즉, 무결성을 만족하지 않는다. 정리하면, (그림 11)의 워크플로우 명세는 무결성 검증에 의해 자가 종속 모순을 검증할 수 있다. □



(그림 11) 자가종속 모순을 포함한 워크플로우 명세



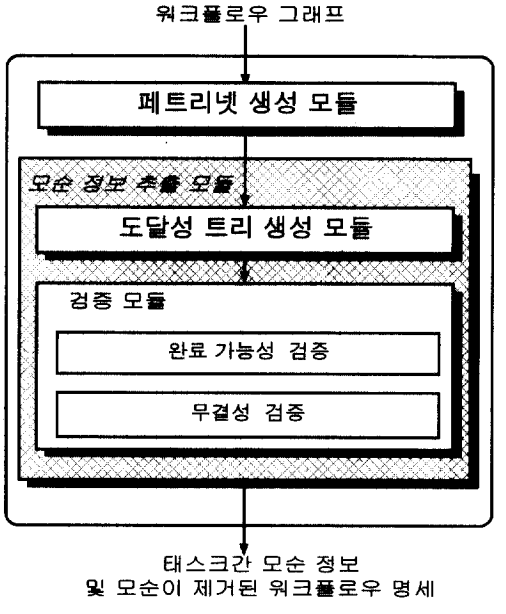
(그림 12) 변환된 페트리넷의 일부

4. 워크플로우 명세의 검증모듈 구현 및 모순 정보 추출

그래프로 표현된 워크플로우 명세를 검증하기 위한 구현방안을 제시한다. 먼저, 구현을 위한 전체 모듈을 간략히 기술한다. 다음으로 검증 단계의 반복 수행을 통해 모순의 원인이 되는 경로간 종속 모순과 자가 종속 모순의 발견 및 추출방안을 제시한다. 모순의 원인은 설계자에게 익숙한 태스크 수준의 정보를 추출한다.

4.1 검증 모듈 구현

워크플로우 그래프로 기술된 워크플로우 명세를 검증하기 위해 페트리넷 기반 모듈을 구현하였다. 구현된 모듈은 (그림 13)과 같고, SUN 워크스테이션 Solaris 2.5.1에서 C언어를 이용 구현하였다. 완료 가능성 및 무결성은 도달성 트리를 기반으로 검증 모듈에서 수행한다. 먼저, 페트리넷 생성 모듈에서는 워크플로우 그래프로 기술된 워크플로우 명세를 3.1장에서 제안된 방안을 이용 페트리넷을 생성한다. 다음으로, 페트리넷에 대한 도달성 트리 생성 모듈은 [11]에서 제안한 방안을 이용 생성한다. 마지막으로 생성된 도달성 트리를 이용 3.2장에서 기술한 완료 가능성 및 무결성 검증을 수행한다.



(그림 13) 워크플로우 명세의 검증 모듈

4.2 모순 정보의 추출

모순 추출 모듈은 검증 모듈을 통해 수행된 결과를 기반으로 태스크간 모순 정보를 추출한다. 이는, 완료 가능성과 무결성 검증 모듈은 워크플로우 명세에 모순이 존재하는지 여부만을 검증할 뿐, 특정 태스크간의 모순을 추출하지 못하기 때문이다. 현재 개발된 모순 추출 모듈은 경로간 종속 모순과 자가 종속 모순 정보를 추출한다. 즉, 특정 태스크간의 경로간 종속 모순이나 자가 종속 모순 존재를 추출한다. 워크플로우 명세에 다양한 자가 종속 및 경로간 종속 모순이 존재할 수 있기 때문에, 모순이 발생하지 않을 때까지 모순의 발견 및 해당 종속성 제거를 반복 적용한다. 해당 모순 정보를 추출하고 반복 적용함으로써, 워크플로우 명세의 모순정보와 모순이 제거된 워크플로우 명세를 생성한다.

모순 정보의 추출은 워크플로우 그래프, 페트리넷, 그리고 도달성 트리를 이용 수행한다. 모순 정보 추출을 위해 첫째, 도달성 트리에서 완료 가능성이나 무결성을 위배한 각 단말노드의 플레이스를 추출한다. 둘째, 각 노드의 추출된 플레이스를 입력 또는 출력으로 갖는 트랜지션 각각을 페트리넷으로부터 추출한다. 셋째, 추출된 트랜지션 각각에 대한 입력 플레이스 중 토큰이 존재하지 않는 입력 플레이스를 추출한다. 이는 해당 트랜지션을 점화시키지 못하게 하는 원인을 발견함을 의미한다. 이때, 표현의 간편성을 위해, 추출된 트랜지션에 대해 입력 플레이스로써 토큰을 갖는 플레이스를 토큰 플레이스라 하고 토큰을 갖지 않는 플레이스를 비토큰 플레이스라 하자. 넷째, 토큰 및 비토큰 플레이스가 소속된 태스크 식별자를 추출한다. 다섯째, 워크플로우 그래프에서 추출된 태스크간의 종속성 정보를 추출한다. 여섯째, 종속성을 갖는 태스크들이 XOR 연결자에 의해 연관된 태스크들이면 경로간 종속 모순을 출력하고, 그렇지 않으면 자가 종속 모순을 출력한다. XOR 연결자에 의해 연관된 태스크관계의 결정은, 워크플로우 그래프에서 간선의 방향을 무시한 그래프에 대해, 두 태스크간의 경로를 구하고 그 경로간에 하나이상의 XOR만의 연결자가 존재하면 두 태스크는 XOR 연결자에 의해 연관된 태스크들이다. 제시된 다섯 단계를 반복 적용하는 모순 추출 알고리즘은 다음과 같고 예제 5에서 예를 제시한다.

알고리즘. 워크플로우 명세에서 경로간 종속 및 자가 종속 모순 추출

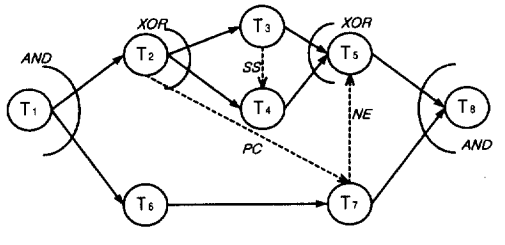
```
ts[]={"Initial",Executing","Committed"};

While(1) { /* 반복 수행 */
Each 도달성 트리에서 단말노드
if( 단말 노드 완료 가능성이나 무결성 위배 )
    TerminalnodeSet = TerminalnodeSet ∪ 단말노드;
if( TerminalnodeSet == φ ) {
    break;
} else {
   Each 노드 ∈ TerminalnodeSet
      노드에서 완료 플레이스의 토큰 개수 = 0;
      Each 토큰을 갖는 플레이스
         Transition_Set = Transition_Set ∪ 플레이스의 출력트랜지션;
         Each Ti ∈ Transition_Set
            InputPlace_Set = InputPlace_Set ∪ Ti의 입력 플레이스;
         Each Ti.ts[j] ∈ InputPlace_Set
            TaskID_Set = TaskID_Set ∪ I;
         워크플로우 그래프에서 TaskID_Set내의 태스크들 간에 종속성 정보를 추출
         Each 종속성 정보/* Ti,Tj,SS는 Ti와 Tj간에 SS 종속성 존재를 의미 */
            워크플로우 명세에서 Ti와 Tj간 종속성 제거
         if(XOR 연결자에 의해 연관된 태스크(Ti,Tj) == TRUE )
            print("Ti와 Tj는 경로간 종속 모순 존재");
         else
```

```
                print("Ti와 Tj는 자가 종속 모순 존재");
        도달성 트리 생성( 변경된 워크플로우 명세 )
워크플로우 명세에서 경로간 종속 및 자가 종속 모순 추출
  }
 } /* While end */
```

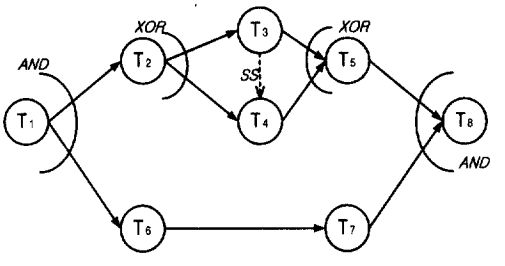
[예제 5] 경로간 종속 및 자가 종속 모순 정보 추출

(그림 14)에는 먼저, T3과 T4간에 경로간 종속 모순이 존재하다. 즉, T3와 T4가 동시에 수행될 수 없음에도 T3 시작 후 T4의 시작을 요구하거나 T4 시작을 위해 T3의 시작을 요구하는 모순이다. 다음으로 T2와 T7간에 PC 종속성, T7과 T5간에 NE 종속성(즉, 정리 1에 의해 T7과 T5간에 SS종속성 그리고 *T5와 T7간에 SC종속성*), 그리고 *T2와 T5간에 순차 종속성*을 갖는 자가 종속 모순이 존재한다. 즉, T2와 T5간에 순차 종속성과 T5와 T7간의 SC 종속성에 의해 T2와 T7은 SC 종속성을 의미하면서 직접 기술된 T2와 T7간의 PC 종속성을 갖는다. T2와 T7간에 SC 및 PC 종속성을 갖는 자가 종속 모순이 존재한다.



(그림 14) 경로간 종속 및 자가 종속 모순을 갖는 워크플로우 명세.

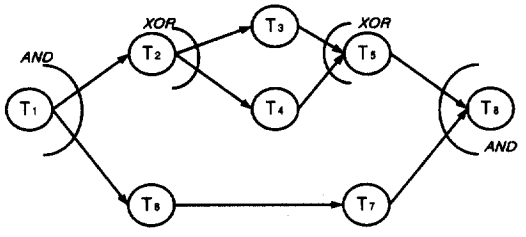
(그림 14)의 명세를 페트리넷으로 변환하여 도달성 트리를 구성하고 무결성 검증을 통해, T2의 수행 플레이스(T2.E), T7의 수행 플레이스(T7.E), 그리고 T7과 T5간의 SS 종속성을 표현하는 임시 플레이스(T7T5.p1)에 토큰이 존재하는 단말노드를 파악할 수 있다. T2.E와 T7.E의 출력 트랜지션에 대한 입력 플레이스 중 비토큰 플레이스는 T5T7.p2 플레이스로 T2,T7,그리고 T5간의 모순 존재를 의미한다. T7T5.p1의 출력 트랜지션의 입력 플레이스 중 비 토큰 플레이스는 T5.I로 T7과 T5간의 모순 존재를 의미한다. *이는, 워크플로우 명세에서 T2와 T7간에 PC 종속성이 존재하고 T7과 T5간에 SS종속성이 존재하기 때문에 발생하는 것으로 이들간의 종속 모순을 제거한다.* 더욱이 이들은 XOR 연결형태 없이 연관관계를 갖기 때문에 자가 종속 모순에 해당된다. 이를 제거한 워크플로우 명세는 (그림 15)과 같다.



(그림 15) 자가 종속 모순이 제거된 워크플로우 명세

다음으로 자가 종속 모순을 제거한 워크플로우 명세(그림 15)에 대해 도달성 트리를 구성하여 모순 추출 과정을 수행한다. 도달성 트리에서 T4의 초기 플레이스(T4.I)와 T7의 수행 상태를 표현하는 플레이스(T7.E)에 각각 토큰이 존재하는 단말 노드와 T3과 T4간의 SS 종속성에 의한 임시 플레이스(T3T4.p1)에 토큰이 존재하는 단말노드를 발견할 수 있다. 먼저 T4.I의 출력 트랜지션에 대한 입력 플레이스 중 비토큰 플레이스는 T3T4.p1으로 T3와 T4간의 모순을 의미한다. 다음으로, T7.E의 출력 트랜지션에 대해 입력 플레이스 중 비토큰 플레이스는 T5.C로 T5와 T7간에 종속성 관계가 존재하지 않기 때문에 모순이의 원인이 아니다. 마지막으로, T3T4.p1의 출력 트랜지션에 대해 입력 플레이스 중 비토큰 플레이스는 T4.I로써 T3와 T4간에 모순을 의미한다. 이때, T3와 T4간에 XOR 연관관계에 있기 때문에 T3와 T4간에 경로간 종속 모순을 발견한다. 제시된 과정에 의해 T3와 T4간의 SS 종속성을 제거한 워크플로우 명세는 (그림 16)과 같다.

결과로 생성된 (그림 16)의 워크플로우 명세에 대한 검증을 수행하면 워크플로우 완료 상태에만 토큰이 존재함으로써 완료 가능성과 무결성을 만족하고 수행 가능한 워크플로우 명세이다. 각 모순 추출과정에서 발견된 모순 원인은 다음 두 가지가 존재한다. 첫 번째는 T2,T7,T5간의 자가 종속 모순이며, 두 번째 모순은 T3과 T4가 XOR 연결자로 연관된 경로간 종속 모순이다.



(그림 16) 모순이 제거된 워크플로우 명세

모순 추출 알고리즘에 의해 (그림 14)의 워크플로우 명세에서 T2,T7,T5간의 자가 종속 모순 정보와 T3과 T4간의 경로간 종속 모순 정보를 추출한다. □

5. 결 론

본 연구는 다양한 병행 종속성을 포함한 워크플로우 명세에 대해 페트리넷을 이용한 완료 가능성 및 무결성 검증 방안을 제안하였다. 이를 위해 첫째, 워크플로우 그래프를 이용한 워크플로우 명세 방안을 간략히 기술하였다. 둘째, 워크플로우 명세의 검증을 위해, 워크플로우 그래프를 페트리넷으로의 변환 방안을 제안하였다. 제안된 방안에서는 다양한 병행 종속성을 페트리넷으로 변환할 수 있다. 셋째, 변환된 페트리넷에서의 발생해야 할 요소가 발생하는지 검증하는 완료 가능성 검증과 발생하지 말아야 할 요소가 발생하는지 검증하는 무결성 검증방안을 제안하였다. 이를 위해 먼저, 워크플로우 명세 관점에서 완료 가능성을 정의하였고, 이를 도달성 트리를 이용 검증할 수 있는 방안을 제안하였다. 다음으로 워크플로우 명세에 존재하지 말아야할 요소를 검증하기 위해 무결성을 정의하였고, 이를 도달성 트리를 이용 검증할 수 있는 방안을 제안하였다. 또한, 완료 가능성과 무결성을 기반으로 모순의 원인이 되는 경로간 종속 모순 및 자가 종속 모순이 페트리넷 상에서 검증됨을 보였다. 넷째, 제안된 검증방안의 구현 및 모순 추출 방안을 제안하였다. 제안된 모순 추출 방안은, 경로간 종속 모순 및 자가 종속 모순을 워크플로우 명세로부터 추출한다.

본 연구는 다양한 병행 종속성을 페트리넷으로 변환할 수 있음으로써 간편한 워크플로우 명세를 가능하게 한다. 이는 페트리넷을 이용하지 않고 방향성 그래프를 이용하여 워크플로우를 명세함으로써 설계자에게 간편성과 단순성을 제공하며 나아가 정형화된 검증방안을 제공한다. 즉, [12-16] 연구에서처럼 페트리넷으로 워크플로우를 명세하기 위해서는, 설계자는 페트리넷에 대한 기본 지식이 필요하고 정확한 명세에 대한 부담이 존재한다. 그러나, 쉽게 이해될 수 있는 방향성 그래프를 이용하여 다양한 병행 종속성을 명세하도록 하고 이를 페트리넷으로 변환함으로써, 워크플로우 설계자에게 쉬운 명세 방안을 제공하고 이를 검증할 수 있다. 또한, 워크플로우 명세의 검증된 정보를 기반으로 모순의 원인에 대한 정보를 추출할 수 있다. 이는 완료 불가능한 워크플로우 명세의 검증뿐만 아니라 모순을 야기하는 태스크간 종속성 정보를 발견한다. 즉, 페트리넷을 이용한 검증방안만을 제안한 기존 연구 [12-16]와 달리, 본 연구에서는 워크플로우 설계자에게 모순 발생 원인에 관한 워크플로우 명세 수준의 정보를 제공한다. 마지막으로, 워크플로우 검증을 위해 페트리넷을 이용함으로써 페트리넷 연구에서 적용할 수 있는 다양한 분석 기법을 쉽게 채용할 수 있다.

향후 연구에서는 모순 추출방안을 보다 효율적으로 수행할 수 있는 방안을 연구고자 한다.

참 고 문 헌

[1] A. Elmagarmid and W. Du, 'Workflow Management : State of the Art vs. State of the Market', Advances in Wrokflow Management Systems and Interoperability, A. Doğaç (Ed.), NATO, pp.1-17, 1997.

[2] F. Leymann and D. Roller,"Workflow-based applications," IBM system journal, Vol.46, No.1, pp.102-123, 1997.

[3] H. Davulcu, Michael Kifer, C.R. Ramakrishnan and I.V. Ramakrishnan,"Logic Based Modeling and Analysis of Workflows," Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.25-33, June 1998.

[4] M. Rusinkiewicz and A. Sheth, "Specification and Execution of Transactional Workflow," Modern Database Systems : The Object, Interoperability and Beyond, W. Kim(Ed.), Addison-Wesley, 1994.

[5] D. Georgakopoulos, M. F. Hornick and F. Manola, "Customizing Transaction Models and Mechanisms

in a Programmable Environment Supporting Reliable Workflow Automation," IEEE transaction on knowledge and data engineering, Vol.8, No.4, pp. 630-649, 1996.

[6] D. Hollingsworth, " Workflow Management Coalition - The Workflow Reference Model," TC00-1003 issue 1.1, 1994

[7] Work Group1, "Interface 1 : Process Definition Interchange Process Model," Workflow Management Coalition Specification, TC-1016, 1998.

[8] P. C. Attie, M. P. Singh, A. Sheth and M. Rusinkiewicz, "Specifying and Enforcing Intertask Dependency," VLDB conference, pp.134-145, 1993.

[9] 정희택, 이도헌, "시간제약을 포함한 워크플로우 모델링", 한국정보과학회 춘계학술대회, 제25권 제1호, pp. 101-103, April 1998.

[10] 정희택, 이도헌, 김문자, 류영철, "시간제약을 포함한 워크플로우 모델링 및 검증", 한국 정보처리학회 논문지, 제6권 제2호, 1999.

[11] James L. Peterson, "Petri net theory and the modeling of systems," Prentice-Hall, 1981

[12] W.M.P. van der Aalst,"Modelling and Analysis of production Systems Using a Petri Net Based Approach," Proc. of Conf. on Computer Integrated Manufacturing in the Process Industries, pp.179-193, USA, 1994.

[13] W.M.P. van der Aalst,"Three Good Reasons for Using a Petri-net-based Work flow Management System," Proc. of th Int. Working Conf. on Information and Process Integration in Enterprises, pp.179-201, Nov 1996.

[14] W.M.P. van der Aalst, "Petri-net-based Workflow Management Software," Proc. of the NFS workshop on Workflow and Process Automation In Information System, pp.114-118, May 1996.

[15] W.M.P. van der Aalst,"The Application of Petri Nets to Workflow Management," The Journal of Circuits, Systems and Computers, 1998.

[16] N. R. ADAM, Vijayalakshmi Atluri and Wei-Kuang HUANG, "Modeling and Analysis of workflows Using Petri Nets," Journal of Intelligent Information Systems, Vol.10, No.2, pp.1-29, March 1998.

정 희 택

e-mail : htceong@cs.yosu.ac.kr

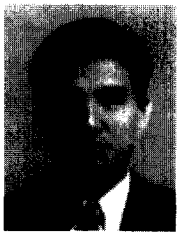
1992년 전남대학교 전산통계학과 (학사)

1995년 전남대학교 전산통계학과 (석사)

1999년 전남대학교 전산통계학과 (박사)

1999년~현재 여수대학교 전자계산학과 전임강사

관심분야 : 워크플로우 시스템, 분산처리 시스템, 데이터 웨어하우징, 데이터 마이닝



이 도 헌

e-mail : dhlee@dbcore.chonnam.ac.kr

1990년 한국과학기술원 전산학과 (석사)

1992년 한국과학기술원 전산학과 (석사)

1995년 한국과학기술원 전산학과 (박사)

1996년~현재 전남대학교 컴퓨터정보학부 조교수

관심분야 : 데이터마이닝, 워크플로우 시스템, 데이터 웨어하우징, 퍼지 데이터베이스