프로세스 마이닝에서의 효율적인 적합성 판단 기법*

김광복○, 허 신
한양대학교 컴퓨터공학과
towya81@hotmail.com, shinheu@hanyang.ac.kr

Efficient Method of "Conformance Checking" in Process Mining

Gwang Bok Kim○, Shin Heu
Dept. of Computer Science and Engineering, Hanyang University

요 약

BPMS, ERP, SCM 등 프로세스 인식 정보시스템들이 널리 쓰이게 되면서 프로세스 마이닝에 대한 연구가 활발하게 이루어지고 있다. 프로세스 마이닝은 프로세스가 실행되는 동안 저장된 이벤트 로그로부터 정보를 추출하는 기법이다. 추출된 로그정보는 비즈니스 프로세스의 분석 및 재설계에 사용될 프로세스 모델을 생성하게 된다. 프로세스 마이닝 기법은 프로세스의 자동화 및 기업의 업무정보들을 관리하는 프로세스 기반 정보시스템의 정확성 및 효율성을 위한 중요한 부분을 차지하지만 현재까지의 연구는 생성된 이벤트 로그로부터 프로세스 모델을 재설계하는 프로세스 발견 기법 (Process Discovery Technique)을 적용한 부분에서만 활발히 진행되었다. 프로세스 마이닝은 프로세스 발견 기법 외에도 프로세스 적합성 검사 기법 (Process Conformance Checking Technique) 및 프로세스 확장 기법 (Process Extension Technique)이 존재한다. 이들은 많은 프로세스 발견 기법에 대한 연구들이 진행되고 나서야 최근 프로세스 마이닝의 이슈로 떠오르고 있다. 본 논문에서는 프로세스 적합성 검사를 위해 수집된 이벤트 로그와 기존에 나와 있는 여러 가지 프로세스 발견 알고리즘을 통해 생성된 프로세스를 수치적으로 비교할 수 있는 두 가지 애트리뷰트를 제시하였다.

1. 서론

최근 정보기술(Information Technology)은 다양한 다른 사업에 적용되어 기업들의 효율성 및 생산성의 향상을 도모하는데 많은 기여를 하고 있다 [1]. 기업들은 기업 내부에서 프로세스 흐름의 자동화를 관리하기 위해 WFMS (Workflow Management System)에 많은 관심을 기울이고 있다. 이를 위해서는 해당 프로세스를 올바르게 설계해야할 필요가 있다. 하지만 워크플로우의 설계와 개발 과정이 분리되어있는 관계로 프로세스 설계자와 개발자 사이의 이해관계를 조정하는 것은 어려운 일이기 때문에 프로세스 설계 초기에 올바른 프로세스를 개발하기 쉽지 않다는 문제점이 발생한다 [1]. 이로 인해 프로세스의 설계가 제대로 이루어졌는지에 대한 관심이 높아졌고 이는 워크플로우의 재설계(Workflow Re-engineering)를 기업의 프로세스 관리에 중요한 요소로 부각되는 것을 초래하였다. 이와 같은 워크플로우의 재설계는 프로세스 마이닝(Process Mining)을 통해 이루어지며, 이에 프로세스 마이닝이 워크프로우의 재설계를 위한 기법으로 많은 주목을 받고 있다. 프로세스 마이닝의 목적은 실제 프로세스가 수행됨으로써 발생하는 이벤트 로그로부터 프로세스에 관한 정보를 얻어 이를 토대로 완성된 형태의 프로세스 모델을 재 생성하는데 있다 [2]. 워크플로우 상의 프로세스 마이닝에서 프로세스를 관리하는 기법은 크게 3가지로 나뉜다. 첫 번째로 정보시스템의 실행 결과가 기록된 프로세스의 이벤트 로그 정보를 이용하여 프로세스 모델을 찾아내는 프로세스 발견 (Process Discovery)기법이 있고, 두 번째로 발견된 프로세스 모델과 이벤트 로그 정보를 비교하여 해당 프로세스가 기업에서 의도한 목적에 적합한지는 평가하는 프로세스 적합성 검사 (Process Conformance Checking)기법, 마지막으로 발견된 프로세스 모델에 적절한 수정을 가하여 적합성을 높이는 프로세스 확장 (Process Extension)기법이 있다 [2]. 프로세스 마이닝에 대한 연구는 지속적으로 활발히 진행되고 있지만 이의 대부분이 프로세스 발견 기법에 관한 연구였다 [3][4][5][6][7]. 이는 프로세스 관리 기법에서 프로세스 발견 기법이 가장 기본이 되기 때문이었는데 이에 관한 많은 프로세스 발견 기법에 대한 연구가 진행된 최근에서야 프로세스 적합성 검사 기법 및 프로세스 확장 기법이 이슈가 되고 있다 [8]. 본 논문에서는 이벤트 로그로부터 생성된 관계행렬 (Relation Matrix)과 발견된 프로세스로부터 생성된 관계행렬을 비교하는 방법과 해당 이벤트 로그를 발견된 프로세스에 재실행 (Replay)시켜 토큰(Token)의 흐름을 통한 비교 기법을 사용하는 프로세스 적합성 검사 방법을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 프로세스 마이닝 기법과 연구 동향, 적합성을 판정하기 위한 애트리뷰트에 쓰인 관계행렬 (Relation Matrix)에 대해 설명한다. 3장에서는 논문에서 제시한 두 가지의 애트리뷰트 (Attribute)를 사용하여 적합성 정도를 수치적으로 나타내는 방법을 기술한다. 4장에서는 실제로 이벤트 로그와 발견된 프로세스 모델 간의 적합성을 판단하는 예

* 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. R01-2007-000-20135-0)

* 본 연구는 지식경제부 및 정보통신 연구 진흥원의 대학 IT연구센터 육성·지원사업 (IITA-2009-c1090-0902-0031)의 연구결과로 수행되었음

제를 제시한다. 마지막 5장에서는 논문의 결론을 맺는다.

2. 관련연구

2.1 프로세스 마이닝 (Process Mining)

프로세스 마이닝 (Process Mining)은 광범위한 정보 시스템에 적용될 수 있다. SAP와 같은 ERP systems, FLOWer와 같은 case handling systems, Windchill과 같은 PDM systems, Microsoft Dynamics CRM과 같은 CRM systems 등의 이벤트 로그를 생성하는 많은 프로세스 인식 정보 시스템 (Process-Aware Information Systems)들이 존재한다 [2]. 이와 같은 시스템들은 액티비티 (Activity)들이 수행된 정보들을 로그 데이터로 기록하는 역할을 수행한다. 프로세스 마이닝의 목적은 이와 같이 수집된 로그 데이터로부터 프로세스 모델을 추출하는데 있다. 이와 같이 프로세스 마이닝을 하기위한 가장 기본적인 요소는 실행된 프로세스로부터 기록된 이벤트 로그인데 현재 프로세스 마이닝 기법을 적용하기 위해서는 수집된 이벤트로그는 이벤트가 일어난 순서대로 기록되었고 각 case별로 저장되었다는 가정을 전제조건으로 한다 [2]. 프로세스 마이닝의 각 기법은 다음과 같다. 첫 번째로 이벤트 로그로부터 새로운 프로세스를 찾아내는 프로세스 발견 (Process Discovery) 기법이 있고, 두 번째로 이벤트 로그와 발견된 프로세스 모델간의 적합성을 비교하는 프로세스 적합성 판단 (Process Conformance Checking) 기법이 있다. 마지막으로 발견된 프로세스 모델에 수정을 가하여 적합성을 높이는 프로세스 확장 (Process Extension) 기법이 있다 [2].

2.2 관계행렬 (Relation Matrix)

관계행렬 (Relation Matrix)은 본 논문에서 제시하는 애트리뷰트인 CCA1에서 적합성의 정도를 수치로 표현하기 위해 사용되는 각 case별로 수행된 작업 (Task)간의 인과관계를 나타내고 쉽게 계산하여 유지하기 위한 행렬이다. (그림 1)과 같이 프로세스로부터 수집된 이벤트 로그를 통해 case 별 관계를 표현하고 이를 통해 관계행렬을 나타낼 수 있으며, 이항관계 (Binary Relation)에 있는 두 작업간의 인과관계를 표현한다. 이 관계는 0 또는 1로 표현이 되며 두 작업 간에 인과관계가 존재하면 1로 표기하고 존재하지 않으면 0으로 표기하여 관계행렬을 유지한다.

Case	Sequence	Frequency
1	T1->T2->T3->T4	53
2	T1->T3->T2->T4	437
3	T1->T2->T3->T4	338
4	T1->T3->T2->T4	49
5	T5->T6	123

(a) Event Log Sequence Data

	T1	T2	T3	T4	T5	T6
T1	0	1	1	0	0	0
T2	0	0	1	1	0	0
T3	0	1	0	1	0	0
T4	0	0	0	0	1	0
T5	0	0	0	0	0	1
T6	0	0	0	0	0	0

(b) 관계행렬

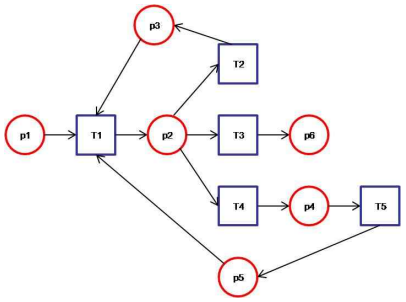
(그림 1) 이벤트 로그 시퀀스 및 이에 따른 관계행렬

3. 적합성 판정 기법

본 논문에서는 수집된 이벤트 로그와 발견된 프로세스 모델간의 적합성을 판정하기 위하여 두 가지의 프로세스 발견 알고리즘과 두 가지 애트리뷰트를 사용한다. 논문에서 사용된 알고리즘은 Alpha++ Algorithm, Heuristic Algorithm이다 [3][4]. 또한 논문에서 적합성 판정을 위해 사용된 애트리뷰트는 다음과 같다. 첫 번째는 CCA1 (Conformance Checking Attribute 1)로 이벤트 로그로부터 생성된 관계행렬과 발견된 프로세스 모델로부터 생성된 관계행렬의 비교를 통하여 둘 사이의 적합성 정도를 0부터 1사이의 값으로 나타낸다. 두 번째는 CCA2 (Conformance Checking Attribute 2)로 발견된 프로세스 모델에 이벤트 로그를 재실행 (Replay)시켜 토큰의 흐름에 따라 발생하는 여분의 토큰과 잃어버린 토큰 및 생성된 토큰, 소모된 토큰의 수를 파악한다. 이를 통해 숨겨진 작업 (Hidden Task)의 유무와 이벤트 로그가 정확히 기록되었는지의 여부를 판단하여 해당 이벤트 로그와 발견된 프로세스 모델간의 적합성 정도를 첫 번째 애트리뷰트와 마찬가지로 0부터 1사이의 값으로 나타낸다. 따라서 3.1절에서는 두 가지 애트리뷰트를 정의할 때 사용할 프로세스 모델에 대해 알아보고 3.2절에서는 프로세스 모델로부터 추출한 이벤트 로그 및 관계 행렬에 대해 살펴본다. 3.3절에서는 프로세스 발견 기법 알고리즘을 이용하여 재 생성된 프로세스 및 그에 따른 이벤트 로그 와 관계행렬에 대해 서술한다. 3.4절과 3.5절에서는 각각 CCA1과 CCA2를 정의한다.

3.1 프로세스 모델 (Process Model)

본 절에서는 프로세스 적합성 판단을 위한 첫 번째 애트리뷰트인 CCA1에 대한 설명을 하기 위해 (그림 2)와 같은 프로세스 모델을 사용한다. (그림 2)에서 보이는 프로세스의 두 place인 p1과 p6은 각각 시작 이벤트 (Start Event)와 종료 이벤트 (End Event)를 나타낸다. 시작 이벤트인 p1에서 토큰 (Token)이 생성되면 T1이 수행되고 T1에서 부터의 place인 p2는 토큰을 생성해 내고 T2와 T3, T4 중 하나가 실행된다. T3가 수행된 경우 종료이벤트인 p6을 통해 프로세스는 종료된다. T2가 실행된 경우 p3에서 토큰을 생성하게 되고 다시 T1로 돌아가서 p2에서 토큰을 생성한다.



(그림 2) 프로세스 모델 예제

T4가 실행된 경우 p4에서 토큰을 생성하여 T5를 수행

시킨 후 마찬가지로 T1로 돌아가 루프구조 (Loop Structure)를 형성하게 된다.

3.2 이벤트 로그 및 관계 행렬

3.1절의 프로세스 모델을 실행하여 만들어진 이벤트 로그는 (표 1)과 같다. 본 논문에서 사용하는 이벤트 로그는 하나의 인스턴스 (Instance)가 프로세스의 처음부터 끝까지 수행된 트랜잭션 (Transaction)들을 나타내는 Case ID와 수행된 작업을 순차적으로 나열한 Sequence, 그리고 해당 케이스가 몇 번 수행되었는지를 나타내는 Frequency로 구성된다. CCA1에서는 Frequency가 사용되지 않지만 3.4절에서 다루게 될 애트리뷰트인 CCA2를 정의하는데 Frequency를 사용한다.

(표 1) Event Log Sequence Data

Case	Sequence	Frequency
1	T1->T3	547
2	T1->T2->T1->T3	224
3	T1->T2->T1->T2->T1->T3	141
4	T1->T5->T1->T3	49
5	T1->T4->T5->T1->T2->T1->T3	27
6	T1->T4->T5->T1->T2->T1->T2->T1->T3	12

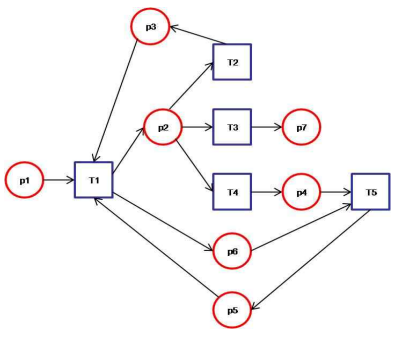
	T1	T2	T3	T4	T5
T1	0	1	1	1	1
T2	1	0	0	0	0
T3	0	1	0	0	0
T4	0	0	0	0	1
T5	1	0	0	0	0

(그림 3) 관계행렬

프로세스로부터 수집된 이벤트 로그를 통해 case 별 관계를 표현하고 이를 통해 나타낼 수 있는 관계행렬은 이항 관계 (Binary Relation)에 있는 두 작업간의 인과관계를 표현한다. 이 관계는 0 또는 1로 표현이 되며 두 작업 간에 인과관계가 존재하면 1로 표기하고 존재하지 않으면 0으로 표기한다. 3.1절의 (그림 2)에 나타난 프로세스로부터의 이벤트 로그와 이로부터 만들어진 관계행렬은 (그림 3)과 같다.

3.3 발견된 프로세스 및 이벤트 로그와 관계행렬

3.4절에서 설명할 CCA1은 본 논문에서 사용하는 2가지 프로세스 발견 알고리즘 중 Heuristic Algorithm을 통해 재 생성된 프로세스를 통해 설명한다. 이에 따라, 3.2절에서 얻은 이벤트 로그로부터 Heuristic Algorithm을 이용하여 발견된 프로세스는 (그림 4)와 같고, 그에 따른 이벤트 로그와 관계 행렬은 (그림 5)와 같다.



(그림 4) Heuristic Algorithm으로 재 생성된 프로세스

Case	Sequence	Frequency
1	T1->T3	
2	T1->T2->T1->T3	
3	T1->T2->T1->T2->T1->T3	
4	T1->T5->T1->T3	
5	T1->T4->T5->T1->T2->T1->T3	
6	T1->T4->T5->T1->T2->T1->T2->T1->T3	

(a) Event Log Sequence Data

	T1	T2	T3	T4	T5
T1	0	1	1	1	1
T2	1	0	0	0	0
T3	0	1	0	0	0
T4	0	0	0	0	1
T5	1	0	0	0	0

(b) 관계행렬

(그림 5) 발견된 프로세스로 부터의 이벤트 로그 시퀀스 데이터 및 관계행렬

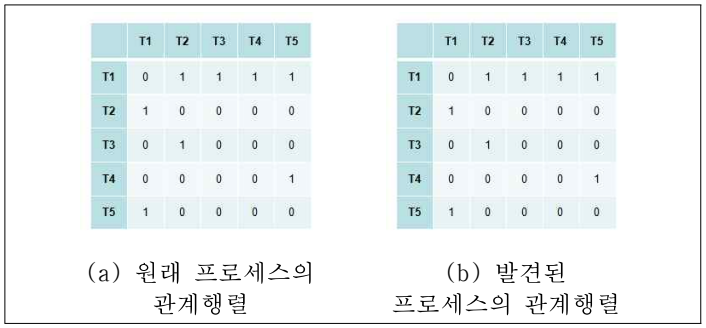
본 절의 이벤트 로그는 발견된 프로세스를 가상적으로 실행 시켜서 생성된 이벤트 로그를 임의로 사용한다.

3.4 CCA1

프로세스 마이닝 기법을 적용하여 이벤트 로그로부터 프로세스를 재 생성 할 때 가장 중요한 것은 발견된 프로세스가 얼마나 이벤트 로그의 정보를 잘 표현할 수 있는지의 여부이다. 프로세스 마이닝 기법을 적용할 때에 이벤트 로그 정보는 작업 (Task)이 수행된 순서대로 기록되어야 한다고 2.3절에서 언급하였다. 이와 같이 작업이 수행된 순서대로 로그 데이터가 기록된다는 것에 착안하여 CCA1은 발견된 프로세스의 가상 실행을 통해 임의의 이벤트 로그 데이터를 추출하여 원래의 프로세스 모델로부터의 이벤트 로그 데이터와 비교하여 그 수치를 계산한다. 3.2절 및 3.3절에서 구한 관계행렬은 (그림 6)에서 비교할 수 있다. 행렬 값의 차이점은 두 관계 행렬의 같은 열, 같은 행의 행렬 값이 같은지 틀린지 여부를 통해 구할 수 있고, 이를 통해 CCA1을 표현할 수 있는데 이는 관계 행렬의 사이즈로 두 행렬 사이의 행렬 값이 다른 것의 개수를 나눈 것이다. 적합성이 높은 것을 1로 나타내기 위해 1에서 계산된 결과를 뺀 값을 CCA1로 정의한다.

정의 1) CCA1 = 1 − (Number of Differences / Size of Matrix)

(그림 6)의 두 행렬간의 CCA1 값을 계산하게 되면 1이 나온다. 이는 CCA1로 적합성을 판단하였을 때 가장 적합한 수치를 보이지만, (그림 2)와 (그림 4)에서 보이는 프로세스는 동일한 역할을 수행하지 않는다. 로그 데이터가 발생한 순서대로 기록되는 특성을 이용하여 이벤트 로그와 발견된 프로세스 모델 간의 적합성 정도를 CCA1을 통해 계산하였지만 CCA1의 경우에 숨겨진 작업 (Hidden Task)이 있는지 여부 혹은 특정 작업이 수행되었는지 여부, 그리고 이벤트 로그가 잘못 기록되었는지 여부를 알 수 없어 적합성 판단이 불확실해 질 수 있다. 3.5절에서는 이와 같은 문제점도 해결 가능하도록 또 다른 애트리뷰트인 CCA2에 대하여 설명한다. 또한 본 절에서 예로 들은 프로세스 모델이 비교적 단순하여 두 관계 행렬의 차이 정도가 미세하지만 4장에서 예를 들어 설명할 프로세스 모델의 경우 비교적 복잡하여 관계행렬간의 차이를 좀 더 명확하게 관찰할 수 있다.



	T1	T2	T3	T4	T5
T1	0	1	1	1	1
T2	1	0	0	0	0
T3	0	1	0	0	0
T4	0	0	0	0	1
T5	1	0	0	0	0

(a) 원래 프로세스의 관계행렬

	T1	T2	T3	T4	T5
T1	0	1	1	1	1
T2	1	0	0	0	0
T3	0	1	0	0	0
T4	0	0	0	0	1
T5	1	0	0	0	0

(b) 발견된 프로세스의 관계행렬

(그림 6) 원래 프로세스와 발견된 프로세스로 부터의 관계행렬

3.5 CCA2

3.4절에서 제시한 CCA1만 적합성 판정에 사용할 경우 위에서 언급한 것처럼 적합성을 판단하는 것이 불확실해 질 수 있다. 페트리 넷으로 표기된 프로세스 모델은 토큰의 흐름으로 그 실행을 표현할 수 있다. 프로세스 실행 중 생성된 토큰이 프로세스가 종료되고 나서도 프로세스 내부에 남아있다는 것은 프로세스 내부의 작업이 수행되지 않았거나 이벤트 로그의 저장에 문제가 있다고 판단할 수 있다. 이에 착안된 두 번째 애트리뷰트인 CCA2는 토큰의 흐름을 통해 이벤트 로그와 발견된 프로세스의 적합도 수치를 계산한다. 토큰의 흐름을 통한 이벤트 로그와 발견된 프로세스 사이의 적합도 수치를 계산하기 위해 발견된 프로세스에 이벤트 로그를 재실행 (Replay)시키는 작업이 필요하다. CCA2 값의 정의를 위해 (그림 7)과 같이 표현식에 사용된 인자들을 정의한다.

```
i : 각 케이스별 인식 ID
R : 여분의 토큰의 개수
P : 생성된 토큰의 개수
M : 잃어버린 토큰의 개수
C : 소모된 토큰의 개수
N : 각 케이스별 생성된 인스턴스의 수
```

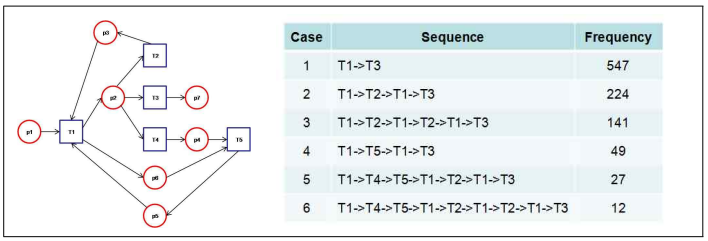
(그림 7) 표현식에 사용된 인자들

i는 각 케이스별 인식 ID로 각 case 번호를 의미한다. R은 남은 토큰의 개수로 프로세스의 실행이 끝난 후에 프로세스에서 생성되어 소모되지 못한 토큰들을 의미한다. P는 프로세스의 실행 중 생성된 토큰의 개수를 나타내고 M은 프로세스의 실행 중 생성되어 소모되지 못하고 있는 토큰의 개수를 나타낸다. M값과 R값의 차이는 프로세스의 실행이 끝난 후의 값인지 실행 중의 값인지의 차이이다. C값은 프로세스의 실행 중 소모된 토큰의 개수를 나타내고 N은 각 케이스 별로 생성된 인스턴스의 수를 의미한다. 프로세스 적합성 판정을 위한 두 번째 애트리뷰트인 CCA2는 각 case별 인스턴스의 수와 생성된 토큰의 수를 곱한 후 더한 값으로 각 case별 인스턴스의 수와 남은 토큰의 수를 곱한 후 더한 값을 나눈 값과 각 case별 인스턴스의 수와 소모된 토큰의 수를 곱한 후 더한 값으로 각 case별 인스턴스의 수와 사라진 토큰의 수를 곱한 후 더한 값을 나눈 값의 합이다. 이 또한 CCA1과 마찬가지로 적합성이 높은 것을 1로 나타내기 위해 1에서 계산된 결과를 뺀 값을 CCA2로 정의한다.

*정의 2) CCA2 = 1 − ((Σ(Ri*Ni) / (Σ(Pi*Ni) + (Σ(Mi*Ni) / (Σ(Ci*Ni))*

※ i = 1 to number of cases

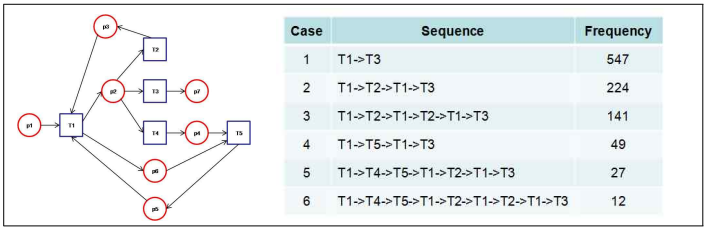
(그림 8)에서 재실행 과정을 살펴보기 쉽게 하기 위해 프로세스 모델과 해당 이벤트 로그 시퀀스 및 인자들의 값을 하나로 묶어 놓았다. 첫 번째로 일반적인 경우를 살펴보기 위해 Case2의 시퀀스를 재실행 시켜보았다. 이벤트 로그정보로 부터 기본적으로 i = 2, N = 224의 값을 갖는다. 프로세스가 재실행되어 먼저 p1에서 토큰이 생성되면 T1이 실행되고 P와 C의 값이 각각 1씩 증가한다. 두 번째로 p2에서 생성된 토큰은 T2를 실행시키고 역시 P와 C의 값이 1씩 증가한다. 이어서 p3에서 생성된 토큰은 다시 T1을 실행시키고 다시 P와 C의 값은 1씩 증가한다. 마지막으로 p2에서 토큰이 생성되어 T3를 수행하고 종료 이벤트인 p7에서 프로세스를 종료한다. (그림 8)의 예제는 사라진 토큰이나 남는 토큰이 발생하지 않았다.



Case	Sequence	Frequency
1	T1->T3	547
2	T1->T2->T1->T3	224
3	T1->T2->T1->T2->T1->T3	141
4	T1->T5->T1->T3	49
5	T1->T4->T5->T1->T2->T1->T3	27
6	T1->T4->T5->T1->T2->T1->T2->T1->T3	12

(그림 8) 이벤트 로그의 재실행 (Case 2)

반면에 (그림 9)은 이벤트 로그의 case 4를 재실행하는 과정을 살펴보기 위한 것이다. 이 또한 이벤트 로그정보로부터 기본적으로 i = 4, N = 49의 값을 갖는다. 프로세스가 재실행되면 위의 경우와 같이 시작 이벤트인 p1에서 토큰이 생성되고 T1이 실행된다. 마찬가지로 P와 C의 값이 각각 1씩 증가하고, p2와 p6에서 토큰이 하나씩 생성된다. 이후 T5가 수행이 되는데 여기서 p2에서 생성된 토큰이 소모되지 않고 사라진 것을 볼 수 있다.



Case	Sequence	Frequency
1	T1->T3	547
2	T1->T2->T1->T3	224
3	T1->T2->T1->T2->T1->T3	141
4	T1->T5->T1->T3	49
5	T1->T4->T5->T1->T2->T1->T3	27
6	T1->T4->T5->T1->T2->T1->T2->T1->T3	12

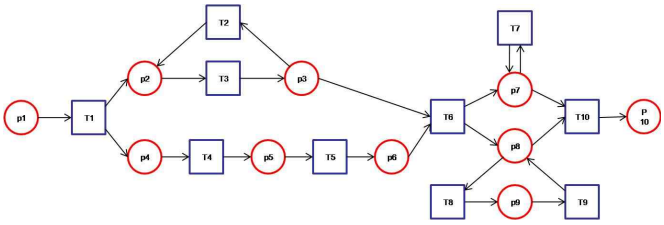
(그림 9) 이벤트 로그의 재실행 (Case 4)

이에 따라 P의 값은 2가 증가하고 C의 값은 1, 그리고 M의 값이 1 증가 하게 된다. 이후 p5에서 생성된 토큰은 T1을 실행시키고 P와 C의 값은 1씩 증가한다. 다시 p2와 p6에서 토큰이 생성되고 T3가 실행되어 p7에서 프로세스가 종료한다. 여기에서도 p6에서 생성된 토큰이 소모되지 않고 사라진 것을 발견할 수 있다. 이에 따라 역시 P의 값도 2, C의 값도 1, M의 값도 1씩 증가한다. 프로세스가

종료함에도 불구하고 사라진 토큰이 존재하는 경우 남은 토큰으로 처리한다. 따라서 R의 값에 M의 값을 대입하여 M의 값을 2 만큼 증가시킨다. case 4의 경우 프로세스의 종료 후 사라진 토큰과 남은 토큰이 0보다 크므로 이벤트 로그 혹은 프로세스 내에 이상이 있다는 것을 알 수 있다. 이와 같이 이벤트 로그가 발견된 프로세스에서 재실행되는 동안 토큰의 흐름을 이용하여 숨겨진 작업의 유무나 특정 작업이 수행되었는지 여부 혹은 이벤트 로그가 잘못 기록된 여부를 확인할 수 있다. 정의 2)에 따라 계산된 (그림 4)의 프로세스와 이벤트 로그 사이의 CCA2 값은 0.308이다. 이는 정의 1)에 따라 적합하다고 판단된 프로세스가 가장 적합한 프로세스가 아니라는 사실을 보여준다.

4. 적합성 판정 기법 예제

본 장에서는 3장에서 예로 들었던 로그 데이터와 발견된 프로세스를 이용하여 둘 사이의 적합성을 판단하여 가장 적합한 프로세스 모델들의 집합을 찾아내는 과정을 설명한다. 이에 따라 프로세스 적합성 판단 기법을 적용시키기 위한 프로세스는 (그림 10)과 같고 이를 통해 추출된 이벤트 로그 시퀀스는 (표 2)과 같다.

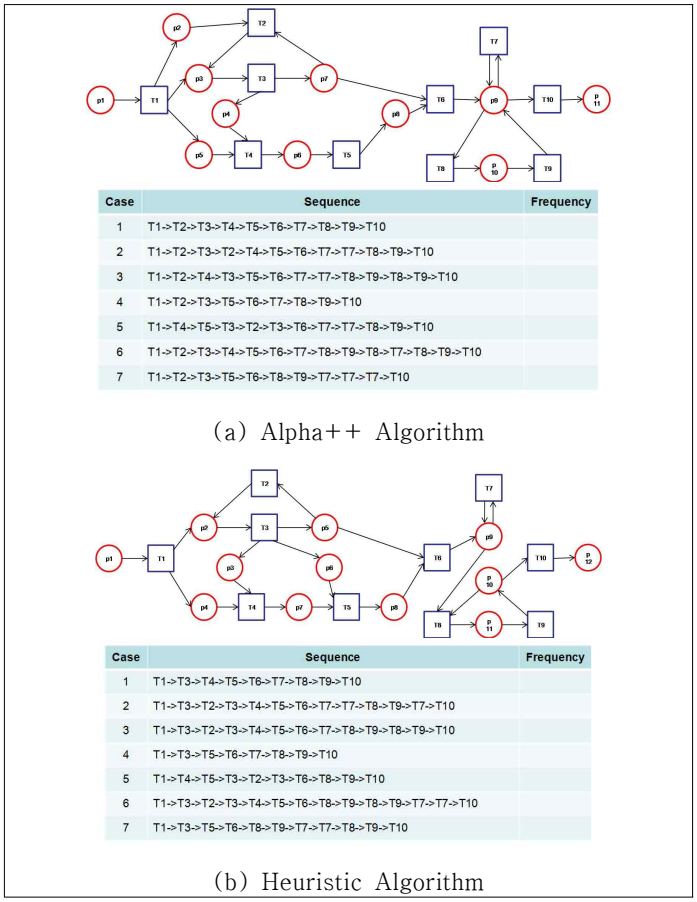


(그림 10) 예제 프로세스 모델

(표 2) 프로세스 모델로부터 추출된 이벤트 로그 시퀀스

Case	Sequence	Frequency
1	T1->T3->T4->T5->T6->T7->T8->T9->T10	53
2	T1->T3->T2->T3->T4->T5->T6->T7->T7->T8->T9->T10	359
3	T1->T3->T2->T3->T4->T5->T6->T7->T7->T8->T9->T8->T9->T10	287
4	T1->T3->T5->T6->T7->T8->T9->T10	49
5	T1->T4->T5->T3->T2->T3->T6->T7->T7->T8->T9->T10	123
6	T1->T3->T2->T3->T4->T5->T6->T8->T9->T8->T9->T7->T7->T10	78
7	T1->T3->T5->T6->T8->T9->T7->T7->T7->T10	51

추출된 이벤트 로그로 속성 값을 다르게 한 2가지의 프로세스 발견 기법 알고리즘을 통하여 10개의 프로세스 모델을 재 생성하였다. (그림 11)는 생성된 프로세스 모델들이 많은 관계로 각 알고리즘 당 하나씩의 재 생성된 프로세스 모델 및 이벤트 로그 시퀀스만을 나타내었다.



Case	Sequence	Frequency
1	T1->T2->T3->T4->T5->T6->T7->T8->T9->T10	
2	T1->T2->T3->T2->T4->T5->T6->T7->T7->T8->T9->T10	
3	T1->T2->T4->T3->T5->T6->T7->T7->T8->T9->T8->T9->T10	
4	T1->T2->T3->T5->T6->T7->T8->T9->T10	
5	T1->T4->T5->T3->T2->T3->T6->T7->T7->T8->T9->T10	
6	T1->T2->T3->T4->T5->T6->T7->T8->T9->T8->T9->T7->T8->T9->T10	
7	T1->T2->T3->T5->T6->T8->T9->T7->T7->T7->T10	

(a) Alpha++ Algorithm

Case	Sequence	Frequency
1	T1->T3->T4->T5->T6->T7->T8->T9->T10	
2	T1->T3->T2->T3->T4->T5->T6->T7->T7->T8->T9->T7->T10	
3	T1->T3->T2->T3->T4->T5->T6->T7->T8->T9->T8->T9->T10	
4	T1->T3->T5->T6->T7->T8->T9->T10	
5	T1->T4->T5->T3->T2->T3->T6->T8->T9->T10	
6	T1->T3->T2->T3->T4->T5->T6->T8->T9->T8->T9->T7->T7->T10	
7	T1->T3->T5->T6->T8->T9->T7->T7->T8->T9->T10	

(b) Heuristic Algorithm

(그림 10) 각 알고리즘으로부터 발견된 프로세스 모델 및 추출한 이벤트 로그 시퀀스 정보

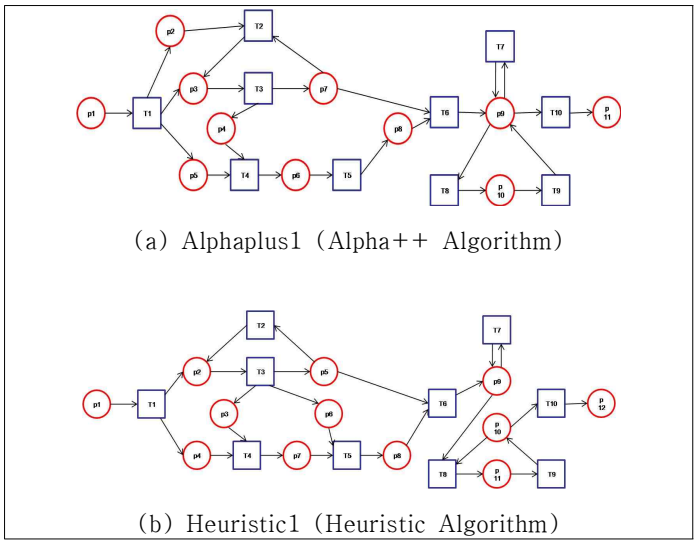
발견된 모든 프로세스와 이벤트 로그간의 적합성 판단을 위해 (표 2)의 이벤트 로그로부터 만들어진 관계행렬과 (그림 11)의 이벤트 로그로부터 만들어진 관계행렬의 비교를 통하여 우선 CCA1의 값을 계산한다. 두 번째로 CCA2의 값을 구하기 위해 (표 2)의 이벤트 로그를 (그림 11)의 프로세스 모델에서 재실행하여 토큰의 흐름을 파악하고 각 토큰들의 수를 정의 2)에 대입하여 CCA2의 값을 계산한다. 프로세스 적합성 판단을 위한 애트리뷰트 CCA1과 CCA2값을 포함한 프로세스 모델의 리스트는 (표 3)과 같다.

(표 3) CCA1과 CCA2의 값을 포함한 프로세스 모델의 리스트

Alpha++ Algorithm	Heuristic Algorithm
AlphaPlus1 (0.96, 0.989)	Heuristic1 (0.99, 0.970)
AlphaPlus2 (0.95, 0.971)	Heuristic2 (0.98, 0.907)
AlphaPlus3 (0.95, 0.916)	Heuristic3 (0.99, 0.915)
AlphaPlus4 (0.92, 0.864)	Heuristic4 (0.98, 0.931)
AlphaPlus5 (0.94, 0.752)	Heuristic5 (0.98, 0.886)

각 알고리즘의 첫 번째 프로세스 모델은 프로세스 마이닝 프로그램에서 기본적으로 세팅되어있는 속성 값을 통하여 재 생성 하였고, 나머지 4가지의 서로 다른 프로세스를 속성 값을 변경하며 재 생성 하였다. (표 3)의 결과를 보면 흥미로운 점이 발견된다. Alpha++ Algorithm에

비해 Heuristic Algorithm이 적합도 수치가 높은 것을 살펴볼 수 있다. 이는 Alpha++ Algorithm에 비해 Heuristic Algorithm이 루프구조를 포함한 프로세스를 마이닝 하기에 적합한 특징을 갖고 있기 때문이다. 또 다른 주목할만한 점은 CCA1의 값은 대부분 높게 측정되었으나 CCA2의 값은 각 알고리즘을 통해 발견된 프로세스 마다 차이를 보였다. 이 프로세스 모델의 리스트에서 각 알고리즘 별로 CCA1값과 CCA2값을 더한 값이 가장 큰 것은 AlphaPlus1과 Heuristic1이다. 그리고 이 두 값 중 더 큰 값을 가지고 있는 Heuristic1이 해당 프로세스 모델을 마이닝 하기에 가장 적합한 알고리즘으로 볼 수 있다. 이 두 값에 관계 된 프로세스 모델들은 (그림 12)과 같고 원래의 프로세스 모델인 (그림 8)과 비교하였을 때 직관적으로도 적합도가 높고 둘 중에서 Heuristic1이 더 적합하다는 것을 살펴볼 수 있다.



(a) Alphaplus1 (Alpha++ Algorithm)

(b) Heuristic1 (Heuristic Algorithm)

(그림 12) 페트리넷으로 표기된 적합성이 가장 높은 프로세스들

5. 결론

본 논문에서는 관계 행렬을 이용한 애트리뷰트와 토큰의 흐름을 나타내는 애트리뷰트 두 가지를 사용하여 그 값을 계산하고 계산된 값의 합을 통해 이벤트로그와 발견된 프로세스 모델간의 적합도가 가장 높은 것을 구하는 방법을 제안하였다. 이는 프로세스 발견 기법에 대한 많은 연구가 진행되었지만 특정 프로세스 모델 혹은 로그 데이터에 대해 완벽한 프로세스 모델을 100% 재 생성할 수 없다는 것에 기반 하였다. 실질적으로 로그 데이터로부터 원래의 프로세스 모델과 100% 일치하는 프로세스 모델을 발견하는 것은 매우 어려운 일이기 때문에 프로세스 관리 기법의 그 두 번째, 세 번째인 프로세스 적합성 검사 기법과 프로세스 확장 기법에 대한 연구가 지속적으로 진행되어야 한다 [2][8]. 본 논문에서는 CCA1과 CCA2값을 이용하여 가장 최적한 모델을 검출해낸다. 하지만 두 가지 애트리뷰트로는 검출된 프로세스 모델이 가장 적합하다는 것을 완벽하게 보여줄 수 없다. 또한 검출된 한 가지의 모델로서 판단하기 보다는 가장 적합한 프로세스 모델의 집합을 구하여 집합 내의 프로세스 모델들을 수정하여 가장 적합성이 높은 모델을 찾아낼 수 있는 프로세스 확장 기법을 적용할 필요성이 있다. 이와 같은 기법을 적용하기 위해서는 이와 같은 과제들에 대한 학계의 지속적인 연구가 필요하다고 사료된다.

향후 과제로는 좀 더 명확한 적합도 판정을 위한 애트리뷰트의 추가 및 가장 적합도가 높은 집합을 구하기 위해 프로세스 모델 검출 시 스카이라인 알고리즘을 적용할 예정이다.

6. 참고문헌

[1] 김재형, 김광복, 김건우, 이정화, 손진현, "MFP-트리 기반의 마이닝" 2008 한국정보과학회, 35, 2, 25, 2008

[2] W.M.P. van der Aalst and A.J.M.M. Weijters. Process Mining: A Research Agenda. Computers in Industry, 53(3):231-244, 2004.

[3] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1128-1142, 2004.

[4] L. Wen, J. Wang, and J.G. Sun. Detecting Implicit Dependencies Between Tasks from Event Logs. In Asia-Pacific Web Conference on Frontiers of WWW Research and Development (APWeb 2006), Lecture Notes in Computer Science, pages 591-603. Springer, 2006. - alpha plus

[5] A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering, 10(2):151-162, 2003. - heuristic

[6] W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. G¨unther. Process Mining: A Two-Step Approach using Transition Systems and Regions. BPM Center Report BPM-06-30, BPMcenter.org, 2006.

[7] A.K. Alves de Medeiros. Genetic Process Mining. PhD thesis, Eindhoven University of Technology, Eindhoven, 2006.

[8] M. La Rosa, F. Gottschalk, M. Dumas, and W.M.P. van der Aalst. Linking Domain Models and Process Models for Reference Model Configuration. In A. ter Hofstede, B. Benatallah, and H.Y. Paik, editors, BPM 2007 International Workshops (BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws), volume 4928 of Lecture Notes in Computer Science, pages 417-430. Springer-Verlag, Berlin, 2008.