CTR을 이용한 자원 할당 제약조건 하에서 워크플로우의 스케줄링

고 재 진[†] · 안 형 근[††] · 이 단 영[†††]

요 약

다양한 환경 속에서 기업이 업무의 효율성과 효과적인 업무처리를 해결하기 위해서 핵심정보시스템으로 워크플로우를 도입하고 있다. 최근에는 워크플로우 확장으로 비즈니스 프로세스 관리(BPM : Business Process Management)가 기업 소프트웨어 시장의 가장 중요한 부분을 차지하고 있다. 비즈니스 프로세스 관리에서 워크플로우는 핵심적인 역할을 수행하고 있지만, 워크플로우 연구의 대부분은 복잡한 업무의 정확한 실행 순서를 찾기 위한 시간적인 제약들에 집중되어 있다. 그런 이유에서 워크플로우 스케줄링의 대부분은 처리해야 할 다양한 자원의 모델링에 집중되었으며, 자원과 연관된 제약조건에서의 워크플로우 스케줄링에 대한 관심은 미흡한 편이다.

본 논문에서는 자원할당 제약조건에 따른 워크플로우 스케줄링에 적합한 CTR(Concurrent Transaction Logic) 기반의 변형 템플릿(Transformation Template)과 논리적 프레임워크를 제안한다. 변형 템플릿은 워크플로우 변형 템플릿(Workflow Transformation Template)과 제약조건 변형 템플릿(Constraint Transformation Template)으로 구성된다. 이 변형 템플릿은 기존의 워크플로우를 자원할당 제약조건 하에서 새로운 워크플로우를 논리적으로 표현하는데 용이하게 사용할 수가 있다.

키워드 : 워크플로우, 스케줄링, 제약조건, 자원할당

Scheduling of Workflows under Resource Allocation Constraints using CTR

Koh Jae Jin[†] · An Hyoung Keun[††] · Lee Dan Young[†††]

ABSTRACT

Many enterprises have introduced workflow to enhance work efficiency and support effective work processes in their various work environments. Recently, Business Process Management(BPM), an extension of workflow, is spotlighted in enterprise software markets. Although workflow plays an important role in Business Process Managements, researches in workflow are mostly concentrated on temporal constraints which finds exact execution sequences for complicated jobs. On that reason, majority of workflow scheduling has concentrated on modeling of various resources which should be handled and the interest for workflow scheduling under constraints related to resources is rather unsatisfactory.

In this paper we presents the Transformation Template based on Concurrent Transaction Logic(CTR) which is suitable for scheduling workflows with resource allocation constraints, and the logical framework. The Transformation Template consists of a Workflow Transformation Template and a Constraint Transaction Template. Especially the Transformation Template can be conveniently used to logically represent new workflows under the existing resource allocation constraints.

Key Words : Workflow, Scheduling, Constraint, Resource Allocation

1. 서 론

최근 수많은 정보를 통해서 증명 되듯이 비즈니스 프로세스 관리는 기업의 소프트웨어 시장의 가장 중요한 부분을 차지하고 있다. 비즈니스 프로세스 관리는 확장된 기업의 프로세스를 자동화하고 통합하고 최적화하기 위해 설계된 새로운 정보 기술이라고 할 수 있다. 비즈니스 프로세스를 체계적으로 관리하는 통합관리 방법론인 비즈니스 프로세스 관리에서 핵심적인 정보시스템 역할을 수행하는 것은 워크플로우로 알려져 있다. 워크플로우는 비즈니스 프로세스를

※ 본 연구는 2004학년도 울산대학교 학술연구 지원으로 수행되었음.
† 정 회 원 : 울산대학교 컴퓨터정보통신공학부
†† 정 회 원 : 울산대학교 컴퓨터정보통신공학부
††† 정 회 원 : 울산대학교 컴퓨터정보통신공학부
논문접수 : 2005년 9월 1일, 심사완료 : 2006년 2월 13일

모델링하고 이의 체계적인 수행 및 관리를 통합적으로 담당하는 소프트웨어시스템이다[2]. 이렇게 볼 때 비즈니스 프로세스 관리는 '90년대 프로세스의 설계와 운영 환경을 제공하고 시스템적인 핵심 수단이 되는 워크플로우와 유사하다. 워크플로우와 비즈니스 프로세스 관리가 추구하는 목표와 제시하는 수단은 동일하지만 비즈니스 프로세스 관리는 새로운 정보기술, 비즈니스 가치에 기반하고 있다는 점에서 차이가 있다. 따라서 워크플로우 기반인 비즈니스 프로세스 관리를 알기 위해서는 먼저 워크플로우에 대한 내용을 먼저 알아야 할 것이다.

워크플로우는 사람, 프로세스, 자원의 3대 요소를 통하여 잘 정의된 업무들과 그에 관련된 사람, 정보 및 기타 자원의 흐름을 통합적으로 관리, 지원해 주는 업무처리 자동화 시스이다[1]. 워크플로우의 전형적인 예로는 은행 거래, 여행 계획, 주문 처리, 기업에서의 제조 공정, 토목 공사 등을 포함한다. 여기서 언급하는 업무는 프로세스(process), 자원을 에이전트(agent)라고 부르기도 한다.

워크플로우 스케줄링은 워크플로우 업무들에 대한 정확한 실행 순서를 찾는 문제이다. 다시 말해서 워크플로우의 비즈니스 로직에 포함되어 있는 제약조건들을 따르는 실행이다. 워크플로우 스케줄링 연구는 대부분 복잡한 업무의 정확한 순서를 명시하는 시간적 제약들에 집중되어 있다[3]. 자원할당에서 발생한 제약조건들의 상당수 항목들은 워크플로우 모델링에 비교적 적은 관심을 받아왔다. 자원의 예로 직원, 물리적인 장비, 업무의 목적을 성취하기 위해서 필요한 작업장의 장비와 같은 것들을 말한다. 자원에 의한 업무의 실행에 비용(예산, 시간)들을 연관시키는 것은 아주 일반적이다. 다른 말로 제약조건(Constraint)이라고 한다. 업무 진행시 만족되어야 하는 조건의 요소가 되는 것이다. 제약조건을 만족시키지 못하면 예외 상태나 다른 정의된 절차를 야기한다. 일반적으로 "자원은 공유될 수 없다", "비용은 무한한 것이 아니다"라는 등의 예를 들 수가 있다. 워크플로우 실행 스케줄링에서는 언제, 어떻게 자원을 사용할 것인지와 같은 결정을 제약조건으로 포함하기도 한다. 비록 자원관리는 워크플로우 관리시스템(WfMS)의 중요한 측면으로 인식되었음에도 불구하고 대부분의 작업은 다양한 자원들을 모델링하는데 초점을 맞추어 왔지, 자원들과 연관된 제약조건 하에서의 스케줄링에 대한 관심은 적은 편이었다[7, 10, 16, 18].

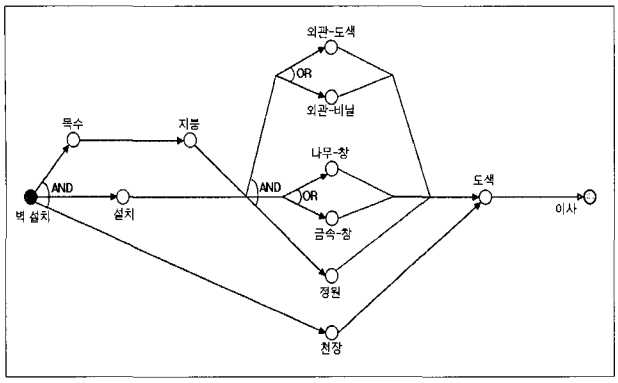
이해를 돕기 위하여, 실행해야 하는 세 개의 업무(*task*1, *task*2, *task*3)가 있다고 하자. *task*2는 *task*1 실행 후에 실행 가능하고, 또 *task*1이 실행되고 난 후에 *task*2와 *task*3이 실행되어야 한다.(단 *task*3은 *task*2 다음에 실행되며 이 둘은 *task*1 다음에 실행 가능하다.) 자원할당 제약조건으로 만약 *task*1이 어떤 에이전트에 의해 실행된다면 *task*2 또한 같은 에이전트에 의해 수행되어져야 하며 또 총 시간 또는 총 비용은 정해진 제한을 초과해서는 안 된다는 것이다. 또 다른 일반적인 자원할당 제약조건으로 같은 에이전트들은 같은 워크플로우 병렬 분기에 할당될 수 없으며, 기계는 동시에 다른 작업에 사용될 수가 없다는 것이다. 여기서 우리는 자원할당 제약조건을 두 개의 항목으로 구분할 수가 있다. 첫째로 비용 제약조건(*cost constraint*)은 몇 몇 총액의 함수에 제약조건이 있다. 둘째로 조정 제약조건(*constrol constraint*)은 어떻게 자원을 할당 할 것인지를 제약한다. 위 예에서 *task*1이 어떤 에이전트에 의해 실행되어지면 후에 *task*2가 같은 에이전트에 의해서 실행되어야 한다는 것이 조정 제약조건이다. 반면에 총 비용(시간)은 주어진 총금액(총시간)을 초과하지 않는다는 것이 비용 제약조건이다.

본 논문에서는 자원할당 제약조건들의 집합을 정확하게 명시하는 워크플로우에 대한 논리적 프레임워크를 소개한다. 이 논리적 프레임워크는 CTR 기반의 논리적인 변형 템플릿을 제시한다. 워크플로우 스케줄링의 업무 처리 프로세스를 상태 논리식으로 표현할 수 있는데, 이 변형 템플릿은 워크플로우 스케줄링을 새로운 워크플로우 스케줄링에 적용할 정확한 자원할당 제약조건 집합을 쉽게 산출하는데 효율적이다. 변형 템플릿에는 전체적인 워크플로우를 변형할 수 있는 워크플로우 변형 템플릿(Workflow Transformation Template)과 워크플로우 변형에서 필요로 하는 제약조건들을 산출할 수 있는 제약조건 변형 템플릿(Constraint Transformation Template)이 있다. 변형 템플릿 적용 후에는 새로운 워크플로우 스케줄링 명세와 정확한 자원할당에서의 제약조건 결과 집합을 얻을 수가 있다.

이후 본 논문은 다음과 같이 전개된다. 1장 서론에 이어 2장에서는 본 논문에서 이용될 집 건설 워크플로우 시나리오를 살펴보고, 3장에서는 CTR이 워크플로우로 어떻게 표현되는지를 알아본다. 4장과 5장에서는 논문에서 적용될 CTR 연산자, 공식 및 자원할당, 제약조건 등의 기본적인 내용을 좀 더 살펴본다. 6장에서는 본 논문에서 제시한 변형 템플릿인 워크플로우 변형 템플릿과 제약조건 변형 템플릿을 기술한다. 7장에서는 전체 흐름의 논리적 프레임워크를 소개하고 8장에서는 결론과 함께 앞으로의 연구방향으로 끝을 맺는다.

2. 시나리오

다음에 나올 예시는 [6]에서 파생된 본 논문에서 사용될 시나리오이다. 회사 A는 집을 짓는데, 정원을 만들고, 새로운 집으로 고객의 가구를 옮긴다. 이 회사는 다양한 하부작업을 위해 다른 회사들과 하도급 계획을 맺는다. 몇 몇 하도급 후보들이 있으며, 또 같은 회사에 여러 하도급 일을 할 수 있는 자격이 주어질 수 있다. 고객의 요구와 우리 이익을 최대화를 만족하기 위하여 회사 A는 가장 적당한 회사를 선택하기를 원한다. 이를 표현한 워크플로우는 (그림 1)에서 보여준다. 이 (그림 1)에서 AND-node는 워크플로우 내에서 하나의 제어 쓰레드가 두 개 이상의 병렬 단위업무로 갈라지는 지점을 말한다.(단 여기서 AND-node 분기로 시작해서 AND-node 결합으로 완료되어야 한다.) OR-node 실행은 선택 경로를 의미한다. 순서적으로 수행되어져야 하는 작업은 방향성 간선으로 연결된다.



(그림 1) 집을 건설하기 위한 워크플로우

이 워크플로우에서 자원할당 제약조건은 다음을 포함한다.

- 집을 건설하기 위한 비용은 주어진 총 경비를 넘어서는 안 된다.
- 집을 건설하기 위한 기간은 주어진 기간보다 오래 걸려서는 안 된다.
- 다른 회사들은 병렬 업무를 위해 선택되어야 한다. (건설을 빠르게 하기 위해서)

위 예제의 내용으로 첫 번째와 두 번째는 경비와 기간으로 한정되는 비용 제약조건이고, 세 번째는 조정 제약조건이다.

3. CTR과 워크플로우

CTR은 복잡한 프로세스의 병렬실행을 모델화하기 위한 연산자로 트랜잭션 로직을 확장한 것이다. 워크플로우 스케줄링에서 처리해야 할 업무를 트랜잭션으로 볼 때, CTR은 유용한 논리적 모델링 도구이면서 추론 도구이다. 특히 워크플로우 스케줄링을 모델링하고 추론하는데 아주 좋은 형식들 중의 하나이다[5]. CTR은 first-order 공식의 연산자 $\otimes$(직렬 결합)와 $|$(병렬 결합)으로 논리적 공식이 확장되었다. CTR은 multi-path(다중 경로, m-path)를 기초로 한다. path는 한 개 또는 그 이상의 데이터베이스 상태의 연속이고($< d_1, d_2 \cdots d_n >$), m-path는 path의 연속이다.($< p_1, p_2 \cdots p_n >, p_i : path$)

공식들은 m-path 사이의 실행트랜잭션으로 보이게 된다. 그렇게 하는 동안 기초적인 데이터베이스 상태를 질의 변경한다. CTR에서 m-path를 따르는 실행은 m-path 전체에서 true가 되는 것과 동일하다. 비공식적으로 만일 $\varnothing$가 m-path Π_1에서 true($\Pi_1 \models \varnothing$)이고 ψ가 Π_2에서 true($\Pi_2 \models \psi$)라면, $\varnothing \otimes \psi$는 전체적으로 연결된 m-path($\Pi_1 \bullet \Pi_2$)에서도 true가 된다. 마찬가지로 $\varnothing | \psi$도 m-path($\Pi_1 \| \Pi_2$)에서 true가 된다. $\Pi_1 \| \Pi_2$는 Π_1과 Π_2를 포함하는 일부분 path들의 인터리빙에 의한 Π_1과 Π_2로부터 얻어지는 m-path이다.

다음은 CTR이 어떻게 워크플로우의 모델에 이용되는지를 나타내어 준다. 아래 식은 (그림 1)에서 보인 집을 건설하기 위한 워크플로우에 CTR 연산자를 이용하여 논리적으로 모델화한 공식을 보여준다.

벽 $\otimes$
((((목수 $\otimes$ **지붕)** $|$ **설치)**
middle-task-part **)** $|$ **천장)**
$\otimes$ **도색** $\otimes$ **이사**

여기서 각 어휘들은 $task$를 의미하고 CTR 연산자들은 그 $task$들을 어떻게 연결하는지를 보여준다.(병렬적, 직렬적) $middle-task-part$ 부분은 표현에 적합하지 않고 분리되어 표현되어도 가능한 워크플로우의 부분을 표현한 것이다. 아래는 *middle-task-part* 부분을 CTR 연산자를 이용하여 논리적으로 모델화한 공식을 보여준다.

(외관-도색 $\vee$ **외관-비닐)** $|$ **(나무-창** $\vee$ **금속-창)** $|$ **정원**

마지막 공식으로부터 보여 지는 연산자로, V는 CTR 공식에서의 선택 실행을 의미한다. 다시 말해서, 워크플로우를 실행하기 위해서 우리는 두 개의 외관 작업이 아닌 단지 외관 관련 작업들 중 하나의 선택 실행만을 의미하는 것이다.

4. CTR 공식의 논리적 확장

앞 장에서 CTR의 기본 연산자와 CTR이 워크플로우 스케줄링을 잘 표현할 수 있는 논리적 모델링 도구라는 것을 간단한 예시로 알아보았다. 이 장에서는 워크플로우 스케줄링에 적당하게 CTR 공식을 논리적으로 확장을 할 것이다. 이 의미는 워크플로우 전체 작업에 제약조건을 가하는 것이 아니라, 변화하는 부분적인 작업 각각에 자원할당과 제약조건을 가함으로써 동적의 워크플로우 스케줄링에 적용하고자 함이다. CTR은 앞에서 언급했듯이 m-path 기반이다. 하지만 부분적인 제약조건을 가하기 위해서 m-path 대신에 부분 스케줄(partial schedule)에 기반을 두고 정의 할 것이다. 이 후에 많은 자원할당 제약조건을 따라야 하는 워크플로우 모델과 스케줄링에 CTR이 어떻게 사용되는지 보여 줄 것이다.

4.1 CTR 확장 연산자

CTR 기반인 m-path는 직렬적이고 동시진행적인 실행에 적당한 반면에 실행의 성공을 위한 자원의 요구사항들을 모델하기에는 충분하지가 않다. 따라서 확장 CTR에서는 두 개의 m-path는 동일 실행의 서로 다른 병렬 분기의 부분이라는 구별이 필요하다. 이해를 돕기 위해서 부분 스케줄의 개념을 소개한다.

부분 스케줄에는 두 개의 연산자($\bullet_p$, $\|_p$)가 있다. 첫 번째($\bullet_p$)는 직렬연결을 의미하고 조합적이다. 두 번째($\|_p$)는 스케줄들을 병렬연결하고 이는 조합적이며 교환 가능하다. 부분 스케줄은 다음과 같이 정리 된다

- m-path에서 w가 부분 스케줄이면

- 두 부분 스케줄들의 직렬 합성, $w_1 \bullet_p w_2$는 하나의 부분 스케줄이 된다.
- 두 부분 스케줄들의 병렬 합성, $w_1 \| _p w_2$는 하나의 부분 스케줄이 된다.

추가적인 설명으로 만약 $w = < p_1 \cdots p_n >$과 $w' = < p_1' \cdots p_n' >$가 m-path($p_1, p_2$등은 sequence path)이면 따라서 $w_1 \bullet_p w_2$도 m-path $< p_1, \cdots p_n, p_1' \cdots p_n' >$이다

4.2 CTR 확장 모델

논리적인 확장은 m-path 구조 기반이다. m-path 구조(M)는 정규식인 first-order semantic 구조를 모든 m-path인 w에 할당하는 맵핑이다.(즉, $M(w)$도 first-order semantic 구조이다.)

CTR에서의 식이 잘 정의된 m-path에서 true라면 이 path를 따라서 실행 가능하다. 유사하게 CTR의 논리적인 확장에서의 기반인 부분 스케줄에서도 true인 식은 그 스케줄을 따라서 실행될 수 있다는 것으로 해석할 수 있다. CTR 결과는 first-order 로직의 일반적인 공식과 연산자($\otimes$,|,$\vee$)로 구성이 된다. 앞 장에서 CTR이 워크플로우 스케줄링에 어떻게 적용되는지 살펴보았다. w를 부분 스케줄로 M을 m-path structure로, α를 CTR 결과라고 하자. 스케줄링의 잘 실행된 결과 표현은 아래와 같은 형식으로 나타낼 수가 있다.

- $M, w \models \alpha$
- $M, w \models \alpha \otimes \beta (w = w_1 \bullet_p w_2,\ M, w_1 \models \alpha \text{ and } M, w_2 \models \beta)$
- $M, w \models \alpha | \beta (w = w_1 \| _p w_2,\ M, w_1 \models \alpha \text{ and } M, w_2 \models \beta)$
- $M, w \models \alpha \vee \beta (either\ M, w \models \alpha \text{ or } M, w \models \beta)$

첫 번째는 공식으로 "트랜잭션 α가 스케줄 w에 따라 M에서 true이다"라고 읽는다. 두 번째, 트랜잭션 $\alpha \otimes \beta$는 두 스케줄이 직렬연결인 경우에만 스케줄 w에 따라 실행을 한다. α는 전위-스케줄에 따라 실행하며, β는 후위-스케줄에 따라 실행을 한다. 즉 α 실행하고 나서 β를 실행한다는 것이다. 세 번째는 트랜잭션 $\alpha | \beta$는 두 스케줄이 병렬연결인 경우에만 스케줄 w에 따라 실행한다. 이 트랜잭션은 둘 모두 동시에 실행할 수 있다. α는 w_1에 따라 실행하고, β는 w_2에 따라 실행 가능하기 때문이다. 네 번째, 트랜잭션 $\alpha \vee \beta$을 실행하기 위해서는 α, β를 분리해도 충분히 실행할 수 있다. 다시 말해서 어느 하나의 트랜잭션만 선택하여 실행 가능하다는 것을 말한다.

5. 자원과 제약조건

이제 워크플로우 단위업무의 실행에서 요구되는 자원들과 이 자원에서의 제약조건들에 관해서 설명한다. 먼저 자원의 개념과 그 자원에 대한 할당 형식에 대하여 알아본다. 이후 제약조건의 두 종류를 소개할 것이다. 첫 번째 비용 제약조건(cost constraint)인데, 실행 스케줄들에 정의 되어진 총 비용 함수들을 포함한다. 두 번째로 조정 제약조건(control constraint)인데, 자원이 다른 공식들에 할당되는 것을 제한한다.

5.1 자원과 자원할당

자원은 애트리뷰트 *token*과 *cost*를 가지는 *object*이다. 워크플로우 모델링에서 자원은 일반적으로 실행 에이전트를 나타낸다. 애트리뷰트 *token*은 에이전트의 이름을 나타내고, 애트리뷰트 *cost*는 에이전트가 사용한 비용을 나타낸다. 표기의 편의를 위하여 $cost\_of()$ 함수는 자원의 cost 애트리뷰트의 값을 반환 받고, *token_of*() 함수는 token 애트리뷰트의 값을 반환 받는다.

자원할당은 부분 스케줄을 자원 집합으로의 부분 맵핑(partial mapping)을 말한다. 자원할당 함수로 $asg(w)$로 나타내며 다음과 같이 정리할 수 있다.

- $asg(w_1 \| _p w_2) = asg(w_1) \cup asg(w_2)$
- $asg(w_1 \bullet_p w_2) = asg(w_1) \cap asg(w_2)$
- $asg(w) = Rs$ (Rs : Resource subset, w : m-path)

5.2 제약조건 시스템(S)

제약조건 시스템(S)는 제약조건 정의들의 집합이며, S_{cost}와 S_{ctrl}로 구성된다. S_{cost}는 비용 제약조건들을 명시하며, (예로 작업은 1일 이내에 실행을 해야 한다.) S_{ctrl}은 조정 제약조건을 명시한다.(예로 2층에 있는 장비는 두 대 동시 작업에는 사용을 할 수 없다)

5.2.1 비용 제약조건(S_{cost})

S_{cost}는 w는 부분 스케줄, asg는 자원할당으로 할 때 $cost\_constraint(\omega, asg)$ 형식의 술어로 구성된다. 좀 더 명확하게 한다면, *value_constraint*가 전체적인 결과 정수 영역을 가지는 술어라 할 때 $cost_constraint(\omega, asg)$는 *value_constraint*($cost(\omega, asg)$) 형식을 가진다. 그리고 $cost()$는 아래 정리의 속성을 가지는 함수이다. 여기서 w_1과 w_2는 $cost(\omega_1, asg)$와 $cost(\omega_2, asg)$를 정의할 수 있는 부분 스케줄이라고 하자.

- $cost(\omega_1 \| _p \omega_2,\ asg) \equiv op_{|}(cost(\omega_1, asg),\ cost(\omega_2, asg))$
- $cost(\omega_1 \bullet_p \omega_2,\ asg) \equiv op_{\otimes}(cost(\omega_1, asg),\ cost(\omega_2, asg))$
- $cost(\omega, asg) \equiv cost\_of(asg(\omega))$

$cost_constraint$의 정의를 위해, $op_{|}$와 $op_{\otimes}$에 대하여 아래와 같이 정리할 수 있다.

〈표 2〉 S_{cost}의 정리

교환법칙	$op_{|}(X, Y) = op_{|}(Y, X)$
결합법칙	$op_{|}(op_{|}(X, Y), Z) = op_{|}(X, op_{|}(Y, Z))$ $op_{\otimes}(op_{\otimes}(X, Y), Z) = op_{\otimes}(X, op_{\otimes}(Y, Z))$

5.2.2 조정 제약조건(S_{ctrl})

S_{ctrl}는 아래의 조건을 만족하는 $ctrl\_constraint(w,asg)$ 형식의 술어로 구성된다. $asg(w_1)$, $asg(w_2)$, $asg(w_3)$와 같이 정의할 수 있는 w_1, w_2, w_3를 부분 스케줄, asg를 할당이라고 하자.

- $ctrl\_constraint(w_1 \bullet_p w_2,\ asg) \equiv$
  $set\_constraint_{\otimes}(asg(w_1),asg(w_2)) \wedge$
  $ctrl\_constraint(w_1,asg) \wedge ctrl\_constraint(w_2,asg)$
- $ctrl\_constraint(w_1 \| _p w_2,\ asg) \equiv$
  $set\_constraint_{|}(asg(w_1),asg(w_2)) \wedge$
  $ctrl\_constraint(w_1,asg) \wedge ctrl\_constraint(w_2,asg)$
- $ctrl\_constraint(w,\ asg) \equiv leaf\_constraint(asg(w))$
  (w : m-path)

$ctrl_constraint$의 정의를 위해, $set_constraint_{\otimes}$와 $set_constraint_{|}$에 대하여 다음과 같이 정리할 수 있다.

〈표 3〉 S_{ctrl}의 정리

교환 법칙	$set\_constraint_{\|}(V_1,V_2) = set\_constraint_{\|}(V_2,V_1)$
배분 법칙	$set\_constraint_{\otimes}(V_1 \cup V_2,V_3) =$ $set\_constraint_{\otimes}(V_1,V_3) \wedge set\_constraint_{\otimes}(V_2,V_3)$ $set\_constraint_{\|}(V_1 \cup V_2,V_3) =$ $set\_constraint_{\|}(V_1,V_3) \wedge set\_constraint_{\|}(V_2,V_3)$

5.3 자원할당 제약조건의 기본논리

기본적인 자원, 자원할당, 제약조건에 대하여 살펴보았으며, 제약조건에는 비용 제약조건 및 조정 제약조건이 있었다. 자원에서 비용(cost)은 $cost\_constraint$, 토큰($token$)은 $ctrl_constraint$를 사용한다. 추가적으로 함수 $op_{\otimes}$와 $op_{|}$는 합계(sum) 또는 최대값(max)이며. $set_constraint_{|}$과 $set_constraint_{\otimes}$는 다양한 제약조건 집합으로 표현된다.

V_1, V_2, c를 자원들의 집합이라고 하자. 자원의 할당은 asg이며, 제약조건은 비용 제약(cost, S_{cost}) 및 조정 제약($control$, S_{ctrl})이다. 또 비용 제약에서 cost 함수는 할당된 값으로 반환되어야 한다. 예시에서 집을 건설하는데 비용자원의 부분으로 건설기간($time$)을 V_1으로 건설비용($amount$)을 V_2로 설정을 한다면, 이를 cost 형식의 리스트로 나타낼 수가 있다.

- $cost(\omega,\ asg) = cost\_of(asg(\omega)) = [V_1,\ V_2]$

건설 작업에서의 건설기간(V_1)은 c_1을 초과해서는 안 되고, 건설비용(V_2)은 c_2를 초과해서는 안 된다는 제약이 있을 경우 아래와 같이 나타낼 수 있다.

- $value\_constraint([V_1,\ V_2]) \equiv V_1 < c_1,\ V_2 < c_2$

함수 $op_{|}$와 $op_{\otimes}$는 할당된 비용이 어떻게 합계가 되는지를 정의한다. 예로 병렬 작업 실행에서는 건설기간의 최대값이 사용되며, 반면에 건설비용은 더해진다.

- $op_{|}([V_1,\ V_2],[V_1',\ V_2']) \equiv [\max(V_1,\ V_1'),\ V_2 +\ V_2']$

다음으로 조정 제약조건으로 병렬 작업 실행에서 할당된 자원들은 공통 원소를 갖지 않아야 한다는 조정 제약조건은 아래와 같다.(자원을 동시에 작업에서 사용하지 못하는 제약 때문이다.)

- $set\_constraint_{|}(V_1,\ V_2) \equiv (token\_of(V_1) \cap token\_of(V_2) = \varnothing)$
  $disjoint(V_1,\ V_2) \equiv (token\_of(V_1) \cap token\_of(V_2) = \varnothing)$

제약조건 시스템(S)의 만족은 다음과 같이 정의할 수 있다. S를 제약조건 시스템, D를 제약조건 전체집합, w를 부분 스케줄, asg를 자원할당, M을 m-path 구조, 워크플로우 명세를 $\varnothing$라 하자. 논리적으로 $(M,S,w,asg) \models \varnothing$와 같이 표현되며, 다음을 따른다.

- $M,\ w \models \varnothing$
- $D \models cost\_constraint(w,\ asg)$, and
- $D \models ctrl\_constraint(w,\ asg)$

다시 말하면 성공적인 워크플로우 실행이 되기 위해서는 $\varnothing$은 w에 따라 실행을 하고, 제약조건 시스템(S)에서 $cost\_constraint$, $ctrl_constraint$를 모두 만족해야 한다는 것이다.

6. CTR을 이용한 변형 템플릿

6.1 워크플로우 변형 템플릿(Workflow Transformation Template)

앞 장까지 자원, 제약조건 등과 함께 워크플로우를 명세하기 위한 필요한 내용들을 모두 갖추었다. 이 장에서는 새로운 워크플로우 명세를 위한 변형(Transformation)을 제시한다. CTR에서의 워크플로우 명세($\varnothing$), 제약조건 시스템(S), 제약조건 전체집합(D), 새로운 워크플로우 명세($\varnothing'$), 자원할당(asg), m-path 구조(M)로 하자.

- 제약조건을 만족하는($D \models cost\_constraint(w,\ asg) \wedge ctrl\_constraint(w,\ asg)$) 모든 부분 스케줄 w에 대해 $M,w \models \varnothing$이면 $M,w \models \varnothing'$가 된다.
- 모든 스케줄 w에 대해 $(M,\ S,\ w,asg) \models \varnothing'$이다.

워크플로우 명세 $\varnothing$와 $\varnothing'$는 제약조건을 만족하는 워크플로우 스케줄에서 서로 동일하다. 모든 실행은 제약조건을

만족하는 스케줄이다. 앞에서 말한 변형은 워크플로우 스케줄을 구성하는 메인 단계이다. 이제 자원할당 제약조건 하에서의 새로운 워크플로우 스케줄링을 쉽게 접근할 수 있는 CTR을 이용한 워크플로우 변형 템플릿을 소개하고자 한다. 워크플로우 변형 템플릿은 (그림 2)와 같이 정의 할 수 있다.

$B(G_1 \vee G_2) \equiv B(G_1) \vee B(G_2)$ G: *workflow*
$B(G) \equiv R(G,T) \otimes cost\_constraint(T) \otimes ctrl\_constraint(T)$
$R(A,T) \equiv A \otimes (T = resource\_asg(A, Agents))$
T: *atomic task*
$R(G_1 | G_2, T) \equiv (T = '|'(T_1, T_2)) \otimes (R(G_1, T_1) \mid R(G_2, T_2))$
$R(G_1 \otimes G_2, T) \equiv (T = '\otimes'(T_1, T_2)) \otimes (R(G_1, T_1) \otimes R(G_2, T_2))$
$R(G_1 \vee G_2, T) \equiv R(G_1, T) \vee R(G_2, T)$

(그림 2) 워크플로우 변형 템플릿

변형 자체는 워크플로우 구조의 유도에 의해 (그림 2)에서 정의되었다. Operator B는 기존 워크플로우 구조를 새로운 워크플로우 구조로 변형한다. 변형에 있어 워크플로우가 가지는 의미(semantic)는 그대로 유지할 뿐만 아니라 비용(cost)과 조정(control) 제약조건들도 포함한다. 또 Operator B는 다른 Operator R을 유도한다. R은 워크플로우 G를 가지고 있으며, 워크플로우의 자원할당 항목을 생성한다. 워크플로우는 후에 G의 파스트리(parse tree)로 보이게 되며, 파스트리의 잎(leave)은 *task*를 의미하면 *task* 자원할당으로 대체한다. 자원할당은 $resource\_asg(task, Agents)$ 형식의 용어로 할 수 있다. 그리고 두 종류의 자원할당 제약조건(*cost_constraint*, *ctrl_constraint*)은 워크플로우 G에 의해 워크플로우 명세에 추가가 된다.

- $resource\_asg(task, Agents)$: *task* 자원할당
- *cost_constraint* : workflow의 총 비용의 제약이다.
- *ctrl_constraint* : workflow의 자원할당을 조정한다.

다음은 앞 2장에서 제시한 논문의 예시 워크플로우의 작업에 대한 자원할당 제약조건들이 변형 템플릿에 의해 워크플로우 명세에 추가되는 것을 알아보자. 변형 템플릿에 적용하기 전에 작업의 긴 이름을 c(목수, carpentry), r(지붕, roof), i(설치, installation), g(정원, gardening)와 같은 약어로 나타낼 것이다. 워크플로우 변형 템플릿 적용은 총 5단계로 구성되어지며, 예시의 워크플로우는 (그림 3)과 같이 적용이 된다.

(*carpentry* ⊗ *roof*) | (*installation* | *gardening*)

step 1	$B((c \otimes r) \mid (i \mid g))$
step 2	$R((c \otimes r) \mid (i \mid g),\ T) \otimes$ $cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 3	$T = '\mid'(T_1, T_2) \otimes (R(c \otimes r, T_1) \mid R(i \mid g, T_2)) \otimes$ $cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 4	$T = '\mid'(T_1, T_2) \otimes$ $((T_1 = '\otimes'(T_3, T_4) \otimes (R(c, T_3) \otimes R(r, T_4))) \mid$ $(T_2 = '\mid'(T_5, T_6) \otimes (R(i, T_5) \otimes R(g, T_6)))) \otimes$ $cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 5	$T = '\mid'(T_1, T_2) \otimes$ $((T_1 = '\otimes'(T_3, T_4) \otimes$ $((c \otimes T_3 = resource\_asg(c, W)) \otimes$ $(r \otimes T_4 = resource\_asg(r, X)))) \mid$ $(T_2 = '\mid'(T_5, T_6) \otimes$ $((i \otimes T_5 = resource\_asg(i, Y)) \otimes$ $(g \otimes T_6 = resource\_asg(g, Z)))))) \otimes$ $cost\_constraint(T) \otimes ctrl\_constraint(T)$

(그림 3) 집 건설 워크플로우의 변형 템플릿 적용 예

6.2 제약조건 변형 템플릿(Constraint Transformation Template)

5.2절에서 소개된 제약조건 시스템에서 *cost constraint* 및 *control constraint*에 대한 내용을 살펴보았으며, 또한 워크플로우 변형 템플릿 적용과정에서 제약조건을 포함하였다. 이 절에서는 두 제약조건의 일반적인 속성들을 변형 템플릿에 적용하여 설명하고자 한다. 아래 (그림 4)는 제약조건에 대한 변형 템플릿이다.

변형 템플릿 ①~③은 *ctrl_constraint*를 정의한다. '|'-nodes, '⊗'-node, '*leaf*'에 대한 조정 제약조건(*constrol constraint*)이다. ④~⑦은 *cost_constraint*를 정의이며 비용 제약조건(*cost constraint*)이다. 아래의 (그림 5)는 본 논문의 예시인 집 건설에 대한 제약조건 변형 템플릿을 위한 정의된 *placeholder*이다.

① $ctrl\_constraint(|(T_1, T_2)) :-$
$set\_constraint_{|}(T_1, T_2),\ ctrl\_constraint(T_1),\ ctrl\_constraint(T_2)$
② $ctrl\_constraint(\otimes(T_1, T_2)) :-$
$set\_constraint_{\otimes}(T_1, T_2),\ ctrl\_constraint(T_1),\ ctrl\_constraint(T_2)$
③ $ctrl\_constraint(T) :-\ leaf(T),\ leaf\_constraint(T)$
④ $cost\_constraint(T) :-\ cost(T, V),\ value\_constraint(V)$
⑤ $cost(\otimes(T_1, T_2), V) :-\ cost(T_1, V_1),\ cost(T_2, V_2),\ op_{\otimes}(V_1, V_2, V)$
⑥ $cost(|(T_1, T_2), V) :-\ cost(T_1, V_1),\ cost(T_2, V_2),\ op_{|}(V_1, V_2, V)$
⑦ $cost(resource\_asg(T, Agent), V) :- cost\_of(resource\_asg(T, Agent), V)$

(그림 4) 제약조건 변형 템플릿

$cost\_of(resource\_asg(T,A),[V,U]) :$
$- duration(T,A,V),\ price(T,A,U)$
$value\_constraint([V,U]) :- V < c_1,\ U < c_2$
$set\_constraint_{|}(T_1,T_2) :- disjoint(T_1,T_2)$
$set\_constraint_{\otimes}(T_1,T_2) :- true$
$leaf\_constraint(T) :- true$
$op_{|}([V_1,U_1],[V_2,U_2],[V,U]) :-$
$V\ is\ \max(V_1,V_2),\ U\ is\ U_1 + U_2$
$op_{\otimes}([V_1,U_1],[V_2,U_2],[V,U]) :$
$- V\ is\ V_1+V_2,\ U\ is\ U_1+U_2$

(그림 5) 제약조건 변형 템플릿을 위한 *placeholder*

워크플로우 변형 템플릿 및 제약조건 변형 템플릿 적용한 후 결과는 집 건설 워크플로우 스케줄링에 적용할 수 있는 자원할당 및 제약조건에 대한 결과 집합을 얻을 수가 있다. 이 결과를 새로운 워크플로우 명세에 추가 적용이 될 수가 있다.

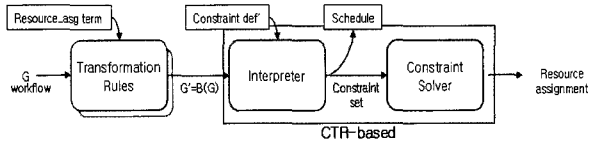
Basic conditions	$duration(c,W,V_1),\ price(c,W,U_1)$ $duration(r,X,V_2),\ price(r,X,U_2)$ $duration(i,Y,V_4),\ price(i,Y,U_4)$ $duration(g,Z,V_5),\ price(g,Z,U_5)$
cost *constraint*	$V_3 = V_1 + V_2,\quad U_3 = U_1 + U_2$ $V_6 = \max(V_4,V_5),\quad U_6 = U_4 + U_5$ $V = \max(V_3,V_6),\quad U = U_3 + U_6$ $V < c_1,\quad U < c_2$
control constraint	$Y \neq Z,\ W \neq Y,\ W \neq Z,\ X \neq Y,\ X \neq Z$

(그림 6) 예시에 적용될 제약조건 집합

7. 논리적 프레임워크

앞에서 워크플로우 변형 템플릿과 제약조건 변형 템플릿을 적용시켜 새로운 워크플로우 명세와 제약조건 결과 집합을 얻을 수 있었으며, 전체 처리 과정을 세 개의 메인으로 설계할 수 있다. 이 모든 것이 새로운 워크플로우 스케줄링에 적용할 자원할당 제약조건의 결과 집합을 구성하는 과정이라 할 수 있다. 세 개의 메인 요소는 다음과 같다.

① Transformation Template
② CTR interpreter(Constraint Template)
③ Constraint Solver



(그림 7) 논리적 프레임워크

위 (그림 7) 처리 과정의 단계를 보면 다음과 같다.

단계1 : 주어진 워크플로우 G는 변형 템플릿(Transformation Template)의 Operator B를 사용하여 새로운 워크플로우 G'로 변형하였다. (즉, $G' := B(G)$)

단계2 : 부분 스케줄은 워크플로우를 위한 기초이다. 이 단계에서 자원할당 제약조건들은 atomic constraint의 집합으로 수집이 된다.

단계3 : 이전 단계의 부분 스케줄 하에서 자원할당 제약들을 Constraint Solver를 사용하여 유효한 자원할당들의 집합을 생성한다.

8. 결 론

워크플로우 스케줄링은 작업의 정확한 실행 순서를 찾는 문제이다. 다시 말해서 워크플로우의 비즈니스 로직을 포함하는 제약조건을 따르는 실행을 말한다. 워크플로우 스케줄링의 연구는 대부분 정확한 업무의 실행 순서를 기술하기 위한 시간적인 제약조건 처리에 집중되어 왔다. 워크플로우 스케줄링에서 자원할당으로 발생하는 제약조건의 항은 많은데 비하여 그러한 자원들과 연관된 제약조건에서의 워크플로우 스케줄링에 대한 관심은 적은 편이며, 또한 모델을 논리적인 표현으로 제시하는 연구조차 미흡한 실정이다.

따라서 본 논문에서는 자원할당 제약조건 하에서의 워크플로우 스케줄링을 위한 논리적은 프레임워크를 제시하였다. 이 프레임워크는 복잡한 프로세스의 병렬실행을 모델화하기 위한 연산자로 트랜잭션 로직을 확장한 CTR(Concurrent Transaction Logic)을 이용하였으며, 이 CTR은 워크플로우 스케줄링을 모델링하고 추론하는데 아주 좋은 형식으로 소개되었다. 제시한 프레임워크에는 쉽게 워크플로우 논리적인 모델에 적용을 할 수 있는 변형 템플릿(Transformation Template)을 소개하였다. 변형 템플릿에는 새로운 워크플로우 스케줄링을 위한 워크플로우 변형 템플릿(Workflow Transformation Template)과 자원할당에서 발생하는 제약조건을 위한 제약조건 변형 템플릿(Constraint Transformation Template)이다. 변형 템플릿 적용으로 새로운 워크플로우 스케줄링 명세와 정확한 자원할당과 제약조건 등의 결과 집합을 얻을 수가 있었으며, 결과의 명세는 워크플로우 스케줄링의 모든 실행에서 만족을 보장받을 수가 있는 정확성을 가졌다.

향후 연구로서 워크플로우 프로세스의 단위업무 수행을 평가하는 논리적인 평가 모형을 CTR을 이용하여 평가식을 산출하는 템플릿으로 확장하거나, 나아가 비즈니스 프로세스 표현의 모델링, 비즈니스 프로세스 인스턴스의 각 단계에서 수행할 필요 자원의 효율적인 할당, 자원의 상호운영 등에 CTR을 이용한 논리적 모델을 제시하는 연구에 기대가 된다.

참 고 문 헌

[1] 안승해, 백창현, 1st Workflow, 시사컴퓨터, 2000.3.

[2] D. Hollingsworth, "Workflow Management coalition specification : the workflow reference model", WfMS specification, 1994.
[3] N. Adam, V. Atluri, and W. Huang. Modeling and analysis of workflows using Petri nets. Journal of Intelligent Information System, 10(2):131-158, March, 1998.
[4] A.J. Bonner and M. Kifer. Concurrency and communication in transaction logic. In Joint Int'l Conference and Symposium on Logic Programming, pp.142-156, Bonn, Germany, September, 1996. MIT Press.
[5] H. Davulcu, M. Kifer, C.R. Ramarkrishnan, and I.V. Ramakrishnan. Logic based modeling and analysis of workflows. In ACM Symposium on Principles of Database Systems, pp.25-33, Seattle, Washington, June, 1998.
[6] C. Schulte and G. Smolka. Finite domain constraint programming in oz. a tutorial. Version 1.1.0, February, 2000.
[7] G.Alonso, D.Agrawal, A. EI Abbadi, M. Kamath, R. Günthö, and C. Mohan. Advanced transaction models in workflow contexts. In international Conference on Data Engineering, New Orleans, Louisiana, February, 1996.
[8] G.Alonso, D.Agrawal, A. EI Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. in IEEE-Expert. special issue on Cooperative Information System, 1997.
[9] M. Brenner. A formal model for planning with time and resources in concurrent domains. In IJ-CAI Workshop on Planning with Resource, Seattle, USA, August, 2001.
[10] Workflow management Coalition. Terminology and glossary ver 3.0. Technical Report(WFMC-TC-1011), Workflow Management Coalition, Brussels, February, 1999.
[11] J.Eder, E. Panagos, and H. Pezewauning, and M. Rabinovich. Time management in workflow systems. In int. Conf. on Bussness Information Systems, pp.265-280, Poznan, Poland, 1999.
[12] J.Eder, E. Panagos, and M. Rabinovich. Time constraints in workflow systems. In Conference on Advanced Information Systems Engineering, pp.286-300, Germany, 1999.
[13] P.Halsum and H. Geffner. Heuristic planning with time and resource. In IJCAI Workshop on planning with Resources, Seattle, USA, August, 2001.
[14] Y. Huang and M. Shan. Policies in a resource manager of workflow systems : Modeling, enforcement and management. In International Conference on Data Engineering, 1999.
[15] W. Du, J. Davis, Y. Huang, and M. Shan. Enterprise workflow resource management. In int'l Workshop on Research Issues in Data Engineering, pp.108-115, Sydney, Australia, 1999.
[16] M. zur Muhlen. Resource modeling in workflow application. In Workflow Management Conference, pp.137-153, Muenster, Germany, November, 1999.
[17] P. Attie, M. Singh, A. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In Int'l Conference on Very Large Data Bases, Dublin, Ireland, August, 1993.
[18] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. Scheduling Computer and Manufacturing Processes. Springer-Verlag, 1996.
[19] A. Nareyek. Applying local search to structural constraint satisfaction. In IJCAI Workshop on Intelligent Workflow and Process Management : The New Frontier fo AI in Business, Stockholm, Sweden, August, 1999.

고 재 진

e-mail : jjkoh@mail.ulsan.ac.kr
1972년 서울대학교 응용수학과(공학사)
1981년 서울대학교 대학원 계산통계학과 (이학석사)
1990년 서울대학교 대학원 컴퓨터공학과 (공학박사)
1975년~1979년 한국후지쯔(주) 기술개발부 사원
1979년~현재 울산대학교 컴퓨터정보통신공학부 교수
관심분야 : DB시스템, 전문가 시스템, DB설계, ERP

안 형 근

e-mail : hkahn@mail.ulsan.ac.kr
2000년 한국방송통신대학교 컴퓨터과학과 (이학사)
2003년 울산대학교 정보통신대학원 정보통신공학과(공학석사)
2006년 울산대학교 대학원 컴퓨터정보 통신공학부(박사과정 수료)
1997년~2004년 현대오토시스템 기술지원부
2004년~현재 (주)CFIC 기업부설연구소 연구소장
관심분야 : 멀티미디어DB, DB설계/분석, ERP, BPM, Workflow

이 단 영

e-mail : danyoung@hanmail.net
1986년 울산대학교 전자계산학과(공학사)
1994년 울산대학교 교육대학원 전자계산학과(교육학 석사)
2002년 울산대학교 대학원 컴퓨터정보 통신공학부(박사과정 수료)
2002년~2004년 울산대학교 IT교육원 객원교수
2006년~현재 울산대학교 컴퓨터정보통신공학부 객원교수
관심분야 : DB분석/설계, ERP, e-Business