CTR 기반 자원할당 제약조건 하에서의 비즈니스 프로세스를 위한 논리적 변환규칙

안형근° 고재진
울산대학교 컴퓨터·정보통신공학부
{hkahn°, jjkoh}@mail.ulsan.ac.kr

A Logical Transformation Rules for Business Process Under Resource Allocation Constraint based CTR

Hyoung-Keun An°, Jae-Jin Koh
School of Computer Engineering & Information Technology, University of Ulsan

요 약

최근 수많은 정보를 통해서 증명 되듯이 비즈니스 프로세스 관리(Business Process Management, BPM)는 기업의 소프트웨어 시장의 가장 중요한 부분을 차지하고 있다. 비즈니스 프로세스 관리는 프로세스 전 라이프 사이클을 지원하고자 하는 개념으로 급변하는 경영 환경의 변화에서 기업의 경쟁력을 재고하기위하여 필요로 되는 새로운 기업 컴퓨팅 패러다임이다. 비즈니스 프로세스 관리의 핵심적인 정보시스템 역할을 수행하는 워크플로우(Workflow)에서의 스케줄링은 정확한 업무 순서를 명세하기 위한 시간적인 제약들에 집중되어 있다. 워크플로우와 BPM과 같은 비즈니스 프로세스에서의 또 하나의 중요한 측면은 자원할당관리이다. 현재 대부분 다양한 자원들을 모델링하는데 초점을 맞추어 왔으며 자원들과 연관된 제약에서의 스케줄링에 대한 관심은 많지 않은 편이었다. 본 논문에서는 First Order Logic 기반의 CTR(Concurrent Transaction Logic)을 이용하여 각 비즈니스 프로세스 스케줄링을 위한 자원할당에 따른 제약들을 논리적인 모형으로 구체화하고, 모형에 필요한 변환 규칙을 소개하고자 한다.

1. 서 론

다양한 환경 속에서 기업이 업무의 효율성과 효과적인 업무처리를 해결하기 위해서 핵심정보시스템으로 워크플로우를 도입하고 있으며, 최근에는 워크플로우의 확장으로 비즈니스 프로세스 관리(Business Process Management, BPM)가 기업의 소프트웨어 시장의 가장 중요한 부분을 차지하고 있다. 이 프로세스 관리에서 핵심적인 역할을 수행하는 것은 바로 워크플로우 관리시스템(Workflow Management System, WfMS)으로 알려져 있다.[1] 이런 워크플로우는 기업의 비즈니스 프로세스를 모델링하고 정보기술과 결합시켜 실제 업무처리가 가능하게 해주는 프로세스 관리도구 이다. 비즈니스 프로세스 스케줄링은 워크플로우 작업에 대한 정확한 실행 순서를 찾는 문제이다. 다시 말하면 실행이란 비즈니스 로직을 포함하는 제약을 따르는 것이다. 현 비즈니스 프로세스의 중요한 측면으로 정확한 업무순서, 자원할당관리, 상호작용이라고 할 수 있다. 그 중에서도 자원할당은 비즈니스 프로세스 모델링에 비교적 적은 관심을 받아 왔다. 여기서 자원(즉, 에이전트)의 예로 직원, 물리적인 장비, 업무, 업무를 성취하기 위한 작업 도구들을 포함한다. 이런 자원들에 비용(예산, 작업 시간)들을 작업의 실행과 연관시키는 것은 아주 전형적이다. 자원들은 공유될 수 없으며, 비용이 무한한 것은 아니다. 스케줄링은 언제 어떻게 자원을 사용할 것인지와 같은 결정을 수반한다. 비록 자원 관리는 비즈니스 프로세스의 중요한 측면으로 인식되어 왔음에도 불구하고 대부분의 작업은 다양한 자원을 모델링하는데 초점을 맞추어 왔지 자원 할당과 연관된 제약에서의 스케줄링에 대한 관심은 적은 편이었다.[2]

여기서 고려되는 자원할당 제약들을 간단한 예시로서 나타내어 보면, 실행해야 하는 task가 셋이 있다. task2는 task1이 실행되고 난 이후에 실행 변화할 수 있으며, task1이 실행되고 난 후에는 task2와 task3이 실행해야 한다.(단 task3은 task2 다음에 실행 변화 가능하다.) 업무 프로세스에서 자원할당의 제약은 다음 형식을 따른다. 만약 task1이 어떤 자원(에이전트)에 의해 실행된다면 task2 또한 같은 에이전트에 의해 수행 되어져야 하며 총 시간 또는 총 비용은 명확한 제약을 초과해서는 안 된다는 조건을 가진다. 다른 전형적인 자원할당 제약은 같은 자원(에이전트)들은 프로세스의 병렬 분기에 할당될 수 없으며, 또 기계는 동시에 다른 작업에 사용되어 질수가 없다. 여기서 자원할당 제약을 두 개의 항목으로 나눈다. 첫째로 비용제약(Cost Constraint)은 총 금액의 함수에 제약이 있다. 둘째로 조정제약(Control Constraint)은 어떻게 자원을 할당할 것인지를 제약한다. 비용제약은 프로세스의 결과 주어진 총금액(총시간)을 초과하지 않는다는 것이며, 조정제약은 task1이 어떤 에이전트에 의해 실행되어지면 후에 task2가 같은 에이전트에 의해서 실행되어야 한다는 것이다. 본 연구에서는 First order logic 기반의 CTR(Concurrent Transaction Logic)을 이용하여 각 비즈니스 프로세스 스케줄링을 위한 자원할당에 따른 제약들을 논리적인 모형으로 구체화하고, 모형에 필요한 변환 규칙을 소개하고자 한다.

2. CTR과 논리적인 표현

CTR은 유용한 모델링과 추론도구들을 보여 준다. 특히 비즈니스 프로세스에 대해 모델링하고 추론하는데 아주 좋은 공식들 중의 하나이다.[3][4] CTR은 정형적인 로직을 몇 개의 새로운 결합으로 양식을 확장 표현한다. 여기서 가장 중요한 양

식은 ⊗(직렬 결합)과 |(병렬 결합)이다. CTR은 다중경로(multi-path, m-path) 기반이다. path는 한 개 또는 그 이상의 데이터베이스 상태($<\sigma_1,\sigma_2\cdots\sigma_n>$)의 연속이고, m-path는 path의 연속이다.($<\rho_1,\rho_2\cdots\rho_n>,\rho_i$:path) m-path를 따르는 실행이라는 것은 m-path 전체에서 true가 되는 것과 동일하다. 비공식적으로 만일 ∅가 m-path $\Pi_1(\Pi_1 \vDash \varnothing)$에서 true이고 ψ가 Π_2에서 true 라면 $(\Pi_2 \vDash \psi)$, $\varnothing \otimes \psi$는 전체적으로 연결된 m-path $\Pi_1 \bullet \Pi_2$가 true이다. 마찬가지로 $\varnothing | \psi$는 전체 m-path $\Pi_1 \| \Pi_2$에서 true이다. $\Pi_1 \| \Pi_2$는 Π_1과 Π_2를 포함하는 일련의 path들의 어떤 인터리빙에 의한 Π_1과 Π_2로부터 얻어지는 m-path이다.[3][4]

다음은 어떻게 CTR이 비즈니스 프로세스 논리적인 모형에 사용되는지를 알아보자. 예로 업무 프로세스의 job이 4개가 있다.(job a, b, c, d) 그리고 두 개의 선행관계가 있다. job a는 job b가 시작되기 전에 끝나야 하고, job c는 job d가 시작되기 전에 끝나야 한다. job a와 job b, job c와 job d는 서로 간에 직렬 결합 즉 job a와 job c가 true의 실행을 하여야 job b와 job d를 실행할 수 있기 때문이다. job a, b와 job c, d는 병렬 결합니다. 이를 CTR의 논리적인 표현은 아래와 같으며, 프로세스 수행을 위한 제약 조건을 가지게 된다.

$$((a \otimes b) \mid (c \otimes d))$$

이 job들을 수행하기 위한 자원할당의 제약들은 다음과 같다.

- 총 시간은 주어진 작업시간보다 작아야만 한다.
- 동시 분기들에 할당된 자원들은 공통원소를 갖지 않아야만 한다.

3. 분할스케줄(Partial Schedule)

CTR의 논리적인 확장을 하기 위해서는 다중 경로 스케줄링보다 분할 스케줄(Partial schedule)들에 기반을 두고 정의하여야 한다. 왜냐하면 m-path들은 직렬모델이기 때문에 동시 실행에는 적합하지만 자원의 요구들을 모델하기에는 충분하지가 못하다. 분할 스케줄은 두 개의 오퍼레이터 용어로 정의 된다. 첫 번째($\bullet_p$)는 연결을 의미하고 조합적이며, 두 번째($\|_p$)는 병렬 연결하고, 조합적이며 교환 가능하다. 분할 스케줄은 다음과 같이 정의된다.

- m-path(Π)는 partial schedule
- 두 분할스케줄의 직렬 연결 : $w_1 \bullet_p w_2$
- 두 분할스케줄의 병렬 연결 : $w_1 \|_p w_2$

4. 자원할당과 제약(Constraint) 이론

앞에서 언급한 두 제약의 타입을 소개하고, CTR 공식들에 자원들을 어떻게 연관시키는지를 알아보자. 첫 번째 비용제약(Cost Constraint)인데, 실행 스케줄들에 정의 되어진 총 비용 함수들을 포함한다. 두 번째로 조정제약(Control Constraint)인데, 자원이 다른 공식들에 할당될 수 있는 방법을 제한한다. 앞서 자원의 할당은 분할스케줄을 자원의 집합으로의 맵핑이다. 자원할당 asg(w)는 다음 조건을 만족해야 한다.

- $asg(w_1 \|_p w_2) = asg(w_1) \cup asg(w_2)$
- $asg(w_1 \bullet_p w_2) = asg(w_1) \cup asg(w_2)$
- asg(w) = S (S : 자원의 일부 집합, w : m-path)

4.1 비용제약(S_{cost})

S_{cost}는 cost_constraint(w, asg)의 술어들로 구성된다. 여기서 w는 분할스케줄, asg는 자원할당이며, cost_constraint(w, asg)는 value_constraint(cost(w, asg))를 가진다. w_1과 w_2를 $cost(w_1, asg)$와 $cost(w_2, asg)$로 정의되는 partial schedule로 놓자.

- $cost(w_1\|_p w_2,asg) \equiv op_|(cost(w_1,asg), cost(w_2,asg))$
- $cost(w_1\bullet_p w_2,asg) \equiv op_\otimes(cost(w_1,asg), cost(w_2,asg))$

4.2 조정제약(S_{ctrl})

S_{ctrl}는 ctrl_constraint(w, asg) form의 술어들로 구성된다. w_1, w_2, w_3을 partial schedule로 놓고, asg는 $asg(w_1)$, $asg(w_2)$, $asg(w_3)$과 같은 할당으로 정의된다.

- $ctrl\_constraint(w_1\|_p w_2, asg) \equiv$
  $set\_constraint_|(asg(w_1),asg(w_2)) \wedge$
  $ctrl\_constraint(w_1,asg) \wedge ctrl\_constraint(w_2,asg)$
- $ctrl\_constraint(w_1\bullet_p w_2, asg) \equiv$
  $set\_constraint_\otimes(asg(w_1),asg(w_2)) \wedge$
  $ctrl\_constraint(w_1,asg) \wedge ctrl\_constraint(w_2,asg)$
- ctrl_constraint(w,asg) ≡ leaf_constraint(asg(w))
  (w : m-path)

4.3 제약들의 함의

비용제약에서 cost() 함수는 할당된 cost로 반환되어야 한다. 실례로 기간(time, V_1)과 총액(amount, V_2)을 cost 함수의 리스트로 표현할 수 있다.

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

기간(V_1)은 제약(c_1)을 초과해서는 안 되고, 총액(V_2)은 제약(c_2)를 초과해서는 안 된다.

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

함수 $op_|$와 $op_\otimes$는 할당된 비용이 어떻게 합계가 되는지를 정의한다. 예를 들어 병렬실행에서는 기간의 최대값이 사용되며, 반면에 총액은 더해진다.

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

조정제약에서 스케줄의 병렬분기에 할당된 자원들은 공통원소를 갖지 않아야 한다.

$set\_constraint_|(R_1, R_2) \equiv$
$(token\_of(R_1) \cap token\_of(R_2)) = \varnothing)$

5. CTR 기반의 논리적인 변환규칙

변환 단계는 비즈니스 프로세스 스케줄을 표현하는 메인이라고 할 수 있다. 또한 논리적인 모형의 선행모형을 생성하기 때문에 변환을 다른 말로 선행 스케줄이라고 말할 수 있다. 기존의 프로세스를 새로운 프로세스 구조로 변환을 하여도 의미(semantic)는 포함하며, 또한 제약들도 포함한다. 변환에서의 기본적인 규칙은 (그림 1)과 같다.

$$B(G_1 \vee G_2) = B(G_1) \vee B(G_2)$$

$$B(G) = R(G, T) \otimes cost\_constraint(T) \otimes ctrl\_constraint(T)$$

$$R(A, T) = A \otimes (T = resource\_asg(A, Agents))$$

$$R(G_1 | G_2, T) = (T = '|'(T_1, T_2)) \otimes (R(G_1, T_1) | R(G_2, T_2))$$

$$R(G_1 \otimes G_2, T) = (T = '\otimes'(T_1, T_2)) \otimes (R(G_1, T_1) \otimes R(G_2, T_2))$$

$$R(G_1 \vee G_2, T) = R(G_1, T) \vee R(G_2, T)$$

(그림 1) 기본적인 변환규칙

프로세스 G를 가지는 Operator B는 다른 Operator R을 유도한다. 여기서 cost_constraint는 총 비용의 제약이며, ctrl_constraint는 자원할당을 제어한다. (그림 2)는 앞 2장의 예시를 기본적인 변환규칙을 적용한 선행스케줄링 변환 결과를 보여 주고 있다.

step 1	$B((a \otimes b) \mid (c \otimes d))$
step 2	$R((a \otimes b) \mid (c \otimes d), T) \otimes$ $cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 3	$((T = '|'(T_1, T_2)) \otimes$ $(R((a \otimes b), T_1) \mid (R(c \otimes d), T_2))) \otimes cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 4	$((T = '|'(T_1, T_2)) \otimes$ $((((T_1 = '\otimes'(T_3, T_4)) \otimes (R(a, T_3) \otimes R(b, T_4))) \mid$ $((T_2 = '\otimes'(T_5, T_6)) \otimes (R(c, T_5) \otimes R(d, T_6))))$ $\otimes cost\_constraint(T) \otimes ctrl\_constraint(T)$
step 5	$((T = '|'(T_1, T_2)) \otimes$ $((T_1 = '\otimes'(T_3, T_4)) \otimes$ $((a \otimes T_3) = rsrc(a, W)) \otimes (b \otimes (T_4 = rsrc(b, X))))) \mid$ $((T_2 = '\otimes'(T_5, T_6)) \otimes$ $(c \otimes T_5 = rsrc(c, Y)) \otimes (d \otimes T_6 = rsrc(d, Z)))) \otimes$ $\otimes cost\_constraint(T) \otimes ctrl\_constraint(T)$

(그림 2) 변환규칙을 통한 선행 스케줄링 단계별 처리

변환규칙의 제약조건에 적용될 cost constraints의 *placeholders*는 (그림 3)과 같이 정의할 수 있다.

$resource\_asg(T, Agents) : rsrc(T, A)$
$cost\_of(rsrc(T, A), V) : -duration(T, A, V)$
$value\_constraint(T) : -T < c$
$set\_constraint_{|}(T_1, T_2) : -disjoint(T_1, T_2)$
$set\_constraint_{\otimes}(T_1, T_2) : -true$
$op_{|}(V_1, V_2, V) : -V\ is\ \max(V_1, V_2)$
$op_{\otimes}(V_1, V_2, V) : -V\ is\ V_1 + V_2$
$laef\_constraint(T) : -true$

(그림 3) 제약조건의 *placehoders*

변환규칙 적용결과 논리적인 비즈니스 프로세스의 자원할당 및 제약조건의 구체적인 명세가 (그림 4)와 같은 결과로 표현될 수 있다.

basic predicates	$duration(a, W, V_1), duration(b, X, V_2),$ $duration(c, Y, V_3), duration(d, Z, V_4)$
cost constraints	$V_5 is\ V_1 + V_2,\ V_6 is\ V_3 + V_4,$ $V\ is\ \max(V_5, V_6),\ V < c$
control constraints	$W \neq Y,\ W \neq Z,\ X \neq Y,\ X \neq Z$

(그림 4) 자원할당 및 제약조건

6. 결 론

비즈니스 프로세스에서의 스케줄링은 정확한 실행의 순서를 찾는 문제이며, 비즈니스 로직을 포함하는 제약을 따르는 것이다. 대부분의 스케줄링은 정확한 업무를 명세하기 위한 시간적인 제약들에 집중되어 왔다. 본 논문에서는 비즈니스 프로세스 자원할당 제약에서의 스케줄링을 위한 논리적인 모델을 제시하였다. CTR 기반이며 예제의 간단한 프로세스를 변환규칙에 적용 최종적으로 논리적인 자원할당 및 제약조건의 구체적인 명세의 결과를 얻을 수가 있었다. 모든 실행은 제약의 만족을 보장해야 한다.

향후 비즈니스 프로세스 관리 모니터링에서의 시간, 비용 관점에서 프로세스 분석과 운영에 있어 동적인 작업관리 및 분배 등의 논리적인 모형을 만들어 연계하는 연구에 기대가 된다.

[Acknowledgement]
본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음.

참고 문헌

[1] D. Hollingsworth, "Workflow management coalition specification:the workflow reference model", WfMC specification, 1994.
[2] 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.
[3] H. Davulcu, M. Kifer, C.R. Ramarkrishnan, and I.V. Ramakrishan. Logic based modeling and analysis of workflows. In ACM Symposium on Principles of Database System, page 25-33, Seattle, Washington, June 1998.
[4] A.J. Bonner and M. Kifer. Concurrency and communication in transaction logic. In Joint Int'l Conference and Symposium on Logic Programming, pages 142-156, Bonn, Germany, September 1996. MIT Press.
[5] G. Alonso, D. Agrawal, A. 띠 Abba0ł, and C. Mohan. Functionality and limitations of current workflow management systems. In IEEE-Expert. Special issue on Cooperative Information Systems, 1997.
[6] C.Schulte and G. Smolka. Finite domain constraint programming in oz. a tutorial. Version 1.1.0, February 2000.
[7] P. Halsum and H. Geffner. Heuristic planning with time and resource. In IJCAI Workshop on Planning with Resources, Seattle, USA, August 2001.