HWAHAK KONGHAK Vol. 37, No. 6, December, 1999, pp. 870-876
(Journal of the Korean Institute of Chemical Engineers)

회분공정의 안전성 평가를 위한 모델 검증 기법의 적용

이한용 · 김진경 · 문 일†

연세대학교 화학공학과
(1999년 4월 14일 접수, 1999년 7월 30일 채택)

Application of Model Verification Technique for Safety Evaluation in Batch Process Schedules

Hanyong Lee, Jinkyoung Kim and Il Moon†

Dept. of Chem. Eng., Yonsei University, Seoul, Korea
(Received 14 April 1999; accepted 30 July 1999)

요 약

갠트차트로 표현되는 회분공정의 운전절차를 검색하여 안전성과 실현가능성을 확인하는 방법을 개발하였다. 본 연구에서는 처음으로 SMV(Symbolic Model Verifier)를 이용하여 갠트차트를 표현하는 방법을 두 가지로 제시하고 이를 비교·분석하였다. 시간에 대한 표현 방법에 따라 실시간 변수(real time variable)를 이용하는 방법과 래치 변수(latch variable)를 이용하는 방법의 두 가지로 알고리듬을 제시하고 이 두 가지 SMV 알고리듬의 효율성을 비교한 결과, 실시간 모델(real time model)이 검색시간과 검색해야 할 가지의 수에서 상대적으로 우수한 것으로 나타났다. 또한 SMV의 질의어를 통해 갠트차트로는 표현하기 어려운 공정 또는 생산품의 특성에 관련한 오류를 찾아낼 수 있었다. 이를 통해 보통의 갠트차트로는 찾기 어려운 회분식 생산공정의 오류들을 자동으로 검색할 수 있는 방법을 제시하였다.

Abstract – A new algorithm is proposed to detect errors in batch process schedules using SMV(symbolic model verifier). Two models, real time model and latch model, are developed to capture information on Gantt charts. The real time model is proved to be more efficient than the latch model. In addition, they can find errors related to the characteristics of the process and the product, which are difficult to be detected only by Gantt chart. It proves that the proposed algorithm can automate the procedure of detecting errors in batch process schedules.

Key words: Symbolic Model Verifier, Batch Process Schedule, Real Time Model, Latch Model, Gantt Chart

1. 서 론

최근의 유사한 공정을 가지는 제품들에 대하여 하나의 회분공정을 통하여 생산하는데 있어 그 장점을 최대한 발휘할 수 있도록 하는 회분식 공정의 생산일정 및 계획에 대한 활발한 연구에도 불구하고[1, 2] 주어진 회분공정의 생산일정에 대한 안전성에 대한 연구는 전무하다. 따라서 본 연구에서는 잘못된 생산계획으로 인한 사고들을 미리 방지하고자 오류들의 다양한 형태를 SMV를 통해 검출하고자 하였다. 회분식 공정은 특성상 복잡하며 생산계획과 실제 생산상의 불일치로 인한 외란이 발생하기 쉬운 단점이 있다[3]. 생산계획 또한 이러한 외란을 고려하여 정확하게 만들어져야 하므로 복잡한 양상을 띠게 되었다. 따라서 점차로 복잡해지는 생산계획은 이전보다 다양한 오류에 대한 가능성을 내포하고 있다. 이러한 오류들은 크게 나누어 실행가능성에 대한 오류, 안전문제와 관련한 오류 등으로 나누어 볼 수 있으며 제품들의 특성상 고려되어야 하는 다양한 생산방법에 따른 오류들은 단순히 갠트차트를 통해서는 찾기가 어려운 문제이다. 이러한 오류들에 의하여 나타날 수 있는 결과는 생산품의 부적절한 생산에 의한 경제적인 영향 뿐 아니라 생산공정과 그에 참여하는 생산자들, 또한 넓게는 생산공정과 관련된 지역의 모든 사람들에게까지 위험을 미칠 수 있는 안전에 관련된 영향을 포함한다.

본래의 SMV(Symbolic Model Verification)는 컴퓨터 프로토콜간의 통신상 오류를 검출하고자 개발된 검증기법이다. 과거에 SMV를 화학공정에 응용하여 회분공정상의 위기상황 및 이에 따른 제어기들의 상태를 분석하고자 하는 연구는 있었으나 본 연구에서는 처음으로 이를 생산일정에 관한 연구에 접목하였다. 이는 SMV가 본래 프로토콜간의 통신 일정상 오류를 검출하기 위하여 만들어진 점을 고려한다면 본래의 목적에 부합되는 응용일 것이다. SMV를 이용하여 생산일정을 표현하는데 있어 래치 변수(latch variable) 및 실시간 변수(real time variable)의 도입을 통하여 시간에 따른 연속적인 상태의 변화에 대하여 SMV에 의한 화학공정의 표현 방법을 확장하였다. 이러한 과정을 통하여 SMV를 이용한 생산일정상의 오류 검색 알고리듬을 제시하고 자동적으로 주어진 생산일정의 안전성(safety) 및 실현 가능성

†E-mail: ilmoon@bubble.yonsei.ac.kr

(feasibility)에 대한 평가를 수행하였다.

2. 회분공정의 중간저장 탱크 전략과 SMV의 이론적 배경

2-1. 회분공정의 중간저장 탱크 전략

일반적인 회분공정에서의 생산계획에 관련된 주제들은 고려되는 공정형태에 따른 분류와 생산계획의 목적에 따른 분류, 문제를 풀기 위한 접근방법에 따른 분류로 나눌 수 있다. 그리고 이들 분류에 따른 주제들이 복합적으로 구성되어 생산계획에 관련한 연구들이 수행되고 있다. 즉, 경험법칙 뿐 아니라 LP, NLP, MILP, MINLP 등의 수학적 기법 등을 활용하여 회분공정 장치들의 작업의 할당이나 배열, 재고의 최소화 등을 통해 이윤극대화를 도모한다. 이러한 생산계획의 전략에는 중간탱크 사용법과 가동시킬 장치의 결정 방법, 적절한 회분의 순서 결정법 등이 있다. 회분공정의 중간저장 탱크 전략은 중간저장 탱크의 존재여부 및 크기 뿐 아니라 생산되는 제품들의 특성도 고려되어야만 적절한 생산계획을 세울 수 있을 것이다. 회분공정의 중간저장 탱크의 활용전략은 다음과 같이 구분할 수 있다[4].

- UIS(Unlimited Intermediate Storage Policy)
- NIS(No Intermediate Storage Policy)
- FIS(Finite Intermediate Storage Policy)
- ZW(Zero-Wait Policy)
- MIS(Mixed Intermediate Storage Policy)

이러한 공정의 형태들은 공정의 효율성 뿐 아니라 생산되는 제품의 종류와 상태, 각 조업 단계의 안전성을 고려하여 적절하게 조합되어져야 한다. 따라서 안전성 제고를 위한 검증기법상에서도 모두 표현되어질 수 있어야 한다. 본 연구에서는 이러한 여러 가지 공정형태를 표현할 수 있고 빠르고 효율적으로 검색할 수 있는 도구로서 SMV를 이용하였다.

2-2. SMV(Symbolic Model Verification)

Clark(1986) 등은 모델 기반 검증 방법(Model-based Verification Method)을 개발하였다. 이 방법은 일반적인 이진 연산 논리에 시간의 개념을 표시할 수 있는 연산자가 추가된 형태의 temporal logic을 이용함으로써, 완전한 모사를 위하여 검색하여야 할 상태 공간의 증대속도를 감소시켰다[5]. 이를 이용하여 복잡한 마이크로프로세서 내·외부의 데이터 전송일정에 따른 프로토콜간의 오류가능성을 검증하였다. Moon(1991) 등은 이를 화학공정에 접목시켜 이산 화학공정 제어 시스템의 안전성과 운전성을 자동으로 검증하는 방법을 개발하였다. 이 연구에서는 순차 제어 시스템의 안전성을 검증하기 위하여 전체 상태 전이 그래프(State Transition Graph, STG)를 생성한다. 하나의 상태는 유한개의 상태 변수들로 정의되는데, 참이나 거짓의 논리 값을 갖는다. 이 논리 값들로 온-오프 밸브, 펌프, 탱크 수위 등의 이산 상태 값을 나타낼 수 있다[6]. McMillan(1992)은 논리 검증법(Symbolic Model Verification, SMV)을 제시하였는데, 직접 상태의 그래프를 생성하는 대신 이진 표현식으로 공정 상태들의 집합과 상태간의 전이들을 표현함으로써, 이전의 연구에서 문제점이었던 "상태 폭발 문제(state explosion problem)"을 크게 감소시켰다. 이 방법은 순차이진 결정도(Binary Decision Diagrams, BDD)를 이용하여 이진표현식을 구현하였는데, Burch(1991) 등은 SMV를 사용하여 검증할 수 있는 시스템의 상태 수가 10^{120}이상이 됨을 보였다[6]. Moon(1992)은 SMV를 화학공정에 적용하여 PLC 기반 시스템의 안전성과 운전성을 검증하는 방법을 연구하였다. 이 연구에서 PLC의 RLL을 모델로 변환하는 방법을 제시하였고 단위 공정에 대한 모델을 개발하였다[7]. Jeong (1995)은 이러한 공정 안전성 검증의 자동화를 위한 통합 환경을 X window 기반의 환경에서 개발하였다. 이것은 사용자 환경을 강조하여 보다 사용을 용이하게 하였다[8]. Probst(1996)는 이러한 단위 공정들을 모듈로 표현하여 이러한 모듈들의 조합으로써 전체 공정에 접근할 수 있도록 하였다. 또한 이러한 것들을 고체 운송 체계나 누출 검사, 가열로의 일반적 모델 등에 적용하였다[9].

SMV는 CTL(Computational Tree Logic)이라는 논리와 BDD를 이용하여 주어진 논리의 참과 거짓을 판별하는데 이것을 회분식 공정의 생산일정에 응용하여 회분식 생산공정의 오류가능성에 관한 논리의 참, 거짓을 판별하는데 이용하였다. SMV는 BDD를 사용하기 때문에 이진 트리(binary tree)의 모든 상태(state)를 검색하지 않고 이것을 축소시키거나 반복되는 구간(loop)을 찾아 간략화하여 검색하게 되었다. 이러한 원리를 이용하여 발생할 수 있는 모든 경우를 검색함으로써 논리적으로 유추해 내기 어려운 경우라도 모두 검색할 수 있다는 장점이 있다[10].

SMV를 이용하여 회분식 공정의 생산일정을 각 상태에 대한 식으로 표현하고 이들 식에 대한 검증 결과를 이용하여 생산일정상의 가능한 여러 가지 오류의 가능성을 확인하고 오류가 발생할 수 있는 부분을 파악할 수 있다. 이러한 결과를 보고 생산일정에서 수정해야할 부분을 찾아내고 이를 다시 검색을 함으로써 복잡한 회분식 생산일정의 외란 및 오류가능성을 줄일 수 있다. 이것은 회분식 공정의 생산일정 계획시에 오류의 가능성을 검증할 경우 효과적이다. 또한 이러한 검증법을 사용하는 가장 큰 장점은 작성된 모든 경우를 검색하여 주기 때문에 신뢰도를 향상시킬 수 있는 장점이 있다.

기존의 SMV에서는 회분공정의 안전을 검색하기 위하여 시간에 따른 공정의 조작을 조건화하여 표현하였다. 이러한 방법을 통하여 회분공정과 그에 부가된 여러 가지 제어기들의 작동상태 및 이상거동 여부를 검증하는데 SMV를 이용하였다. 하지만 본 연구에서는 SMV가 기존의 마이크로프로세서의 통신일정에 관련한 내용을 검증하려는 의도로 개발된 점을 중시하고 이를 화학공정에 응용하여 회분공정의 생산일정 계획에 대한 오류가능성을 검증하는데 이용하였다.

3. SMV를 이용한 회분공정의 생산일정 검색

회분식 공정의 생산일정은 일반적으로 갠트차트로 표현된다. 갠트차트는 손쉽게 생산일정을 각 단계별, 시간별로 표현할 수 있는 장점이 있는 반면, 화학공정 상에서 발생하는 복잡한 오류에 대한 가능성을 찾아내기는 쉬운 일이 아니다. 따라서 본 연구를 통하여 우선 일반적으로 손쉽게 마련할 수 있는 갠트차트를 SMV를 이용하여 구현하고 이를 바탕으로 화학 공정상 발생할 수 있으면서도 갠트차트로는 손쉽게 찾기 어려운 몇 가지 오류의 가능성에 대하여 검증하는 방법을 연구하였다.

3-1. SMV를 이용한 갠트차트의 구현

갠트차트는 앞서 설명한 바와 같이 일반적으로 회분공정의 생산계획을 세우는데 일반적으로 사용되는 방법 중의 하나이다. 따라서 본 연구에서는 첫 번째로 갠트차트를 SMV로 표현하는 방법에 대하여 연구하였다.

대상 공정의 생산일정은 Table 1과 같이 한 회분(batch) 500 kg을 생산하기 위한 조업시간을 필요로 하는 회분식 공정의 갠트차트로 정의하였다.

대상공정은 생산물 A, B, C를 각각 두 회분씩 A→B→C→A→B→C의 순서로 생산한다. 여기에서 이 대상공정의 중간저장 탱크 전략은 FIS(Finite Intermediate Storage Policy)로, 첫 번째 단(stage)의 반응기는 조업시간의 단축을 위하여 반응기와 같은 용량의 저장 탱크

Table 1. Processing time for each product in the batch plant

(unit: hr)

Stage \ Product	A	B	C
Stage 1	3	1	2
Stage 2	4	5	2
Stage 3	2	4	5

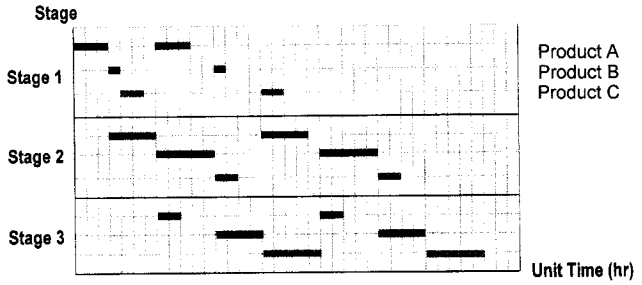

Fig. 1. Gantt chart of batch plant.

를 갖추고 있으며 모든 단의 반응기는 다음 반응이 개시되지 않는다면 반응 후에도 모든 중간생산물을 반응기 자체에 저장할 수 있다. 이러한 회분공정의 생산일정을 갠트차트로 나타내면 Fig. 1과 같다.

3-1-1. 실시간 변수의 SMV 표현

첫 번째 방법은 실시간 변수를 도입하는 방법이다. 이 방법은 전체 조업시간을 불연속적이고 동일한 시간의 구간으로 나누고 각각의 구간별로 각 단별 조업상태를 표현하는 방법이다. 따라서 전체 조업시간에 걸쳐 움직이는 Timer를 ProcessTimer={0, 1, 2,...Upper limit}모듈로 정의한다.

또한 시간이 지남에 따라 각 단에서의 변화는 Schedule 모듈로서 표현한다. Schedule 모듈은 ProcessTimer 모듈과 연결되어 갠트차트에서 나타나는 작업 대상 생산물과 각각의 생산물에 대한 작업시간을 반영한다. Schedule 모듈은 각 단별로 하나씩 정의되며 Schedule 모듈 내의 변수도 생산물의 종류를 모두 반영할 수 있도록 정의한다. Fig. 3에 나타난 Schedule 모듈은 A, B, C의 세 가지 생산물을 반영한 표준모듈이다.

```
MODULE ProcessTimer(Init,rate)
VAR
  t : 0..220 ;
  End : boolean ;
DEFINE
  Upper := 220 ;
ASSIGN
  next(t) :=
     case
       !Init : 0;
       (t+rate) < Upper : t + rate ;
       1 : t;
     esac ;
  next(End) :=
       case
          t=Upper : 1;
          1 : 0;
       esac ;
```

Fig. 2. Process timer module.

```
MODULE Schedule(PT,Run_time,A_On,B_On,C_On)
DEFINE
  Upper_time := 35 ;
VAR
  t : 0..35 ;
  condition : A_on,B_on,C_on,A_off,B_off,C_off,off ;
ASSIGN
  next(t) :=
       case
         (A_On | B_On | C_On) &(t<Run_time) : t + 1 ;
         (A_On | B_On | C_On) & t=Run_time : 0 ;
         (A_On=0 & B_On=0 & C_On=0) &(t>0) : 0 ;
         1 : t ;
       esac ;
  condition :=
       case
         A_On & (t<Run_time) : A_on ;
         B_On & (t<Run_time) : B_on ;
         C_On & (t<Run_time) : C_on ;
         A_On & (t=Run_time) : A_off ;
         B_On & (t=Run_time) : B_off ;
         C_On & (t=Run_time) : C_off ;
         1 : off ;
       esac ;
```

Fig. 3. Schedule module.

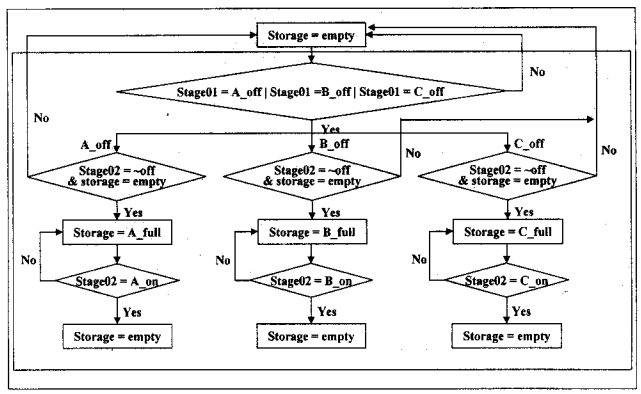

Fig. 4. Algorithm for storage tank.

시간의 함수는 ProcessTimer 모듈과 Schedule 모듈을 통해 모두 표현할 수 있다. 다음으로 만들어져야 하는 모듈은 중간저장 탱크 모듈과 각 단별 장치 모듈이다. 중간저장 탱크 모듈은 현재 단에서 생산된 생산물의 존재 여부와 다음 단에서의 조업가능여부에 따라 중간저장 탱크에 생산물을 저장할 것인가의 문제를 자동으로 판단할 수 있도록 제작하였다.

각 단의 장치 모듈은 조업시간에 따른 조업여부와 조업대상 생산물, 그리고 저장 탱크의 사용가능여부에 따라 장치의 조업상태를 판단하도록 제작되었다.

이렇게 제작된 ProcessTimer 모듈, Schedule 모듈, 저장 탱크 모듈, 장치모듈을 조합하여 하나의 주 모듈에서 연결시키고 그 안전성 여부를 검증하였다. 검증질의어는 주어진 생산방법에 따라 조업을 할 경우 저장 탱크내에 생산물이 저장되어 있는 상태에서 첫 번째 단의 조업이 끝나 다른 생산물을 공급받고 동시에 두 번째 단에서는 조업중인 상태가 어떠한 시간에라도 일어날 수 있는가를 검색할 수 있도록 하였다.

3-1-2. 래치 변수의 SMV 표현

두 번째 방법은 래치 변수(latch variable)를 도입하는 방법이다. 이 방법을 통하여 일련의 작업 내용을 정의하고 래치(latch) 함수를 도입함으로써 작업순서대로 각 단의 상태를 표현한다. 래치 함수는 다음 상태로 정의된 변수에 의하여 초기화될 때까지 현재의 상태로 정의

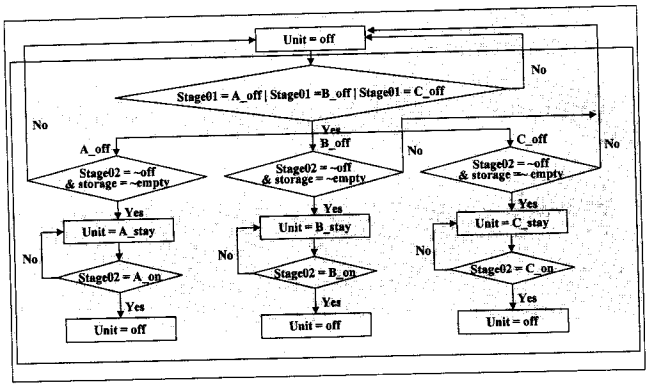

Fig. 5. Algorithm for each unit.

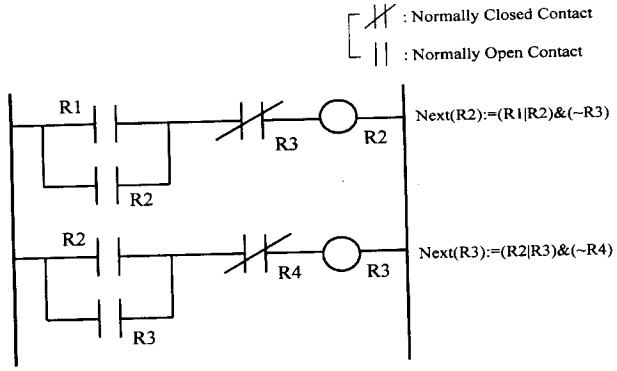

Fig. 6. RLL(relay ladder logic) for latch variables.

된 변수를 유지하는 함수를 말한다. 이것을 RLL(relay ladder logic)에 의한 표현으로 나타내면 Fig. 6과 같다.

Fig. 6에서 R1이 참값이 되면 다음 단계에서는 R2가 참값을 가지게 되며, 다시 그 다음 단계에서 R1은 거짓이 되고 R3은 참값을 가지게 된다. 이러한 래치 함수를 논리식으로 표현하면 그림에서 보인 바와 같이 $Next(R_i):=(R_{i-1}|R_i)\&(\sim R_{i+1})$이 된다. 래치 변수의 도입으로 SMV에서 순차적인 상태의 변화를 표현할 수 있다.

이 함수를 개량하여 또 하나의 precondition variable을 도입한다. 이 precondition variable은 시간적으로는 다르게 시작하지만 같은 조건을 가지는 두 개 이상의 상태를 구분하는 역할을 한다. 예를 들어 두 번째 시간에 첫 번째 단의 회분반응기가 A라는 반응물로 공정을 진행하고 같은 공정을 같은 단에서 네 번째 시간에서도 진행한다고 하면 세 번째 시간의 시작을 알리는 중간 변수가 다섯 번째 시간의 시작을 알리는 중간 변수와 같은 조건을 가지게 되어 동시에 참값을 가지게 된다. 따라서 이러한 중간 변수가 동시에 참값을 가지지 못하도록 하기 위하여 precondition variable P_i를 도입한다. 이렇게 변형

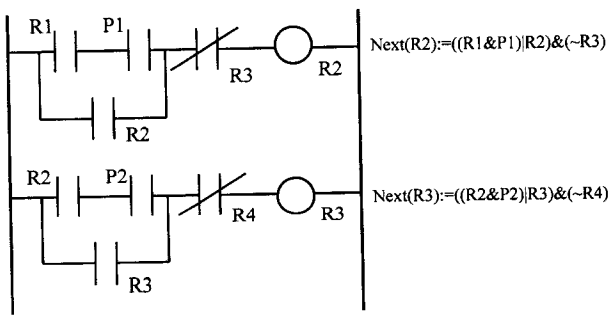

Fig. 7. RLL of latch variables with preconditions.

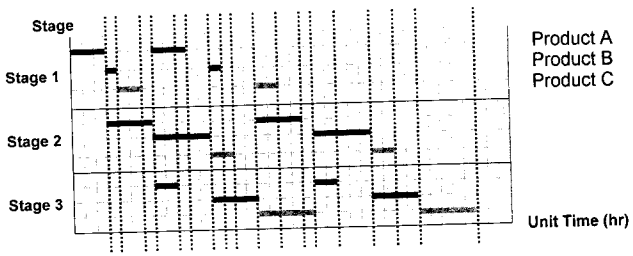

Fig. 8. Discretized time unit for the latch model.

```
next(R1) := ~R2 & (P=1 | R1);
next(R2) := ~R3 & ((R1 & P=2) | R2);
next(R3) := ~R4 & ((R2 & P=3) | R3);
                    ...
next(Ri) := ~Finish & ((Ri-1 & P=i) | Ri);

next(Stage1) :=
      case
            (R1 | R4 |···) & (R2 | R5···) : Operating_A ;
            (R11 | R14 |···) & (R12 | R15···) : Operating_B ;
            (R21 | R24 |···) & (R22 | R25···) : Operating_C ;
            1 : Stage1 ;
      esac ;

next(Stage2) :=
      case
            (R2 | R5 |···) & (R3 | R6···) : Operating_A ;
            (R12 | R15 |···) & (R13 | R16···) : Operating_B ;
            (R22 | R25 |···) & (R23 | R26···) : Operating_C ;
            1 : Stage2 ;
      esac ;

next(Stage3) :=
      case
            (R3 | R6 |···) & (R3 | R7···) : Operating_A ;
            (R13 | R16 |···) & (R13 | R17···) : Operating_B ;
            (R23 | R26 |···) & (R23 | R27···) : Operating_C ;
            1 : Stage3 ;
      esac ;

next(Storage) :=
      case
            (R5 | R8 |···) & (R6 | R9 |···) : 1 ;
            1 : 0 ;

next(P) :=
      P + 1 ;
```

Fig. 9. SMV representation of latch variables.

된 논리를 RLL로 나타내면 Fig. 7과 같다.

래치 변수를 도입하기 위해서는 회분식 공정을 시간에 따른 구간별 구분보다는 공정상태별로 구간을 나누는 것이 편리하다. 따라서 각 단별 반응물의 도입과 생성물의 완성을 기준으로 구간을 나누었다. Fig. 8은 가상 공정의 갠트차트를 설명한 기준에 의하여 구간별로 나눈 것을 보이고 있다.

위에서 설명한 precondition variable을 도입한 래치 함수를 공정상태에 의한 조건으로 나눈 구간에 도입하면 다음과 같은 SMV 알고리듬으로 나타낼 수 있다.

3-2. 생산일정의 오류 검색

실제로 두 모델이 생산일정의 오류를 정확히 찾아내는지 검증하기 위하여 앞에서 서술한 FIS정책을 가진 공정에 오류를 삽입하였다. 그 결과로 그려지는 갠트차트는 Fig. 10과 같다.

첫 번째 단에서 생산품 C에 대한 조업일정을 두 단위 시간 앞당김

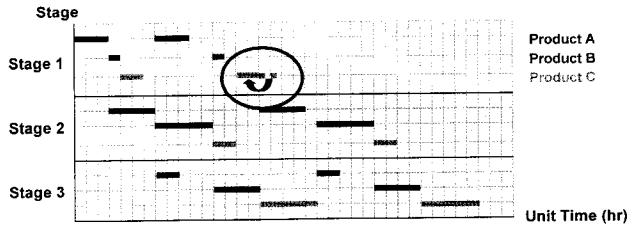

Fig. 10. Gantt chart imbedded in an error.

으로써 나타나는 결과는 FIS정책에 관련한 오류이다. FIS정책에 의하면 첫 번째 단에서 생산품 B를 생산한 후에도 두 번째 단에서 조업이 끝나지 않았을 경우에는 중간저장 탱크에 저장을 하거나 생산품을 체류시켜야 한다. 그러나 Fig. 10과 같이 일정상의 변화가 생겼을 경우, 중간저장탱크와 첫 번째 단의 반응기는 각각 중간생산품이 이미 차있는 상태에서 B를 생산하기 위한 원료물질이 첫 번째 단으로 도입되므로 생산일정은 실현이 불가능하다. 이러한 상태를 찾아내기 위한 질의 검증어는 실시간 모델에서 다음과 같은 CTL(computational tree logic)로 표현할 수 있다.

"첫번째 단의 반응기가 조업 중이고 중간저장 탱크는 저장한 중간생산품이 있는 상태와 첫 번째 단에 중간생산품이 그대로 남아있는 상태가 공존하는 경우는 없다."→

AG(!(!(stage01_schedule.condition=off) &
!(storage.condition=empty) & !(reactor.condition=0))) (1)

마찬가지로 래치 모델에서도 다음과 같은 CTL이 질의 검증어로 사용된다.

"첫번째 단의 반응기가 중간생산품을 저장하고 있는 상태와 어떠한 생산품이라도 만들기 위해 조업하는 상태가 공존하는 경우는 없다."
→ AG (!((Stage1_P=1) & ! ((Stage1=Operating_A) |
(Stage1=Operating_A) | (Stage1=Operating_B)))) (2)

여기서 Stage1_P는 첫 번째 단의 반응기가 반응물을 저장하고 있는 상태인가를 나타내는 이산변수이다. 즉, Stage1_P는 중간저장 탱크가 반응물을 저장하고 있기 때문에 첫 번째 단의 반응기가 반응물을 저장하고 있는 상태라면 1의 값을, 중간저장 탱크가 비워져 있어서 반응물을 중간저장 탱크로 넘겨주었거나 두 번째 단의 반응기가 계속해서 조업을 할 수 있는 상태이어서 반응물을 두 번째 단으로 넘겨줌으로써 0의 값을 가질 수 있는 변수이다.

3-3. 실시간 모델과 래치 모델의 비교

두 모델을 이용한 SMV 알고리듬을 바탕으로 주어진 갠트차트를 표현하는 프로그램을 제작하고 실행시켜본 결과, 각각 Fig. 11 및 12와 같이 모두 오류가 일어나는 경로를 성공적으로 찾아내었다. 그러

```
-- specification AG (!(!stage01_Schedule.condition = off ... is false
-- as demonstrated by the following execution sequence

state 1.1:
stage03_Off_time = 57
stage02_C_Off_time = 88
stage02_B_Off_time = 75
stage02_A_Off_time = 44
stage01_C_Off_time = 39
stage01_B_Off_time = 26
stage01_A_Off_time = 19
operator = 1
PTime.Upper = 220
PTime.t = 0
PTime.End = 0
stage01_Schedule.Upper_time = 35
stage01_Schedule.t = 0
stage01_Schedule.condition = off
stage01_Run_time = 18
stage01_A_On_time = 1
stage01_B_On_time = 20
stage01_C_On_time = 27
stage02_Schedule.Upper_time = 35
stage02_Schedule.t = 0
stage02_Schedule.condition = off
stage02_Run_time = 24
stage02_A_On_time = 20
stage02_B_On_time = 45
stage02_C_On_time = 76
stage03_Schedule.Upper_time = 35
stage03_Schedule.t = 0
stage03_Schedule.condition = off
stage03_Run_time = 12
stage03_On_time = 45
storage.condition = empty
reactor.condition = 0

state 1.2:
PTime.t = 1
stage01_Schedule.condition = A_on

state1.3:
PTime.t = 2
stage01_Schedule.t = 1
state 1.4:PTime.t = 3
stage01_Schedule.t = 2

~

state 1.85:
PTime.t = 84
stage02_Schedule.t = 8
stage03_Schedule.t = 8

state 1.86:
PTime.t = 85
stage02_Schedule.t = 9
stage03_Schedule.t = 9

state 1.87:
PTime.t = 86
stage02_Schedule.t = 10
stage03_Schedule.t = 10

state 1.88:
PTime.t = 87
stage02_Schedule.t = 11
stage03_Schedule.t = 11

state 1.89:
PTime.t = 88
stage02_Schedule.t = 12
stage02_Schedule.condition = C_off
stage03_Schedule.t = 12

state 1.90:
stage02_C_Off_time = 175
PTime.t = 89
stage01_Schedule.condition = C_on
stage02_Schedule.t = 0
stage02_Schedule.condition = off
stage02_Run_time = 24
stage02_C_On_time = 163
stage03_Schedule.t = 13

resources used:
user time: 19.1167 s, system time: 0.1 s
BDD nodes allocated: 88002
Bytes allocated: 2359296
BDD nodes representing transition relation: 34561 + 1517
```

Fig. 11. Results of real time model based calculation.

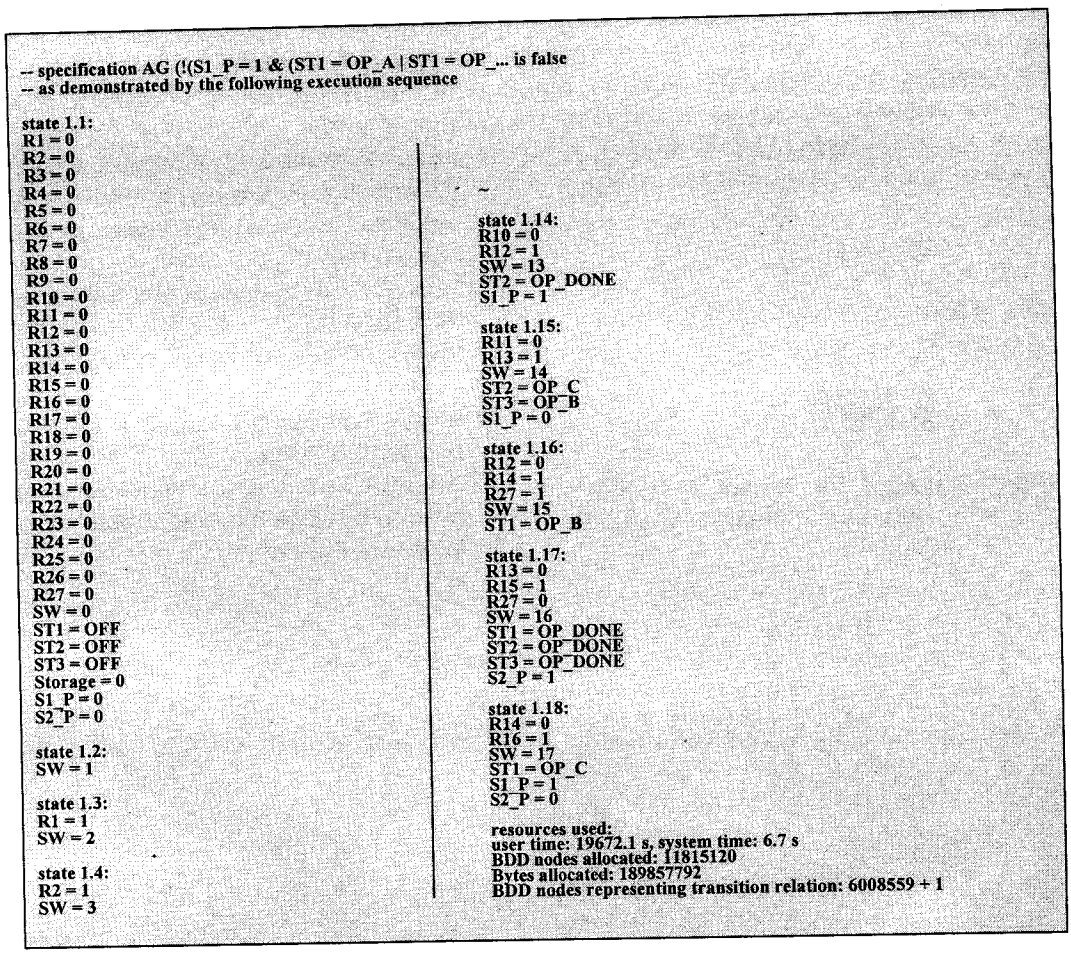

Fig. 12. Results of latch model based calculation.

Table 2. Comparison of real time model and latch model

Category \ Applied model	Latch model	Real time model
User time	19672.1 s	19.1 s
System time	6.7 s	0.1 s
BDD nodes allocated	11815120	88002
Bytes allocated	189857792	2359296
BDD nodes representing transition relation	6008560	36078

나 알고리듬에 따라 큰 탐색시간의 차이를 보였다. 그 이유는 갠트차트를 SMV로 표현하는 방법으로 제시한 두 가지 방법은 시간을 어떠한 방법으로 나누는가에서 차이점이 나타난다.

실시간 변수를 이용한 방법은 시간을 digital 시계와 같이 일정한 구간으로 나누고 상태를 규정하는 모든 변수들을 시간에 종속적으로 다른 값들을 가지도록 한다. 따라서 시간을 나타내는 변수에 모든 변수들은 종속적이다. 또한 회분공정을 하나의 갠트차트로 나타낼 수 있다면 전체 시스템의 실시간 변수는 전체 조업시간과 같은 값의 범위를 가진 하나의 변수로 표현할 수 있다. 그러므로 대상공정의 검색을 필요로 하는 대상시간이 길수록 BDD로 나타나는 가지의 수는 많아진다.

래치 변수를 이용한 방법은 상태의 변화가 없는 구간을 하나의 시간 구간으로 나눈다. 이러한 시간 구간에 따라 래치 변수에 종속적인 상태를 나타내는 변수를 변화시킨다. 그리고 각 시간 구간을 구분하기 위하여 precondition variable을 이용한다. BDD의 수는 실시간 변수를 적용한 방법과 마찬가지로 대상공정의 검색을 필요로 하는 대상시간에 따라 늘어난다. 본 연구에서 수행된 두 가지 방법에 의한 SMV프로그램을 동일한 예제에 적용시킨 결과 실시간 변수에 의한 방법이 훨씬 적은 BDD를 가지며, 연산 수행 속도 또한 상대적으로 우수한 것으로 나타났다.

Table 2에서 보인 바와 같이 컴퓨터가 실제로 연산을 수행하는데 소모된 시간을 의미하는 system time을 기준으로는 약 67배, 상태 전이의 가지수를 나타내는 BDD nodes representing transition relation을 기준으로는 약 167배, 실시간 모델이 래치 모델보다 우수한 성능을 가진 것으로 나타났다.

4. 결 론

이상과 같이 실시간 변수와 래치 변수를 도입하여 갠트차트에 나타난 회분식 공정의 생산일정을 SMV 알고리듬으로 각각 표현하였다. 그 결과 갠트차트에서는 표현하기 어려운 회분공정의 공정특성 및 생산물의 특성에 관련된 오류에 대하여 SMV를 통한 표현과 안전성 검증이 가능하였다. 그 절차로 실시간 변수를 도입한 모델과 래치 변수를 도입한 모델을 이용하여 갠트차트를 표현하였고 각각 이를 적용하여 프로그램을 제작하였다. 공정특성 및 생산물의 특성에 관련한 오류에 대한 안전성 검증은 CTL에 의한 검증질의어에 의하여 마련되었다. 본 연구를 통하여, 검증질의어는 장치용량에 관련한 오류에 대하여 검색하도록 마련되었으나, 검증질의어에 따라 예상되는 오류

에 대한 다양한 분석이 가능하다. 따라서 이러한 알고리듬을 통해 회분공정에 대한 오류 검색의 자동화 뿐 아니라 여러 가지 오류에 대한 통합적인 검색도 이루어질 수 있다. 동일한 대상공정에 대하여 적용한 결과, 실시간 모델과 래치 모델, 모두 검증질의어를 통하여 성공적으로 생산일정상의 오류를 찾아내었으나 검색해야 할 가지의 수 및 연산수행 속도 면에서는 실시간 모델이 효율적임을 알 수 있었다.

감 사

본 연구는 1999년도 한국과학재단의 한·영 국제공동연구과제에 의하여 이루어진 결과의 일부이며, 이에 감사드립니다.

참고문헌

1. Ko, D. and Moon, I.: *Comp. Chem. Eng.*, **21.suppl.**, 1067(1997).
2. Lee, H., Jung, J. H., Moon, I. and Lee, I.: *HWAHAK KONGHAK*, **35**, 599(1997).
3. Ko, D. and Moon, I.: *HWAHAK KONGHAK*, **35**, 338(1997).
4. Biegler, L. T., Grossmann, I. E and Westerberg, A. W.: "SYSTEMATIC METHODS OF CHEMICAL PROCESS DESIGN," Prentice Hall, 187(1997).
5. Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L. and Hwang, J.: "Symbolic Model Checking : 10^{20} States and Beyond," In Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science(1990).
6. Moon, I., Powers, G. J., Burch, J. R. and Clarke, E. M.: *AIChE J.*, **38**(1), 66(1992).
7. Moon, I.: *IEEE Control Systems*, **14**(2), 53(1994).
8. Jeong, S., Lee, K. and Moon, I.: *ICASE*, **2**(1), 53(1996).
9. Probst, S. T., Powers, G. J., Long, D. E. and Moon, I.: *Comp. Chem. Eng.*, **21**(4), 417(1997).
10. Probst, S. T. and Powers, G. J.: "Automatic Verification of Chemical Process in the Presence of Failure Modes," AIChE Annual Meeting, San Francisco, CA(1994).