실시간 프로세스의 스케줄가능성 분석을 위한 프로세스 상태 분석기 구현

o
서상진, 서동진, 최동한, 김춘배, 박홍복
부경대학교 전자계산학과

Implementation of Process States Analyzer for Schedulability Analysis of Real-Time Process

o
Sang-Jin Seo, Dong-Jin Seo, Dong-Han Choi, Choon-Bae Kim, Hung-Bog Park
Dept. of Computer Science, PuKyong National University

요 약

실시간 스케줄가능성 분석에 대한 기존의 방법들은 프로세스의 모든 상태공간을 추적하거나 고정 우선 순위 스케줄 방법을 사용하기 때문에 시간 및 공간에 대한 복잡성이 증가된다. 본 논문에서는 프로세스 대수의 전이규칙을 이용하여 프로세스의 최소 수행시간, 주기, 마감시간, 동기화 시간을 고려하여 실시간 프로세스가 마감시간을 지키는가를 판단하고, GUI 환경을 기반으로 스케줄이 불가능한 프로세스에 대해 스케줄 가능하게 하는 회복 알고리즘을 제안하고, 이들의 상태 공간을 화면상에 표시해 주는 실시간 프로세스의 상태 분석기를 구현한다.

1. 서론

실시간 시스템이란 시스템의 운영과 외부 세계와의 상호관계가 시간과 깊이 관련되어 있어 외부에서 발생한 요구에 대하여 제한된 시간 내에 정확하게 처리될 수 있도록 구축된 시스템이다. 대표적인 예로서, 핵 발전, 미사일 제어, 공장 자동화, 항공 교통 제어 등이 있으며, 마감시간 준수 여부를 판단하기 위한 시스템 기능의 수행 여부는 시스템 전체 운용에 따른 인력과 비용이 소요될 수도 있는 중요한 문제이다. 더욱이 시스템 작업 수행을 위한 프로세스가 마감시간 내에 스케줄 불가능할 때 발생하는 문제를 처리하기 위해서는 전체 번역시간에 대해 최소 수행시간으로 스케줄링하는 회복 알고리즘의 사용은 불가피하다.

스케줄가능성 분석(schedulability analysis)이란 프로세스가 특정 스케줄링 정책하에서 수행될 때 프로세스가 마감시간을 만족하는가를 판단하는 것이다[3,6,7]. Fredette[3]는 프로세스 대수를 이용하여 전체 프로세스 상태공간을 탐색하는 스케줄가능성 분석을 제안하였으며, Bruno 등은 페트리 넷(Petri Net)을 이용하여 주어진 시간 내에 전체 프로세스의 스케줄가능성을 분석하는 방법을 제안했으며[1], [6,7]에서는 스케줄 가능분석이 제안되었으나 스케줄 불가능할 때 회복기법은 제시되지 않았다. 또한, Liu와 Layland는 비율단조 스케줄링시 독립적인 프로세스에 대해 주어진 조건들에 대해 마감시간을 지키는가를 판단하는 방법을 제안하였다.

본 논문에서는 실시간 명세언어로 기술된 각 프로세스 정보를 입력덱으로 받아 기존에 제안된 프로세스의 스케줄가능성 분석[7]시에 스케줄이 불가능한 프로세스에 대하여 스케줄 가능하도록 하는 회복 알고리즘을 제안하며, 스케줄 불가능한 프로세스의 상태공간과 그 상태를 회복한 스케줄가능 상태의 상태공간을 선택적으로 화면상에 표시해 주는 프로세스 상태 분석기를 Visual C++5.0으로 구현한다.

2. 실시간 명세 언어

실시간 명세언어는 실시간 프로세스의 마감시간 만족 여부를 번역시간에 판단하기 위해 프로세스의 시간특성과 프로세스간의 관련성을 표현하는 명세언어이다.

2.1 프로세스 대수

프로세스 대수의 구문과 연산의미(operational semantics)는 다음과 같이 CCS[1,5]에서 찾아 볼 수 있다. 프로세스 term P는 그림 1과 같은 BNF 문법으로 주어진다.

P::=nil | X::=P | a.P | P+Q | P||Q

(그림 1) CCS의 구문

nil은 프로세스 종료를 나타내고, X::=P: 주어진 환경내 변수 X에 한정되는 프로세스 P를 의미한다. a.P는 동작 a를 수행한 후, 프로세스 P와 같이 동작하며, P+Q는 P 또는 Q 두 프로세스 사이의 선택을 나타내며, P||Q는 P와 Q 두 프로세스의 병렬 조합을 의미한다. 대문자는 프로세스, 소문자는 프로세스를 구성하는 동작이며, 두 프로세스는 독립적으로 동작을 수행할 수도 있고, 보동작(complementary action)으로 동기화 될 수 있다. 동작 a의 보동작은 $\bar{a}$로 표현된다.

2.2 명세언어 CCS-R

실시간 명세언어 CCS-R은 CCS에 시간 개념을 추가한 것으로 수행할 시간과 수행시간에 대한 제약을 갖는 실시간 프로그램의 특성을 추출하기 위한 실시간 명세언어이다. CCS-R로 기술된 프로세스는 수행을 위한 명령 동작들로 구성되며, 각 동작은 그 동작을 수행하는데 필요한 시간간격 d를 갖는다[1,6].

CCS-R 프로세스의 구문은 (그림 2)와 같이 BNF 문법으로 정의된다[3,6].

프로세스가 포함하는 시간제약과 시간특성은 시간 구성자 sw, fw, e로 표현된다. d를 마감시간이라 할 때, 마감시간은 프로세스가 시작된 시간으로부터 상대적으로 계산된다. sw(P, d)는 프로세스 P가 d 단위시간이내에 시작해야 한다는 것을 나타내며, fw(P, d)는 프로세스 P가 d 단위시간이내에 종료하는 프로세스로 주기 프로세스를 표현하는데 사용된다. 만약 P가 d′에 끝나고 d>d′이면, P는 단위시간동안 지연되어야 한다.

$$P::=nil|\ \gamma^{d}|\ \delta^{d}|\ a^{d}|P+Q|P;P|sw(P,d)|fw(P,d)|\ e(P,d)$$

(그림 2) CCS-R의 구문

nil은 프로세스 종료를 나타내고, γ^{d}는 d 단위시간이 걸리는 동작을 의미하며, δ^{d}는 d 단위시간 동안 지연하는 동작, a^{d}은 d 단위시간이 걸리는 동기화 동작을 나타낸다. P+Q는 P와 Q 프로세스 사이의 선택을 나타내며, P;P는 P와P의 순차조합을 의미한다. sw(P,d)는 d 단위시간 내에 시작하는 주기 프로세스를 나타내고, fw(P,d)는 d 단위시간 내에 종료하는 주기 프로세스를 의미하고, e(P,d)는 d 단위시간 내에 정확히 종료하는 프로세스를 나타내고 있다.

3. 스케줄가능성 분석 및 회복기법

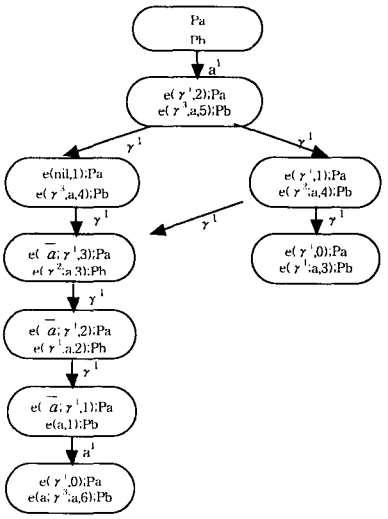
3.1 스케줄가능성 분석 방법

Fredette의 알고리즘은 (그림 4)와 같이 실시간 프로세스가 스케줄 가능한가를 판단하기 위해 생성되는 실시간 프로세스들의 모든 상태들 중 마감시간이 0인 프로세스를 포함하는 상태가 있는 지를 검사하여 마감시간이 0인 상태에 도달하는 프로세스가 있다면 스케줄이 불가능하다고 판단하는 것이다. 예를 들어, (그림 3)의 실시간 프로세스 Pa, Pb를 Fredette의 방법에 따라 스케줄가능성 분석을 하면 (그림 4)와 같다. 프로세스 P_a는 3 단위시간마다 채널 a에 메시지를 보내고, 1 단위시간 동안 수행한 후 다시 P_a를 수행하는 것이며, P_b는 6 단위시간마다 메시지를 채널 b에서 받고 3 단위시간 동안 수행하고 채널 b에 메시지를 받은 다음에 다시 P_b를 수행한다.

$$Pa::=e(\bar{a}^{1};\gamma^{1},3);Pa;$$
$$Pb::=e(a^{1};\gamma^{3};a^{1},6);Pb;$$
$$d(a)=d(\bar{a})=1$$

(그림 3) 실험용 프로세스

이들 프로세스는 어떤 경우에도 마감시간이 0인 상태에 도달하므로 단일 처리기 시스템에서는 마감시간을 지키지 못한다.



(그림 4) Fredette 분석을 통한 프로세스의 상태공간

그러나 본 논문에서는 동기화 동작을 수행하는 프로세스에 대하여 동기화 동작을 수행하기 위해 지나야 하는 시간, 동기화 시간, 동기화 동작을 수행한 후 나머지 동작을 수행하는 시간의 합이 한 프로세스의 마감 시간보다 길면 그 프로세스는 마감시간을 지키지 못한다는 것을 미리 판단하는 스케줄가능성 분석 알고리즘[6]을 이용하여 스케줄 불가능할 경우의 스케줄 회복 알고리즘을 제안한다.

3.2 회복 알고리즘

Liu와 Layland가 주기 실시간 프로세스의 집합을 처리할 능력이 있는가를 판단하기 위해 제시된 스케줄가능성을 위한 식(1)과 제시된 스케줄가능성을 판단하는 알고리즘을 이용하여 표 1의 처리기 할당 기준을 설정하고, 스케줄이 불가능한 경우, 스케줄이 가능하게 하는 회복 알고리즘을 제안한다.

$$\sum_{i=0}^{n}(Ci+\frac{Ai}{2})\times\frac{W}{Ti}\le W \quad \text{--- 식 (1)}$$

Ci는 프로세스 Pi의 계산시간이며, Ai는 프로세스 Pi의 동기화 시간이다. W는 프로세스 Pi의 주기의 최소 공배수이며, Ti는 프로세스 Pi의 주기이다.

표 1 처리기 할당 기준

식(1)	기존의 알고리즘	처리기 할당 기준
스케줄 가능	스케줄 불가능	프로그램 코드의 순서를 변경하여 단일 처리기에 할당한다
스케줄 불가능	스케줄 불가능	프로그램 코드의 순서와 마감 시간을 변경하고 프로세스를 여러 처리기에 분담하여 할당한다

식(1)과 스케줄가능성 분석 알고리즘을 통해 처리된 결과를 조합하여 적용될 회복기법을 선택할 수 있다.

알고리즘 1은 스케줄가능성을 분석한 후, 스케줄이 불가능한 경우, 알고리즘 2에서 식(1)을 이용하여 최선의 회복기법으로 스케줄 가능하도록 하는 회복 알고리즘을 제시한다.

```
[알고리즘 1] 식(1)의 결과가 스케줄 가능할 때 적용될 회복 알고리즘
  시스템 변수 초기화;
  명령어 순서 초기 조합;
  for(factorial(각 프로세스에 포함된 명령어 수 - 1))
    스케줄가능성 분석;
    if(스케줄 가능)
      회복 처리 작업 종료;
    end of if
    다음 프로세스 명령 순서 조합;
  end of for
```

```
[알고리즘 2] 식(1)의 결과가 스케줄 불가능할 때 적용될 회복 알고리즘
  시스템 변수 초기화;
  프로세스 수만큼 프로세스 처리기 할당;
  do
    실행 프로세스 결정;
    for(프로세스 수)
      switch(실행 프로세스의 명령어 코드)
        동기화 동작 일때:
          동기화 대상 프로세스 탐색;
          if(동기화 대상 프로세스가 존재할 때)
            동기화 동작 수행;
          else
            동기화 대기;
          end of if
        계산 동작 일때:
          계산 동작 수행;
        모든 명령어이 종료되었을 때:
          if(마감시간 == 1)
            프로세스 초기화;
          end of if
      end of switch
    end of for
      마감 시간 및 주기수 감소;
      if(수행될 작업시간>마감시간)
        스케줄 가능 플래그 = false;
        return
      end of if
  while(마감시간의 최소 공배수)
    스케줄 가능 플래그 = true;
  end of while
  return;
```

제시된 알고리즘 1과 2를 이용한 회복 처리가 실패할 경우 다시 마감시간을 1씩 증가시켜 가면서 회복 알고리즘을 재수행한다.

스케줄가능성 분석 및 처리된 결과는 다음 과정의 처리 혹은 출력을 위해 구조체에 저장된다. 본 논문에서 각 단계의 분석을 통해 구축된 구조체의 내용을 화면에 시각적으로 나타내는 화면 출력기법은 [알고리즘 3]과 같다.

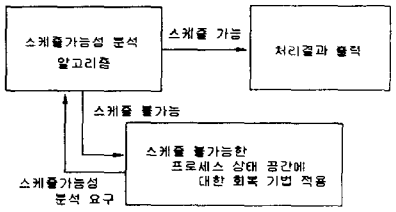
```
[알고리즘3] 화면 출력 알고리즘
디바이스 초기화 및 초기 출력 x, y 좌표값 설정;
 for(처리된 interval의 가중치)
    상태 도형 출력 위치 설정;
    if(초기 출력)
      시작 문자열과 화살표를 표시;
    end of if
    if(마지막 출력도형)
      if(스케줄 불가능하다면)
        반전된 상태 도형 출력;
        '*'문자 출력;
      else
        초기 상태 화살표 및 문자 출력;
    else
      상태 도형 및 해당 명령어 출력;
      다음 상태 지정 화살표 및 실행 명령어 출력;
    end of if
    다음 상태 출력을 위한 y축 좌표값을 보정;
 end of for
```

4. 프로세스 상태 분석기의 설계 및 구현

스케줄 불가능한 프로세스에 대하여 스케줄 가능하게 하기 위한 프로세스 상태 분석기를 설계하고, 회복 알고리즘을 구현한다.

4.1 프로세스 상태 분석기의 설계

프로세스 상태 분석기는 3개의 처리과정을 통해 목표로 하는 프로세스 상태공간을 표시한다. 그리고 프로세스 상태 분석기를 위한 일련의 분석 및 처리과정은 (그림 5)와 같다.



(그림 5) 프로세스 상태 분석기의 구성도

프로세스 상태 분석기는 실시간 명세언어를 입력덱으로 받아 처리한 후, 출력을 위한 구조체에 결과를 저장한다. 저장 구조체는 (그림 6)과 같은 인자를 가진다.

```
typedef struct NODE_LIST{
   int   Pi_Num;    //프로세스 번호
   char Pi[2];      //프로세스명
   OP_  OP_CODE[Max];//계산동작들의 집합
   int   St;        //슬랙
   int   Limit;     //제한시간
   char OP_array[MAX_OP_CODE_NUM];//코드 배열
   int   OP_index; //현재 수행중인 OP 코드의 위치
   int   OP_num;  //현재 가진 OP 코드의 수
   int   ct;        //총 계산시간(Cti)
   int   at;        //총 동기화 시간
}NODE_;
```

(a) 실시간 명세 언어 입력덱 저장 구조체

```
typedef struct ANALY_TABLE{
    int   A;  //프로세스의 동기화가 끝나는 시간
    int   B;  //동기화 동작 후, 남은 동작의 최소수행시간
    int   C;  //프로세스의 마감시간
    int   si;  //슬랙
    CString exec_code;//수행 명령어
    CString exec_flag; //명령어 처리 플래그
    CString OP_str;   //프로세스 상태공간 명령 문자열
    char  Pi[2];      //프로세스명
}ANALY_;
```

(b) 스케줄가능성 분석 테이블 저장 구조체

(그림 6) 저장 구조체

프로세스 상태 분석기는 이상의 구조체들을 통해 자료를 입·출력한다.

4.2 프로세스 상태 분석기의 구현

본 연구에서 구성된 알고리즘과 시스템 설계를 통해 (그림 8)과 같은 프로세스 상태 분석기를 구현하였다. 프로세스 상태 출력기는 입력덱 관리를 위한 추가 및 삭제, 선택 기능이 주어지고, 스케줄가능성 분석 기능과 회복 기능, 그리고 출력 기능을 개별적으로 분리함으로써 각 단계별 처리 상태를 자세히 관찰할 수 있도록 하였다. 그리고 처리 과정과 결과를 유도하는 과정에서 도출되는 유용한 정보는 Analysis Information 컨트롤과 Status Displayer 컨트롤에 텍스트로 출력되며, 프로세스 상태공간을 화면에 도시할 수 있다.

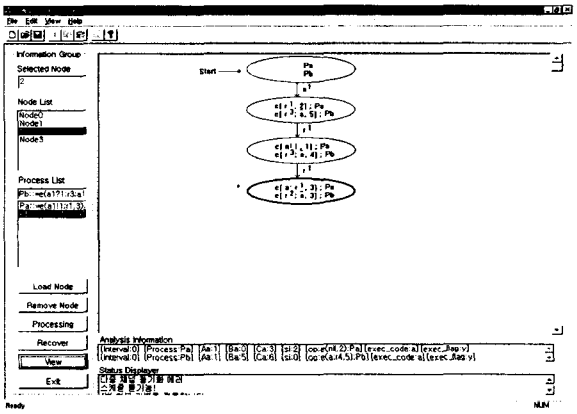
프로세스 상태 분석기의 기능 및 성능 평가를 위해 (그림 3)의 실시간 프로세스를 아래와 같이 실험용 실시간 프로세스로 바꾸어 적용한다.

```
Pa::=e(a1?1;r1,3);Pa;
Pb::=e(a1!1;r3;a1!1,6);Pb;
```

(그림 7) (그림 3)에 대한 실험용 실시간 프로세스

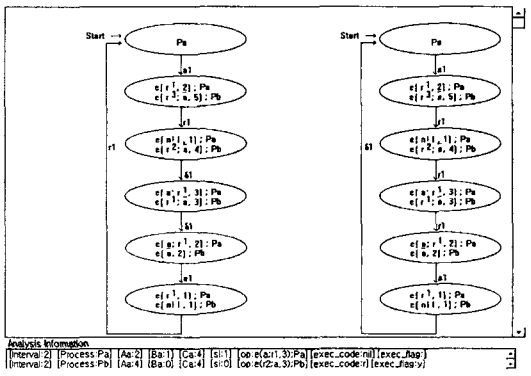
즉, 프로세스의 이름은 'P'로 시작하며 각 프로세스는 다음 문자의 입력으로 서로 다른 프로세스를 식별한다. 동기화 동작은 'a'로 시작하는 식별자(identifier)와 a 이후에 번호를 붙여 동기화 동작을 구분하며, send 동작은 '!'로 receive 동작은 '?'를 동기화 동작의 이름 뒤에 붙여 구분한다.

스케줄가능성 분석 결과는 (그림 8)에서 분석된 프로세스 상태공간과 같이 네번째의 상태공간에서 스케줄 불가능하다는 것을 판단할 수 있다.



(그림 8) 스케줄 불가능한 프로세스 상태공간

프로세스 상태 분석기는 실험 프로세스에 대해서 스케줄가능성 식(1)의 결과와 스케줄가능성 분석 결과 모두가 부정이므로, 본 논문에서 제시된 회복 [알고리즘 2]를 적용하여 구성된 회복 처리 함수를 수행하면, (그림 9)와 같은 스케줄 가능한 프로세스 상태공간으로 표시된다. 특히, 프로세스 상태 분석기는 프로세스의 처리를 위해 각 프로세스 당 한 개의 처리기를 할당하여 작업을 수행시킨다.



(그림 9) 회복 알고리즘을 적용한 유도된 스케줄가능한 프로세스 상태공간

따라서, Fredette의 방법에서는 9개의 상태공간 모두를 검사하여야 스케줄이 불가능함을 판단할 수 있었으나, 본 논문에서 3개의 상태공간을 검사함으로써 스케줄이 불가능함을 판단할 수 있었다.

(그림 10)은 식(1)이 긍정일 경우에 적용될 회복 알고리즘을 평가하기 위해 사용된 실시간 프로세스 명세이다.

Pa::=e($\overline{a}^1$,3);Pa;
Pb::=e(a^1;a^1; γ^4 ,6);Pb;

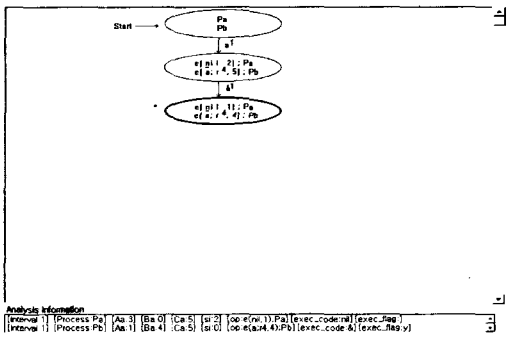
(a) 실시간 프로세스

Pa::=e(a1?1,3);Pa;
Pb::=e(a1!1;a1!1;r4,6);Pb;

(b) 실험용 언어로 작성된 실시간 프로세스

(그림 10) 실험용 실시간 프로세스

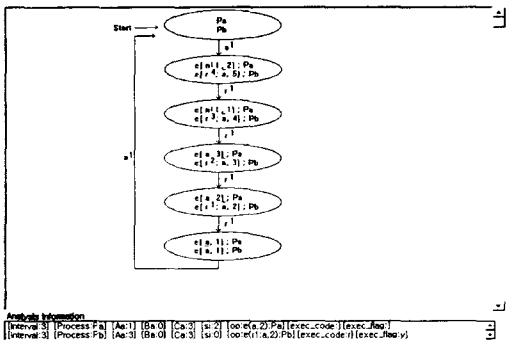
(그림 10)의 프로세스를 프로세스 상태 분석기를 통해 적용하면, (그림 11)과 같이 2번째 단계의 상태공간에서 스케줄이 불가능함을 알 수 있다.



(그림 11) 프로세스 상태 분석기의 메인 윈도우

스케줄가능성 분석에 대한 첫번째 과정 처리 결과(SPT 테이블)는 수행 후, 각 행에 대해 메시지 박스를 이용해서 화면에 출력되며, 두번째 과정에 대한 처리 결과(분석 테이블) 및 기타 정보는 상태 출력 윈도우 하단의 Analysis Information 컨트롤에 문자열로서 출력된다.

스케줄 불가능한 프로세스 상태공간에 대해 적용될 회복기법을 선택하기 위해서 식(1)을 통한 스케줄 가능 여부를 판단한다. 결과는 해당 처리 함수의 BOOL 형 지역 변수 sch_jug에 저장된다.



(그림 12) 회복 알고리즘을 적용한 스케줄가능한 프로세스의 상태공간

그리고 스케줄가능성 분석 결과는 전역 구조체 SPT2_GLV의 unschable에 저장된다. 그러므로 두 변수가 가진 값의 경우가 조합되어 분기되므로, 여기에서는 스케줄가능성 분석 결과가 저장된 unschable이 부정이며, 식(1)의 처리 결과는 긍정이다. 따라서, 제시된 회복기법의 적용을 통해 프로그램 코드의 순서를 변경하여 단일 처리기를 할당하는 회복기법을 선택할 수 있다. 처리된 결과는 (그림 12)와 같다.

5. 결론

본 논문에서는 스케줄가능성 분석 방법을 이용하여 스케줄이 불가능한 프로세스를 스케줄이 가능하도록 하는 스케줄 회복 알고리즘을 제안하고, Visual C++ 5.0으로 스케줄가능성 분석 결과를 시각적으로 나타내는 프로세스 상태 분석기를 구현하였다.

따라서, 이 프로세스 상태 분석기는 실시간 프로세스의 스케줄가능성 판단에 효과적으로 응용될 것이라고 생각되며, 향후의 연구 분야는 분산 시스템에 이 방법을 적용한다.

참고문헌

[1] G. Bruno, A. Castella, I. Pavesio and M. P. Pescarmona, "A New Petri Net Based Formalism for Specification, Design and Analysis of Real-Time Systems", Proceeding Real-Time Systems Symposium, pp. 294-301, 1993.

[2] C. L. Liu and J. Layland, "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment", JACM Vol20, No1, pp.46-61, Jan. 1973.

[3] A. N. Fredette and R. Cleaveland, "A Generalized Approach to Real-Time Schedulability Analysis", 10th IEEE Workshop on Real-Time Operating Systems and Software, 1993.

[4] A. N. Fredette and R. Cleaveland, "RTSL: A Language for Real-Time Schedulability Analysis,", Proceedings of the IEEE Real-Time Systems Symposium, pp. 274-283, 1993.

[5] R. Milner, Communication and Concurrency, Prentice-Hall, 1989.

[6] 최정률, 김춘배, 박홍복. "프로세스 대수를 이용한 실시간 프로세스의 스케줄가능성 분석 알고리즘", 정보처리학회 '98 추계 학술발표 논문집 제5권 2호, pp. 1239-1242, 1998. 10

[7] 박홍복, 유원희, "실시간 프로그램의 스케줄가능성 분석 방법", 한국정보처리학회 Vol.2, No.1, pp. 119-129, 1995