SPIN 과 ACSR-VP 를 이용한 Process Scheduling 모델링*

방 기석, 허 윤정, 최 진영, 유 혁
고려 대학교 컴퓨터 학과

Process scheduling modeling using SPIN and ACSR-VP

Ki-Seok Bang , Yoon-Jung Hur, Jin-Young Choi, Chuck-Yoo
Department of Computer Science and Engineering, Korea University

요 약

본 논문에서는 Holzmann 이 제시한 커널 스케쥴링 방식이 liveness 를 만족하지 못하며, 자원의 충돌이 일어남을 프로세스 알제브라 ACSR-VP 를 이용해서 확인하였고, SPIN 과 LTL formula 를 이용하여 검증하였다. 이것을 SPIN 과 ACSR-VP 를 이용하여 오류가 없는 새로운 모델을 제시하였다.

1. 서론

Holzmann 이 1997 년 IEEE Transactions on Software Engineering 에 게재한 'The Model Checker – SPIN"[10]이라는 논문에 보면 프로세스 스케쥴링 알고리즘의 검증 모델이 제시되어 있다. 그러나 이 검증 모델은 liveness 를 만족하지 못한다. 그러나 liveness 를 만족시키도록 설계한 모델에서도 클라이언트와 서버 사이에서 자원의 충돌이 일어나게 된다. 이 오류는 SPIN 을 이용하여 검증하였고 ACSR-VP[8]로 모델링하여 확인하였다. 이번 연구에서는 LTL[13]과 SPIN 을 이용하여 오류의 원인을 확인하고, 새로운 모델링을 제시한다 또한 ACSR-VP 로 검증을 하여 완벽한 프로세스 스케쥴링 모델링을 완성하였다.

2. Client-Server Program

2.1 Spin 의 예

Holzmann[10]의 커널 스케쥴링 모델의 다이어그램(diagram)과 Promela 소스는 논문[10]을 참고하기 바란다. 이 커널 스케쥴링은 서버가 자원을 제공하고 클라이언트는 사용 가능할 때마다 그 자원을 소모하는 단순한 스케쥴링이다. 사용할 수 있는 자원이 없으면 클라이언트는 sleep 을 하고 서버는 클라이언트의 상태를 체크해서 자원을 할당해 주게 된다

그러나 Holzmann 의 모델을 SPIN 으로 검증해 보면 클라이언트가 진행하지 못하고 서버만이 클라이언트의 상태를 검사하는 작업을 반복하게 된다. 즉 스케쥴링 모델이 liveness 특성을 만족하지 못하게 된다[10] 이 특성을 만족시키기 위해 Holzmann 이 해결안을 제안했으나 SPIN 을 이용해서 검증해 본 결과 그 역시 만족하지 못함을 확인하였다. 또한, SPIN 의 검증기를 이용하여 검증을 한 결과 두 개의 프로세스 사이에서 자원의 충돌이 생겨 올바른 동작을 하지 않음을 확인할 수 있었다 Holzmann 의 스케쥴링 모델에 LTL formula 를 적용해서 검증한 결과 이 모델에서 클라이언트와 서버가 동작하는 동안 임계구역을 보호하지 못하고 접근하고 있는 것을 확인하였다. 이 자원의 충돌은 Holzmann 의 스케쥴링 모델 자체에서 오류를 포함하고 있는 것을 보여준다.

2.2 ACSR-VP 의 예

Holzmann 의 커널 스케쥴링을 ACSR-VP 를 이용하여 모델링하고, 그 스케쥴링이 liveness 를 만족하는지를 확인하였다. 그 결과 Holzmann 의 수정모델도 liveness 를 만족하지 못함을 알 수 있었다. ACSR-VP 를 이용하여 명세한 모델은 참고문헌[14]를 참조하기 바란다

다음은 VERSA 를 이용해서 커널 스케쥴링을 검증했을 때의 출력 결과이다

```
:-) SYS == CH ?
ufi failed—following pair could not be matched.
false (by prioritized strong equivalence)
false (by prioritized weak equivalence)
.-)
```

그림 1 VERSA 를 이용한 커널 스케쥴링 검증 결과

* 본 연구는 1997 년도 특정기초연구지원사업(97-01-00-09-01-3)으로 이루어졌습니다.

위에서 보는 바와 같이 SYS 와 CH 에 대한 위크 바이시뮬레이션(weak bisimulation) 검증 결과가 false 를 나타냄을 알 수 있다.

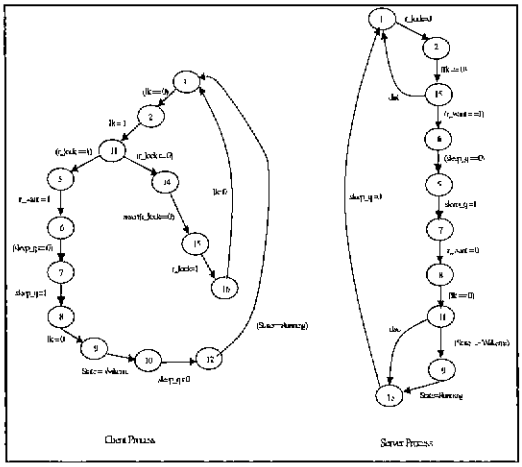

그림 2 완벽한 프로세스 스케줄링의 상태전이도

3. 오류의 수정

Liveness 를 만족하기 위해 논문의 바탕이 되고 있는 Ruane 의 UTS 커널논문[12]을 살펴보았다. Ruane 의 논문에 따르면 두 프로세스 간의 동기화를 위해 waitlock()를 이용하도록 제시되어 있다. 즉 클라이언트가 sleep()신호를 보낸 뒤에 자신이 갖고 있던 lock 을 풀고 sleep 을 하게 된다. 이 때 서버는 비로소 lock 을 획득하고 클라이언트의 상태를 검사하도록 하는 것이다. 그러나 Holzmann 의 스케줄링 모델에서는 서버가 lock 을 획득하지 않고 클라이언트의 상태를 판단하게 된다. 이렇게 되면 클라이언트가 sleep 상태로 바꾸기 전에 서버가 클라이언트의 상태를 판단하게 되고, 클라이언트의 상태를 잘못 판단하게 되는 것이다. 그 후에 클라이언트가 sleep 을 하게 되더라도 서버는 자원을 원하는 클라이언트가 없다고 판단을 해서 클라이언트를 깨우지 않게 되는 것이다. 이것이 Holzmann 의 모델이 liveness 를 만족시키지 않는 이유이다. 이 문제를 해결하기 위해 서버 프로세스의 r_want=0 다음에 (lk==0)을 추가해야 한다 이렇게 하면 클라이언트가 sleep() 신호를 낸 뒤에 lock 을 풀고, 그 후에 서버가 lock 을 획득해 클라이언트의 상태를 판단하게 된다 즉 두 프로세스 사이에 동기화가 이뤄지게 되는 것이다. 그러나 liveness 를 해결한 뒤에도 Holzmann 의 스케줄링 모델에서는 자원의 충돌이 발생하였다. Ruane 의 커널 모델에서는 sleep queue 를 접근할 때 서버와 클라이언트가 동시에 접근하는 것을 허용하지 않는다 이를 위해서 spinlock()을 이용하고 있다. sleep queue 를 접근하기 위해서는 클라이언트와 서버가 모두 spinlock()을 획득해야만 한다 클라이언트나 서버가 spinlock()을 획득하지 않거나 동시에 sleep queue 를 접근하면 sleep queue 의 정보가 일관되지 않기 때문에 자원의 충돌이 발생하는 것이다. Holzmann 의 설계에는 이것이 결여되어 있다. 이를 확인하기 위해 [](! sw == sr)이라는 LTL formula 를 이용해서 검증을 해 보았다 결과 Holzmann 의 설계에서 sleep queue 가 보호되지 못함을 확인하였고, 클라이언트에 sleep queue 접근시 spinlock()를 획득하는 부분을 추가함으로써 완벽한 모델을 구현하였다. 그림 3 은 LTL formula 를 SPIN 에 적용시켜서 검증을 한 결과이다 스케줄링 모델이 LTL formula 를 만족하여 자원이 충돌하지 않음을 나타내고 있다

```
warning  for p.o. reduction to be valid the never claim must be
stutter-closed
(never claims generated from LTL formulae are stutter-closed)
(Spin Version 3.2.0 -- 8 April 1998)
            + Partial Order Reduction

Full statespace search for:
    never-claim                 +
    assertion violations + (if within scope of claim)
    acceptance    cycles        + (fairness enabled)
    invalid endstates   - (disabled by never-claim)

State-vector 24 byte, depth reached 102, errors: 0
        91 states, stored
        52 states, matched
       143 transitions (= stored+matched)
        44 atomic steps
hash conflicts: 0 (resolved)
(max size 2^19 states)

2.542       memory usage (Mbyte)

unreached in proctype client
            line 39, state 27, "-end-"
            (1 of 27 states)
unreached in proctype server
            line 56, state 14, "Sr = 1"
            line 62, state 25, "-end-"
            (2 of 25 states)

real         0.0
user         0.0
sys          0.0
```

그림 3 LTL 을 이용한 프로세스 스케줄링 검증결과

SPIN 으로 검증한 모델을 ACSR-VP 로 다시 명세하고 VERSA 로 검증하였을 때 liveness 특성을 만족하고 자원충돌이 발생하지 않음을 확인할 수 있었다.

다음은 ACSR-VP 를 이용하여 수정된 커널 스케줄링 모델을 명세한 프로그램과 VERSA 를 이용하여 검증한 결과 화면이다. 결과에서 볼 수 있듯이 새로운 모델링에서는 오류가 발생되지 않고 올바른 동작을 하고 있음을 확인할 수 있다.

```
SYS = (C11 || S1. .)\{wr,rr,wr2,rr2,wr3,rr3,wr4,rr4 wr5,rr5};

CH = (check?).CH;
C11 = (check?).C11a;
C11a = (rr?x) ((x=1)→(wr2!1).C6
          + (x=0)→C14);
C6 = (rr5?x).((x=0)→(wr5!1).(wr4!0).(wr3!0).(wr5!0).C13
          +(x=1)→C6);
C13 = (rr3?x).((x=1)→C3
          +(x=0)→C13);
C14 = (rr?x).((x=0)→C15
          + (x=1)→C14);
C15 = (wr!1).(wr4!0).C3;
C3 = (rr4?x) ((x=0)→C2
          + (x=1)→C3);
C2 = (wr4!1).C11;

S1 = (wr!0) S2;
S2 = (rr4?x) ((x=0) →S15 + (x=1) →S2);
S15 = (rr2?x).((x=1) →S6 + (x=0) →S1);
S6 = (rr5?x).((x=0) →S5 + (x=1) →S6);
S5 = (wr5!1) S7;
S7= (wr2!0).S8;
S8 = (rr4?x) ((x=0) →S11 + (x=1) →S8);
S11 = (rr3?x).((x=0) →S9 + (x=1) →S13),
S9 = (wr3!1).S13;
S13 = (wr5!0) S1;
```

그림 4. 완벽한 모델의 ACSR-VP 정형명세

```
:-) SYS==CH?
UFI not applied--operands are not guarded.
false (by prioritized strong equivalence)
true (by prioritized weak equivalence)
```

그림 5. 완벽한 모델의 VERSA 검증 결과

4. 결론

이번 연구에서는 Holzmann의 커널 스케줄링 모델을 검증해 보았다. 그 결과 Holzmann의 모델이 liveness를 만족시키지 못함을 확인하였고, LTL formula를 SPIN에 적용함으로 검증하였다 이것을 ACSR-VP로 다시 명세하여 liveness의 문제를 갖고 있음을 확인하였다 이 문제를 해결하기 위해 Ruane의 스케줄링 모델을 참고하여 서버와 클라이언트 프로세스의 동작이 동기화 되지 못함을 발견하고 새롭게 모델링하여 liveness를 만족하도록 수정하였다. 그러나 이렇게 수정된 모델에서는 클라이언트와 서버 사이에서 자원의 충돌이 발생하는 것을 확인하였다 Ruane의 커널 스케줄링을 참고하여 Holzmann의 모델에서는 두 프로세스가 sleep queue를 보호하지 않고 동시에 접근하고 있음을 확인하였다. 이 문제를 해결하기 위해 클라이언트에 spinlock()을 획득하도록 하여 새롭게 모델링을 하였고, 이 모델에도 LTL formula를 적용하여 SPIN으로 자원의 충돌이 발생하지 않음을 검증하였다. 또 ACSR-VP로 명세하여 VERSA를 이용해서 검증한 결과 새로운 모델은 liveness도 만족하고, 자원의 충돌도 일어나지 않음을 확인하였다.

연구 결과 LTL formula를 SPIN에 적용하여 시스템을 설계하고 ACSR-VP를 이용하여 검증을 할 수 있음을 확인하였다.

5. 참고 문헌

[1] Andre M. van Tilborg, "Foundations of Real-Time Computing-Formal Specifications and Methods", Kluwer Academic Publishers, 1991

[2] Andrew Harry. " Formal Methods-FACT FILE, VDM and Z". Wiley. 1996

[3] Patrice Brémond-Grégoire, J I Choi, Insup-Lee, "A Complete Axiomatization of Finite-state ACSR Processes", Information and Computation, No 138, 1997.

[4] C Heitmeyer and D. Mandrioli, "A Process Algebraic Method for the Specification and Analysis of Real Time Systems", Formal Methods for Real-Time Computing, Wiley, 1996

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

[6] Duncan Clarke. "VERSA:Verification, Execution and Rewrite System for ACSR", University of Pensylvania, 1994

[7] Insup Lee, Duncan Clark, and Hong-Liang Xie, "The algebra of communicating shared resources and its toolkit.", In Sang H Song, editor, Advances in Real-Time Systems, chapter 12, pp 275-298, Prentice Hall

[8] I. Lee, H. Ben-Abdallah, and J.-Y. Choi, "A Process Algebraic Method for Real-Time Systems", Formal Methods for Real-Time Computing C Heitmeyer and D. Mandrioli (eds), John Wiley & Sons Ltd, 1996

[9] 최 진영, 임 성묵, "ACSR-VP를 이용한 실시간 시스템의 스케줄링 기법에 대한 정형 명세와 검증", 1998, 정보과학회 논문지(A) 제 25권 제 6호, pp 568 - 581

[10] Gerald J. Holzmann, "The Model Checker SPIN", IEEE Transactions on Software Engineering, VOL. 23, NO 5, MAY 1997, pp279 – 295

[11] Gerald J Holzmann, "Design and Validation of Computer Protocols", Prentice Hall, 1991

[12] Lawrence M. Ruane, "Process Synchronization in the UTS Kernel", Computing Systems, VOL. 3, NO 3, 1990, pp 387 – 421

[13] Zohar Manna, Amir Pnueli, "The Temporal Logic of Reactive and Concurrent Systems – Specification", Springer-Verlag, 1996

[14] 방 기식, 최 진영, "Gerald J. Holzmann의 "The Model Cheker SPIN"에 대하여", 1998, 춘계정보과학회 춘계학술발표 논문집(A) 제 25권 1호, pp 596 – 598