CSP/FDR을 이용한 리더 선출 알고리즘의 검증

전철욱[0], 김일곤, 안영아, 최진영
고려대학교 컴퓨터학과
{cwjeon[0], igkim, ellaahn, choi}@formal.korea.ac.kr

Verification of Leader Election Algorithm with CSP/FDR

Chul-Wuk Jeon, Il-Gon Kim, Young-a Ahn, Jin-Young Choi
Dept of Computer Science & Engineering, Korea University

요약

시스템이 대형화가 되어가고 네트워크 환경이 발전함에 따라 분산 환경이 점점 더 증대되어 가고 있다. 이러한 분산 환경에서 사용되는 리더 선출 알고리즘(Leader Election Algorithm)은 다양하게 제시 되었고 본 논문에서는 Garcia-Molina가 제시한 Bully 알고리즘을 프로세스 알제브라 언어인 CSP로 명세하고 FDR 모델체킹 도구를 이용해 해당 요구사항을 만족하는지 검증하였다.

1. 서론

성능이 우수한 마이크로프로세서의 개발과 통신망이 발전함에 따라 컴퓨팅 환경이 경제성, 속도, 신뢰성 등 여러 가지 장점을 가진 분산 환경으로 계속해서 변화되고 있다. 이러한 분산 환경에서 사용되는 리더 선출 알고리즘(Leader Election Algorithm)은 다양하게 제시되어 왔고 그 중 Bully 알고리즘은 Garcia-Moina가 1982년 제시했다. 이 알고리즘은 각각의 프로세스가 자신만의 식별자를 가지고 있고 리더 선출을 시작하게 되면 모든 프로세스가 동의한 새로운 리더 선출을 보장한다. 분산 알고리즘을 검증하기 위해 다양한 정형 기법과 도구들이 적용되어왔다[1]. 본 논문에서는 프로세스 알제브라 언어인 CSP로 Bully 알고리즘을 명세하고 모델 체커인 FDR을 이용해서 "반드시 Coordinator가 존재한다" 요구 사항을 검증하였다. 2장에서는 Bully 알고리즘과 CSP를 소개하고 3장에서는 Bully 알고리즘을 CSP로 명세하고 FDR로 검증한 내용을 설명하고, 마지막 4장에서는 결론 및 향후 연구에 대해 논하도록 하겠다.

2. Bully 알고리즘 및 CSP 소개

2.1 Bully 알고리즘

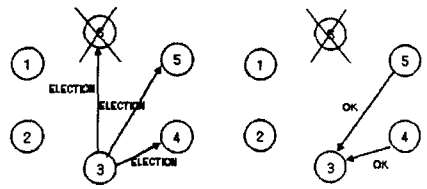
Bully 알고리즘은 다음과 같이 간략히 설명할 수 있다[2].

Coordinator가 제대로 실행하지 않고 있다는 것을 발견한 프로세스가 election을 초기화 한다.

- 프로세스 P는 다음과 같은 절차를 갖는다.
  1) P는 ELECTION 메시지를 자신보다 높은 식별자를 가진 모든 프로세스에서 보낸다.
  2) 만약 응답이 없다면, P 프로세스는 Coordinator가 된다.
  3) 만약 자신보다 높은 식별자를 가진 프로세스가 OK 응답을 할 경우 P 프로세스의 작업은 끝이 난다.

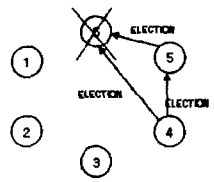
- 만약 프로세스 Q가 자신보다 낮은 식별자를 가진 프로세스에게 ELECTION을 받을 경우
  1) Q는 sender에게 OK 응답을 보낸다.
  2) Q가 election 을 할 수 있게 된다.

- 새로 정해진 Coordinator는 모든 프로세스에게 COORDINATOR 메시지를 보낸다.

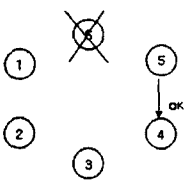


[그림 1-a]

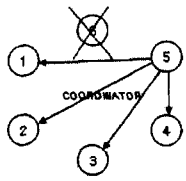
[그림 1-b]



[그림 1-c]



[그림 1-d]



[그림 1-e]

다음은 간단한 예제로서 식별자 3을 가진 프로세스는 식별자 6을 가진 프로세스가 제대로 실행 하지 못하는 것을 알고 election을 초기화한다. 식별자 3을 가진 프로세스 보다 큰 식별자를 가진 프로세스에게 ELECTION 메시지를

보내고[그림 1-a], 이를 받은 프로세스는 식별자 3을 가진 프로세스에게 OK라는 메시지로 응답한다. OK라는 메시지를 받은 식별자 3을 가진 프로세스는 election을 끝낸다[그림 1-b]. election을 할 수 있는 식별자 5, 6을 가진 프로세스는 자신보다 높은 식별자를 가진 프로세스에게 ELECTION 메시지를 다시 보내고[그림 1-c], 식별자 4를 가진 프로세스는 보다 큰 식별자 5를 가진 프로세스에게 OK 메시지로 응답한다[그림 1-d]. 식별자 5를 가진 프로세스는 자신이 현재 실행하고 있는 프로세스 중 가장 큰 식별자를 가진 프로세스라는 것을 알고 모든 프로세스에게 자신이 Coordinator가 되었다고 COORDINATOR 메시지를 보낸다[그림 1-e].

2.2 CSP 소개

CSP(Communicating Sequential Processes)는 컴포넌트가 상호작용하는 시스템을 명세하기 위해 개발된 정형명세 언어이다. 프로세스(컴포넌트)는 자신의 환경과 상호작용을 하는 특정한 인터페이스를 가진 독립적 개체이고 인터페이스는 이벤트의 집합이다[3][4].
만약 PRINTER 프로세스의 행위를 간략하게 CSP로 프로세스와 이벤트로 나타나게 되면 다음과 같다.

PRINTER = accept -> print -> STOP

즉, PRINTER는 accept라는 이벤트를 실행한 다음 print를 실행하고 멈추게 된다.
프로세스는 choice(|)도 가능하다.

PRINTER = startup -> (accept -> print -> STOP
| shutdown -> STOP)

PRINTER는 startup을 실행하고 accept, print, STOP을 실행하거나 또는 shutdown, STOP을 수행한다.

COMPUTER = PRINTER_ON [] PRINTER_OFF

연산자 []는 프로세스를 선택할 수 있다. 즉, COMPUTER 프로세스는 PRINTER_ON이라는 프로세스를 실행하거나 PRINTER_OFF라는 프로세스를 실행한다.

CSP는 Alphabetized Parallel, Interleaving, Common Event Parallel로 병렬성(parallelism)을 표현할 수 있지만 본 논문에서는 Alphabetized Parallel만 간략하게 설명하겠다.
Parallel 연산자는 동기화(synchronization)를 기반으로 서로 다른 프로세스 중에서 각 프로세스 중에서 같은 이벤트가 있다면 같은 순서로 함께 실행된다. 만약, 주차 영수증 발급기를 다음 두 가지 프로세스로 나누고 이것을 Alphabetized Parallel로 이용하여 명세할 수 있다.

TICKET = cash -> ticket -> TICKET
CHANGE = cash -> change -> CHANGE

MACHINE = TICKET [{cash, ticket} || {cash, change}] CHANGE

TICKET은 현금을 받고 표를 주고 다시 TICKET으로 돌아오며 CHANGE 또한 현금을 받고 잔돈을 주고 다시 CHANGE로 돌아온다. MACHINE에서는 cash라는 공통의 이벤트가 같은 순서에서 함께 실행되며 각 프로세스가 명세된 이벤트가 실행된다.

3. Bully 알고리즘에 대한 CSP 명세와 검증

[그림 1-a]와 같은 형태로 총 6개의 프로세스가 있다고 명세하였다. 만약 프로세스 수를 증가하려면 6이 아닌 더 높은 수로 명세하면 된다.

Net = {1..6}

프로세스는 크게 3개로 나누어 아직 Coordinator가 되지 못하고 election을 실행하는 프로세스를 Pre_Leader라 하였고, Coordinator가 되는 프로세스를 Leader라 하였다. 마지막으로 election을 하지 않는 프로세스를 Node라 하였다. 이벤트는 다음과 같다.

channel select : Net
channel election : Net.Net
channel coordinator : Net.Net
channel ok
channel fail

select는 알고리즘에서 ELECTION 메시지를 보낼 프로세스를 선택하는 이벤트이다. election은 자신보다 높은 식별자를 가진 프로세스에게 보내는 이벤트로서 Pre_Leader보다 높은 식별자를 가진 Node 프로세스와 동기화가 발생한다. coordinator는 Coordinator가 된 Leader 프로세스가 각 프로세스들에게 알리는 이벤트이다. ok는 알고리즘에서 사용되는 것과 마찬가지로 election에 응답하는 이벤트이다. 마지막으로 fail은 실제 알고리즘에는 없지만 응답 시간이 지나서 ok라는 메시지를 못 받을 경우나, 프로세스가 실행하고 있지 않은 경우 ok라는 메시지를 보내지 못하기 때문에 fail이라는 이벤트가 생긴다고 가정해서 명세했다.

others(id, R) = diff(R, {id})
subGroup(G1, G2) = diff(G2, G1)
another(id) = {1..id}

간단하게 집합을 나타낼 수 있는 함수를 생성하였다. others는 R이라는 집합에서 id라는 원소를 ■는 함수이고 subGroup은 G2라는 집합에서 G1이라는 집합을 ■는 함수이다. 마지막으로 another는 1부터 id까지 집합을 포함하는 함수이다. Pre_Leader는 다음과 같이 명세했다.

Pre_Leader(id, Group) = empty(subGroup(another(id), Group)) &Leader(id)
[] not empty(subGroup(another(id), Group)) & select?j:subGroup(another(id), Group) -> Pre_Leader2(id, j, others(id, Group))

Pre_Leader2(id, j, Group) = (if j > id then election.id.j -> (ok -> Node(id, Group) [] fail -> Pre_Leader(id, others(j, Group)))

else Pre_Leader(id, Group))

empty는 예약어로서 집합이 공집합인가를 체크해 준다. Pre_Leader가 자신보다 높은 식별자를 가진 프로세스와 모두 election 이벤트를 실행했는가를 알려준다. 다시 말해, Pre_Leader는 자신보다 바로 높은 식별자를 가진 프로세스와 election을 함께 실행한다고 가정하고 계속해서 식별자가 낮은 것부터 높은 순으로 실행한다고 하면 Net이라는 집합이 공집합이 될 수 있다. 공집합이 될 동안 자신 보다 높은 식별자를 가진 프로세스를 만나지 못하면 바로 지금 실행하고 있는 프로세스가 Coordinator가 되면 Leader라는 프로세스가 된다. election 이벤트를 실행하고 다른 프로세스와 ok 이벤트를 실행하게 되면 지금 실행하고 있는 프로세스는 Node라는 프로세스가 된다. 만약 fail 이벤트가 실행된다면 그대로 Pre_Leader가 되고 다시 처음부터 Pre_Leader 프로세스가 실행된다.

Coordinator가 된 프로세스 Leader는 다음과 같다.

Leader(id) = coordinator.id?m -> STOP

다른 모든 Node에게 자신의 식별자(id)로 coordinator라는 이벤트가 실행된다.

Node 프로세스는 다음과 같다.

Node(id, Group) = election?k?l ->
(if l==id then ok -> Pre_Leader(id, Group) [] fail -> Node(id, Group) else Node(id, Group))
[] ok -> Node(id, Group)
[] fail -> Node(id, Group)

Node는 Pre_Leader의 election 이벤트와 동기화가 일어난다. 만약 ok가 실행되면 자신이 Pre_Leader 프로세스가 되며 fail인 경우 Node로 남아 있게 된다.

Leader_election(P, pre_leader_id) =
[|Alpha|] i:P @ (if i == pre_leader_id then Pre_Leader(i, P) else Node(i, P))

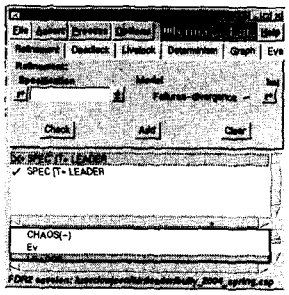
LEADER = Leader_election(Net, 3)

Leader_election은 여러 개의 프로세스가 Alphabetized Parallel을 할 수 있도록 만든 프로세스이며 LEADER 라는 프로세스는 6개의 프로세스를 가진 Net과 식별자 3을 가진 프로세스가 Pre_Leader인 것을 나타낸다.

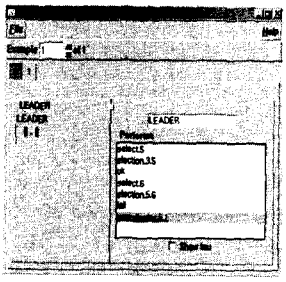
Bully 알고리즘의 속성중 하나는 "반드시 Coordinator가 존재해야 한다."이다. 이것을 SPEC이라 프로세스로 나타내면 다음과 같다.

SPEC = election?m?l -> SPEC
[] ok -> SPEC
[] coordinator?p?q -> STOP
[] fail -> SPEC
[] select?n -> SPEC

이러한 명세를 CSP를 입력어로 받는 FDR[5] 모델 체커에 입력하여 실행하게 되면 위에서 언급한 속성을 만족한다고 나타난다. 만약 "식별자 6을 가진 프로세스가 반드시 coordinator가 되어야 한다"라는 속성을 넣어 실행 시키면 다음과 같은 반례(counterexample)을 보여주게 된다. 이것은 "식별자 6을 가진 프로세스가 실행하지 못할 경우 식별자 5를 가진 프로세스가 coordinator가 된다."라는 것을 나타낸다.



[그림 2. 검증 결과]



[그림3. 반례]

4. 결론

지금까지 CSP라는 프로세스 알제브라 언어로 Bully 알고리즘을 명세하였고 FDR이라는 모델 체커를 이용해 "반드시 Coordinator가 존재해야 한다." Property를 검증해 보았다. 이러한 리더 선출 알고리즘은 wireless 환경에서 많이 사용되면 향후 연구로서는 wireless 환경에서 사용되는 리더 선출 알고리즘을 연구하고자 한다.

5. 참고 문헌

[1] 박찬국, 최진영, "Process Algebra 를 이용한 Chang 의 Leader Election Algorithm", 한국정보과학회'98 봄 학술발표논문집(A), 1998.
[2] Andrew S. Tanenbaum, "Distributed Operating System", PRENTICE HALL, 1995.
[3] Steve Schneider, "Concurrent and Real-time System", JOHN WILEY & SONS, LTD, 2000.
[4] A. W. Roscoe, "The Theory and Practice of Concurrency", PRENTICE HALL, 1998.
[5] Formal Systems(Europe) Ltd. Failure Divergence Refinement-FDR2 User Manual, Aug. 1999.