고급소프트웨어 공학
NuSMV - Dining Philosophers

201011334 박진성
201471200 박성훈

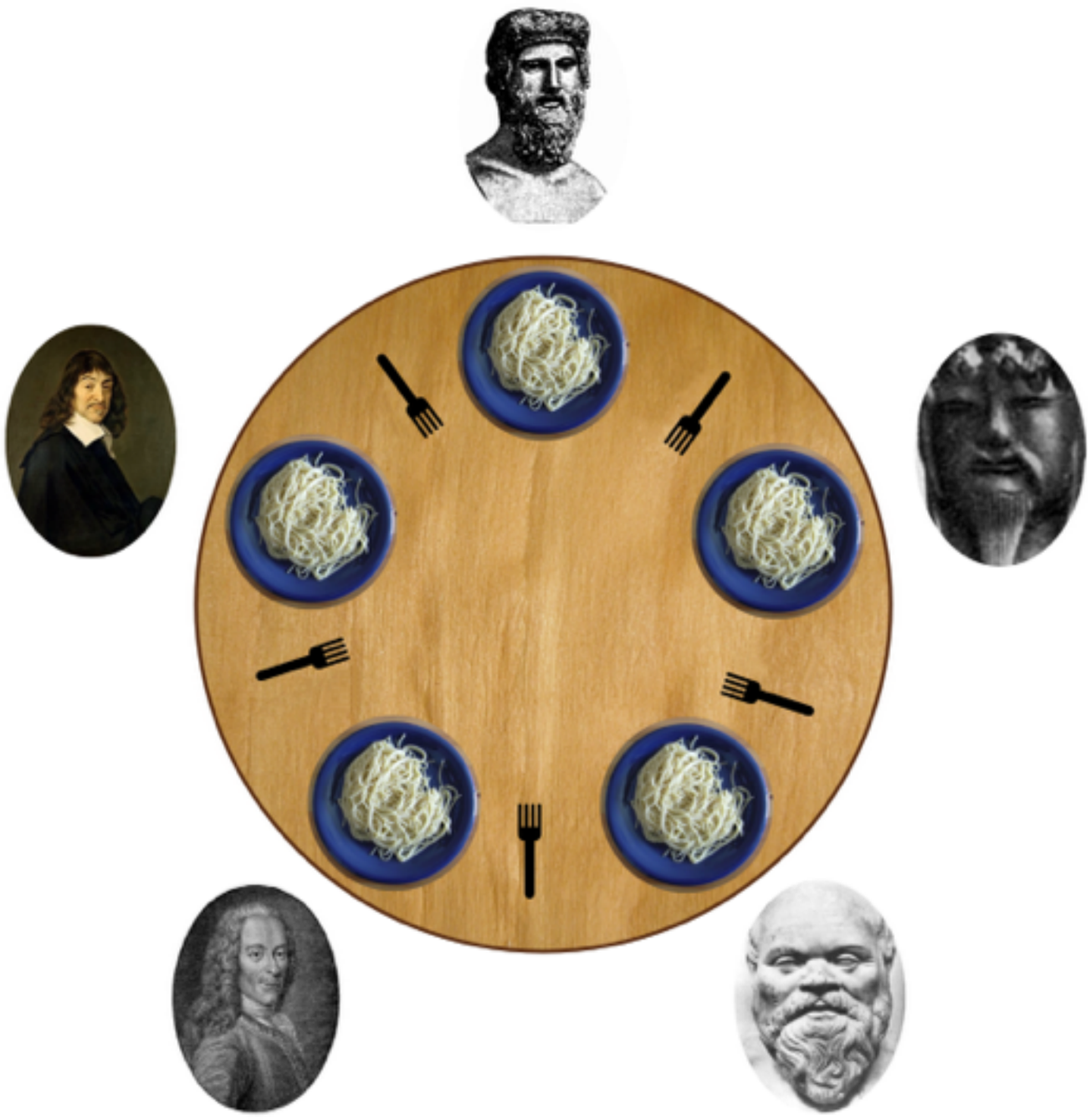
Index

- Dining Philosophers 문제
- Dining Philosophers 구현
- Dining Philosophers 검증

Dining Philosophers 문제

- 식사하는 철학자들 문제는 동시성과 교착상태를 설명하는 예시

- 여러 프로세스의 교착 상태 원인을 직관적으로 알기 쉬움



Dining Philosophers 구현

- 주요 로직

```
MODULE philosopher(lfork, rfork)
  VAR
    phil: {ready, rwait, lwait, eat};
  ASSIGN
    init(phil) := ready;
    next(lfork) := case
      phil != eat & lfork = usable : unusable;
      phil = eat & lfork = unusable : usable;
      TRUE: lfork;
    esac;
    next(rfork) := case
      phil != eat & rfork = usable : unusable;
      phil = eat & rfork = unusable : usable;
      TRUE : rfork;
    esac;
    next(phil) := case
      (phil = ready & lfork = usable & rfork = usable) | (phil = rwait & rfork = usable & lfork =
unusable) | (phil = lwait & lfork = usable & rfork = unusable) : eat;
      phil = ready & lfork = unusable & rfork = usable : lwait;
      phil = ready & lfork = usable & rfork = unusable : rwait;
      phil = eat: ready;
      TRUE : phil;
    esac;
  FAIRNESS
    running
```

왼쪽 포크를 집기

오른쪽 포크를 집기

철학자의 상태 업데이트

Dining Philosophers 구현

- 5인

```
MODULE main()
  VAR
    fork0: {usable, unusable};
    fork1: {usable, unusable};
    fork2: {usable, unusable};
    fork3: {usable, unusable};
    fork4: {usable, unusable};
    phil0: process philosopher(fork0, fork1);
    phil1: process philosopher(fork1, fork2);
    phil2: process philosopher(fork2, fork3);
    phil3: process philosopher(fork3, fork4);
    phil4: process philosopher(fork4, fork0);
  ASSIGN
    init(fork0) := usable;
    init(fork1) := usable;
    init(fork2) := usable;
    init(fork3) := usable;
    init(fork4) := usable;

  SPEC AG ((phil0.phil != lwait & phil0.phil != rwait) | (phil1.phil != lwait & phil1.phil !=
rwait) | (phil2.phil != lwait & phil2.phil != rwait) | (phil3.phil != lwait & phil3.phil !=
rwait) | (phil4.phil != lwait & phil4.phil != rwait));
  SPEC EF (phil0.phil = lwait & phil1.phil = lwait & phil2.phil = lwait & phil3.phil = lwait &
phil4.phil = lwait);
  SPEC EF (phil0.phil = rwait & phil1.phil = rwait & phil2.phil = rwait & phil3.phil = rwait &
phil4.phil = rwait);
  SPEC AG (phil0.phil != eat -> AF (phil0.phil = eat));
```

5개 포크 생성

5명 철학자에 대해서 포크를 할당

5개 포크 초기화

데드락이 걸리는지 검증

한명은 절대 못먹는 검증

Dining Philosophers 구현

- 검증 실행

```
-> Input: 1.7 <-
      _process_selector_ = phil0
      running = FALSE
      phil4.running = FALSE
      phil3.running = FALSE
      phil2.running = FALSE
      phil1.running = FALSE
      phil0.running = TRUE
-> State: 1.7 <-
      fork0 = unusable
      fork1 = unusable
      fork2 = unusable
      fork3 = unusable
      fork4 = usable
      phil0.phil = rwait
      phil1.phil = rwait
      phil2.phil = rwait
      phil3.phil = rwait
      phil4.phil = ready
-> Input: 1.8 <-
      _process_selector_ = phil4
      running = FALSE
      phil4.running = TRUE
      phil3.running = FALSE
      phil2.running = FALSE
      phil1.running = FALSE
      phil0.running = FALSE
-> State: 1.8 <-
      fork0 = unusable
      fork1 = unusable
      fork2 = unusable
      fork3 = unusable
      fork4 = unusable
      phil0.phil = rwait
      phil1.phil = rwait
      phil2.phil = rwait
      phil3.phil = rwait
      phil4.phil = rwait
```

데드락이 걸린다!!!

Dining Philosophers 구현

- 검증 실행

```
-- specification EF ((((phil0.phil = lwait & phil1.phil = lwait) & phil2.phil = lwait) & phil3.phil = lwait) & phi
l4.phil = lwait)  is true
-- specification EF ((((phil0.phil = rwait & phil1.phil = rwait) & phil2.phil = rwait) & phil3.phil = rwait) & phi
l4.phil = rwait)  is true
-- specification AG (phil0.phil != eat -> AF phil0.phil = eat)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-- Loop starts here
-> State: 2.1 <-
  fork0 = usable
  fork1 = usable
  fork2 = usable
  fork3 = usable
  fork4 = usable
  phil0.phil = ready
  phil1.phil = ready
  phil2.phil = ready
  phil3.phil = ready
  phil4.phil = ready
-> Input: 2.2 <-
  _process_selector_ = phil4
  running = FALSE
  phil4.running = TRUE
  phil3.running = FALSE
  phil2.running = FALSE
  phil1.running = FALSE
  phil0.running = FALSE
-> State: 2.2 <-
```

철학자가 스파게티를 못 먹는다!!

Dining Philosophers 구현

- 검증 실행
  - 실행 시간: **0.3초**
  - 아직까지는 잘 돌아간다…

Dining Philosophers 구현

- 10인

```
MODULE main()
  VAR
    fork0: {usable, unusable};
    fork1: {usable, unusable};
    fork2: {usable, unusable};
    fork3: {usable, unusable};
    fork4: {usable, unusable};
    fork5: {usable, unusable};
    fork6: {usable, unusable};
    fork7: {usable, unusable};
    fork8: {usable, unusable};
    fork9: {usable, unusable};
  phil0: process philosopher(fork0, fork1);
  phil1: process philosopher(fork1, fork2);
  phil2: process philosopher(fork2, fork3);
  phil3: process philosopher(fork3, fork4);
  phil4: process philosopher(fork4, fork5);
  phil5: process philosopher(fork5, fork6);
  phil6: process philosopher(fork6, fork7);
  phil7: process philosopher(fork7, fork8);
  phil8: process philosopher(fork8, fork9);
  phil9: process philosopher(fork9, fork0);
ASSIGN
  init(fork0) := usable;
  init(fork1) := usable;
  init(fork2) := usable;
  init(fork3) := usable;
  init(fork4) := usable;
  init(fork5) := usable;
  init(fork6) := usable;
  init(fork7) := usable;
  init(fork8) := usable;
  init(fork9) := usable;
```

10개 포크 생성

10명 철학자에 대해서 포크를 할당

10개 포크 초기화

Dining Philosophers 구현

- 10인 검증 코드

```
  SPEC AG ((phil0.phil != lwait & phil0.phil != rwait) | (phil1.phil != lwait &
phil1.phil != rwait) | (phil2.phil != lwait & phil2.phil != rwait) | (phil3.phil != lwait &
phil3.phil != rwait) | (phil4.phil != lwait & phil4.phil != rwait) | (phil5.phil != lwait &
phil5.phil != rwait) | (phil6.phil != lwait & phil6.phil != rwait) | (phil7.phil != lwait &
phil7.phil != rwait) | (phil8.phil != lwait & phil8.phil != rwait) | (phil9.phil != lwait &
phil9.phil != rwait));
  SPEC EF (phil0.phil = lwait & phil1.phil = lwait & phil2.phil = lwait & phil3.phil =
lwait & phil4.phil = lwait & phil5.phil = lwait & phil6.phil = lwait & phil7.phil = lwait &
phil8.phil = lwait & phil9.phil = lwait);
  SPEC EF (phil0.phil = rwait & phil1.phil = rwait & phil2.phil = rwait & phil3.phil =
rwait & phil4.phil = rwait & phil5.phil = rwait & phil6.phil = rwait & phil7.phil = rwait &
phil8.phil = rwait & phil9.phil = rwait);
  SPEC AG (phil0.phil != eat -> AF (phil0.phil = eat));
```

데드락이 걸리는지 검증

한명은 절대 못먹는 검증

Dining Philosophers 구현

- 검증 실행
  - 실행 시간: **9.463초**
  - 힘들어하기 시작한다

Dining Philosophers 구현

- 15인

```
MODULE main()
  VAR
    fork0: {usable, unusable};
    fork1: {usable, unusable};
    fork2: {usable, unusable};
    fork3: {usable, unusable};
    fork4: {usable, unusable};
    fork5: {usable, unusable};
    fork6: {usable, unusable};
    fork7: {usable, unusable};
    fork8: {usable, unusable};
    fork9: {usable, unusable};
    fork10: {usable, unusable};
    fork11: {usable, unusable};
    fork12: {usable, unusable};
    fork13: {usable, unusable};
    fork14: {usable, unusable};
    phil0: process philosopher(fork0, fork1);
    phil1: process philosopher(fork1, fork2);
    phil2: process philosopher(fork2, fork3);
    phil3: process philosopher(fork3, fork4);
    phil4: process philosopher(fork4, fork5);
    phil5: process philosopher(fork5, fork6);
    phil6: process philosopher(fork6, fork7);
    phil7: process philosopher(fork7, fork8);
    phil8: process philosopher(fork8, fork9);
    phil9: process philosopher(fork9, fork10);
    phil10: process philosopher(fork10, fork11);
    phil11: process philosopher(fork11, fork12);
    phil12: process philosopher(fork12, fork13);
    phil13: process philosopher(fork13, fork14);
    phil14: process philosopher(fork14, fork0);
```

15개 포크 생성

15명 철학자에 대해서 포크를 할당

Dining Philosophers 구현

- 15인 검증 코드

```
  SPEC AG ((phil0.phil != lwait & phil0.phil != rwait) | (phil1.phil != lwait &
phil1.phil != rwait) | (phil2.phil != lwait & phil2.phil != rwait) | (phil3.phil != lwait &
phil3.phil != rwait) | (phil4.phil != lwait & phil4.phil != rwait) | (phil5.phil != lwait &
phil5.phil != rwait) | (phil6.phil != lwait & phil6.phil != rwait) | (phil7.phil != lwait &
phil7.phil != rwait) | (phil8.phil != lwait & phil8.phil != rwait) | (phil9.phil != lwait &
phil9.phil != rwait) | (phil10.phil != lwait & phil10.phil != rwait) | (phil11.phil !=
lwait & phil11.phil != rwait) | (phil12.phil != lwait & phil12.phil != rwait) |
(phil13.phil != lwait & phil13.phil != rwait) | (phil14.phil != lwait & phil14.phil !=
rwait));
  SPEC EF (phil0.phil = lwait & phil1.phil = lwait & phil2.phil = lwait & phil3.phil =
lwait & phil4.phil = lwait & phil5.phil = lwait & phil6.phil = lwait & phil7.phil = lwait &
phil8.phil = lwait & phil9.phil = lwait & phil10.phil = lwait & phil11.phil = lwait &
phil12.phil = lwait & phil13.phil = lwait & phil4.phil = lwait);
  SPEC EF (phil0.phil = rwait & phil1.phil = rwait & phil2.phil = rwait & phil3.phil =
rwait & phil4.phil = rwait & phil5.phil = rwait & phil6.phil = rwait & phil7.phil = rwait &
phil8.phil = rwait & phil9.phil = rwait & phil10.phil = rwait & phil11.phil = rwait &
phil12.phil = rwait & phil13.phil = rwait & phil14.phil = rwait);
  SPEC AG (phil0.phil != eat -> AF (phil0.phil = eat));
```

데드락이 걸리는지 검증

한명은 절대 못먹는 검증

Dining Philosophers 구현

- 검증 실행
  - States: **약 700만개**
  - 실행이 불가능하다!!

```
  forward step done, size = 261055
  new frontier computed, size = 618
  iteration 23: BDD size = 261055, frontier size = 618, states = 7.75348e+06
          Size of intermediate product =       5075 (BDD nodes).
          Size of intermediate product =       3028 (BDD nodes).
Max. BDD size for intermediate product =       5075 (BDD nodes)
```

감사합니다