객체지향 CHILL 프로그램을 위한 시험모델 생성

김상운[+] ○ 서희석[+] 권용래[+] 이동길[*]
[+]한국과학기술원 전산학과
[*]한국전자통신연구원

Generating Test Models from OO CHILL programs

Kim, Sang-Woon[+] ○ Seo, Heui-Seok[+] Kwon, Yong-Rae[+] Lee, Dong-Gil[*]
[+]Dept. of Computer Science, KAIST
[*]Electronics and Telecommunications Research Institute.

요 약

객체지향 CHILL과 같은 병렬 객체지향 프로그램은 객체지향 개념과 함께, 효율적인 통신을 위해 병렬 프로그램의 다양한 동기화 통신 방법을 지원한다. 병렬 객체지향 시험에서는 이 특성을 모두 고려해야 한다. 본 논문에서는 병렬 객체지향 시험을 위해서 객체지향 CHILL 프로그램에 대한 시험 모델을 생성하는 방법을 제안한다. 먼저 UML의 상태 다이어그램과 시퀀스 다이어그램을 바탕으로 하여 시험 모델에서 객체지향 개념을 표현하고, Region 모드, Event, Buffer, Signal과 같은 객체지향 CHILL의 동기화 통신 방법을 표현할 수 있도록 UML 표현을 확장하며, 각각의 동기화 통신 방법에 대한 시험 모델을 생성 방법을 제안한다. 생성된 시험 모델은 UML을 바탕으로 하기 때문에, 기존의 UML 기반 시험 기법을 적용하기가 용이하다.

1. 서론

객체지향 방법론에서는 실세계를 반영하는 객체를 바탕으로 시스템을 분석, 설계, 구현함으로써 재사용성과 유지보수성을 향상시키는데 기여한다. 따라서 다양한 분야에서 객체지향 방법론을 적용하려는 노력이 이루어져 왔으며, 통신용 소프트웨어 분야에서도 객체지향 CHILL과 같은 병렬 객체지향 프로그래밍 언어가 개발되어 사용되고 있다[1][2]. 그러나, 병렬 객체지향 프로그램은 일반적인 객체지향 프로그램과는 다른 특징들을 가지기 때문에 병렬 객체지향 프로그램에 적합한 시험 기법이 요구된다.

일반적인 객체지향 프로그램에서 객체는 정보 은닉(information hiding)에 의해서 자신의 정보를 보호하게 되며, 프로그램은 객체들 간의 메소드 호출에 의해서 실행된다. 이와 같은 객체지향 프로그램을 시험하기 위해서는 프로그램에 대한 시험 모델을 구축하고, 생성된 시험 모델에서 시험 데이터를 생성하여 시험하게 된다. 현재의 많은 객체지향 시험에서는 객체지향 표준 모델링 언어인 UML(Unified Modeling Language)로 작성된 명세를 시험 모델로 이용한다[3][4][5].

그러나, 객체지향 CHILL과 같은 병렬 객체지향 프로그램에서는 객체들 간의 통신이 빈번하게 발생하며, 효율적인 통신을 위하여 이벤트나 시그널 같이 메소드보다 작은 단위의 메시지를 이용한 통신을 지원한다. 이와 같은 메시지에 의한 통신은 병렬 객체지향 프로그램의 중요한 행위이지만 기존의 UML로는 표현하지 못하며, 따라서 UML 기반 시험에서는 메시지에 의한 프로그램의 행위를 시험하는 것은 불가능하다. 그렇기 때문에 병렬 객체지향 시험을 위해서 객체지향 개념과 함께 다양한 통신 방법을 표현하는 시험 모델을 생성하고, 그로부터 시험 데이터를 생성하는 과정이 필요하다.

본 논문에서는 병렬 객체지향 시험의 전단계로 객체지향 CHILL 프로그램에 대한 시험 모델 생성을 목적으로 한다. 생성된 시험 모델은 메소드 호출에 의한 프로그램의 행위 뿐만 아니라 메시지와 같은 다양한 통신 방법에 의한 프로그램의 행위를 모델링한다. 이를 위해 UML에서 프로그램의 행위를 표현하기에 적합한 상태 다이어그램과 시퀀스 다이어그램을 확장한다. 생성된 시험 모델은 UML을 바탕 으로 하기 때문에 기존의 많은 UML 기반 시험 기법을 적용함으로써 시험 데이터를 생성할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 병렬 객체지향 시험과 관련하여 기존의 UML 기반 시험과 병렬 프로그램 시험에 대해서 살펴본다. 3장에서는 객체지향 CHILL 프로그램에 나타나는 통신 방법들의 특징들을 살펴보고, 이를 반영하여 시험 모델을 생성하는 방법을 제안한다. 4장에서는 결론 및 향후 연구 방향을 기술한다.

2. 관련 연구

2.1 UML 기반 시험 기법

UML 기반 시험은 객체지향 표준 모델링 언어인 UML을 시험 모델로 하여 시험 데이터를 생성하는 객체지향 시험 기법으로, 최근에 많은 연구가 이루어지고 있다. 특히, UML의 여러 다이어그램들 중 프로그램의 행위를 모델링 하는 상태 다이어그램과 시퀀스 다이어그램을 중점으로 연구되고 있다. 상태 다이어그램에서는 메소드에 의한 객체 상태의 전이를 표현함으로써 한 객체의 행위를 모델링하며, 시퀀스 다이어그램에서는 객체들 간의 메소드 호출을 표현함으로써 객체들 간의 행위를 모델링한다. 따라서, 대부분의 클래스 시험 기법들은 상태 다이어그램을 바탕으로 하고 있으며, 시퀀스 다이어그램은 통합 시험(integration testing)에 적합하다. 이와 같은 다이어그램들로부터 생성되는 시험 데이터들은 프로그램의 행위를 시험할 수 있는 메소드 호출 순서로 구성된다[4][5][6].

병렬 객체지향 시험에서는 먼저 메소드 호출과 메시지 전달에 의한 프로그램의 행위를 표현하는 시험 모델 을 생성하고, 이들로 이루어진 시험 데이터를 생성해야 한다. 그러나, 병렬 객체지향 프로그램에 적합한 명세 언어가 없기 때문에 프로그램으로부터 시험 모델을 생성해야 한다. 이 때 시험 모델은 상태 다이어그램과 시퀀스 다이어그램을 확장함으로써, 기존의 UML 기반 시험 기법을 적용을 쉽게 한다.

2.2 병렬 프로그램 시험

병렬 프로그램은 동시에 수행 가능한 여러 개의 프로세스로 구성되며, 그들 간에는 정보교환 및 수행 순서의 동기화와 같은 상호 작용이 발생한다. 이와 같은 병렬 프로그램의 중요한 특징으로는 프로세스들 간의 병행성(concurrency) 과 그에 의한 비결정성(nondeterminacy)이 있으며,

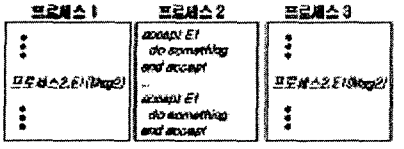

그림 1: 메시지 기반 병렬 프로그램에서의 메시지 경쟁

이들은 병렬 프로그램의 성능 향상에 기여한다. 그러나 병행성과 비결정성은 프로세스들이 동기화 될 때 여러가지 문제를 발생시킬 수 있다. 예를 들어 그림 1에서 프로세스 2는 프로세스 1의 메시지를 먼저 받고 프로세스 3의 메시지를 받을 수 있으며, 프로세스 3의 메시지를 먼저 받고 프로세스 1의 메시지를 받을 수도 있다. 이 경우에 2가지의 수행 시나리오를 모두 시험해야 한다. 즉, 병렬 프로그램 시험에서 통신에 의한 동기화는 중요한 문제가 되며, 프로그램을 시험 할 때 입력 데이터와 함께 시나리오 데이터를 확인해야 한다.

병렬 프로그램 시험 기법을 살펴보면, Taylor, Levine, Kelly는 병렬 프로그램의 수행 경로를 상태 그래프(state graph)로 표현하고, 그에 대해서 다섯 가지 계층의 시험 기준을 제시하여 시험 데이터를 생성하였다[7]. 그러나 일반적으로 프로그램의 모든 수행 경로를 모델링하는 것은 불가능하다. 따라서 프로그램의 모든 가능한 선후 관계가 아닌 항상 만족되는 선후 관계를 모델링하는 MHB(Minimal Happened-Before) 모델이 제안되었고, MHB 모델로부터 시험 데이터를 생성하였다[8].

3. 객체지향 CHILL을 위한 시험 모델 생성

병렬 객체지향 언어인 객체지향 CHILL은 객체의 역할을 반영하는 모레타 모드 (*MORETA : Module, Region, Task* 모드)로 구성되며 병렬 프로그램에서 중요한 효율적인 통신 방법을 지원한다. 먼저 객체지향 CHILL을 이루는 3가지의 모드부터 알아보면 다음과 같다.

- Module 모드
  다른 모드에의 종속되어지는 모드로 일반적인 클래스에 해당한다.
- Task 모드
  Module 모드와는 달리 자기 자신이 스스로 수행되는 컨트롤을 가지고 있는 메시지 통신 및 메소드 호출을 지원하는 프로세스로 객체지향의 액티브 객체(Active Object)이며 그 예로 JAVA의 쓰레드가 있다.
- Region 모드
  동기화 수단중의 하나인 Monitor를 구현한 특수한 클래스이다. Region 모드 내부의 메소드가 배타적으로 수행되고, 프로그램 어디에서도 접근 가능하다.

한편 효율적인 통신을 위한 동기화 수단으로는 Event, Buffer, Signal등의 메시지 기반의 방법이 존재한다. 하지만 메소드 기반의 통신방법을 기반으로하는 객체지향의 개념으로는 메소드 단계보다 세부적인 메시지 기반의 동기화 방법을 모델링 할 수가 없다. 따라서 본 논문에서는 그림 2의 기호를 추가하여 UML바탕의 시퀀스 다이어그램과 상태 다이어그램을 기반으로하는 시퀀스 시험모델과 상태 시험모델을 제안한다.

시퀀스 시험모델에서는 메시지 전달을 의미하는 점선을 추가한다. 화살표 머리에 따른 전달방식의 구분은 메소드 호출방식을 그대로 따른다. 또 Event 큐와 메시지 Buffer를 나타내는 상자를 추가하였으며 이들은 자신에게 메시지를 전달하는 프로세스와 메시지 이외의 정보를 저장하는 의미이다. 병행적 접근은 어느 객체에서도 동시에 접근 가능하고, 그 수행이 비결정적임을 의미한다.

상태 시험모델에서는 마찬가지로 메시지 전달의 경우 메소드 호출의 실선과 구분하여 점선을 사용하고, 상태를 나타내는 타원이 아닌 둥근 사각형으로 외부 저장공간을 표현한다. Fork/Join은 RT-UML(RealTime UML)에서 제안한 표기법으로, 메소드에 의해 상태 전이 도중 Fork에 의해서 메시지를 발생시키고, Join에서 상태전이가 멈추고 있던중 원하는 메시지를 전달받을때 상태전이를 계속 하게 한다. 이와 같은 새로운 표기법을 사용하여 객체지향 표기법을 바탕으로 객체지향 CHILL에 대한 시험 모델을 생성한다.

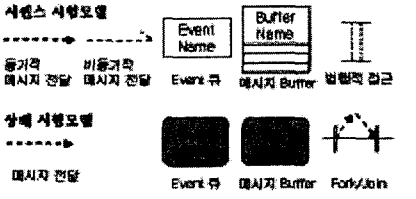

그림 2: 시퀀스 시험모델과 상태 시험모델에 추가한 모델 요소

3.1 Region 모드

Region 모드는 위험지역을 보호하는 Monitor를 객체화시킨 것으로 다음과 같은 3가지 특징을 만족시킨다.

- Region 모드의 생명주기는 프로그램과 같이 한다.
- 메소드에 대한 호출은 비결정적이다.
- 메소드의 실행이 배타적으로 발생한다.

위의 특성을 고려하여 시험 모델을 표현하면 시퀀스 시험모델은 그림 3과 같이 프로그램 전체의 생명주기와 Region 모드의 생명주기를 같게 만들고, 비결정적 메소드 호출을 일으키는 부분은 병행적 접근으로 나타냈다.

상태 시험모델에서는 Region 모드를 클래스로 간주하고 모델링한다. 확장된 모델 요소를 사용하지 않고서도 객체들의 병행적 수행과 각각의 메소드 호출에 대한 비결정적 순서를 보장해 줄 수 있다.

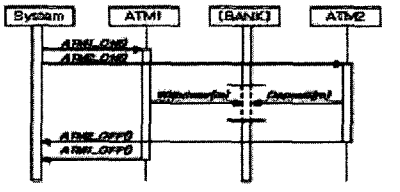

그림 3: Region 모드에 대한 시험모델 생성

3.2 Event

비동기적 특징을 가지는 메시지 동기화 방법인 Event는 CONTINUE/DELAY 메시지에 의해 수행된다. DELAY는 지정한 Event가 발생 할 때까지 현재의 프로세스를 지연시키는 역할을 수행하고, 다른 프로세스에서 그 이벤트에 대해 CONTINUE 메시지를 전달함으로써 지연되던 프로세스가 다시 실행되어진다. Event의 특징은 다음과 같다.

- DELAY/CONTINUE 메시지는 큐를 통해 전달되는 간접 통신 방법이다.
- CONTINUE 메시지는 비동기적이고, DELAY 메시지는 동기적이다.

이러한 특성을 고려한 시퀀스 시험모델은 그림 4과 같다. 모든 메시지는 목적 프로세서에 대한 지정을 하지 않기 때문에 Event와 관련된 프로세서를 저장할 큐가 필요하게 되고, 이를 위해 앞에서 제안했던 Event 큐를 사용한다. Event 큐는 메시지에 의해서만 통신 가능하다. 그리고 메시지 전달은 메소드의 수행중에 나타남을 명시해주었다. 이는 나머지 동기화 방법에서도

마찬가지로 나타난다.
상태 시험모델은 시퀀스 시험모델에서와 마찬가지로 Event에 관계된 프로세스들을 저장해놓을 Event 큐가 필요하고, 이 큐로의 메시지 전달은 Fork/Join에서 메소드 수행중 메시지를 발생하여 접근하게 된다. Fork에 의해 DELAY 메시지가 큐로 접근하고, 다른 객체에 의해 비동기적 메시지인 CONTINUE가 전달되어지면 큐로부터 메시지를 받아 Join에의 조건을 만족하여 다음 상태로 전이가 가능하다.

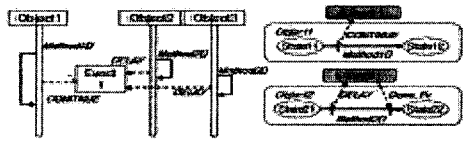

그림 4: Event에 대한 시험모델 생성

3.3 Buffer

저장장소를 사용하여 메시지와 정보를 동기적으로 전달하는 Buffer는 SEND/RECEIVE 메시지를 사용한다. Buffer는 다음과 같은 특성을 지닌다.

- SEND/RECEIVE 메시지는 Buffer를 통하는 간접 통신 방법이다.
- 메시지 Buffer에 대해 SEND/RECEIVE 메시지는 동기적이다.

시퀀스 시험모델은 SEND 메시지와 RECEIVE 메시지를 동기화 메시지 전달로 표현하고, Event 시험모델에서와 같이 메시지 전달은 메소드의 호출중에 발생함을 표현한다.
상태 시험모델은 SEND/CONTINUE 메시지가 동기적 전달방법으로 Fork와 Join을 통해 동기적인 표현을 한다. 외부의 저장공간으로 메시지 Buffer를 사용한다.

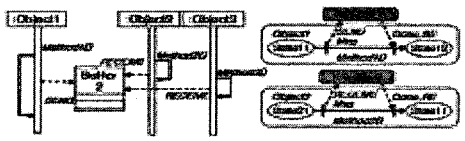

그림 5: Buffer에 대한 시험모델 생성

3.4 Signal

유일한 직접 통신 방법으로 Buffer와 같이 메시지 이외의 정보를 전달하는 Signal은 SEND/RECEIVE 메시지 전달을 통해서 수행된다. Signal을 표현하기 위해서는 다음과 같은 특성을 지녀야 한다.

- SEND 메시지에서 목적 프로세스를 지정하는 직접 통신 방법이다.
- SEND 메시지는 비동기적 방법이다.

Signal을 모델링 하는데 있어서 나머지 2가지 메시지 기반의 동기화 방법과 가장 큰 차이는 직접 통신이라는 것이다. 즉 추가적인 모듈을 필요로 하지 않는다는 것이다. RECEIVE를 기다리는 객체에 Signal을 보낼 때는 시간적으로 나중에 Signal이 도착하도록 표현한다.
상태 다이어그램에서 한 객체가 다른 상태로 전이하는 도중 Fork를 사용하여 SEND 메시지를 전달하고, RECEIVE 메시지는 Join에서 비동기적으로 메시지를 기다리지만 Signal이 직접 통신방법으로 Event 큐나 메시지 Buffer를 사용하지 않고, 객체 내부에 Signal을 기다리는 새로운 상태를 추가하여 모델링 한다.

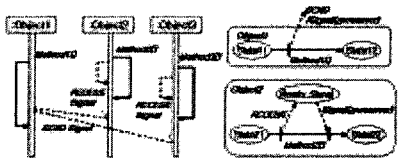

그림 6: Signal에 대한 시험모델 생성

4. 결론 및 향후 연구방향

객체지향 CHILL로 구현된 병렬 객체지향 프로그램은 일반적인 객체지향 프로그램이 가지는 메소드 호출과 함께 Region 모드, Buffer, Event, Signal에 의한 통신 방법을 지원한다. 본 논문에서는 이와 같은 객체지향 CHILL의 통신 방법들을 고려하여 시험 모델을 생성하는 방법을 제시하였다. 즉, 각각의 통신 방법에 대해서 상태 다이어그램과 시퀀스 다이어그램을 확장한 시험 모델을 제시하였다. 생성되는 시험 모델에서는 메소드 호출에 의한 프로그램의 행위와 함께 메시지 전달에 의한 프로그램의 행위를 모델링하기 때문에, 병렬 객체지향 프로그램의 시험 모델로 적합하다. 또한 시험 모델은 UML의 상태 다이어그램과 시퀀스 다이어그램을 바탕으로 하고 있기 때문에, 기존의 UML 기반 시험 기법을 적용하기가 용이하다.
앞으로 생성한 시험 모델에서 시험 데이터를 생성하는 연구가 필요하다. 시험 데이터 생성 방법은 기존의 UML 기반 시험 기법을 바탕으로 메시지 부분에 대한 확장에 의해서 만들어질 수 있다. 또한 본 논문에서는 객체지향 CHILL 프로그램에 대한 시험 기법을 제안하였는데, 이를 확장하여 일반적인 병렬 객체지향 프로그램에 적용할 수 있도록 시험 기법을 일반화 시키고자 한다.

참고 문헌

[1] International Telecommunication Union(.ed): CCITT High Level Language (CHILL), ITU-T Recommendation Z.200, 1997

[2] J. F. H. Winkler and G. DieBl, "Object CHILL - An Object Oriented Language for System Implementation", *Proceedings of the 1992 ACM annual conference on Communications*, Mar, 1992

[3] H. Eriksson and M. Penker, *UML Toolkit*, John Wiley & Sons, INC. 1998

[4] Y. G. Kim, H. S. Hong, S. M. Cho, D. H. Bae and S. D. Cha, "Test Cases Generation from UML State Diagrams" *IEEE Proceedings Software*, vol. 146, no. 4, pp. 187-192, Aug 1999

[5] B. Y. Tsai, S. Stobart, N.Parrington and I. Mitchell, "An Automatic Test Case Generator Derived form State-based Testing", *Proceedings of Asia-Pacific Software Engineering Conference*, pp. 270-277, Dec 1998

[6] P. C. Jorgensen and C. Ericson, "Object-Oriented Integration Testing" *Communications of the ACM*, vol. 37, no. 9, pp. 30-38, Sep 1994

[7] H. S. Bae, I. S. Chung, H. S. Kim and Y. R. Kwon, "Validation of Timing and Communication Constraints in Real-Time Parallel Programs" *PhD Thesis, Korea Advanced Institute of Science and Technology(KAIST)*, 1999

[8] R. N. Taylor, D. L. Levine, and C. D. Kelly, "Structural Testing of Concurrent Programs", *IEEE Trans. on Software Engineering*, vol. 18, no. 3, pp. 206-215, Mar 1992