Esterel 문법구조 바탕의 오토마타 생성*

이철우[O+] 김철주[++] 윤정한[+] 한태숙[+] 최광무[+]
[+] 한국과학기술원 전산학과
{posticaruss[O], dolgam}@gmail.com han@cs.kaist.ac.kr choe@kaist.ac.kr
[++] 삼성전자 DMC 연구소
chuljoo.kim@gmail.com

Syntactic Approach to automata generation for Esterel

Chul-Woo Lee[O+] Chul-Joo Kim[++] Jeong-Han Yun[+] Taisook Han[+] Kwang-Moo Choe[+]
[+] Division of Computer Science, KAIST
[++] DMC R&D Center, Samaung Electronics Co., Ltd.

1. 서론

반응(reactive)형 실시간(real-time) 시스템은 항공기나 자동차와 같은 물리적인 환경과 반응하는 기계의 제어 장치에 내장된다. 이러한 제어 장치 기능은 안전성이 매우 중요하기 때문에 안전성에 대한 충분한 분석과 검증이 필요하다.

Esterel[1][2][3][4]은 절차형 동기식 언어이다. 동기식 언어[5]는 실제 시간을 논리적 단위시간(instant)으로 나누어, 언어 수준에서 프로그램의 수행 시간을 단위 시간 단위로 관리할 수 있도록 한다. Esterel은 유한 상태 기계를 기반으로 한 정형적(formal) 의미구조를 가지고 있어 정적 검증에 매우 용이하다. 유한 상태 기계와 동일한 의미를 가지고 있기 때문에 Esterel 프로그램을 오토마타로 변환하면 기존의 오토마타 이론이나 모델 체킹 기법들을 이용한 분석이 가능하다[6].

Esterel 컴파일러[7][8]는 프로그램의 시뮬레이션을 통해 비교적 빠른 시간 안에 오토마타를 생성할 수 있는 옵션을 지원한다. 하지만 시뮬레이션을 통한 프로그램의 오토마타 변환은 상태 폭발 문제에 영향을 받기 때문에 큰 프로그램을 대상으로 하기는 현실적으로 어렵다[11].

기존 연구[9]에서 Esterel 프로그램의 동작 의미(operational semantics)를 기술하기 위하여 오토마타를 LTS[10] 형태로 정의하였다. 즉, LTS로 기술된 오토마타는 프로그램의 의미에 따라 수행되는 진행 과정을 정형적으로 표현한다. 프로그램의 동작 의미가 외부 환경에 따라 달라지기 때문에 이 생성 규칙을 바탕으로 Esterel 프로그램에 대한 오토마타를 생성하기 위해서는 외부 환경에 대한 고려가 필요하다. 결과적으로 프로그램 전체 구조를 나타내는 오토마타를 직접적으로 표현하기 위해서는 가능한 모든 외부 환경에 대한 프로그램의 수행 동작을 살펴봐야 한다. 결국 이 오토마타 생성 방법은 시뮬레이션과 유사한 추가적인 프로그램 수행 과정을 필요로 한다.

위와 같은 프로그램 수행 과정을 거치지 않고 오토마타를 생성하기 위해서 본 논문에서는 문법 구조 바탕의 오토마타 생성 규칙을 제안한다. 우리의 생성 규칙은 프로그램의 수행 과정 없이 문법 구조를 바탕으로 오토마타를 생성한다. 또한 우리의 오토마타 생성 규칙을 바탕으로 생성 과정 중에 오토마타 축약 과정을 적용할 수 있다.

2. 본론

본 논문에서는 kernel Esterel[3] 중 "`signal S in p end`"을 제외한 "`emit S`", "`pause`", "`nothing`", "`exit t`" 4개의 단위문장과 sequence, signal test, parallel, loop, exception handling, suspension 6개의 블록문장을 대상으로 한다.

문장의 오토마타를 생성할 때 기본적으로 고려해야 할 점 두 가지가 있다. 첫 번째, Esterel 프로그램은 적어도 하나의 단위시간을 소모한다. 프로그램 전체에서 "`pause`"를 한 번도 사용하지 않았다면 이 프로그램은 한 단위시간만을 소모하는 프로그램이 된다.

두 번째, 문장들의 결합을 통한 블록문장을 생성할 때 의 동일한 단위시간에 일어나는 시그널 입출력을 일치시켜 주어야 한다. 그림 1의 예제를 살펴보자. "`pause; emit O1;`" 과 "`emit O2; pause;`" 두 문장의 오토마타는 (3)과 (4)에 나타나 있다. 위의 두 문장을 순차적으로 수행할 때 시그널 O1과 O2는 동일한 단위시간에 켜진 상태로 존재한다. 이 문장에 대한 오토마타를 표현하기 위해서는 (3)의 마지막 단위시간과 (4)의 첫

* 본 연구는 교육과학기술부/한국연구재단 우수연구센터 육성사업(2010-0001712)과 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2010-C1090-1031-0004)의 지원으로 수행되었음.

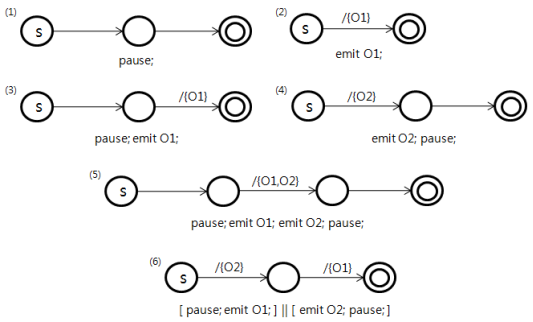

그림 1 오토마타 예제

번째 단위시간을 일치시켜주어야 한다. 이를 고려한 합성된 결과 오토마타는 (5)와 같은 형태로 표현되어 시그널 O1과 O2가 동일한 단일시간에 나타나 있음을 볼 수 있다. 두 문장을 병렬 수행할 때의 예를 살펴보자. 두 문장을 병렬 수행할 때 동일한 단위시간은 수행되는 순서가 같은 단위시간이다. (3)의 첫 번째 단위시간과 (4)의 첫 번째 단위시간은 수행 순서가 같기 때문에 동일한 단위시간으로 고려해야 한다. (6)에서 병렬 수행 오토마타를 나타내고 있다.

위와 같은 점들을 고려하여 우리는 Esterel의 문법구조에서부터 직접 오토마타를 생성하는 규칙을 제안한다. 오토마타 생성 규칙의 형식은 [11]에서 정의한 형식을 참고하였다. 단위문장의 오토마타는 하나의 형태를 가진다. 블록문장은 문장 간의 합성으로 구성되기 때문에 블록문장을 구성하는 각 문장의 오토마타들을 합성하여 블록문장에 대한 오토마타를 표현한다. 합성 과정에서 블록문장의 의미에 따라 노드와 간선을 대상으로 병합 및 삭제를 수행한다.

exit문에 의해서 발생하는 예외 처리를 위해 exit문에 대한 오토마타 생성 시 트랩 노드를 추가한다. 이 트랩 노드는 동일한 예외 식별자를 가지는 trap문 바깥의 외부 간선과 연결된다. suspend문은 시그널이 켜짐 상태인 단위시간 동안 문장의 순차 수행을 멈춘다. 이를 나타내기 위해 suspend문을 구성하는 문장의 내부 노드에 재귀 간선을 추가한다.

병렬문에서는 병렬 수행될 오토마타의 종료 시점에 따라서 세 가지 경우가 발생할 수 있다. 첫 번째는 두 오토마타가 동일한 단위시간에 종료 될 경우이다. 두 번째는 두 오토마타가 서로 다른 단위시간에 종료 될 경우이다. 즉, 어느 한 오토마타가 소모하는 단위시간이 더 많을 경우이다. 마지막으로 두 오토마타 중 어느 하나라도 종료 노드 도달 없이 끝나는 경우이다. 예를 들어, 예외 발생, loop문의 수행 등으로 종료 노드에 도달하지 못하는 경우가 발생할 수 있다. 위 세 가지 경우를 고려하여 병렬 수행 오토마타 생성 알고리즘을 통해 병렬문에 대한 오토마타를 생성한다.

3. 결론

본 논문에서는 Esterel 문법 구조 바탕의 오토마타 생성 규칙을 제안하였다. 제안한 규칙들은 Esterel 프로그램의 수행 과정 없이 직접적으로 프로그램의 전체 구조를 표현할 수 있는 오토마타를 생성한다. 생성된 오토마타는 반응 시스템 상에서 일어나는 각 단위시간에서의 입력 시그널과 출력 시그널의 상태들을 나타낸다.

현 오토마타 생성 규칙에서는 의미구조상수행 불가능한 경우를 제외하지 않았다. 추후 이에 대한 분석을 통해서 생성된 오토마타를 최적화 할 수 있는 방법에 대한 연구를 진행할 계획이다. 이를 바탕으로 생성 규칙 바탕의 요약 분석이나 생성된 오토마타를 이용하여 Esterel 에서 나타날 수 있는 특이한 성질들을 분석할 수 있는 분석 도구를 개발할 예정이다.

참고문헌

[1] D. Potop-Butucaru, S. A. Edwards, and G. Berry, "Compiling ESTEREL". Springer, 2007.
[2] G. Berry, "The Esterel Primer", Included in the Esterel distribution, Available on http://www-sop.inria.fr/meije/esterel/esterel-eng.html. , 1998
[3] G. Berry, "The Constructive Semantics of Pure ESTEREL", Draft book available at http://www-sop.inria.fr/meije/esterel/esterel-eng.html, , 1999.
[4] G. Berry, "The Foundations of Esterel,", Proof, Language, and Interaction: Essays in Honour of Robin Milner, pp.425-454, MIT Press, 2000.
[5] N.Halbwachs, "Synchronous programming of reactive systems", Kluwer Academic Publishers, Dordrecht, 1993.
[6] G. Berry, "Preemption in Concurrent Systems", In Proceedings of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science, vol. 761, pp. 72-93, 1993.
[7] "Esterel Compiler version 5.21", http://www-sop.inria.fr/meije/esterel/esterel-eng.html.
[8] "CEC: The Columbia Esterel Compiler", http://www.cs.columbia.edu/~sedwards/cec/.
[9] S. Tini, "An axiomatic semantics for Esterel", Theoretical Computer Science, 2001.
[10] GD Plotkin, "A Structural Approach to Operational Semantics", Lectures Notes, Aarhus University, Aarhus, Denmark, 1981.
[11] 김철주, 윤정한, 서선애, 최광무, 한태숙, "Esterel에서 근사-제어흐름그래프의 효율적인 생성, 정보과학회논문지 : 컴퓨팅의 실제 및 레터 제15권 제11호, pp. 876~880, 2009. 11.