PtolemyII 의 CCS 도메인 구현

황혜정[0], 김일곤, 최진영
{hjhwang[0], igkim, choi}@formal.korea.ac.kr

The Implementation of CCS Domain in PtolemyII

Hye-Jung Hwang[0], Il-Gon Kim, Jin-Young Choi

Dept. of Computer Science and Engineering,
Korea University

요 약

PtolemyII 의 CSP 도메인은 병렬 시스템을 효과적으로 명세할 수 있는 프로세스 알제브라 언어인 CSP 를 구현한 것이다. CCS 도 프로세스 알제브라 언어로써 정형적으로 병렬시스템을 명세하고 검증할 수 있다. 이 두 언어는 비슷한 목적으로 가지고 개발되었으나 통신의 세부적인 부분의 개념에 있어서 차이가 존재한다. 특히 CCS 를 기반으로 하여 개발된 ACSR 의 경우 실시간 시스템을 정형적으로 명세하고 검증하기 위해 필요한 시간과 자원의 개념을 추가하였다. 본 논문에서는 ACSR 도메인을 PtolemyII 에 구현하기 위한 단계적인 방법으로 CCS 와 CSP 의 차이점을 밝혀서 , 기존의 CSP 도메인의 Rendezvous 알고리즘을 CCS 기반의 통신이 이루어 질 수 있도록 수정하여 PtolemyII 에 CCS 도메인을 구현하였다.

1.서론

내장형 시스템이란 범용적인 목적을 가진 워크스테이션이나 데스크탑 컴퓨터 등과는 달리 특정 기능을 수행하기 위해, 컴퓨터 하드웨어와 소프트웨어가 조합된 전자제어 시스템을 일컫는다. 내장형 시스템은 가전제품에서부터 항공기 제어시스템, 원자력 발전소에 이르기 까지 폭 넓은 범위에 사용되고 있다. 특히 정확성을 요구하는 원자력 발전소에 쓰이는 내장형 시스템의 경우 정형적 설계와 개발이 필수적으로 요구된다. 이러한 내장형 시스템의 발달과 함께 PtolemyII[3]는 내장형 시스템의 통합적이고도 정형적인 개발을 위해 버클리에서 개발된 도구이다. PtolemyII 는 여러 도메인을 제공하는데, 그 중 프로세스 알제브라 언어인 CSP[1]를 구현한 CSP 도메인[7]을 가지고 있다.

기존의 오토마타는 병렬 시스템(concurrent system)을 정확히 표현하는데 한계가 있었다. 이러한 한계 때문에 병렬적으로 통신하는 시스템을 묘사하고 설계하기 위해 CSP, CCS[6], ACP[2]와 같은 시간의 개념이 없는 프로세스 알제브라 언어들이 개발 되었다. 그러나 프로세스 동기화를 위한 시간개념과 공유자원의 활용도가 중요한 부분을 차지하는 실시간 시스템(real-time system)을 정형적으로 명세하고 검증 할 수 없었다. 이러한 것을 가능하게 하고자 하여 CCS 를 기반으로 ACSR[5]이 개발되었다. 본 논문에서는 최종적으로 PtolemyII 에 실시간 시스템을 위한 ACSR 도메인을 추가하기 위해 단계적 방법으로 CCS 와 CSP 의 차이를 밝혀서 CCS 도메인을 PtolemyII 에 구현하였고, 알고리즘을 제시하고자 한다.

2. 본론

2.1 PtolemyII

PtolemyII[4]는 버클리에서 내장형시스템을 디자인 하고 모델링하고 시뮬레이션하기 위해 자바로 구현된 도구이다. 시스템을 구현할 경우 하나의 프레임워크[3]를 제공하는 도구로는 구현하고자 하는 시스템을 완벽하게 구현하는 것이 불가능하다. 예를 들면 Statechart 의 경우는 Finite State Machine 의 의미론을 기반으로 하여서 모델을 명세할 수 있으나, 실세계의 지속되는 시간을 모델에 명세할 수 없다. 이러한 제한점을 극복한 것이 PtolemyII 이다. PtolemyII 의 큰 장점은 여러가지 프레임워크를 하나의 도구에서 구현할 수 있고, 각각 다른 프레임워크로 디자인된 모델을 계층적으로 연결하여 디자인 할 수 있다. 또한, PtolemyII 는 그래픽적인 에디터 Vergil 를 제공함으로써 보다 디자이너에게 친숙한 환경을 제공한다. 현재 PtolemyII 에서는 총 11 개[4]의 프레임워크를 제공하고 있으며 현재 매뉴얼에 추가되지 않고 도구에 이외에 추가된 것 2 개를 합하여 총 13 개의 다른 의미론(semantic)을 가진 프레임워크가 존재한다. 프레임워크는 다른 말로 '도메인'이라고도 일컬어지는데, 이러한 여러 도메인들은 각 구성 요소들 사이에 계산(computation)을 지배하는 물리적인 규칙을 말하는 'Model of computation'이다. 이 예로 Discrete Event, PetriNet, Continuous Time 등을 들 수 있다.

2.2 CSP (Communication Sequential Processes)

CSP[1]는 Hoare 에 의해서 1980 년에 개발된 프로세스알제

브라의 언어로써 통신 프로토콜의 정형적 설계 및 검증을 위해서 개발 되었다.CSP 의 기본적인 연산으로는 외부 *choice* (□), 내부 *choice* (⊓), *concurrency* (||), *interleaving* (|||), *Hiding* (\)과 *Composition* (;)이 있다. CSP 에서 프로세스간의 통신 과 프로세스의 행위의 기본단위는 이벤트이며 이벤트는 더 이상 나눠지지 않는 기본단위로써 Atomic 이거나 행위와 관련된 데이터를 포함할 수 있다. 프로세스는 행위를 가지는 객체의 행동이며 이벤트에 연관되어진다. 또, 프로세스 'P'에 연관되는 모든 이벤트들의 집합을 'α(P)'라고 표시하며, 알파벳 혹은 인터페이스로 일컬어진다. *Hiding* 은 이러한 이벤트를 외부에서 보이지 않게 하는 것이다. CSP 의 중요한 기본적 연산의 하나인 *choice* 를 살펴보자. 외부(external) *choice* (□)는 프로세스의 외부 환경에 의해서 결정되어진다. 반면에 내부 (internal) *choice* (⊓)는 외부 환경에 독립적으로 프로세스안에서 결정되어 지고, 프로세스 외부에서 보이지 않는다. 내부(internal) *choice*(⊓)같은 경우는 비결정성(non-determinism)이 존재하게 된다. *concurrency* (||)연산은 'P||Q' 로 표현되어지며, 프로세스 'P'가 프로세스 'Q'와 병렬적으로 행동하는 것을 뜻한다. 그리고 두 프로세스가 통신을 할 경우는 공통되는 이벤트가 두 프로세스에 존재할 때 두 프로세스는 단일 채널을 기반으로 하여서 메시지 통신을 하는 것을 의미한다. *interleaving* (|||) 연산은 *concurrency* (||)연산과 표현과 비슷하나 두개의 프로세스가 서로 상호작용하지 않으면서 행위하는 것을 표현한다. 이외에 프로세스간의 동치성을 표현하는 *trace, failure, refusal, divergence* 가 있다.

2.3 PtolemyII 에서의 CSP 도메인

PtolemyII 의 CSP 도메인[7]에서는 단방향 채널을 사용하여서 메시지 통신을 하는 프로세스들의 네트웍 시스템을 모델링 할 수 있다. PtolemyII 의 CSP 도메인에서는 Rendezvous 를 사용하여서 프로세스간 통신을 한다. Rendezvous 통신은 프로세스가 통신을 하기 원할 때 상대방 프로세스가 통신준비가 될 때까지 기다린 후 준비가 되어지면 통신을 하게 된다. PtolemyII 의 CSP 도메인에서 [그림 1]에서 보는 바와 같이 CSP 의 guarded statement 인 CIF(ConditionalIF)와 CDO(ConditionalDo)를 구현 하였다. 또 통신의 경우는 통신을 하기위한 기본적인 동작으로 CSP 의 ?(input) !(output)을 나타내는 put()과 get()으로 구현했다. [그림 1]의 guard communication statement 는 자바의 쓰레드로 구현되어진다. guard 는 statement 의 수행 가능 여부를 논리값으로 표현하고, communication 은 수행 할 때 입출력 여부를 표현한다. 결국 guard 가 참일 경우 입력 혹은 출력을 수행하는 statement 가 수행되어진다. [그림 1]에서 guard 문장 안에 ' [] ' 로 구분되어지는 문장들은 각각 쓰레드로 표현되어서 CSP 의 비결정성을 나타낸다. PtolemyII 의 CSP 도메인 커널에서는 기존의 Rendezvous 를 CSPReceiver 에 구현했다. 또 guard 문장들은 *ConditionalSend* 혹은 *ConditionalReceiver* 로 구현되어있으며 이 클래스 안에는 쓰레드를 제어하기 위해서 보완된 알고리즘이 첨가 되어있다. 또 livelock 의 발생과 deadlock 의 발생을 제어하기 위해 제한적인 알고리즘을 구현하고 있다. PtolemyII 에서 구현된 CSP 는 기존의 CSP 에 시간을 추가한 것으로 Rendezvous 를 하기위해 각 프로세스들은 임의의 시간동안 정지되어지는 특징을 가지고있다[7].

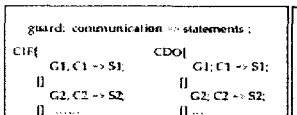

그림 1. guard ,CIF, CDO

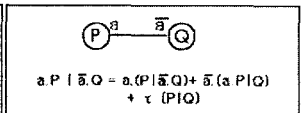

그림 2. CCS 의 tau, restriction

2.4. CSP 와 CCS 의 차이점

CCS[6]란 a Calculus of Communicating Systems 의 준말로써 Milner 에 의해 1980 년에 발표되었다. CCS 와 CSP 는 표현력은 같지만 그 문법이나 법칙에서 차이를 보인다. 본 논문에서는 CCS 와 CSP 에서 의 프로세스의 동치관계를 증명할 수 있는 의미론의 차이는 생략하도록 하겠다. CCS 에서는 CSP 의 프로세스를 에이전트(agent), 이벤트를 행위(action)라고 일컫는다. CCS 는 CSP 와 다르게 [그림 2]에서 보듯이 두개의 에이전트가 하나의 채널로 연결되어 있을 경우 각 채널의 양끝에서 발생하는 action (a)과 complement action($\bar{a}$)으로 구분하여 나타낸다. CSP 는 pure parallelism (||)과 Interleaving(|||)의 개념을 모두 채택하고 있는 반면 CCS 의 경우 기본적으로 pure parallelism 을 제공하지 않는다. 모든 에이전트들은 interleaving 하게 통신하며 composition(|)을 사용하여서 동시성(concurrency)의 개념을 나타낸다. 그리고 CSP 에서는 두 가지의 choice 를 제공하는 반면 CCS 는 선택적(alternative) choice(+) 하나만이 제공된다. [그림 2]의 아랫 부분에서 보듯이 어떤 에이전트에 특정한 action 을 외부에서 보이지 않도록 제한을 가하는 Restriction 이 있다. CSP 에서 두개의 프로세스에 공통적으로 연관된 이벤트가 있을 경우 그 이벤트가 발생할 때 동기화 되어서 프로세스간 통신이 이루어지게 된다. 그러나 CCS 는 action 과 그것의 complement action 이 발생이 되더라도 tau(τ)가 발생되지 않으면 agent 간의 통신은 이루어 지지 않는다. tau(τ) 란 외부에서는 보이지 않는 내부 action 으로써 CCS 에서는 Restriction 이 가해질 경우 어떤 action 과 complement action 이 발생하지 않고, tau(τ)의 발생으로 인해 두 agent 간에 통신이 일어나게 된다. 즉 프로세스간의 동치성을 증명하는 것을 제외한 상태에서 CCS 와 CSP 의 차이점 중 가장 큰 것은 [그림 2]에서 나타내고 있는 restriction 과 composition 에서 볼 수 있는 tau(τ)의 발생에서 찾아볼 수 있다.

2.5. PtolemyII 에서 CCS 도메인 구현 알고리즘

[그림 2]에서 보여진 수식은 action 'a' 가 발생하거나 complement action, ' $\bar{a}$ '가 발생하여 각각 에이전트 P 와 Q 가 한 스텝씩 다음으로 진행하거나 tau(τ)가 발생하여서

통신이 이루어지는 세가지 중의 하나가 발생하는 것을 의미한다. 즉 'a'와 '$\overline{a}$'가 발생해도 임의적으로 tau(τ)가 발생할 때만 통신이 일어나게 된다. 또 restriction 이 가해졌을 경우는 tau(τ)의 발생과 함께 통신이 이루어져야 한다. 여기서부터는 에이전트를 PtolemyII 에 맞춰 설명하기 위해 ' 프로세스' 로 명칭을 바꾸도록 하겠다. PtolemyII 의 커널의 구조를 사용하면서 위의 두가지 조건을 만족시키기 위해 본 논문에서는 tau 구조체와 tau flag, restriction 매개변수를 기존의 CSP 도메인의 커널에 추가하는 방법을 제안하고 [그림 3]과 같이 기존의 Rendezvous 알고리즘을 수정하였다. 굵은 선으로 표시된 부분이 CCS 의 개념을 Rendezvous 에 적용하기 위해 추가된 부분이다. 우선 통신을 하려고 기다리는 프로세스들은 보내거나 받는 역할을 하며, 통신 하고자 하는 프로세스가 아직 준비 되어지지 않았을 경우에는 각각의 프로세스는 Getwaiting(receiver) 혹은 Putwaiting(sender) flag 를 사용하여서 자신이 기다리는 것을 표시한다. 그리고 통신하려는 상대 프로세스가 도착했을 경우 다른 쪽의 flag 를 확인하고 Rendezvous 를 시도하게 된다. 위에서 언급된 두 가지의 CCS 와 CSP 의 차이점 중 [그림 2]의 첫번째 수식을 구현하기 위해 tau 구조체와 tau flag 가 사용되어진다. 일단 통신을 하기 위한 프로세스는 tau 구조체에 자신의 ID 와 token 을 등록하게 되고 tau flag 를 확인한 뒤 통신을 하게된다. 즉 tau 구조체는 get(),put()메소드가 발생하더라도 통신이 항상 일어나지는 않도록 하기 위한 클래스이다. Restriction 이 가해진 경우를 제외하고는 tau 의 임의적 발생을 표시하기 위해 임의 적으로 tau 를 결정하고 그것이 발생했다는 flag 를 참으로 만든다. 여기서 임의적으로 tau 를 결정할 수 있는 것은 PtolemyII CSP 도메인 에서는 Rendezvous 를 하기 위해서 각 프로세스들이 원하는 만큼 지연될 수 있기 때문에 tau 역시 제공되는 시간의 개념을 따라서 얼마간의 간격 다음에 임의적으로 tau flag 를 설정 할 수 있게 된다. 두 번째의 차이점인 restriction 을 구현하는 방법으로 restriction 매개변수를 PtolemyII 의 커널인 IOPort 클래스에 추가하여 확장하는 것이다. PtolemyII 에서의 통신은 각각의 프로세스는 서로 가지고 있는 IOPort 를 연결한 채널을 통해서 통신이 이루어진다. 이때 각각의 프로세스에 연결된 IOPort 안의 get() 또 IOPort 를 부모로 두고 있는 TypedIOPort 클래스 안의 send()함수를 이용한다. 그러나 get()과 send()함수에서는 getReceiver() 메소드를 사용하여서 각 채널에 연결된 Receiever 의 내부에 정의된 get()과 put()함수를 호출하여 통신을 한다. 즉 CSP 도메인의 경우는 CSPReceiver 에 구현된 get()과 put()메소드가 사용 되는 것이다. 그러므로 IOPort 안에 Restriction 매개변수가 참일 경우는 tau flag 를 참으로 설정하여서 통신이 이루어 지게 된다. Restriction 의 경우는 IOPort 의 매개변수로 추가되어지므로 알고리즘에는 표시되어 지지 않는다. PtolemyII 는 guard 문장을 쓰레드로 표현함으로 인해서 추가적인

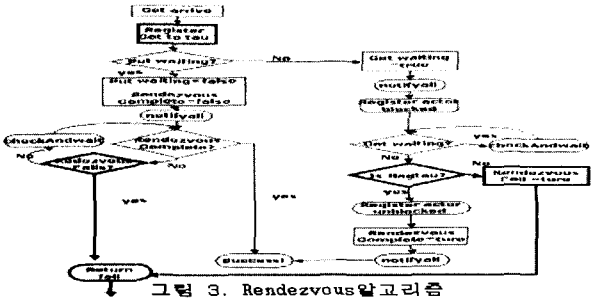

그림 3. Rendezvous알고리즘

알고리즘이 사용되었다. 이 알고리즘은 세가지 경우로 나누어 지고 각각의 경우는 Sender 와 Receiver 가 바뀌어서 해석 될 수 있다. 첫번째는 put 이 기다리고 있고 ConditionalReceiver 가 도착했을 경우이다. 이 경우는 ConditionalReceiver 가 통신할 수 있는 첫번째 Receiver 인지를 확인하고 참일 경우는 다시 tau flag 를 확인하여 tau flag 의 참 거짓에 따라서 통신의 여부가 결정된다. 두번째 ConditionalSender 가 처음으로 도착했을 경우, 세번째 ConditionalSender 가 ConditionalReceiver 를 기다리고 있는 상태에서 ConditionalReceiver 가 도착한 경우 모두 Rendezvous 할 수 있는 조건을 확인한 다음 tau flag 를 확인하여서 그 값에 따라 통신의 여부가 결정되어진다.

3. 결론 및 향후과제

CCS 와 CSP 의 차이점을 tau 구조체와 tau flag, 그리고 IOPort 의 restriction 매개변수를 넣음으로써 기존의 PtolemyII 의 CSP 도메인의 Rendezvous 알고리즘을 수정하였다. 이것은 CCS 도메인의 구현을 가능하게 하였다.

향후과제로는 ACSR[5]도메인을 PtolemyII 에 구현하고자 한다.

4. 참고문헌

[1] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall International,1985

[2] Bergstra, J.A., & J.W. Klop, *Algebra of communicating processes*, Proc. Of the CWI Symp. Math & Comp. Sci., eds. J.W. de Bakker, M. Hazewinkel & J.K. Lenstra, pp.89-138, North-Holland, Amsterdam 1986.

[3] Edward A. Lee, What' s Ahead for Embedded Software? IEEE Computer Magazine, pp. 18-26, September 2000,

[4] Edward A. Lee 외 17 명 Heterogeneous Concurrent Modeling and Design in Java, UCB/ERL M01/12, University of California, Berkeley, March 15, 2001

[5] 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

[6] Robin Milner, Communication and Concurrency, Prentice Hall international Ltd, 1989

[7] Neil Smyth, Communicating Sequential Processes in Ptolemy II, University of California, berkeley , December 15, 1998