한국시뮬레이션학회 '99춘계학술대회 논문집 1999.4.17 동의대학교

관계대수를 이용한 이산사건 시스템 모델링

홍기정, 김탁곤

한국과학기술원, 전기.전자공학과
시스템 모델링 시뮬레이션 연구실
URL:http://smsl.kaist.ac.kr
E-mail:{kjhong@smslab, tkim@ee}.kaist.ac.kr

Abstract

본 논문은 이산사건 시스템 모델링에 관계대수를 이용함으로써 이산사건 시스템 모델을 시뮬레이션 할 때 data consistency를 보장하는 방법을 다룬다. 복잡한 이산사건 시스템은 모델링 및 분석에 형식론적인 프레임웍이 필요하며 모델의 추상화와 재사용이 용이한 DEVS형식론이 널리 사용되고 있다. 본 논문에서는 DEVS형식론으로 기술된 모델을 정보의 손상 없이 관계대수형 모델로 변환하여 관계대수형 이산사건 모델로써 이용하는 방법론을 제시한다.

1. 서론

이산 사건 시스템을 모델링하고 시뮬레이션하기 위한 형식론에는 여러 가지 종류가 있다. 그중에 우리가 관심을 가지는 것은 DEVS형식론에 기반 한 모델링과 시뮬레이션이다. DEVS형식론은 계층적 모듈라한 형태로 모델링하는 방법을 제시한다. 이러한 형태의의 형식론이 필요한 분야가 Workflow에서 업무과정을 모델링하고 분석하여 업무의 효율화를 도모하는 곳이다. 복잡한 업무과정은 복잡한 조직도 상에서 계층적이며 모듈라한 업무의 모델링이 필요하고 업무과정에서 data consistency를 보장해야한다.[1]

이러한 필요성을 충족시키기 위한 프레임웍으로써 DEVS 모델을 관계대수형 모델로 변환하고 변환된 모델을 시뮬레이션하는 환경을 제안한다.

2. DEVS형식론의 관계대수적 변환

2.1 DEVS형식론에대한 간단한 소개

DEVS형식론은 계층적 모듈라한 형태로 모델링하는 방법을 제시한다. 복잡한 이산사건 시스템은 계층적으로 분리해서 표시할 수 있는 데 더 이상 나눌 수 없는 가장 작은 개체를 DEVS형식론에서는 atomic 모델이라고 한다. atomic 모델의 수학적 정의 다음과 같

다.

$AM = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$
X : 입력사건집합
S : 상태집합
Y : 출력사건집합
δ_{int} : $S \rightarrow S$: 내부상태전이함수
δ_{ext} : $Q \times X \rightarrow S$: 외부상태전이함수
λ : $S \rightarrow Y$: 출력함수
ta : $S \rightarrow Real$: 시간전진함수
$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$
: total state of AM(e: elapsed time)

DEVS의 coupled모델은 계층적인 모델 구조를 기술한다. 이것은 새로운 모델을 구성하기 위해서 컴포넌트 모델을 coupling하여 더 작은 모델에서 더 크고 복잡한 모델을 만들 수 있도록 한다. Coupled모델의 수학적인 정의는 다음과 같다.

$CM = \langle X, Y, \{M_i\}, EIC, EOC, IC, SELECT \rangle$
X : 입력사건집합
Y : 출력사건집합
$\{M_i\}$: DEVS컴포넌트집합
$EIC \subseteq X \times \bigcup_i X_i$: 외부입력관계
$EOC \subseteq \bigcup_i Y_i \times Y$: 외부출력관계
$IC \subseteq \bigcup_i Y_i \times \bigcup_i X_i$: 내부입출력관계
$SELECT$: $2^{\{M_i\}} - \emptyset \rightarrow \{M_i\}$

DEVS 형식론에 대한 자세한 논의와 모델링은 [2]에서 찾을수 있다.

2.3 관계형 대수

관계형 대수에는 6가지 기본적인 연산자가 있으며 이 6가지 연산자를 이용하면 관계대수적모델을 기술할 수 있다.

(1) Rename ρ : 관계 대수로 표현된 식 E가 있을 때 $\rho_x(E)$는 E의 결과를 x로 명명한다.

(2) Select σ : $\sigma_F(R) = \{t | F(t) \vee t \in R\}$

(3) Projection Π : $\Pi_X(R) = \{t[X] | t \in R\}$

(4) Union $\cup$: $R \cup S = \{t | t \in R \vee t \in S\}$

(5) Difference - : $R - S = \{t | t \in R \wedge t \notin S\}$

(6) Cartisian Product $\times$:

$R \times S = \{\langle r, s \rangle | r \in R \wedge s \in S\}$

(7) Natural Join $\triangleright\triangleleft$:

$R \triangleright\triangleleft S = \Pi_{R \cup S}($

$\sigma_{R.A_1 = S.A_1 \wedge R.A_2 = S.A_2 \wedge \cdots \wedge R.A_n = S.A_n}(R \times S))$

where $R' \cap S' = \{A_1, A_2, \cdots, A_n\}$

2.4 DEVS의 관계대수형모델 변환

계층적 모듈러한 DEVS모델은 tree 구조로 표현될 수 있다. 그러한 트리 구조는 root 노드를 필요로 하기 때문에 root노드를 나타낼 수 있으며 구조정보를 공유하기 위해서 Model테이블을 도입한다. 이것은 DEVSIF을 이용한 모델의 서술에서 interface부분에 해당하는 것이다.

다음의 Model테이블의 정의에서 X_{DEVS}와 Y_{DEVS}는 atomic모델과 coupled모델의 정의에 포함된 입력 사건과 출력 사건이다.

$Model' = \{id, parent\_id, model\_name\}$
$X' = \{id, ev\_name, type\_id, val\}$
$Y' = \{id, ev\_name, type\_id, val\}$
$X_{DEVS} = \Pi_{(ev\_name, type\_id)}(Model \triangleright\triangleleft X)$
$Y_{DEVS} = \Pi_{(ev\_name, type\_id)}(Model \triangleright\triangleleft Y)$

문자에서 '은 관계대수에서 속성 집합을 나타낸다.

모델 테이블을 도입하여 원래의 DEVS형식론에서 다루는 atomic모델과 coupled모델을 관계대수적으로 표시한다. atomic 모델은 7개의 tuple을 가지며 그 중에 X,Y는 Model

테이블에 의해서 정의 된다. 남은 S와 4가지 특성 함수들은 다음과 같이 정의 된다.

$$
\begin{aligned}
S' &= \{id, name, type\_id, val\} \\
Init\_cond' &= \{id, func\} \\
\delta'_{int} &= \{id, func, act\_func\} \\
\delta'_{ext} &= \{id, func, iport\_id, act\_func\} \\
\lambda' &= \{id, func, oport\_id, act\_func\} \\
ta' &= \{id, func, type, taval, ta\_id\} \\
ta'.type &\in \{INFINITY, SIGMA, RAND\} \\
func' &= \{id, func, op, e1, e2\} \\
func'.op &\in \{STATEID, PLUS, \cdots\}
\end{aligned}
$$

Atomic 모델의 4가지 특성 함수를 테이블 형태로 변환하는 것은 매우 복잡한 것이다. 4가지 특성 함수를 관계대수 모델로 변환하기 위하여 함수 번역기 func_I를 도입하며 이 func_I에 관한 것은 다음 장에 설명한다. Coupled 모델은 단지 모델의 구조에 관한 정보와 tie-breaking rule을 가지고 있는데 Model테이블에서 정의되어지는 입력 사건과 출력 사건을 가진다. 관계대수를 이용한 coupled모델의 정의는 다음과 같다.

$$
\begin{aligned}
M' &= \{id, Cid, Mid, Cname\} \\
EIC' &= \{id, Mid, Miportid, Cid, Ciportid\} \\
EOC' &= \{id, Cid, Coportid, Mid, Moportid\} \\
IC' &= \{id, src\_Cid, src\_Coportid, dst\_Cid, \\
&\quad dst\_Cid, dst\_Ciportid\} \\
SELECT' &= \{id, Cid, priority\}
\end{aligned}
$$

2.5 함수 번역기 : func_I

함수 번역기는 DEVSIF과 동일한 표현력을 가진다. 그리고 func_I는 함수 호출 방법으로 atomic 모델의 4가지 특성함수에서 수행된다. func_I는 다음과 같이 정의 된다.

proc func_I(*cstate,cfunc*:integer)
begin
  val : **integer**;
  $(op, e1, e2) := \Pi_{(T.op, T.e1, T.e2)}($
    $\sigma_{T.func = cfunc}(\rho_T(func)))$
  if (*op* = PLUS) **then**
    val := func_I (*cstate,e1*)
      + func_I (*cstate,e2*);
  else if (*op* = STATEID) **then**
    val := $\Pi_{val}(\sigma_{id = cstate \wedge name = e1}(S'))$
  else
  ...
  endif
  return *val*;
end

다음의 4가지 특성함수 $\delta_{int}, \delta_{ext}, \lambda, ta$는 func_I에 의해서 정의 된다.

proc $\delta_{int}(id,cstate)$
begin
  action_id := $\Pi_{T.action\_id}($
  $\sigma_{\text{func\_I}(cstate, T.func) \wedge T.id = id}(\rho_T(\delta'_{int})))$;
  func_I(*cstate,action_id*);
end
proc $\delta_{ext}(id,cstate,e,portid)$
begin
  action_id := $\Pi_{T.action\_id}($
  $\sigma_{T.iport\_id = portid \wedge \text{func\_I}(cstate, T.func) \wedge T.id = id}(\rho_T$
  $(\delta'_{ext}))$;
  func_I(*cstate,action_id*);
end
proc $\lambda(id,cstate)$
begin
foreach $(port\_id, action\_id) \in \Pi_{($
  $T.port\_id, T.action\_id)}(\sigma_{T.id = id \wedge}$
  $\text{func\_I}(cstate, T.func)(\rho_T(\lambda')))$
  $Y \leftarrow \sigma_{T.id = id \wedge T.ev\_name = port\_id}($
  $\sigma_{T.val = \text{func\_I}(cstate, action\_id)}(\rho_T(Y')))$;
endforeach
end
proc $ta(id,cstate)$
begin
  (*tatype,taval,ta_id*) :=
  $\Pi_{T.tatype, T.tavle, T.ta\_id}($
  $\sigma_{T.id = id \wedge \text{func\_I}(cstate, T.func)}(\rho_T(ta')))$;
  if (*tatype* = INFINITY) **then**
    val := INFINITY;
  else if (*tatype* = SIGMA) **then**
  val := $\Pi_{T.val}(\sigma_{(T.id = id \wedge T.name = cstate)}($
    $\rho_T(S')))$;
  else if (*tatype* = RAND) **then**
    val := random();
  endif
  return *val*;
end

3 관계형 모델의 시뮬레이션

변환된 관계대수형 모델은 DEVSIF라는 모델링 언어를 이용해 기술 될수 있다. 이것을 그림으로 표시하면 그림 1과 같다.

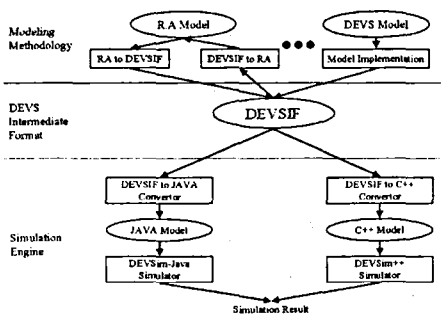

그림 1 전체 시뮬레이션 과정도

3.1 DEVSIF (DEVS Intermediate Format)

DEVSIF은 전체적인 모델을 기술하기 위해서 3가지 부분(모델, atomic 모델,coupled 모델)으로 구성된다. DEVSIF은 DEVS형식론에의해 나타낼 수 있는 모델링 정보를 보존하며 객체 지향적인 특성을 지원하고 있다.

1. 모델은 입력 사건집합과 출력 사건집합을 가지며 서술방법은 다음과 같다.

INTERFACE model_name
  [:parent_model_name]
  input_def
  output_def
END mode_name;

2. Atomic 모델은 상태변수, 초기조건과 4가지 특성함수(δ_{int}, δ_{ext}, λ, ta)를 가진다

ATOMIC MODEL atomic_name
  [: parent_atomic_name]
  state_def
  initial_cond
  int_trans_def
  ext_trans_def
  output_func_def
  time_adv_def
END atomic_name;

3. Coupled 모델은 자식정보, coupling정보와 tie-breaking rule을 가진다.

COUPLED MODEL coupled_name
  [: parent_coupled_name]
  child_def
  coupling_def_set
  select
END coupled_name;

4. 관계대수적 DEVS모델링 예

지금 까지 제시된 프레임웍을 이용하여 Buffer Processor 예를 보인다. Buffer Processor는 이산사건 시스템을 기술하는 가장 기본적인 모델이다. Buffer Processor모델을 그림으로 표현하면 그림 2와 같다.

Generator부분은 임의로 사건을 발생 시키는

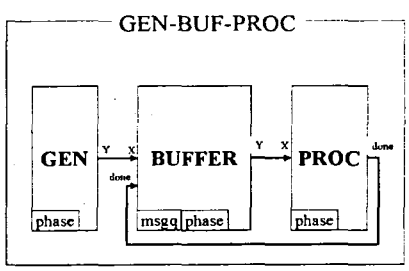

그림 2 Buffer Proc 모델

부분이고 Buffer는 queue이다. Processor는 queue에 입력을 요청하고 그것을 일정 시간에 걸쳐서 처리하는 모델이다.

Buffer가 3가지 모델중에서 가장 복잡한 모델인데 버퍼를 관계대수로 모델링하면 그림 3과 같다. 그림 3 의 관계대수 모델을 DEVSIF 형태로 변환하면 그림4와 같다. 그림 5는 전체 모델의 DEVSIF 예를 보여주고 그림 6는 그림 4와 마찬가지로 전체 모델의

model name

Table Name		Attribute Name
IPORT	name	
	done	
	X	
OPORT	name	
	Y	
STATE	name	
	phase	
	msgq	
	Y0	
Init Condition	0	

BUFFER

	Expr	Action
δint	3	6

	Expr	Port	Action
δext	9	done	17
	25	X	28
	33	X	36

	Expr	Port	Action
λ	39	Y	42

	Expr	type	val
ta	43	INFINITY	
	46	INTEGER	1

그림 3 Buffer 모델 table

```
interface BUFFER
        inputs : { done in integer, X in integer }
        outputs : { Y in integer }
end BUFFER;

atomic model BUFFER
        state variables :
                phase in { WAIT, SEND };
                msgq in seq of integer;
                Y0 in integer;
        initial condition:
                phase := WAIT;
        internal transition:
                (phase=SEND) => { phase := WAIT; };
        external transition:
                ((phase=WAIT) && (msgq.length > 0)) * done => {
                        phase := SEND;
                        Y0 := msgq.head;
                };
                (phase=SEND) * X => {
                        msgq.append X;
                        continue;
                };
                (phase=WAIT) * X => {
                        msgq.append X;
                };
        output function :
                (phase = SEND) => { Y:=Y0; };
        time advance:
                (phase = WAIT) => ta:=infinity;
                (phase = SEND) => ta:=1;
end BUFFER;
```

그림 4 Buffer DEVSIF예

```
interface GBP
        inputs :
        outputs :
end GBP;

coupled model GBP
        components : {
                G in GENERATOR,
                B in BUFFER,
                P in PROCESSOR
        }
        internal coupling: {
                G.Y -> B.X,
                B.Y -> P.X,
                P.done -> B.done
        }
end GBP;
```

그림 5 BUFFER PROC Coupled모델

model name

Table Name		Attribute Name			
		Name		Type	
GBP	Child	G		GENERATOR	
		B		BUFFER	
		P		PROCESSOR	
		Src Model	Src Oport	Dst Model	Dst Iport
	Internal Coupling	G	Y	B	X
		B	Y	P	X
		P	done	B	done

그림 6 GBP 전체 모델의 테이블 관계대수형 표현이다.

5 결론

이 논문에서는 DEVS 형식론을 기반으로 하여 모델을 관계 대수 모델로 변환하고 변환된 모델을 이용해 시뮬레이션 하는 방법론을 제시하였다. DEVS형식론을 이용하여 모델링 할 수 있는 DEVSIF이라는 모델링 언어를 정의 하였으며 DEVS to RA라는 번역기를 이용하여 DEVSIF로 정의 된 모델을 관계대수형 모델로 변환하여 관계형 데이터베이스에 DEVS형식론으로 정의된 모델의 의미의 손상이 전혀 없이 보존하는 것이 가능 하도록 하였다. 그리고 RA to DEVSIF을 이용하여 DEVSim-Java에서 관계 대수형 모델의 시뮬레이션을 지원하였다.

6. 참고문헌

[1] Andrzej Cichocki, Abdelsalam (Sumi) Helal, Marek Rusinkiewicz, and Darrell Woelk. *Workflow and Process Automation :Concepts and Technology* , Norwell, MA, Kluwer Academic Publishers, 1998

[2] B.P Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Orlando, FL, Academic PRess, 1984

[3] G.P Hong, T.G Kim "A Framework for Verifying Discrete Event Models Within a DEVS-Based System Development Methodology", *TRANSACTIONS of The Society for Computer Simulation*, Vol 13, no.1, pp.19-34, 1996

[4] Tag Gon Kim, *DEVSim++ User's Manual*, SMS Lab KAIST, 1994, (http://smsl.kaist.ac.kr).