Modelling Discrete-Event Systems Using DEVS

Assignment 1

**News Integration and Filtering Tool (NIF tool)**

Peiwen Chen

6373872

University of Ottawa

**Part I**

Almost all of us have several social networks accounts, such as facebook, twitter, blog. In China, we have douban, weibo, renren. And sometimes we would like to read all the news from these accounts at the same time , not to log in to these accounts one by one. So we need a platform to integrate information and push for us. This is the motive for the tool.

This focuses on integrating and filtering news from the social networks or blogs. With this tool, we could access to all the social networks account supported by it, and read all the news at the same time in one platform.

It composes of four submodels: Account, Write, Datageneration, Dataprocessing. And Dataprocessing is further decomposed into three submodels: Filter, Write, Display. (Figure 1)

The structure is as follows.



Figure 1. The structure of news integration and filtering(NIF) tool.

**Part II**

**Formal Specifications**

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

**Account:**



Account is to set the number of account, here, we suppose 4 at most ( accountNO<=4) for convenience.

“Account” has three inputs: account\_setting, plus\_in, minus\_in and one output: accountNO\_out.

When account\_setting has received “1”, it means we could start to set the number of accounts by the other two inputs: plus\_in and minus\_in. Plus\_in adds 1 to the number of the account number while minus\_in substracts 1 from the account number. At last, when “1” received again at account\_setting, it means the setting process ends. And we output the account number through accountNO\_out.

Account:

state variables:

phase={active, passive};

phase={active, passive};

Time receiving\_time; //the time duration

accountNO=0;// the number of accounts

bool processing; //the tag of processing

S = {phase, accountNO, processing}

X = {account\_setting, plus\_in, minus\_in}

Y = {accountNO\_out}

δext (s, e, x)

{

if ( x is from account\_setting)

{

case (phase)

{

passive:

{ phase=active; ta(phase)=receiving\_time; processing=true;}

active:

{ phase=passive; ta(phase)=INFINITY; processing=false;}

}

if ( x is from plus\_in==1)

{ accountNO++; phase=active; ta(phase)=receiving\_time;}

if ( x is from minus\_in==1)

{accountNO--; phase=acive; ta(phase)=receiving\_time;}

}

δint (s)

{

if (processing)

{phase=active; ta (phase)=receiving\_time; }

else

{ accountNO=0; passivate();}

}

λ(s)

{

if (!processing)

sendOutput(accountNO\_out,accountNO); //after all input read in, output the accountNO.

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Refresh:**



Refresh is to set the time period for data generation , here, we initialize the time period integer as 5. “Refresh” has three inputs: refresh\_setting, plus\_in, minus\_in and one output: refresh\_out.

When refresh\_setting has received “1”, it means we could start to set the number of time period by the other two inputs: plus\_in and minus\_in. Plus\_in adds 1 to the number while minus\_in substracts 1 from the number. At last, when “1” received again at refresh\_setting, it means the setting process ends. And we output the time period number LT through refresh\_out.

Refresh:

phase={active, passive};

Time receiving\_time; //the time duration

refreshtime=5;// the time period integer, will be transferred into Time class in the next atomic model

bool processing; //the tag of processing

S = {phase, refreshtime, processing}

X = {refresh\_setting, plus\_in, minus\_in}

Y = {refresh\_out}

δext (s, e, x)

{

if ( x is from refresh\_setting)

{

case (phase)

{

passive:

{ phase=active; ta(phase)=receiving\_time; processing=true;}

active:

{ phase=passive; ta(phase)=INFINITY; processing=false;}

}

if ( x is from plus\_in==1)

{ refreshtime++; phase=active; ta(phase)=receiving\_time;}

if ( x is from minus\_in==1)

{refreshtime--; phase=active; ta(phase)=receiving\_time;}

}

δint (s)

{

if (processing)

{phase=active; ta(phase)=receiving\_time; }

else

{ refreshtime=5; passivate();}

}

λ(s)

{

if (!processing)

sendOutput(refresh\_out,refreshtime); //after all input read in, output the refreshtime.

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Data Generation:**



Data Generation is to generate data for all the accounts (at most 4). It has two inputs: accountNO\_in and LT\_in and one output: data\_out. accountNO\_in will set the number of accounts to generate data and LT\_in will set the time period between two times of generating data. And data\_out will generate data through four ports: data1\_out, data2\_out, data3\_out, data4\_out. If the accountNO is 4, all of the four ports will generate data and if accountNO is only 1, only the data1\_out will generate data. And all of the data generated will be sent out at the same time.

Data Generation:

state variables:

phase={active,passive};

N4=4;N3=3;N2=2;N1=1; //output values for all accounts

Time receiving\_time;

Time periodtime;

bool sending;

int accountNO;

S = {phase, sending, N1, N2, N3, N4,LT}

X = {accountNO\_in, LT\_in}

Y = {data1\_out,data2\_out,data3\_out,data4\_out}

δext(s, x, e):

{

if ( x is from LT\_in)

{ sending=true;

LT=LT\_in.value();

periodtime=Time(0,0,LT,0);

phase=active; ta(phase)=receiving\_time;

}

if (x is from accountNO\_in and sending==true)

{

accountNO=accountNO\_in.value();

sending=false;

phase=active; ta(phase)=receiving\_time;

}

}

δint(s)

{

if (sending)

{ phase=active; ta(phase)=periodtime;}

else

{ accountNO=0; passivate();}

}

λ(s):

{

if (!sending)

{

case (accountNO)

4: //four account ports have output data

{sendOutput(data\_out,N4);

sendOutput(data\_out,N3);

sendOutput(data\_out,N2);

sendOutput(data\_out,N1);}

3:

{sendOutput(data\_out,N3);

sendOutput(data\_out,N2);

sendOutput(data\_out,N1);}

2:

{ sendOutput(data\_out,N2);

sendOutput(data\_out,N1);}

1: //only one account port has output data

{sendOutput(data\_out,N1);}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Filter:**



Filter is to filter data input from “data generation” by setting choice\_in. If we have some data read in from “data generation” through four ports: bit1\_in, bit2\_in, bit3\_in, bit4\_in, and choice\_in will decide which of the account can be output through the port “bit\_out”. Since account number is limited within 4, so choice\_in is less than 4, too.

For instance, if choice\_in is read in 3, then only bit3\_in port will be read in and then data will be output through bit\_out.

Filter:

<S, X, Y, δint, δext, λ, ta>

state variables:

phase={active,passive};

bit1=bit2=bit3=bit4=0; //initialize the input bits to 0

Time receiving\_time;

bool sending1, sending2, sending3, sending4;

bool processing;

S = {phase, bit1,bit2,bit3,bit4, processing}

X={bit1\_in,bit2\_in,bit3\_in,bit4\_in, choice\_in}

Y={bit\_out}

δext(s, e, x)

{

if (x is from bit1\_in)

{ bit1=bit1\_in.value(); processing=true; phase=active; ta(phase)=receiving\_time;}

if (x is from bit2\_in)

{ bit2=bit2\_in.value(); processing=true; phase=active; ta(phase)=receiving\_time;}

if (x is from bit3\_in)

{ bit3=bit3\_in.value(); processing=true; phase=active; ta(phase)=receiving\_time;}

if (x is from bit4\_in)

{ bit4=bit4\_in.value(); processing=true; phase=active; ta(phase)=receiving\_time;}

if (x is from choice\_in )&& (phase==active)

{

if (choice\_in.value()==4) sending4=true;

if (choice\_in.value()==3) sending3=true;

if (choice\_in.value()==2) sending2=true;

if (choice\_in.value()==1) sending1=true;

phase=active; ta(phase)=receiving\_time;

}

}

δint (s)

{

if (processing)

{ phase=active; ta(phase)=preparationTime; }

else { sending1=sending2=sending3=sending4=false;

bit1=bit2=bit3=bit4=0; passivate();}

λ (s)

{

if (!processing)

{

if (sending4) sendoutput(bit\_out,bit4);

if (sending3) sendoutput(bit\_out,bit3);

if (sending2) sendoutput(bit\_out,bit2);

if (sending1) sendoutput(bit\_out,bit1);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Write:**



Write is to write some data and send to the corresponding account. It has one input “choice\_in” and one output “write\_out”. We choose one account through the value of “choice\_in” and then send data through the output of “write”. For instance, if choice\_in is read in 3, then we send 3 through “write\_out”, which means the data is from account 3.

Write:

<S, X, Y, δint, δext, λ, ta>

state variables:

phase={active,passive};

int choice;

int bit;

Time preparationtime;

S = {phase, bit,choice}

X={choice\_in}

Y={write\_out}

δext(s, e, x)

{

if ( x is from choice\_in )&& (phase==passive)

{

choice=choice\_in.value();

if (choice==4) bit=4;

if (choice==3) bit=3;

if (choice==2) bit=2;

if (choice==1) bit=1;

phase=active; ta(phase)=preparationtime;

}

}

δint (s)

{ bit=0; passivate();}

λ (s)

{ sendOutput(write\_out,bit);}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Display:**



Display is to show the data got from “filter” and “write”. “filter” and “write ” has one output port for each and then when data from this two ports is received, “display” sends the data received to the output port “display\_out”.

Display:

<S, X, Y, δint, δext, λ, ta>

state variables:

phase={active, passive};

int bit1\_display, bit2\_display;

Time receiving\_time;

bool get\_filter, get\_write;

S={phase,bit1\_display, bit2\_display, get\_filter, get\_write}

X={filter\_in,write\_in}

Y={display\_out}

δext(s, e, x)

{

if ( x is from filter\_in)

{ bit1\_display=filter\_in.value();

get\_filter=true;

phase=active; ta(phase)=receiving\_time;

}

if ( x is from write\_in)

{ bit2\_display=write\_in.value();

get\_write=true;

phase=active; ta(phase)=receiving\_time;

}

}

δint(s)

{

get\_filter=get\_write=false;

passivate();

}

λ(s)

{

if (filter\_in)

sendOutput(write\_out,filter\_in);

if (write\_in)

sendOutput(write\_out,write\_in);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Data processing:**



The formal specifications <X, Y, D, {Mi}, {Ii}, {Zij}, SELECT > for the coupled model Account, Refresh, Data generation, Data processing and NIF tool is defined as follows:

data processing:

X = {choice\_in, bit1\_in, bit2\_in, bit3\_in, bit4\_in};

Y = {display\_out};

D = {filter, write, display};

I(filter) = {self};

I(write) = {self};

I(display)={self};

Z(filter) = self; Z(write) = self; Z(display)=self;

SELECT: ({filter, write, display}) = filter;

**NIF tool:**

X = {account\_setting,refresh\_setting, plus\_in, minus\_in, choice\_in};

Y = {display\_out};

D = {account, refresh, datageneration, dataprocessing};

I(account) = {datageneration, self};

I(refresh) = {datageneration, self};

I(datageneration) = {account, refresh, dataprocessing};

I(dataprocessing) = {datageneration, self};

Z(account) = datageneration; Z(account) = self;

Z(refresh) = datageneration; Z(refresh) = self;

Z(datageneration) = account; Z(datageneration) = refresh; Z(datageneration) = dataprocessing;

Z(dataprocessing) = datageneration; Z(dataprocessing)=self;

SELECT: ({ account, refresh, datageneration, data processing}) = account;

({refresh, datageneration, data processing}) = refresh;

({datageneration, data processing}) = datageneration;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Part III**

In order to verify the atomic models and coupled models, test cases are created to test these models.

**Test Cases and Execution Analysis**

**Atomic Model** **Account:**

The Account number is supposed to a positive integer, and I have limited the number less than 4 for convenience, which means we can only have four accounts at most at the same time. The input of Account “plus\_in” and “minus\_in” will add or subtract the account number.

If “account\_setting” receives a “1”, it begins setting the account number. Then we could choose plus or minus the number by inputting “1”. This process will hold until another “account\_setting” receives a “1”, it means the setting ends.

In the test cases, we will get the accurate account number no matter what the time duration between two consecutive inputs, since a state tage “processing” is used. So until the other “account\_setting” is received corresponding to the first one, the process ends.

This is the account.ev:

00:00:10:00 account\_setting 1

00:00:15:00 plus\_in 1

00:00:16:00 plus\_in 1

00:00:50:00 account\_setting 1

00:01:00:00 account\_setting 1

00:01:05:00 plus\_in 1

00:01:07:00 plus\_in 1

00:01:08:00 plus\_in 1

00:01:09:00 plus\_in 1

00:01:10:00 minus\_in 1

00:01:12:00 plus\_in 1

00:01:15:00 account\_setting 1

and the .out file is:

00:00:50:000 accountno\_out 2

00:01:15:000 accountno\_out 4

It shows even the time duration between (00:00:16:00) and (00:00:50:00) is greater than the preparationTime(0,0,10,0), the process does not terminates until the second “account\_setting” is received.

**Atomic Model** **Refresh:**

Refresh has almost the same structures with “Account”.

In the test cases, we will get the accurate number no matter what the time duration between two consecutive inputs, since a state tage “processing” is used. So until the other “refresh\_setting” is received corresponding to the first one, the process ends.

the .ev file:

00:00:10:00 refresh\_setting 1

00:00:11:00 plus\_in 1

00:00:40:00 refresh\_setting 1

00:00:45:00 refresh\_setting 1

00:00:50:00 minus\_in 1

00:00:55:00 plus\_in 1

00:01:05:00 plus\_in 1

00:01:10:00 plus\_in 1

00:01:20:00 plus\_in 1

00:01:30:00 refresh\_setting 1

the .out file:

00:00:40:000 refresh\_out 6

00:01:30:000 refresh\_out 8

**Atomic Model** **Datageneration:**

Datageneration will send data through four output ports when the account number is received. For instance, if accountNO is received as 4, then data will be sent out at all the four output ports, which means all of four accounts have generated data; but if accountNo is 1, then the data will be sent only through port 1, which means only the account 1 has generated data.

The data will be sent out at the time when accountNO\_in is received. So as shown below in the test cases, “ 00:00:10:00 accountNO\_in 4” leads to output 4 data through 4 account ports at (00:00:10:00), as shown in the .out file:

00:00:10:000 data4\_out 4

00:00:10:000 data3\_out 3

00:00:10:000 data2\_out 2

00:00:10:000 data1\_out 1

the .ev file:

00:00:00:00 LT\_in 10

00:00:10:00 accountNO\_in 4

00:00:20:00 LT\_in 20

00:00:30:00 accountNO\_in 3

00:00:40:00 LT\_in 5

00:00:50:00 accountNO\_in 2

00:01:10:00 LT\_in 10

00:01:20:00 accountNO\_in 4

00:01:30:00 LT\_in 30

00:01:35:00 accountNO\_in 3

00:01:40:00 LT\_in 10

00:01:50:00 accountNO\_in 2

the .out file:

00:00:10:000 data4\_out 4

00:00:10:000 data3\_out 3

00:00:10:000 data2\_out 2

00:00:10:000 data1\_out 1

00:00:30:000 data3\_out 3

00:00:30:000 data2\_out 2

00:00:30:000 data1\_out 1

00:00:50:000 data2\_out 2

00:00:50:000 data1\_out 1

00:01:20:000 data4\_out 4

00:01:20:000 data3\_out 3

00:01:20:000 data2\_out 2

00:01:20:000 data1\_out 1

00:01:35:000 data3\_out 3

00:01:35:000 data2\_out 2

00:01:35:000 data1\_out 1

00:01:50:000 data2\_out 2

00:01:50:000 data1\_out 1

**Atomic Model** **Filter:**

Filter has four input ports corresponding to four accounts and if the data sent in is 0, it means no data sent through this account. By choosing the account through the value got from the port “choice\_in”(only one account could be chosen at a time), some data from other accounts could be filtered out and not sent out. This is how filter works.

All the data will be sent out at the same time when “choice\_in” receives a value to choose. As shown in test cases, when “00:00:10:00 choice\_in 3” is received, then filter will send out the data chosen and filter out the other account data. Since we choose account 3, so only data at account 3 will be sent out , it is “00:00:10:000 bit\_out 3”.

When we choose an account, but no data is received in this account, then “0” will be sent out. When we choose account 2, but only account 1 receives data “4”, then “0” will be sent out: “00:00:30:000 bit\_out 0”.

the .ev file:

00:00:00:00 bit4\_in 4

00:00:00:00 bit3\_in 3

00:00:00:00 bit2\_in 2

00:00:00:00 bit1\_in 1

00:00:10:00 choice\_in 3

00:00:20:00 bit1\_in 4

00:00:30:00 choice\_in 2

00:00:40:00 bit2\_in 4

00:00:50:00 choice\_in 1

the .out file:

00:00:10:000 bit\_out 3

00:00:30:000 bit\_out 0

00:00:50:000 bit\_out 0

**Atomic Model** **Write:**

Write has only one input port to write and send data. And if the time between two consecutive inputs is less than the preparationTime, the new input will be ignored. Because the state ”active” will be held for “preparationTime” and other input can not be received.

In the test cases, “ 00:01:05:00 choice\_in 3 “ account 3 is chosen to write data after the former input “00:01:00:00 choice\_in 1 “, however, this choice\_in 3 will be discarded.

As can been seen, all the data is sent out after a “preparationTime” (0,0,10,0).

the .ev file:

00:00:00:00 choice\_in 3

00:00:20:00 choice\_in 4

00:00:40:00 choice\_in 2

00:01:00:00 choice\_in 1

00:01:05:00 choice\_in 3

00:01:15:00 choice\_in 4

the .out file:

00:00:10:000 write\_out 3

00:00:30:000 write\_out 4

00:00:50:000 write\_out 2

00:01:10:000 write\_out 1

00:01:25:000 write\_out 4

**Atomic Model** **Display:**

Display has two input data from “filter\_in” and “write\_in”, and it will send out data received out, even the data is received at the same time.

In the test cases, “00:00:10:00 filter\_in 2 00:00:10:00 write\_in 2” two ports input data at the same time and display will send out the data at the same time “00:00:10:000 display\_out 2 00:00:10:000 display\_out 2”. And display will send out the data once received.

the .ev file:

00:00:00:00 filter\_in 4

00:00:05:00 write\_in 3

00:00:10:00 filter\_in 2

00:00:10:00 write\_in 2

00:00:30:00 filter\_in 1

00:00:40:00 write\_in 2

00:00:45:00 write\_in 3

00:00:50:00 filter\_in 4

00:01:20:00 write\_in 3

00:01:30:00 filter\_in 2

00:01:35:00 write\_in 1

the .out file:

00:00:00:000 display\_out 4

00:00:05:000 display\_out 3

00:00:10:000 display\_out 2

00:00:10:000 display\_out 2

00:00:30:000 display\_out 1

00:00:40:000 display\_out 2

00:00:45:000 display\_out 3

00:00:50:000 display\_out 4

00:01:20:000 display\_out 3

00:01:30:000 display\_out 2

00:01:35:000 display\_out 1

**Coupled Model Dataprocessing**

Coupled model dataprocessing consists of three submodels: filter, write and display. “choice\_in” chooses the data to send out for filter while it writes data to the corresponding account and sent it out.

In the test cases, when we choice\_in receives 3, it means only account 3 can be sent out through this model and other account data will be filtered. While it also writes data for account 3 and to send it out after a preparetionTime. So there would be two “3” sent at the output, meaning account 3 has data from filter and write.

the .ev file:

00:00:00:00 bit4\_in 4

00:00:00:00 bit3\_in 3

00:00:00:00 bit2\_in 2

00:00:00:00 bit1\_in 1

00:00:10:00 choice\_in 3

the .out file:

00:00:10:000 display\_out 3

00:00:20:000 display\_out 3

**Coupled Model NIF Tool**

Coupled model NIF Tool will set number of accounts and the refresh, then choose the account to display. In the test cases, choice\_in receives “3”, it means account 3 is chosen. With four accounts input into dataprocessing, only account 3 passes and data is written to send out at account 3. So two account 3 data is received after the whole coupled model, one is from filter, the other one is from write.

the .ev file:

00:00:00:00 account\_setting 1

00:00:00:00 refresh\_setting 1

00:00:05:00 plus\_in 1

00:00:10:00 plus\_in 1

00:00:11:00 plus\_in 1

00:00:12:00 refresh\_setting 1

00:00:12:00 plus\_in 1

00:00:15:00 account\_setting 1

00:00:20:00 choice\_in 3

the .out file:

00:00:20:000 display\_out 3

00:00:30:000 display\_out 3