COM6116 Theory of Distributed System

Report

Hao Qian

hqian5@sheffield.ac.uk

Contents

1 CCS Model

1.1 Introduction

Basically, based on requirements of the assignment, we need to design and implement a communication system which simulates a chemistry reaction: the combustion of Methanol (CH_3OH) in Oxygen (O_2). Generally, when Oxygen is sufficient, products of this reaction are Water (H_2O) and Carbon Dioxide (CO_2), which means the reaction has reached a stable status. However, when Oxygen is not enough, it will generate some intermediate hydrocarbon molecules, including CH_3, CH_4, HCO, CH_3O and CH_2OH. The main task is to use CCS model to simulate this reaction when Oxygen is enough or insufficient, and then design an Erlang system based on CCS model.

1.2 Transform Chemistry Forms to CCS Model

CCS is calculus of communicating system which is mainly to represent and analyse the communication of simple processes. Meanwhile, chemistry reactions are interactions between different chemical substances, similar with communication between different processes. Therefore, chemistry reactions can be modelled into this form. First of all, chemical molecules can be represented by CCS processes. Then, atoms and ions which are interchanged between molecules can be transformed to CCS actions. More specific, sending out an atom or ion can be regarded as output a signal; vice versa, receiving an atom or ion can be seen as input a signal.

1.3 A Possible Solution

Personally speaking, in order to perfectly simulate this chemistry reaction, except hydrocarbon molecules mentioned before, some other intermediates should be created. Despite some of these substances may not exist in chemical fields, fortunately, CCS model has been applied to represent these molecules. Therefore, six chemistry reactions (communication processes) will be listed.

1. $CH_3OH + O_2 \rightarrow CH_3O + HO_2$
2. $CH_3O + HO_2 \rightarrow CH_3 + HO_3$
3. $CH_3 + O_2 \rightarrow H_3 + CO_2$
4. $HO_3 + H_3 \rightarrow H_2O_3 + H_2$
5. $H_2O_3 + H_2 \rightarrow H_2O_2 + H_2O$
6. $H_2O_2 + CH_3 \rightarrow H_2O + CH_3O$

With these processes, it is possible to simulate the combustion of Methanol whether Oxygen is sufficient or not. Based on these processes, it will fully burn when Oxygen is enough, generating CO_2 and H_2O, then reaching a steady state. However, for insufficient Oxygen, underburning will occur, which brings not only CO_2 and H_2O, but also some intermediates. Thus, all these molecules will be expressed in CCS model.

1.4 CCS Forms

As mentioned before, molecules are CCS processes, atoms and ions interchanges are CCS actions. Therefore, according to six reactions above, CCS model for this system is as follow.

$\mathrm{React} = (\mathrm{CH_3OH} \mid O_2)\backslash\{\mathrm{c,h,o,oh}\}$

$\mathrm{CH_3OH} = \overline{oh}.CH_3 + \bar{h}.CH_3\mathrm{O}$
$\mathrm{O_2} = \mathrm{c}.\mathrm{CO_2} + \mathrm{h}.\mathrm{HO_2} + \bar{o}.O$

$CH_3\mathrm{O} = \mathrm{h}.\mathrm{CH_3OH} + \bar{o}.CH_3$
$\mathrm{HO_2} = \bar{h}.O_2 + \mathrm{o}.\mathrm{HO_3}$

$\mathrm{CH_3} = \bar{c}.H_3 + \mathrm{o}.\mathrm{CH_3O} + \mathrm{oh}.\mathrm{CH_3OH}$
$\mathrm{HO_3} = \bar{o}.HO_2 + \overline{oh}.O_2 + \mathrm{h}.\mathrm{H_2O_3}$

$\mathrm{H_3} = \bar{h}.H_2$
$\mathrm{H_2O_3} = \bar{o}.H_2\mathrm{O_2} + \bar{h}.HO_3 + \overline{oh}.HO_2$
$\mathrm{H_2} = \mathrm{h}.\mathrm{H_3} + \mathrm{o}.\mathrm{H_2O}$
$\mathrm{H_2O_2} = \mathrm{o}.\mathrm{H_2O_3} + \bar{h}.HO_2 + \bar{o}.H_2\mathrm{O}$

$\mathrm{H_2O} = \mathrm{H_2O}$
$\mathrm{CO_2} = \mathrm{CO_2}$

$\mathrm{C} = \bar{c}.0$
$H = \bar{h}.0 + oh.H_2\mathrm{O} + \mathrm{h}.\mathrm{H_2}$
$\mathrm{O} = \bar{o}.0 + o.O_2$

2 Erlang System

2.1 System Design

Obviously, according to CCS model, relative functions should be defined. There are forteen functions altogether in this Erlang system. Basically, thirteen of them simulate molecules in this reaction respectively. The rest one is a main function called react(X, Y) which is responsible for starting the whole system. There are two parameters in this function, X is the number of Methanol (CH_3OH) and Y is the number of Oxygen (O_2). These two numbers will be typed in by the user, then the system will run relative reactions, finally display the number of each molecule after the process to the user. In Erlang system, a PID is used to direct messages between processes. Hence, in function react(X, Y), PIDs for all processes have been registered at the beginning, just like Figure 1 shows. By doing this, it is possible to use PIDs of relative processes in the system. At first, except CH_3OH and O_2, the number of other processes is zero. After the system starts, relative processes will send messages to those which should be generated, telling them to increase the number of themselves. So that, the system is able to simulate the combustion of Methanol(CH3OH) in Oxygen (O2).

```
react(X, Y) ->

    register(o, spawn(chemistry, oof, [0])),
    register(h2o, spawn(chemistry, h2of, [0])),
    register(h2o2, spawn(chemistry, h2o2f, [0, ch3, h2o, o])),
    register(h2o3, spawn(chemistry, h2o3f, [0, h2, h2o2])),
    register(h2, spawn(chemistry, h2f, [0, h2o3, h2o])),
    register(h3, spawn(chemistry, h3f, [0, ho3, h2])),
    register(co2, spawn(chemistry, co2f, [0])),
    register(ch3, spawn(chemistry, ch3f, [0, o2, h3, h2o2, ch3o])),
    register(ho3, spawn(chemistry, ho3f, [0, h3, h2o3])),
    register(ho2, spawn(chemistry, ho2f, [0, ch3o, ho3])),
    register(ch3o, spawn(chemistry, ch3of, [0, ho2, ch3])),
    register(o2, spawn(chemistry, o2f, [Y, ho2, ch3oh, ch3, co2])),
    register(ch3oh, spawn(chemistry, ch3ohf, [X, o2, ch3o])),

%stop all reactions after 3 seconds
    receive
        after 3000 ->
            o ! h2o ! h2o2 ! h2o3 ! h2 ! h3 ! co2 ! ch3 ! ho3 ! ch3o ! ho2 ! o2 ! ch3oh ! finished
    end.
```

Figure 1: The function react(X, Y)

As mentioned before, processes are used to simulate molecules and actions represent substance interchanges between molecules. In the Erlang system, every function is a process, and for each of them, it is able to send and receive messages. Based on CCS model, communication is important in this system, which means some processes will output messages to others, some will input messages from others. Therefore, it is necessary to consider the conditions of communication. In this system, a process is required to send a request to those it wants to communicate with. Only if receivers reply an agree message, the communication can be executed. This ensures that processes are only able to communicate with those who have been generated. Take the communication between process H_2O_3 and process H_2 as an example, at first, H_2O_3 sends a request message to process H_2, if there is no H_2, it will send a refuse message back to H_2O_3. If there are some H_2 existing, it will send an agree message back.

Then, following communication will be implemented. The whole process in Erlang is showed in Figure 2. In this system, all communication processes are exactly the same with this one.

```
h2f(N, h2o3, h2o) ->
    receive
        finished ->
            format("H2 number: ~w~n", [N]);

        add_h2 ->
            h2f(N + 1, h2o3, h2o);

        h2o3_request ->
            h2o3 ! ok_h2o3,
            h2f(N, h2o3, h2o);

        add_o ->
            h2o ! add_h2o,
            h2f(N -1, h2o3, h2o)
    end.

%Function for H2O3
h2o3f(0, h2, h2o2) ->
    receive
        finished ->
            format("H2O3 number: ~w~n", [0]);

        add_h2o3 ->
            h2o3f(1, h2, h2o2)
    end;

h2o3f(N, h2, h2o2) ->
    h2 ! h2o3_request,

    receive
        finished ->
            format("H2O3 number: ~w~n", [N]);

        add_h2o3 ->
            h2o3f(N + 1, h2, h2o2);

        no_h2o3 ->
            h2o3f(N, h2, h2o2);

        ok_h2o3 ->
            h2 ! add_o,
            h2o2 ! add_h2o2,
            h2o3f(N - 1, h2, h2o2)
```

Figure 2: A communication example

As Figure 1 showing, there is a stop criterion in function react(X, Y). After three seconds from the beginning, a message will be sent to other thirteen processes, informing them the whole process has ended. By doing this, plenty of time is given to every reaction, therefore, the number of every molecule will be correctly displayed.

2.2 Outcome of the System

Figure 3 shows the outcome of the system. As we can see, if there are 4 CH_3OH and 6 O_2, 2 CO_2 and $8H_2O$ will be generated. It means this system is able to simulate the reaction correctly. However, this outcome may does not exactly fit chemical requirements, which is also a drawback of this system.

```
Eshell V10.2  (abort with ^G)
1> c(chemistry).
{ok,chemistry}
2> chemistry:react(4,6).
H2O number: 8
CH3OH number: 0
O number: 4
O2 number: 0
HO2 number: 0
CH3O number: 0
HO3 number: 0
CH3 number: 0
CO2 number: 2
H3 number: 0
H2 number: 0
H2O3 number: 0
H2O2 number: 0
finished
```

Figure 3: The outcome of the system

2.3 Eunit Test

Besides the implementation of the system, an Eunit test has also be done. It tests all processes in the system whether they can correctly receive messages or not. However, due the complexity of the system, it is difficult to test the whole system. Despite I have tried my best to write a better Eunit test, it still has some problems. But through Erlang shell test, the system does have a stable function.