1. User requirements:

As a patient, I want to check and record my state so that I can get in touch with my doctors and know the situation of myself, especially a visualization showing the treatments and reactions of the patient.

As a medical staff, I want to provide proper and opportune real-time treatments to the patients so that patients can get the best care. Also, I want to track the state of each patient and their Medication situation with the responses so that I can analyze the problem and give the next step.

1. Literature Review

POP-PL is a Patient-Oriented Prescription Programming Language. It’s based around the idea that programs and humans have complementary strengths. POP-PL combined them to make for safer and more accurate performance of prescriptions. The human language syntax of POP-PL is similar to what we want. However, POP-PL mainly aims at giving a prescription to patient. It can’t run synchronously and it doesn’t give patients a way to check their treatment history or record their reactions to the does.

In the clinical trials, finding suitable patients and reduce errors are crucial. Cause in many clinical trials, only one of the new drug can reach the market. Sub-optimal patient cohort selection and recruiting techniques, paired with the inability to monitor patients effectively during trials are two main reasons. To reduce our cost, we need to make sure that the programming language can give a real-time and proper command.

Above all, we need a programming language, which has human language syntax and can run synchronously. Since our programming language is based on clinical trials, it’s supposed to record the treatment history of each patient. Using the recorded data from nurses, the system can generate a visualization of each patient. Using the data from treatment history, researchers can get the general reaction of a does and optimize their treatment next time.

High Level programming environment,

Reread HIPHOP, how parallel synchronous preserved in hip hop

Motivated use case, what actually we need -> Syntax; Code generate

Satisfy all requirements

synchronous

IEC-61499, “Model Driven Design using IEC-61499”

Chapter 2&4

I need a symbol to represent the parallel process.

***3. Syntax:***

(1) Does(action):

**giveDose** \_what drug \_how much\_ units/kg, client1

**increase** \_what drug \_by \_how much\_ units/kg, client1

**decrease** \_what drug \_by \_how much\_ units/kg, client1

**keep \_\_\_**

(2) Check(action):

**if** result **is** \_ to \_, **then** \_\_\_

**if** result **>** \_, **then** \_\_\_

**if** result **<** \_, **then** \_\_\_

**if** result is \_\_\_, **then** \_\_\_

**anxietyCheck**:

**checkAnxiety** \_\_\_, timerId

**pressureCheck**:

**repeat** (**checkPressure** \_\_, timerId)

(3) Synchronous:

**multi@**

**###multiClients** (client1, client2):

{(action1, client1), (action2, client2)}

**until@**: ‘abort’

(Design a synchronous function called ‘multi’ and use @ to tell our system that I want to run the following function in parallel until we got an abort signal from user. However, to make this synchronous function works in the trial, it needs a timer to set an interval for the action.)

**(4)** Loop:

A) **every** \_\_ unit time:

**repeat** action

B) **every** \_\_ unit time:

**repeat** action

**if** \_\_\_, then **stop**

(5) Record(action):

**receiveDoes** \_what\_drug\_  \_how\_much\_ units/kg at 9am:

anxiety \_\_, pressure\_\_, mood \_\_

(Some Digital-Mood need to be recorded.)

(6) Questionnaire(action):

**giveQuestionnaire** to client send the questionnaire to the client

**getResult** from client return the result from the client

(The action **giveQuestionnaire** will give a questionnaire to a client and the action **getResult** will get the result of questionnaire from client.)

(7) Get all the actions need to be done for a client:

**getActions** from client. return/print a dict

(It will return/print a dictionary containing all the actions still in the timer interval and the interval time left for client1.)

***4. User Cases***

Then we can try to deal with the user cases using the syntax:

1. Alesha would like to micro-dose client1 with α dose of β units/kg, each γ hours:

**every** γ hours:

**giveDose** to client α with β units/kg

2. Alesha would like to log the medication taken by client1 at Δ time and λ date:

**receiveDose** from client α with β units/kg at Δ time and λ date

3. Alesha would like to do a questionnaire for clinet1 every γ hours:

**every** γ hours:

**giveQuestionnaire** to client1

4. Nurse/Client would like to know when Alesha would like to do what action for client1:

**getActions** from client

5. Fred would like to micro-dose client1 with α dose of β units/kg every 24 hours, unless client1 become sad:

**every** period:

**giveDose** to client1 α with β units/kg

**giveQuestionnaire** to client1

**if getResult** from client1 **is** ‘sad’, **then** ***stop***

6. Fred would like to micro-dose client1 withαdose ofβunits/kg and give the questionnaires to client1 and all the family members of client1(called client 2 and client3) every period

**every** period:

**multi@**

**giveDose** to client1 α with β units/kg

**giveQuestionnaire** to client1 **and** client2 **and** client3

**until@**: ‘abort’

    ……, **then** ***stop***

------------------------------------------clinical trial----------------------------------------------

In the clinical trial, we may have a huge system with several doctors/nurses and many clients. Our system should give each action a time-stamp and a label from the medical staff, and if there occurs trying to do a repeat treatment on one client in a short period, the system should raise an alert.

And also, there should be an alert for the maximum units of drug per person per period.

Saying we have 30 clients, 3 doctors and 6 nurses in the Phase 1.

**Initial:**

**Client:** client1, client2, client3, client4, client5, client6, client7, client8, client9, client10, client11, client12, client13, client14, client15, client16, client17, client18, client19, client20, client21, client22, client23, client24, client25, client26, client27, client28, client29, client30

**Doctor:** D1, D2

**Nurse:** N1, N2, N3, N4, N5, N6

**Default Dose**: β units/kg

**Default period**: 3 days

For example, the doctors will take care of 15 clients and give treatments, and the nurses may take care of 5 clients doing the treatments from doctors and recording done. That is each treatment will be sent to the nurse that the client is allocated to. And each time a treatment is labeled as finished, a normal questionnaire will be sent to the client so that the doctors can get the feedback from clients.

At first, D1 and D2 want to give the default dose to all clients in the default period.

For example, if I’m N1, I will receive messages below:

*\*\*\*Logged as N1, be responsible for client1, client2, client3, client4, client5\*\*\**

*To do list:*

giveDose to client1 with β1 units/kg

giveDose to client2 with β2 units/kg

giveDose to client3 with β3 units/kg

giveDose to client4 with β4 units/kg

giveDose to client5 with β5 units/kg

And after N1 finish one dose, N1 will label the task as finished. After that, the client will automatically receive a questionnaire.

The system has a built-in timer for each client, and after a period, N1 will receive the same messages again.

Now, if D1 think client1 needs more dose and want to change the treatment to client1:

**giveDose** to client1 with γ units/kg

Our system will check the period between the last time client1 was treated and now. If the period is less than the default, our system will raise:

*\*\*\*Alert: The last dose for client1 is 5 minutes ago, do you want to continue?\*\*\**

If D1 chose to continue and the dose has reached or beyond the highest level of safe dose level of one person, our system will raise:

*\*\*\*Alert: You can not give more than \_\_\_ units dose to client1 in a period.\*\*\**

Then D1 may change the treatment as:

**after** 2 days:

**every** period:

**giveDose** to client1 with γ units/kg

Or:

**increase** dose of client1 to γ units/kg

Since the former treatment was give client1 β1 units/kg dose every period, this increase action will inherit the loop and give a message to D1:

*\*\*\*It will give client1 γ units/kg dose every period, do you want to continue?\*\*\**

If D1 want to use the parallel operation and give a stop statement:

multi@:

**every** period1:

**giveDose** to client1 with γ units/kg

**every** period2:

**giveDose** to all client except client1 with β units/kg

if the result of client is ‘bad’, **stop**

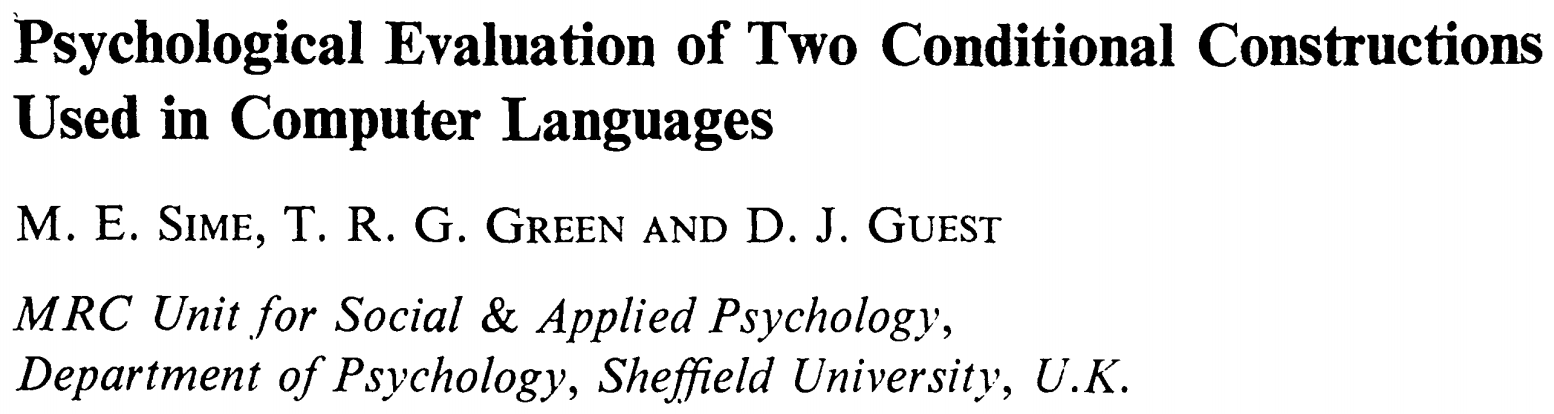
And if N1 haven’t finished the dose for client in maybe half an hour, our system will raise a reminder for N1:

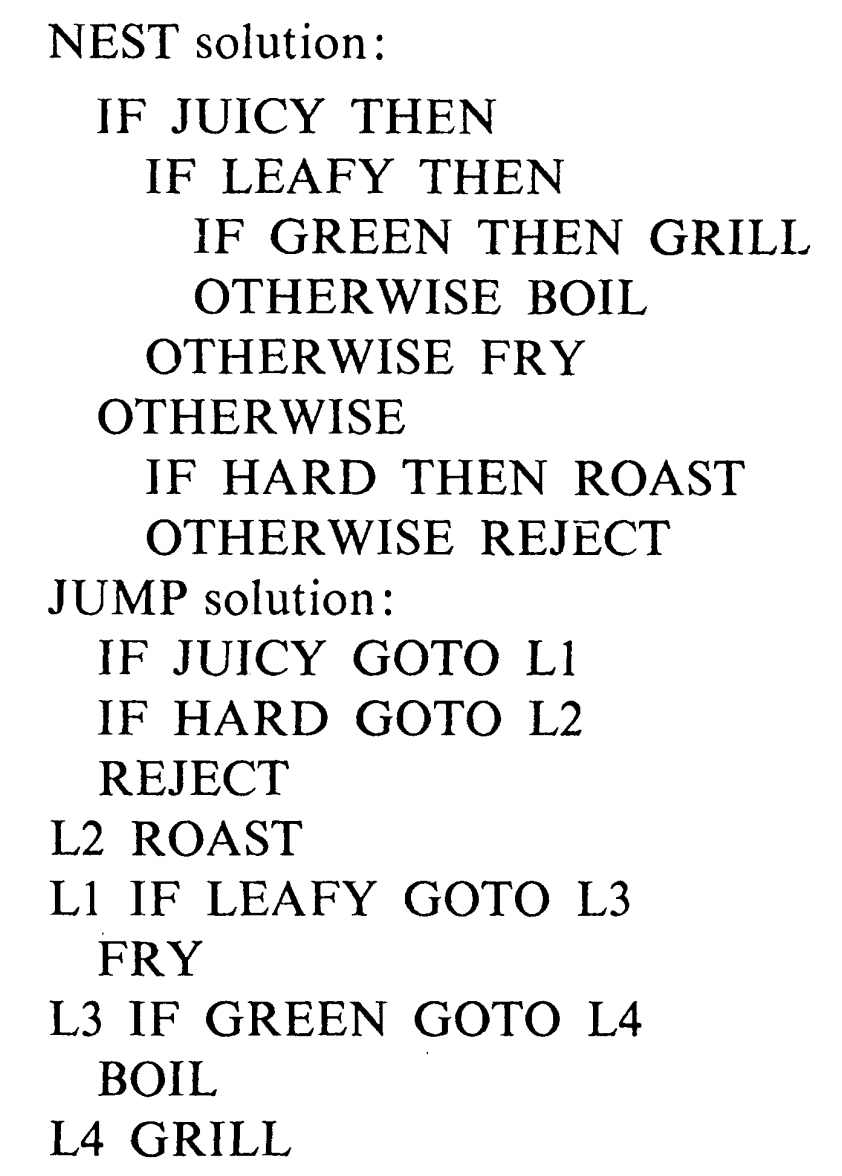
*\*\*\*Reminder: You haven’t finish the dose yet: giveDose to client1 with γ units/kg\*\*\**

And we may need to add an urgent alert for the client. If the client1 felt really bad in the clinical trial, he can click urgent button or something and then the N1 will receive an urgent message like:

*\*\*\*Urgent: client1 need help!\*\*\**

**Adding other relevant if-else statements.**





This thesis compared the nest language and jump language and it showed that more semantic errors were made by JUMP group than by NEST group.

**Function Block**

Basic function block: Execution Control Chart (ECC) graphical state-machine

Nowadays, most programming in the pharmaceutical industry focuses on transforming and analyzing clinical data. For example, **SAS** is a statistical software system. It uses data steps to read data into memory and proc steps to calculate descriptive or inferential statistics, generate summary reports, and create summary graphs and charts.