HO CHI MINH UNIVERSITY OF TECHNOLOGY

INTERNET OF THINGS APPLICATION DEVELOPMENT

Assignment 1:

CALCULATOR (STATE MACHINE)

Author:
Nguyen Huu Anh Hieu 1652741

Teacher:
Le Trong Nhan

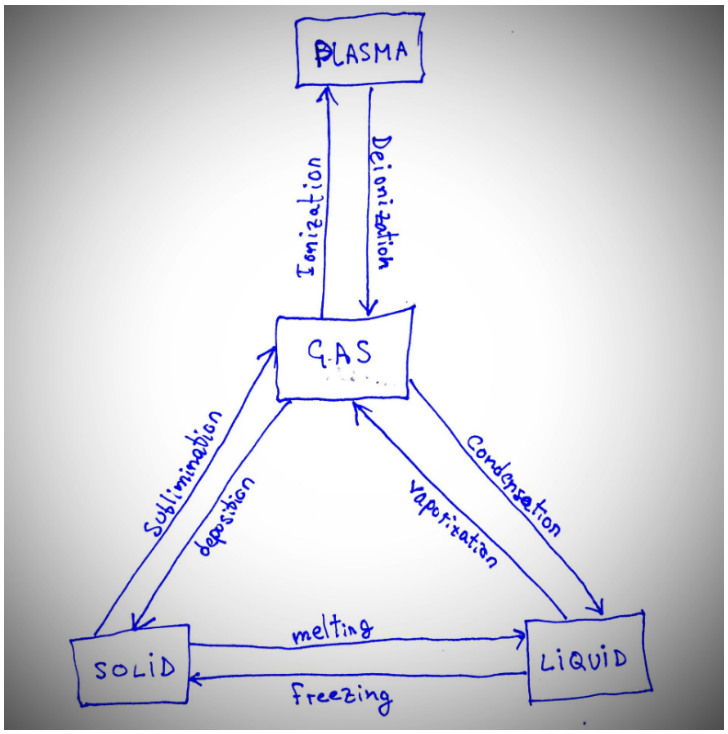
Mar 24, 2020

0.1 Introduction:

0.1.1 What is State Machine?

An State Machine is a mathematical objects made of states, state transitions, and inputs. State Machines are extensively used in computer technology and engineering to version the behaviors of machines. An State Machine has one state and can receive inputs. They provide us with a way of modelling anything in real life. Base on State Machines, we can used in probabilistic applications, such as processing the action event of robot looking for a thing. State machines allow us to easily to expanding our processing (both through design and through code).

Example: the State of Matter:



This state machine has 4 states : Plasma, Gas, Solid, Liquid. There are transitions between states, which are input that caused by nature or human. when transitions is done, states change.

0.1.2 Calculator and state machine

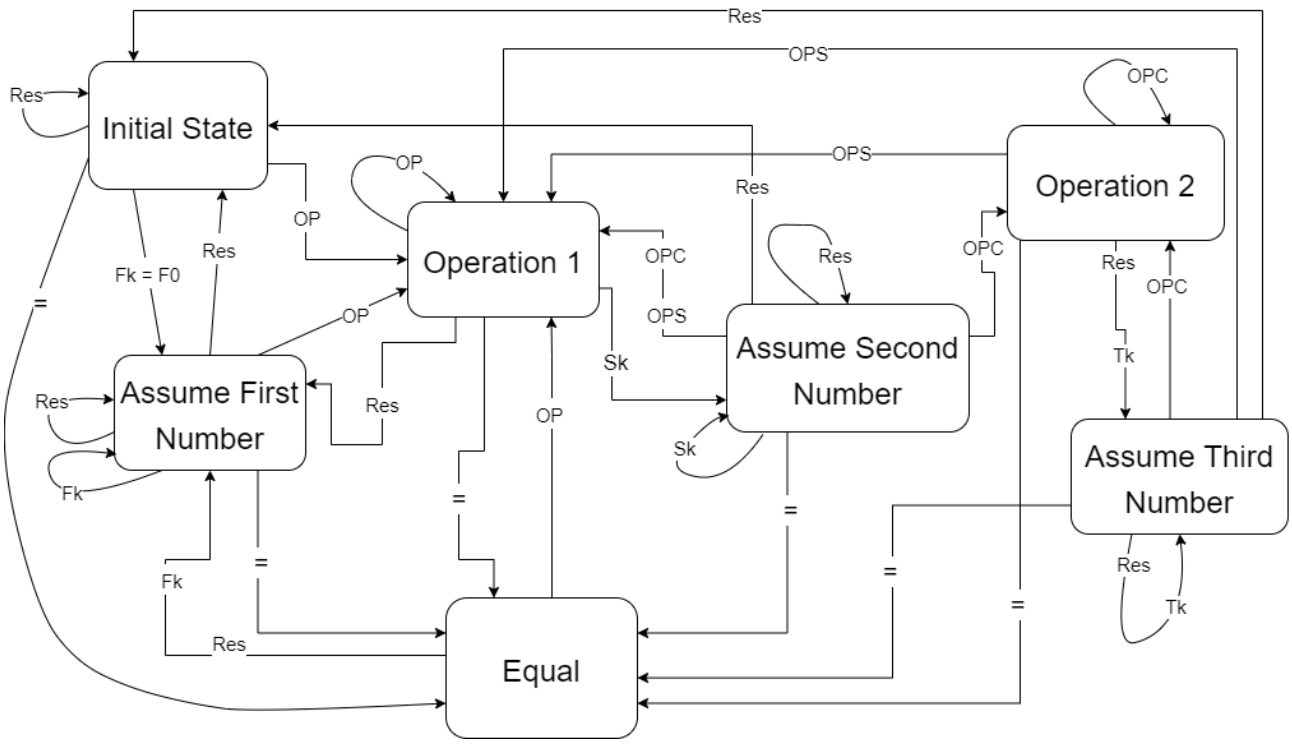
Calculator:

I decided to use the iPhone's Calculator to be the example product for all state in my calculator state machine.

Calculator without using state machine is very simple to process. However, simple also deal with error, Calculator without state machine can sold multiple operation exactly , such as $2 + 3 * 4$ which reality is $2 + (3 * 4) = 14$ or erroneously evaluate in this way $(2 + 4) * 2 = 12$

State machine diagram:

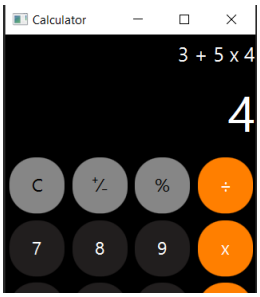
The only way to sold this issues of unstate machine calculator is using this State machine diagram:



To perform this state machine. I using java code with javafx library (`https://openjfx.io/`) for GUI and easy-state (`https://github.com/j-easy/easy-states`) for event-driven.

0.2 iPhone's Calculator State Machine: introduction

0.2.1 Design GUI:



With javafx library in intellij idea, i create java application to easy to describe interface for user. GUI is learning on the concept of default iPhone's Calculator. However, Calculator State Machine has the 2 second screen in top of result screen, which display hold all context of cal-

culation (can hold 3 digits and 2 operation - The purpose of this is check the 2 operation which will evaluate first). Additionally, intellij idea support for building this appliaction to .jar and .exe file

0.2.2 Design State Machine:

With easy-state library, that are to easy to describe state of machine

```
Transition inittoequal = (Transition) new TransitionBuilder()
        .name("inittoequal")
        .sourceState(init)
        .eventType(EQUAL.class)
        .targetState(equal)
        .eventHandler(new equad())
        .build();
```

one transition has a name, source State, target State, event Type (trigger event change state) and event Handler (handle event with will be call when app enter target State.

Additionally, Coming from the state machine diagram there are 7 state with a lot of transition between them.
Symbol in this diagram with be solve like:

F = First Number ← input number event

OP = Operation ← input operation event

Res = AC/C ← input All clear/clear event

S = Second Number ← input number event

T = Third number ← input number event

OPS = simple operation ← (+ or -) event

OPC = complex operation ← (* or /) event

I using 7 variable for remember separate number and operation and output :

```
public static String s_f = "0", s_s = "", s_op1 = "+", s_op2 = "+", s_t = "", s_display = s_f, s_display2 =
```

s_f ← first number← input from button has number text

s_s ← second number← input from button has number text

s_t ← third number ← input from button has number text

s_op1 ← first operation ← input from button has operation text

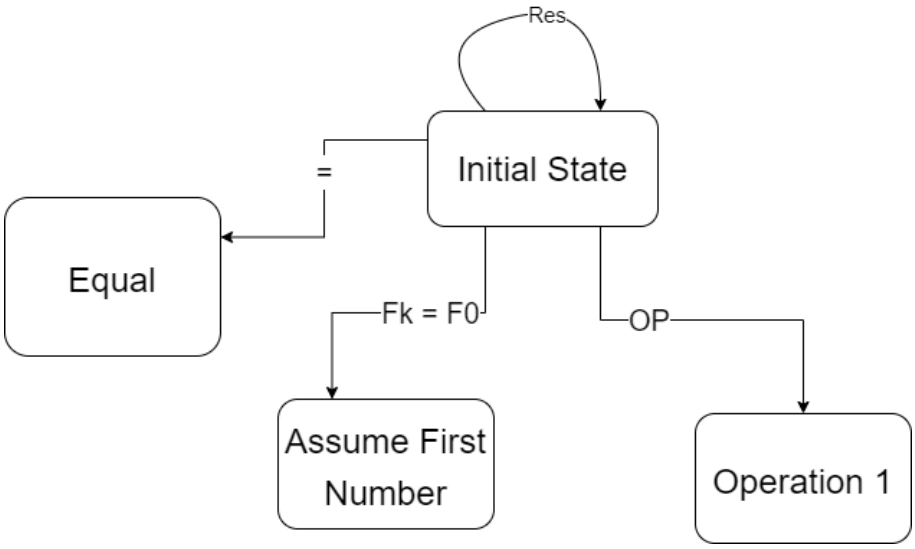
s_op2 ← second operation ← input from button has operation text

s_display ← first display ← output the result equation

s_display2 ← first display ← output of hold the equation

0.3 iPhone's Calculator State Machine: State Description

0.3.1 Initial State



This is the start state so (all the number (s_f, s_s, s_t) set 0 all operation (s_op1, s_op2) also set to +, s_display will show s_f

Pressing AC (Res) button will take back to **Initial state**.

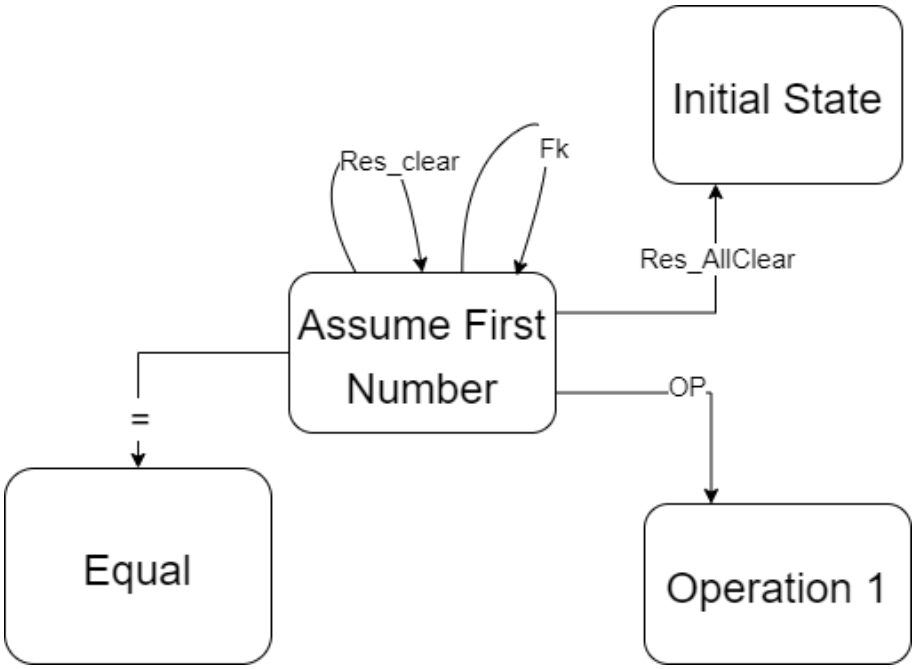
Pressing = take to **Equal state**

Pressing any number, denote by Fk and s_f[0] will fill by

Fk and transfer to state **Assume First Number**

Pressing any operation OP will take into **Operation 1**. Note that this will change s_op1 to become OP , whatever OP may be among +,-,*,/

0.3.2 Assume first number State



This is the submit state for first number s_f will change by adding new number pressing input, s_display and s_display2 will show input number of s_f

Pressing C (Res) will take this state to **Assume first number** if s_f != 0, it will clear s_f (set s_f = 0)

Pressing AC (Res) will take this state to **Initial State** when s_f == 0.

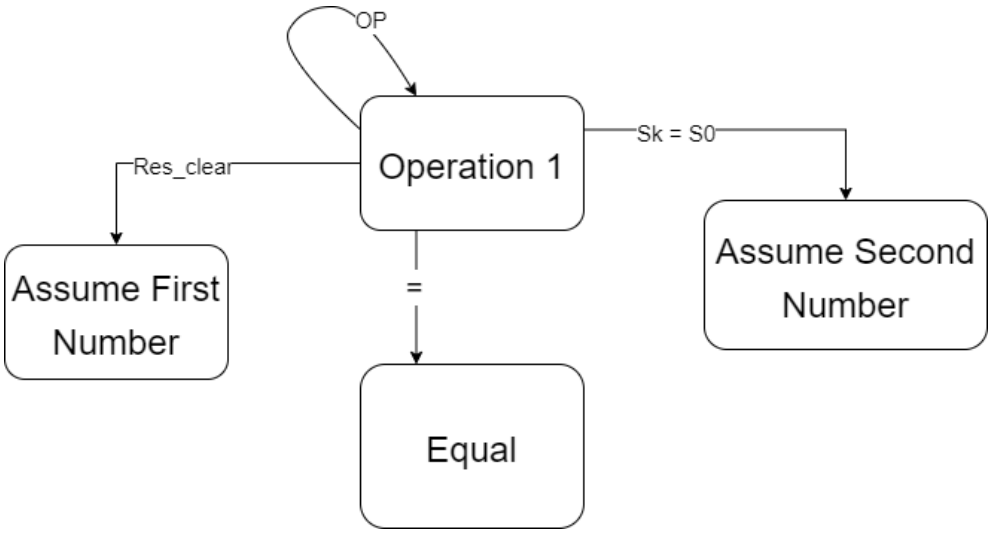
Pressing Fk will take back to this same state,**Assume first number** , and set s_f += Fk.

Pressing = will take to the **Equal state**. Through this,

it will make the evaluation s_f s_op1 s_s and place the result in the s_f when it reaches the textbfEqual state.

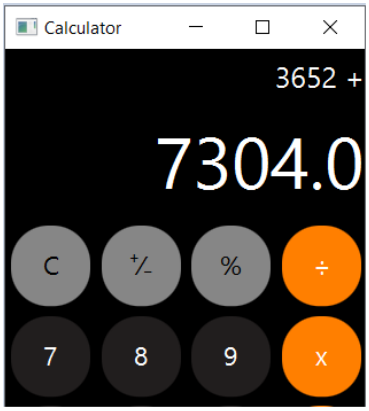
Pressing OP will take to the **Operation 1 state**, s_op1 = OP and duplicate s_f into s_s.

0.3.3 Operation 1 State



This is the submit state for first number s_op1 will change by adding new operation pressing input. s_display2 will show input operation of s_op1

Pressing = will take to the **Equal state**. Through this, it will make the evaluation s_f s_op1 s_s with s_s = s_f and place the result in the s_f

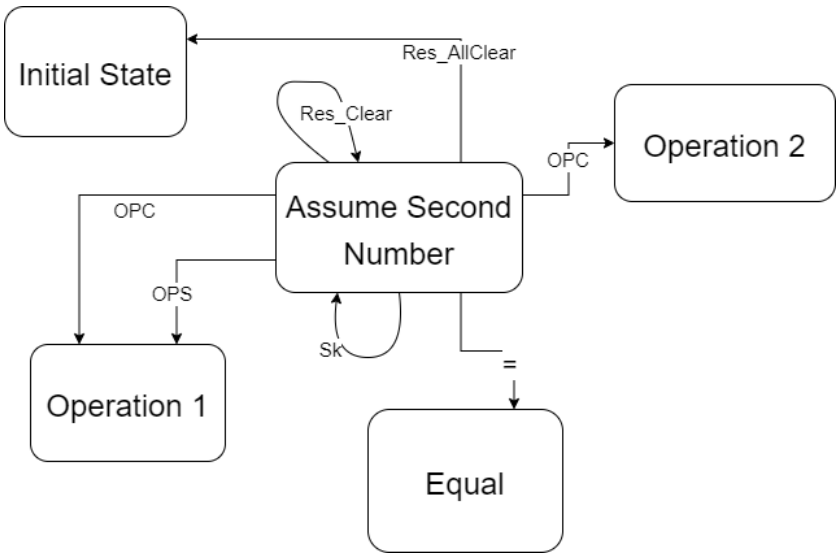


Pressing C (Res) will take this state to **Assume first number**, it will clear s_f and s_op1 (set s_f = 0)

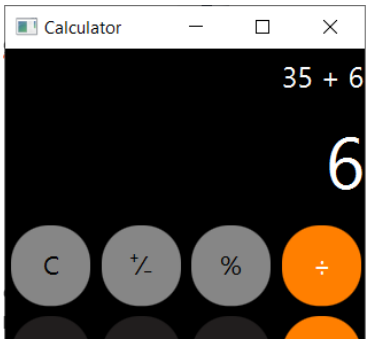
Pressing Sk will transfer to **Assume second number** and set s_s += Sk.

Pressing OP will take to the **Operation 1 state**, s_op1 = OP.

0.3.4 Assume second number State



This is the submit state for first number s_s will change by adding new number pressing input, s_display and s_display2 will show input number of s_s



Pressing Sk will take back to this same state,**Assume second number** , and set s_s += Sk.

Pressing = will take to the **Equal state**. Through this, it will make the evaluation s_f s_op1 s_s and place the result in the s_f.

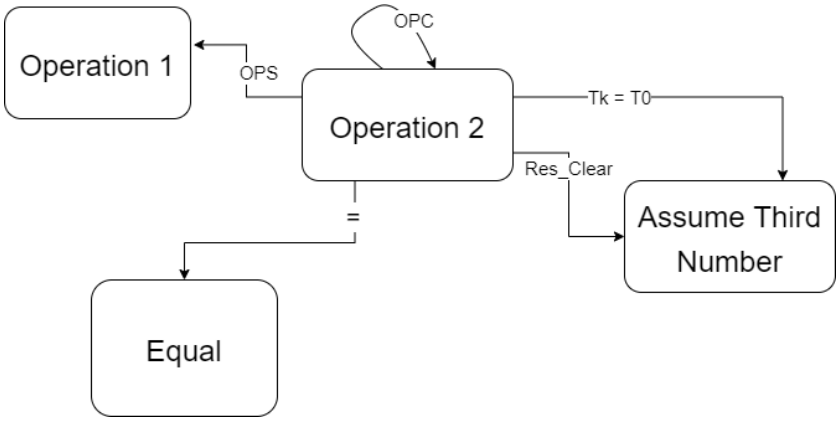
Pressing C (Res) will take this state to **Assume second number** if s_s != 0, it will clear s_s (set s_s = 0)

Pressing AC (Res) will take this state to **Initial state** when s_s =

Pressing OPS (+ or -) will take to the **Operation 1 state**, with s_f = s_s = s_f s_op1 s_s, and replace OPS into s_op1.

Pressing OPC (* or /) will take to the **Operation 1 state**, when OPC = s_op1 and work as same as the above transition.
However when OPC != s_op1 it will take to the **Operation 2 state**, s_op2 = OPC and s_t will replaced by s_s

0.3.5 Operation 2 State



This is the submit state for first number s_op2 will change by adding new operation pressing input.In this state we able to solve this equation 4 + 8 * 9 rightly.

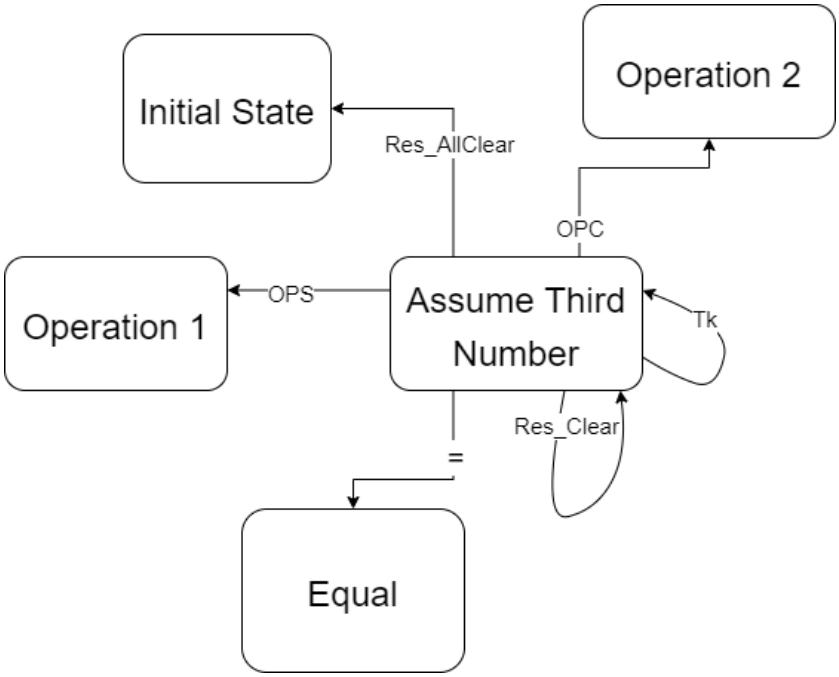
Pressing = will take to the **Equal state**. The evaluation is different however. First, we evaluate s_s s_op2 s_t, place result into s_s (note that making this evaluation before moving to **Equal state**), and then evaluate s_f s_op1 s_s and replace to s_f.
Now s_f = s_f s_op1 (s_s s_op2 s_t).

Pressing C (Res) will take this state to **Assume third number** if s_t = 0, and all parameters will remain unchanged. The display will become s_t. And then pressing Tk is essentially equivalent to pressing C (Res). s_t = Tk

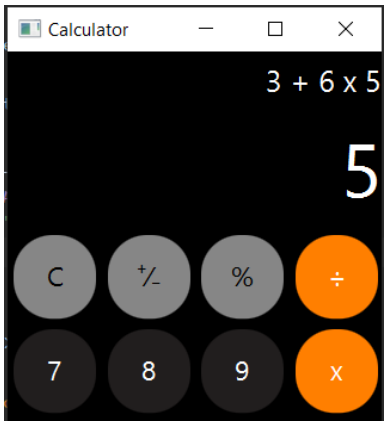
Pressing OPC loop state in **Operation 2 state** and simply change s_op2 = OPC.

Pressing OPS will run the same evaluation done with pressing =, change s_op1 = OPS.

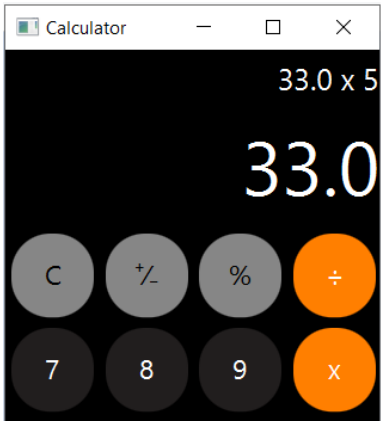
0.3.6 Assume third number State



This is the submit state for first number `s_t` will change by adding new number pressing input.



Pressing AC (Res) if `s_t` = 0 will take back to **Initial state**. Everything will be cleared. However, if `s_t` != 0 , Pressing C (Res) will just clear `s_t` remain in **Assume third number**.



Pressing Tk will just append Tk into the current value of s_t.

Pressing = will evaluate the expression just as evaluated when pressing = during the **Operation 2**, and it will take us to the **Initial state**.

Pressing OPC will take us to the **Initial state**. This will evaluate s_s s_op2 s_t and place the result in both s_s and s_t . Then, it will change s_op2 to be the new input OPC . The display will change back to s_s.

Pressing OPS will take us to the **Operation 1**. This will evaluate the expression similar to how it's evaluated in the **Operation 1**.

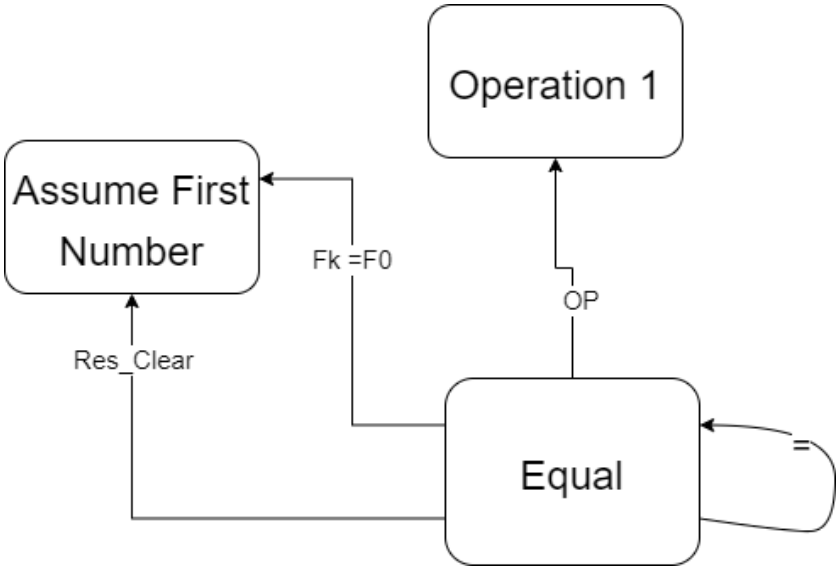
0.3.7 Equal

This state the display s_display alway display s_f.

Pressing = re-evaluates s_f s_op1 s_s and places the result into s_f.

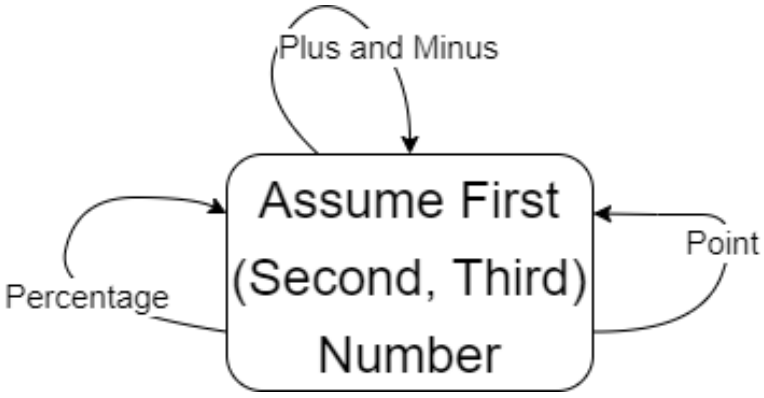
Pressing OP will take to the **Operation 1**. Then, it will make a copy of s_f and place it into s_s . Then, s_op1 will be the newly received OP.

Pressing Fk will take to the **Assume first number**.



Pressing C (Res) will also take us back to the **Assume first number**. However, it will delete s_f and replace it with 0.

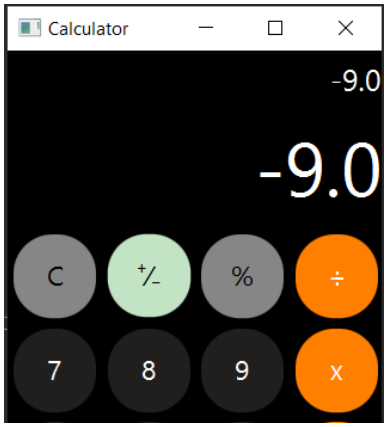
0.3.8 Bonus Transition



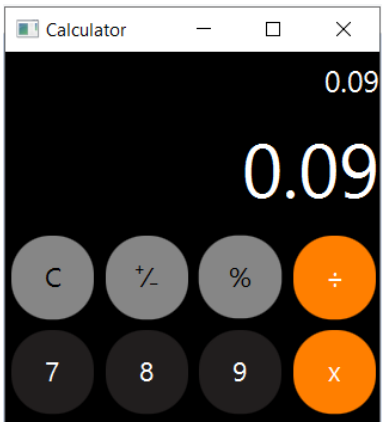
This state is bonus state with some Transition to take more complex form number

Pressing Plus and Minus (+/-), adding plus (+) and minus(-) sign to s_f or s_s or s_f will take back to the **same state**.

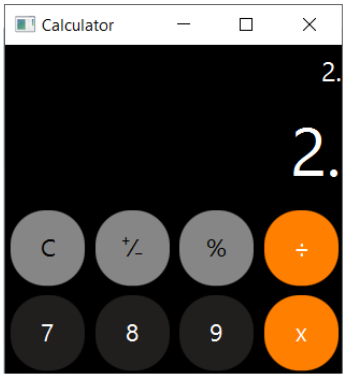
Pressing Percentage (%), adding 0. to s_f or s_s or s_f



by (take s_f or s_s or s_f divide 100) will take back to the **same state**.



Pressing Point (,), adding . and to s_f or s_s or s_f will take back to the **same state**.



0.4 Reference

https://rosettacode.org/wiki/Finite_state_machine
https://github.com/hekailiang/squirrel
https://github.com/j-easy/easy-states
https://www.genuinecoder.com/calculator-javafx-source-code-
https://fsharpforfunandprofit.com/posts/calculator-complete-

Mục lục