Project

Discrete Structures (CAB203)

Semester 1, 2021

Topic

Complicated Wrist Watch

1.0 Introduction

The new digital watch I have acquired has multiple functions, including a timer, alarm, and stopwatch. It only has two buttons, but has three types of button presses or inputs, a Mode button, a Set button and the final input type is long pressing the Set button. However, the user manual for the watch appears to be quite unhelpful in explaining how to access the watch's features, and I am unable to use the watch's features or navigate the watch to my satisfaction. Thankfully, my cousin who possess a collection of old watches is coming to assist me in navigating through the watch's features.

After some extensive studying of the watch's user manual, my cousin was unsuccessful in trying to access the Date setting mode to modify the current date set on the watch. To assist him appropriately, my cousin has requested that I find the sequence of button presses required to get into the Date setting mode.

From reading the user manual, I have discovered all the different mode options that each button press can take me to and have demonstrated this in table 2 below. A brief example of this table can be viewed below in table 1. It was also observed that if the user manual did not specify a mode transition for a button press type, that the watch would remain on the current watch mode it was on.

	Button Presses		
Watch Mode	**Mode**	**Set**	**Long press Set**
Time mode	Timer mode	TimeZone 2 mode	Time set mode

Table 1: Button Press Transitions for Time Mode

2.0 Instance Model

An instance of this problem can be described by a starting mode, a destination mode and by the set of button presses required to take the watch from the start to the destination mode. Let *B* be a set of pairs of Modes (*s, d)* where table 2 describes the mode-to-mode link between *s* and *d*. For example, the table given in section 1.0, is given by:

B = {(Time mode, Timer mode), (Time mode, TimeZone 2 mode),

(Time mode, Time set mode), … }

The instance is modelled by (*B, s, d*), where *s* is the starting mode and *d* is the destination mode. An example is as follows:

({(Time mode, Timer mode), (Time set mode, Time mode), (Timer mode, Stopwatch mode), … }, Time mode, Stopwatch mode)

3.0 Solution Model

We need to identify the sequence of button presses to take the watch from the Time mode, the starting state, to the Date setting mode, the destination state. The first button press must begin from the Time mode and the last button press should ensure the watch goes to the Date setting mode. Therefore, the solution will be a sequence of button presses *s*, *a*, *b*, …, *d* where the first state *s* is the starting mode and the last state *d* is the destination mode, and each pair of consecutive modes in the sequence are joined by a mode-to-mode link from a button press.

4.0 Problem Model

If we let an instance of the problem (*B, s, d*) be given, we can model the watch and its modes as a finite-state automaton (FSA). FSAs are a mathematical model of computation based on the ideas of a system changing state (Old Dominion University, 2020). FSAs have a set of states and inputs, with the transition of state to state movement depends on the input type. FSAs can be described by a set of states, a starting state, a set of accepting states, input symbols and a transition function between states.

Let *S* be the set of states, such that:

$$S = \{ s_0, s_1, \ldots, s_n \}$$

Let the designated starting state be s_0, where $s_0 \in S$.

Let *A* be the subset of accepting states in *S,* where $A \subseteq S$. Typically, the FSA will reject inputs that do not finish on a state $s_n \in A$. However, since we are searching for the exact button presses instead, and not feeding an input string into the FSA, we will not have a rejection state. This will be discussed in detail further in section 5.0, the problem model. For this problem instance, the accepting state will only be the destination state *d*, such that:

$$A = \{ d \}$$

Therefore, let Σ be the set of input symbols, where each button press type is represented as a string. Since we know there are three button types; Mode button (*a)*, set button (*b)* and long pressing the set button (*c)*, we can conclude that:

$$\Sigma = \{a, b, c\}$$

The concatenation of each input type will form a string literal, a regular expression representing the required button presses to take the watch from the start state to the desired state. An example of this is "*aab*", where the button sequences are "Mode button, Mode button and Set button", which takes the watch from the Time mode to the Time set mode.

Let δ be the transition function (or state change function), where it can be defined as:

$$\delta : A \times \Sigma \rightarrow S$$

The following transition table represents all the possible transitions for each state for a possible input type.

State	a (Mode)	b (Set)	c (Long Press Set)
s0 (Time mode)	s2	s10	s1
s1 (Time set mode)		s0	s4
s2 (Timer mode)	s5	s3	
s3 (Timer set mode)		s2	
s4 (Date set mode)		s1	
s5 (Stopwatch mode)	s6		
s6 (Alarm 1 mode)	s8		s7
s7 (Alarm 1 set mode)		s6	
s8 (Alarm 2 mode)	s0		s9
s9 (Alarm 2 set mode)		s8	
s10 (TimeZone2 mode)		s0	

Table 2: Transition Function Table

Each blank in the above table represents the state remaining at its current position on an input type.

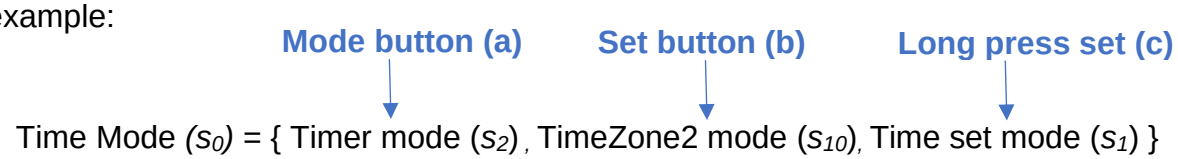
This instance can be modelled by the starting state (*s*), the destination state (*d*) and transitions between the starting state and the destination state, such that there is a point-to-point transition from the starting state to each state before reaching the destination state.

Therefore, the instance model can be modelled by (S, s_0, A, Σ, δ).

5.0 Solution Method

The general solution to the problem will be a regular expression recognised by the FSA, such that when the expression is inputted into the FSA, it will take the FSA from the starting state to the destination state, the only accepting state.

Firstly, we will organise each mode in the total set of modes as a dictionary, for example:



The order of the dictionary represents the order of the input symbols, so for example, in s_0's transition states, on a Mode button press represented by the symbol *a*, it would transition to s_2, as shown in table 2, the transition function table.

Initially, we will create a helper list of zeroes matching the length of the total amount of modes, with each watch mode corresponding to an index value in the list of zeroes. Beginning from the starting state s_0, we will search all the possible transitions for each button press type to verify whether s_0 can immediately transition to the destination state. If it can we will append the button press required to another helper list called sequence. After the starting state is searched, we will set its index equivalent in the list of zeroes to the value of 1 so we can ensure that it has been searched.

If the starting state is unable to immediately transition, we will recursively search all the possible transition states from s_0 until we have found the destination state. If the transition states from s_0 still cannot transition to the destination state, we will recursively search all of their possible transition states and so on until every index value in the list of zeroes is set to 1 and we have found the destination state. Finally, we will then append all the button press required to the sequence list.

The sequence list all of the individual button presses stored as strings, which when all the strings are concatenated, will form the regular expression that will take the FSA from the starting state to the destination state. Additionally, let *x* represent this regular expression.

As there may be more than one transition occurring, we will want to extend δ to multiple transitions, such that:

$$\delta^* : S \times \Sigma^* \rightarrow S$$

Where s is the starting state of the FSA and d is the destination and only accepted state, such that:

$$\delta^*(s, x) = d$$

Therefore, if $x = a_1 a_2 \ldots a_n$, where $\forall\, a \in \Sigma$, the set of input symbols

And if $\delta(s, a_1) = p_1$, and $\delta(p_1, a_2) = p_2$ and … $\delta(p_{n-1}, a_n) = d$, then:

$$\delta^*(s, a_1 a_2 \ldots a_n) = d$$

Therefore, for any destination state, $d \in A$, the set of accepting states, we can identify the sequence of literals by the function:

$$d = \delta^*(s, x)$$

To summarise, the solution can be found by applying the following steps to an instance of the problem:

1. Check if the destination state is within the starting state's transition states
2. Recursively check if the destination state is within all of the transition states of the starting state, and if not, again recursively check in their transition states until we reach the destination state
3. Append all the button presses based on the transitions required from the starting state

References

Old Dominion University. (2020, August 19). *Finite State Automata*. Retrieved from Old Dominion University website: https://www.cs.odu.edu/~zeil/cs390/latest/Public/fsa/index.html