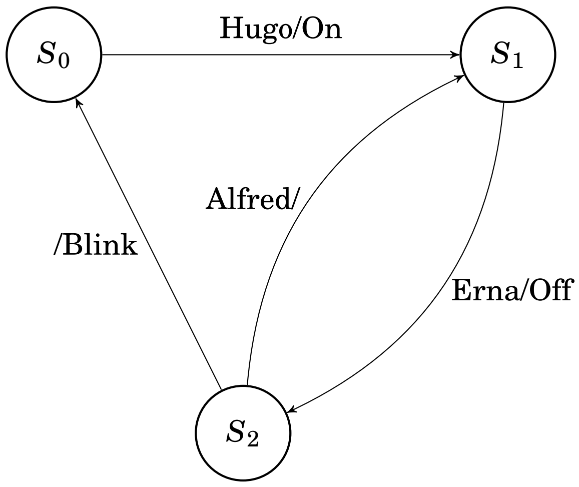
CSC364/CSCM64 Lab 1

To be solved in groups of two.
Last day for lab sign-off: ~~28th February~~ 7th March 2022

Task 1. Consider the "revised" specification of the program for computing square roots, which is found on page 7 (slide 26) of the first set of lecture slides. Is the specification complete? Is it unambiguous? Is it a good specification? Would you make further changes? If so, how much would the implementation and verification have to change to accommodate your changes?

Expected Outcome. A short discussion of the above questions.

Task 2. Consider the following simple state machine:



$$\begin{tikzcd}
S_0 \arrow[rr, "Hugo/On"] & & S_1 \arrow[dl, bend left, "Erna/Off"] \\
& S_2 \arrow[ul, "/Blink"] \arrow[ur, bend left, "Alfred/"] &
\end{tikzcd}$$

1. Download, study, compile, and run the Java program **SimpleStateMachine**, to be found under

   `http://cs.swan.ac.uk/~csmarkus/Tools/`

   SimpleStateMachine is (supposed to be) an implementation of the above simple state machine.

Expected Outcome. Screenshot of the running program.

Task 3.

1. Describe an interface for testing the **SimpleStateMachine** program in Java: what is its input alphabet? what is its output alphabet?

2. Can you tell in which state the program starts?

3. Design a testing strategy for showing that the **SimpleStateMachine** program is a valid implementation of the above simple state machine. To this end, formulate your own testing aims (*e.g.*, after inputs `hugo` and `erna` it is possible to obtain the output `blink`; whenever `hugo` is the input, we see output `on` or no output; it is not possible to obtain output `off` without seeing output `on` before.)

4. Realise your testing strategy by

   (a) Giving a test suite (relating test cases to your testing aims), and

   (b) testing **SimpleStateMachine** with your test suite.

Expected Outcomes. A list of testing aims; a test suite; documentation of your testing and your test evaluation.

Task 4.

1. Implement the simple state machine for the toaster as given in the lectures. To this end, create a new Java program **Toaster** by modifying the Java program **SimpleStateMachine**.

2. Describe an interface for testing your **Toaster** program in Java: what inputs can you enter? what outputs can you observe?

3. Design a testing strategy for showing that your **Toaster** program is a valid implementation of the simple state machine. To this end:

   - formulate your own testing aims

   and realise your testing strategy by

   - giving a test suite (relating test cases to your testing aims), and
   - testing **Toaster** with your test suite.

Expected Outcomes. Your toaster program; a list of testing aims; a test suite; documentation of your testing and your test evaluation.