Logic Flow: The SNF Tool Picker

Samuel Mumford
smumfor2@stanford.edu

January 29, 2021

1 General Apology

First, I would like to apologize to anyone forced to read my HTML code. It is not a language I normally use and it was the first ever website that I built. The code is functional, but inelegant and hard to read. If all you need to do is describe the program, the “General Approach" section should be sufficient. For a full breakdown of the program, read the rest.

2 General Approach

I tried to account for the fact that there are many possible priorities of users, and each user may only care about 1-2 of the available tool features. So instead of creating a logic flow/decision tree where users have to make a decision on every tool option, they instead specify the things that they need and allow all the other options to be “any of the above". So what the program essentially does is make a list of tool options specified by a user x, initially set to “accept anything". When the user hits a button to change the answer to a question, it changes one of the components of x. The program then looks through the list of tools to see if the newly updated x is acceptable to each tool.

3 Initializing Variables

3.1 Lists Needed for User Specifications

The program begins with:

- A list of user options x initialized to be $[-1, -1, -1, -1, -1, -1, -1]$ for 7 tool options[1]. The -1 denotes “unspecified/don’t care" for each tool options.
- A list of tool names, for metal tools this is $AllTools$ = ["aja", "lesker", "innotec", "intlvac evaporator", "intlvac sputter", "fiji-1", "fiji-2", "fiji-3"].
- A list of acceptable options for each tool, provided in the same order as $AllTools$. This is called $AllGuides$. As an example, the first item in $AllGuides$, for the aja, is:

  [[-1, 2],

[1] x is initialized to be [] and then -1 is appended to x for each option/question. In HTML, all variables are global unless otherwise specified, so x can also be passed into the other functions and altered

[-1, 'Ti', 'SiO2', 'Au', 'Fe', 'Cr', 'Ni', 'Ag', 'Pt', 'Al', 'Pd', 'NiO', 'Cu'],

[-1, 1],

[-1, 0, 1, 2],

[-1, 1],

[-1],

[-1, 0, 1]]

Note that it is a list of lists. Each sub-list tells you which options are acceptable. So the aja accepts -1 or 2 as user specifications to option 1.[2]

3.2 Lists Needed for Buttons

First, two lists are needed to define all the buttons that the user can interact with:

- $AllPrompts$ = ["Contamination Status", "Deposited Material", "Conformal", "Wafer Size", "Process Gas", "Temperature", "Batch Size", "Result"] defines all the options which can be specified by a user.
- A list of lists $Allnames$ which says the names of all the buttons for each option. Note, the length of each sub-list also tells you how many buttons you will need to make. Here $Allnames$ is:

  [["Clean", "Semiclean", "Gold Cont", "Clear Answer"],

  ["Show Material Menu", "Not Listed/Personal Source", "Clear Answer"],

  ["Conformal/Sputter", "Vertical/Evaporated", "Angled", "Hyperconformal/ALD", "Clear Answer"],

  ["4" Wafer", "6" Wafer", "Piece/Chip", "Clear Answer"], ["Need Reactive Gas", "No Gas Contamination", "Clear Answer"],

  ["Want Substrate Heating", "Clear Answer"],

  ["<5 Wafers", "<10 Wafers", "Large Batch", "Clear Answer"]]

4 Buttons/User Inputs

4.1 Option Specifying Button

- The "Option Specifying" *os* button: This is the most common type of user input, used to specify a tool option. An example would be the "Clean" button specifying the "Contamination Status" option. Each *os* button is associated with a (a, b) tuple of integers when it is initially made. Also, this was the first type of button I built, so many elements or functions used in this button are repurposed in later functions. When this button is pressed, three things occur:

[2] A much better way to do this would be as a dictionary with keys and items, but I didn't know how to do that in HTML

- The a-th element of x is changed to b using the $MyFunction(a, b)$ function[3]. So when os button $(1, 2)$ is pressed, if x were initially $[-1, -1, -1, -1, -1, -1, -1]$, it becomes $[-1, 2, -1, -1, -1, -1, -1]$
- The $AllGuides$ list of lists is searched through using the updated x in the $WriteGuide()$ function. In a for-loop using index i over each element or tool in the $AllGuides$ list of lists:
  * Pick out $AllGuides[i]$, calling that $TempGuide$. $TempGuide$ is the list of lists specifying all options acceptable to tool i.
  * Set a boolean $Viable$ to true.
  * In a for-loop using index j over each element in $TempGuide$:
    · Check if $x[j]$ is one of the elements of $TempGuide[j]$. If it is not an acceptable option for the tool, set a boolean $Viable$ to false.
  * After finishing looping over j, if $Viable$ is still true, add the i-th tool name to a string. Print that string in the textboxes on the top and bottom of the page.
- The function $writeFunction(a, b)$ is called. It appends text box a with the button name of button b.

4.2 Clear Option Button

- The “Clear Option" buttons are secretly option specifying buttons, but instead of being specified for (a, b), b is always set to -1. So it is just an option specifying button with $(a, -1)$.
- The “Clear All Options” button also does the same thing, but calls $MyFunction(i, -1)$, $writeFunction(i, -1)$ are called for each i, and then $WriteGuide()$

4.3 Open Links Button

- The “Open Links" button calls a very similar function to the $WriteGuide()$ used to make a list of viable tools on the top and bottom of the page. It does the exact same procedure, but instead of appending the name of tool i to a string when a viable tool is found, it opens a new tab with the link to the tool info page for tool i.

4.4 Show Material Menu Button

When the “Show Material Menu" button is pressed, it displays/opens a new element called $myForm$ where materials to be deposited can be specified. In that form, when the “Close” button is pressed, $myForm$ is closed/not displayed. There are three other buttons:

- The “Search" button uses the text specified by the user and performs a search over the list of materials. The list of materials is in fact a list of lists with aliases. For each element in the list of lists called $AKAs$, look to see if the specified user text matches a material (note, the matching is not case-sensitive, but it does need to be exact otherwise). Once a match is found, display the base material name in the “Result" textbox.

[3] It is called this because it was the first ever HTML function I made...I did say I don’t normally code in HTML. Also recall that x is a global variable and does not need to be passed as an argument in any of the functions

A relatively quick extension of the searching would be to look for near matches.

- The ‘Accept" button is just an option specifying *os* button with (1, *mat*) where *mat* is the material displayed in the “Result" textbox.
- The “Show Materials" button displays all the available materials to be deposited in the top textbox.

4.5 Defining the Buttons

With every type of user input needed defined, perform two for-loops:

- for *j* in the range of 0 to length($AllPrompts$):
  - Make a text box with the prompt name in it.
  - Make a “Help" button with associated number *j*
  - Make a “Clear Answer" button with associated number *j*
- for *i* in the range of 0 to length($Allnames$[j]):
  - Make a “Option Specifying" button with associated numbers (*j*, *i*).