Testing Automation: Language & Automata

Abstract

Automated testing is the most sought after field by testers. The basic knowledge of automation from the theory of computation along with programming knowledge can provide the requisite solution. The problem need be dwelled in to parts such that the common programming practices along with the simple 'logical machine' design can help reaching the resolution. The proposed mechanism is good enough for solving unary problems but a far more sophisticated approach is requisite for apprehending 'Universal Mechanism'.

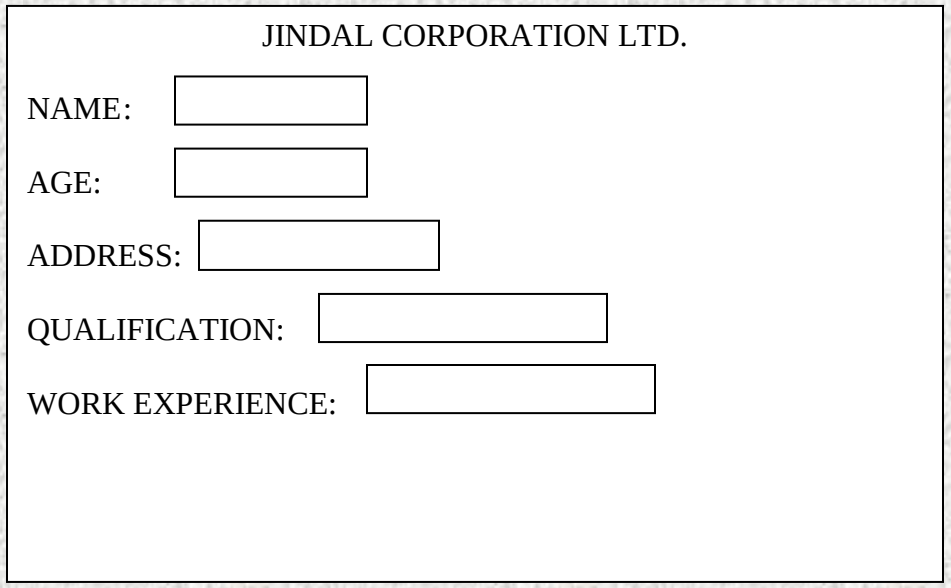
Gaurav Jindal
Asst. Professor
Gitarattan International Business School
Madhuban Chowk, Rohini

Testing Automation: Language & Automata

Testing can be automated if the processes involved in testing can be described in a way that can be apprehended by a language. More generally speaking, we need to express the requirements of testing considering the concepts of software analysis and design. The general principles involved in software making include the requirements analysis, design, coding, implementation, testing and maintenance. As we wish to make our testing phase automated because this is most tedious and lengthiest job, we have to observe this phase alone as a complete project.

While automating 'Testing' we know the requirements. So, we can move straightforward towards the 'Design' phase of 'Testing Automation'. Here let us examine a situation:

We want to test a webpage that accepts 'NAME', 'AGE', 'ADDRESS', 'QUALIFICATION' and 'WORK EXPERIENCE' as given:



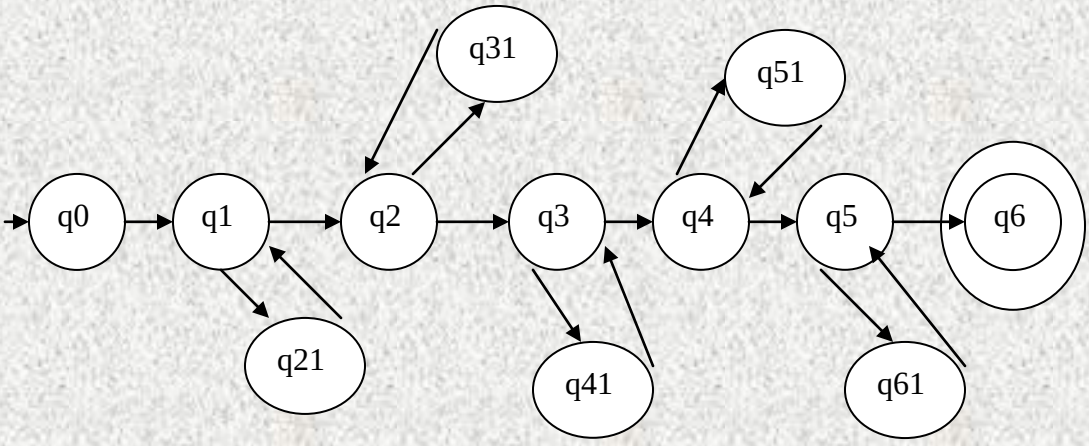
To complete the task, we need to do several iterations of putting different names of different lengths, several values for age less than a specific value as well as more than a specific value and also values other than pure numeric. Similarly, for address part, various combinations need to be checked. For qualification part, some values may be predefined and the inputted value needs to be compared among them. Lastly, the experience should be much lesser than the already entered age value.

The analysis of an automated testing for the given situation has revealed the requirements. Now we can move towards the design phase.

From the theory of Automata, we may take the following notations:

S.No.	State	Reference
1	q0	Empty Form
2	q1	A name entered
3	q2	Name Ok, Age Entered
4	q3	Age Ok, Address Entered
5	q4	Address Ok, Qualification Entered
6	q5	Qualification Ok, Experience Entered
7	q6	Form Accepted
8	q21	Name Incorrect
9	q31	Age Incorrect
10	q41	Address Incorrect
11	q51	Qualification Incorrect
12	q61	Experience Incorrect

For the given problem following automata can be designed:



We know that if automata can be generated for a task, then definitely a solution exists for the same. Now, the solution can be easily derived by

choosing an appropriate language and inserting various test cases at each clause culminating to a proper solution whether the software under testing is acceptable. Though, the problem seems to have been sorted, but there still remains an important task unattended. We have not yet discovered the method to automatically generate various 'testing inputs'. What we should be looking forward is the complete test suit generated automatically and be clinically associated with the above mentioned automata in order to provide complete solution to the problem.

Let us divide the problem in three parts: the first part need be 'Name' as it consists only of alphabets. Second part may consist of 'Age' and 'Experience' those consists of precisely numeric values and the third category consisting of 'Address' and 'Qualification' those may contain alphanumeric values.

Generating different integral values is the most simple task, as we may recall the random () and randomize () functions of 'C Language'. Again, generating a random string of varying length can be accomplished by keeping in view a few things:

- ASCII values of characters are simple integers that can be generated by using random function.
- Sought length of a string is again an integer.
- Lastly, difference between permissible types of characters can be verified by the knowledge of ASCII values.

Hence by the proper usage of a language and our knowledge of automata, we can definitely automate the testing procedure but there seems to be a big catch in the proposed system. So, far we have been talking about only one type of situation. It follows that for each kind of testing a new automation technique need to be developed. We still need to find a universal solution. May be a deep insight of the problem with due understanding of a 'Universal Turing Machine' accompanied with the knowledge of a good language would help us cross the barriers and reach towards a full fledged 'Automated Testing'.