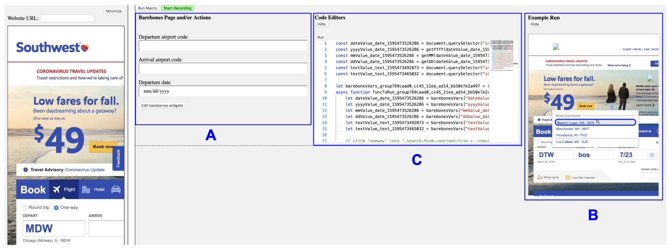
Designing Tools for Intuitive Creation, Generalization, and Maintenance of Web Macros

Rebecca Krosnick
Computer Science and Engineering — University of Michigan, Ann Arbor
rkros@umich.edu

I. Background and Completed Work

Web automation tools enable users to write or record macros to be replayed in the future, to reduce the burden of manually performing tedious and repetitive web tasks (e.g., paying a bill, ordering supplies, or scraping data). However, the representation of such macros is typically text-based program scripts, which lack context about semantics and web page UIs, making them difficult to understand, generalize, and repair.

I have begun designing a new representation of macros that goes beyond just code to provide a concise semantic description of each macro step. The macro is summarized as a "barebones web page" (see Figure 1-A) that illustrates through the most basic, unstyled UI elements the inputs and outputs expected. The idea is that the user can take a quick glance at this barebones page and understand what input they need to provide, and what output they can expect. They can then enter input via these UI elements (e.g., text fields, dropdown menus, radio buttons). Additionally, a sequence of UI animations (Figure 1-B) accompanies the barebones page to illustrate the UI state and macro actions corresponding to each section of the barebones page. Finally, there is a code editor (Figure 1-C) available for users to refine the macro logic. I have built an initial version of this tool (see Figure 1) that lets users record their actions on a website of their choosing, manually create the barebones page, modify the automatically-generated code to reference these barebones elements, and view automatically-generated UI animations.



```
const dateValue_date_1595473526286 = document.querySelector("ir
const yyyyValue_date_1595473526286 = getYYYY(dateValue_date_159
const mmValue_date_1595473526286 = getMM(dateValue_date_1595473
const ddValue_date_1595473526286 = getDD(dateValue_date_1595473
const textValue_text_1595473492873 = document.querySelector("ir
const textValue_text_1595473465032 = document.querySelector("ir

let barebonesVars_group769caad0_cc45_11ea_ad14_bb50e7e2a497 = {
async function funcToRun_group769caad0_cc45_11ea_ad14_bb50e7e2a
    let dateValue_date_1595473526286 = barebonesVars["dateValue
    let yyyyValue_date_1595473526286 = barebonesVars["yyyyValue
    let mmValue_date_1595473526286 = barebonesVars["mmValue_dat
    let ddValue_date_1595473526286 = barebonesVars["ddValue_dat
    let textValue_text_1595473492873 = barebonesVars["textValue
    let textValue_text_1595473465032 = barebonesVars["textValue

    // click 'oneway' into '.search-form--options-trip > .input
```

Fig. 1. A macro that performs a flight query: A) the barebones UI elements, B) an animation of the recorded macro, and C) the macro's code

II. Future Work

One piece of upcoming work involves making the creation and integration of this barebones page simpler, for example inferring or suggesting likely barebones elements based on the user's recorded actions and updating the macro code accordingly to reference these elements for input. This might also involve designing a library of helper functions to help users more intuitively program semantic actions, for example to "click *the element whose value equals* ⟨month⟩" without needing to write code logic to appropriately query the DOM.

Another important line of future work is making the generalization of macros more intuitive. Actions recorded from one user demonstration of desired macro behavior likely represent only some subset of user input and scenarios. Additional branching and logic is needed to support other user input and scenarios. For example, selecting a date in the current month from a calendar widget likely involves only two clicks (opening the widget, and selecting the date), whereas selecting a date for several months down the line also requires clicking the "next month" button multiple times. I am considering a couple different approaches to help users generalize their macros: 1) tools to help them uncover where their macro is failing and 2) the ability to create multiple demonstrations (i.e., with different input values and workflows), review the generated code for each, and appropriately combine snippets with conditional and looping logic. These tools and more would likely be helpful for macro repair as well, e.g., for when a website's DOM structure or UI has changed.

Finally, generalizing macros to work across websites, in other words "semantic macros", could be an exciting new direction. Tasks such as paying a bill are semantically similar across websites even though user workflows and website implementations differ. A joint representation of semantically similar actions could make it easier for users to understand, maintain, and adapt macros.

III. Doctoral Symposium

During the symposium I am most interested in hearing thoughts on what parts of my system are appropriate to automate and which are better left to the user. Currently my plans for the system primarily rely on the user to manually create and integrate the barebones page, and generalize the macro. It would save the user time if barebones pages could automatically be created and integrated, but I am not sure what semantic understanding is possible for the machine. Additionally, it could be great if the machine could automatically synthesize generalized macro code from multiple user demonstrations, but my background in program synthesis is limited; what is possible with current state of the art program synthesis, and what are the input requirements?