Towards Semantic User Intent Expression and Refinement in Web Automation

Kevin Pu [1], **Tovi Grossman** [1] **and Yan Chen** [2]

[1] *University of Toronto, Toronto, Canada*
[2] *Virginia Tech, Blacksburg, VA*

Abstract

The automation of data science tasks alleviates repetitive efforts but requires user efforts in program creation, either through explicit specifications or demonstrations. The vast variety of data representations (e.g. text, image, UI), complex task requirements, and users' nuanced intents prevent non-experts from adopting automation in their workflow. This paper summarizes the recent progress in semantic user intent expression and task requirement decomposition to aid high-level program definition and refinement with generalizable automation execution. The paper further proposes the future direction for utilizing multiple modalities to represent web data and user intents, as well as a novel refinement technique that incorporates implicit human feedback via user demonstrations on subtask variants. This design exploration aims to shed light on how to enable more robust automation with less effortful user specifications.

Keywords: Web automation. PBD. Neurosymbolic programming. Semantic UI.

1 INTRODUCTION

Data science tasks, such as data collection and analysis, have become increasingly prevalent and vital in today's organizations, spanning enterprises, governments, and schools. However, non-expert end-users cannot easily conduct these tasks in their workflows. Existing tools often demand programming expertise to implement task-specific scripts that are not generalizable. Mastering data science tasks can also be challenging as they exist across various systems like Microsoft Excel, mobile applications, and backend servers, demanding proficiency in multiple platforms with vastly different operations. In particular, web-based applications have emerged as a specific domain for automation that leverages the enormous domain of online data. Within this context, users including clerks, data workers, and researchers extensively rely on web platforms to execute essential yet repetitive tasks such as data scraping and data entry. The integration of the Web and web-based applications into task workflows has become a crucial component of the global trend toward digital transformation. Despite their importance, these processes often involve monotonous work that lacks fulfillment. Manual execution of these tasks on a regular basis can lead to human errors like duplicate or missed entries, along with feelings of frustration [1].

To alleviate manual effort, Robotic Process Automation (RPA) utilizes software robots to automate otherwise tedious processes. Web automation, specifically, simulates human interaction with web-based systems to efficiently execute web actions (macros). It assists users with repetitive tasks and has proven to be faster and more accurate for various task types compared to manual effort [1]. While RPA is expanding rapidly to many domains, it is still technically demanding to create such automation programs based on the specific task requirement. Past research has developed techniques to help users of all expertise levels to create their intended web automation programs [2]–[6]. However, these techniques are limited to creating programs with specifications at the website syntax or symbolic level (e.g., scraping the first two items in each row of a table), requiring users to have expertise in the web structure. Users can not easily specify their high-level intents and conditions to create program logic that achieves their goals. Therefore, the **first key challenge** in the adoption of RPA is the barrier it presents to non-expert end-users, who often possess the required domain knowledge but lack the necessary programming skills to create automation programs, thus impeding their ability to leverage its benefits effectively. Moreover, tools generate programs with fixed logic for repetitive tasks but struggle with generalization when faced with varied input data requiring diverse graphical user interface (GUI) interactions. This poses the **second challenge** in RPA: developing generalizable programs that meet user criteria across the constraints of unorganized data input format and various web symbolic structures. Furthermore, it is crucial to recognize that the complexity of automation often requires

PLATEAU
13th Annual Workshop at the Intersection of PL and HCI

DOI: 10.35699/1983-3652.yyyy.nnnnn

Organizers:
Sarah Chasins, Elena Glassman, and Joshua Sunshine

the resulting program to refine its logic to adapt to different decision-making processes encountered during its execution. Parts of this adaptive refinement involve incorporating the user's human feedback and reconstructing the automation to tailor it to the user's specifications and needs. Existing RPA tools elicit the user's refinement need via additional demonstration efforts or explicitly defining better program specifications [3], [7]–[9]. However, this increases the workload for the user and demands the user to maintain a mental model of the existing program's execution logic and the capacity to edit and/or add additional program logic for more accurate automation. This **third challenge** necessitates a more efficient user refinement strategy that efficiently incorporates human feedback in program synthesis with minimum added burden.

In order to lower the barriers of automation for non-expert users, it is crucial to extend the current techniques and address these three challenges. This paper presents a neurosymbolic approach that aims to empower non-expert users in conducting data science tasks through systems that infuse symbolic program synthesis and statistical inferences. With initial progress in semantic-level program specification and automated inference of task steps [7], [8], this paper illustrates the future steps to further transform the way people conduct data science tasks by connecting user needs with complex program space by leveraging large-language models (LLMs) and visual-language models (VLMs). Emerging works utilize multiple modalities in understanding UI elements at a semantic level [10] or employ automated crawling to construct vision models that predict UI categories and affordances [11], [12]. This paper intends to illustrate future directions for semantifying UI for automation purposes, incorporating advanced multi-modal techniques to both encode the complex data representation on the Web platform and also translate the user's macro or natural language intents to the corresponding UI actions that fulfill the user's request. Additionally, this research expands on the existing human-AI collaboration paradigm to propose novel designs of user expression methods that elicit intents via implicit demonstrations of sub-task variations. When users encounter corner cases that afford opportunities for automation refinement, the envisioned system subtly alters the current task goal and asks users to demonstrate how the automation program should execute in different branched scenarios. Through these demonstrations, the system hopes to imply the user's decision-making process and reason about the additional specifications needed to reconstruct the same decisions for future conditions. By adopting a neurosymbolic approach, the envisioned system represents complex web data through multi-modal semantification and enables end-users to communicate nuanced automation logic using implicit demonstrations on task variations, refining generalizable programs without requiring explicit specifications.

2 RELATED WORKS AND CURRENT PROGRESS

2.1 Automation Tools and Programming-by-Demonstration

Many tools have been developed to help users to create automation programs. For instance, commercial tools like Selenium [13] and Beautiful Soup [14] allow developers to select elements and define actions to automate. Research tools like Sikuli [15] allow users to identify a GUI element (e.g., an icon or a toolbar button) by taking its screenshot. Although these tools help lower the effort of creating programs, they all require some level of programming knowledge and can only operate on symbolic rules. Even for professional developers, creating automation programs is a non-trivial task. A study showed that experienced programmers have difficulty writing web macros using common web automation frameworks [16]. Participants pointed out that a primary hurdle was the labor of checking syntactical element selectors to create their programs, causing inefficiency and errors. In addition, the program might not generalize to cross-webpage selections where the elements don't have syntactic similarities.

A programming-by-demonstration (PBD) approach has then been adopted by many tools in order to reduce the expertise required, since users only have to interact with the target applications rather than write code [17]–[20]. Among these application domains are text manipulation [21]–[25], image or video editing [26]–[28], and GUI synthesis [29]–[32]. For web applications, PBD helps build automation program without requiring users to understand browser internals or reverse-engineer target pages manually. CoScripter [1], Vegemite [33], Rousillon [9], UiPath [34], and iMacros [35] are examples of the PBD approach to web automation. The resulting programs from user demonstration are represented

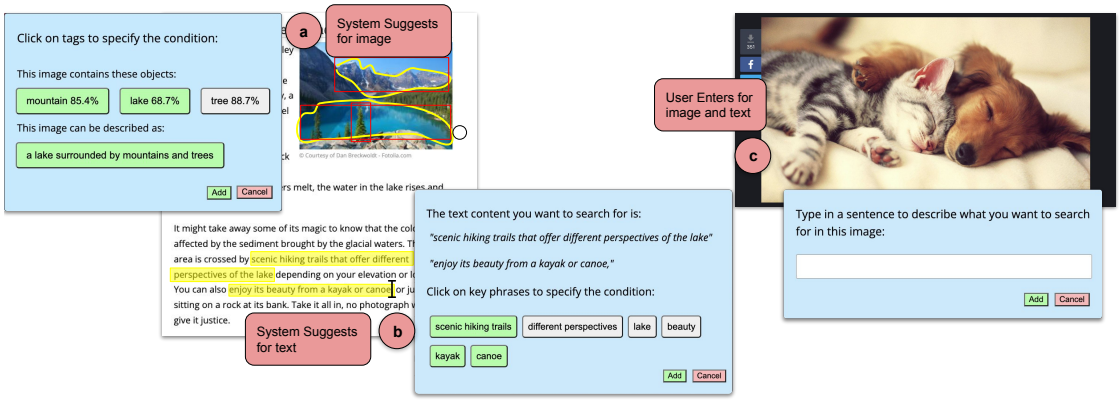

Figure 1. Two condition specification methods in SemanticOn [7]. After entering the system suggests mode, the user may draw on images. Then, a prompt ⓐ containing detected objects within that region, and a general description of the image will be shown. Similarly, the user may highlight sections of text, and the system will prompt ⓑ important key phrases within the highlighted portion. Users may manually enter conditions by clicking on an image or section of text and entering their conditions ⓒ.

in visual formats such as a workflow chart [34], a for-loop [9], or in domain-specific-language (DSL) code [35]. These representations still require programming expertise, and users have to manually edit program logic which is often nested and convoluted.

2.2 SemanticOn: Specifying Semantic Conditions in Automation

The adoption of RPA for non-expert users is hindered by the abstraction gap that exists between the user's intended purpose of the program and the technical implementations required to achieve it. Thus, effectively communicating user intent is a major challenge in these PBD systems, and many systems have proposed bridging the gap between user intent and system understanding. Systems like PLOW [36] and PUMICE [37] allow users to express concepts (e.g. order ice coffee when weather is hot) in natural language and then learn the concepts to generalize the automation. Scout [38], Designscape [39], and Iconate [40] allow users to iteratively refine their intent by directly manipulating the AI-generated artifacts. Another work, APPINITE [41], also encapsulates the user's intent in natural language instructions and clarifies the intention in a back-and-forth conversation with the AI. Despite promises, specifying intents to cover every case of action can be tedious. Furthermore, users may not know all the cases in the first place. This suggests that tools need to elicit users to better formulate their intent before creating automation programs.

To this, the author proposed SemanticOn [7], a human-centered interactive AI system that allows users to *specify*, *refine*, and *incorporate* visual and textual semantic information as conditions in web automation programs via two methods: natural language description via prompts or detailed information highlighting with system support (Fig. 1). In addition to the conventional PDB approach, SemanticOn combines the relative strengths of statistical machine learning models for unstructured information (e.g. image and text) and rule-based program synthesis techniques for web automation. Users can avoid the need to write web macros for each task and the labor of manually filtering scraped content in post-processing. By interacting with SemanticOn, users can compose semantic conditions for the desired content to scrape and demonstrate actions on the target website, resulting in the synthesis of automation programs across different webpages without any coding required. This no-code process involves the collaborative refinement of the condition set and rectification of result selection errors, facilitating continuous improvement in both the system's and the user's comprehension of the filter criteria. By doing so, a new interaction paradigm is introduced for users to continuously add/refine semantic specifications in a programming-by-demonstration system.

2.3 Neurosymbolic Systems in Automation

One limitation for systems like SemanticOn is that they are designed to handle uniform logic – a binary conditional that determines action or no action (e.g. scrape or skip) applied universally to all content

and input. Examples include downloading an image when it contains key objects, or running a macro when the weather is above a certain temperature [37]. In this case, despite lower creation effort, users still need to recreate a program when there are multiple specifications that correspond to different web actions. Commercial tools like UiPath [34] and iMacros [35] allow users to set conditional actions on specific page elements via programming. But, they also lack task understanding to generalize the conditional outside of the symbolic element (i.e. HTML tag) and could not execute different actions based on different input data.

Alternatively, researchers designed neurosymbolic languages with both neural and symbolic elements to create programs that satisfy new specifications via statistical inference. Neurosymbolic programming is a generalization of classical program synthesis, bridging the gap between deep learning and program synthesis. Unlike deep learning, neurosymbolic programs can often represent long-horizon, procedural tasks that are difficult to execute using deep networks, and they are also generally easier to interpret and formalize than neural networks [42], [43]. In contrast to symbolic approaches, neurosymbolic programming does not require all specifications to be hard logical constraints.

However, this approach has been little explored in the context of web automation. For many years, ML researchers have promoted a "hybrid model" that combines the best of both worlds. As an example, WebQA developed a neurosymbolic system with DSL for extracting desired information from datasets that have similar contents but differ in the underlying structures (e.g. DOM structures) [42]. But it omitted user actions during upstream activities (e.g., data collection), limiting it to tasks involving a particular dataset (e.g., data extraction). Additionally, while these systems are capable of conditional behavior, their program logic is still limited to binary decisions between action and no action, unable to handle diverse specifications with different corresponding actions. Therefore, current neurosymbolic approaches are either restricted to uniform program logic tasks or require the development of a DSL to encode neural model output into symbolic systems (or vice versa), making these approaches not scalable.

2.4 DiLogics: Web Automation with Diverse Program Logics

To this, the author proposed DiLogics [8], an AI-infused PBD system built upon a program synthesizer [20] that assists non-experts in creating web automation programs with diverse and generalizable

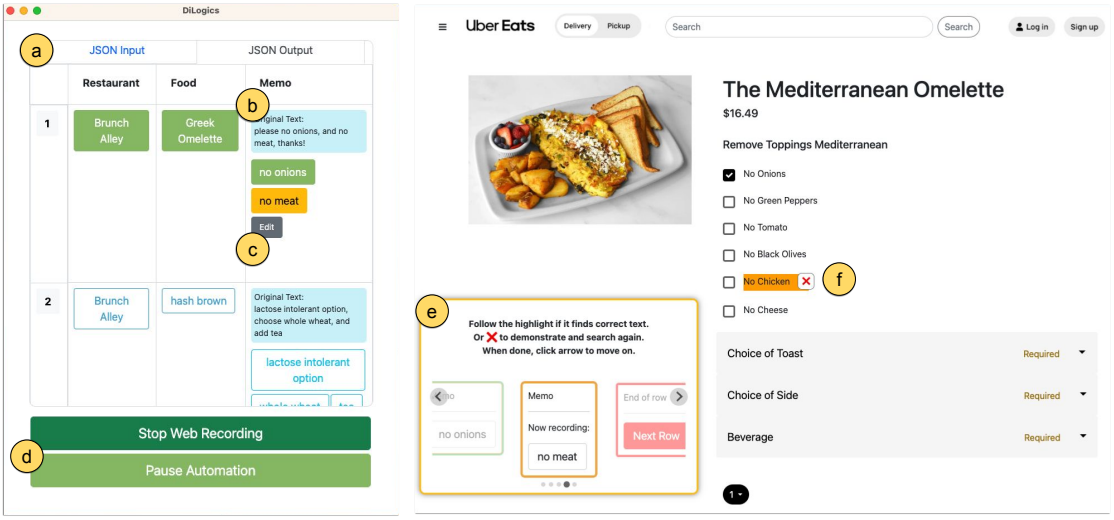

Figure 2. DiLogics Workflow. Left is DiLogics' UI. Users upload input file ⓐ, and data is semantically segmented into task steps ⓑ and rendered into a table where users can modify ⓒ. They can also control the flow of the task, and pause automation ⓓ. Right is the target website, with a carousel displaying the progress of the current iteration ⓔ. Steps are color-coded for their completion status. Users follow the carousel to demonstrate. DiLogics semantically searches web page and highlight relevant elements. If the highlight is incorrect, users can cancel it ⓕ, then edit the step ⓒ or navigate the page to reveal relevant content (e.g. expand the drop-down menu). After demonstrating the current step, users can advance to the next slide on the carousel.

programming logics. DiLogics creates scalable automation programs by semantically segmenting input data into tractable steps and representing them in a table with a carousel widget to track task progress (Fig. 2). Web demonstrations are elicited for each step, mapping UI actions to step descriptions. During each step, DiLogics leverages language models to scrape relevant web content and dynamically associates UI actions with semantically similar elements. Program synthesis techniques are employed to record user actions and their symbolic relationships in the web DOM structure, enabling the system to generate automation programs based on action patterns.

As users demonstrate each step, DiLogics builds a catalog of task steps and their corresponding UI sequence mappings. During automation, the system matches encountered task steps to similar ones in the catalog, extending the program logic to fulfill the current condition. When a new step is not similar to any previous ones, the user is prompted to demonstrate new UI actions, adding the step to the catalog for future generalization. This approach ensures flexibility in executing the automation program by employing the most fitting program logic based on semantic similarity and performing actions on the relevant elements of the current page. Combining language models and program synthesis, DiLogics generates automation programs that incorporate rule-based structural repetitions and diverse program logics based on different input data semantics.

3 FUTURE STEPS

The high-level end goal of the proposed research is to transform how data science tasks can be accomplished and design novel interactions and systems to easily integrate computational techniques into the workflows of non-expert end users. Building on the existing works and my current progress, the next steps of my research involve two major thrusts: (1) allow users to semantically express their task requirements and map them to the diverse representations of data and affordances to create complex and generalizable RPA applications and (2) design refinement techniques through demonstrating task variations to incorporate implicit human feedback and reconstruct complex decision-making processes for more robust automation.

3.1 Semantic Specification and Complex Data Representation

3.1.1 *Semantic Representation of Data Content*

Although SemanticOn [7] introduced novel ways of expressing user's automation goal using language and content highlighting, the system evaluation revealed efforts in composing and refining precise natural language descriptions that encapsulate the right amount of information to filter data. In addition, while DiLogics [8] allows for semantic UI automation by segmenting and categorizing input data, the method does not work on more complex GUI widgets that do not contain any text information. Thus, there still exists a gap between the user's semantic intention to create automation programs and the means to realize that effort.

To address this, the platform that holds the data (i.e. web page) needs to represent its information in a unified way that can be semantically related and compared. Past works have explored annotating and summarizing web and mobile user interfaces to build large datasets of pages/screens with content semantics [44], [45]. Vision- and code-based approaches have been proposed for adding semantic and functional annotations to the elements in a mobile UI [11], [12]. The semantifying-UI method can be adopted in the web environment, where content meaning and functional descriptions are encoded based on the web page and used for RPA purposes. The author plans on utilizing these datasets in conjunction with the semantic understanding in LLMs and VLMs to create an RPA system specialized and fine-tuned for parsing the target web pages using the text content and visual screen layout. This way, the system can semantically organize the information to be automated and the web elements to interact with, achieving automation for more complex requests that often require human-level understanding (e.g. select the window seat on a plane booking website, plan meeting availability on a calendar website). In tasks across multiple web pages or states, the semantic representation could be updated as automation executes. In complement, the task specification and completion status can be represented in the same space.

3.1.2 *Semantic Expression of User Intent*

In semantic RPA, different interaction designs can be utilized to further lower user effort, for example by reducing initial demonstration. Previous PBD automation tools often require a small set of initial user demonstrations to create a baseline for program creation. However, this can be circumvented if the system is built upon existing UI datasets and has the semantic inference ability to translate the user's intent in natural language to action steps in the automation. Then, following the design of SemanticOn and DiLogics, users can have a stage of verification and refinement to collaborate with the system to achieve the intended automation program. The semantic processing based on large datasets of UI screens and descriptions, or even existing automation tasks, enables users to express their intents at a high abstraction level, without the need to be meticulous in defining the task programmatically or demonstrating the actions.

3.2 Implicit User-Intent Construction Through Task Variation Demonstration

With the growing understanding of GUI, the challenge of web automation lies beyond simply streamlining repetitive tasks but in fact in accommodating the nuanced decision-making processes inherent in user preferences. Traditional automation tools often struggle when faced with intricate user decisions. For example, a user might express their preferred selection of a window seat on a plane booking website, but when found out the only window seat left costs an extra $20, the user may decide to make a compromise. An automation system might not be able to capture the full decision-making process in the user's mind and replicate the same rationale in its execution. In these situations, users could choose to update their specifications to account for edge scenarios, but the intricacy of decision-making can introduce conflicting factors. Moreover, users may not always have a clear articulation of their preferences, even if being prompted by the system. This complexity necessitates the need for better refinement techniques that leverage implicit, rather than explicitly acquired, human feedback.

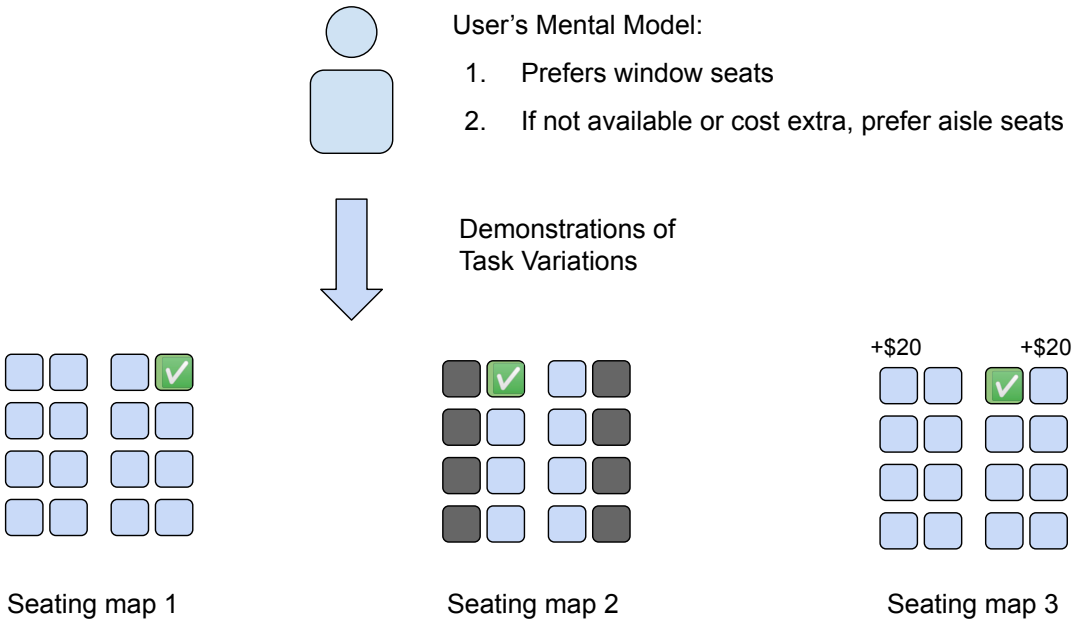

Figure 3. Task Variation Demonstration Example. The user prefers window seats in general, but when not available or costly would switch to aisle seats. To extract this decision-making process and encode it in the specification of automation, the system prompts the user to demonstrate seat selection on various seat maps to implicitly construct the user's mental model.

This research proposes a design that elicits intents through implicit demonstrations of sub-task variations. When users encounter scenarios that call for automation refinement, like the flight booking one illustrated above, the automation system can subtly adjust the current task goal and prompt users to demonstrate how the automation program should execute in different branched scenarios. For

example, the user can be shown hypothetical seating map availabilities for the same flight and asked to demonstrate how they would make the selection in each scenario (Fig.3). The system can then infer by the user's preference that they prefer window seats, but when it's not available or costs extra, they would prefer aisle seats. Through these demonstrations, the system aims to discern the user's decision-making process and extract additional specifications necessary to reconstruct similar decisions under varying conditions. This approach emphasizes the incorporation of implicit human feedback to enhance the adaptability and robustness of automation processes, ensuring that the system can better understand and accommodate intricate user preferences in web automation tasks.

4 CONCLUSION

This paper summarizes recent progress in RPA and web automation through neurosymbolic systems. The research aims to shed light on how we can design neurosymbolic AI systems that support non-expert users in conducting data science tasks that require automation. The paper covers the initial progress in creating novel interactions to allow users to specify their program requirements at a semantic level and in expanding the capability of automation systems to infer new task steps based on the semantics of established program logics. Over the next few years, this research strives to further lower users' effort in adopting RPA in their tasks and expand the generalizability of such systems by leveraging existing semantic UI datasets and LLMs and VLMs. In addition, the paper proposes a design approach to elicit user's implicit feedback to refine automation. By providing users with subtle variations of the task at hand, the envisioned system infers user's decision-making model and modify the program specification to account for complex automation scenarios.

References

[1] G. Leshed, E. M. Haber, T. Matthews, and T. Lau, "Coscripter: Automating & sharing how-to knowledge in the enterprise," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 1719–1728.

[2] M. H. Fischer, G. Campagna, E. Choi, and M. S. Lam, "Diy assistant: A multi-modal end-user programmable virtual assistant," in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 312–327.

[3] T. J.-J. Li, A. Azaria, and B. A. Myers, "Sugilite: Creating multimodal smartphone automation by demonstration," in *Proceedings of the 2017 CHI conference on human factors in computing systems*, 2017, pp. 6038–6049.

[4] I. Drosos, T. Barik, P. J. Guo, R. DeLine, and S. Gulwani, "Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–12.

[5] K. Ferdowsifard, A. Ordookhanians, H. Peleg, S. Lerner, and N. Polikarpova, "Small-step live programming by example," in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 2020, pp. 614–626.

[6] A. R. Sereshkeh, G. Leung, K. Perumal, *et al.*, "Vasta: A vision and language-assisted smartphone task automation system," in *Proceedings of the 25th international conference on intelligent user interfaces*, 2020, pp. 22–32.

[7] K. Pu, R. Fu, R. Dong, X. Wang, Y. Chen, and T. Grossman, "Semanticon: Specifying content-based semantic conditions for web automation programs," in *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '22, Bend, OR, USA: Association for Computing Machinery, 2022, ISBN: 9781450393201. DOI: 10.1145/3526113.3545691. [Online]. Available: https://doi.org/10.1145/3526113.3545691.

[8] K. Pu, J. Yang, A. Yuan, *et al.*, "Dilogics: Creating web automation programs with diverse logics," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '23, San Francisco, CA, USA: Association for Computing Machinery, 2023, ISBN: 9798400701320. DOI: 10.1145/3586183.3606822. [Online]. Available: https://doi.org/10.1145/3586183.3606822.

[9] S. E. Chasins, M. Mueller, and R. Bodik, "Rousillon: Scraping distributed hierarchical web data," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 963–975.

[10] A. Yan, Z. Yang, W. Zhu, *et al.*, *Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation*, 2023. arXiv: 2311.07562 [cs.CV].

[11] J. Wu, R. Krosnick, E. Schoop, A. Swearngin, J. P. Bigham, and J. Nichols, "Never-ending learning of user interfaces," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '23, San Francisco, CA, USA: Association for Computing Machinery, 2023, ISBN: 9798400701320. DOI: 10.1145/3586183.3606824. [Online]. Available: https://doi.org/10.1145/3586183.3606824.

[12] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, "Learning design semantics for mobile apps," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '18, Berlin, Germany: Association for Computing Machinery, 2018, pp. 569–579, ISBN: 9781450359481. DOI: 10.1145/3242587.3242650. [Online]. Available: https://doi.org/10.1145/3242587.3242650.

[13] *Selenium*, Accessed: July, 2023, 2023. [Online]. Available: https://www.selenium.dev/.

[14] *Beautiful soup*, Accessed: July, 2023, 2023. [Online]. Available: http://www.crummy.com/software/BeautifulSoup/.

[15] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using gui screenshots for search and automation," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, 2009, pp. 183–192.

[16] R. Krosnick and S. Oney, "Understanding the challenges and needs of programmers writing web automation scripts," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, IEEE, 2021, pp. 1–9.

[17] T. Lau, S. A. Wolfman, P. Domingos, and D. S. Weld, "Programming by demonstration using version space algebra," *Machine Learning*, vol. 53, no. 1, pp. 111–156, 2003.

[18] D. Kurlander, A. Cypher, and D. C. Halbert, *Watch what I do: programming by demonstration*. MIT press, 1993.

[19] A. Blackwell, "Your wish is my command: Giving users the power to instruct their software, chapter swyn: A visual representation for regular expressions," *M. Kaufmann*, pp. 245–270, 2000.

[20] R. Dong, Z. Huang, I. I. Lam, Y. Chen, and X. Wang, "Webrobot: Web robotic process automation using interactive programming-by-demonstration," *arXiv preprint arXiv:2203.09993*, 2022.

[21] A. F. Blackwell, "Swyn: A visual representation for regular expressions," in *Your wish is my command*, Elsevier, 2001, pp. 245–XIII.

[22] T. A. Lau, P. M. Domingos, and D. S. Weld, "Version space algebra and its application to programming by demonstration.," in *ICML*, Citeseer, 2000, pp. 527–534.

[23] D. H. Mo and I. H. Witten, "Learning text editing tasks from examples: A procedural approach," *Behaviour & Information Technology*, vol. 11, no. 1, pp. 32–45, 1992.

[24] A. J. Werth and B. A. Myers, "Tourmaline (abstract) macrostyles by example," in *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*, 1993, p. 532.

[25] W. Ni, J. Sunshine, V. Le, S. Gulwani, and T. Barik, "Recode: A lightweight find-and-replace interaction in the ide for transforming code by example," in *The 34th Annual ACM Symposium on User Interface Software and Technology*, 2021, pp. 258–269.

[26] D. Kurlander and S. Feiner, "A history-based macro by example system," in *Proceedings of the 5th annual ACM symposium on User interface software and technology*, 1992, pp. 99–106.

[27] H. Lieberman, "A user interface for knowledge acquisition from video," in *AAAI*, Citeseer, 1994, pp. 527–534.

[28] D. L. Maulsby, I. H. Witten, and K. A. Kittlitz, "Metamouse: Specifying graphical procedures by example," *ACM SIGGRAPH Computer Graphics*, vol. 23, no. 3, pp. 127–136, 1989.

[29] F. Modugno and B. A. Myers, "Pursuit: Visual programming in a visual domain," CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1994.

[30] B. A. Myers, D. A. Giuse, R. B. Dannenberg, *et al.*, "Garnet comprehensive support for graphical, highly interactive user interfaces," in *Readings in Human–Computer Interaction*, Elsevier, 1995, pp. 357–371.

[31] J. Nichols and T. Lau, "Mobilization by demonstration: Using traces to re-author existing web sites," in *Proceedings of the 13th international conference on Intelligent user interfaces*, 2008, pp. 149–158.

[32] P. Vaithilingam and P. J. Guo, "Bespoke: Interactively synthesizing custom guis from command-line applications by demonstration," in *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, 2019, pp. 563–576.

[33] J. Lin, J. Wong, J. Nichols, A. Cypher, and T. A. Lau, "End-user programming of mashups with vegemite," in *Proceedings of the 14th international conference on Intelligent user interfaces*, 2009, pp. 97–106.

[34] *Uipath*, Accessed: July, 2023, 2023. [Online]. Available: https://www.uipath.com/.

[35] *Imacro*, Accessed: July, 2023, 2023. [Online]. Available: https://www.progress.com/imacros.

[36] J. Allen, N. Chambers, G. Ferguson, *et al.*, "Plow: A collaborative task learning agent," in *AAAI*, vol. 7, 2007, pp. 1514–1519.

[37] T. J.-J. Li, M. Radensky, J. Jia, K. Singarajah, T. M. Mitchell, and B. A. Myers, "Pumice: A multi-modal agent that learns concepts and conditionals from natural language and demonstrations," in *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, 2019, pp. 577–589.

[38] A. Swearngin, C. Wang, A. Oleson, J. Fogarty, and A. J. Ko, "Scout: Rapid exploration of interface layout alternatives through high-level design constraints," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[39] P. O'Donovan, A. Agarwala, and A. Hertzmann, "Designscape: Design with interactive layout suggestions," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 1221–1224.

[40] N. Zhao, N. W. Kim, L. M. Herman, *et al.*, "Iconate: Automatic compound icon generation and ideation," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[41] T. J.-J. Li, I. Labutov, X. N. Li, *et al.*, "Appinite: A multi-modal interface for specifying data descriptions in programming by demonstration using natural language instructions," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2018, pp. 105–114. DOI: 10.1109/VLHCC.2018.8506506.

[42] Q. Chen, A. Lamoreaux, X. Wang, G. Durrett, O. Bastani, and I. Dillig, "Web question answering with neurosymbolic program synthesis," in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 328–343.

[43] E. Parisotto, A.-r. Mohamed, R. Singh, L. Li, D. Zhou, and P. Kohli, "Neuro-symbolic program synthesis," *arXiv preprint arXiv:1611.01855*, 2016.

[44] B. Wang, G. Li, and Y. Li, "Enabling conversational interaction with mobile ui using large language models," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI)*, New York, NY, USA: Association for Computing Machinery, 2023.

[45] J. Wu, S. Wang, S. Shen, Y.-H. Peng, J. Nichols, and J. P. Bigham, "Webui: A dataset for enhancing visual ui understanding with web semantics," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Hamburg, Germany: Association for Computing Machinery, 2023, ISBN: 9781450394215. DOI: 10.1145/3544548.3581158. [Online]. Available: https://doi.org/10.1145/3544548.3581158.