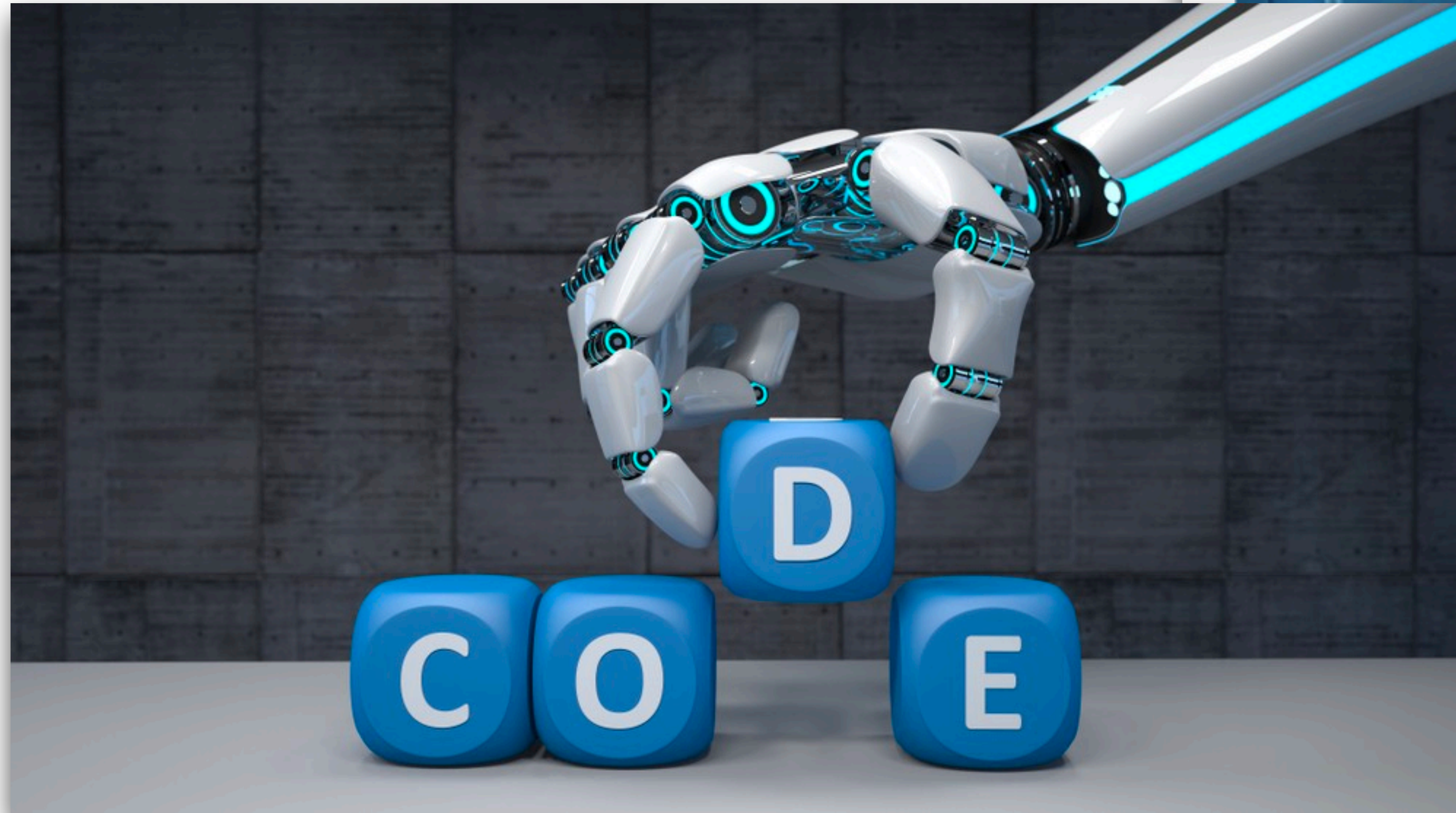


Program Synthesis (Part 2)



July 8, 2019

Program Synthesis Moves a Step Closer to Reality

George Leopold



As data scientists and software developers sort through the plethora of tools and APIs ranging from Python to Apache Spark, automation schemes are emerging to help programmers navigate those tools and the accompanying

Two Days Ago...

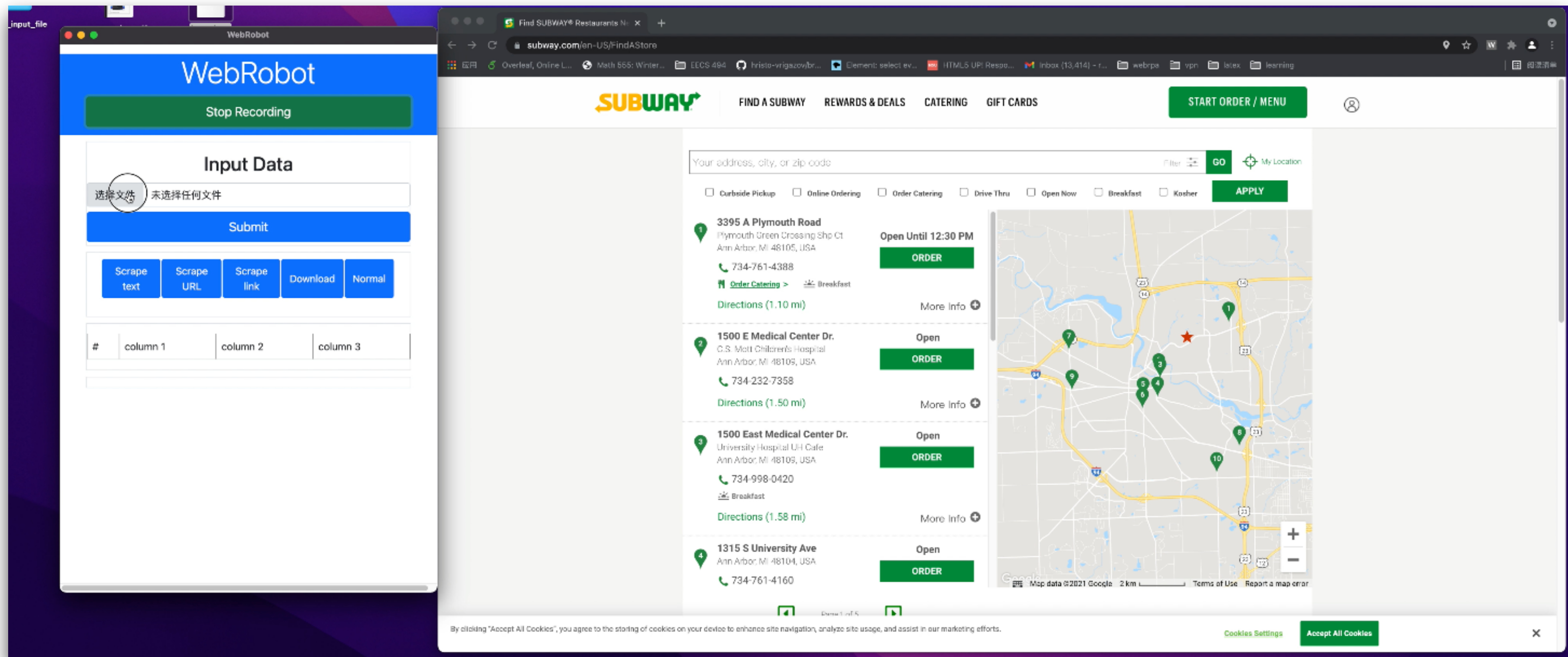
- FlashFill: generate string transformations from examples



	A	B	C
1	DEC	December	
2	NOV	November	
3	OCT	October	
4	APR	Aprember	
5	AUG	Augember	
6	FEB	Febember	
7	JAN	Janember	
8	JUL	Julember	
9	JUN	Junember	
10	MAR	Marember	
11	MAY	Mayember	
12	SEP	Sepember	
13			

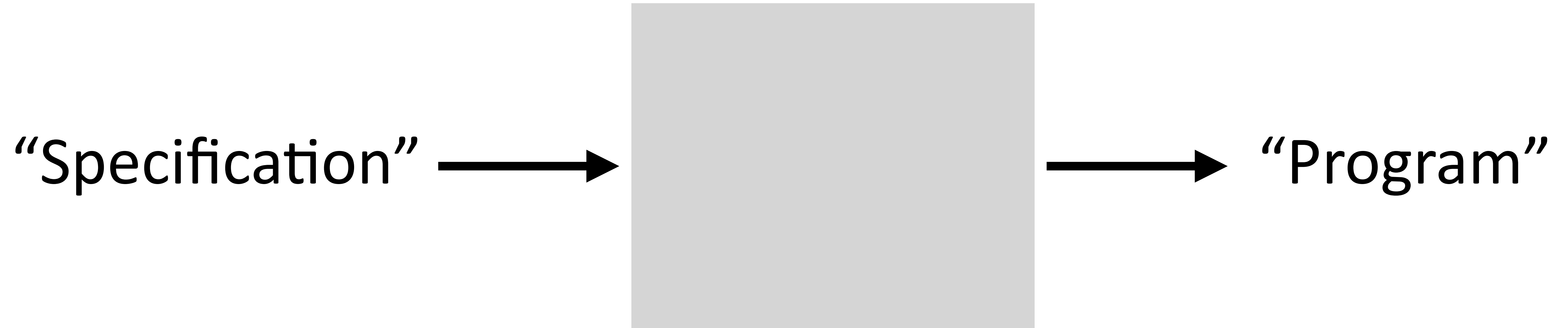
Two Days Ago...

- WebRobot: generate web automation programs by “watching” what you do



Two Days Ago...

- Program synthesis

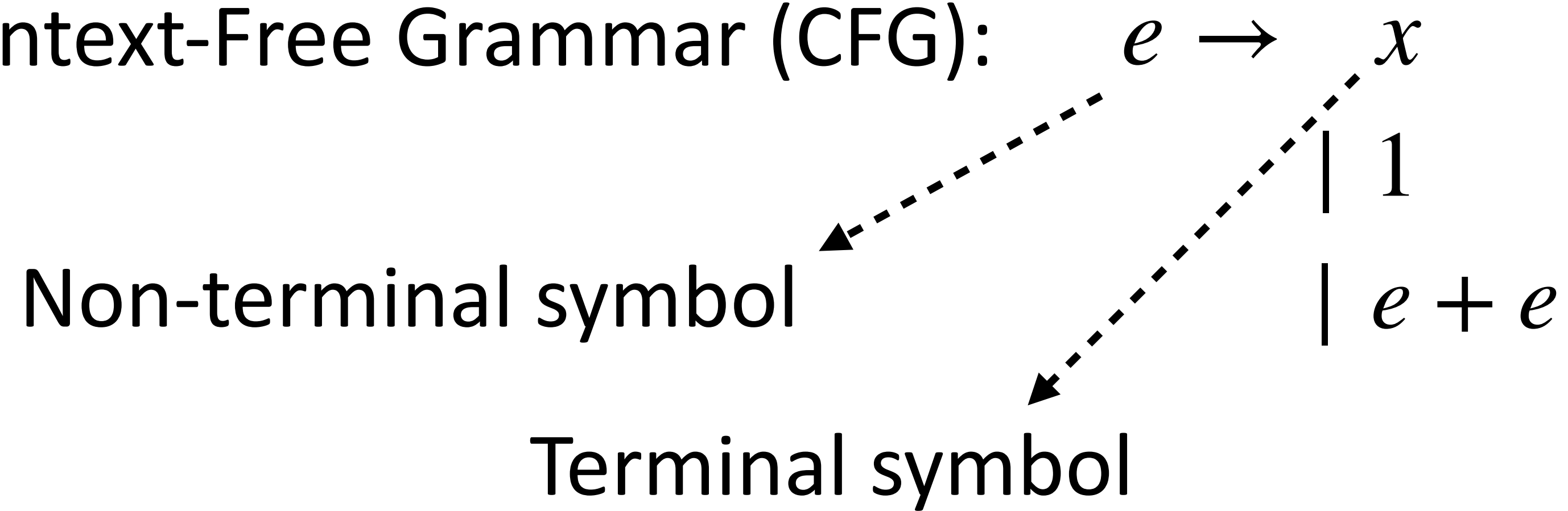


Two Days Ago...

- How to synthesize programs from **input-output examples**
 - By **systematically** searching within a **context-free grammar**
 - Check if program satisfies examples
 - One approach: top-down search algorithm

Two Days Ago...

- Context-Free Grammar (CFG):

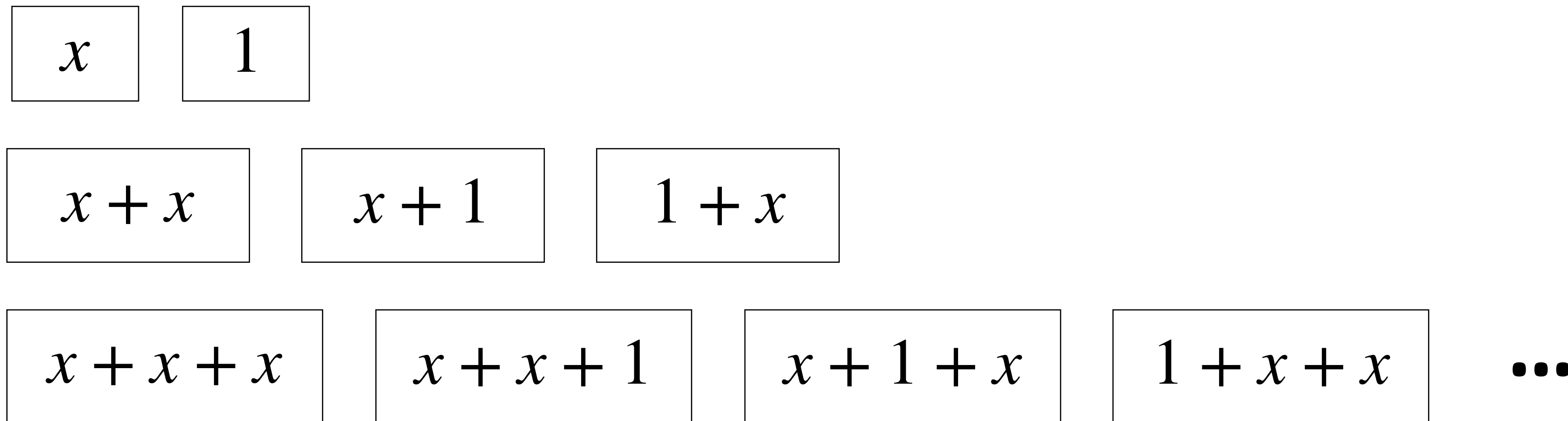


“All programs that can use x , 1 and $+$.”

Two Days Ago...

- Context-Free Grammar (CFG):
$$e \rightarrow \begin{array}{l} x \\ | 1 \\ | e + e \end{array}$$
“All programs that can use x , 1 and $+$.”

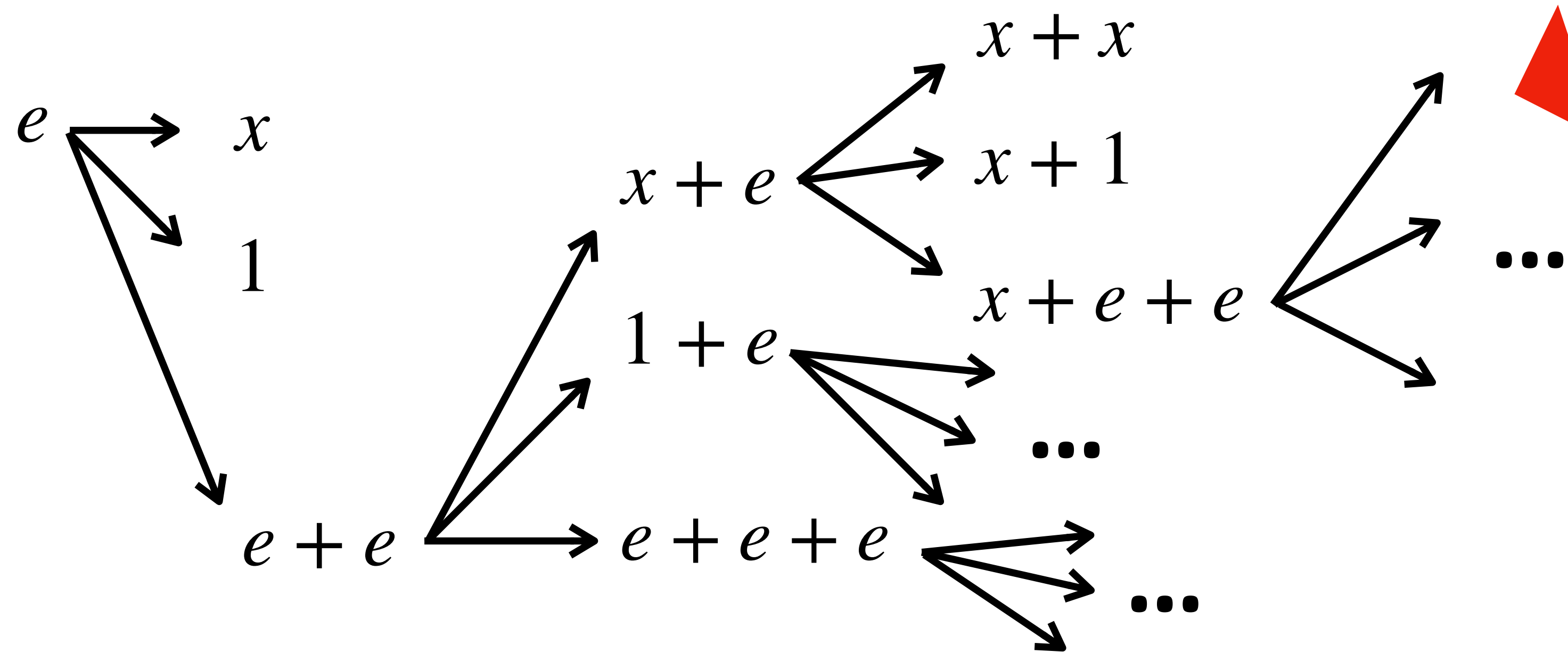
- This CFG defines a set of programs



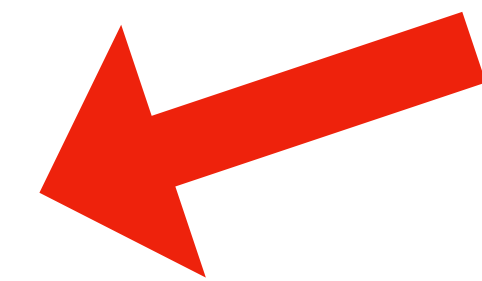
- Goal: find a program in this set that satisfies a given example

Two Days Ago...

- How top-down search works...



This is a “search tree”



Today

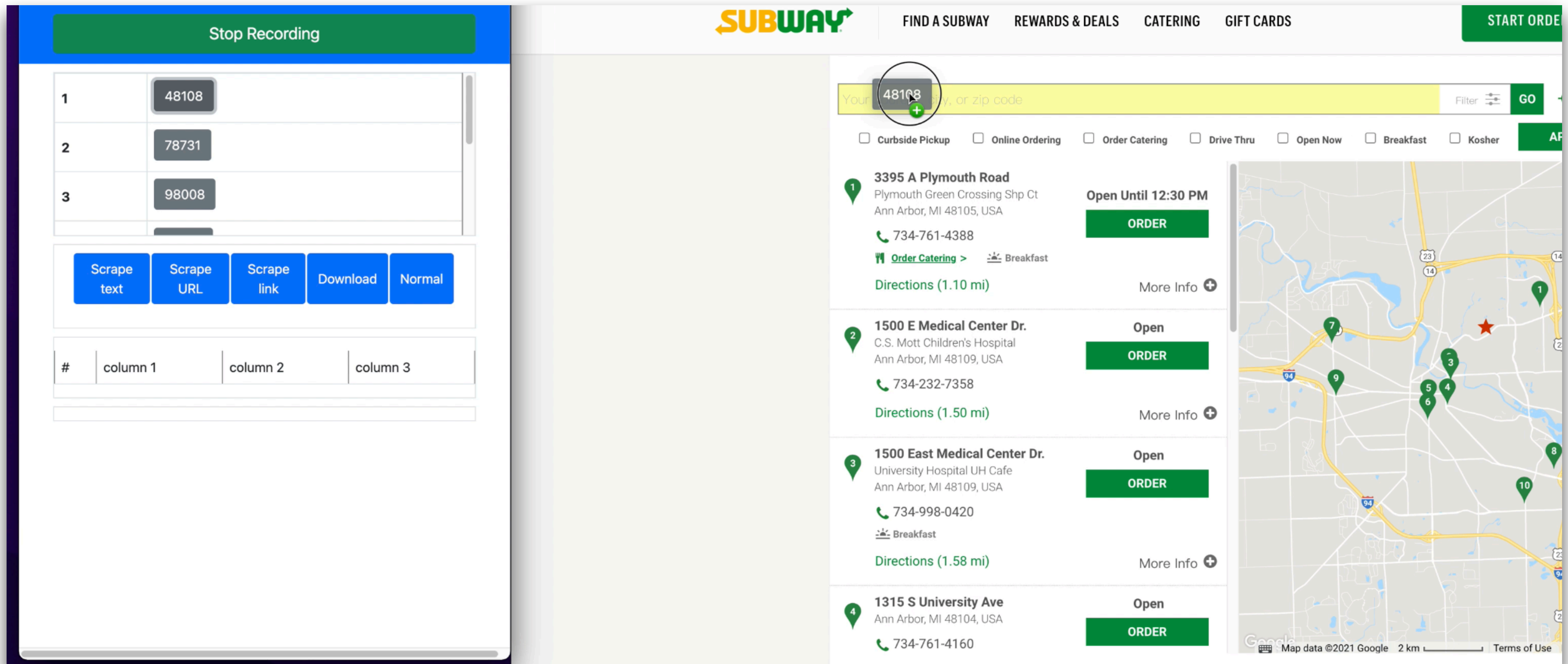
- Programming-by-Example -> Programming-by-Demonstration
 - PBE: specification is **input-output example**
 - PBD: specification is **execution trace**
 - “Trace” in this lecture: a sequence of instructions executed
 - But “demonstration” can be interpreted more broadly..
- Use WebRobot to illustrate how PBD works
- Some recent development of WebRobot
 - Demos/figures..

WebRobot demo (one more time!)

- Let's re-watch the WebRobot demo..
 - Pay attention to the demonstration

Demonstration (Action Trace) in WebRobot

- Action 1: drag-and-drop.
- Recorded action: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])



The image shows a side-by-side comparison of a WebRobot interface and a Subway website. On the left, the WebRobot interface displays a 'Stop Recording' button at the top. Below it is a table with three rows of recorded actions:

1	48108
2	78731
3	98008

Below the table are five buttons: 'Scrape text', 'Scrape URL', 'Scrape link', 'Download', and 'Normal'. At the bottom, there is a table with columns labeled '#', 'column 1', 'column 2', and 'column 3'.

On the right, the Subway website is shown. The search bar contains the zip code '48108', which is circled in red. The website displays a list of Subway locations with details such as address, phone number, and an 'ORDER' button. A map on the right side of the website shows the location of the selected store marked with a red star.

Demonstration (Action Trace) in WebRobot

- Action 1: drag-and-drop.
- Recorded action:

CP(input[1], /html/body/main/div[3]/section/div/div/div[1])



Selector: locates an element on the webpage

Webpage ≈ (DOM) Tree

STORE LOCATION

PICKUP DELIVERY

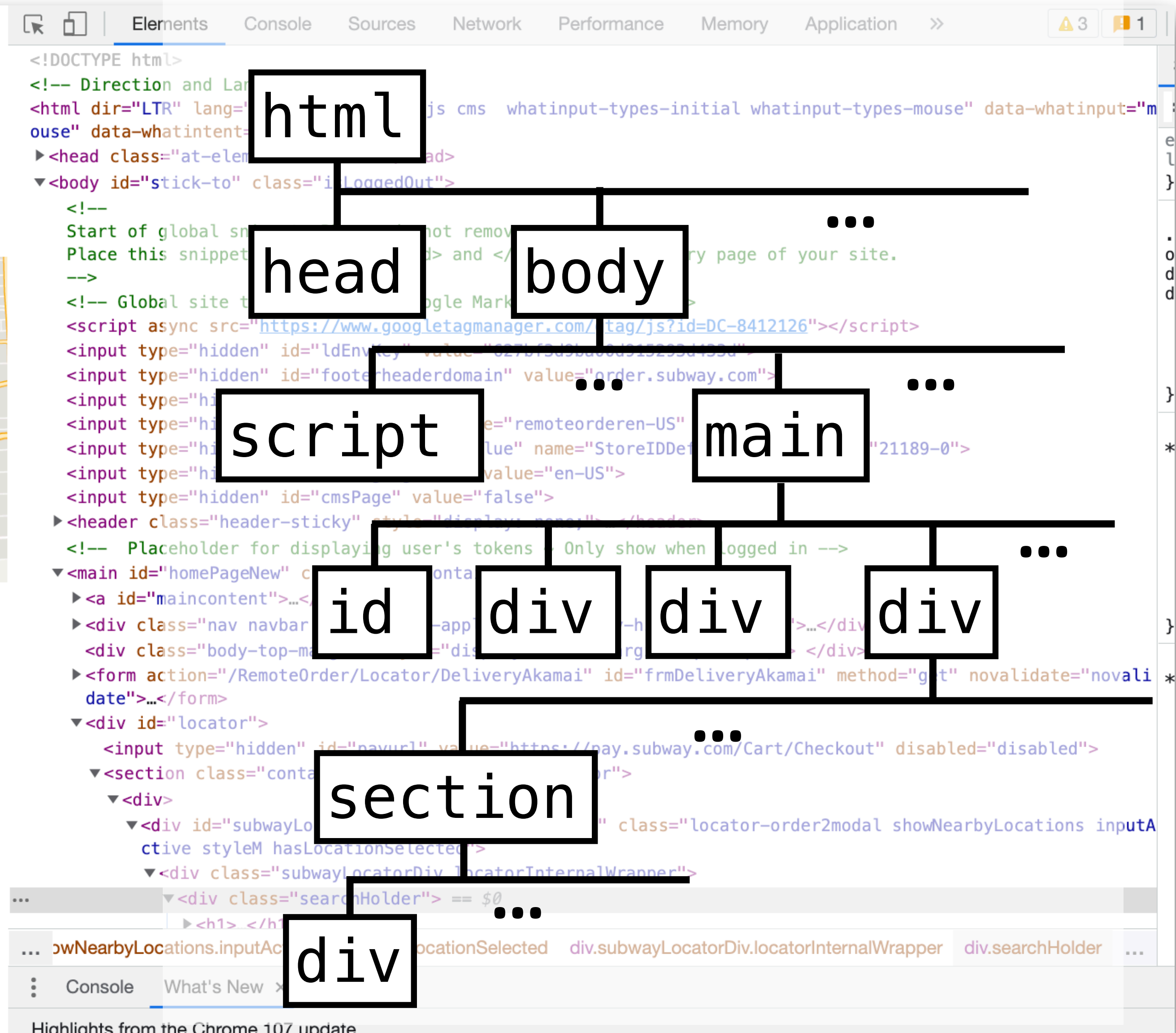
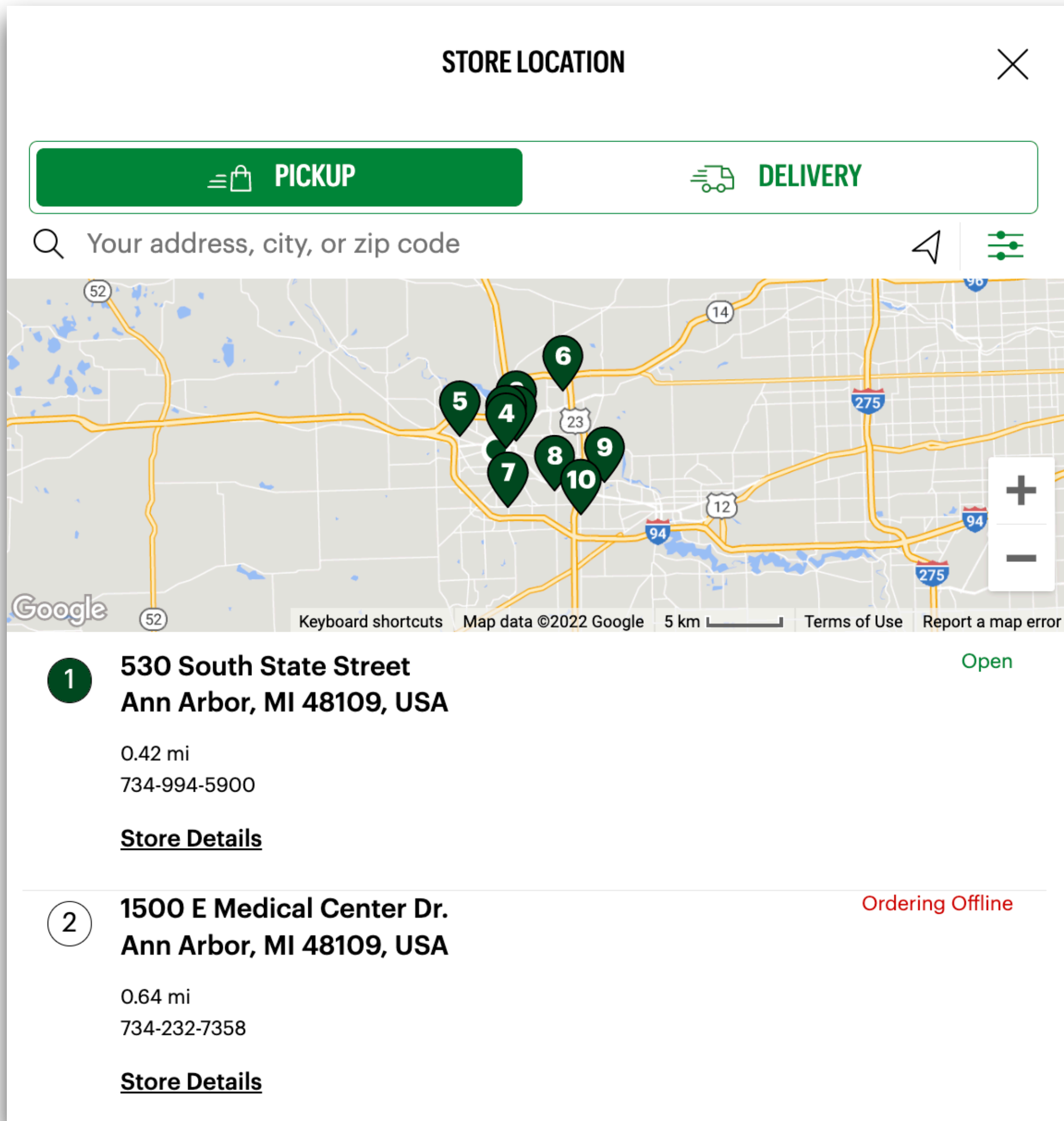
Your address, city, or zip code

1 530 South State Street
Ann Arbor, MI 48109, USA
0.42 mi
734-994-5900
Store Details

2 1500 E Medical Center Dr.
Ann Arbor, MI 48109, USA
0.64 mi
734-232-7358
Store Details Ordering Offline

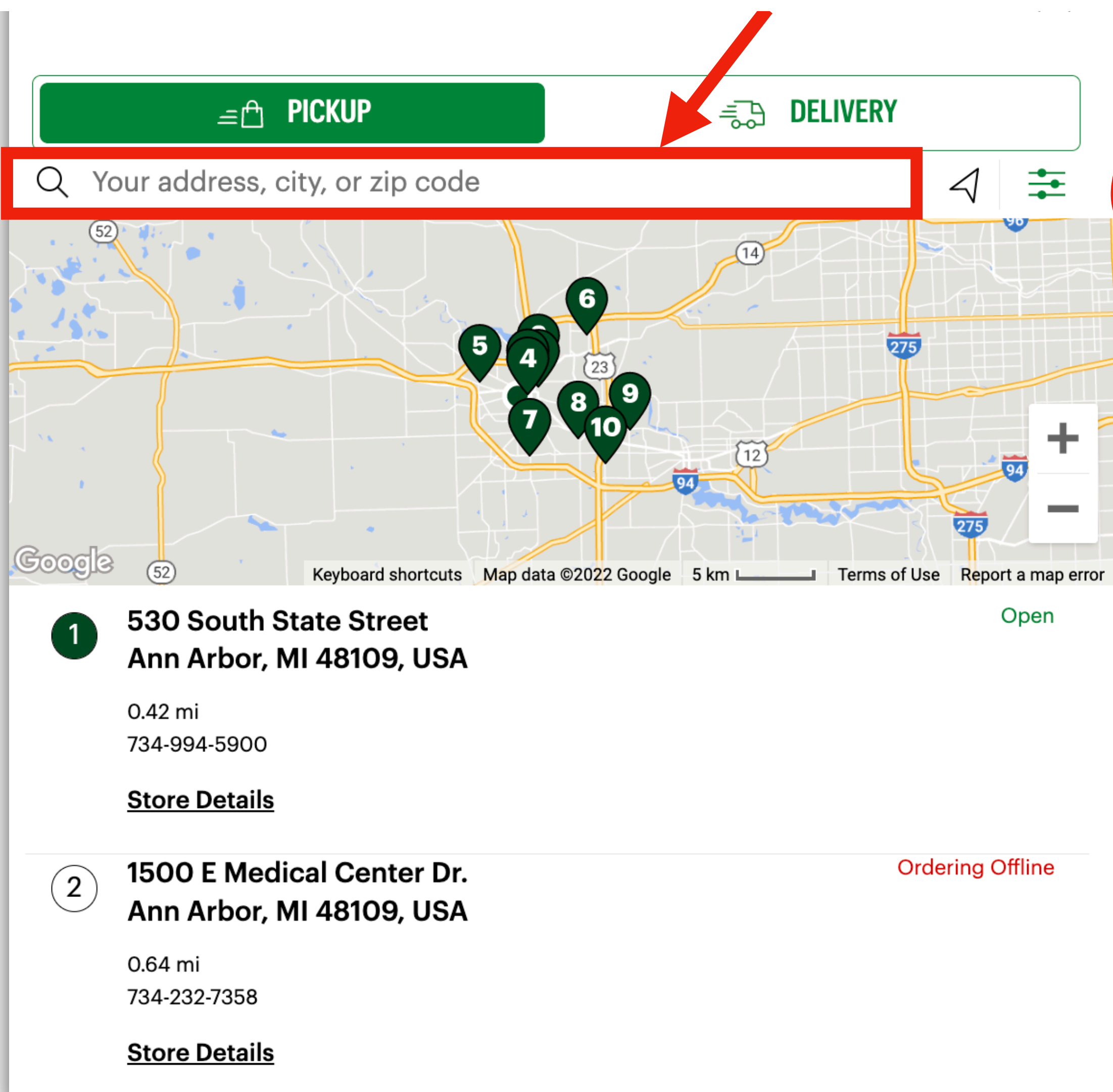
```
<!DOCTYPE html>
<!-- Direction and Language -->
<html dir="LTR" lang="en-US" class="no-js cms whatinput-types-initial whatinput-types-mouse" data-whatinput="mouse" data-whatintent="mouse">
  <head class="at-element-marker">...</head>
  <body id="stick-to" class="isLoggedOut">
    <!--
    Start of global snippet: Please do not remove
    Place this snippet between the <head> and </head> tags on every page of your site.
    -->
    <!-- Global site tag (gtag.js) - Google Marketing Platform -->
    <script async src="https://www.googletagmanager.com/gtag/js?id=DC-8412126"></script>
    <input type="hidden" id="ldEnvKey" value="627bf3d9ba00d915293d433d">
    <input type="hidden" id="footerheaderdomain" value="order.subway.com">
    <input type="hidden" id="isUserLoggedIn">
    <input type="hidden" name="sitename" value="remoteorderen-US" id="hdnSiteName">
    <input type="hidden" id="StoreIDDefaultValue" name="StoreIDDefaultValue" value="21189-0">
    <input type="hidden" id="scLanguageCode" value="en-US">
    <input type="hidden" id="cmsPage" value="false">
    <header class="header-sticky" style="display: none;">...</header>
    <!-- Placeholder for displaying user's tokens ~ Only show when logged in -->
    <main id="homePageNew" class="body_container">
      <a id="maincontent">...</a>
      <div class="nav navbar navbar-deal-applied deal-apply-hide nav-shadow">...</div>
      <div class="body-top-margin" style="display: none; margin-top: 38px;"> </div>
      <form action="/RemoteOrder/Locator/DeliveryAkamai" id="frmDeliveryAkamai" method="get" novalidate="novalidate">...</form>
      <div id="locator">
        <input type="hidden" id="payurl" value="https://pay.subway.com/Cart/Checkout" disabled="disabled">
        <section class="container" data-component="locator">
          <div>
            <div id="subwayLocator" style="direction:ltr;" class="locator-order2modal showNearbyLocations inputActive styleM hasLocationSelected">
              <div class="subwayLocatorDiv locatorInternalWrapper">
                <div class="searchHolder"> == $0
                <h1> </h1>
              </div>
            </div>
          </div>
        </section>
      </div>
    </main>
  </body>
</html>
```

Webpage ≈ (DOM) Tree



DOM Selectors

/html/body/main/div[3]/section/div/div/div[1]

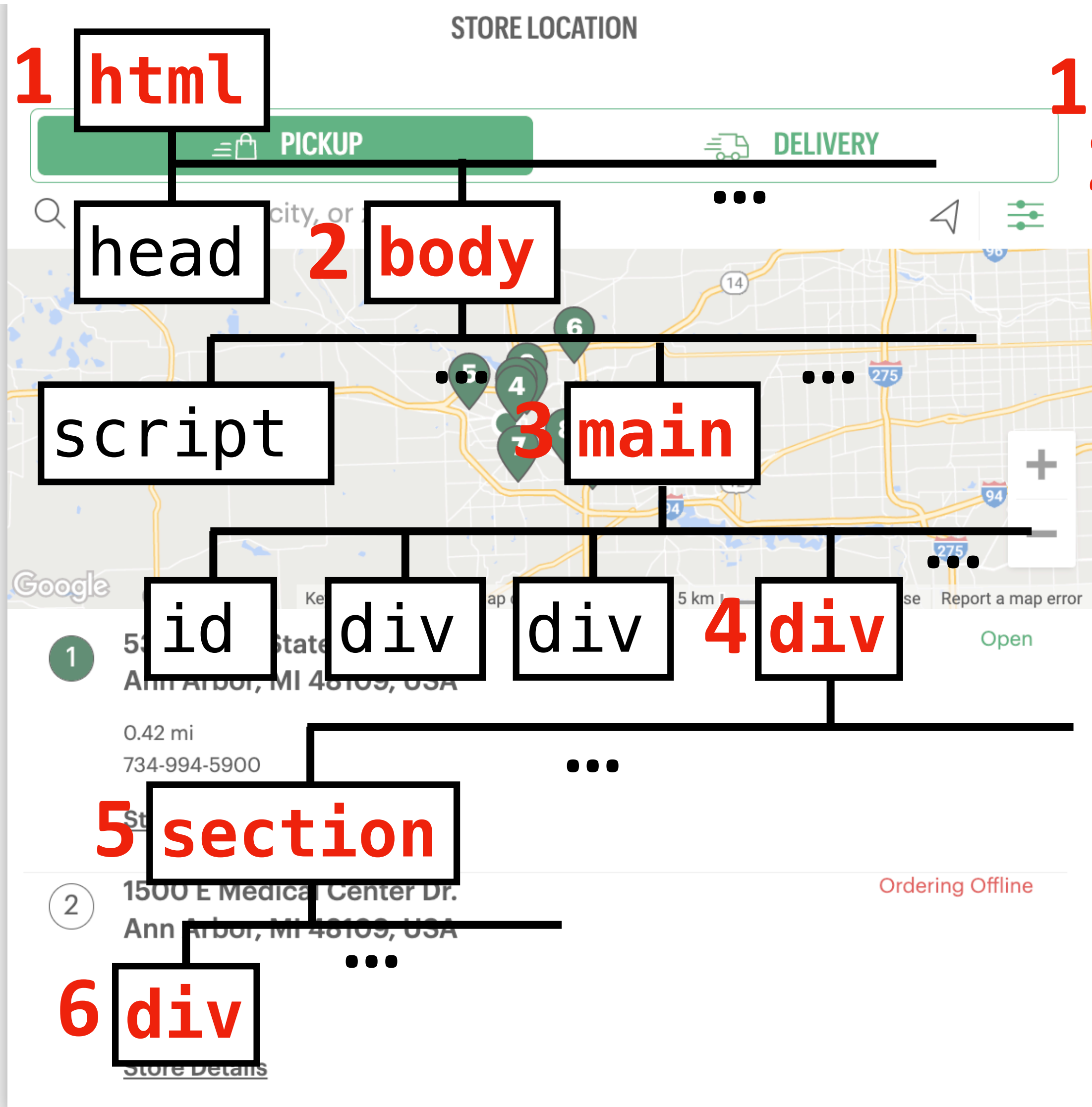


```
Performance Memory Application >> 3 1
<!-- Direction and Language -->
<html dir="LTR" lang="en-US" class="no-js cms whatinput-types-initial whatinput-types-mouse" data-whatinput="mouse" data-whatintent="mouse">
<head class="at-element-marker">...</head>
<body id="stick-to" class="isLoggedOut">
<!--
Start of global snippet: Please do not remove
Place this snippet between the <head> and </head> tags on every page of your site.
-->
<!-- Global site tag (gtag.js) - Google Marketing Platform -->
<script async src="https://www.googletagmanager.com/gtag/js?id=DC-8412126"></script>
<input type="hidden" id="ldEnvKey" value="627bf3d9ba00d915293d433d">
<input type="hidden" id="footerheaderdomain" value="order.subway.com">
<input type="hidden" id="isUserLoggedIn">
<input type="hidden" name="sitename" value="remoteorderen-US" id="hdnSiteName">
<input type="hidden" id="StoreIDDefaultValue" name="StoreIDDefaultValue" value="21189-0">
<input type="hidden" id="scLanguageCode" value="en-US">
<input type="hidden" id="cmsPage" value="false">
<header class="header-sticky" style="display: none;">...</header>
<!-- Placeholder for displaying user's tokens ~ Only show when logged in -->
<main id="homePageNew" class="body_container">
  <a id="maincontent">...</a>
  <div class="nav navbar navbar-deal-applied deal-apply-hide nav-shadow">...</div>
  <div class="body-top-margin" style="display: none; margin-top: 38px;"> </div>
  <form action="/RemoteOrder/Locator/DeliveryAkamai" id="frmDeliveryAkamai" method="get" novalidate="novalidate">...</form>
  <div id="locator">
    <input type="hidden" id="payurl" value="https://pay.subway.com/Cart/Checkout" disabled="disabled">
    <section class="container" data-component="locator">
      <div>
        <div id="subwayLocator" style="direction:ltr;" class="locator-order2modal showNearbyLocations inputActive styleM hasLocationSelected">
          <div class="subwayLocatorDiv locatorInternalWrapper">
            <div class="searchHolder"> == $0
              <h1> </h1>
            </div>
          </div>
        </div>
      </div>
    </section>
  </div>
  <div class="showNearbyLocations.inputActive.styleM.hasLocationSelected div.subwayLocatorDiv.locatorInternalWrapper div.searchHolder">
  </div>
</main>

```

DOM Selectors

`/html/body/main/div[3]/section/div/div/div[1]`

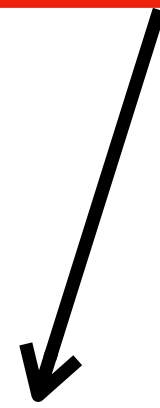


A screenshot of a browser's developer tools showing the DOM tree for the same page. The tree structure is visible on the left, with the selected element highlighted. The corresponding HTML code is shown on the right. Red boxes and numbers (1-6) are placed over the code to match the diagram on the left. A red arrow points from the `div` element in the diagram to the corresponding `div` element in the code. The code includes the `html` root, `body`, `main`, and several nested `div` elements, ending with the selected `div` element.

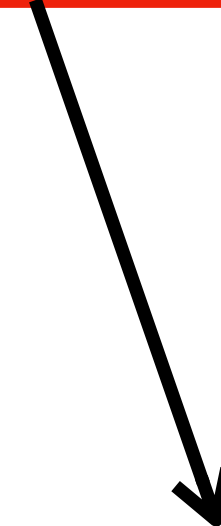
Demonstration (Action Trace) in WebRobot

- Action 1: drag-and-drop.
- Recorded action:

CP(input[1], /html/body/main/div[3]/section/div/div/div[1])



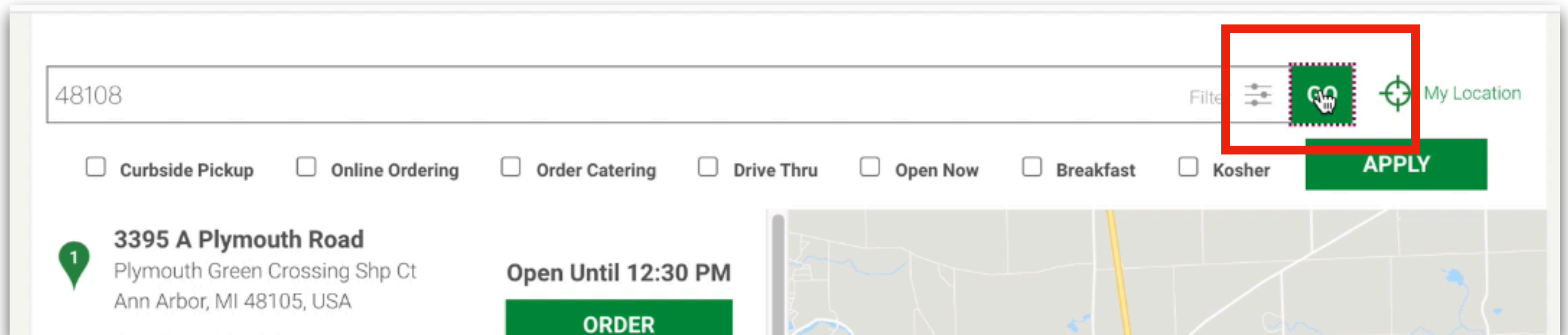
Locates data from input



Selector: locates an element on the webpage

Demonstration (Action Trace) in WebRobot

- Action 1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
- **Action 2: Click(/html/body/main/div[3]/section/.../button[2])**



Demonstration (Action Trace) in WebRobot

- Action 2: Click(/html/body/main/div[3]/section/.../button[2])
- **Action 3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)**

The screenshot displays the WebRobot interface on the left and a Subway website on the right. The WebRobot interface includes a 'Stop Recording' button at the top. Below it is a table with three rows of data:

1	48108
2	78731
3	98008

Below the table are several buttons: 'Scrape text' (highlighted with a red box), 'Scrape URL', 'Scrape link', 'Download', and 'Normal'. Below these buttons is a text input field with the text '3650 South State Street' (also highlighted with a red box). At the bottom of the WebRobot interface is another table with four columns: '#', 'column 1', 'column 2', and 'column 3'. The first row contains the value '1' in the first column and '3650 South State Street' in the second column (highlighted with a red box).

The Subway website on the right shows a search for '48108'. The results list two locations. The first location, '3650 South State Street', is highlighted with a red box. It includes the address '3650 South State Street', 'Citgo Gas/C-Store', and a phone number '734-665-7277'. The second location, '3098 Platt Rd', includes the address '3098 Platt Rd', 'Ann Arbor, MI 48108, USA', and a phone number '734-975-8860'.

Demonstration (Action Trace) in WebRobot

- Action 3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
- **Action 4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])**

The screenshot displays the WebRobot interface during a demonstration. On the left, the 'Action Trace' table shows the following data:

#	column 1	column 2	column 3
1	48108		
2	78731		
3	98008		

Below the table, the 'Scrape text' button is highlighted with a red box. The 'Select an element on the web page' section shows a table with the following data:

#	column 1	column 2	column 3
1	3650 South State Street	Ann Arbor, MI 48108, USA	

On the right, the web page shows a Subway store listing. The address 'Ann Arbor, MI 48108, USA' is highlighted in yellow and enclosed in a red box. The store name is '3650 South State Street' and the status is 'Open'. The 'ORDER' button is visible next to the address.

Demonstration (Action Trace) in WebRobot

- Action 4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
- **Action 5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)**

The screenshot shows the WebRobot interface on the left and a Subway website on the right. The WebRobot interface includes a 'Stop Recording' button, a table of recorded actions, and a table of scraped data. The 'Scrape text' button is highlighted with a red box. The scraped data table has a red box around the phone number '734-665-7277'. The Subway website shows a search for '48108' with a list of locations. The phone number '734-665-7277' is highlighted with a red box on the website.

#	column 1	column 2	column 3
1	3650 South State Street	Ann Arbor, MI 48108, USA	734-665-7277

#	column 1	column 2	column 3
1	48108		
2	78731		
3	98008		

Demonstration (Action Trace) in WebRobot

- Action 5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
- **Action 6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)**

The screenshot displays the WebRobot interface on the left and a Subway website on the right. The WebRobot interface includes a 'Stop Recording' button, a table of recorded actions, a toolbar with 'Scrape text' highlighted, and a table of scraped data. The Subway website shows search results for ZIP code 48108, with the address '3098 Platt Rd' highlighted in yellow and a red box around it.

#	column 1	column 2	column 3
1	3650 South State Street	Ann Arbor, MI 48108, USA	734-665-7277
2	3098 Platt Rd		

#	column 1	column 2	column 3
1	48108		
2	78731		
3	98008		

Demonstration (Action Trace) in WebRobot

- Action 6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)
- **Action 7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])**

Stop Recording

1	48108
2	78731
3	98008

Scrape text | Scrape URL | Scrape link | Download | Normal

Select an element on the web page

#	column 1	column 2	column 3
1	3650 South State Street	Ann Arbor, MI 48108, USA	734-665-7277
2	3098 Platt Rd	Ann Arbor, MI 48108, USA	

SUBWAY

FIND A SUBWAY | REWARDS & DEALS | CATERING

48108

Curbside Pickup Online Ordering Order Catering Drive

3650 South State Street Open
Citgo Gas/C-Store
Ann Arbor, MI 48108, USA ORDER
734-665-7277
Curbside Pickup: 11:00 AM - 7:00 PM
Order Catering >
Directions (1.57 mi) More Info +

3098 Platt Rd Open
734-975-8860 ORDER
Order Catering > Breakfast
Directions (2.69 mi) More Info +

7000 East Michigan Avenue Open
Well Meet #5479

Demonstration (Action Trace) in WebRobot

- Action 7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])
- **Action 8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)**

The screenshot displays the WebRobot interface on the left and the Subway website on the right. The WebRobot interface shows a table with the following data:

#	column 1	column 2	column 3
1	3650 South State Street	Ann Arbor, MI 48108, USA	734-665-7277
2	3098 Platt Rd	Ann Arbor, MI 48108, USA	734-975-8860

The 'Scrape text' button in the WebRobot interface is highlighted with a red box. The Subway website shows the phone number 734-975-8860 highlighted with a red box.

Start To See Repetitions?

Repetition
Repetition
Repetition
Repetition

So does WebRobot!

WebRobot Detects Repeating Patterns

- Find the pattern in the trace?

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

2: Click(/html/body/main/div[3]/section/.../button[2])

3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)

4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])

5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)

7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])

8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)



WebRobot Detects Repeating Patterns

- Find the pattern in the trace?

```
1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
2: Click(/html/body/main/div[3]/section/.../button[2])
3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)
7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])
8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
```



How Do We Automate Data Scraping?

- Now we've identified the pattern...
- But how can we automate this task?
 - Write program!
 - ... **based on the identified pattern**

What Should The Program Look Like?

- 1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
- 2: Click(/html/body/main/div[3]/section/.../button[2])
- 3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
- 4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
- 5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
- 6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)
- 7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])
- 8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)



Notecard: what is one program that can automate this task?

What Should The Program Look Like?

- Different ways to write the program...
- What “properties” should these programs have?
- **In terms of what they do, rather than how they look**

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

2: Click(/html/body/main/div[3]/section/.../button[2])

3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)

4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])

5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)

7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])

8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)



Property 1: Reproduce Trace

- When executed, program should **reproduce** actions in trace
 - First action executed by program is first action in trace
 - Same for second, third, ..., last actions

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

2: Click(/html/body/main/div[3]/section/.../button[2])

3: Scrape(/html/body/main/div[3]/section/.../**div[1]**/addr)

4: Scrape(/html/body/main/div[3]/section/.../**div[1]**/.../div[3])

5: Scrape(/html/body/main/div[3]/section/.../**div[1]**/phone)

6: Scrape(/html/body/main/div[3]/section/.../**div[2]**/addr)

7: Scrape(/html/body/main/div[3]/section/.../**div[2]**/.../div[3])

8: Scrape(/html/body/main/div[3]/section/.../**div[2]**/phone)



Property 2: Generalize Trace

- When executed, program should **generalize** the trace
- Reproduce trace + **produce at least one more action**

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
2: Click(/html/body/main/div[3]/section/.../button[2])
3: Scrape(/html/body/main/div[3]/section/.../**div[1]**/addr)
4: Scrape(/html/body/main/div[3]/section/.../**div[1]**/.../div[3])
5: Scrape(/html/body/main/div[3]/section/.../**div[1]**/phone)
6: Scrape(/html/body/main/div[3]/section/.../**div[2]**/addr)
7: Scrape(/html/body/main/div[3]/section/.../**div[2]**/.../div[3])
8: Scrape(/html/body/main/div[3]/section/.../**div[2]**/phone)

9: some action not seen before

Let's Look At Some Programs..

- For each program, ask two questions:

- Does it reproduce trace?
- Does it generalize trace?

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
2: Click(/html/body/main/div[3]/section/.../button[2])
3: Scrape(/html/body/main/div[3]/section/.../**div[1]**/addr)
4: Scrape(/html/body/main/div[3]/section/.../**div[1]**/.../div[3])
5: Scrape(/html/body/main/div[3]/section/.../**div[1]**/phone)
6: Scrape(/html/body/main/div[3]/section/.../**div[2]**/addr)
7: Scrape(/html/body/main/div[3]/section/.../**div[2]**/.../div[3])
8: Scrape(/html/body/main/div[3]/section/.../**div[2]**/phone)

Program 1

- Trace itself

CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

Click(/html/body/main/div[3]/section/.../button[2])

Scrape(/html/body/main/div[3]/section/.../div[1]/addr)

Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

Scrape(/html/body/main/div[3]/section/.../div[2]/addr)

Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[2]/phone)

Program 2

- Trace + one more action in the end

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])  
Click(/html/body/main/div[3]/section/.../button[2])  
Scrape(/html/body/main/div[3]/section/.../div[1]/addr)  
Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])  
Scrape(/html/body/main/div[3]/section/.../div[1]/phone)  
Scrape(/html/body/main/div[3]/section/.../div[2]/addr)  
Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])  
Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
```

Trace

```
Scrape(/html/body/main/div[3]/section/.../div[3]/addr)
```

Program 3

- Using a loop

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

```
Click(/html/body/main/div[3]/section/.../button[2])
```

For i = 1, 2, ... do:

```
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

- **What does it do?**

Program 3

- Using a loop

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

```
Click(/html/body/main/div[3]/section/.../button[2])
```

For i = 1, 2, ... do:

```
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

- **What does it do?**

- **Does it reproduce/generalize trace? And why?**

Program 3

- Using a loop

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

```
Click(/html/body/main/div[3]/section/.../button[2])
```

For i = 1, 2, ... do:

```
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

- **Can you write a different loopy program?**

Program 3

- Another loopy program..

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

```
Click(/html/body/main/div[3]/section/.../button[2])
```

```
Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
```

```
Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
```

```
Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
```

For i = 2, ... do:

```
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

- **But which one is “better”?**

Let's Look At These Programs: Pros & Cons

	Pros	Cons
Trace itself		
Trace + One Action		
Loop		

Let's Look At These Programs: Pros & Cons

	Pros	Cons
Trace itself	Can reproduce Easy to synthesize	Does not generalize
Trace + One Action	Can generalize	Generalization may be wrong
Loop	More reasonable generalization	Potentially harder to synthesize

Let's Look At These Programs: Pros & Cons

	Pros	Cons
Trace itself	Can reproduce Easy to synthesize	Does not generalize
Trace + One Action	Can generalize	Generalization may be wrong
Loop	More reasonable generalization	Potentially harder to synthesize

WebRobot synthesizes loopy programs!

How To Synthesize Loopy Programs?

- Trace -> Program: find a program that generalizes the given trace

```
1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
2: Click(/html/body/main/div[3]/section/.../button[2])
3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
5: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
6: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])
8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
```



Specification: Trace



Program with loops

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
Click(/html/body/main/div[3]/section/.../button[2])
For i = 1, 2, ... do:
    Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
    Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
    Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

How To Synthesize Loopy Programs?

- Trace -> Program: find a program that generalizes the given trace

```
1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
2: Click(/html/body/main/div[3]/section/.../button[2])
3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])
5: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)
6: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])
8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)
```



Specification: Trace



Program with loops

**Exam 2 does NOT
cover the rest of
lecture!**

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
Click(/html/body/main/div[3]/section/.../button[2])
```

For i = 1, 2, ... do:

```
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

Take A Step Back..

- Previously, **programming-by-example**

- Specification: input-output pairs

(2,4)

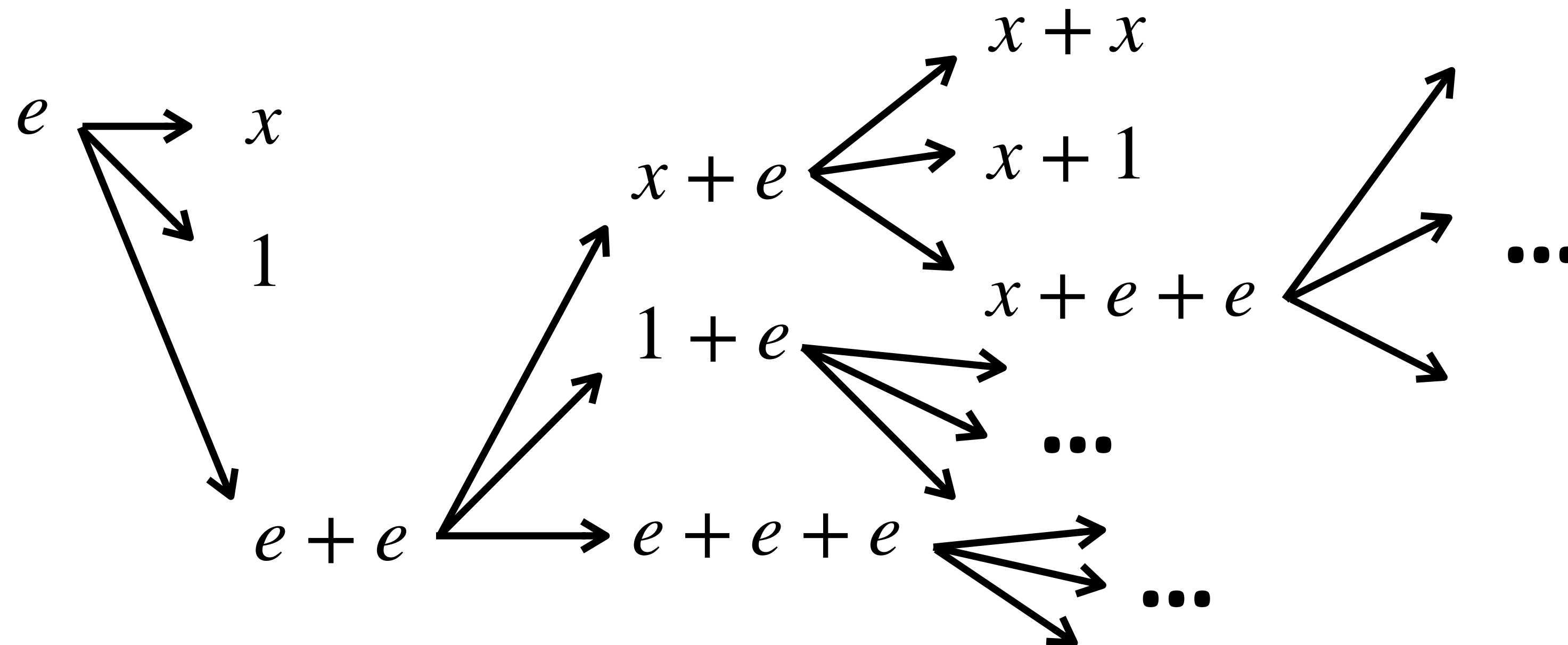
- Program: defined by context-free grammar

$e \rightarrow x$

- Synthesis approach: top-down search

| 1

| $e + e$





Take A Step Back..

- Previously, **programming-by-example**
 - Specification: input-output pairs
 - Program: defined by context-free grammar
 - Synthesis approach: top-down search
- Now, **programming-by-demonstration**
 - Specification:
 - Program:
 - Synthesis approach:



Take A Step Back..

- Previously, **programming-by-example**
 - Specification: input-output pairs
 - Program: defined by context-free grammar
 - Synthesis approach: top-down search
- Now, **programming-by-demonstration**
 - Specification: a sequence of actions
 - Program: still context-free grammar (but a different one) 
 - Synthesis approach: 

WebRobot Grammar

```
Program      P ::= S; ...; S
Statement    S ::= Click(DOM-selector)
              | Scrape(DOM-selector)
              | CP(input-selector, DOM-selector)
              | Enter(string, DOM-selector)
              | foreach var in children(DOM-selector) do { P }
              | while(true) do { P; Click(DOM-selector) }
DOM-selector ::= a_normal_DOM_selector
              | var/a_normal_DOM_selector
```


WebRobot Grammar

This entire
thing is a
“program”

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
Click(/html/body/main/div[3]/section/.../button[2])
For i = 1, 2, ... do:
    Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
    Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
    Scrape(/html/body/main/div[3]/section/.../div[i]/phone)
```

```
Program      P ::= S; ...; S
Statement    S ::= Click(DOM-selector)
              | Scrape(DOM-selector)
              | CP(input-selector, DOM-selector)
              | Enter(string, DOM-selector)
              | foreach var in children(DOM-selector) do { P }
              | while(true) do { P; Click(DOM-selector) }
DOM-selector ::= a_normal_DOM_selector
              | var/a_normal_DOM_selector
```

WebRobot Grammar

`CP(input[1], /html/body/main/div[3]/section/div/div/div[1])` → This is a “statement”

`Click(/html/body/main/div[3]/section/.../button[2])`

For $i = 1, 2, \dots$ do:

`Scrape(/html/body/main/div[3]/section/.../div[i]/addr)`

`Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])`

`Scrape(/html/body/main/div[3]/section/.../div[1]/phone)`

```
Program    P ::= S; ...; S
```

```
Statement  S ::= Click(DOM-selector)
           | Scrape(DOM-selector)
           | CP(input-selector, DOM-selector)
           | Enter(string, DOM-selector)
           | foreach var in children(DOM-selector) do { P }
           | while(true) do { P; Click(DOM-selector) }
```

```
DOM-selector ::= a_normal_DOM_selector
             | var/a_normal_DOM_selector
```

WebRobot Grammar

`CP(input[1], /html/body/main/div[3]/section/div/div/div[1])` → This is a “statement”

`Click(/html/body/main/div[3]/section/.../button[2])` →

For $i = 1, 2, \dots$ do:

Scrape(/html/body/main/div[3]/section/.../div[i]/addr)

Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

```
Program      P ::= S; ...; S
Statement    S ::= Click(DOM-selector)
              | Scrape(DOM-selector)
              | CP(input-selector, DOM-selector)
              | Enter(string, DOM-selector)
              | foreach var in children(DOM-selector) do { P }
              | while(true) do { P; Click(DOM-selector) }
DOM-selector ::= a_normal_DOM_selector
              | var/a_normal_DOM_selector
```

WebRobot Grammar

`CP(input[1], /html/body/main/div[3]/section/div/div/div[1])`

→ This is a “statement”

`Click(/html/body/main/div[3]/section/.../button[2])`

→ Another “statement”

For i = 1, 2, ... do:

Scrape(/html/body/main/div[3]/section/.../div[i]/addr)

Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

```
Program    P ::= S; ...; S
```

```
Statement  S ::= Click(DOM-selector)
```

```
           | Scrape(DOM-selector)
```

```
           | CP(input-selector, DOM-selector)
```

```
           | Enter(string, DOM-selector)
```

```
           | foreach var in children(DOM-selector) do { P }
```

```
           | while(true) do { P; Click(DOM-selector) }
```

```
DOM-selector ::= a_normal_DOM_selector
```

```
           | var/a_normal_DOM_selector
```

WebRobot Grammar

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

→ This is a “statement”

```
Click(/html/body/main/div[3]/section/.../button[2])
```

→ Another “statement”

```
For i = 1, 2, ... do:
```

```
  Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
  Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
  Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
```



```
Program      P ::= S; ...; S
```

```
Statement    S ::= Click(DOM-selector)
```

```
              | Scrape(DOM-selector)
```

```
              | CP(input-selector, DOM-selector)
```

```
              | Enter(string, DOM-selector)
```

```
              | foreach var in children(DOM-selector) do { P }
```

```
              | while(true) do { P; Click(DOM-selector) }
```

```
DOM-selector ::= a_normal_DOM_selector
```

```
              | var/a_normal_DOM_selector
```

WebRobot Grammar

```
CP(input[1], /html/body/main/div[3]/section/div/div/div[1])
```

→ This is a “statement”

```
Click(/html/body/main/div[3]/section/.../button[2])
```

→ Another “statement”

```
For i = 1, 2, ... do:
```

```
  Scrape(/html/body/main/div[3]/section/.../div[i]/addr)
```

```
  Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])
```

```
  Scrape(/html/body/main/div[3]/section/.../div[1]/phone)
```

→ Also a “statement”!

```
Program      P ::= S; ...; S
```

```
Statement    S ::= Click(DOM-selector)
```

```
              | Scrape(DOM-selector)
```

```
              | CP(input-selector, DOM-selector)
```

```
              | Enter(string, DOM-selector)
```

```
              | foreach var in children(DOM-selector) do { P }
```

```
              | while(true) do { P; Click(DOM-selector) }
```

```
DOM-selector ::= a_normal_DOM_selector
```

```
              | var/a_normal_DOM_selector
```

WebRobot Grammar

`CP(input[1], /html/body/main/div[3]/section/div/div/div[1])` → This is a “statement”
`Click(/html/body/main/div[3]/section/.../button[2])` → Another “statement”
For i = 1, 2, ... do:
 `Scrape(/html/body/main/div[3]/section/.../div[i]/addr)` → Also a “statement”!
 `Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])` →
 `Scrape(/html/body/main/div[3]/section/.../div[1]/phone)` →

```
Program    P ::= S; ...; S
Statement  S ::= Click(DOM-selector)
           | Scrape(DOM-selector)
           | CP(input-selector, DOM-selector)
           | Enter(string, DOM-selector)
           | foreach var in children(DOM-selector) do { P }
           | while(true) do { P; Click(DOM-selector) }
DOM-selector ::= a_normal_DOM_selector
             | var/a_normal_DOM_selector
```

WebRobot Grammar

CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

→ This is a “statement”

Click(/html/body/main/div[3]/section/.../button[2])

→ Another “statement”

For i = 1, 2, ... do:

Scrape(/html/body/main/div[3]/section/.../div[i]/addr)

→ Also a “statement”!

Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

→ This is a “program”!

```
Program    P ::= S; ...; S
```

```
Statement S ::= Click(DOM-selector)
```

```
           | Scrape(DOM-selector)
```

```
           | CP(input-selector, DOM-selector)
```

```
           | Enter(string, DOM-selector)
```

```
           | foreach var in children(DOM-selector) do { P }
```

```
           | while(true) do { P; Click(DOM-selector) }
```

```
DOM-selector ::= a_normal_DOM_selector
```

```
           | var/a_normal_DOM_selector
```

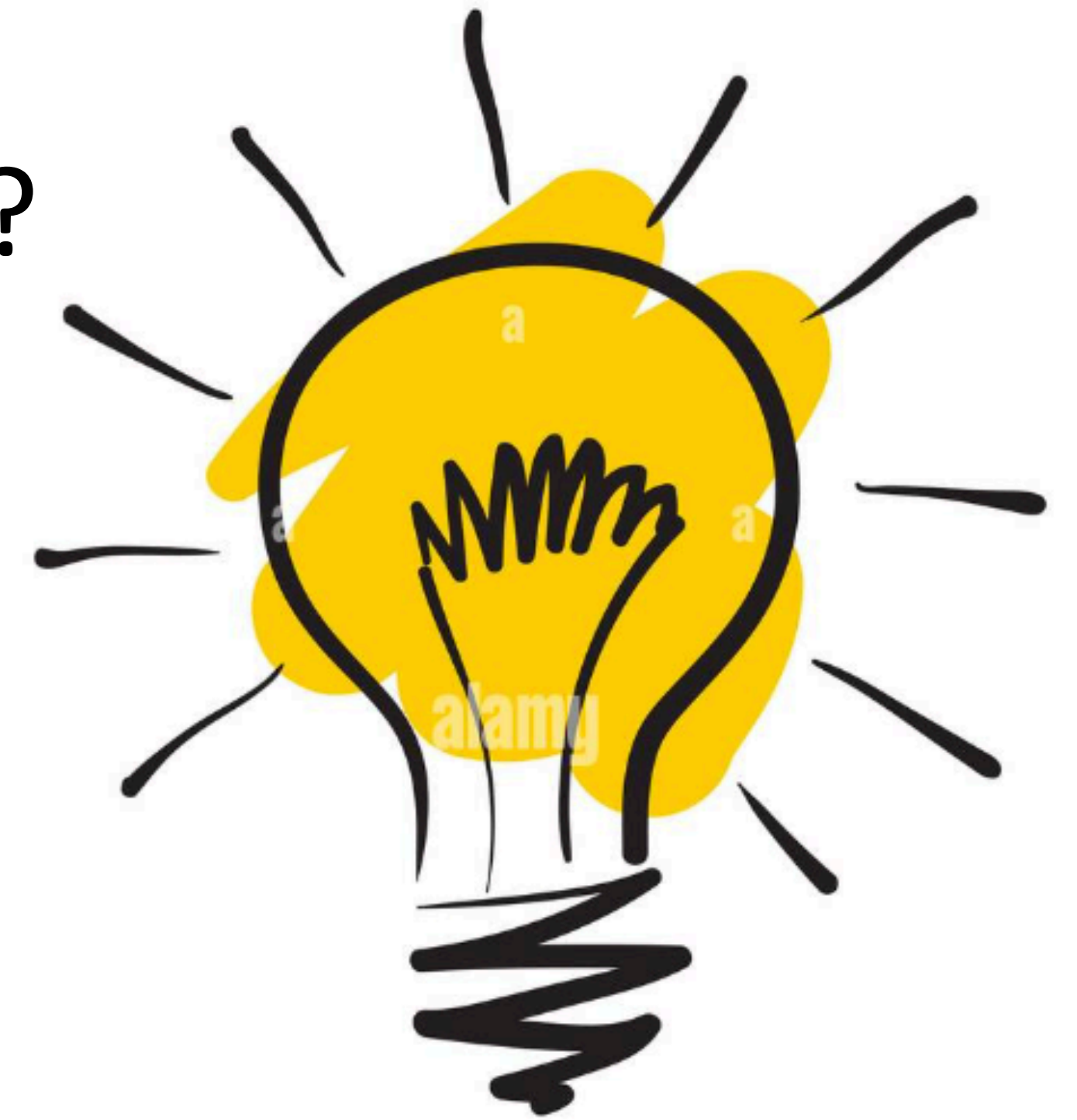

Take A Step Back..

- Previously, **programming-by-example**
 - Specification: input-output pairs
 - Program: defined by context-free grammar
 - Synthesis approach: top-down search
- Now, **programming-by-demonstration**
 - Specification: a sequence of actions
 - Program: still context-free grammar (but a different one)
 - Synthesis approach: ?



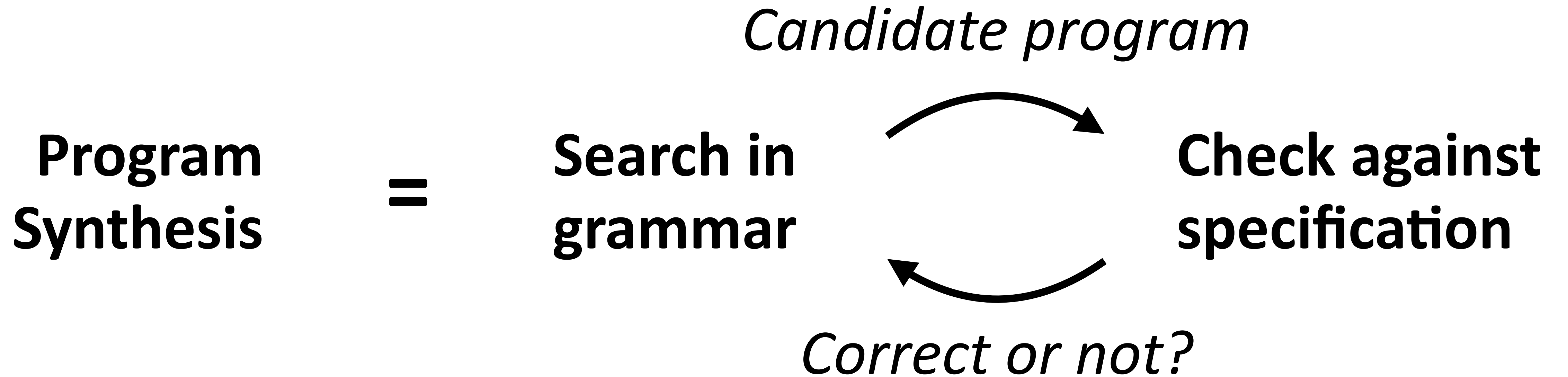
Any Ideas?

- What's fundamental in program synthesis?
- What's difference between PBE vs. PBD?
- Anything we can borrow from PBE to solve PBD?

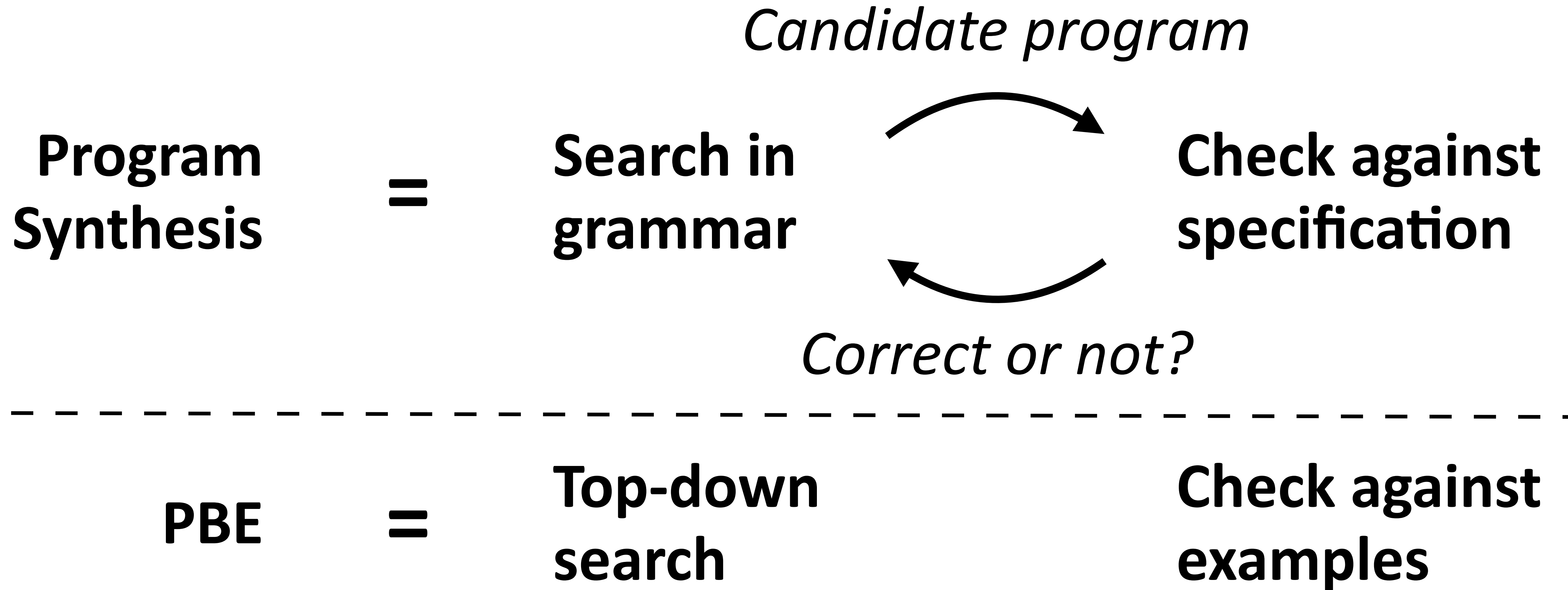


idea

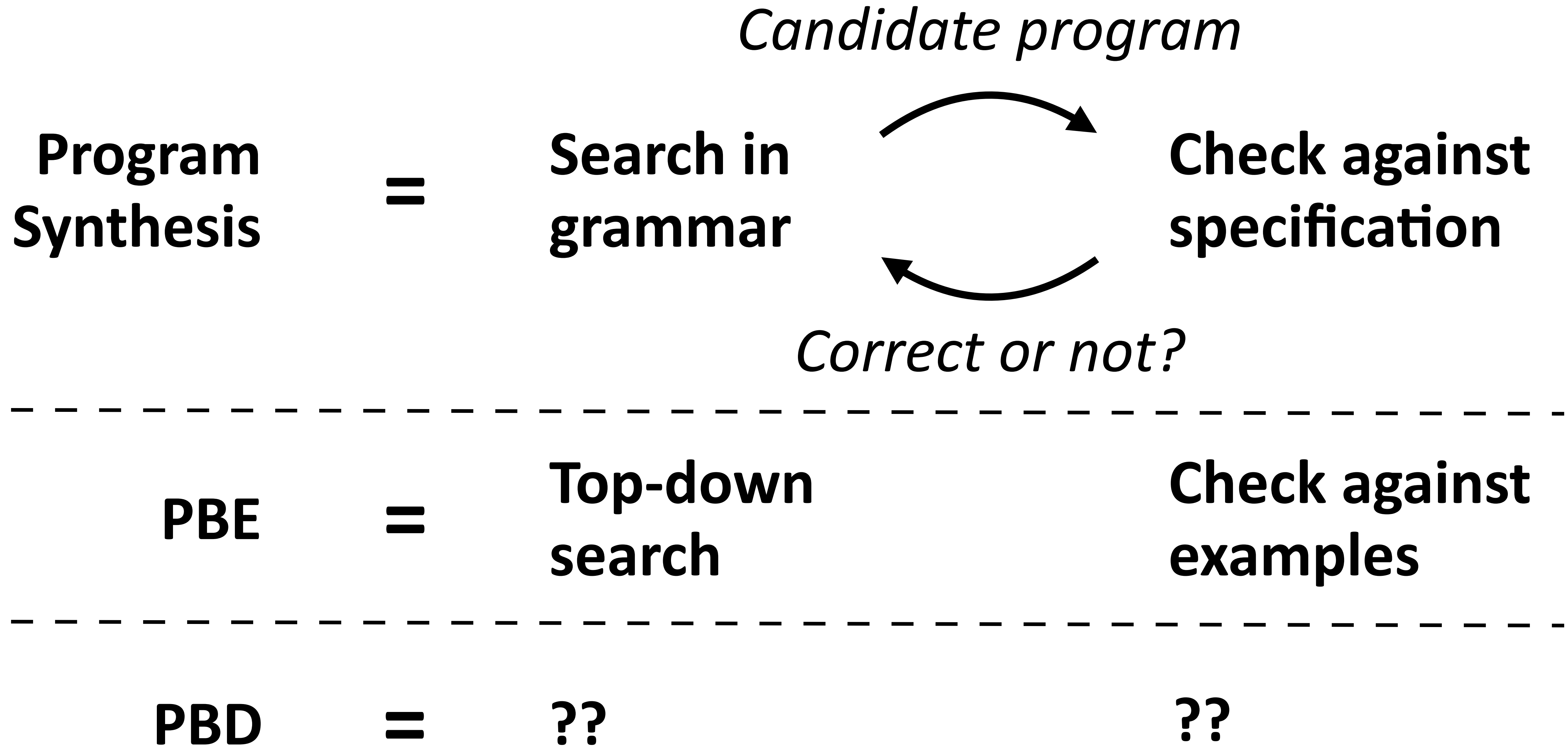
Synthesis = Search + Check



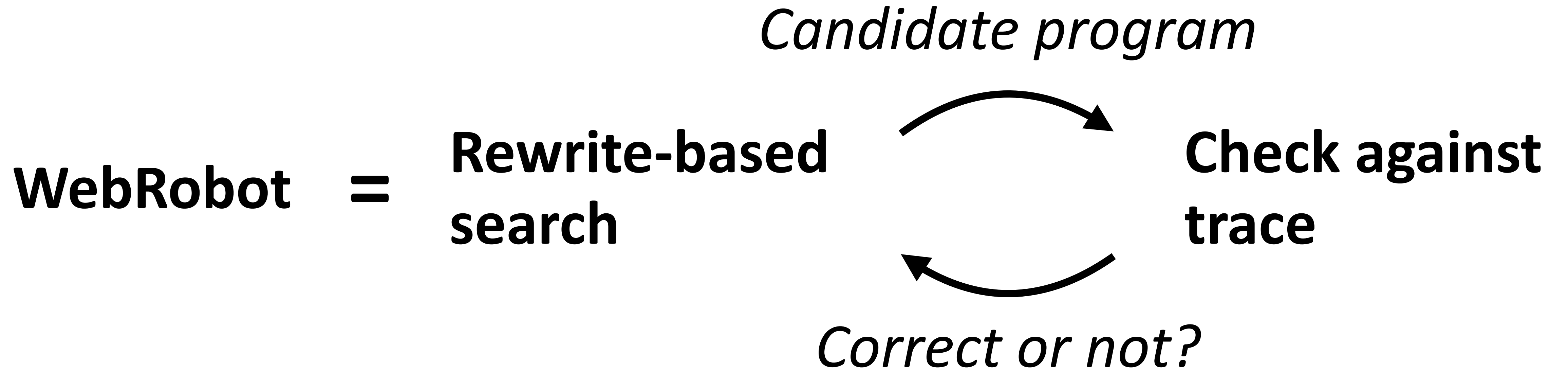
Synthesis = Search + Check



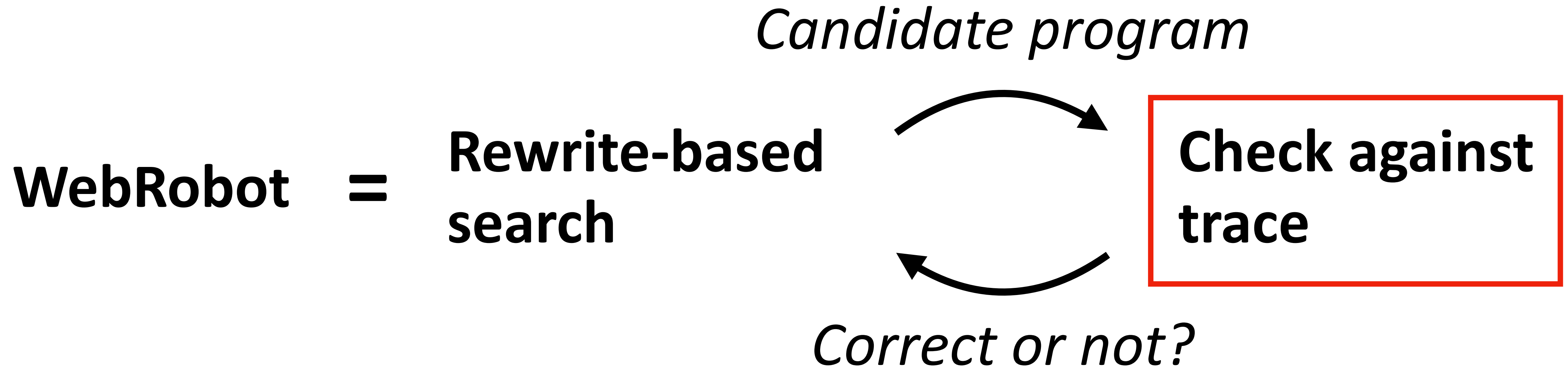
Synthesis = Search + Check



WebRobot = Rewrite + Trace Semantics

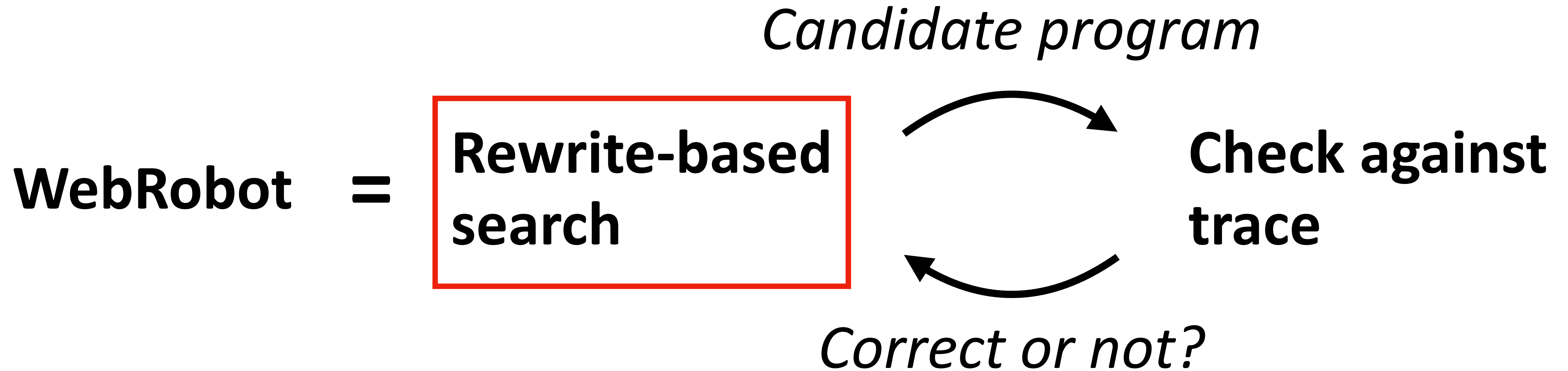


WebRobot = Rewrite + Trace Semantics



- Still use an interpreter to run the program..
- But not a normal interpreter!
- A new interpreter based on “trace semantics”
 - Idea: it runs program but logs actions program executes

WebRobot = Rewrite + Trace Semantics



- Use **both** grammar and trace!
- Idea: identify some repeating pattern from trace, replace a sequence of repetitive actions by a loop

WebRobot = Rewrite + Trace Semantics

1: CP(input[1], /html/body/main/div[3]/section/div/div/div[1])

2: Click(/html/body/main/div[3]/section/.../button[2])

3: Scrape(/html/body/main/div[3]/section/.../div[1]/addr)

4: Scrape(/html/body/main/div[3]/section/.../div[1]/.../div[3])

5: Scrape(/html/body/main/div[3]/section/.../div[1]/phone)

6: Scrape(/html/body/main/div[3]/section/.../div[2]/addr)

7: Scrape(/html/body/main/div[3]/section/.../div[2]/.../div[3])

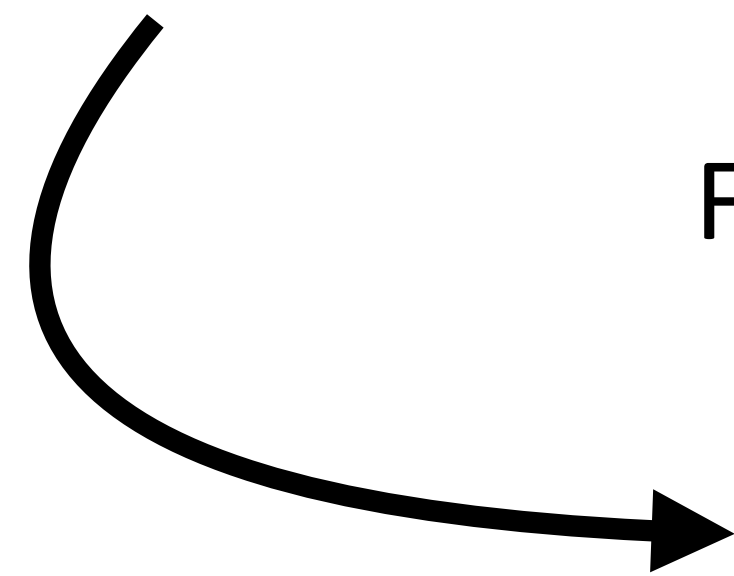
8: Scrape(/html/body/main/div[3]/section/.../div[2]/phone)

For i = 1, 2, ... do:

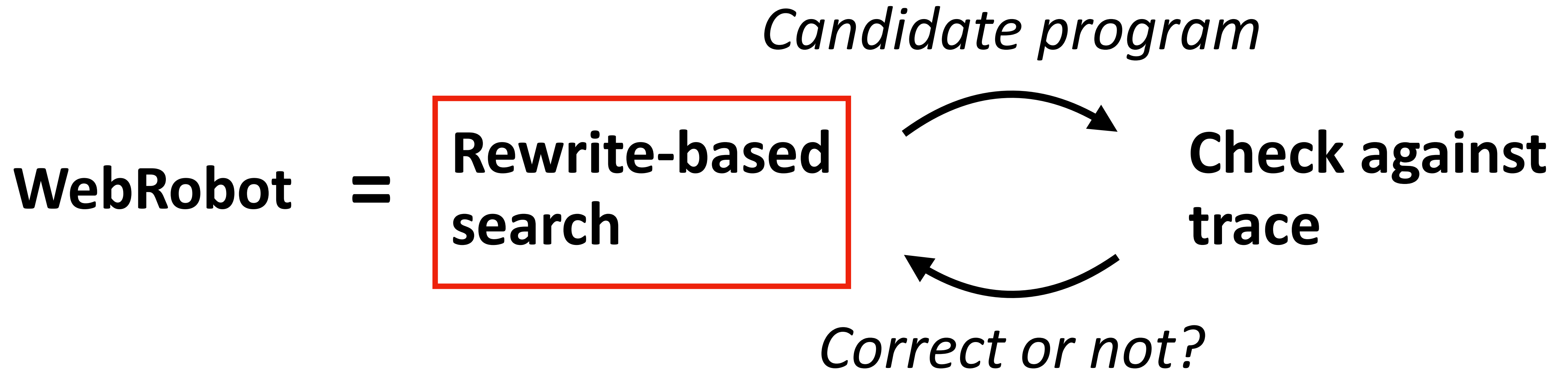
Scrape(/html/body/main/div[3]/section/.../div[i]/addr)

Scrape(/html/body/main/div[3]/section/.../div[i]/.../div[3])

Scrape(/html/body/main/div[3]/section/.../div[i]/phone)



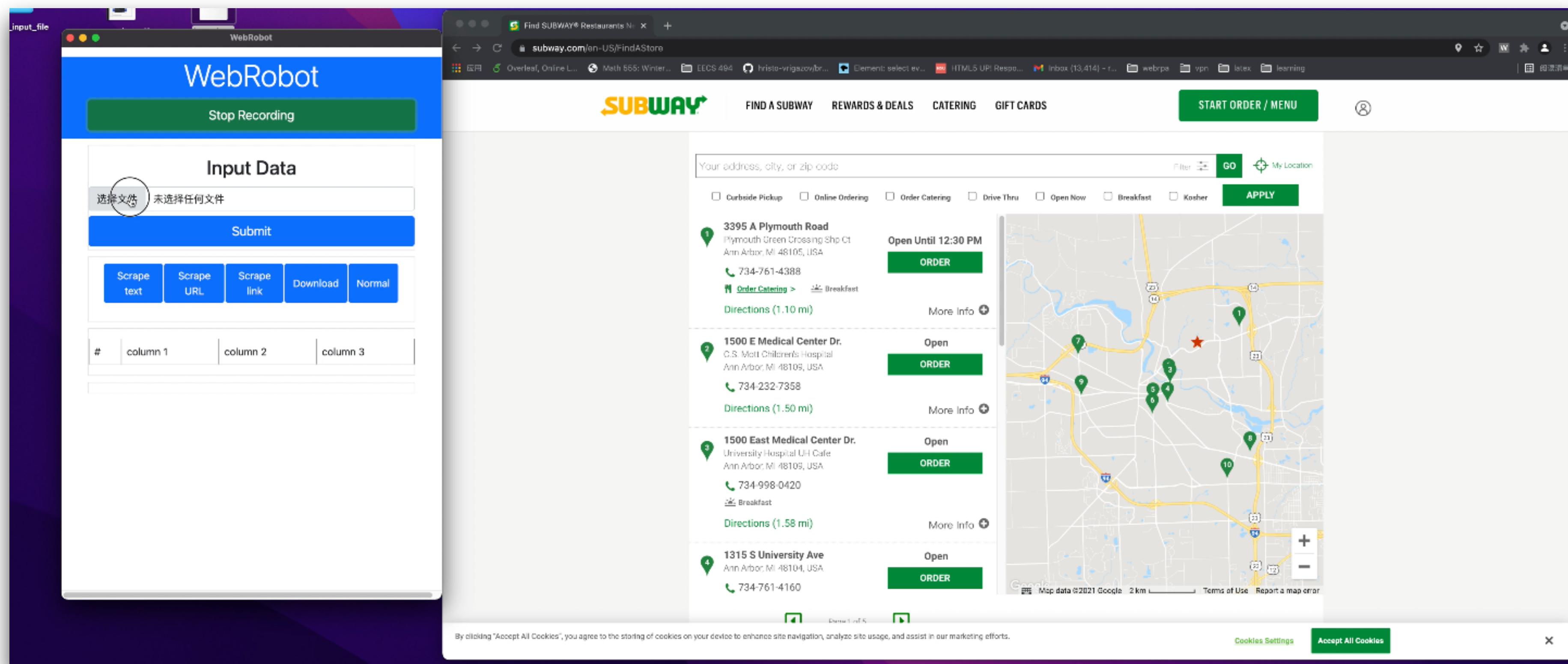
WebRobot = Rewrite + Trace Semantics



- Use **both** grammar and trace!
- Idea: identify some repeating pattern from trace, replace a sequence of repetitive actions by a loop
- **Challenge: many different patterns, many different ways to replace —> need to track a large number of rewrites**

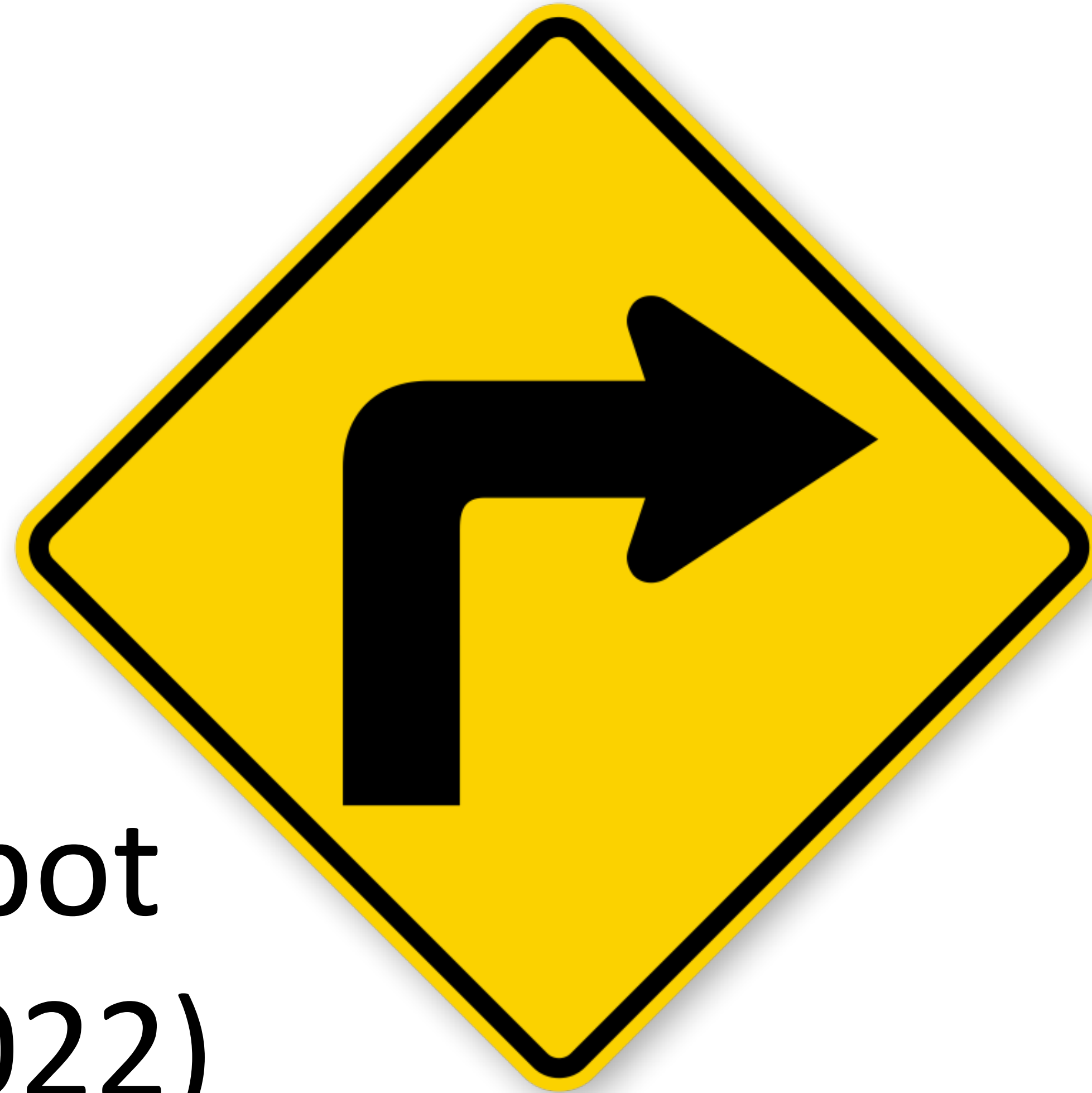
WebRobot Recap

- Automatically generate web automation scripts from user interactions with web browser
- Tasks: data scraping, data entry, form filling, etc.



Switch Gears...

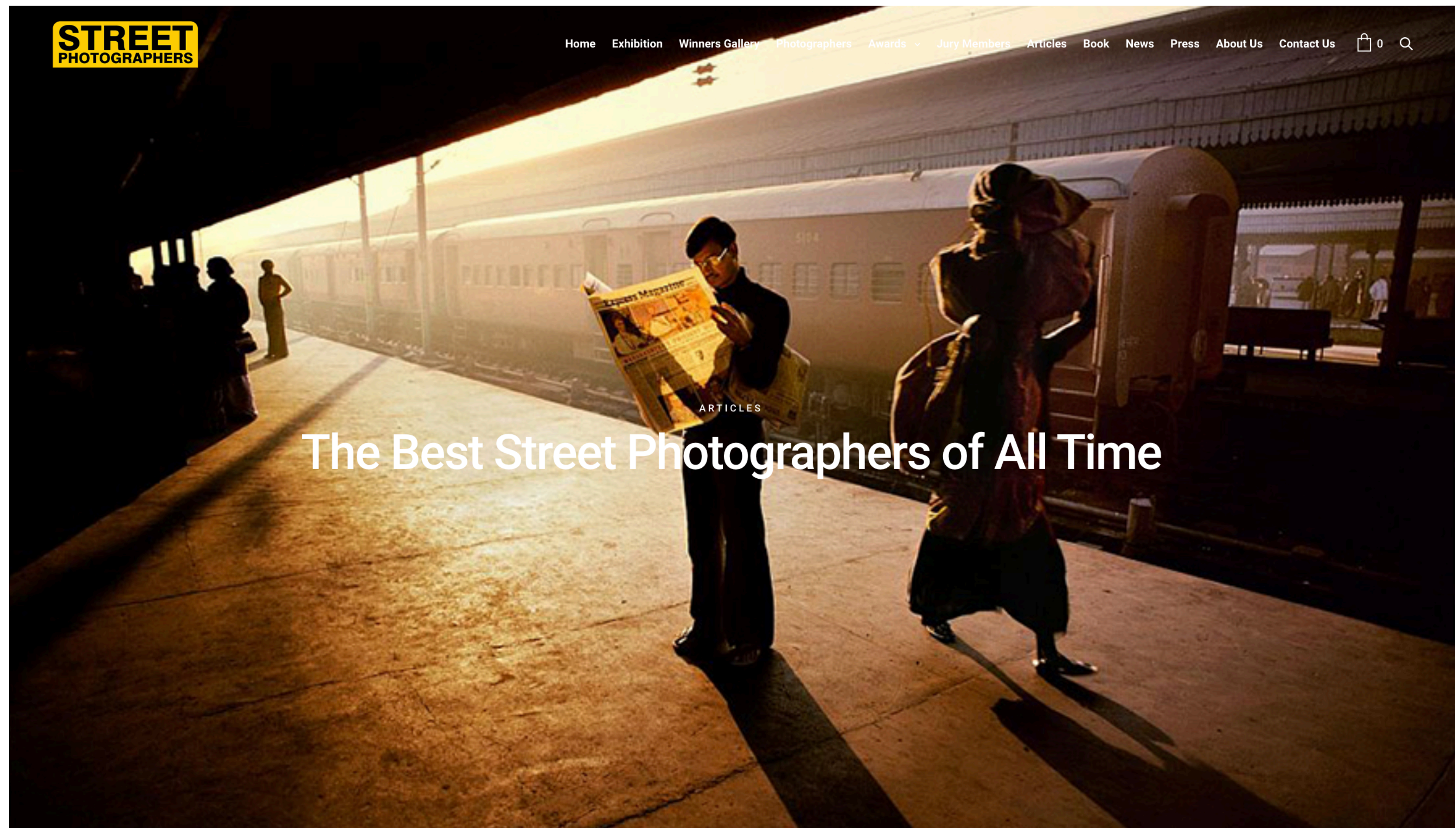
SemanticOn
(UIST 2022)



WebRobot
(PLDI 2022)

Can WebRobot Automate This Task?

- Go to “The Best Street Photographers of All Time”
- Scrape all images with **at least two people interacting with each other**



Scrape Images with People Interacting

The Best Street Photographers of All Time

Street Photography is one of the most challenging and exciting genres of photography. Sometimes when we see a photo, it gets into our mind and remains eternal for life. This could be moments of sorrow and pain, a disaster happening in the world, usual human interaction in our society, or even a moment of everyday life.

Having awareness of photography history and the famous actors of that history will definitely make you a better photographer. If you are trying to capture perfect photographs, knowing the masters of photography will show you what makes a photograph perfect.

In this article, we will introduce some of impressive master photographers in this genre. Stay with us.

Alex Webb

Alex Webb (born 1952) is a Magnum photographer who uses strong colors, light, and emotion to capture beautifully complex images. Depth is a strong element in his works, he is a master of capturing serendipitous moments in full-blown technicolor. He has an eye for the unusual and a knack for capturing unexpected moments. With an acute focus on the more serendipitous frames, as opposed to simply documenting one thing, he finds influence in a melting pot of countries, cultures and subcultures.

Quotes:

“A street photographer wanders and responds spontaneously to what he or she finds, rather than consciously searching for specific things, letting the world—and one’s unconscious—lead one where it will. This initial approach or attitude makes street photography different from more directed photojournalism, in which there is a conscious effort to find a ‘story’—and also makes street photography different from more conceptual photography, in which there is often a preconceived agenda.”



Alfred Eisenstaedt

Alfred Eisenstaedt (born 1898) was one of the luminaries of German-born American photographer. He is best known for his candid black-and-white shots of celebrities, politicians, and captivating street shots. He was among those Europeans who pioneered the use of the 35-millimeter camera in photojournalism to get closer to his subjects and create more candid pictures.

Eisenstaedt was a master at finding the details that tell the big story. His style was unaffected and naturalistic; he let his subjects speak for themselves.

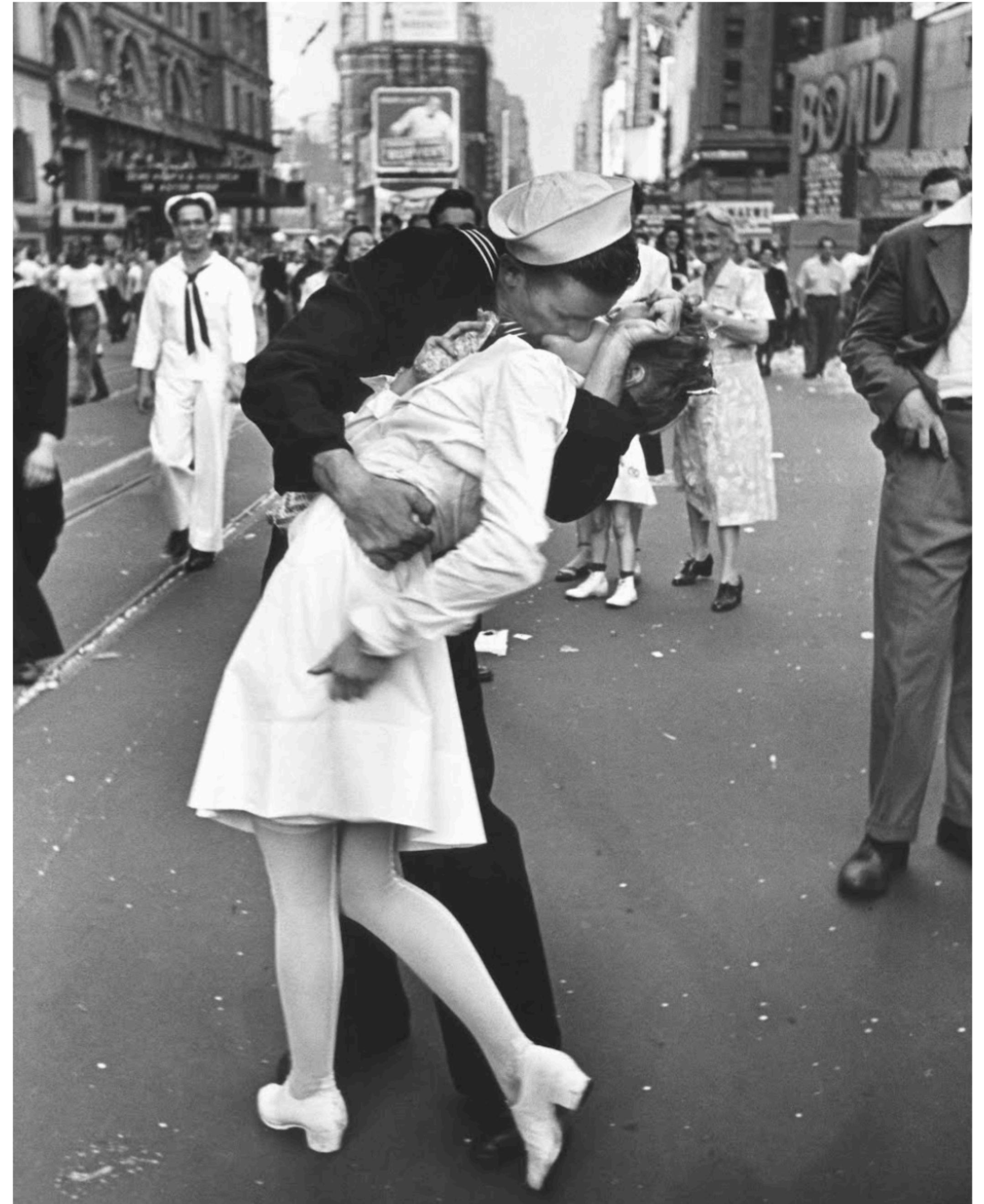
Eisenstaedt perfected certain techniques for capturing the spontaneous moment that has given us some of our most enduring photographic images. Also, he was a favorite among editors, not only for his quick eye but also for his ability in making good photographs of any situation or event. He became one of only four original staffers on *Life* magazine, winning numerous awards for his memorable images. Referred to as one of the founding fathers of photojournalism, he completed around 2500 photo-essays and ninety cover photographs for the magazine.

Quotes:

"Today's photographers think differently. Many can't see real light anymore. They think only in terms of strobe – sure, it all looks beautiful but it's not really seeing. If you have the eyes to see it, the nuances of light are already there on the subject's face. If your thinking is confined to strobe light sources, your palette becomes very mean – which is the reason I photograph only in available light."

"I dream that someday the step between my mind and my finger will no longer be needed. And that simply by blinking my eyes, I shall make pictures. Then, I think, I shall really have become a photographer."

"It is more important to click with people than to click the shutter."



Alfred Stieglitz

Alfred Steiglitz (born 1864) was a renowned American photographer who played a crucial role in establishing photography as an integral part of modern art in America, and contributed greatly to the development of Modernism during the 20th century.

Alfred Stieglitz, founder of the Photo-Secession, a Pictorialist group of photographers, he elevated the discourse and practice of photography, forming key connections between American and European movements.

Stieglitz was feverishly devoted to his work and mission and produced thousands of editions in his lifetime, covering numerous themes that captured a period of rapid transition in American society.

Quotes:

“Photography is not an art. Neither is painting, nor sculpture, literature or music. They are only different media for the individual to express his aesthetic feelings... You do not have to be a painter or a sculptor to be an artist. You may be a shoemaker. You may be creative as such. And, if so, you are a greater artist than the majority of the painters whose work is shown in the art galleries of today.”



Andre Kertesz

Andre Kertesz (born 1894) is known for his realistic and sensitive scenes of everyday life. He was one of the founders of photojournalism. He pioneered in using a small camera, producing snapshot images with unexpected details.

Kertész remains best known for his contributions to photojournalism, employing distinctively dynamic compositions throughout his influential photo essays.

Prizing emotional impact over technique, he famously remarked, "I just walk around, observing the subject from various angles until the picture elements arrange themselves into a composition that pleases my eye. I do what I feel, that's all. I am an ordinary photographer working for his own pleasure. That's all I've ever done."

Quotes:

"Technique isn't important. Technique is in the blood. Events and mood are more important than good light and the happening is what is important," he says. "I still regard myself as an amateur today and I hope that's what I'll stay until the end of my life. Because I'm forever a beginner who discovers the world again and again."

"The moment always dictates in my work. What I feel, I do. This is the most important thing for me. Everybody can look, but they don't necessarily see. I never calculate or consider; I see a situation and I know that it's right, even if I have to go back to get the proper lighting."

"If you want to write you should learn the alphabet. You write and write and in the end you have a beautiful, perfect alphabet. But it isn't the alphabet that is important. The important thing is what you are writing, what you are expressing. The same thing goes for photography. Photographs can be technically perfect and even beautiful, but they have no expression."



Berenice Abbott

Berenice Abbott (born 1898) was a pioneer American documentary photographer.

She is remembered as one of the most independent, determined and respected photographers of the twentieth century.

She is best known for her striking, black-and-white documentation of New York City and preservation of the works of Eugène Atget.

Berenice Abbott was an enthusiastic proponent of modernism in photography, and was strongly opposed to pictorialism, the painterly style that dominated photography in the early 20th century. In her view a good photograph was shaped by the specific characteristics of photography itself, and not by those of painting.

She worked on a specially designed lighting process, which she called Projection Photography. Abbott also invented and patented other photographic related equipment and gadgets.

Quotes:

"A photograph is not a painting, a poem, a symphony, a dance. It is not just a pretty picture, not an exercise in contortionist techniques and sheer print quality,"

"There are many teachers who could ruin you. Before you know it you could be a pale copy of this teacher or that teacher. You have to evolve on your own."

"Photography can never grow up if it imitates some other medium. It has to walk alone; it has to be itself."



Bill Cunningham

Bill Cunningham (born 1929) was an American street and fashion photographer who considered the forefather of street style photography.

His works show that street style is not only about fashion; it's about the people and the changing culture. He shared his photos in a fashion column for the **New York Times**, called "On the Street." His photography was capturing the evolution of style, of trends, and of every day, both in New York City and in Paris.

He made a career taking unexpected photographs of everyday people, socialites, and fashion personalities, many of whom valued his company. He was a self-taught photographer. Most of his pictures were never sold or published. He said: "I'm really doing this for myself. I'm stealing people's shadows, so I don't feel as guilty when I don't sell them."

Quotes:

"The problem is I'm not a good photographer. To be perfectly honest, I'm too shy. Not aggressive enough. Well, I'm not aggressive at all. I just loved to see wonderfully dressed women, and I still do. That's all there is to it."



Bruce Davidson

Bruce Davidson (born 1933) remains one of the world's greatest photographers. A member of the Magnum Photos agency.

He is known for capturing images of communities and individuals living on the fringes of society and dedication to the documentation of social inequality. His images frequently convey the loneliness and isolation of the subjects portrayed.

Although Davidson wouldn't consider himself a street photographer, we can learn very much from Davidson's experiences. His photographs show a keen desire to reveal and understand the complexities of individual lives and reflect universal truths and concerns. Most of Davidson's protagonists show the proximity between the photographer and his subjects, both spatially and emotionally.

When he chooses a project, he sticks with it and pursues it for several months, and often several years. He is a man that cares deeply for his subjects and cares more about his relationships with them than the photos themselves.

Quotes:

"I am a photographer who uses various professional cameras and film formats to express the way I see and explore reality. Cameras become an extension of my vision and I need to love the thing. Each tool has its purpose and it is up to me to choose one to use for a particular photographic project"

"All my photographs are portraits—self-portraits, because you can't photograph someone without reflecting/echoing, like a bat sending out a signal that comes back to you. You get not only a picture of who you're photographing, but you get a picture of yourself at the same time," he says. "Most of my photographs are compassionate, gentle, and personal. They tend to let the viewer see himself. They tend not to preach. And they tend not to pose as art."

"Taking photographs, taking candid photographs, means that the photographer is an invisible man. Whereas there is still a feeling that in having a photograph taken there is loss of face: something of the soul is gone."



Elliott Erwitt

Elliott Erwitt (born 1928) is an American advertising and documentary photographer. Known for his candid and often humorous black-and-white images. He has been a Member of **Magnum Photos** since 1954.

Erwitt is responsible for some of the most iconic photographs of the 20th century.

He is renowned for his humanistic photographs. "You just have to care about what's around you and have a concern with humanity and the human comedy," he says.

With his career spanning over several decades, he has certainly used this framing to his advantage.

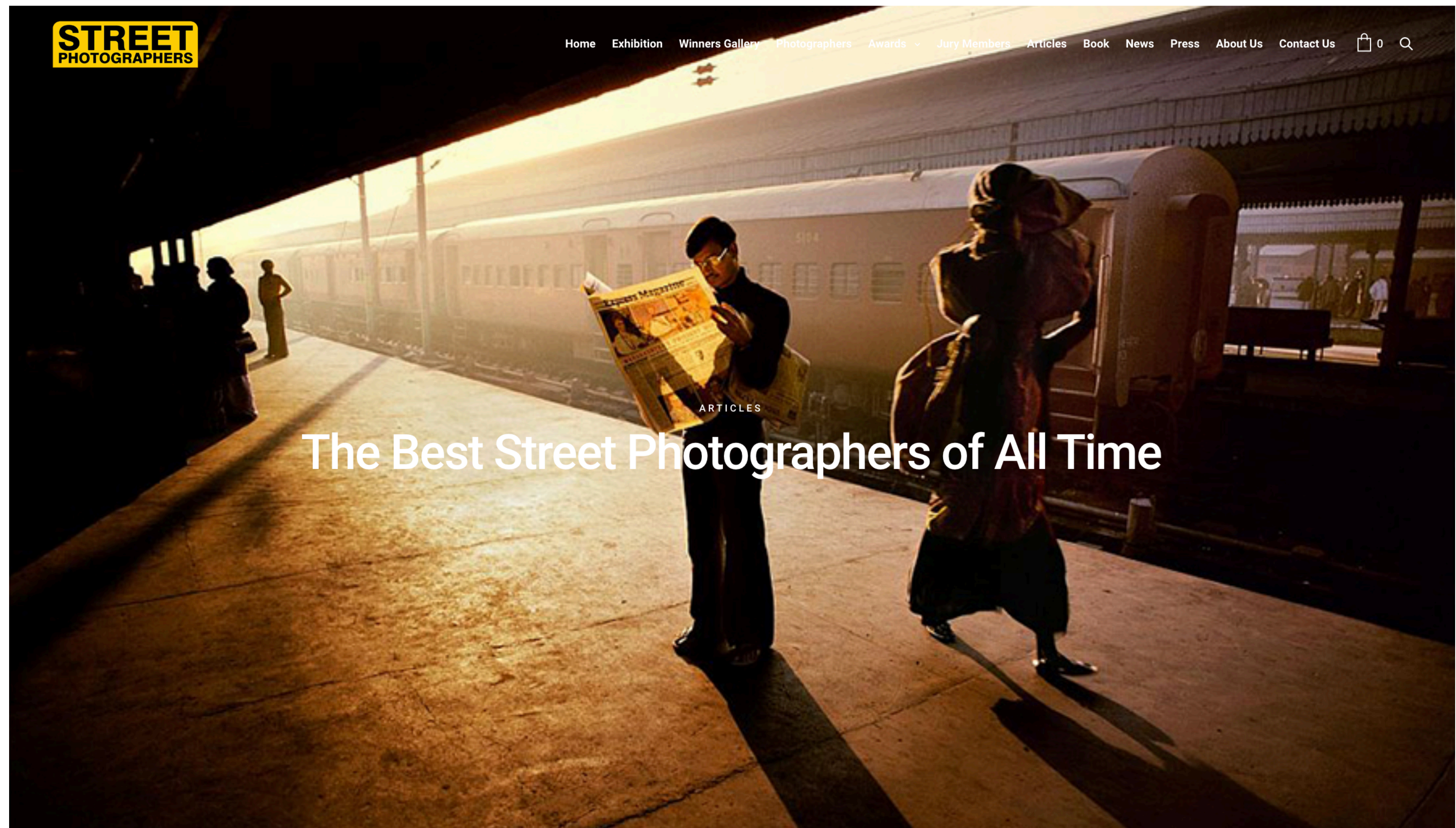
Because he finds the most interesting photo, the next photo he is going to take. So he continues to strive to go out and hunt for that next photo.

Erwitt didn't make out to become a great or famous photographer. Rather, he saw it as an enjoyable activity and let his photography be an extension of himself. "I'm an amateur photographer, apart from being a professional one, and I think maybe my amateur pictures are the better ones." He is naturally curious, quirky, and humorous and used his camera to capture that in the world around him.



Can WebRobot Automate This Task?

- Go to “The Best Street Photographers of All Time”
- Scrape all images with **at least two people interacting with each other**



What's “New” In This Task?

- Scrape images (rather than text)
- “At least two people interacting with each other” — a new form of condition!
- How to write program with this new logic?

SemanticOn (UIST 2022)

SemanticOn: Specifying Content-Based Semantic Conditions for Web Automation Programs

Kevin Pu
jpu@dgp.toronto.edu
University of Toronto

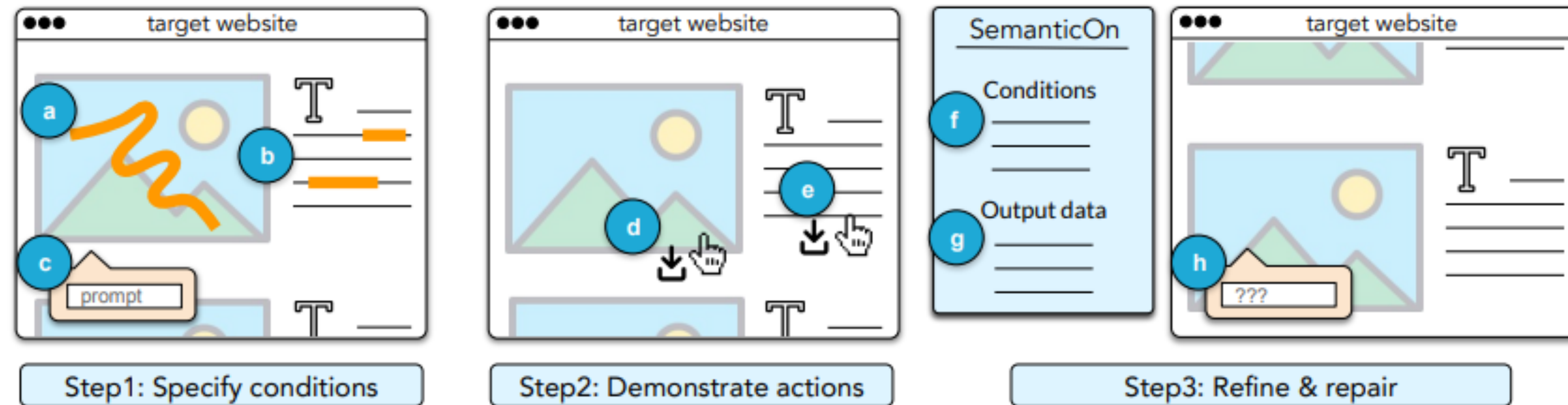
Rainey Fu
rainey.fu@mail.utoronto.ca
University of Toronto

Rui Dong
ruidong@umich.edu
University of Michigan

Xinyu Wang
xwangsd@umich.edu
University of Michigan

Yan Chen
yanchen@dgp.toronto.edu
University of Toronto

Tovi Grossman
tovi@dgp.toronto.edu
University of Toronto



UIST 2022 Best Paper Honorable Mention Award

SemanticOn Demo

The image shows a web-based interface for SemanticOn. On the left is a control panel with three sections: 'Condition Panel' with buttons for 'Text Condition' and 'Image Condition', a text input field, and a 'Logic' toggle; 'Action Panel' with buttons for 'Scrape Text', 'Download Image', and 'Normal'; and 'Output Panel'. On the right is a browser window displaying the article 'The Best Street Photographers of All Time'. A search overlay is active, showing the input 'a few people interacting with each other' and 'Add'/'Cancel' buttons. A blue callout box on the right says 'Download pictures of two or three people interacting with each other'. The article text includes a quote about street photography and a section for Alfred Eisenstaedt.

<https://www.youtube.com/watch?v=OmJVaji-GJI>

<https://www.youtube.com/watch?v=eM3un7IpORQ>

Key Idea: “Neurosymbolic”

- Synthesized programs use both symbolic and neural components
- Use **loop** to iterate over all images
- Use **neural net** to check if to scrape an image

That's All

- HW6a due today