Token Expression Determinizer

...

“TED”

Team and Responsibilities

Konstantin Itskov
Manager

Theodore Ahlfeld
Coding Guru

Matthew Haigh
Testing

Gideon Mendels
Architect

Overview

The TED programming language is a web-parsing language designed to simplify web scraping and serve as a bridge between complex high-level web-scraping languages like Javascript and imperative programming languages like C.

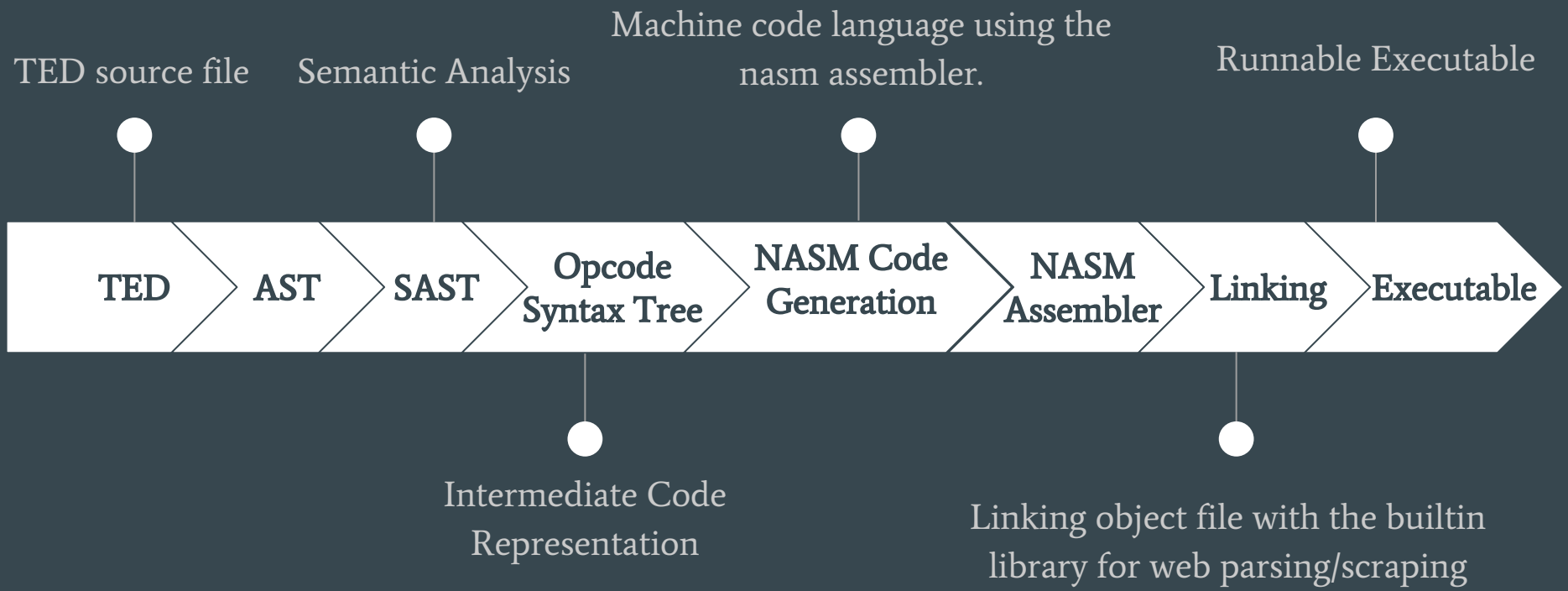
History

- Several group members have used web scraping professionally.
- Per discussion decided to compile to x86 assembly and add web manipulation.
- Original project design turned out to be java-to-java compiler.

Why is it interesting?

TED is the first language designed for web scraping with the power of a low level programming language.

Code Generation



Language Syntax

Highlights

- Specialized Data Types designed for web parsing (Page, Element)
- Simple C-like syntax without pointers
- No memory management
- Web parsing with built-in CSS selection

Code Generation

Similar to GCC code generation

TED

```
4  SECTION .text
5  main:
6    push  rbp
7    mov rbp, rsp
8    sub rsp, 40H
9    mov rdi, Str0
0    call  pageFetch
1    mov qword [rbp-8H], rax
2    mov rsi, Str1
3    mov rdi, [rbp-8H]
4    call  pageFind
5    mov qword [rbp-18H], rax
6    mov rdi, [rbp-18H]
7    call  listHead
8    mov qword [rbp-20H], rax
9    mov rsi, Str8
0    mov rdi, [rbp-20H]
1    call  elementAttr
2    mov qword [rbp-28H], rax
3    mov rdx, [rbp-28H]
4    mov rsi, Str3
5    mov rdi, [stdout]
```

GCC

```
24 main:   ; Function begin
25         push    rbp
26         mov     rbp, rsp
27         sub     rsp, 48
28         mov     dword [rbp-24H], edi
29         mov     qword [rbp-30H], rsi
30         mov     edi, ?_001
31         call    pageFetch
32         mov     qword [rbp-20H], rax
33         mov     rax, qword [rbp-20H]
34         mov     esi, ?_002
35         mov     rdi, rax
36         call    pageFind
37         mov     qword [rbp-18H], rax
38         mov     rax, qword [rbp-18H]
39         mov     rdi, rax
40         call    listHead
41         mov     esi, ?_003
42         mov     rdi, rax
43         call    elementAttr
44         mov     rsi, rax
45         mov     edi, ?_004
```

The Infamous GCD

```c
int gcd(int x, int y) {
    int a = x;
    int b = y;
    while (a!=b) {
        if (a > b) {
            a = a-b;
        } else {
            b = b - a;
        }
    }
    return a;
}
```

```asm

SECTION .text
gcd:
    push    rbp
    mov rbp, rsp
    sub rsp, 40H
    mov rax, [rbp-18H]
    mov qword [rbp-8H], rax
    mov rax, [rbp-20H]
    mov qword [rbp-10H], rax
    mov [rbp-18H], rdi
    mov [rbp-20H], rsi
gcd0w:
    mov rcx, [rbp-10H]
    mov rax, [rbp-8H]
    cmp rax, rcx
    setne   dl
    cmp dl, 1
    jnz gcd0wend
    mov rcx, [rbp-10H]
    mov rax, [rbp-8H]
    cmp rax, rcx
    setg dl
    cmp dl, 1
    jz gcd0w0t
    mov rcx, [rbp-8H]
    mov rax, [rbp-10H]
    sub rax, rcx
    mov qword [rbp-10H], rax
    jmp gcd0w0bend
gcd0w0t:
    mov rcx, [rbp-10H]
    mov rax, [rbp-8H]
    sub rax, rcx
    mov qword [rbp-8H], rax
gcd0w0bend:
    jmp gcd0w
gcd0wend:
    mov rax, [rbp-8H]
    leave
    ret

```

Demo

Built In Functions

List

- listNew();
- listHead(list);
- listTail(list);
- listSet(list, data);
- listAddAfter(list, data);
- listRemove(list, index);
- listConcat (list1, list2);
- listAddLast(list, data);

Page

- pageFetch(“http://www.sample.com/”);
- pageFind(page, “#sample_id”);
- pageURL(page);
- pageHTML(page);
- pageRoot(page);

Element

- elementText(element);
- elementType(element);
- elementAttr(element, “sample”);
- elementChildren(page, element);

Functionality

The built in functions work by communicating over underlying integrational layer with the PhantomJS interpreter that serves as the functionality for all of the easy to use functionalities of web scraping and parsing. That library opens the entire library of css-selector language that allows for easy selection of information to be collected from the parsed web-page that the developer is visiting.

CSS-Selectors Overview

This is a subset selection language similar to the way regex functioning. Below is just a tiny sample of the array of data selection available.

- “*” - Selects all elements.
- “.class” - Selects all elements with the given class.
- [name=”value”] - Selects elements that have the specified attribute with a value exactly equal to a certain value.
- “parent > child” - Selects all direct child elements specified by “child” of elements specified by “parent”.

Regression Testing

TEST 1

The first series of tests are basic parsing and variable declarations for syntax only.

TEST 2

Next we introduced string, Page, List and Element and a series of tests were developed for declaration and syntax only.

TEST 3

Next we needed to implement *print* so tests were designed to print integers and strings.

TEST 4

For loop and while loops

TEST 5

The library was becoming functional so this round of tests included Lists, file, and web data

TEST 6

The final rounds of testing involved modifying existing tests to match TED as the language evolved. While we maintained the original language design, syntax and implementation changed as TED became more complex

Future Work

1. Improve language syntax by introducing nested function definitions and better function invocation methods.
2. Introduce syntax for formatting the web-scraped data to shape it in a meaningfully presentable format such as csv and ascii tables as well as mysql insert queries.
3. Remove the dependency on the PhantomJS layer and build the functionality directly into the language.
4. Improve syntax compromises that were made due to implementation such as declaring all variables prior to function calls, improving readability of built-in functions, etc.

Questions?