LAPORAN TUGAS PEMROGRAMAN

IF2124 Teori Bahasa Formal dan Otomata

HTML *Checker* dengan Pushdown Automata (PDA)

Oleh:

1. Samy Muhammad Haikal (13522151)
2. Muhammad Roihan (13522152)
3. Chelvadinda (13522154)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

DAFTAR ISI **2**
BAB I **3**
DESKRIPSI PERSOALAN **3**
BAB II **4**
TEORI DASAR **4**
2.1 PDA (Pushdown Automata) 4
2.2 Pengaplikasian PDA pada sintaksis HTML 6
BAB III **7**
IMPLEMENTASI DAN PENGUJIAN **7**
3.1 PDA.py 7
3.2 FileHandler.py 8
3.3 lexer.py 8
3.4 Test Case 9
PEMBAGIAN TUGAS **12**
DAFTAR PUSTAKA **13**

BAB I
DESKRIPSI PERSOALAN

Pada tugas pemrograman kali ini, kami ditugaskan untuk membuat sebuah program pendeteksi error untuk HTML. HTML (Hypertext Markup Language) merupakan bahasa markup yang berperan penting dalam menentukan struktur dan tampilan konten di web. Program yang akan diimplementasikan memiliki tujuan utama untuk memeriksa kebenaran penggunaan nama tag dan atribut dalam struktur HTML. Dalam pengembangan web, ketidaksesuaian atau kesalahan dalam penulisan HTML dapat menyebabkan tampilan yang tidak diinginkan dan masalah aksesibilitas. Oleh karena itu, program ini dapat menjadi hal penting dalam menjamin kualitas dan keberlanjutan proyek web.

Untuk mencapai tujuan tersebut, kami menggunakan konsep Pushdown Automata (PDA) dalam pengembangan program. PDA dalam konteks tugas pemrograman ini, berperan sebagai model untuk mengekspresikan aturan dan struktur yang benar dalam dokumen HTML. Melalui implementasi menggunakan bahasa pemrograman Python, diharapkan dapat memberikan solusi yang efektif dan efisien dalam mendeteksi dan mencegah potensi kesalahan dalam pemrograman HTML. Dengan demikian, program ini tidak hanya memberikan kontribusi pada pengembangan web yang berkualitas, tetapi juga memberikan hasil pemrograman yang akurat dan berkelanjutan.

BAB II
TEORI DASAR

2.1 PDA (Pushdown Automata)

Pushdown Automata (PDA) adalah suatu jenis mesin otomata yang memiliki kemampuan untuk mengenali tata bahasa yang dihasilkan oleh konteks bebas (context-free grammar). PDA dapat diilustrasikan sebagai suatu entitas penyimpanan yang tidak terbatas, yang umumnya direpresentasikan dalam bentuk tumpukan atau stack. Stack ini berfungsi sebagai struktur data yang terlihat seperti tumpukan, di mana setiap elemen diletakkan di atas elemen lainnya. Pentingnya sifat Last In First Out (LIFO) dari stack ini menandakan bahwa elemen yang terakhir dimasukkan akan menjadi elemen pertama yang dikeluarkan. Operasi-operasi kunci pada stack melibatkan pengambilan elemen dengan operasi pop dan penambahan elemen dengan operasi push.

Dalam konteks PDA, stack ini berperan sebagai mekanisme penyimpanan sementara untuk membantu dalam proses pengenalan dan pemrosesan tata bahasa. Proses pengambilan dan penambahan elemen pada stack ini dinyatakan melalui operasi pop dan push, yang menunjukkan pengeluaran elemen dari puncak stack (*top stack*) dan penambahan elemen ke *top stack*. PDA menggunakan stack untuk melacak dan memahami struktur tata bahasa yang dihasilkan oleh *context-free grammar*. Kombinasi dari operasi-operasi stack ini memungkinkan PDA untuk secara efektif mengenali bahasa yang kompleks dan dihasilkan oleh aturan-aturan *context-free grammar* melalui pengelolaan tumpukan elemen sepanjang proses komputasinya. Selain itu, dengan adanya fitur stack maka mesin dapat melakukan beberapa hal sekaligus pada saat melakukan transisi yaitu, membaca simbol input, mem-pop simbol, mem-push simbol, dan ber transisi ke state lain.

PDA memiliki beberapa bagian, yaitu input stream, finite state, mekanisme kontrol yang memungkinkan berpindah dari state 1 ke state lain, initial state, final state, stack, stack symbol. PDA dapat didefinisikan secara formal menjadi 7 tuple dalam bentuk (Q, Σ, Γ, T, i, S, F):

- Q adalah himpunan finite state

- Σ adalah machine alphabet (input symbol)
- Γ adalah himpunan stack symbol
- T adalah himpunan transisi
- i adalah initial state (bagian dari Q)
- S adalah start symbol untuk initial stack
- F adalah himpunan accepted state

Proses transisi pada PDA bisa dinotasikan ke dalam bentuk:
T(p, x, s) = (q, t) dengan:

- p menyatakan current state
- x menyatakan simbol yang dibaca atau ε (string kosong)
- s menyatakan simbol yang siap di-pop (top stack)
- q menyatakan next state
- t menyatakan simbol yang di-push

Karena pada pushdown automata memiliki kemampuan transisi T(p, x, s) = (q, t) ini, maka kita harus membuat perubahan pada diagram transisi • Jika pada FA label dari suatu busur dari satu state ke state lain hanya menyatakan simbol yang dibaca, pada PDA menjadi x, s / t

- x menyatakan simbol yang dibaca
- s menyatakan simbol yang siap di-pop (top stack)
- t menyatakan simbol yang di-push

Variasi transisi pada PDA, yaitu:

- Label 0, 0 / 1 berarti membaca 0, mem-pop 0 dan 1 akan di-push.
- Label 0, 1 / 01 berarti membaca 0, tidak melakukan pop, dan mempush 0 ke dalam Stack (atau: mem-pop 1, mem-push 01).
- Label 0, 1 / ε berarti membaca 0, mem-pop 1 dari dalam Stack, dan tidak melakukan push.
- Label 0, 1 /1 berarti membaca 0, tidak mem-pop, dan tidak mem-push.
- Label ε, 0 / 0 berarti membaca string kosong, tidak mem-pop, dan tidak mem-push.

PDA memiliki 2 cara untuk menentukan apakah suatu input diterima:

- Berakhir di final state (acceptance by final state, seperti FA)

  Mesin akan mulai di initial state dengan initial stack (berisi satu start symbol), kemudian mesin akan menerima jika string memungkinkan mesin menjangkau accepted state setelah membaca semua string. Hal ini tidak mewajibkan stack di akhir harus kosong.

- Stack kosong (acceptance by empty stack)

  Mesin akan mulai di initial state dengan initial stack (berisi satu start symbol). Mesin tersebut akan menerima jika setelah membaca semua string, dimana stack di akhir harus kosong.

2.2 Pengaplikasian PDA pada sintaksis HTML

HTML merupakan bahasa dengan aturan tertentu yang harus dipahami oleh suatu mesin. PDA (Pushdown Automata) dapat membantu untuk memodelkan cara mesin tersebut dapat mengenali dan memproses struktur HTML. Pertama, kita dapat memandang setiap elemen HTML, seperti <html>, <head>, dan lainnya, sebagai "kondisi" pada PDA. Masing-masing kondisi ini mencerminkan elemen HTML yang sedang diproses. Alfabet PDA terdiri dari simbol-simbol terminal yang mewakili tag HTML, atribut, dan teks yang dapat ditemukan dalam HTML.

Selanjutnya, setiap transisi pada PDA mencerminkan perpindahan dari satu kondisi ke kondisi lainnya berdasarkan simbol dari alfabet. Sebagai contoh, jika PDA sedang berada di kondisi <html> dan menerima tag pembuka <head>, maka PDA dapat berpindah ke kondisi <head>. PDA juga menggunakan konsep tumpukan untuk menyimpan informasi tambahan. Misalnya, untuk memastikan bahwa tag dibuka dan ditutup dengan benar. PDA dapat menggunakan tumpukan untuk memantau tag yang telah dibuka dan mempop tumpukan ketika tag ditutup.

BAB III
IMPLEMENTASI DAN PENGUJIAN

3.1 PDA.py

Program ini merupakan implementasi Pushdown Automaton (PDA) dalam bahasa pemrograman *python*. PDA digunakan untuk mengenali apakah suatu string masukan dapat diterima oleh mesin yang dijelaskan oleh aturan-aturan pada PDA tersebut. Berikut penjelasan bagian-bagian dari program ini.

Fungsi dan Prosedur	Spesifikasi Teknik Program
def __init__(self, states, alphabet, stack_alphabet, initial_state, initial_stack_symbol, transitions, final_states):	Menginisialisasi PDA seperti states, alphabet, stack alphabet, initial state, initial stack symbol, transitions, final states, dan stack (tumpukan).
def process_input(self, inputString):	Metode untuk memproses string masukan dengan menggunakan PDA. Untuk setiap simbol, mencoba mencocokkan transisi yang sesuai. Jika ditemukan yang sesuai maka akan melakukan perubahan kondisi dan tumpukan sesuai dengan aturan transisi. Jika tidak ditemukan transisi yang sesuai, mengembalikan False karena string ditolak. Setelah selesai membaca semua simbol, kemudian memeriksa apakah tumpukan kosong dan kondisi saat ini adalah kondisi akhir. Jika kondisi akhir, mengembalikan True (string diterima), jika tidak merupakan kondisi akhir, mengembalikan False.
def main():	Membaca File PDA dan string masukan dengan menggunakan FileHandler dan lexer. Kemudian membuat PDA dari aturan-aturan FileHandler dan memproses string masukan menggunakan PDA. Setelah itu menampilkan hasil apakah string diterima atau tidak.

3.2 FileHandler.py

Program ini merupakan penanganan file dan penguraian (parsing) informasi dari file PDA (Pushdown Automaton) yang mungkin digunakan sebagai konfigurasi untuk mesin PDA. Berikut penjelasan bagian-bagian dari program ini.

Fungsi dan Prosedur	Spesifikasi Teknik Program
def __init__(self):	Menyertakan "pass" yang menandakan tidak ada operasi yang perlu dilakukan. Hal ini berguna untuk mencegah adanya kesalahan sintaksis saat blok kode kosong.
def readFile(self, filePath):	Membaca isi file yang diberikan oleh filePath. Jika file tersebut ada, maka akan membaca setiap baris dari file. Jika file tidak dapat dibuka dan tidak dapat ditemukan, maka akan mencetak pesan kesalahan dan keluar dari program. Fungsi ini akan mengembalikan list yang berisi setiap baris dari file.
def parseFile(self,lines):	Fungsi ini melakukan penguraian (parsing) terhadap isi file yang telah dibaca. Setiap informasi dari file diuraikan. Informasi tersebut kemudian disusun ke dalam sebuah dictionary (parsedLines). Dictionary tersebut berisi informasi seperti, states, input symbols, stack symbols, starting state, starting stack, accepting states, accept by, dan list of production. Kemudian akan mengembalikan informasi yang dapat digunakan untuk menginisialisasi objek PDA dalam program utama.

3.3 lexer.py

Program ini digunakan untuk memproses pada suatu teks. Program ini berisi fungsi-fungsi atau aturan-aturan token yang digunakan untuk membentuk token dari suatu string masukan. Berikut penjelasan bagian-bagian dari program ini.

Dalam tubes ini kami merubah setiap tag HTML dan atribut dalam bentuk token-token agar memudahkan dalam mengimplementasikan dalam bentuk PDA.

Fungsi dan Prosedur	Spesifikasi Teknik Program
def lex(text, token_rules):	Fungsi lex pada kode ini menerima dua parameter, yaitu text yang merupakan string yang akan dianalisis, dan aturan dari token. Fungsi ini bertugas melakukan analisis terhadap string text berdasarkan aturan-aturan yang telah ditentukan. Proses tersebut mencocokkan setiap karakter dengan aturan token. Jika suatu pola cocok, token akan ditambahkan ke dalam list token. Namun, jika tidak ada aturan token yang cocok, program akan mencetak pesan kesalahan dan keluar dari program. Fungsi ini akan mengembalikan list tokens yang berisi token-token yang berhasil ditemukan selama proses analisis dan membentuk token dari string masukan sesuai dengan aturan-aturan yang telah ditentukan.
def createToken(text):	Fungsi ini membaca isi file, kemudian memanggil fungsi lex untuk melakukan proses analisis pada teks. Fungsi ini akan mengembalikan list tokenResult yang berisi token-token yang dihasilkan dari proses analisis tersebut.

3.4 Test Case

Berikut beberapa percobaan sebuah kasus dan analisisnya:

Contoh Test Case	Analisis

	Rejected Tag heading <h1> tidak ditutup dengan benar. Sehingga input ini akan ditolak (reject) karena adanya kesalahan penutupan tag-heading <h1>. Dalam HTML yang benar, tag-heading harus dibuka dengan <h1> dan ditutup dengan </h1>. Jadi, string HTML yang diberikan tersebut dapat menyebabkan kesalahan dalam proses analisis dan dianggap tidak valid dari segi sintaksis HTML, sehingga tidak dapat diterima.
	Accepted Berdasarkan struktur HTML yang diberikan, string HTML ini mematuhi aturan-aturan dasar HTML dan tidak memiliki kesalahan sintaksis. Oleh karena itu, dengan menggunakan PDA yang telah diimplementasikan untuk mengenali struktur HTML dengan aturan-aturan yang diberikan, string HTML ini dapat diterima (accept).

<table>
<tr>
<td>

```
<!-- ojojdaojadojad -->
<html>
    <!-- ijiji-->
    <head class="kokok">
        <title class="loksok" id="kkkk" style="oaoaoao">
        <small></small>
        <strong></strong>
        <em></em>
        <!-- lkdskka

        -->
        </title>
        <script src="ioo"></script>
    </head>
    <!-- inininini -->
    <body>
        <h1></h1>
        <h2>sadad dsfsdfdfs</h2>
        <h3></h3>
        <p>vdvd<br>dffvd</p>
        <button type="reset" class="oko"></button>
        <button></button>
```

```
        <link class="ooo" id="jiiji" rel="stylesheet">
        <b></b>
        <b></b>
        <script src="okoko"></script>
        <img src="./welcome.jpeg">
        <br class="okok">
        <table>
            <tr>
            </tr>
        </table>
        <!-- khlidahihs -->
        <h2></h2>
    </body>
    <!-- ijiij -->
</html>
<!-- ijijijij -->
```

</td>
<td>Accepted

String HTML yang diberikan akan diterima (accept) karena sesuai dengan aturan token yang telah diimplementasikan, seperti pembukaan dan penutupan tag, atribut-atribut, serta penggunaan elemen-elemen HTML yang diizinkan. Meskipun terdapat komentar-komentar dalam HTML, komentar tersebut tidak mempengaruhi proses analisis sintaksis HTML tersebut. Oleh karena itu, string HTML tersebut dapat diterima oleh PDA yang telah diimplementasikan.</td>
</tr>
<tr>
<td>

```
<hmif>
  <head>
    <title>Simple Webpage</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a simple webpage.</p>
  </body>
</hmif>
```

</td>
<td>Rejected

Tidak token yang mentokenisasi <hmif> sehingga file HTML tersebut akan ditolak oleh program.</td>
</tr>
</table>

PEMBAGIAN TUGAS

Nama / NIM	Tugas
Samy Muhammad Haikal / 13522151	Token, PDA, Diagram
Muhammad Roihan / 13522152	Parser, PDA, Diagram
Chelvadinda / 13522154	PDA, Laporan, Diagram

DAFTAR PUSTAKA

Link Repository Github: https://github.com/mroihn/Tubes_TBFO
Link Diagram State: https://app.diagrams.net/#G123KgYKACz9E_MSSxy9ZaQ1p-8ydImuDh
Link Alternatif Diagram :
https://drive.google.com/file/d/123KgYKACz9E_MSSxy9ZaQ1p-8ydImuDh/view?usp=sharing

Jain, S. (2022, June 8). *Introduction of Pushdown Automata*. GeeksforGeeks. Retrieved November 23, 2023, from https://www.geeksforgeeks.org/introduction-of-pushdown-automata/

Pushdown Automata: Theory, Diagram, Types. (n.d.). StudySmarter. Retrieved November 26, 2023, from https://www.studysmarter.co.uk/explanations/computer-science/theory-of-computation/pushdown-automata/