LAPORAN TUGAS BESAR
TEORI BAHASA DAN AUTOMATA

PARSER SEDERHANA UNTUK MEMERIKSA KEVALIDAN STRUKTUR KALIMAT BERBAHASA INDONESIA

Disusun Oleh:
IF 46-09
Sherly Angelina (1301223370)
Fadhilah Kartika Firdausi (1301223180)
Ellisa Tamara Yulianti P (1301223420)

FAKULTAS INFORMATIKA
S1 INFORMATIKA
UNIVERSITAS TELKOM
2024

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bahasa merupakan alat komunikasi utama yang digunakan manusia dalam kehidupan sehari-hari. Salah satu aspek penting dalam bahasa adalah struktur kalimat yang membentuk suatu kalimat yang bermakna dan dapat dipahami dengan baik. Dalam bahasa Indonesia, struktur kalimat memiliki aturan-aturan tertentu yang harus diikuti agar kalimat tersebut dapat dianggap valid dan bermakna.

Kalimat berita aktif merupakan salah satu jenis kalimat yang paling sering digunakan dalam bahasa Indonesia, baik dalam percakapan sehari-hari maupun dalam tulisan formal. Struktur kalimat berita aktif ini terdiri dari subjek (S), predikat (P), objek (O), dan keterangan (K). Meskipun aturan pembentukan kalimat berita aktif terdengar sederhana, namun pada praktiknya sering kali ditemukan kesalahan-kesalahan dalam penyusunan kata-kata menjadi sebuah kalimat yang valid.

1.2 Rumusan Masalah

Permasalahan Permasalahan yang dihadapi dalam tugas besar ini adalah bagaimana merancang dan mengimplementasikan sebuah parser yang dapat mengenali struktur kalimat berita aktif dalam bahasa Indonesia dengan benar. Struktur kalimat yang perlu dikenali adalah:

- S – P – O – K
- S – P – K
- S – P – O
- S – P

Di mana S, P, O, dan K masing-masing merupakan kelompok kata yang mewakili Subjek, Predikat, Objek, dan Keterangan. Setiap kelompok kata terdiri dari 5 kata yang telah ditentukan.

1.3 Tujuan

Tujuan laporan ini dibuat adalah untuk memenuhi tugas besar Teori Bahasa dan Automata yang ditugaskan untuk membangun sebuah parser sederhana yang dapat memeriksa kevalidan struktur kalimat berita aktif dalam bahasa Indonesia.

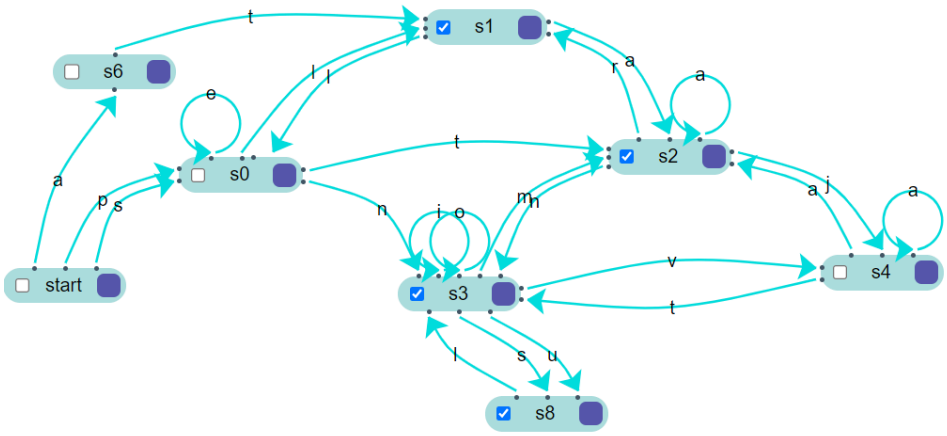
BAB II
PEMBAHASAN

Pendekatan Solusi Untuk menyelesaikan permasalahan ini, kami menggunakan pendekatan kombinasi Finite Automata (FA) dan Pushdown Automata (PDA). FA digunakan untuk membangun sebuah token recognizer yang dapat mengenali setiap kata masuk ke dalam kelompok S, P, O, atau K. Sedangkan PDA digunakan untuk membangun parser yang dapat mengenali apakah kalimat yang diinputkan valid berdasarkan struktur yang disebutkan di atas.

2.1 Finite Automata (FA)

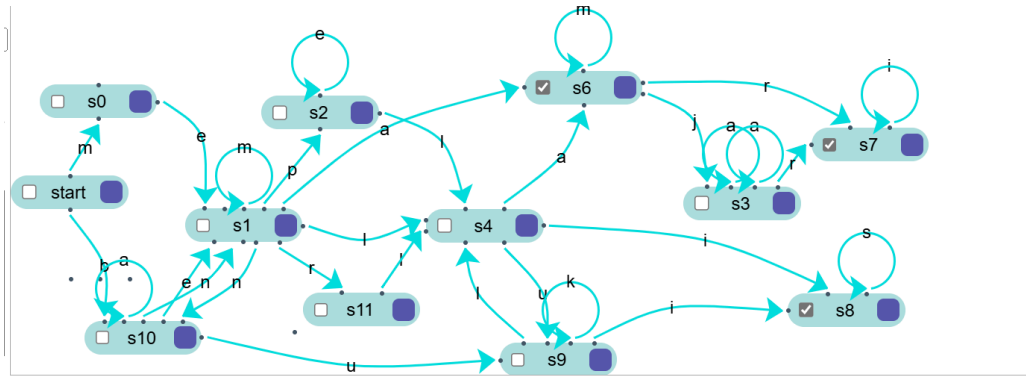
Setiap FA dirancang sedemikian rupa sehingga dapat mengenali kata-kata yang termasuk dalam kelompok tersebut. Berikut adalah rancangan FA dari masing masing kelompok S, P, O, dan K.

Kelompok S:




```
Test Results:
Accept: pelajar -- Pass
Accept: seniman -- Pass
Accept: atlet -- Pass
Accept: penulis -- Pass
Accept: petani -- Pass
Reject: [Empty String] -- Pass
```

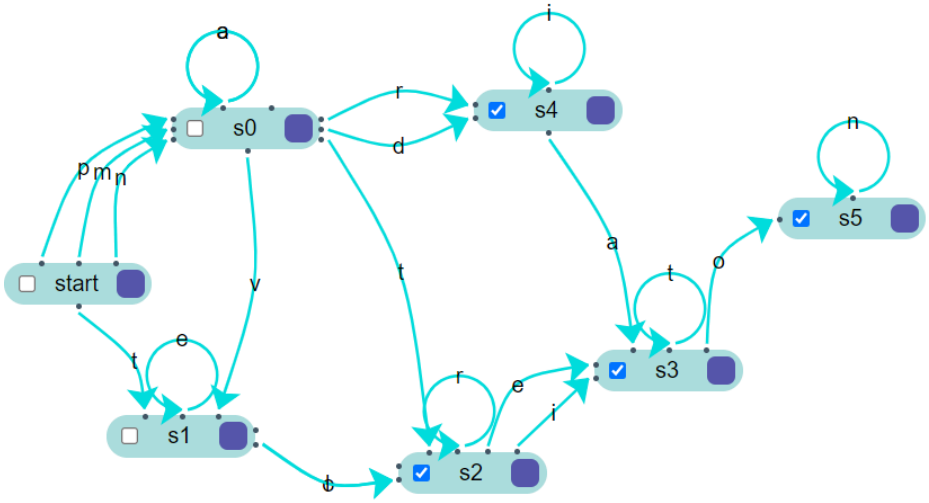
Kelompok P:



Test Results:

Accept: mempelajari -- Pass
Accept: menulis -- Pass
Accept: melukis -- Pass
Accept: berlari -- Pass
Accept: menanam -- Pass

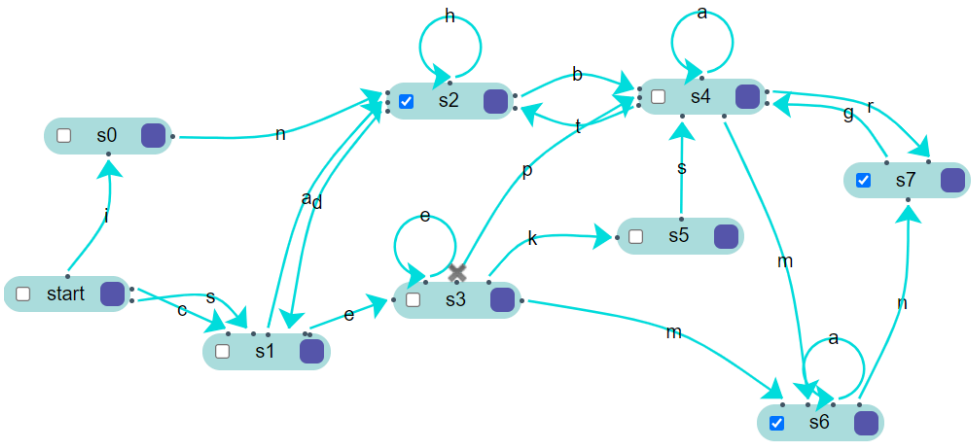
Kelompok O:



Test Results:

Accept: teori -- Pass
Accept: potret -- Pass
Accept: maraton -- Pass
Accept: novel -- Pass
Accept: padi -- Pass
Reject: [Empty String] -- Pass

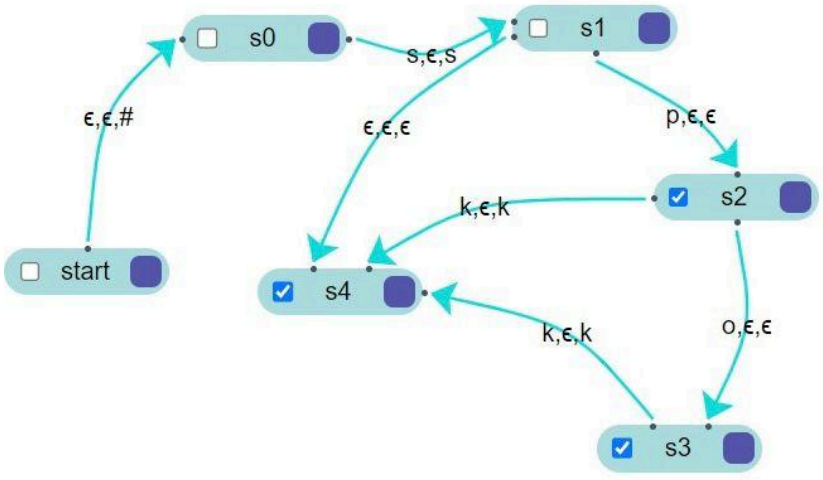
Kelompok K:



Test Results:

Accept: seksama -- Pass
Accept: indah -- Pass
Accept: cepat -- Pass
Accept: semangat -- Pass
Accept: sabar -- Pass
Reject: [Empty String] -- Pass

2.2 Pushdown Automata (PDA)



Test Results:

Accept: spok -- Pass
Accept: spo -- Pass
Accept: spk -- Pass
Accept: sp -- Pass
Reject: [Empty String] -- Pass

2.3 Context Free Grammar (CFG)

Selanjutnya, kami merancang Context Free Grammar (CFG) untuk struktur kalimat yang valid. CFG ini akan digunakan sebagai dasar untuk membangun parser menggunakan PDA. Berikut adalah rancangan CFG yang kami buat:

Aturan CFG

- S → FN FK
- FN → S
- FK → V FN | V FN FP | V FP | V
- FP → K

Di mana:

- FN (Frasa Nomina) adalah Subjek (S)
- FK (Frasa Kata Kerja) adalah Predikat (P), Predikat (P) + Objek (O), atau Predikat (P) + Keterangan (K)
- FP (Frasa Preposisi) adalah Keterangan (K)

2.4 Token Recognizer dan Parser

Setelah rancangan FA dan CFG telah dibuat, kami mengimplementasikan token recognizer dan parser menggunakan bahasa pemrograman Python. Token recognizer dibangun berdasarkan rancangan FA yang telah dibuat, sedangkan parser dibangun menggunakan PDA yang mengacu pada CFG yang telah dirancang sebelumnya.

tubes.py > main

```python
class StateFAMachine:
    def __init__(self, state):
        # Menginisialisasi dengan daftar status dan status awal
        self.state = state
        self.currentState = 0

    def reset(self):
        # Mengatur ulang status ke awal
        self.currentState = 0

    def processChar(self, char):
        # Memproses setiap karakter dalam kata
        possible_next_state = []
        for state in self.state:
            # Memeriksa apakah karakter saat ini cocok dengan karakter pada posisi saat ini dalam salah satu status yang valid
            if len(state) > self.currentState and state[self.currentState] == char:
                possible_next_state.append(state)

        if not possible_next_state:
            # Jika tidak ada status yang cocok, kembalikan False
            return False

        # Memperbarui daftar status yang valid dan menaikkan status saat ini
        self.state = possible_next_state
        self.currentState += 1
        return True

    def isAccepted(self):
        # Memeriksa apakah kata yang sedang diproses cocok dengan salah satu status yang valid
        return any(len(state) == self.currentState for state in self.state)

```

tubes.py > main

```python
def tokenize(sentence):
    # Definisi set untuk masing-masing kategori kata
    subjek = {'pelajar', 'seniman', 'atlet', 'penulis', 'petani'}
    predikat = {'mempelajari', 'menulis', 'melukis', 'berlari', 'menanam'}
    objek = {'teori', 'potret', 'maraton', 'novel', 'padi'}
    keterangan = {'seksama', 'indah', 'cepat', 'semangat', 'sabar'}

    # Gabungkan semua kategori dalam satu set untuk pengecekan
    all_words = subjek | predikat | objek | keterangan

    # Memisahkan kalimat menjadi token
    tokens = sentence.split()
    recognizedTokens = []

    # Mengenali setiap token
    for token in tokens:
        sm = StateFAMachine(all_words)  # Membuat instansi StateMachine baru untuk setiap token
        for char in token:
            if not sm.processChar(char):
                return False, []  # Token tidak dikenali, kembalikan False dan daftar kosong

        if sm.is_accepted():
            # Menambahkan jenis token yang dikenali ke daftar recognized_tokens
            if token in subjek:
                recognizedTokens.append('S')  # Tambahkan 'S' jika token adalah subjek
            elif token in predikat:
                recognizedTokens.append('P')  # Tambahkan 'P' jika token adalah predikat
            elif token in objek:
                recognizedTokens.append('O')  # Tambahkan 'O' jika token adalah objek
            elif token in keterangan:
                recognizedTokens.append('K')  # Tambahkan 'K' jika token adalah keterangan
        else:
            return False, []  # Token tidak dikenali, kembalikan False dan daftar kosong

    return True, recognizedTokens  # Kembalikan True dan daftar recognized_tokens
```

tubes.py > main

```python
def parse_sentence(sentence):
    # Memanggil fungsi tokenize untuk mengenali token
    isValid, token = tokenize(sentence)
    if not isValid:
        return False, []  # Jika kalimat tidak valid, kembalikan False dan daftar kosong

    # Definisikan struktur kalimat yang valid
    validStructures = [
        ['S', 'P', 'O', 'K'],  # Struktur S-P-O-K
        ['S', 'P', 'K'],       # Struktur S-P-K
        ['S', 'P', 'O'],       # Struktur S-P-O
        ['S', 'P']             # Struktur S-P
    ]

    # Memeriksa apakah urutan token sesuai dengan salah satu struktur yang valid
    if token in validStructures:
        return True, token  # Jika sesuai, kembalikan True dan urutan token
    else:
        return False, token  # Jika tidak sesuai, kembalikan False dan urutan token
```

tubes.py > ...

```python
def main():
    while True:
        # Menerima input kalimat dari pengguna
        sentence = input("Masukkan kalimat (atau ketik 'keluar' untuk berhenti): ")

        # Keluar dari loop jika pengguna mengetik 'keluar'
        if sentence.lower() == 'keluar':
            break

        # Memeriksa validitas kalimat menggunakan parse_sentence
        isValid, token = parse_sentence(sentence)

        # Menampilkan hasil pemeriksaan
        if isValid:
            print(f"Kalimat valid: {sentence}")
            print(f"Struktur token: {token}")
        else:
            print(f"Kalimat tidak valid: {sentence}")
            print(f"Struktur token: {token}")

# Menjalankan fungsi utama
if __name__ == "__main__":
    main()
```

Berikut output untuk program diatas untuk mengecek apakah kalimat yang di input termasuk ke dalam struktur S-P-O-K, S-P-O, S-P-K, atau S-P

```
PS D:\TUBES TBA> & C:/Users/Sherg/AppData/Local/Programs/Python/Python312/python.exe "d:/TUBES TBA/tubes.py"
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): penulis menulis novel indah
Kalimat valid: penulis menulis novel indah
Struktur token: ['S', 'P', 'O', 'K']
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): pelajar mempelajari teori
Kalimat valid: pelajar mempelajari teori
Struktur token: ['S', 'P', 'O']
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): atlet belari
Kalimat tidak valid: atlet belari
Struktur token: []
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): atlet melukis
Kalimat valid: atlet melukis
Struktur token: ['S', 'P']
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): atlet berlari
Kalimat valid: atlet berlari
Struktur token: ['S', 'P']
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): petani menanam semangat
Kalimat valid: petani menanam semangat
Struktur token: ['S', 'P', 'K']
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): seniman makan
Kalimat tidak valid: seniman makan
Struktur token: []
Masukkan kalimat (atau ketik 'keluar' untuk berhenti): keluar
PS D:\TUBES TBA>
```

BAB III

PENUTUP

3.1 Kesimpulan

Kami telah menguji parser yang telah dibuat dengan berbagai kalimat berita aktif dalam bahasa Indonesia. Hasilnya menunjukkan bahwa parser dapat mengenali dengan baik struktur kalimat yang valid sesuai dengan struktur yang telah ditentukan. Namun, parser juga menunjukkan beberapa kelemahan, seperti tidak dapat mengenali kalimat yang memiliki struktur yang sedikit berbeda dari struktur yang telah ditentukan.

3.2 Link

Tautan GDrive :

TUGAS BESAR TBA IF-46-09 (1301223370, 1301223180, 1301223420)