**Pertemuan 9 – Penyederhanaan Tata bahasa bebas konteks**

Tujuan penyederhanaan untuk melakukan pembatasan sehingga tidak menghasilkan pohon penurunan yang memiliki kerumitan yang tidak perlu (dengan simbol awal) S

**S > AB | a**

**A > a**

Kelemahan tata Bahasa Bebas Konteks di atas aturan produksi **S > AB** tidak berarti karena B tidak memiliki pemilihan / penurunan

Suatu tata bahasa bebas konteks dapat disederhanakan dengan :

1. Penghilangan Produksi useless
   1. Produksi yang memuat simbol variabel yang **tidak memiliki penurunan** yang akan menghasilkan terminal selanjutnya.
   2. Produksi yang **tidak akan pernah dicapai dengan penurunan apapun** dari simbol awal (Redundan)

Contoh:

**S > aSa | Abd | Bde** Penyederhanaan

**A > Ada** **S > aSa | Bde**

**B > BBB | a B > BBB | a**

CFG yang useless :

**A > AdA**

1. Penghilangan Produksi Unit
   1. Produksi unit yaitu produksi dimana ruas kiri dan ruas kanan aturan produksinya hanya berupa satu simbol variabel

**A > B**

**C > D**

Contoh :

**S > Sb**

**S > C**

**C > D**

**C > ef**

**D > dd**

Kita lakukan pergantian berurutan mulai dari aturan produksi yang paling dekat (paling bawah) menuju terminal

**C > D => C > dd**

**S > C => S > dd | ef**

Sehingga aturan produksi setelah penyederhanaan adalah :

**S > Sb**

**S > dd | ef**

**C > dd**

**C > ef**

**D > dd**

1. Penghilangan produksi empty

**S > bcAd S > bcAd**

**A > bd | empty A > bd**

**Pertemuan 10 – Bentuk Normal Chomsky**

Bentuk Normal Chomsky atau CFG sangat berguna untuk tata bahasa bebas konteks

Bentuk Normal Chomsky dapat dibuat / dibentuk dari sebuah tata bahasa yang **sudah mengalami penyederhanaan.**

Suatu CFG dapat dibuat menjadi Bentuk Normal Chomsky dengan syarat :

1. Tidak memiliki produksi useless
2. Tidak memiliki produksi unit
3. Tidak memiliki produksi empty

Aturan produksi dalam bentuk normal chomsky ruas kanannya tepat berupa **sebuah termina**l / **dua buah variabel**. Contoh :

**A > BC**

**A > b**

**B > a**

**C > BA | d**

Contoh : (kita anggap sudah mengalami penyederhanaan)

**S > bA | aB**

**A > bAA | aS | a**

**B > aBB | bS | b**

Aturan yang sudah termasuk bentuk normal chomsky

**A > a**

**B > b**

Lakukan penggantian aturan produksi yang belum normal chomsky

**=> artinya menjadi**

Lakukan pergantian menggunakan inisialisasi / inisial P1, P2, P3, ... , Pn

untuk **1 buah terminal** / **2 buah variabel**

**S > bA** => **S > P1A**

**S > aB** => **S > P2B**

**A > bAA** => **A > P1AA** => **A > P1P3**

**A > aS** => **A > P2S**

**B > aBB** => **B > P2BB** => **B > P2P4**

**B > bS** => **B > P1S**

Hasil Akhir Aturan produksi baru

**b = P1 A > a P1 > b**

**a = P2 B > b P2 > a**

**AA = P3 S > P1A P3 > AA**

**BB = P4** **S > P2B P4 > BB**

**A > P1P3**

**A > P2S**

**B > P2P4**

**B > P1S**

**Pertemuan 11 – Penghilangan Rekursif kiri**

Tujuan : Untuk menghindari penurunan yang bisa menghasilkan loop

Tahap penghilangan rekursif kiri :

1. **Pisahkan aturan produksi yang rekursif kiri dan tidak rekursif kiri.**

Aturan produksi yang rekursif kiri

Simbol variabel di paling kiri

**A > Aα1 | Aα2 | Aα3 | ... | Aαn**

Aturan yang **tidak** rekursif kiri

**A > β1 | β2 | β3 | ... | βn**

1. **Tentukan alpha dan beta**

**α1,α2,α3,αn**

dan

**β1,β2,β3,βn**

1. **Lakukan penggantian**

A > β1Z | β2Z | ... | βnZ

Z > α1 | α2 | α3 | αn

Z > α1Z | α2Z | α3Z

Contoh :

**S > Sab | aSc | dd | ff | Sbd**

1. Aturan rekursif kiri

**S > Sab | Sbd**

Aturan yang bukan rekursif kiri

**S > aSc | dd | ff**

1. Rekursif kiri => α1

**α1 = ab**

**α2 = bd**

Bukan rekursif kiri => β

**β1 = aSc**

**β2 = dd**

**β3 = ff**

1. **Lakukan pergantian rekursif kiri**

**S > Sab | Sbd menjadi**

**A > aScZ1 | ddZ1 | ffZ1**

**Z1 > ab | bd**

**Z1 > abZ1 | bdZ1**

1. **Hasil akhir**

**S > aSc | dd | ff**

**A > aScZ1 | ddZ1 | ffZ1**

**Z1 > ab | bd**

**Z1 > abZ1 | bdZ1**

**Pertemuan 12 – Bentuk Normal Greibach**

Suatu CFG dalam bentuk normal Greibach bila hasil produksinya (ruas kanan) di awali dengan satu simbol terminal, selanjutnya diikuti oleh rangkaian simbol variabel :

**S > a | aB**

**A > aB**

**B > cS**

Untuk dapat diubah ke dalam bentuk normal greibach, syaratnya adalah :

1. Tidak dalam bentuk normal chomsky
2. Tidak bersifat rekursif kiri
3. Tidak menghasilkan empty

Pembentukan bentuk normal greibach dengan substitusi :

**S > CA  
A > a | d**

**B > b**

**C > DD**

**D > AB**

1. Tentukan urutan variabel

**S, A, B, C, D => (S < A < B < C < D)**

1. Periksa aturan produksi simbol pertama pada ruas kanan adalah variabel

**S > CA (sudah memenuhi, karena S < C)**

**C > DD (sudah memenuhi, karena C < D)**

**D > AB (tidak memenuhi, karena D > A)**

1. Lakukan substitusi untuk **D > AB** (tidak memenuhi syarat)

**A** menurunkan **a | d**

maka **D > AB** diubah menjadi **D > aB | dB**

1. Lakukan substitusi mundur pada aturan yang dalam bentuk greibach (mulai dari variabel paling akhir dalam urutan)

**C > DD** menjadi **C > aBD | dBD**

**S > CA** menjadi **S > aBDA | dBDA**

1. Hasil Akhir

**S > aBDA | dBDA  
A > a | d**

**B > b**

**C > aBD | dBD**

**D > aB | dB**

**Bentuk Normal Greibach dengan rekursif kiri**

**A > BC**

**B > CA | b**

**C > AB | a**

1. Tentukan simbol urutan

**A, B, C (A < B < C)**

1. Periksa aturan

**A > BC (A < B)**

**B > CA | b (B < C)**

**C > AB | a (C > A)**

1. Lakukan Substitusi untuk aturan produksi yang tidak memenuhi syarat

**C > AB**

**C > AB => C > BCB => C > CACB | bCB**

**CACB** merupakan rekursif kiri karena penurunannya menghasilkan loop variabel (tidak bisa mencapai terminal pada ruas paling kiri

maka untuk **C > CACB** dilakukan penghilangan rekursif kiri

**C > CACB | bCB | a**

**α1 = ACB**

**β2 = bCB**

**β2 = a**

Lakukan penghilangan rekursif kiri

**C > bCBZ1 | aZ1**

**Z1 > ACB**

**Z1 > ACBZ1**

Hasil akhir penghilangan rekursif kiri

**C > bCBZ1 | aZ1 | bCB | a**

Keterangan :   
**bCBZ1 | aZ1** diambil dari hasil penghilangan rekursif kiri

**bCB | a** diambil dari hasil produksi yang tidak dalam kondisi rekursif

Setelah sesuai, maka lakukan substitusi mundur (dimulai dari variabel paling bawah)

**B > CA** menjadi **B > bCBZ1A| aZ1A | bCBA | aA**

**A > BC** menjadi **A > bCBZ1AC| aZ1AC | bCBAC | aAC | bC**

Jangan lupa lakukan substitusi untuk aturan produksi baru yang terbentuk dari **penghilangan rekursif kiri** tadi yaitu Z1

**Z1 > ACB** menjadi **bCBZ1ACCB| aZ1ACCB | bCBACCB | aACCB | bCCB**

**Z1 > ACBZ1** menjadi **bCBZ1ACCBZ1 | aZ1ACCBZ1 | bCBACCBZ1 | AacCBZ1 | bCCBZ1**

Hasil Akhir

**A > bCBZ1AC| aZ1AC | bCBAC | aAC | bC**

**B > bCBZ1A| aZ1A | bCBA | aA**

**C > bCBZ1 | aZ1 | bCB | a**

**Z1 > ACB menjadi bCBZ1ACCB| aZ1ACCB | bCBACCB | aACCB | bCCB**

**Z1 > ACBZ1 menjadi bCBZ1ACCBZ1 | aZ1ACCBZ1 | bCBACCBZ1 | AacCBZ1 | bCCBZ1**

**Pertemuan 13 – Push Down Automata**

Push Down Automata dinyatakan dalam 7 tupel

M = {Q, ∑, r, Δ, S, F, Z)

Q = himpunan state

∑ = himpunan simbol input

r = simbol simbol tumpukan / stack

Δ = fungsi transisi

S = state awal, S ∑ Q

F = himpunan final state, F ≤ Q => f

Z = simbol awal tumpukan / top stack

Contoh sebuah PDA :

Q = {q1, q2}

∑ = {a, b}

S = q1

f = {q2}

Z = Z

PDA tersebut memiliki fungsi transmisi sebagai berikut :

Δ(q1, empty, Z) = {(q2, Z)}

Δ(q1, a, Z) = {(q1, AZ)}

Δ(q1, b, Z) = {(q1, BZ)}

Δ(q1, a, A) = {(q1, AA)}

Δ(q1, b, A) = {(q1, empty)}

Δ(q1, a, B) = {(q1, empty)}

Δ(q1, b, B) = {(q1, BB)}

Misal ingin diketahui apakah string ‘abba’ diterima oleh PDA ini

|  |
| --- |
| Z |
|  |

1. konfigurasi awal mesin state **q1** dengan top stack **Z** dan membaca input **‘a’**

Fungsi transisinya **Δ(q1, a, Z) = {(q1, AZ)}**

konfigurasi mesinnya menjadi :

state saat ini **q1** dengan **A dipush** karena **hasil dari fungsi transisinya** adalah **AZ**

|  |
| --- |
| A |
| Z |
|  |

=> PUSH A

1. **Δ(q1, b, A) = {(q1, empty)}** > POP A karena empty

|  |
| --- |
| Z |
|  |

1. **Δ(q1, b, Z) = {(q1, BZ)}** PUSH B karena BZ

|  |
| --- |
| B |
| Z |
|  |

=> PUSH B

1. **Δ(q1, a, B) = {(q1, empty)}** POP B karena empty

|  |
| --- |
| Z |
|  |

1. **Δ(q1, empty, Z) = {(q2, Z)},** final di q2 (string ‘abba’ diterima)

|  |
| --- |
| Z |
|  |

**Pertemuan 14 & 15 – Mesin Turing**

Mesin Turing bersifat umum, dan memiliki kemampuan lebih tinggi dari pada Finite State Automata maupun PDA dari segi aksi dan komponennya.

Pada mesin Turing, memori akan berupa sebuah pita yang pada dasarnya berupa Array (deretan) sel-sel penyimpanan.

Terdapat sebuah head yang menunjukkan posisi yang diakses pada pita

Head dapat bergerak ke kiri ataupun ke kanan untuk membaca input dari pita yang sekaligus juga bisa melakukan penulisan pada pita / mengubah isi pita

Mesin turing terdiri dari 7 tupel

Q = himpunan state

∑ = himpunan simbol input

r = simbol pada pita (meliputi pula blank)

δ = fungsi transisi

S = state awal

F = himpunan state akhir

~~b~~ = simbol kosong (blank) (bukan bagian dari ∑)

Bagian pada pita yang belum ditulisi dianggap berisi simbol ~~b~~ (blank)

Contoh:

Q = {q1, q2}

∑ = {a, b}

r = {a, b, ~~b~~}

S = {q1}

F = {q2}

Fungsi transisinya

δ(q1,a) = (q1,a,R) R = Right / kanan

δ(q1,b) = (q1,a,R) L = Left / kiri

δ(q1,~~b~~) = (q2,~~b~~,L)

Cara mesin turing beroperasi :

Misalkan pita yang akan dibaca adalah ‘abbaa’ dan warna kuning menandakan state q1, warna hijau menandakan state q2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | b | b | a | a |  |

1. Fungsi transisi δ(q1,a) = (q1,a,R) menyebabkan head bergerak ke kanan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | b | b | a | a |  |

1. Fungsi transisi δ(q1,b) = (q1,a,R) menyebabkan head menulis ‘a’ dan bergerak ke kanan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | a | a | a | a |  |

1. Fungsi transisi δ(q1,a) = (q1,a,R) menyebabkan head bergerak ke kanan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | a | a | a | a |  |

1. Fungsi transisi δ(q1,a) = (q1,a,R) menyebabkan head bergerak ke kanan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | a | a | a | a |  |

1. Fungsi transisi δ(q1,a) = (q1,a,R) menyebabkan head bergerak ke kanan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | a | a | a | a | ~~b~~ |  |

1. Head menunjuk ~~b~~, karena bagian pita belum ditulisi maka dianggap berisi ~~b~~

Fungsi transisi δ(q1,~~b~~) = (q2,~~b~~,L) menyebabkan head bergerak ke kiri

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | a | a | a | a | ~~b~~ |  |

Tidak ada transisi lagi dari state q2 (warna hijau), mesin turing akan berhenti (halt state),karena state q2 adalah state akhir maka input tersebut diterima