Push Down Automata (2)

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by: Rahmad Mahendra (2015)

Updated by: Maya Retno Ayu S (2016)
Suryana Setiawan (2016-2020)

Audio by: Suryana Setiawan (2020)

Materi dan Tujuan Pembahasan

- Di bagian ini akan dibahas ekivalensi antara CFG dan PDA dengan menunjukkan adanya algoritma-algoritma:
  - konversi dari CFG menjadi PDA
  - Konversi dari PDA menjadi CFG
- Catatan:
  - algoritma ditunjukkan untuk tujuan pembuktian teoritis bahwa bahwa dari setiap CFG dapat dibuat suatu PDA dan sebaliknya dari setiap PDA dapat dibuat suatu CFG.
  - Untuk tujuan pengembangan aplikasi parsing tentu saja algoritma konversi ini tidak efisien; ada algoritma parsing yang efisien yang biasa digunakan: Algoritma CYK dan Algoritma Earley.

Membangun PDA dari Grammar

- Teorema

  Diberikan suatu CFG $G = (V, \Sigma, R, S)$, terdapat sebuah PDA M sehingga $L(M) = L(G)$

- Pembuktian dengan cara konstruksi (dua alternatif cara)
  - *Top-down parsing*

    Mulai dari S, menerapkan sejumlah *rule* R, dan memeriksa apakah ada derivasi dari G ke w
  - *Bottom-up parsing*

    Mulai dari w, menerapkan sejumlah *rule* R *backward*, dan memeriksa apakah S dicapai

CFG ke PDA top down

- Algoritma ***CFGtoPDAtopdown***

  Diberikan CFG $G = (V, \Sigma, R, S)$

  PDA $M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$ dapat dibentuk, yang mana Δ mengandung:

  - Transisi $((p, \varepsilon, \varepsilon), (q, S))$, *push start symbol S* ke *stack* dan pindah ke *state q*
  - Transisi $((q, \varepsilon, X), (q, \gamma_1\gamma_2 \ldots \gamma_n))$ untuk setiap *rule* $X \rightarrow \gamma_1\gamma_2 \ldots \gamma_n$
  - Transisi $((q, c, c), (q, \varepsilon))$ untuk setiap $c \in \Sigma$

Contoh-1

- $G = (\{E, T, F, \text{id}, +, *, (,)\}, \{\text{id}, +, *, (,)\}, R, E)$

  $R = \{$

  $E \rightarrow E+T \mid T$

  $T \rightarrow T*F \mid F$

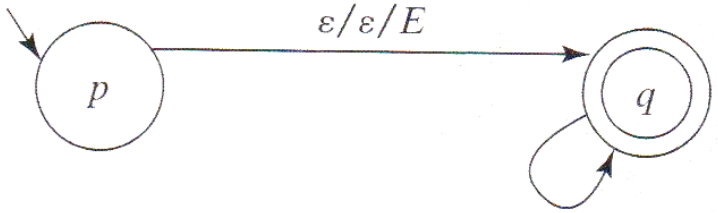
  $F \rightarrow (E) \mid \text{id}$

  $\}$

Buatlah PDA yang ekuivalen dengan grammar di atas!

Contoh-1 (top-down)

- PDA yang dibentuk dengan *top-down parsing*



$G = (\{E, T, F, id, +, *, (,)\}, \{id, +, *, (,)\}, R, E)$

$R = \{ \; E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

$\}$

ε / *E* / *E* +*T*
ε / *E* / *T*
ε / *T* / *T* **F*
ε / *T* / *F*
ε / *F* / (*E*)
ε / *F* / *id*
id / *id* / ε
(/ (/ ε
) /) / ε
+ / + / ε
* / * / ε

Contoh-2 $\{a^n b^* a^n\}$

Rule-rule:	Δ:
	(p, ε, ε), (q, S)
$S \rightarrow \varepsilon$	(q, ε, S), (q, ε)
$S \rightarrow B$	(q, ε, S), (q, B)
$S \rightarrow aSa$	(q, ε, S), (q, aSa)
$B \rightarrow \varepsilon$	(q, ε, B), (q, ε)
$B \rightarrow bB$	(q, ε, B), (q, bB)
	(q, a, a), (q, ε)
input = a a b b a a	(q, b, b), (q, ε)

trans(Δ)	***status***	***unread input***	***stack***
	p	a a b b a a	ε
0	q	a a b b a a	S
3	q	a a b b a a	aSa
6	q	a b b a a	Sa
3	q	a b b a a	aSaa
6	q	b b a a	Saa
2	q	b b a a	Baa
5	q	b b a a	bBaa
7	q	b a a	Baa
5	q	b a a	bBaa
7	q	a a	Baa
4	q	a a	aa
6	q	a	a
6	q	ε	ε

CFG to PDA bottom up

- Algoritma ***CFGtoPDAbottomup***

  Diberikan CFG $G = (V, \Sigma, R, S)$

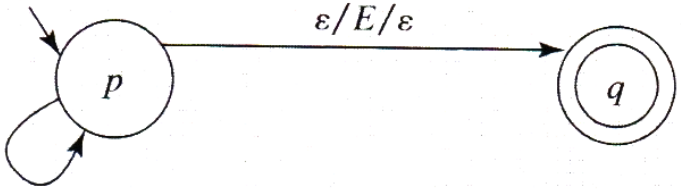
  PDA $M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$ dapat dibentuk, yang mana Δ mengandung:

  - Transisi *shift* $((p, c, \varepsilon), (p, c))$ untuk setiap $c \in \Sigma$
  - Transisi *reduce* $((p, \varepsilon, (\gamma_1\gamma_2 \ldots \gamma_n)^R), (p, X))$ untuk setiap *rule* $X \rightarrow \gamma_1\gamma_2 \ldots \gamma_n$
  - Transisi $((p, \varepsilon, S), (q, \varepsilon))$

- *Shift-reduce parser*

Contoh-1 (bottom-up)

- PDA yang dibentuk dengan *bottom-up parsing* dari Contoh-1 sebelumnya



id / ε / *id*
(/ ε / (
) / ε /)
+ / ε / +
* / ε / *
ε / *T*+*E* / *E*
ε / *T* / *E*
ε / *F***T* / *T*
ε / *F* / *T*
ε /)*E*(/ *F*
ε / *id* / *F*

$G = (\{E, T, F, id, +, *, (,)\}, \{id, +, *, (,)\}, R, E)$

$R = \{\ E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$
$\}$

Latihan 1

$L = \{a^n b^m c^p d^q : m + n = p + q\}$

(1) $S \rightarrow aSd$

(2) $S \rightarrow T$

(3) $S \rightarrow U$

(4) $T \rightarrow aTc$

(5) $T \rightarrow V$

(6) $U \rightarrow bUd$

(7) $U \rightarrow V$

(8) $V \rightarrow bVc$

(9) $V \rightarrow \varepsilon$

Input: a a b c d d

trans	***state***	***unread input***	***stack***

Latihan 1 (Cont'd)

$L = \{a^n b^m c^p d^q : m + n = p + q\}$

(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$

input = a a b c d d

0 $(p, \varepsilon, \varepsilon), (q, S)$
1 $(q, \varepsilon, S), (q, aSd)$
2 $(q, \varepsilon, S), (q, T)$
3 $(q, \varepsilon, S), (q, U)$
4 $(q, \varepsilon, T), (q, aTc)$
5 $(q, \varepsilon, T), (q, V)$
6 $(q, \varepsilon, U), (q, bUd)$
7 $(q, \varepsilon, U), (q, V)$
8 $(q, \varepsilon, V), (q, bVc)$
9 $(q, \varepsilon, V), (q, \varepsilon)$
10 $(q, a, a), (q, \varepsilon)$
11 $(q, b, b), (q, \varepsilon)$
12 $(q, c, c), (q, \varepsilon)$
13 $(q, d, d), (q, \varepsilon)$

Latihan 2

- $R = \{ S \rightarrow AAB$

  $A \rightarrow a$

  $B \rightarrow b\}$

Buatlah PDA-nya dengan Top Down / Bottom Up dan simulasikan konstruksinya untuk string ***aab****!*

Membangun Grammar dari PDA

- **Teorema**: untuk setiap PDA *M*, terdapat CFG $G = (V, \Sigma, R, S)$ yang mengenali Bahasa yang diterima *M*
- Teorema ini dibuktikan dengan algoritma yang mengkonversi PDA *M* menjadi CFG G.
- **Ide**: satu transisi akan dinyatakan sebagai rule-rule grammar dengan semua kemungkinan kondisi yang dapat terjadi pada stack.
- **Syarat**: *M* harus berbentuk *Restricted Normal Form*, dimana setiap transisi harus melakukan pop satu symbol (**single-pop**).

PDA *Restricted Normal Form (PDA-RNF)*

- Suatu PDA *M* disebut *restricted normal form (RNF)* **iff**:
  - *M* memiliki *start state* yang hanya melakukan *push* bottom-of-stack #, kemudian pindah ke *state* berikut di mana komputasi dimulai.
  - *M* memiliki *accepting state* tunggal *a* dan seluruh transisi ke *a* tidak membaca input dan melakukan *pop* # dari *stack*.
  - Setiap transisi dalam *M*, kecuali transisi dari *s*, *pop* **tepat satu** simbol dari *stack*.
- Maka: konversi PDA dilakukan dengan memodifikasi setiap transisi selalu pop tepat satu symbol (**single-pop**) kecuali dari start state hanya push #**.**
  - Pop-ε ➔ dengan satu transisi single-pop x & push x
  - Multipop $\gamma_1\gamma_2 \ldots \gamma_n$ ➔ dengan *n* transisi single-pop γ_i

Konversi PDA ke PDA-RNF

- Input PDA $M = (K, \Sigma, \Gamma, \Delta, s, A)$
- Output PDA-RNF $M' = (K', \Sigma, \Gamma', \Delta', s', A')$, dengan
  - $\Gamma' = \Gamma \cup \{\#\}$; $A' = \{a\}$; s' *start state* baru, dan K' dan Δ' dibentuk algoritma sbb.
- Algoritma:
  1. Inisialisasi $\Delta' = \Delta$, dan $K' = K \cup \{s', a\}$
  2. Tambahkan transisi $((s', \varepsilon, \varepsilon), (s, \#))$
  3. Untuk setiap $q_k \in A$ tambahkan $((q_k, \varepsilon, \#), (a, \varepsilon))$
  4. Pemecahan transisi Multi-pop $((q_1, c, \gamma_1\gamma_2 \ldots \gamma_n), (q_2, \gamma_P))$ dalam Δ ke transisi single-pop (Algo **PRNF-4**)
  5. Modifikasi transisi Non-ε $((q_1, c, \varepsilon), (q_2, \gamma_P))$ dalam Δ ke transisi Single-pop (Algo **PRNF-5**).

Algo PRNF-4 (Pemecahan Multi-pop)

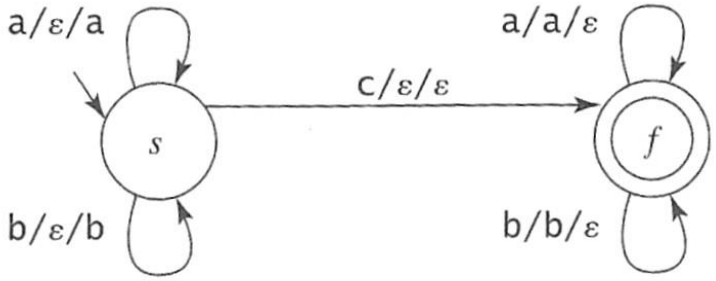
- Mengganti setiap transisi yang *pop n* simbol, $n > 1$, dengan n transisi Single-pop.
- Semula: $((q_1, c, \gamma_1\gamma_2 \ldots \gamma_n), (q_2, \gamma_P))$
- Tambahkan $(n–1)$ state baru yang belum ada sebelumnya dalam K': $v_1, v_2, \ldots, v_{n-1}$
- Buat transisi n transisi melalui $(n\text{-}1)$ status baru tsb. dari q_1 ke q_2, sabil mempop satu demi satu symbol stack γ_i :
  - $((q_1, \varepsilon, \gamma_1), (v_1, \varepsilon))$,
  - $((v_1, \varepsilon, \gamma_2), (v_2, \varepsilon))$,
  - ...,
  - $((v_{k-1}, c, \gamma_n), (q_2, \gamma_P))$
- Hapus transisi $((q_1, c, \gamma_1\gamma_2 \ldots \gamma_n), (q_2, \gamma_P))$

Algo PRNF-5 (Modifikasi Pop- ε)

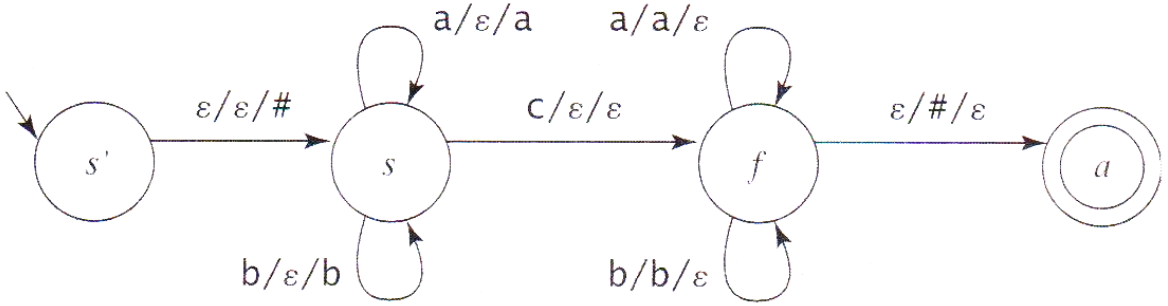
- Mengganti setiap transisi yang tidak pop, dengan sebanyak $|\Gamma \cup \{\#\}|$ transisi Single-pop.
- Semula: $((q_1, c, \varepsilon), (q_2, \gamma))$
- Untuk setiap simbol $\alpha \in \Gamma \cup \{\#\}$, tambahkan transisi $((q_1, c, \alpha), (q_2, \gamma\alpha))$
- Hapus transisi $((q_1, c, \varepsilon), (q_2, \gamma))$

Contoh-2

- Akan mengkonversi PDA berikut untuk Bahasa WcW^R $\{wcw^R: \ w \in \{a, b\}^*\}$, ke dalam *restricted normal form*!



- Step 1-4: membuat *start state* dan *accepting state* yang baru pada mesin *M*' menjadi:



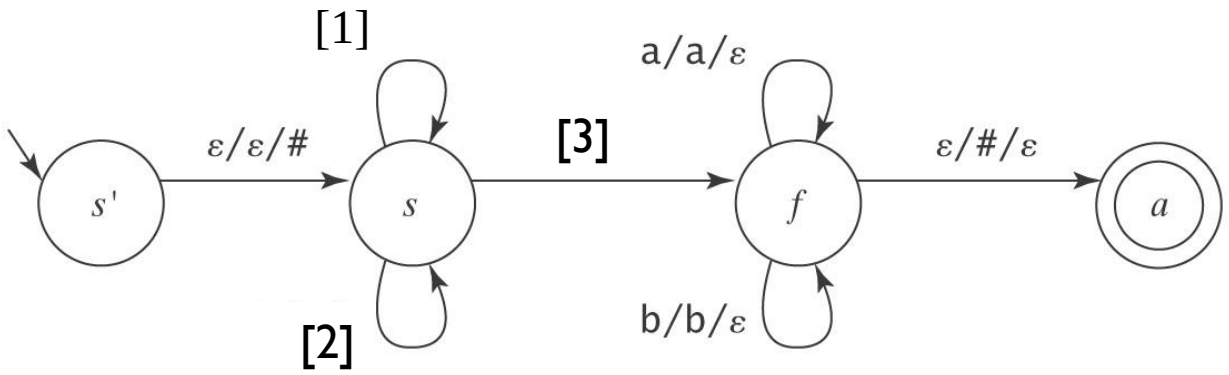
Contoh-2 (cont'd)

- Step 4 (PRNF-4) : tidak ada transisi yang multi-pop.
- Step 5 (PRNF-5): ada 3 transisi dari *s* yang melakukan non-pop diubah menjadi 9 transisi single-pop sebagai berikut:

[1] Dari: ((*s*, a, ε), (*s*, a)),
menjadi: ((*s*, a, #), (*s*, a#)), ((*s*, a, a), (*s*, aa)), ((*s*, a, b), (*s*, ab))

[2] Dari: ((*s*, b, ε), (*s*, b)),
menjadi: ((*s*, b, #), (*s*, b#)), ((*s*, b, a), (*s*, ba)), ((*s*, b, b), (*s*, bb)),

[3] Dari: ((*s*, c, ε), (*f*, ε)),
menjadi: ((*s*, c, #), (*f*, #)), ((*s*, c, a), (*f*, a)), ((*s*, c, b), (*f*, b))



Ide Pembentukan CFG

- Rule-rule akan mensimulasikan mekanisme setiap transisi membaca input, push dan pop stack serta perubahan status.
  - Untuk setiap transisi $((q, c, \gamma), (r , \alpha_1 \ldots \alpha_n))$, dibentuk: $\langle q,\gamma,v_n\rangle \rightarrow c \langle r,\alpha_1,v_1\rangle \langle v_1,\alpha_2,v_2\rangle \ldots \langle v_{n-1},\alpha_n,v_n\rangle$ dimana $v_1, v_2, \ldots, v_n$ status-status K' kecuali s'.
- Start symbol S ke kondisi awal setelah terjadi push # adalah status s, isi stack #, untuk menuju accepting a, dibentuk:
  - $S \rightarrow \langle s,\#,a\rangle$
- Berikutnya PDA-RNF berisikan transisi-transisi yang single-pop, kecuali dari start-state, tapi terdapat sejumlah kemungkinan push: push-ε, single-push, dan multi-push.
- Transisi-ε dibuat pada setiap status q bukan s' dibentuk:
  - $\langle q,\varepsilon,q\rangle \rightarrow \varepsilon$

Algoritma *PDAtoCFG*

- Input: PDA *RNF* $M = (K, \Sigma, \Gamma, \Delta, s', \{a\})$.
  Output yang diharapkan: CFG $G = (V_G, \Sigma, R, S)$
  1. Σ sama dengan Σ dari M
  2. Definisikan start symbol S dan
     buat rule $S \rightarrow < s,\#,a>$,
     s adalah state setelah s' dari transisi $((s, \varepsilon, \varepsilon),(s,\#))$
     dan *a accepting state*.
  3. Untuk setiap status q selain s',
     buat rule $<q,\varepsilon,q> \rightarrow \varepsilon$
  4. Rule-rule lain dalam R dibentuk dengan algoritma PDAtoCFG-S4.
  5. $V_G = \Sigma \cup$ himpunan simbol non terminal dalam rule R yang ditambahkan pada langkah 3.

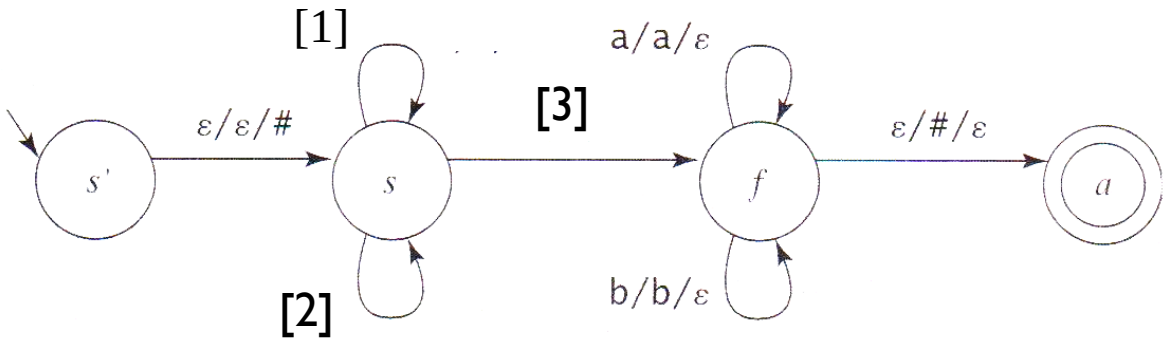
Algoritma PDAtoCFG-S4

A. Untuk setiap transisi yang **tidak *push* simbol** apapun $((q, c, \gamma), (r, \varepsilon))$ dan **setiap status** w selain s',
   - tambahkan rule $\langle q,\gamma,w\rangle \rightarrow c \langle r,\varepsilon,w\rangle$
     - Jika dalam $|K| = k$ status maka akan dibentuk $(k\text{-}1)$. Rule untuk setiap rule seperti ini.

B. Untuk setiap transisi $((q, c, \gamma), (r, \alpha_1 \alpha_2 \ldots \alpha_{n-1} \alpha_n))$ yaitu yang ***push n* simbol** $(n \geq 1)$, maka **setiap status-status** $v_1, v_2, \ldots, v_n$ di K selain s':
   - tambahkan rule $\langle q,\gamma,v_n\rangle \rightarrow c \langle r,\alpha_1,v_1\rangle \langle v_1,\alpha_2,v_2\rangle \ldots \langle v_{n-2},\alpha_{n-1},v_{n-1}\rangle \langle v_{n-1},\alpha_n,v_n\rangle$
     - Jika dalam $|K| = k$ status maka akan dibentuk $(k\text{-}1)^n$. rule untuk setiap rule seperti ini.

Contoh-2 (Cont'd)

- PDA *restricted normal form* untuk WcW^R



- Push-ε:
  - ((*f*, a, a), (*f*, ε)), ((*f*, b, b), (*f*, ε)), ((*f*, ε, #), (*a*, ε))
- Single Push:
  - ((*s*, c, #), (*f*, #)), ((*s*, c, a), (*f*, a)), ((*s*, c, b), (*f*, b))
- Push-double:
  - ((*s*, a, #), (*s*, a#)), ((*s*, a, a), (*s*, aa)), ((*s*, a, b), (*s*, ab)), ((*s*, b, #), (*s*, b#)), ((*s*, b, a), (*s*, ba)), ((*s*, b, b), (*s*, bb))

Contoh-2 (Cont'd)

- Langkah 2 algoritma menghasilkan 1 rule
  $S \rightarrow \langle s,\#,a \rangle$
- Langkah 3 algoritma menghasilkan 3 rule
  $\langle w,\varepsilon,w \rangle \rightarrow \varepsilon, \ \ w \in \{s, f, a\}$
  Langkah PDAtoCFG-S4 bagian A sebagai berikut:
  - Dari $((f, a, a), (f, \varepsilon))$ menghasilkan 3 rule
    $\langle f,a,w \rangle \rightarrow a \langle f,\varepsilon,w \rangle, \ \ w \in \{s, f, a\}$
  - Dari $((f, b, b), (f, \varepsilon))$ menghasilkan 3 rule
    $\langle f,b,w \rangle \rightarrow b \langle f,\varepsilon,w \rangle, \ w \in \{s, f, a\}$
  - Dari $((f, \varepsilon, \#), (a, \varepsilon))$ menghasilkan 3 rule
    $\langle f,\#,w \rangle \rightarrow \varepsilon \langle a,\varepsilon,w \rangle, \ w \in \{s, f, a\}$

Contoh-2 (Cont'd)

- Langkah PDAtoCFG-S4 bagian B (double push) sebagai berikut:
  - dari rule ((s,a,a),(s,aa)) menghasilkan 9 rule
    $\langle s,a,v_2\rangle \rightarrow a\langle s,a,v_1\rangle\langle v_1,a,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$
  - dari rule ((s,a,b),(s,ab)) menghasilkan 9 rule
    $\langle s,b,v_2\rangle \rightarrow a\langle s,a,v_1\rangle\langle v_1,b,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$
  - dari rule ((s,a,#),(s,a#)) menghasilkan 9 rule
    $\langle s,\#,v_2\rangle \rightarrow a\langle s,a,v_1\rangle\langle v_1,\#,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$
  - dari rule ((s,b,a),(s,ba)) menghasilkan 9 rule
    $\langle s,a,v_2\rangle \rightarrow b\langle s,b,v_1\rangle\langle v_1,a,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$
  - dari rule ((s,b,b),(s,bb)) menghasilkan 9 rule
    $\langle s,b,v_2\rangle \rightarrow b\langle s,b,v_1\rangle\langle v_1,b,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$
  - dari rule ((s,b,#),(s,b#)) menghasilkan 9 rule
    $\langle s,\#,v_2\rangle \rightarrow b\langle s,b,v_1\rangle\langle v_1,\#,v_2\rangle$ dengan $v_1, v_2 \in \{s, f, a\}$

Contoh-2 (Cont'd)

- Langkah PDAtoCFG-S4 bagian B (single-push) sebagai berikut:
  - dari rule ((s, c, a),(f, a)) menghasilkan 3 rule <s,a,*v*> $\rightarrow$ c <f,a,*v*> dengan $v \in \{s, f, a\}$
  - dari rule ((s, c, b),(f, b)) menghasilkan 3 rule <s,b,*v*> $\rightarrow$ c <f,b,*v*> dengan $v \in \{s, f, a\}$
  - dari rule ((s, c, #),(f, #)) menghasilkan 3 rule <s,#,*v*> $\rightarrow$ c <f,#,*v*> dengan $v \in \{s, f, a\}$

- Total dihasilkan 76 rule yang dihasilkan
- Terdapat *unreachable/nonproductive rule* => tersisa 16 rule.
  - Contoh: <*s*,#,*a*> $\rightarrow$ a <*s*,a,*a*> <*a*,#,*a*> non produktif karena tidak ada rule dengan LHS simbol <*s*,a,*a*> dan <*a*,#,*a*>.

Contoh-2 (Cont'd)

Rule-rule tersisa adalah 16 rule:

[1] $S \rightarrow <s,\#,a>$

[2] $<s,\#,a> \rightarrow a <s,a,f> <f,\#,a>$

[3] $<s,a,f> \rightarrow a <s,a,f> <f,a,f>$

[4] $<s,a,f> \rightarrow b <s,b,f> <f,a,f>$

[5] $<s,b,f> \rightarrow c <f,b,f>$

[6] $<f,\#,a> \rightarrow \varepsilon <a,\varepsilon,a>$

[7] $<f,a,f> \rightarrow a <f,\varepsilon,f>$

[8] $<f,b,f> \rightarrow b <f,\varepsilon,f>$

[9] $<f,\varepsilon,f> \rightarrow \varepsilon$

[10] $<a,\varepsilon,a> \rightarrow \varepsilon$

[14] $<s,b,f> \rightarrow a <s,a,f> <f,b,f>$

[15] $<s,\#,a> \rightarrow b <s,b,f> <f,\#,a>$

[16] $<s,b,f> \rightarrow b <s,b,f> <f,b,f>$

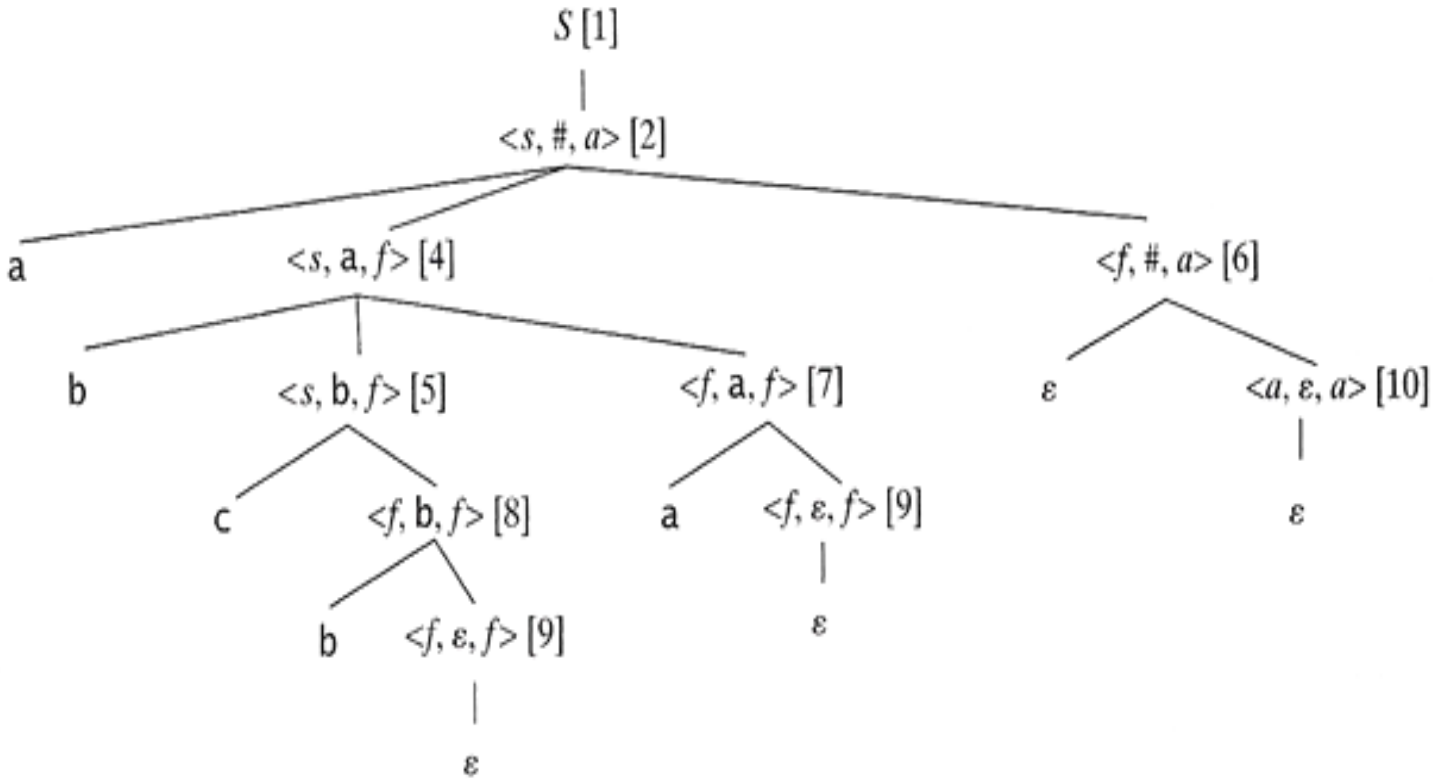
[17] $<s,\#,a> \rightarrow c <f,\#,a>$

[18] $<s,a,f> \rightarrow c <f,a,f>$

Note: [15],[17] dl buku teks salah $<s,\#,f>$ seharusnya $<s,\#,a>$

Contoh-2 (Cont'd)

- Parsing dari grammar yang dihasilkan pada string abcba



Contoh-2 (Cont'd)

Jika Dilakukan Penamaan Ulang:

$S \rightarrow T$

$T \rightarrow \text{a}DB$

$D \rightarrow \text{a}DE$

$D \rightarrow \text{b}AE$

$A \rightarrow \text{c}C$

$B \rightarrow \varepsilon F$

$E \rightarrow \text{a}G$

$\text{C} \rightarrow \text{b}G$

$G \rightarrow \varepsilon$

$F \rightarrow \varepsilon$

$A \rightarrow \text{a}DC$

$T \rightarrow \text{b}AB$

$A \rightarrow \text{b}AC$

$T \rightarrow \text{c}B$

$D \rightarrow \text{c}E$

JIka unit-rule dan ε-rule dihilangkan akan menghasilkan:

$S \rightarrow \text{a}D \mid \text{b}A \mid \text{c}$

$D \rightarrow \text{a}D\text{a} \mid \text{b}A\text{a} \mid \text{ca}$

$A \rightarrow \text{b}A\text{b} \mid \text{a}D\text{b} \mid \text{cb}$

Penting!

- Algoritma-algoritma telah menunjukkan bahwa antara CFL dan PDA convertible satu dengan lainnya:
  - Setiap CFL selalu ada PDA yang dapat menerimanya, dan sebaliknya setiap PDA menerima CFL.
- Algoritma-algoritma untuk kebutuhan praktis kurang efisien akibat nondeterminisme yang digunakan.
- Tapi, ide top-down/bottom-up dari PDA menginspirasi algoritma-algoritma popular dan efisien
  - CYK (buttom-up)
  - Early (buttom-up dengan top-down prediction).
- Materi kedua algoritma tsb tidak masuk dl kuliah ini.