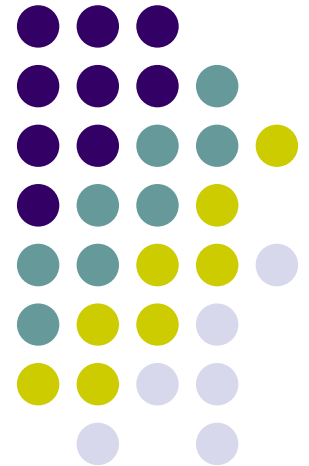


Teknik Kompiler 6

oleh: **antonius rachmat c,**
s.kom





Analisis Sintaks (Parser)

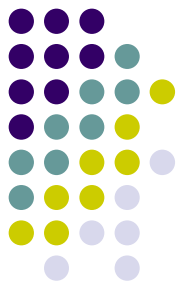
- Analisis Sintaks bergantung pada bahasa pemrograman masing-masing. Karena masing-masing bahasa pemrograman memiliki bentuk sintaks yang berbeda-beda.
- Analisis Sintaks memiliki input berupa **token** yang berasal dari scanner dan source code.
- Analisis Sintaks (Parser) bertugas mengecek kebenaran sintaks dan menghasilkan & memproses pohon sintaks.
- **Sintaks** adalah aturan-aturan bahasa dalam suatu bahasa pemrograman tertentu.

Parser



- Sebuah Parser akan membentuk Pohon Sintaks (Parse Tree).
- Sebuah tree adalah suatu graph terhubung yang tidak sirkuler dan memiliki satu buah root (akar) dan dari situ memiliki lintasan ke setiap simpul (daun/leaf).
- Parse Tree berfungsi untuk menggambarkan bagaimana memperoleh suatu string dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal, sampai **tidak ada** simbol yang belum tergantikan.

Tata Bahasa Bebas Konteks



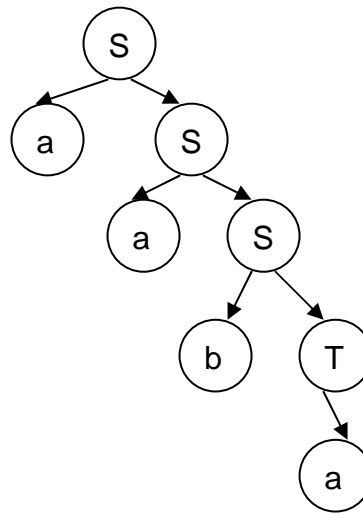
- Untuk mengimplementasikan Parser diperlukan TBBK (Context Free Grammar)
- TBBK adalah sekumpulan simbol-simbol variabel (non-terminal), yang masing-masing merepresentasikan bahasa. Bahasa yang direpresentasikan dengan simbol-simbol non terminal tersebut diproses secara rekursif dengan suatu aturan-aturan yang disebut aturan produksi.
- Tata bahasa bebas konteks (tipe 2) memiliki elemen:
 - Terminal : simbol dasar yang tidak dapat diturunkan lagi. Terminal disebut juga token.
 - Non terminal : variabel sintaktik yang masih dapat diturunkan lagi.

TBBK (2)

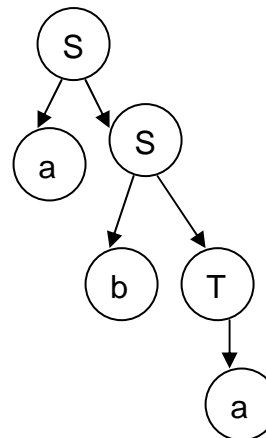


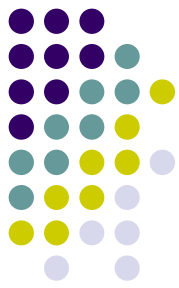
- Contoh TBBK untuk pasangan kurung yang selalu berpasangan:
 $S \Rightarrow R$
 $R \Rightarrow \{ \}$
 $R \Rightarrow (R)$
 $R \Rightarrow RR$
- Contoh TBBK untuk palindrom:
 $S \Rightarrow R$
 $R \Rightarrow \{ \} \mid a \mid b$
 $R \Rightarrow aRa \mid bRb$
- Contoh TBBK lain
 $S \Rightarrow AB$
 $A \Rightarrow aA \mid a$
 $B \Rightarrow bB \mid b$

TBBK (3)



- Contoh TBBK:
 $S \Rightarrow aS$
 $S \Rightarrow bT$
 $T \Rightarrow a$
- Maka misalkan untuk string “aaba” maka TBBK diatas dapat diturunkan menjadi :
 $S \rightarrow aS$
 $S \Rightarrow aaS$
 $S \Rightarrow aabT$
 $S \Rightarrow aaba$
- Artinya string “aaba” cocok dan diterima oleh TBBK diatas.
- Misalkan untuk string “aba” dan terdapat aturan produksi sebagai berikut:
 $S \Rightarrow aS$
 $S \Rightarrow abT$
 $S \rightarrow aba$
- Pohon Sintaks :

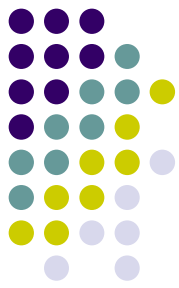




Contoh – contoh TBBK

- Contoh :
S \Rightarrow aS | b
S \Rightarrow bT
T \Rightarrow a | bS
- Untuk string “abbaab” :
S \Rightarrow aS
S \Rightarrow abT
S \Rightarrow abbS
S \Rightarrow abbaS
S \Rightarrow abbaaS
S \Rightarrow abbaab
- Pohon Sintaks ?

Contoh – contoh TBBK (2)



- Contoh :
S \Rightarrow AB
A \Rightarrow aA | a
B \Rightarrow bB | b
- Untuk string “aabbb”
S \Rightarrow AB
S \Rightarrow aAB
S \Rightarrow aaB
S \Rightarrow aabB
S \Rightarrow aabbB
S \Rightarrow aabbb
- Pohon Sintaks?

Contoh Parsing dalam Bhs Inggris



- S: Sentence
- SP: Subject Phrase
- VP: Verb Phrase
- NP: Noun Phrase
- V: Verb
- O: Object
- A: Article
- N: Noun

Parsing Inggris (2)



- Aturan Produksi:
 - $S \Rightarrow SP VP$
 - $SP \Rightarrow A N$
 - $A \Rightarrow 'a'$
 - $A \Rightarrow 'the'$
 - $N \Rightarrow 'monkey'$
 - $N \Rightarrow 'banana'$
 - $N \Rightarrow 'cat'$
 - $N \Rightarrow 'mouse'$
 - $N \Rightarrow 'tree'$
 - $VP \Rightarrow V O$
 - $V \Rightarrow 'ate'$
 - $V \Rightarrow 'climb'$
 - $O \Rightarrow NP$
 - $NP \Rightarrow A N$

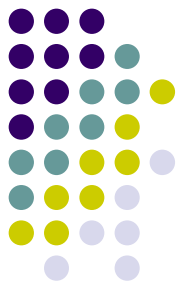


Parsing Inggris (3)

- Penurunan untuk “the cat ate a mouse”
 - S => SP VP
 - => A N VP
 - => the N VP
 - => the cat V O
 - => the cat ate O
 - => the cat ate NP
 - => the cat ate A N
 - => the cat ate a N
 - => the cat ate a mouse

Apakah berlaku untuk: the cat ate a banana, the monkey climbs a tree, the banana ate a cat?

Cara Penurunan



- Penurunan dapat dilakukan :
 - Dengan penurunan terkiri : nonterminal terkiri yang disubstitusi.
 - Dengan penurunan terkanan : nonterminal terkanan yang disubstitusi.
 - Contoh 1:
 $S \Rightarrow aAS \mid a$
 $A \Rightarrow SbA \mid ba$
 - Untuk string “aabbaa”:
 - Dengan penurunan terkiri : $S \Rightarrow aAS \Rightarrow a**Sb**AS \Rightarrow aabAS \Rightarrow aab**b**aS \Rightarrow aabbaa$.
 - Bagaimana Parse Tree?
 - Dengan penurunan kanan : $S \Rightarrow aAS \Rightarrow a**A**a \Rightarrow a**Sb**Aa \Rightarrow aS**b**baa \Rightarrow aabbaa$.
 - Bagaimana Parse Tree?

Penurunan (2)



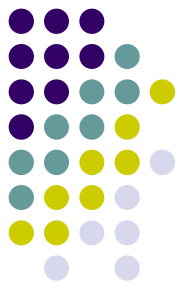
- Contoh 2 :
- Misal TBBK = $E \Rightarrow E + E \mid E * E \mid (E) \mid -E \mid id$
- String “-(id + id)” diterima karena :
- $E \Rightarrow - E \Rightarrow - (E) \Rightarrow - (E+E) \Rightarrow - (id +E) \Rightarrow -(id + id)$

- Exercise:
- Is “**id-id**” a sentence of TBBK? No
- Is “**-id+id**” a sentence of TBBK? Yes

Why TBBK for Parser?



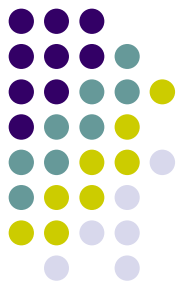
- Mendukung tata bahasa yang bersifat rekursif
- Spesifikasi bahasa pemrograman menggunakan TBBK
 - Contoh : $(x + 2) * 3$
 - Expr \Rightarrow expr op expr
 - Expr \Rightarrow (expr)
 - Expr \Rightarrow - expr
 - Expr \Rightarrow **id**
 - op \Rightarrow +
 - op \Rightarrow -
 - op \Rightarrow *
 - op \Rightarrow /
 - op \Rightarrow ^
- Setiap RE dapat dideskripsikan dengan TBBK
 - Contoh:
 - $(a|b)^*abb$
 - $A \Rightarrow aA \mid bA \mid abb$



Why RE for Scanner?

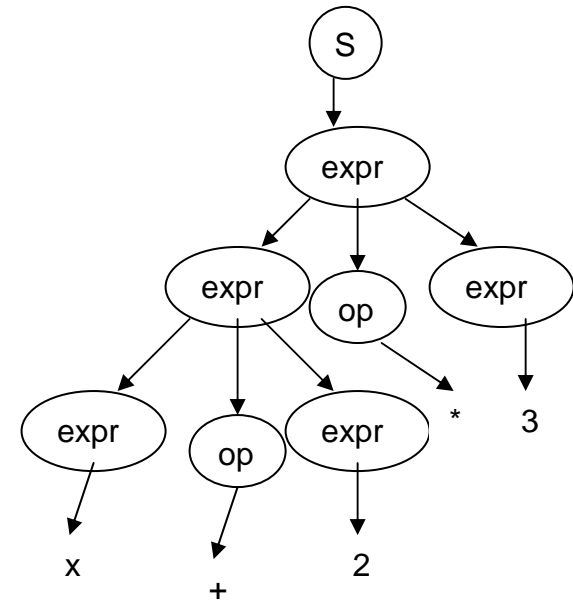
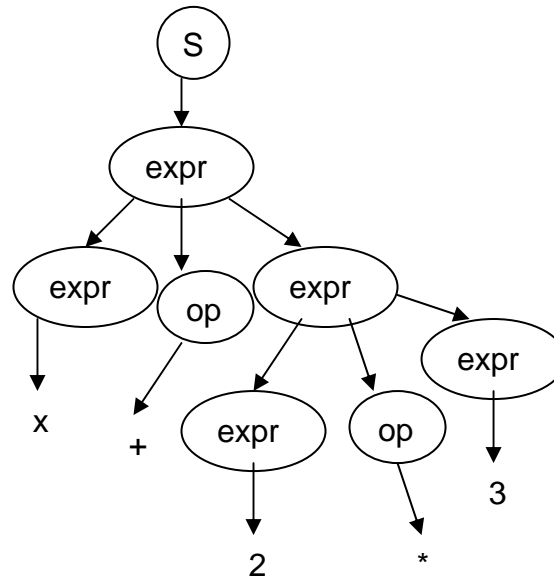
- Scanner memiliki aturan tata bahasa yang sederhana
- RE menghasilkan notasi yang jelas dan mudah dimengerti untuk token
- Scanner tidak memerlukan suatu tata bahasa yang rekursif

NB: dengan adanya scanner dan parser sangat memudahkan modularisasi dan mengurangi kebutuhan resource



TBBK dan Prioritas

- Contoh $x + 2 * 3$
 $S \Rightarrow S + S$
 $S \Rightarrow id + S * S$
 $S \Rightarrow id + id * id$
- Parse Tree dari $x + 2 * 3$

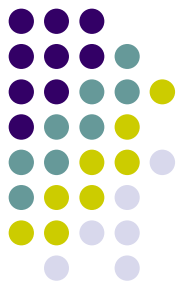


- **Belum bisa mendeteksi prioritas**

Mengubah NFA ke TBBK

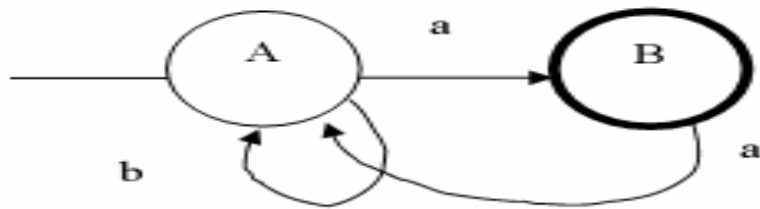


- Untuk setiap state i dari NFA buat simbol non terminal A_i .
- Jika state i memiliki transisi ke state j pada simbol a , maka buat satu produksi $A_i \Rightarrow aA_j$.
- Jika state i masuk ke state j dengan masukkan ϵ maka buat $A_i \Rightarrow A_j$.
- Jika i state penerima dengan buat $A_i \Rightarrow \epsilon$.
- Jika i state awal, buat A_i sebagai simbol awal dari TBBK



Contoh NFA ke TBBK

	a	b
A	B	A
B	A	{}



String "aabba" diterima.

A => aB

A => bA

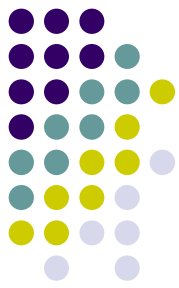
B => aA

B => {}

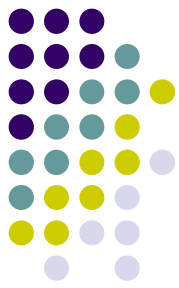
A => aB => aaA => aabA => aabbA

=> aabbaB => aabba{} => aabba

Parsing



- Proses Parsing merupakan tahapan yang berfungsi untuk memeriksa urutan kemunculan token. Di dalam mengimplementasikan sebuah metode parsing perlu diperhatikan :
 - Rentang waktu eksekusi
 - Penanganan Kesalahan



Metode Parsing

- **Top Down**

- Metode ini menelusuri pohon, dari root menuju ke daun (leaf). Metode ini meliputi:
 - Backtracking Mode : Metode Brute Force
 - Non Backtracking Mode : Recursive Descent Parser dan Predictive Parser
- Top Down memarsing tree secara pre order.

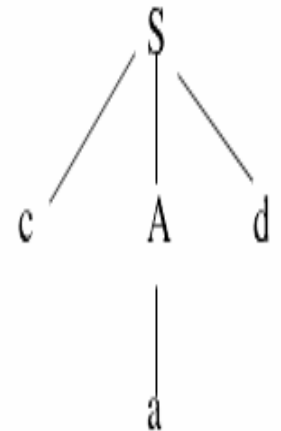
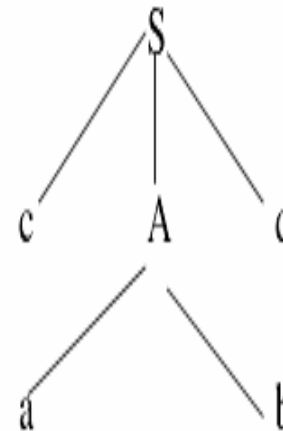
- Contoh :

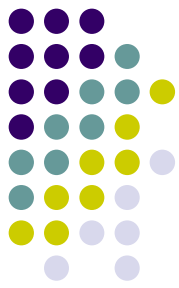
$S \Rightarrow cAd$

$A \Rightarrow ab \mid a$

- Untuk inputan “cad”

- Pohon Sintaks ?





Metode Parsing (2)

- **Bottom Up**

- Metode ini menelusuri pohon dari daun menuju ke root. Biasanya mengurangi string/daun sampai pada akarnya.

- Contoh :

$S \Rightarrow aABe$

$A \Rightarrow Abc \mid b$

$B \Rightarrow d$

- Maka untuk string “abbcde”

- abbcde
- aAbcde
- aAde
- aABe
- S



Metode Parsing lainnya

- **Metode Brutal Force**

- Metode ini akan melakukan substitusi semua simbol non terminal yang ada. Jika terjadi salah parsing (atau tidak cocok), maka dapat dilakukan backtracking.

- Contoh :

$S \Rightarrow aAd \mid aB$

$A \Rightarrow b \mid c$

$B \Rightarrow ccd \mid ddc$

- Misal ingin memarsing : “accd”.

$S \Rightarrow aAd$

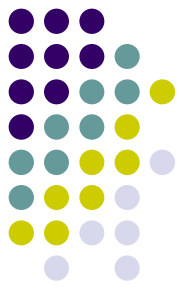
$S \Rightarrow abd$: gagal, maka dilakukan backtrack:

$S \Rightarrow acd$: gagal, maka dilakukan backtrack:

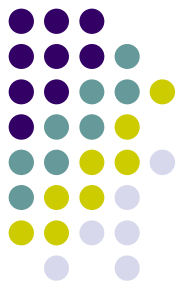
$S \Rightarrow aB$

$S \Rightarrow accd$: berhasil!

Metode Parsing lainnya (2)



- Kelemahan Brutal Force :
 - Mencoba semua aturan produksi sehingga akan menjadi lambat
 - Sulit melakukan backtracking dan pemulihan kesalahan
 - Memakan banyak memori karena perlu mencatat lokasi backtrack

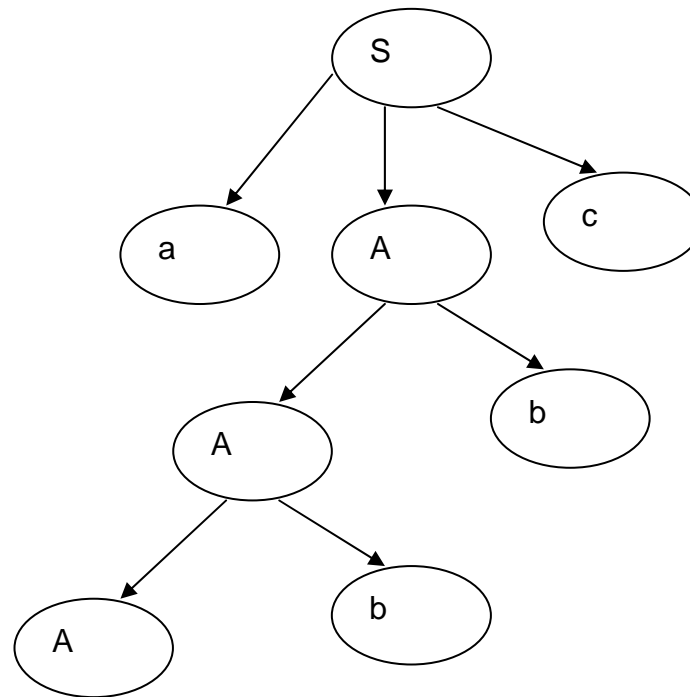


TBBK Rekursif Kiri

- TBBK yang memiliki simbol non terminal di ruas kanan dari simbol non terminal yang ada di ruas kiri. Simbol non terminal itu terletak di ruas kanan terdepan.
- TBBK ini juga tidak memiliki kemungkinan aturan produksi lain yang berupa simbol terminal.
- Contoh :
 $S \Rightarrow Sd$
 $A \Rightarrow Aad$
- Menyebabkan pohon tumbuh ke kiri

Contoh Pohon

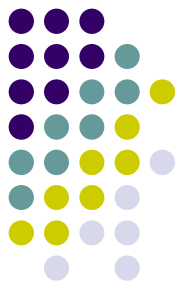
- $S \Rightarrow aAc$
- $A \Rightarrow Ab \mid \{\}$





Penghilangan Rekursif Kiri

- Pisahkan aturan produksi yang rekursif kiri dengan yang tidak.
 - Yang rekursif kiri :
 - $A \Rightarrow Aa_1 \mid Aa_2 \mid Aa_3 \mid \dots$
 - Yang tidak rekursif kiri, termasuk produksi epsilon:
 - $A \Rightarrow b_1 \mid b_2 \mid b_3 \mid \dots$
- Kita bisa tentukan a_1, a_2, a_3, \dots dan b_1, b_2, b_3, \dots
- Lakukan penggantian aturan produksi yang rekursif kiri menjadi:
 - $A \Rightarrow b_1Z \mid b_2Z \mid b_3Z \mid \dots$
 - $Z \Rightarrow a_1 \mid a_2 \mid a_3 \mid \dots$
 - $Z \Rightarrow a_1Z \mid a_2Z \mid a_3Z \mid \dots$
- Penggantian itu dilakukan untuk setiap aturan produksi dengan simbol di ruas kiri yang sama. Bisa muncul simbol variabel baru Z_1, Z_2, Z_3, \dots sesuai dengan banyaknya variabel yang menghasilkan produksi yang rekursif kiri.
- Hasil akhir berupa aturan produksi pengganti ditambah dengan aturan produksi semula yang tidak rekursif kiri



Penghilangan Rekursif Kiri (2)

- Contoh :
 $S \Rightarrow Sab \mid aSc \mid dd \mid ff \mid Sbd$
- Aturan produksi yang rekursif kiri :
 $S \Rightarrow Sab \mid Sbd$
- Kita simbolkan : $a1 = ab$ dan $a2 = bd$
- Aturan produksi yang tidak rekursif kiri:
 $S \Rightarrow aSc \mid dd \mid ff$
- Maka simbolkan $b1 = aSc$, $b2 = dd$, dan $b3 = ff$
- Lalu lakukan penggantian rekursif kiri $S \Rightarrow Sab \mid Sbd$ menjadi:
 $S \Rightarrow aScZ1 \mid ddZ1 \mid ffZ1$
 $Z1 \Rightarrow ab \mid bd$
 $Z1 \Rightarrow abZ1 \mid bdZ1$

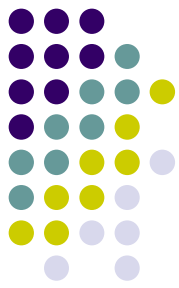
Hasil akhir:

$S \Rightarrow aSc \mid dd \mid ff$

$S \Rightarrow aScZ1 \mid ddZ1 \mid ffZ1$

$Z1 \Rightarrow ab \mid bd$

$Z1 \Rightarrow abZ1 \mid bdZ1$



Contoh lain:

- $S \Rightarrow Sab \mid Sb \mid cA$
- $A \Rightarrow Aa \mid a \mid bd$

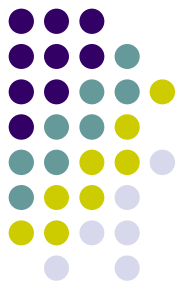
- Yang rekursif : $S \Rightarrow Sab \mid Sb$ dan $A \Rightarrow Aa$
- Yang tidak: $S \Rightarrow cA$ dan $A \Rightarrow a \mid bd$

Pergantian:

- Untuk $S \Rightarrow Sab \mid Sb$
 - $S \Rightarrow cAZ1$
 - $Z1 \Rightarrow ab \mid b$
 - $Z1 \Rightarrow abZ1 \mid bZ1$
- Untuk $A \Rightarrow Aa$
 - $A \Rightarrow aZ2 \mid bdZ2$
 - $Z2 \Rightarrow a$
 - $Z2 \Rightarrow aZ2$

Hasil:

$S \Rightarrow cA$
 $A \Rightarrow a \mid bd$
 $S \Rightarrow cAZ1$
 $Z1 \Rightarrow ab \mid b$
 $Z1 \Rightarrow abZ1 \mid bZ1$
 $A \Rightarrow aZ2 \mid bdZ2$
 $Z2 \Rightarrow a$
 $Z2 \Rightarrow aZ2$



Contoh lain:

- $S \Rightarrow aA \mid b \mid cS$
- $A \Rightarrow Sd \mid e$ #rekursif kiri

Jadi:

$S \Rightarrow aA \mid b \mid cS$

$A \Rightarrow aAd \mid bd \mid cSd \mid e$



TBBK Rekursif Kanan

- TBBK Rekursif Kanan: TBBK yang memiliki simbol non terminal di ruas kanan dari simbol non terminal yang ada di ruas kiri. Simbol non terminal itu terletak di ruas kanan terbelakang.
- TBBK ini juga tidak memiliki kemungkinan aturan produksi lain yang berupa simbol terminal.
- Contoh:
 $S \Rightarrow dS$
 $B \Rightarrow adB$

Membuat pohon sintaks melebar ke kanan

NEXT

- Analisis Sintaks 2
- Transformasi TBBK dan CYK

