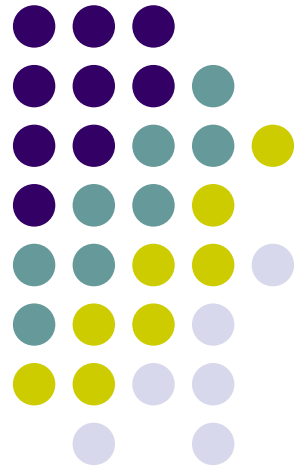
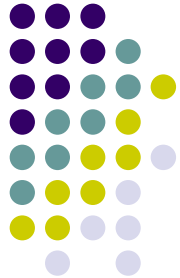
Teknik Kompiler 6

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



Analisis Sintaks (Parser)

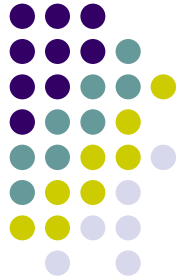


- Analisis Sintaks bergantung pada bahasa pemrograman masing-masing. Karena masing-masing bahasa pemrograman memiliki bentuk sintaks yang berbeda-beda.
- Analisis Sintaks memiliki input berupa **token** yang berasal dari scanner dan source code.
- Analisis Sintaks (Parser) bertugas mengecek kebenaran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

Parser

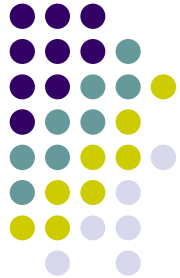
- Sebuah Parser akan membentuk Pohon Sintaks (Parse Tree).
- Sebuah tree adalah suatu graph terhubung yang tidak sirkuler dan memiliki satu buah root (akar) dan dari situ memiliki lintasan ke setiap simpul (daun/leaf).
- Parse Tree berfungsi untuk menggambarkan bagaimana memperoleh suatu string dengan 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 => R

  R => {}

  R => (R)

  R => RR
- Contoh TBBK untuk palindrom:

  S => R

  R => {} | a | b

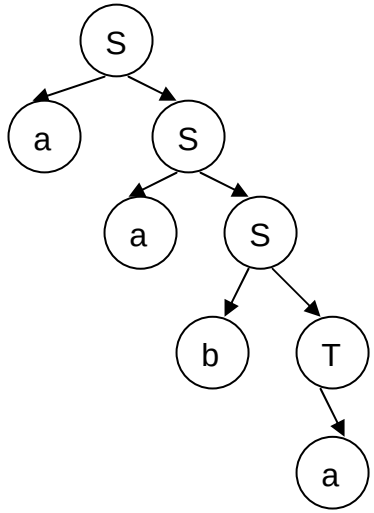
  R => aRa | bRb
- Contoh TBBK lain

  S => AB

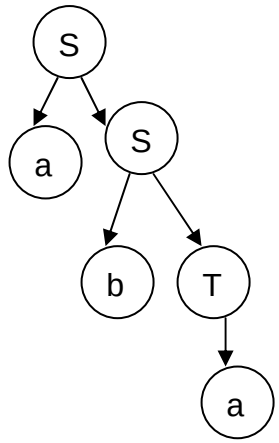
  A => aA | a

  B => bB | b

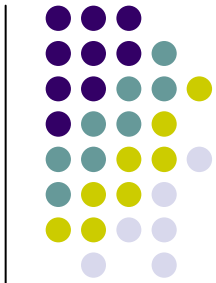
TBBK (3)



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



Contoh – contoh TBBK



- Contoh :

  S => aS | b

  S => bT

  T => a | bS

- Untuk string “abbaab” :

  S => aS

  S => abT

  S => abbS

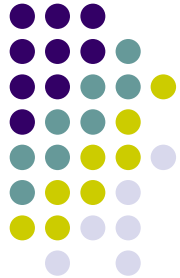
  S => abbaS

  S => abbaaS

  S => abbaab

- Pohon Sintaks ?

Contoh – contoh TBBK (2)



- Contoh :

  S => AB

  A => aA | a

  B => bB | b
- Untuk string “aabbb”

  S => AB

  S => aAB

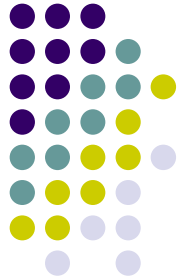
  S => aaB

  S => aabB

  S => aabbB

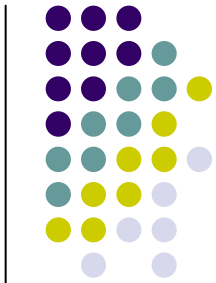
  S => aabbb
- Pohon Sintaks?

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 => SP VP
  - SP => A N
  - A => ‘a’
  - A => ‘the’
  - N => ‘monkey’
  - N => ‘banana’
  - N => ‘cat’
  - N => ‘mouse’
  - N => ‘tree’
  - VP => V O
  - V => ‘ate’
  - V => ‘climb’
  - O => NP
  - NP => 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?

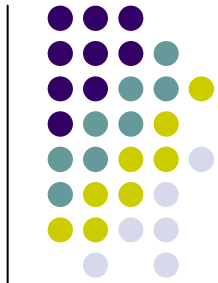
Cara Penurunan

- Penurunan dapat dilakukan :
  - Dengan penurunan terkiri : nonterminal terkiri yang disubstitusi.
  - Dengan penurunan terkanan : nonterminal terkanan yang disubstitusi.
  - Contoh 1:

    S => aAS | a

    A => SbA | ba
  - Untuk string “aabbaa”:
    - Dengan penurunan terkiri : S => aAS => a**SbA**S => a**a**bAS => aab**ba**S => aabba**a**.
      - Bagaimana Parse Tree?
    - Dengan penurunan kanan : S => aAS => aA**a** => a**SbA**a => aSb**ba**a => a**a**bbaa.
      - Bagaimana Parse Tree?

Penurunan (2)



- Contoh 2 :
- Misal TBBK = E => E + E | E * E | (E) | -E | id
- String “-(id + id)” diterima karena :
- E => - E => - (E) => - (E+E) => - (id +E) => -(id + id)

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

Penurunan (3)

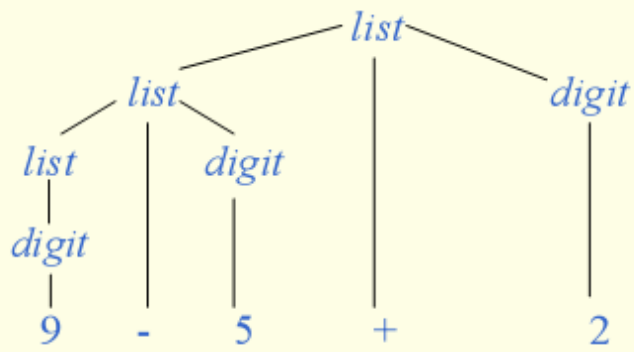
- Contoh 3

list → *list* + *digit*	(2.2)
list → *list* - *digit*	(2.3)
list → *digit*	(2.4)
digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	(2.5)

Is 9-5+2 a list?

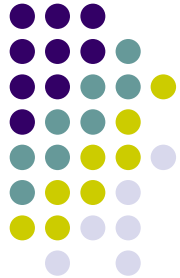
9 is a *list* (2.4), because 9 is a *digit* (2.5)
9-5 is a *list* (2.3), because 9 is a *list* and 5 is a *digit*
9-5+2 is a *list* (2.2), because 9-5 is a *list* and 2 is a *digit*

What is the parse tree for 9-5+2?



(Aho,Sethi,Ullman, pp. 28)

Why TBBK for Parser?



- Mendukung tata bahasa yang bersifat rekursif
- Spesifikasi bahasa pemrograman menggunakan TBBK
  - Contoh : (x + 2) * 3

    Expr => expr op expr

    Expr => (expr)

    Expr => - expr

    Expr => **id**

    op => +

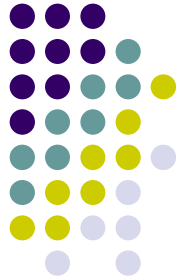
    op => -

    op =>*

    op => /

    op => ^
- Setiap RE dapat dideskripsikan dengan TBBK
  - Contoh:
    - (a|b)*abb
    - A => aA | bA | abb

Why RE for Scanner?

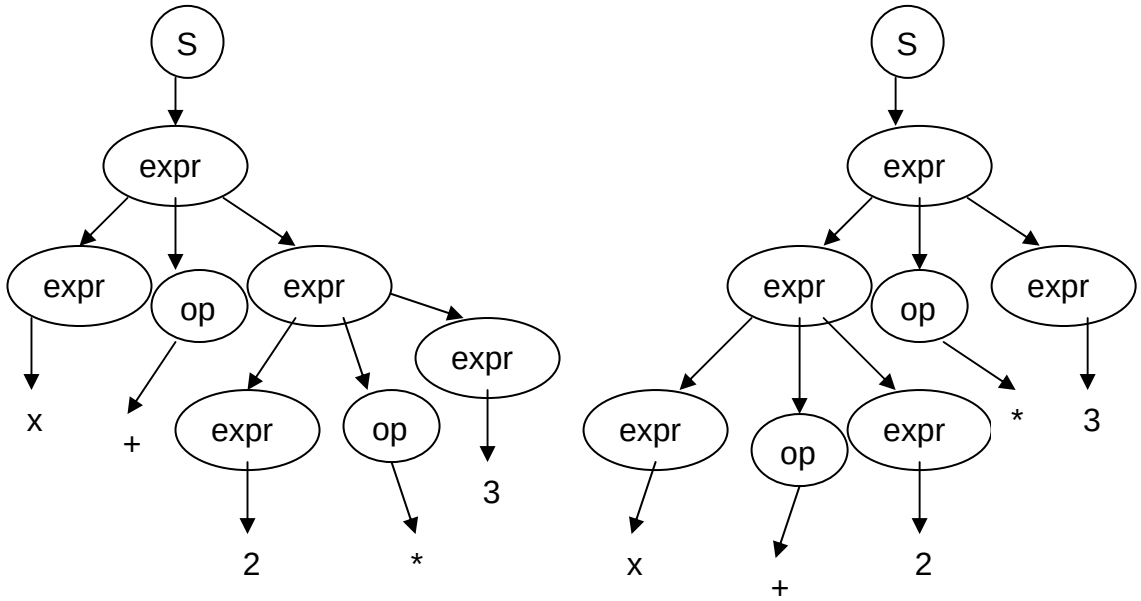


- Scanner memiliki aturan tata bahasa yang sederhana
- RE menghasilkan notasi yang jelas dan mudah dimengerti untuk token
- Scanner tidak memerlukan suatu tata bahasa yang rekursif

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

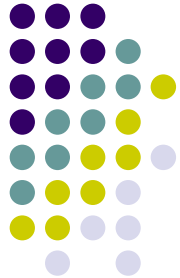
TBBK dan Prioritas

- Contoh **x + 2 * 3**
  S => S + S
  S => id + S * S
  S => id + id * id
- Parse Tree dari
  x + 2 * 3



- **Belum bisa** mendeteksi prioritas

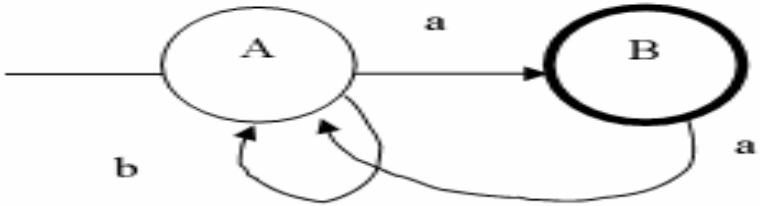
Mengubah NFA ke TBBK



- Untuk setiap state i dari NFA buat simbol non terminal Ai.
- Jika state i memiliki transisi ke state j pada simbol a,maka buat satu produksi Ai => aAj.
- Jika state i masuk ke state j dengan masukkan e maka buat Ai => Aj.
- Jika i state penerima dengan buat Ai => ε.
- Jika i state awal, buat Ai sebagai simbol awal dari TBBK

Contoh NFA ke TBBK

	a	b
A	B	A
B	A	{}

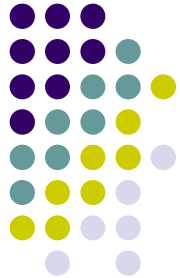


String "aabba" diterima.

A => aB
A => bA
B => aA
B => {}

A => aB => aaA => aabA => aabbA
=> aabbaB => aabba{} => aabba

Parsing



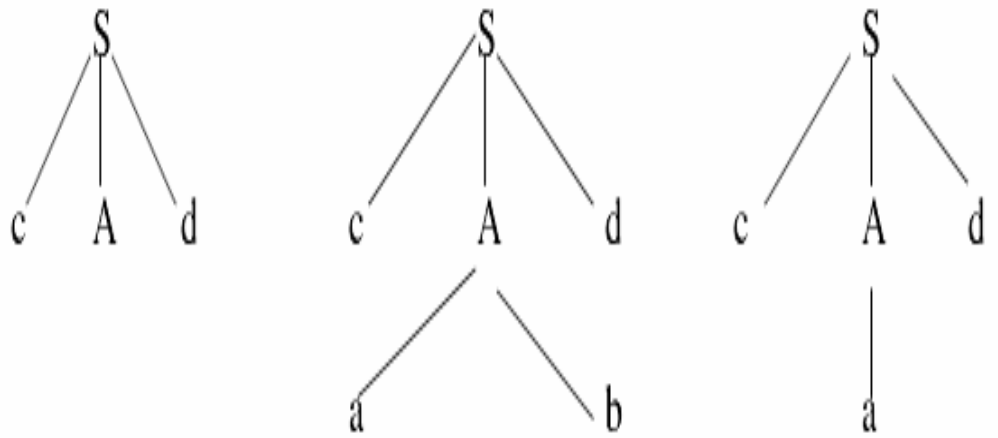
- Proses Parsing merupakan tahapan yang berfungsi untuk memeriksa urutan kemunculan token. Di dalam mengimplementasikan sebuah metode parsing perlu diperhatikan :
  - Rentang waktu eksekusi
  - Penanganan Kesalahan

Metode Parsing

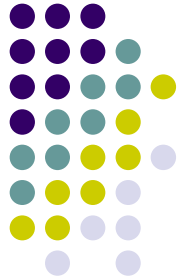
- **Top Down**
  - Metode ini menelusuri pohon, dari root menuju ke daun (leaf). Metode ini meliputi:
    - Backtracking Mode : Metode Brute Force
    - Non Backtracking Mode : Recursive Descent Parser dan Predictive Parser
  - Top Down memparsing tree secara pre order.
  - Contoh :

    S => cAd

    A => ab | a
  - Untuk inputan “cad”
  - Pohon Sintaks ?



Metode Parsing (2)



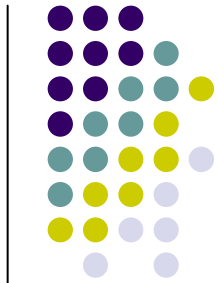
- **Bottom Up**
  - Metode ini menelusuri pohon dari daun menuju ke root. Biasanya mengurangi string/daun sampai pada akarnya.
  - Contoh :

    S => aABe

    A => Abc | b

    B => 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 => aAd | aB

    A => b | c

    B => ccd | ddc
  - Misal ingin memparsing : “accd”.

    S => aAd

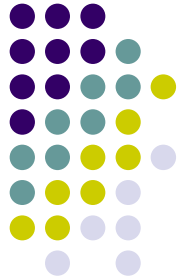
    S => abd : gagal, maka dilakukan backtrack:

    S => acd : gagal, maka dilakukan backtrack:

    S => aB

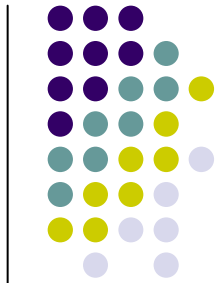
    S => accd : berhasil!

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



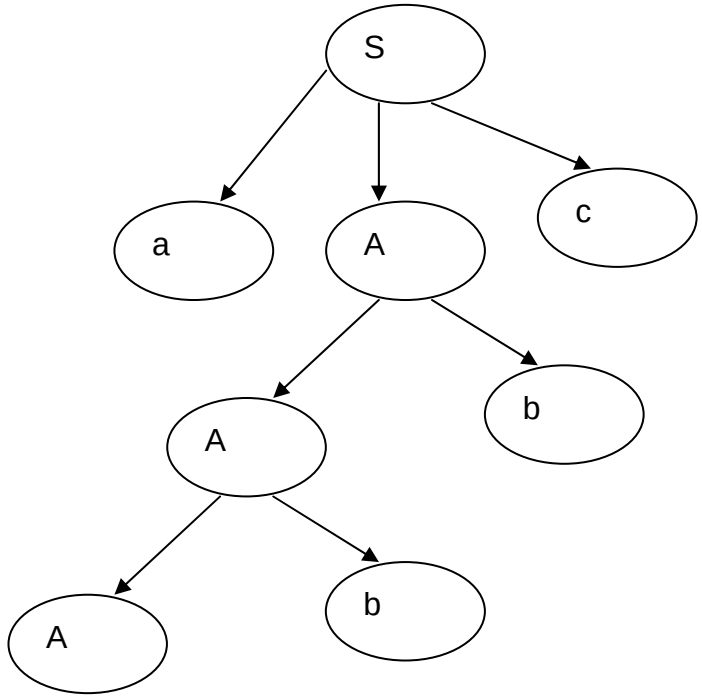
- TBBK yang memiliki simbol non terminal di ruas kanan dari simbol non terminal yang ada di ruas kiri. Simbol non terminal itu terletak di ruas kanan terdepan.
- TBBK ini juga tidak memiliki kemungkinan aturan produksi lain yang berupa simbol terminal.
- Contoh :

  S => Sd

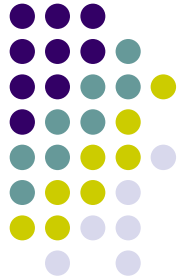
  A => Aad
- Menyebabkan pohon tumbuh ke kiri

Contoh Pohon

- S => aAc
- A => Ab | {}

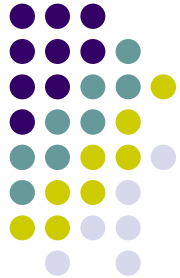


Penghilangan Rekursif Kiri



- Pisahkan aturan produksi yang rekursif kiri dengan yang tidak.
  - Yang rekursif kiri :
    - A => Aa1 | Aa2 | Aa3 | …
  - Yang tidak rekursif kiri, termasuk produksi epsilon:
    - A => b1 | b2 | b3 | …
- Kita bisa tentukan a1, a2, a3, … dan b1, b2, b3, …
- Lakukan penggantian aturan produksi yang rekursif kiri menjadi:
  - A => b1Z | b2Z | b3Z | …
  - Z => a1 | a2 | a3 | …
  - Z => a1Z | a2Z | a3Z | …
- Penggantian itu dilakukan untuk setiap aturan produksi dengan simbol di ruas kiri yang sama. Bisa muncul simbol variabel baru Z1, Z2, Z3… 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 => Sab | aSc | dd | ff | Sbd

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

Hasil akhir:
S => aSc | dd | ff
S => aScZ1 | ddZ1 | ffZ1
Z1 => ab | bd
Z1 => abZ1 | bdZ1

Contoh lain:

- S => Sab | Sb | cA
- A => Aa | a | bd

- Yang rekursif : S => Sab | Sb dan A => Aa
- Yang tidak: S => cA dan A => a | bd

Pergantian:

- Untuk S => Sab | Sb
  - S => cAZ1
  - Z1 => ab | b
  - Z1 => abZ1 | bZ1
- Untuk A => Aa
  - A => aZ2 | bdZ2
  - Z2 => a
  - Z2 => aZ2

Hasil:
S => cA
A => a | bd
S => cAZ1
Z1 => ab | b
Z1 => abZ1 | bZ1
A => aZ2 | bdZ2
Z2 => a
Z2 => aZ2

Contoh lain:

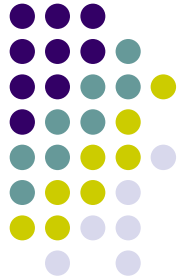
- S => aA | b | cS
- A => Sd | e #rekursif kiri

Jadi:

S => aA | b | cS

A => aAd | bd | cSd | 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 => dS

  B => adB

Membuat pohon sintaks melebar ke kanan

NEXT

- Analisis Sintaks 2
- Transformasi TBBK dan CYK