計算理論　第9回
第5章 : 文脈自由文法と言語(2/2)

基礎工学部情報科学科
中川　博之

本日の概要

- 第5章 : 文脈自由文法と言語 (の後半)
  - テキスト : p.216〜
  - 5.3 文脈自由文法の応用
  - 5.4 文法と言語のあいまいさ

- 重要概念
  - 構文木，パーサー，あいまいな文法

（復習）文脈自由言語，文脈自由文法

- 文脈自由言語 (CFL)
  - 正則言語よりも広い言語クラス
  - 再帰的な構造を扱うことが可能
    - 正則言語では再帰的構造を扱うことができない
- 文脈自由文法 (CFG)
  - 文脈自由言語の記述に用いる文法
  - 以下の4つ組で定義： G = (V, T, P, S)
    - V: 変数（非終端記号）集合
    - T: 終端記号集合
    - P: 生成規則集合
    - S: 出発記号

5.3 文脈自由文法の応用

文脈自由文法の歴史

- Chomskyが考案
  - 言語学者
  - 元々は，自然言語を記述するために考案
  - 文脈依存文法，句構造文法なども考案
- 計算機科学にて再帰的構造の有用性が評価
  - 構造データ処理の自動化に使える
  - 応用：パーサー，マークアップ言語

パーサー (Parser)

- 構文解析器とも呼ばれる
  - ソースコードに記載された構文を解析する
- ソースコードでは再帰構造がよく出現
  - 数式における括弧: “(” と “)”
  - ブロックの範囲記述: “{” と “}” や begin-end
- 再帰構造の開始と終了は正しく対応する必要がある
  - ○: {{{}}{}{}}
  - ×: {{{{}}}

正則言語の限界

- バランスの取れた括弧の一例を表す記述
  - 言語L = { $(^n)^n$: n≥1}
- 言語Lは正則言語ではない
  - L’ = { $a^n b^n$: n≥1} が正則言語でないのと同じ
    → 言語Lを表現した正則表現は存在しない
    → 正則表現では再帰構造を表現できない
    → プログラミング言語は正則表現では十分に表現できない
- 文脈自由文法であれば再帰構造を表現可能

バランスが取れた括弧を表す文法

- バランスが取れた括弧の列の例
  - ε, (()), ()(), ((()(())))

- 文法 G_{bal} = ({B}, {(,)}, P, B)
- 生成規則の集合P
  - B→ BB
  - B → (B)
  - B → ε

IF文を表す文法

- IF文にはelse節が無くても良い
- 文法 G_{if} = ({S}, {i, e}, P, S)
  - iはif節を，eはelse節を表す
- 生成規則の集合P
  - S→ ε | SS | iS | iSeS
- 導出の例
  - S ⇒ iS ⇒ iiSeS ⇒ iieS ⇒ iie
  - S ⇒ SS ⇒ iSeSS ⇒ iiSeeSS ⇒ iieeSS ⇒ iieeS ⇒ iieeiS ⇒ iieei

yacc

- yacc: Parser-Generatorのひとつ
- Parser-Generator
  - 入力：文脈自由文法
  - 出力：パーサー（構文解析器）のプログラムコード
    - 文法に合致するパーサープログラムを生成
- 代表的なParser-Generator
  - yacc, bison (yaccの上位互換), JavaCC

yaccの一例

```
Exp: Id {...}
     | Exp '+' Exp {...}
     | Exp '*' Exp {...}
     | '(' Exp ')' {...}
     ;
```

```
Id : 'a'  {...}
     |'b'  {...}
     | Id 'a'  {...}
     | Id 'b'  {...}
     | Id '0'  {...}
     | Id '1'  {...}
     ;
```

- 生成記号の “→” は “:” で表現
- 終端記号は引用符（single quote （’））で囲む
- 引用符無しの文字は変数
- {...} 内には実行して欲しいコードを記述（アクション）

マークアップ言語

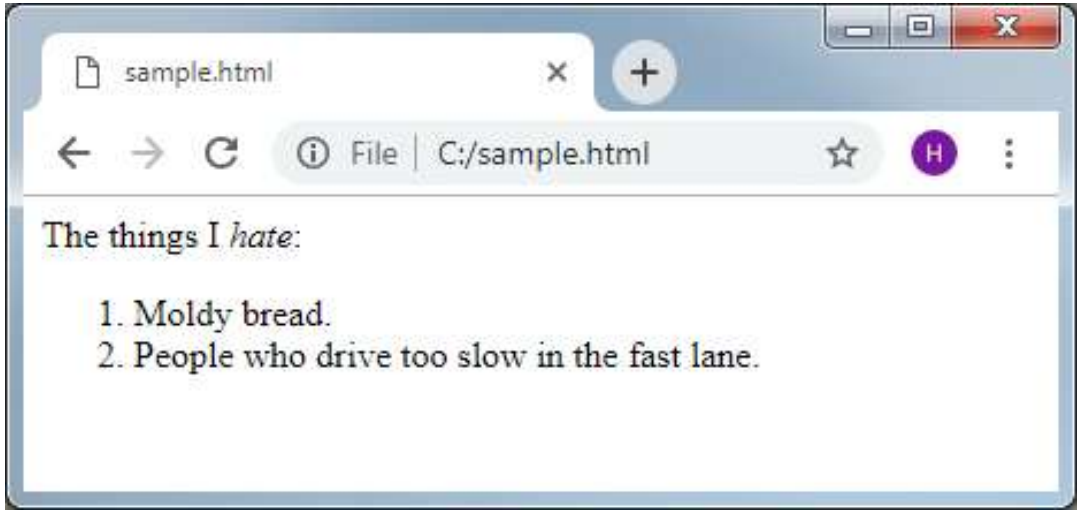
- 各種マーク(タグ)を文書内に埋め込むための規則
  - plain text上で構造を記述する手法
- 例: HTML (Hypertext Markup Language)
  - Webページ用の文書ファイル記述言語
- 例 : XML (eXtensible Markup Language)
  - 文書構造を記述するための言語

HTMLの例（図5.12）

```
<P>The things I <EM>hate</EM>:
<OL>
<LI>Moldy bread.
<LI>People who drive too slow in the fast lane.
</OL>
```

HTML文法の一部

- Doc→ε | Element Doc
- Element → Text |
  <EM> Doc <EM> |
  <P> Doc |
  <OL> List </OL> | ...
- List → ε | ListItem List
- ListLitem → <LI> Doc
- ...



XML

- XML (eXtensible Markup Language)
  - DTD (Document-Type Definition)により文書構造を定義
  - DTDは本質的に文脈自由文法
  - DTDに従うXML文書のみが受理される

パソコン仕様記述用DTD（図5.14）

```
<!DOCTYPE PcSpecs[
        <!ELEMENT PCS (PC*)>
        <!ELEMENT PC (MODEL, PRICE, PROCESSOR, RAM, DISK+)>
        <!ELEMENT MODEL (\#PCDATA)>
        <!ELEMENT PRICE (\#PCDATA)>
        <!ELEMENT PROCESSOR (MANF, MODEL, SPEED)>
        <!ELEMENT MANF (\#PCDATA)>
        ...
        <!ELEMENT DISK (HARDDISK | CD | DVD)>
        ...
]>
```

CFG PcSpecsの生成規則：

- PCS → PC*
- PC → MODEL PRICE PROCESSOR RAM DISK+
- MODEL → 文字列
- …
- DISK → HARDDISK | CD | DVD
- …

DTDに従うXMLの記述例（図5.15）

```
<PCS>
  <PC>
    <MODEL>4560</MODEL>
    <PRICE>$2295</PRICE>
    <PROCESSOR>
      <MANF> Intel</MANF>
      <MODEL>Pentium
              </MODEL>
      <SPEED>800MHz</SPEED>
      </PROCESSOR>
      <RAM>256</RAM>
      <DISK><HARDDISK>
        ...
    </PC>
    <PC>
      ...
    </PC>
</PCS>
```

5.4 文法と言語のあいまいさ

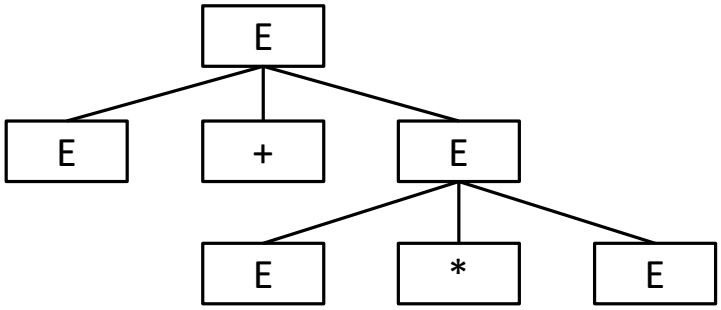
あいまいな文法

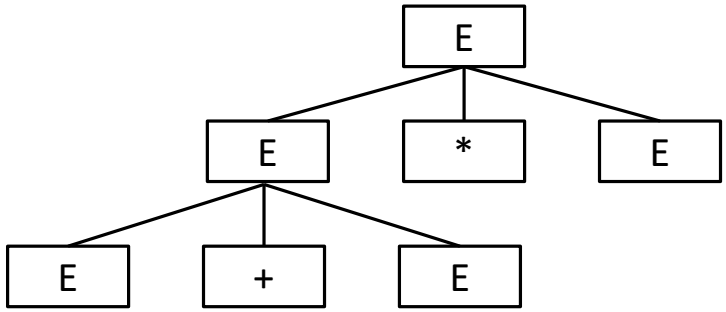
- 定義 : 文法 G = (V, T, P, S)があいまいである
  → ある終端記号列wに2つ以上の異なる構文木が存在

- 定義 : 文法 G = (V, T, P, S)があいまいでない
  → 任意の終端記号列wの構文木は高々1つ存在

例5.25：式文法 G_{exp}

- (再掲) 式文法 G_{exp}
  - E → I | E+E | E*E | (E)
  - I → Ia | Ib | I0 | I1 | a | b
- 文形式 E+E*Eに対し2通りの構文木が存在

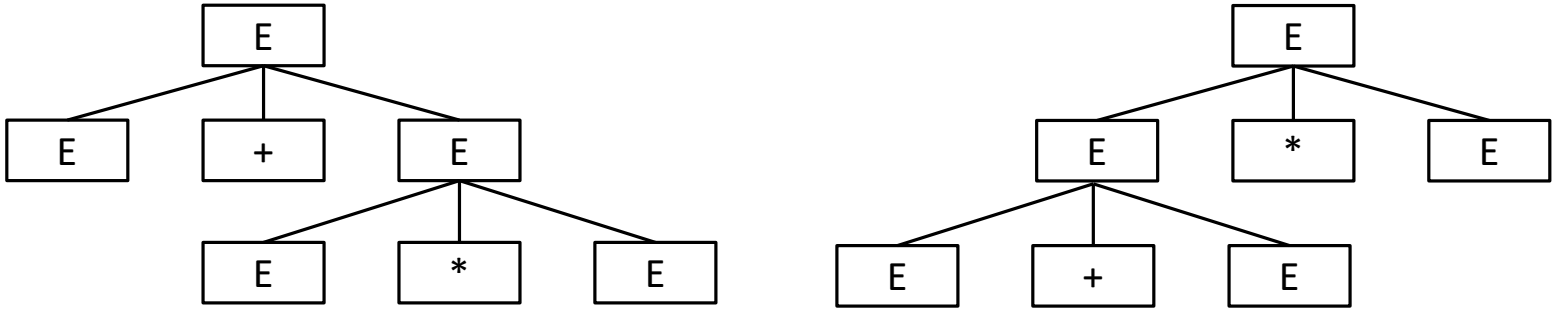




  - 左の構文木: E ⇒ E+E ⇒ E+E*E
  - 右の構文木: E ⇒ E*E ⇒ E+E*E

あいまいだとどうなるか

- 複数の解釈が存在

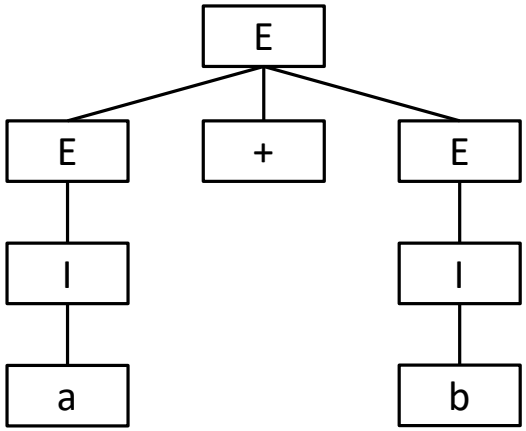


  - 解釈1：E+(E*E)
  - 解釈2：(E+E)*E

→ 意図と違う解釈，計算結果となる可能性があるが，
文形式から正しい解釈(意味)を決定できない

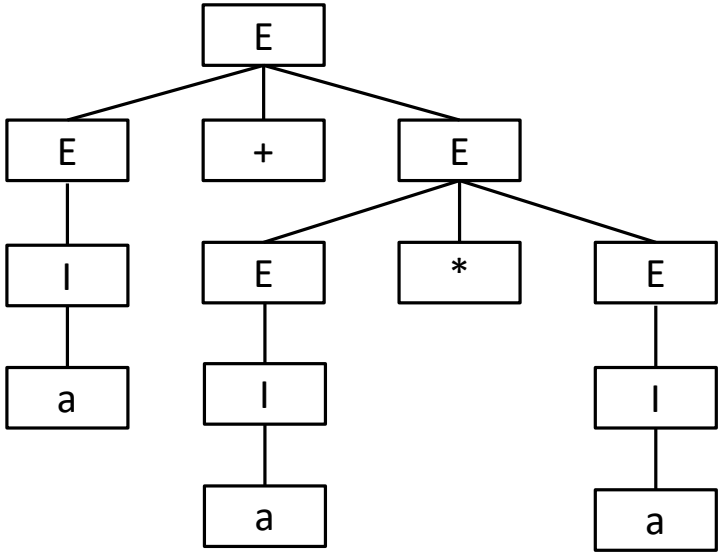
（参考）複数の導出過程

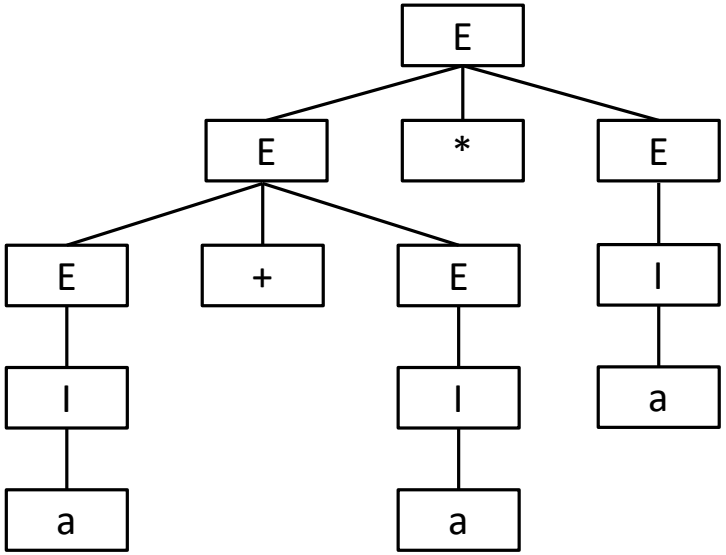
- 1つの列に対する複数通りの導出
  - あいまいでない文法でもあり得る
- 例:式文法 G_{exp}における a+bの導出
  - E ⇒ E+E ⇒ I+E ⇒ a+E ⇒ a+I ⇒ a+b
  - E ⇒ E+E ⇒ E+I ⇒ E+b ⇒ I+b ⇒ a+b
  - 構文木は同じ



あいまいであることの証明

- ある終端記号列wが存在して，2つ以上の異なる構文木の存在を示せばよい
- 例: w = a+a*a





あいまいであることの判別

- 与えられたCFGがあいまいであるかどうかの判別

  → 判定アルゴリズムは存在しない

  – 決定不能であるという

- あいまいなCFGしか存在しないCFLもある

  – 後ほど紹介

あいまいさの除去

- 通常のプログラミング言語で現れるあいまいさは軽微
  - 除去のための良く知られた技法がある
- 先の数式の例：あいまいさの原因は？
  - 原因1：演算子の優先度が考慮されていない
    - *より+が先に計算されることを許している
  - 原因2：優先度が同一の演算子に対して結合順序が考慮されていない
    - E+E+Eに対し，E+(E+E)と(E+E)+Eの両方を許している

あいまいさ除去の方針

- 方針1: 演算子の優先度を導入
  - *は+よりも優先度を高く
- 方針2: 同一優先度の演算子の結合順序を決定
  - 左結合性，右結合性のいずれかに決めておく
  - E+E+Eに対して
    - 左結合性: (E+E)+E と解釈する
    - 右結合性: E+(E+E) と解釈する

具体的な手段

- 結合の強さごとに異なる変数を用意
  - <式> (E) : 一つ以上の<項>の和
  - <項> (T) : 一つ以上の<因数>の積
  - <因数>(F) : <識別子>あるいは(<式>)
    - (<式>) : 括弧で囲まれた式

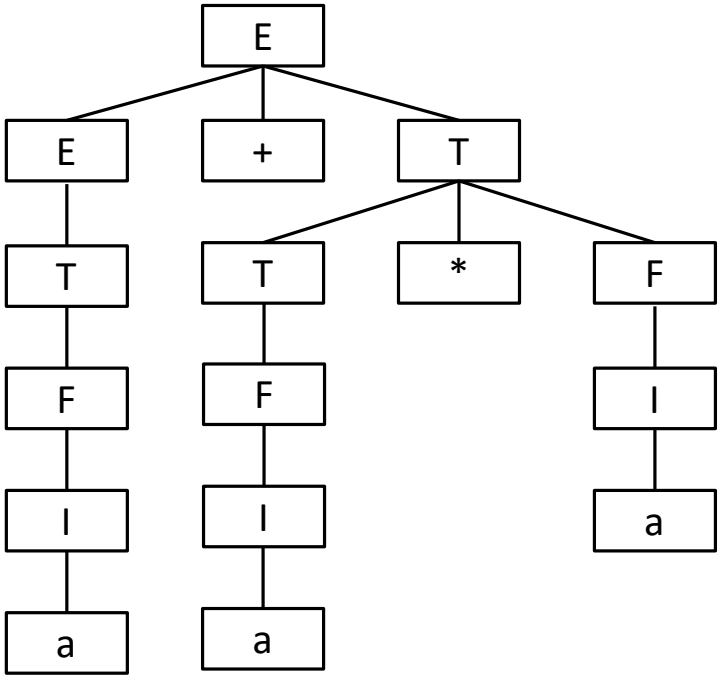
式に対するあいまいでない文法

- 以下の生成規則を持つ文法
  - <式>： E→ T | E+T
  - <項>： T→ F | T*F
  - <因数>： F→ I | (E)
  - <識別子>: I→a | b | Ia | Ib | I0 | I1

  - この文法での演算子は左結合性を持つ

式に対するあいまいでない文法

- a+a*aに対する構文木（これ1つしかない）



（参考）yaccでのあいまいさ対策

- 演算子の優先順位を指定できる
  - 順位の低いものから高いものへと記述
- 演算子の結合性（左結合or右結合）を指定できる
  - キーワード%left あるいは %right
  - %left ‘+’
  - %left ‘*’

*の方が+より優先順位が高く，
+と*はいずれも左結合

あいまいさと最左(最右)導出

- 文法があいまいでなくても，導出はただ一つとは限らない
- ただし，あいまいでない文法では，最左導出も最右導出もそれぞれただ一つ

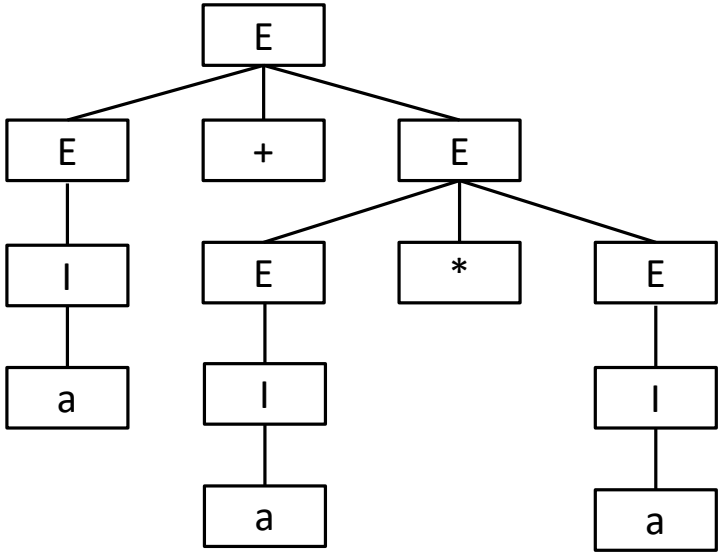
- [定理5.29] 文法G=(V,T,P,S)と終端記号列wに対して，wが2つの構文木を持つ

  ⇔ Sからwを導く2つの異なる最左導出が存在

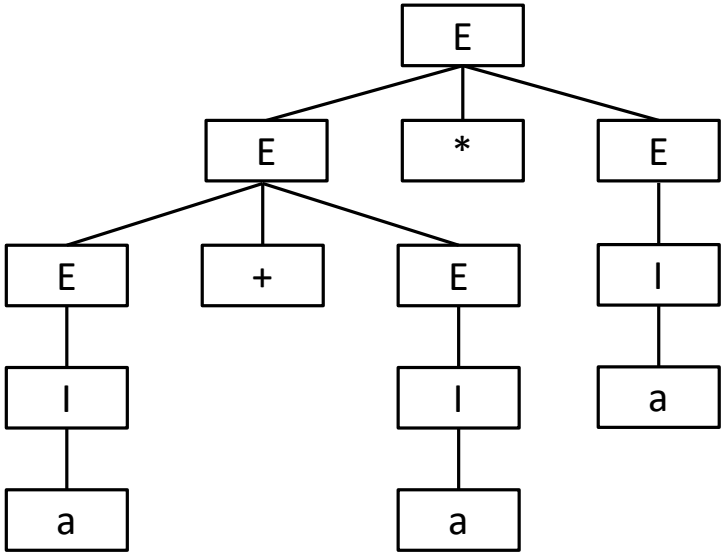
  – 証明は省略
  – 最右導出についても同様

2つの構文木と最左導出

- (a) $E \underset{左}{\Rightarrow} E+E \underset{左}{\Rightarrow} I+E \underset{左}{\Rightarrow} a+E \underset{左}{\Rightarrow} a+E*E \underset{左}{\Rightarrow} a+I*E \underset{左}{\Rightarrow} a+a*E \underset{左}{\Rightarrow} a+a*I \underset{左}{\Rightarrow} a+a*a$
- (b) $E \underset{左}{\Rightarrow} E*E \underset{左}{\Rightarrow} E+E*E \underset{左}{\Rightarrow} I+E*E \underset{左}{\Rightarrow} a+E*E \underset{左}{\Rightarrow} a+I*E \underset{左}{\Rightarrow} a+a*E \underset{左}{\Rightarrow} a+a*I \underset{左}{\Rightarrow} a+a*a$



(a)



(b)

本質的なあいまいさ

- [定義] 文脈自由言語Lが本質的にあいまい
  ⇔ Lを生成する文法のすべてがあいまい
  - Lを生成するあいまいでない文法が1つでも存在
    ⇔ Lは本質的にあいまいではない

- 本質的にあいまいな言語の例
  - $L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

Lの本質的なあいまいさの直観的説明 - Lを生成する文法G -

- $L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

- Lを生成する文法G
  - S→ AB|C
    - ABが1つ目の集合，Cが2つ目の集合に対応
  - A → aAb | ab
  - B → cBd | cd
  - C → aCd | aDd
  - D → bDc | bc

Lの本質的なあいまいさの直観的説明 - 複数の最左導出の存在 -

- このとき，$a^n b^n c^n d^n$ は以下の2通りの方法で導出できる
  - 導出1: $S \underset{左}{\Rightarrow} AB \underset{左}{\Rightarrow} aAbB \overset{*}{\underset{左}{\Rightarrow}} a^n b^n B \underset{左}{\Rightarrow} a^n b^n cBd \overset{*}{\underset{左}{\Rightarrow}} a^n b^n c^n d^n$
  - 導出2: $S \underset{左}{\Rightarrow} C \underset{左}{\Rightarrow} aCd \overset{*}{\underset{左}{\Rightarrow}} a^n Dd^n \underset{左}{\Rightarrow} a^n bDcd^n \overset{*}{\underset{左}{\Rightarrow}} a^n b^n c^n d^n$
- 最左導出（構文木）が2つ存在
  → この文法はあいまい

ミニレポート

ミニレポート：9-1

- テキスト p.231　問5.3.2：
  - (概要) 二つの型の括弧，つまり丸括弧() と角括弧[]のバランスの取れた列のすべて，またそれらだけを生成する文法を設計せよ.

- 例：
  - ()
  - []()
  - ([()]()([]))[][([][])]](()(()))

ミニレポート

ミニレポート:9-2

- テキストp231 問5.3.5
- 図5.16(下図)のDTDを文脈自由文法に変換せよ.

```
<!DOCTYPE CourseSpecs [
  <!ELEMENT COURSES (COURSE+)>
  <!ELEMENT COURSE (CNAME, PROF, STUDENT*, TA?)>
  <!ELEMENT CNAME (\#PCDATA)>
  <!ELEMENT STUDENT (\#PCDATA)>
  <!ELEMENT TA (\#PCDATA)>
]>
```