計算論 A / 2015年度

第5章 文脈自由文法と言語 (2/2)

文脈自由文法の応用

角川　裕次

第5章 文脈自由文法と言語 (2/2)
【教科書 p216〜】

＊＊＊ 本日の重要概念 ＊＊＊

パーサー生成系, あいまいな文法

φ(・ω・)メモメモ

5.3 文脈自由文法の応用

文脈自由文法の考案の経緯

当初の目的は自然言語の記述

- 正規文法, 文脈自由文法, 文脈依存文法, 句構造文法
- 考案はチョムスキー(言語学者)

計算機科学における再帰的構造の記述での有用性を発見

- 構造データの処理の自動化に有用

文脈自由文法の応用: プログラミング言語の文法定義

パーサー (Parser)

- プログラムソースコードの文法構造の算出
- 構文解析器とも称す

パーサー生成系 (Parser-Generator)

- 文法からパーサーを自動生成するソフトウエア
- コンパイラコンパイラとも称す

文脈自由文法の応用: 構造文書

XML (Extensible Markup Language)

- 構造型文書の表現法のひとつ
- 文書型定義により使用可能なタグや入れ子構造を定義
- 文書型定義はDTD (Document-Type Definition)とも称す

5.3.1 パーサー

プログラミング言語の文法と文脈自由文法

プログラムのソースコードでは再帰構造が有用

- 数式での開き括弧 (と閉じ括弧)
- 配列の添字の括弧 [と]
- ブロックの開始 { と終了 } あるいは begin と end

再帰構造の開始と終了は正しく対応する必要有り

- ○ {{}{}{{}}}
- × $\underline{\{}\{\{\}\}$ — 最初の開き括弧に対応する閉じ括弧がない

入れ子構造と文脈自由文法

言語 $L = \{(^n)^n \ : \ n \geq 1\}$ は正規言語でない

- $L' = \{a^n b^n \ : \ n \geq 1\}$ が正規言語でないのと同じ理由

→ 言語 L を生成する正規文法は存在しない

→ 再帰構造は正則表現では表せない

文脈自由文法を使えば再帰構造を表現できる

例5.19: バランス括弧文法 G_{bal} (1/2)

バランスがとれた括弧の列の例

- ε
- ()()
- ((()))
- (())()

文法 $G_{bal} = (B, \{(,)\}, P, B)$

生成規則の集合 P

- $B \to BB$
- $B \to (B)$
- $B \to \varepsilon$

例5.19: バランス括弧文法 G_{bal} (2/2)

示すべきこと:
$L(G_{bal}) =$ バランスがとれた括弧の列すべての集合

w は文法 G_{bal} で導出される語
$\Rightarrow$ w はバランスがとれた括弧の列

- 生成規則から明らか

w は任意のバランスがとれた括弧の列
$\Rightarrow$ w 文法 G_{bal} で導出できる (要証明)

- 証明は w の長さに関する帰納法 (詳細省略)

入れ子になったIF文

IF文にはELSE部があってもなくても良い

文法: $S \ \to \ \varepsilon \mid SS \mid iS \mid iSe$

- ただし i はIF部を, e は ELSE部をそれぞれ表す
- 例: 「if (v=0) x++; else y++;」の場合
  i = 「if (v=0) x++;」
  e = 「else y++;」

導出の例

- $S \Rightarrow SS \Rightarrow iSeS \Rightarrow ieS \Rightarrow ieiSe \Rightarrow ieie$
- $S \Rightarrow SS \Rightarrow iSS \Rightarrow iS \Rightarrow iiSe \Rightarrow iie$
- $S \Rightarrow SS \Rightarrow iSeS \Rightarrow ieS \Rightarrow ieiS \Rightarrow iei$

5.3.2 YACC パーサー生成プログラム

パーサー(Parser)

入力: ソースコード

出力: 入力ソースコードの構文木

- 文法に従ってソースコードの構文を解析

パーサー生成系(Parser-Generator)

入力: 文法

出力: パーサーのプログラムコード

- 文法に合致したパーサーのプログラムを生成

パーサー生成系の例

- YACC
- BISON (YACC の上位互換)

YACC パーサー生成系の動作

入力は以下のものの対

- 文法における生成規則の集合
- 各生成規則に対応するアクション

アクション(動作規則)とは

- 生成規則に対応する構文木の節点が作られる時の動作
- 例: 対応するオブジェクトコード断片の生成

詳しいことはコンパイラの講義で勉強して下さい

5.3.3 マークアップ言語

マークアップ言語

各種のマーク(タグ)を文書内に埋め込むための規則

文書の構造を記述する手法

- 文書はプレインテキストで書いてある

例: SGML

- 組版, 文書自動処理用途の文書ファイル書式
- SGML = Standard Generalized Markup Language

例: HTML

- ウェブページ用の文書ファイル書式
- HTML = Hypertext Markup Language

5.3.4 XML と文書型の定義

XML

マークアップ言語のひとつ

DTDで文書構造を定義

- DTD = Document-Type Definition; 文書型定義

DTDは本質的に文脈自由文法

XML文書： DTD(文法)で生成されるもの(語)

DTDの例

教科書を見ておいて下さい

5.4 文法と言語のあいまいさ

5.4.1 あいまいな文法

あいまいな文法: 定義

文法 $G = (V, T, P, S)$ があいまいであるとは

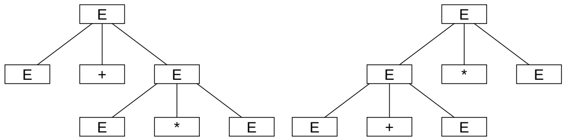
- ある終端記号列wに2つ以上の異なる構文木が存在

文法 $G = (V, T, P, S)$ があいまいでないとは

- 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}

文形式 $E + E * E$ に対し2通りの導出木が存在



- $E \Rightarrow E + E \Rightarrow E + E * E$ (左の構文木)
- $E \Rightarrow E * E \Rightarrow E + E * E$ (右の構文木)

再掲: 式文法 G_{exp}

- $E \to I \mid E + E \mid E * E \mid (E)$
- $I \to Ia \mid Ib \mid I0 \mid I1$

例5.26: 複数通りの導出

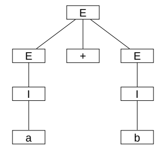
ひとつの文形式に複数通りの導出が存在する場合あり

- あいまいでない文法でもそうなる場合あり

例: 式文法 G_{exp} での $a + b$ の導出2種

1. $E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$
2. $E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$

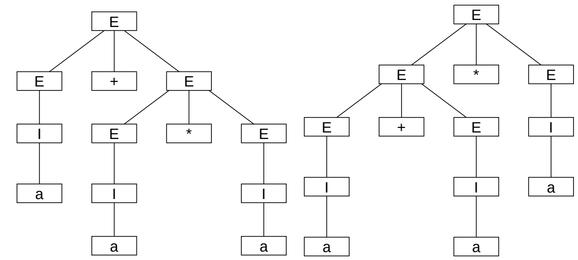
- 構文木は互いに同じ



式文法 G_{exp}はあいまいである

証明方法: ある終端記号列wに対し
2つ以上の異なる構文木の存在を示す

具体例: $w = a + a * a$

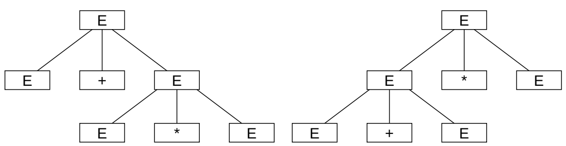


文形式 $E + E * E$ に対する2通りの導出

2通りの式の意味が存在

意味1: $E + (E * E)$

意味2: $(E + E) * E$



どちらの意味にもとれてしまう

対処法

優先度を導入して採用する導出木を1つに定める

文法を直して複数の導出木ができないようにする

5.4.2 文法のあいまいさの除去

文法のあいまいさの除去: 一般論

CFGのあいまい性判定問題は決定不能

- 判定するアルゴリズムは存在しない, という意味
- 決定不能性については本講義最終回で説明

あいまいなCFGしか存在しない文脈自由言語Lが存在

- 5.4.4節にて説明

文法のあいまいさの除去: 現実的な対処法

普通のプログラミング言語の文法: あいまいさは軽微

- 軽微なあいまいさを除去する経験則あり

あいまいさの典型例: 数式, IF-THEN-ELSE

- $E + E * E$
  — $E + (E * E)$ と $(E + E) * E$ どっち?
- IF v THEN w IF x THEN y ELSE z
  — 最後の「ELSE z」はどちらの IF に対応してるの?

式文法 G_{exp}のあいまいさの原因

原因1: 演算子の優先度の考慮がない

- $*$より$+$を先に計算することを許している

原因2: 同一優先度の演算子の結合順序の考慮がない

- $E + E + E$に対し
  $E + (E + E)$と$(E + E) + E$の両方を許している

式文法の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する

方針2: 同一優先度の演算子の結合順序を決める

- 左結合性あるいは右結合性のどちらかに決めておく
- 左結合性: $E + E + E$は$(E + E) + E$の意味にとる
- 右結合性: $E + E + E$は$E + (E + E)$の意味にとる

あいまいでない式文法の構成

結合の強さが同じ水準の式を表す変数をいくつか導入

《式》

- ひとつ以上の《項》の和

《項》

- ひとつ以上の《因数》の積

《因数》

- 《識別子》　あるいは
- (《式》)　　(括弧で囲まれた式)

例5.27. あいまいでない式文法

《式》 $E \rightarrow T \mid E + T$

《項》 $T \rightarrow F \mid T * F$

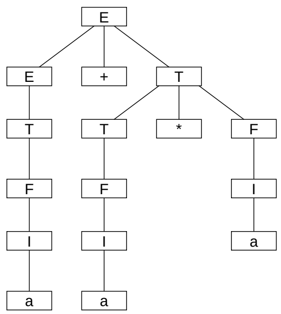
《因数》 $F \rightarrow I \mid (E)$

《識別子》 $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

- この文法での演算子は左結合性を持つ

例5.27. あいまいでない式文法

$a + a * a$ に対する構文木 (これ1つしかない)



YACC/BISONでのあいまいさ対処法

演算子の優先順位を指定できる

演算子の結合性(左結合性/右結合性)を指定できる

5.4.3 あいまいさを表現する手段としての最左導出

定理 5.29: あいまいさと複数の最左導出

文法 $G = (V, T, P, S)$ と終端記号列 w に対し

w が2つの異なる構文木を持つ

$\Leftrightarrow$ S から w への2つの異なる最左導出が存在

(証明略)

最右導出においても同様な定理が成立

5.4.4 本質的なあいまいさ

本質的なあいまいさ

定義: 文脈自由言語 L は本質的にあいまい

$\Leftrightarrow$ L を生成する文法のすべてがあいまい

注意 : CFL L にあいまいでない CFG が存在

$\Rightarrow$ 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

本質的にあいまいな言語 L

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

注: L は以下の条件を満たす $a^+ b^+ c^+ d^+$ に属する列

- a と b が同数かつ c と d が同数, あるいは
- a と d が同数かつ b と c が同数

Lの本質的なあいまいさの直観的説明

導出では以下の2通りが行なわれるはず

- a と b が同数かつ c と d が同数、を維持する導出
- a と d が同数かつ b と c が同数、を維持する導出
- どんな文法でも必ずそうしているはず

$a^n b^n c^n d^n$ (全ての文字が同じ数)の導出

- 上記2通りの方法があるはず
- → 異なる2つの構文木が存在

おわり

本日のCLEミニレポート

教科書231ページ 問5.3.5