プログラミング応用
http://bit.ly/kosen02

Week12@後期(week27 in 2016)
2016/12/15

復習：正規表現

- 正規表現を用いると記号列を
  受理するか却下するか決定可能
  (以下はgrepコマンドでの正規表現の例)

- 5の倍数を受理(末尾が0もしくは5で終わる)
  [0-9]*[05]

- Helloもしくはworldが0回以上連続する文字列の
  受理
  (hello | world)*

質問1

受理する英単語は正規表現で記述可能
文法的に正しい英文を受理する正規表現は記述できる？

肯定文を受理したい
　I have a pen
　I have an apple
　→ 正規表現: I have (a|an) (pen|apple)

否定文も受理したい
　I don’t have a pen.
　→ 正規表現: I (don’t | doesn’t)* have (a|an) (pen|apple)

肯定文、否定文は書けるかもしれない。

質問2

- 関係代名詞を含む文や命令文も受理するか判定する正規表現は書ける？

  関係代名詞(that)
  I said that he has an apple.
  命令文(動詞やDon’tから始まる)
  Don’t have an apple.

  → 複雑な規則を1つの正規表現で書くのは
  <u>不可能</u>

本日の内容

- 講義(短縮):
  - 文脈自由文法(CFG)とバッカスナウア記法(BNF)
- 演習:
  - 文脈自由文法が受理する文字列
  - 英文法の生成規則を記述する

文脈自由文法(CFG)と例

- 文脈自由文法(CFG; Context-free Grammar)
  - 受理される文の構文規則を定義したもの
    (= 生成規則という)
  - 「○○はxxである」という規則を記述
- 例）英文法(一部)のCFGによる生成規則
  - <文>　→ <叙述文> | <命令文>
  - <叙述文> → <主語> <動詞> | <主語> <動詞> <名詞>
    <命令文> → <動詞> | <動詞> <名詞>
  - <主語> → I | You
    <動詞> → play|buy | run
    <名詞> → an apple| a pineapple

終端記号/非終端記号

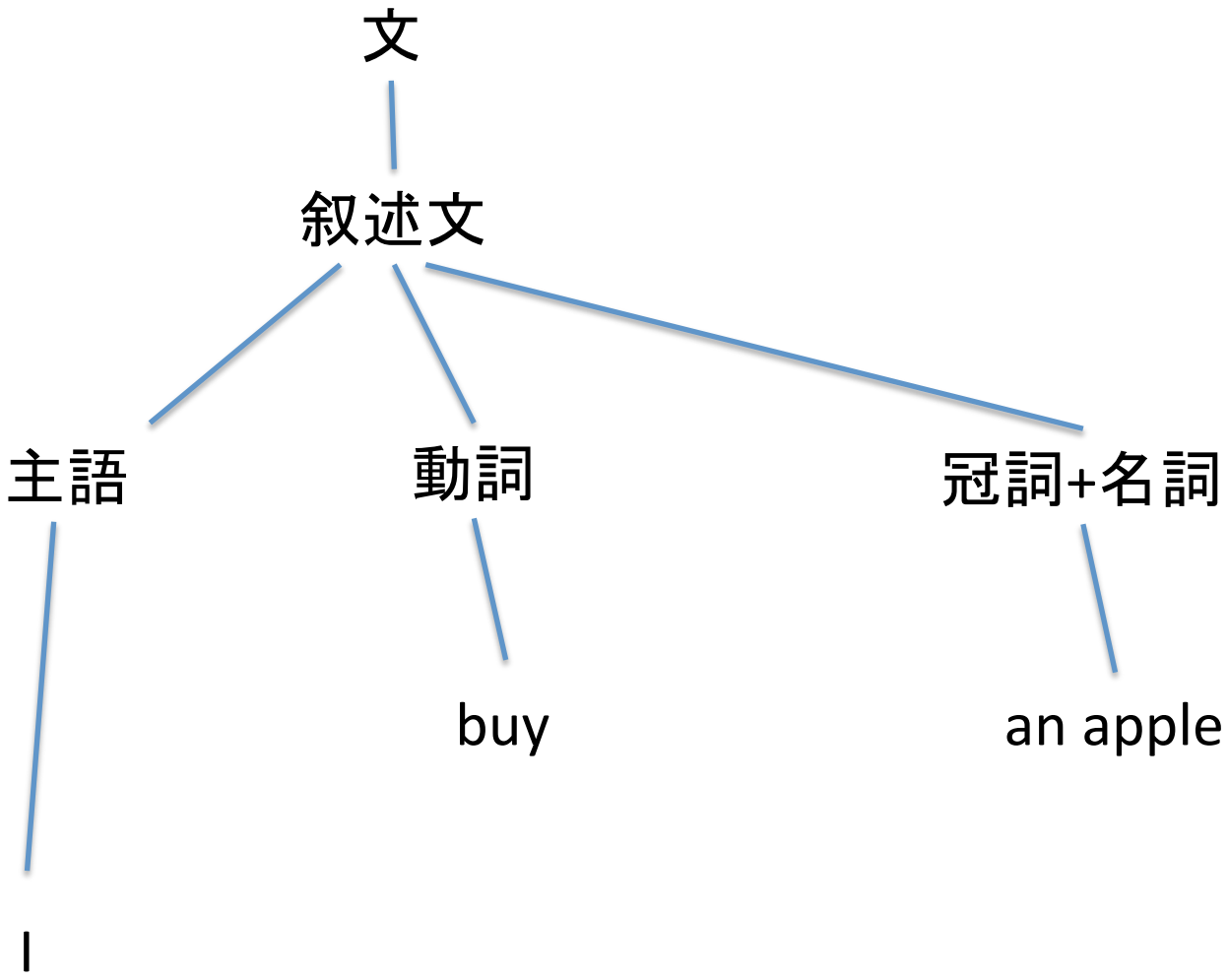
- 終端記号(Terminal Symbol)
  - これ以上展開されない文字列
    例) an apple, a pineappleは終端記号
    <名詞> → an apple | a pineaaple
- 非終端記号(Non-terminal Symbol)
  - さらに展開される文字列
    例) <叙述文><主語><動詞>は非終端記号
    <叙述文> → <主語> <動詞> | …

文脈自由文法の表現方法

1. 展開
   <記号1> → <記号2>　記号3
2. 記号の連続
   <記号1><記号2> 記号3 <記号4>
3. 論理和
   <記号1> | <記号2>

文脈自由文法の受理

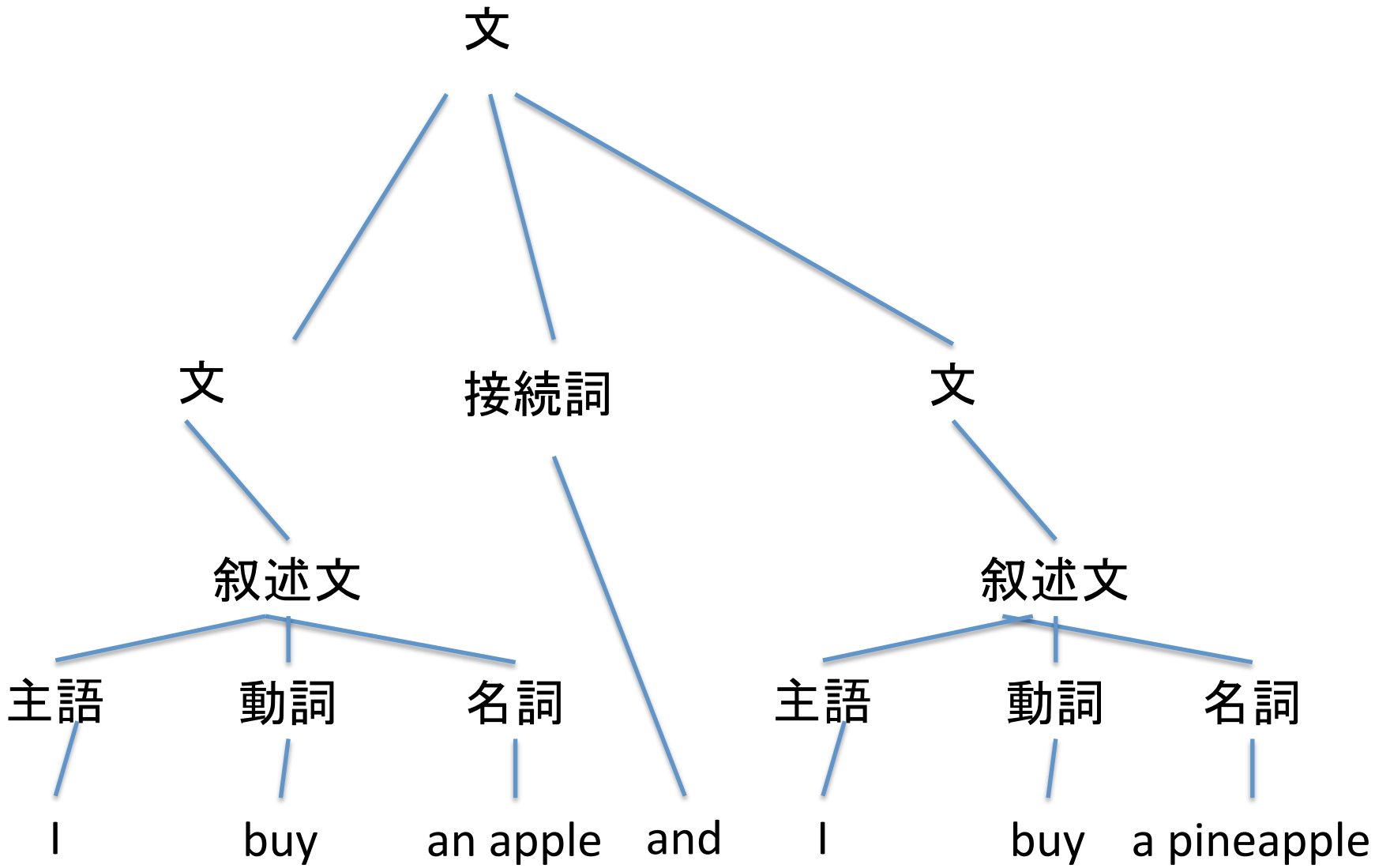
- 受理される文
  - 規則を展開していき文が生成されれば受理
- 却下される文
  - 規則をどのように展開しても文が生成なければ却下



文から順に展開していくと「I buy an apple」が生成されるので受理
このような木のことを構文木と呼ぶ

再帰するCFG(例1)

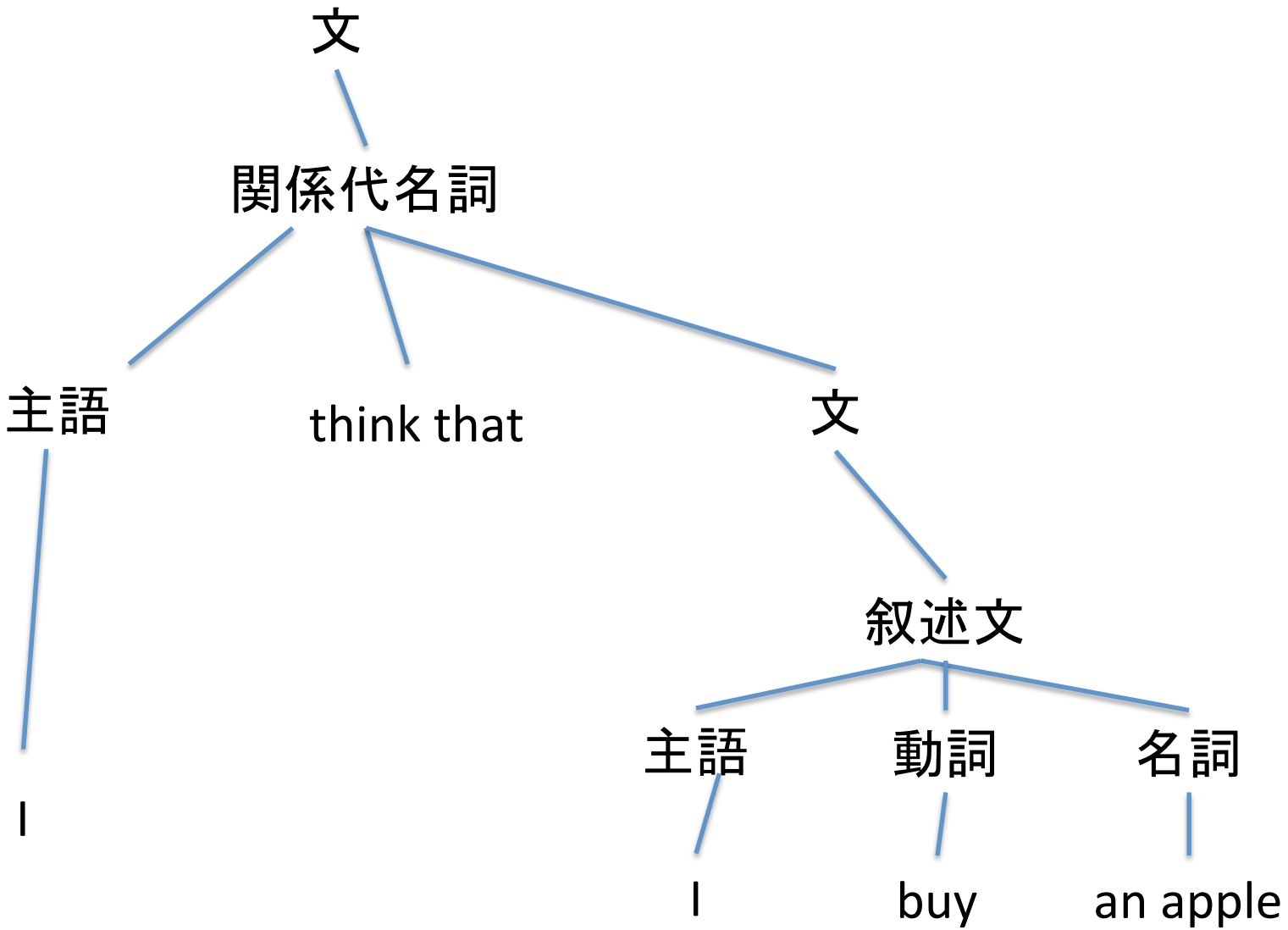
- 例) 接続詞(and/or)も含むより複雑な英文法
  - <文>　→ <叙述文> |<命令文> | <文><接続詞><文>
  - <接続詞> → and | or
  - <叙述文> → <主語> <動詞> | <主語> <動詞> <名詞>
    <命令文> → <動詞> | <動詞> <名詞>
  - <主語> → I | You
    <動詞> → play|buy | run
    <名詞> → an apple| a pineapple



「I buy an apple and I buy a pineapple」がCFGにより生成された
→ 受理！

再帰するCFG

- 例) 関係代名詞も含むより複雑な英文法
  - <文>　→ <叙述文> |<命令文> | <関係代名詞節>
  - <関係代名詞節> → <主語> think that <文>
  - <叙述文> → <主語> <動詞> | <主語> <動詞> <名詞>
    <命令文> → <動詞> | <動詞> <名詞>
  - <主語> → I | You
    <動詞> → play|buy | run
    <名詞> → an apple| a pineapple



「I think that I buy an apple」を構文解析した結果
→ 受理！

バッカスナウア記法(BNF)

- バッカスナウア記法(BNF; Backus-Naur form)
  - CFGをプログラム上で表現するのによく使われる記法

- 規則1：規則の右辺と左辺は::=で繋ぐ
  規則2：非終端記号は<>でくくる
  規則3：論理和は|で記述する

  <関係代名詞> → <主語> “think” “that” <文>

  <関係代名詞> ::= <主語> ”think” “that” <文>

CFGの応用先

- プログラミング言語の作成
  - 言語仕様の策定
    - C/Javaなどメジャーな言語の仕様はBNFで記述される
  - コンパイラの構築
- 人間の使う言語(自然言語)を計算機で扱う際の基盤技術

演習1

– 以下のBNF記法で表現されたCFGが受理する文を3つ書きなさい。
(cfg_example.txtというファイル名で保存し提出)

<文> ::= <叙述文> |<命令文> | <関係代名詞>

<関係代名詞> ::= <主語> think that <文>

<叙述文> ::= <主語> <動詞> | <主語> <動詞> <名詞>
<命令文> ::= <動詞> | <動詞> <名詞>

<主語> ::= I | You
<動詞> ::= play|buy | run
<名詞> ::= an apple| a pineapple

演習2

- 演習1で作成した文に対して構文木を書きなさい。
  (提出不要)

演習3(疑問文)

- 演習1のBNFによる英文法の規則定義に、
「疑問文」を生成する規則を新たに追加し、
bnf_gimon.txtというファイル名で保存しなさい。

なお、疑問文は以下の3条件を満たすものとする。
1. doもしくはdoesから始まる
2. do/doesのあとに<文>が続く
3. 末尾が?で終わる

演習4(optional)

- 以下のURLではC言語の仕様がBNFで公開されています。読み解いてみましょう。
  - http://www.cs.man.ac.uk/~pjj/bnf/c_syntax.bnf
    補足:
    このページでは次回学習するlexというプラグラムに読み込ませる形式でBNFが記述されています。
    終端記号がシングルクオーテーション(')でくくられており、非終端記号に<>が付いていません。
  - 例) 変数宣言の構文を定義しているのは
    type_spec
  - 例) if文やswitch case文の構文を定義しているのは
    selection_stat

次回

- 終わらなかった演習は宿題にします。
- 次回は字句解析、構文解析、意味解析といったコンパイラ構築や計算機による言語理解の基礎になる概念を学習します。
- Lex/Yaccというツールを用いて文脈自由文法を用いたプログラムを実装します。
  - Lex: 字句解析用のツール
  - Yacc: 構文解析用のツール(BNFを記述する)