Generalized Parsing Expression Grammars

横浜国立大学
○千田忠賢　倉光君郎
2017/03/03 PRO113

解析表現文法 (PEG)

- 2004年に B. Ford が提案した形式文法

特徴

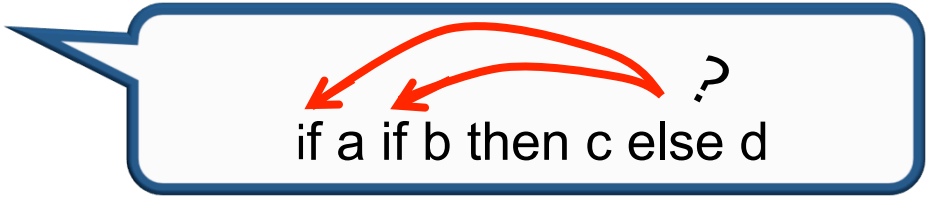
- 線形時間での解析が可能
  - Packrat構文解析
    (バックトラック付き再帰下降構文解析+メモ化)
- 曖昧性がない
- 実用的な演算子を持つ
  - **優先度付き選択**
  - 先読み

優先度付き選択

優先度付き選択 e_1/e_2

- e_2 はe_1が失敗した場合にのみ試される

- 良い点
  - 選択の曖昧性が原因となるような問題が起きない
    - dangling else 問題



- 悪い点
  - 直感に反する[1]

[1] Redziejowski, R. R.: Some Aspects of Parsing Expression Grammar, *Fundam. Inform.,* Vol. 85, No. 1-4, pp. 441-451, 2008

非直感的な振る舞い

例 1

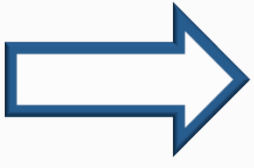
PEG　：(a / aa) b
入力1：ab
入力2：aab

(a / aa) b に次の入力を与えたとき、
マッチする文字列は何？もしくは失敗する？

Aは非終端記号

例 2

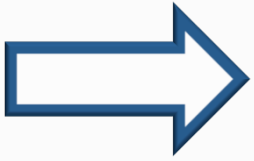
PEG　：A = a A a / aa
入力1：aa
入力2：aaaaaaaaaa
　　(=a^{10})



非直感的な振る舞い

例 1

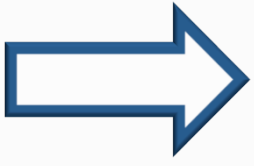
PEG　：(a / aa) b
入力1：ab
入力2：aab



入力１：abにマッチ
入力２：失敗

例 2

PEG　：A = a A a / aa
入力1：aa
入力2：aaaaaaaaaa
　　(=a^{10})



入力１：aaにマッチ
入力２：aaaaにマッチ

問題

- 優先度付き選択の非直感的な振る舞い
  - バグの原因となる

- 実際に文法を開発している際に発生したバグ

```
ConditionalExpression
        = LogicalOrExpression
        / ( LogicalOrExpression '?' Expression ':' AssignmentExpression )

ParameterDeclarationClause
        = ( ParameterDeclarationList? '...'? )
        / ( ParameterDeclarationList "," '...' )
```

EBNFのままの順番で定義してしまっている

前の表現が空文字を受理するため後ろに行くことはない

直感に反する理由

入力がe_1とe_2の両方にマッチする場合、
e_2 が試されることはない

文法を定義した人が意図しないところで
選択が終了してしまう

問題

- 優先度付き選択の非直感的な振る舞い
  - バグの原因となる

文法を定義した人が意図しないところで
停止することのない選択が欲しい

優先度無し選択

目的

- PEGに優先度無し選択を追加した新たな形式文法の定義
  - Generalized PEG(GPEG)として定義

Challenge :

- 優先度無し選択を使った場合でも、その結果に曖昧性がなければ線形時間で解析する

Positive Aspects :

- より直感的な文法定義が可能
- 自然言語処理への応用が可能

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

PEGの定義

- A ＝ e の形で表す
  - A：非終端記号
  - e：解析表現
- e ::= ε　　空文字
  - | a　　終端記号
  - | .　　任意の終端記号
  - | e e　連接
  - | e / e　優先度付き選択
  - | e*　　貪欲な繰り返し
  - | !e　　否定先読み
  - | &e　　肯定先読み（ ＝ !!e ）
  - | A　　非終端記号

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

Generalized PEG(GPEG)の定義

- A = e の形で表す
- e ::= ε　　空文字
  - | a　　終端記号
  - | .　　任意の終端記号
  - | e e　連接
  - | e / e　優先度付き選択
  - | e | e　**優先度無し選択**
  - | e*　貪欲な繰り返し
  - | !e　否定先読み
  - | &e　肯定先読み（ = !!e ）
  - | A　非終端記号

優先順位は最も低い

GPEG

例 1. A = a / b | c / d

- A = (a / b) | (c / d) と等価
- a,b,c,dのいずれかにマッチ

例 2. Sentence = NP VP

NP = the (man | bird)
VP = Verb NP
Verb = is (talking to)?

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

GPEGの解析アルゴリズム

- 方針：Packrat構文解析を拡張する
  - 結果が曖昧な場合は全て試すようにする
    - Packrat構文解析
      - 入力上の位置を一つの変数で管理
    - 拡張Packrat構文解析
      - 入力上の位置を集合で管理

Packrat構文解析を拡張

GPEG：
A ＝ (a | aa) b

Curr ＝ {0}

解析結果を保持する集合
入力上の位置か
失敗したという情報が入る

0 1 2 3
入力： a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {0}

Next = {}

解析結果を一時的に
保持する集合

0 1 2 3
入力：a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {0}
Next = {}

マッチ！

0 1 2 3
入力： a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {0}

Next = {1}

次に解析を開始する
入力上の位置は1

```
    0 1 2 3
入力： a a b
```

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {0}
Next = {1}

0 1 2 3
入力： a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {0}
Next = {1,2}

```
    0 1 2 3
入力：a a b
```

Packrat構文解析を拡張

GPEG：
A ＝ (a | aa) b

Curr ＝ {1,2}

(a|aa)の解析が終わったので
その結果をCurrに代入

0 1 2 3
入力： a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {1,2}
Next = {}

```
    0 1 2 3
入力： a a b
```

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {1,2}

Next = {fail}

解析に失敗した場合は
failが入る

```
    0 1 2 3
入力： a a b
```

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {1,2}
Next = {fail}

0 1 2 3
入力： a a b

Packrat構文解析を拡張

GPEG：
A = (a | aa) b

Curr = {1,2}
Next = {fail,3}

　　0 1 2 3
入力：a a b

Packrat構文解析を拡張

GPEG：
A ＝ (a | aa) b

Curr ＝ {fail,3}

Aに入力aabを与えると、
解析失敗もしくはaabにマッチする

```
    0 1 2 3
入力： a a b
```

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

計算量の解析

GPEGは

1. GPEGの各表現が曖昧性を含まないのであれば線形時間で解析可能

2. 曖昧性を持つ場合でもO(n^3)で解析可能
   - nは入力長

であることを確認する

計算量の解析

GPEGは

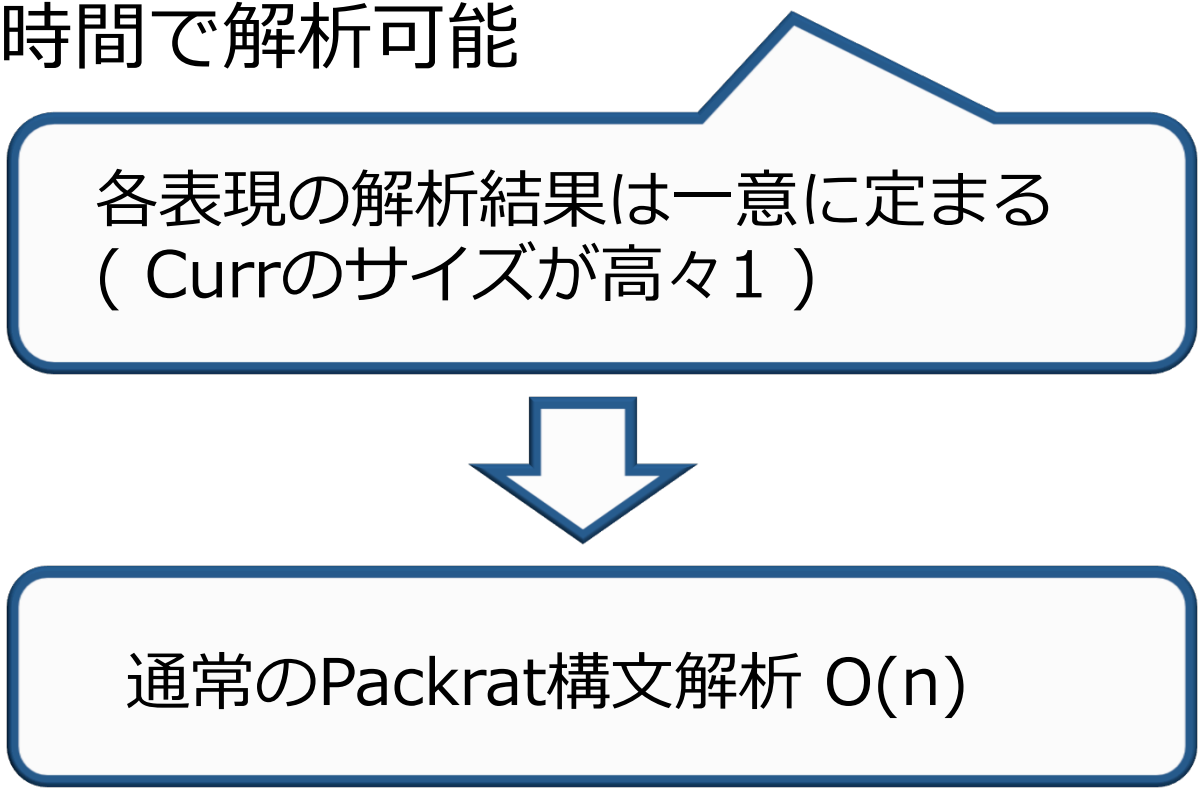
1. GPEGの各表現が曖昧性を含まないのであれば線形時間で解析可能

2. 曖昧性を持つ場合でもO(n^3)で解析可能
   - nは入力長

であることを確認する

1. PEGなら線形時間で解析可能

- GPEGの各表現が曖昧性を含まないのであれば線形時間で解析可能



計算量の解析

GPEGは

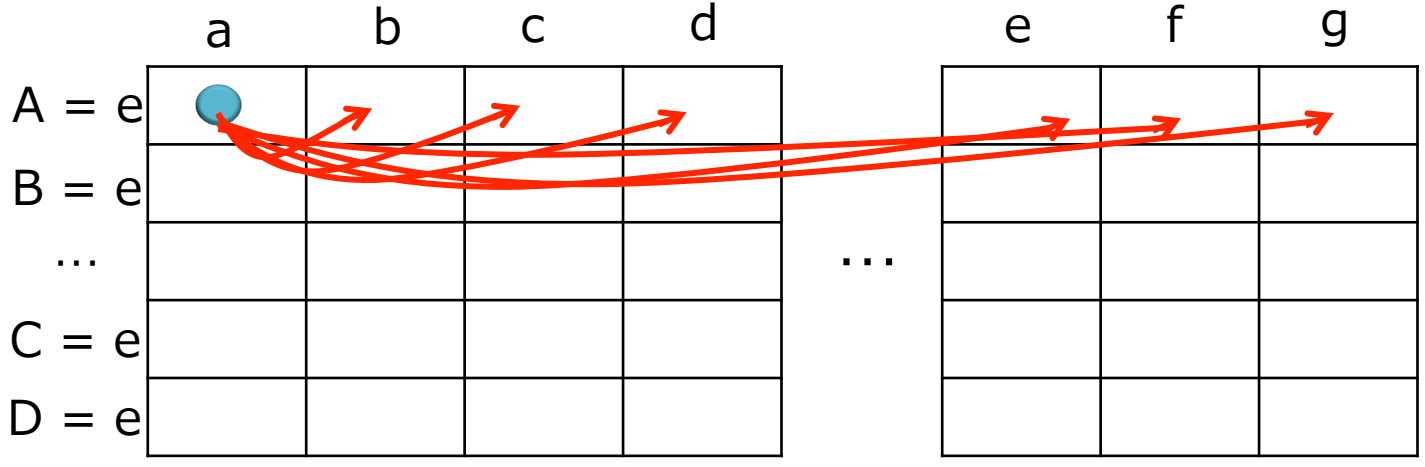
1. GPEGの各表現が曖昧性を含まないのであれば線形時間で解析可能

2. 曖昧性を持つ場合でも$O(n^3)$で解析可能
   - nは入力長

であることを確認する

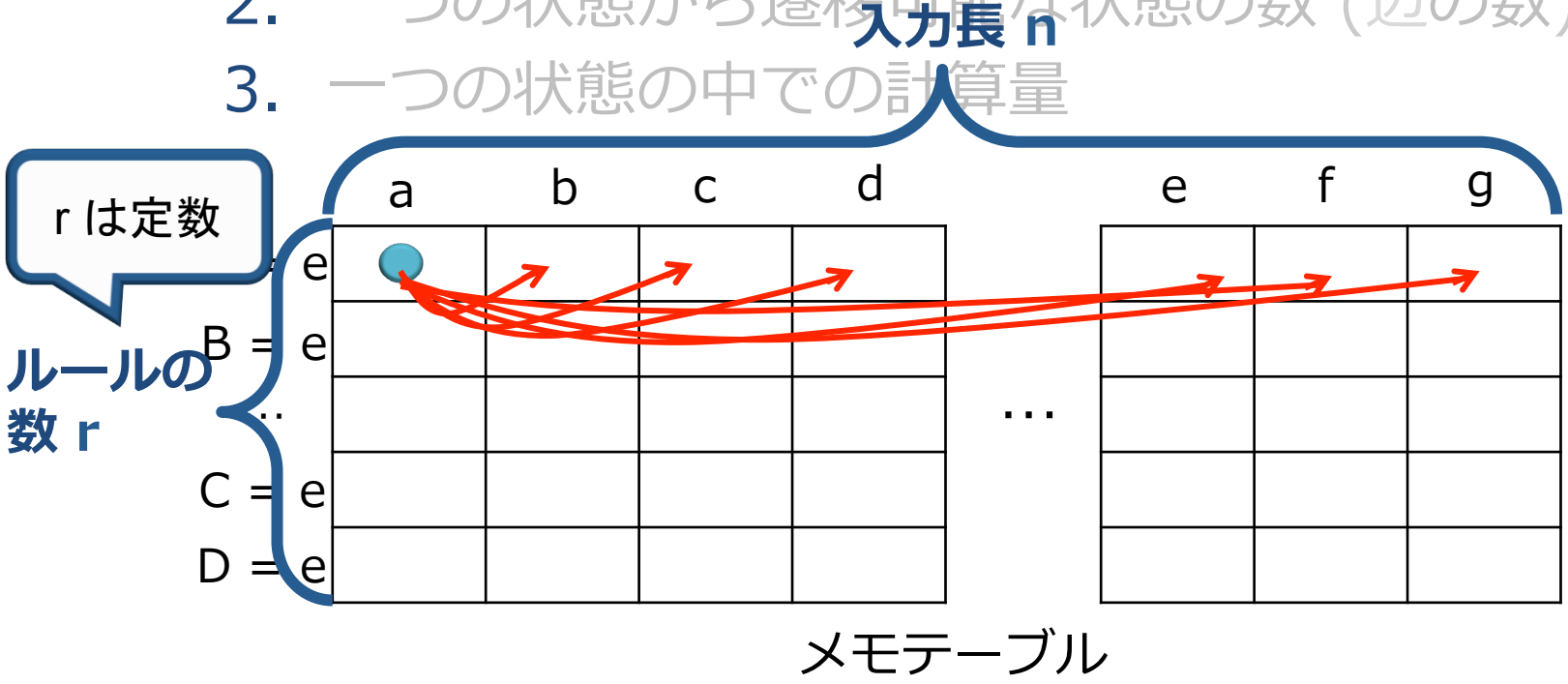
2. 曖昧性を持つ場合でもO(n³)で解析可能

- 曖昧性を持つ場合の計算量は次の要素から計算される
  1. メモテーブルのサイズ　(状態の数)
  2. 一つの状態から遷移可能な状態の数 (辺の数)
  3. 一つの状態の中での計算量



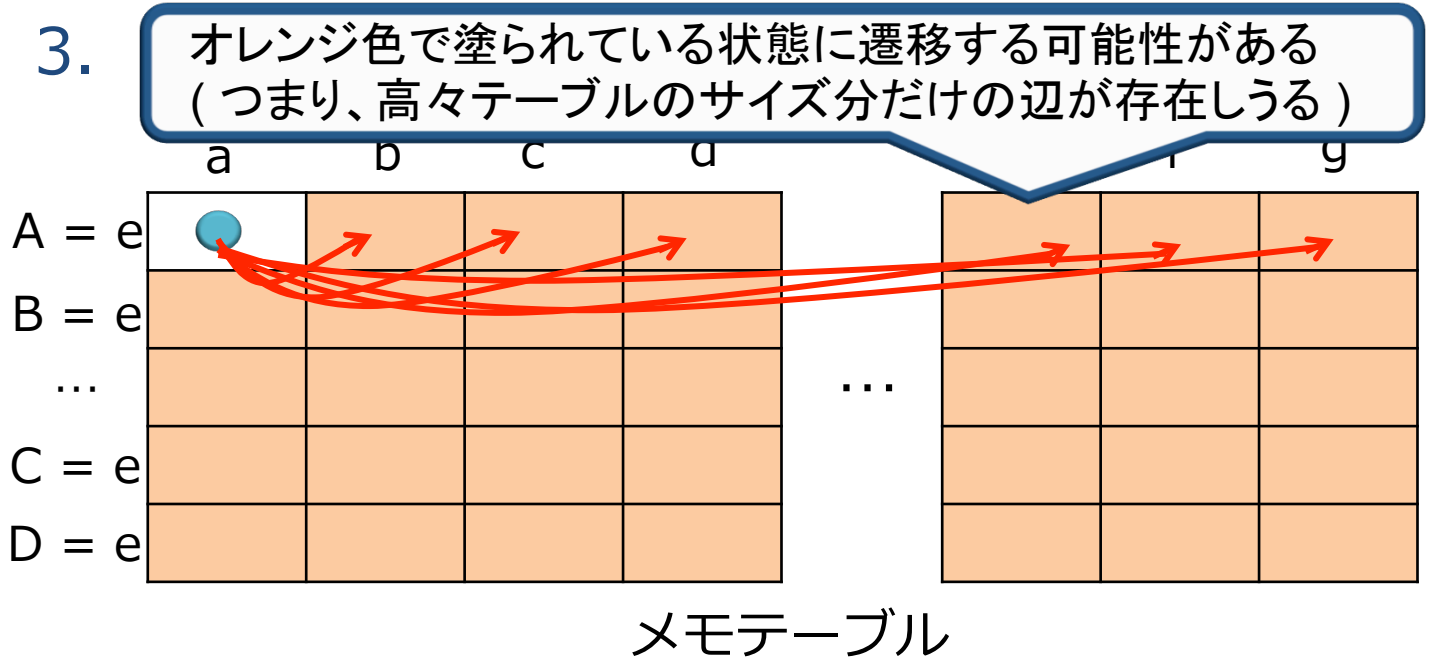
2. 曖昧性を持つ場合でもO(n³)で解析可能

- 曖昧性を持つ場合の計算量は次の要素から計算される
  1. メモテーブルのサイズ　(状態の数) = O(n)
  2. 一つの状態から遷移可能な状態の数 (辺の数)
  3. 一つの状態の中での計算量



2. 曖昧性を持つ場合でもO(n³)で解析可能

- 曖昧性を持つ場合の計算量は次の要素から計算される
  1. メモテーブルのサイズ （状態の数）= O(n)
  2. 一つの状態から遷移可能な状態の数 (辺の数) = O(n)
  3.



2. 曖昧性を持つ場合でもO(n³)で解析可能

- 曖昧性を持つ場合の計算量は次の要素から計算される
  1. メモテーブルのサイズ （状態の数）= O(n)
  2. 一つの状態から遷移可能な状態の数 (辺の数) = O(n)
  3. 一つの状態の中での計算量
     - 各演算子の振る舞いを示した擬似コードのなかで計算量が最もかかるものを探す
       - **優先度付き選択**（ = O(n) ）
       - 優先度無し選択（ = O(n) ）
       - 貪欲な繰り返し（ = O(n) ）

2. 曖昧性を持つ場合でもO(n³)で解析可能

- 優先度付き選択
  - O(n)

```
Next = ∅
Temp = ∅
for j = 1 to n
  Temp = ∅
  foreach i ∈ Curr
    foreach k ∈ parse_ej(i)
      if k == fail
        Temp = Temp ∪ {i}
      else
        Next = Next ∪ {k}
  Curr = Temp
if Curr == ∅
  Next = {fail}
Curr = Next
```

Fig. 7 code($e_1/.../e_n$) : A code for an ordered choice $e_1/.../e_n$

2. 曖昧性を持つ場合でもO(n³)で解析可能

- 優先度付き選択
  - O(n)

このループは選択の部分表現の数だけしか回らないため高々定数

次の状態への遷移なので2. で既に計算量に含めた

関数の戻り値である集合のサイズはO(n)

```
N
Temp = ∅
for j = 1 to n
  Temp = ∅
  foreach i ∈ Curr
    foreach k ∈ parse_ej(i)
      if k == fail
        Temp = Tem
      el
  Curr =
if Curr == ∅
  Next = {fail}
Curr = Next
```

Fig. 7 code(e_1/.../e_n) : A code for an ordered choice e_1/.../e_n

2. 曖昧性を持つ場合でもO(n^3)で解析可能

- 曖昧性を持つ場合の計算量は次の要素から計算される
  1. メモテーブルのサイズ　(状態の数) = O(n)
  2. 一つの状態から遷移可能な状態の数 (辺の数) = O(n)
  3. 一つの状態の中での計算量 = O(n)

- これらをまとめると、O(n^3)

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- 関連研究
- まとめ

関連研究-1

Boolean Grammars [A. Okhotin, 2003]

- CFGを拡張した文法
  - 積と否定の状態を表現できるように拡張
  - $A = e_1 \ \& \cdots \& \ e_m \ \& \ \neg e'_1 \ \& \cdots \& \ \neg e'_n$
  - 時間計算量 $O(n^3)$
    - nは入力長

- 言語クラスとしてみた場合、GPEGはBoolean Grammarと等価もしくは含まれるかも？
  - GPEGの先読み演算子を Boolean Grammarの & と対応付けることができる？

関連研究-2

Scannerless Boolean Parsing [A. Megacz, 2006]

- Boolean GrammarをベースとしたスキャナーレスなパーサであるSBP: a Scannerless Boolean Parserを紹介
  - 優先度付き選択 e_1 / e_2 が使える
    - e_1/ e_2 は内部的には e_1 | (e_2 & ¬ e_1) として扱われる
- 解析アルゴリズムとしてGLR法を採用
- 時間計算量 $O(n^3)$

関連研究-3

GLL Parsing [E. Scott and A. Johnstone,2010]

- 再帰下降構文解析-likeにCFGを解析するアルゴリズム
- 幅優先探索で解析を行い、解析中の状態は
  文法中の位置を表すラベルと入力上の位置を表す値を
  キーとしてグラフ構造でメモ化される
- 時間計算量 $O(n^3)$

Outline

- PEGの定義
- Generalized PEG(GPEG)の定義
- GPEGの解析アルゴリズム
- 計算量の解析
- まとめ

まとめ

PEGに曖昧性を追加した形式文法GPEGを定義

- PEGの演算子に加え、優先度無し選択を持つ
- PEGとCFGの両方を含む

GPEGの解析アルゴリズムを紹介

- 各表現に曖昧性がない場合
  - O(n)で解析可能
- 曖昧性がある場合
  - $O(n^3)$で解析可能
    nは入力長

実装：https://github.com/NariyoshiChida/GPEG