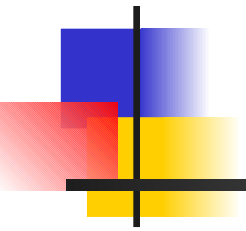


アルゴリズムとデータ構造III
6回目：11月15日

構文解析　チャート法
グラフ　ダイクストラ法

授業資料　http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html

ハードウェア実験II受講者へ

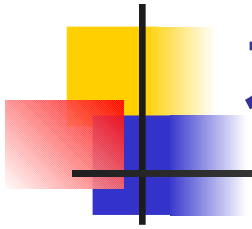
- 12月06日（木）　会社見学
  - 見学場所：ファナック株式会社（忍野村）
  - 12:30 大学発（観光バス）
  - 14:00～16:00 会社見学
  - 17:30 大学着（の予定）
  - ファナック
    - FAとロボット
    - http://www.fanuc.co.jp/

授業の予定（中間試験まで）

10/11	スタック（後置記法で書かれた式の計算）
10/18	文脈自由文法
10/25	構文解析　CKY法
11/01	構文解析　CKY法，チャート法
11/08	構文解析　CKY法，チャート法
11/15	構文解析（チャート法），グラフ（ダイクストラ法，動的計画法，DPマッチング）
11/29	グラフ（ビームサーチ，A*アルゴリズム）グラフ（トライ構造，トライサーチ）
12/06	中間試験

授業の予定(中間試験以降)

1. 全文検索アルゴリズム(simple search, KMP, BM)
2. 全文検索アルゴリズム(Aho-Corasick)
3. テキスト圧縮　暗号　(例:モールス信号，黄金虫，踊る人形，ハフマン符号，Zipfの法則)
4. テキスト圧縮　zip
5. 音声圧縮　ADPCM，MP3
6. 音声圧縮(CELP)，画像圧縮(JPEG)
7. 期末試験

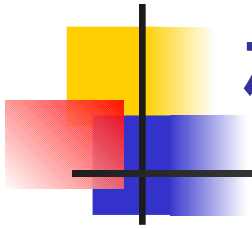


本日のメニュー

- 構文解析
- チャート法
  - 解析例
  - アルゴリズム

構文解析とは(Wikipediaより)

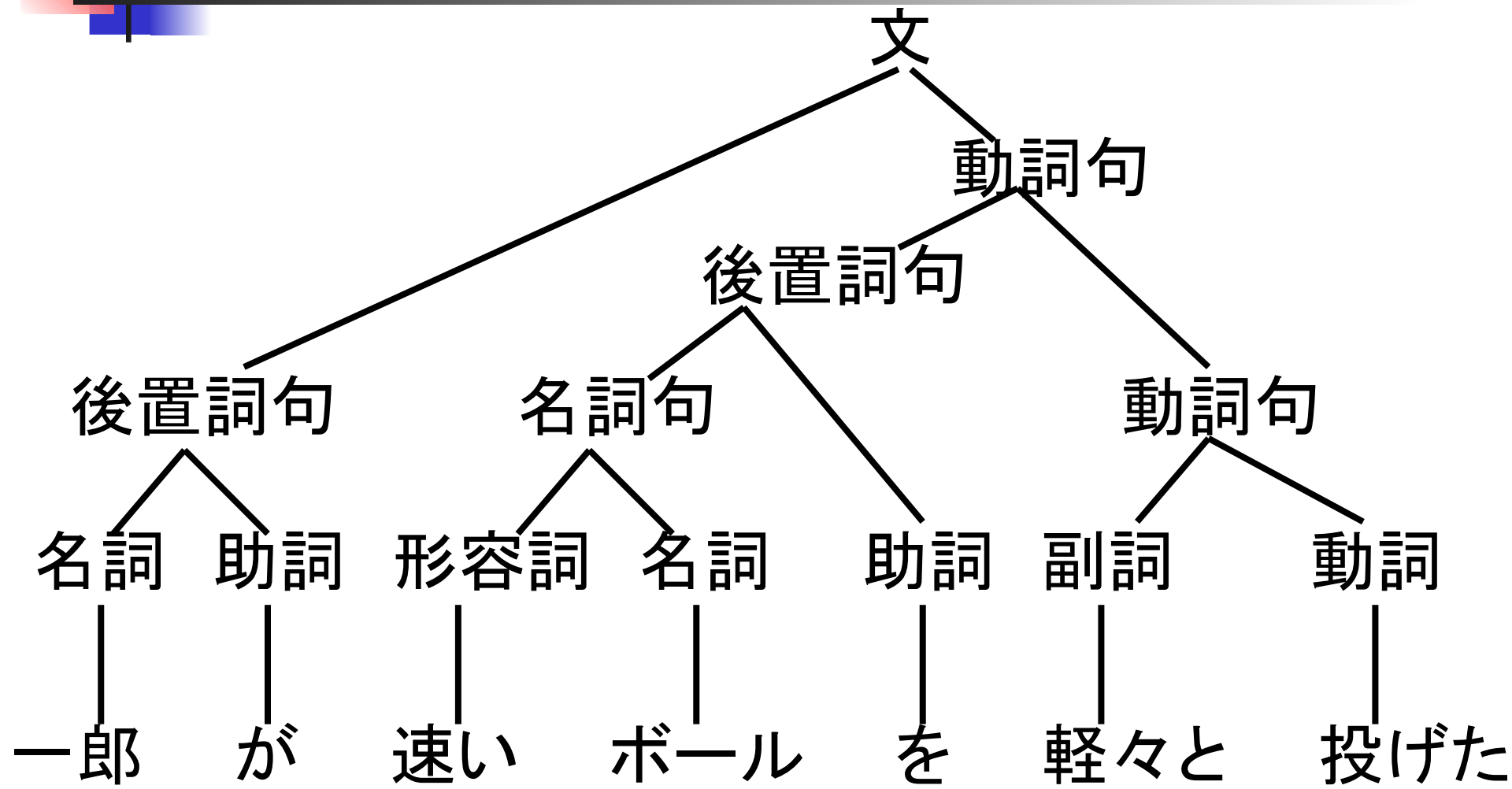
- ある文章の文法的な関係を説明すること(*parse*)。計算機科学の世界では、構文解析は字句解析(*Lexical Analysis*)とともに、おもにプログラミング言語などの形式言語の解析に使用される。また、自然言語処理に応用されることもある。
- コンパイラにおいて構文解析を行う機構を**構文解析器**(Parser)と呼ぶ。
- 構文解析は入力テキストを通常、木構造のデータ構造に変換し、その後の処理に適した形にする。字句解析によって入力文字列から字句を取り出し、それらを構文解析器の入力として、構文木や抽象構文木のようなデータ構造を生成する。

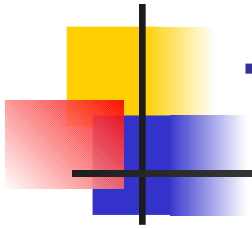


構文解析

- 入力文（記号列）が与えられたとき，文法によってその文を解析し，その構造を明らかにする
- 代表的な構文解析アルゴリズム
  - CKY法
  - チャート法
  - アーリー法
  - LR法

構文木
（一郎が速いボールを軽々と投げた）





チャート法（構文解析）

- トップダウンチャート法
  - Sから出発
  - 目的の単語列を導出　→　解析終了
- ボトムアップチャート法
  - 単語列から出発
  - Sを導出　→　解析終了

チャート法

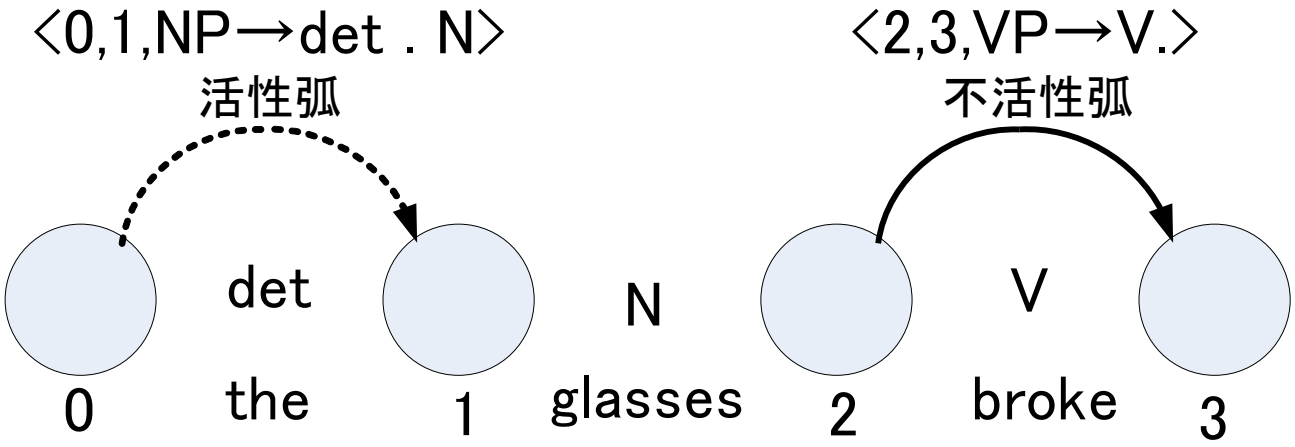
- 節点（ノード）
  - 単語と単語の間に存在する仮想的な点
- 弧（アーク）
  - 節点間を結び，分の部分的な構造を表す
  - <i,j,C → α.β>
    - iは弧の始点，jは弧の終点
    - .は解析が終了している位置
    - 節点iからjまで解析するとα
    - βまで解析できるとC

チャート法

- 不活性弧
  - ・が右辺の最後にある弧
- 活性弧
  - 不活性弧以外の弧
- チャート
  - ノード，弧の集合
- アジェンダ
  - チャートに追加するべき弧のリスト

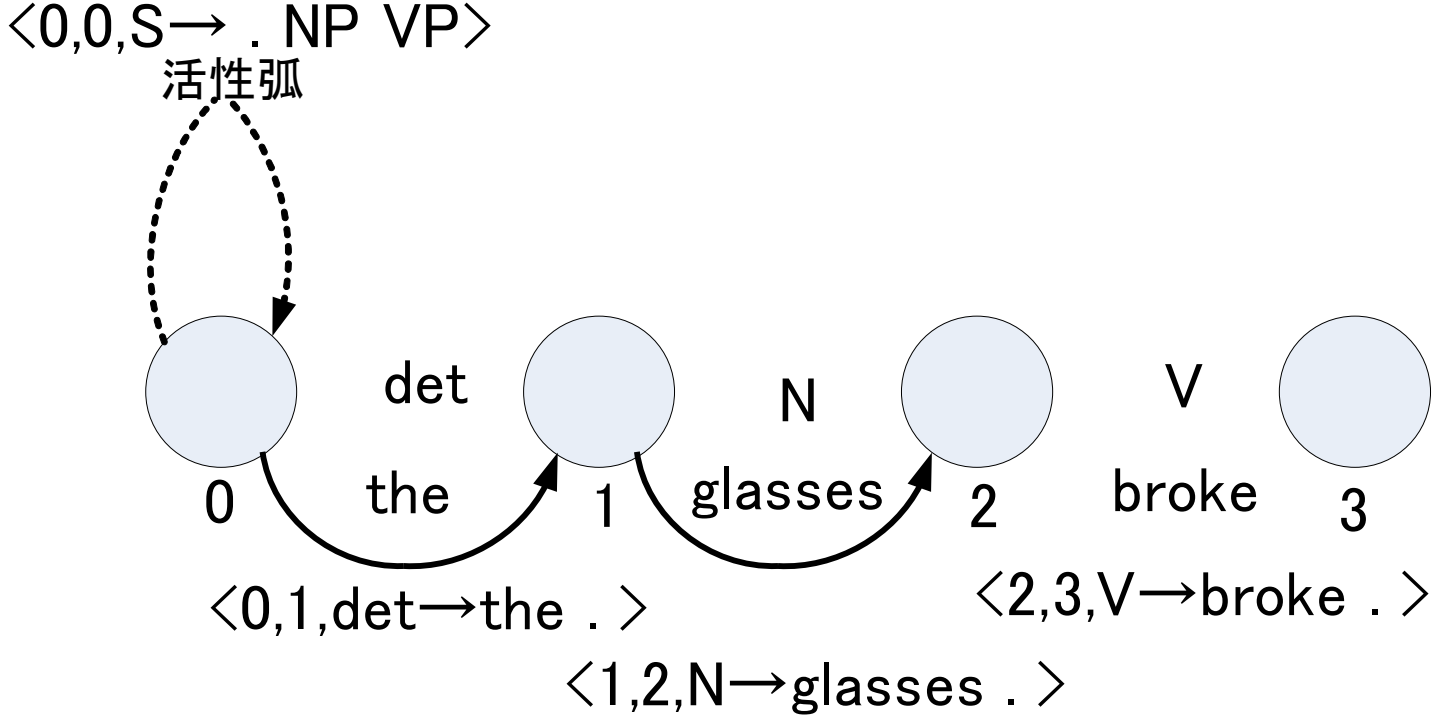
チャート法

弧の例



トップダウンチャート法のアルゴリズム(1/2)

- 辞書規則の適用
  - 入力文の各単語w_kについて，
  - 不活性弧<k,k+1, A→w_k.>をアジェンダに追加
- 活性弧<0,0,S→.α>をアジェンダの先頭に追加

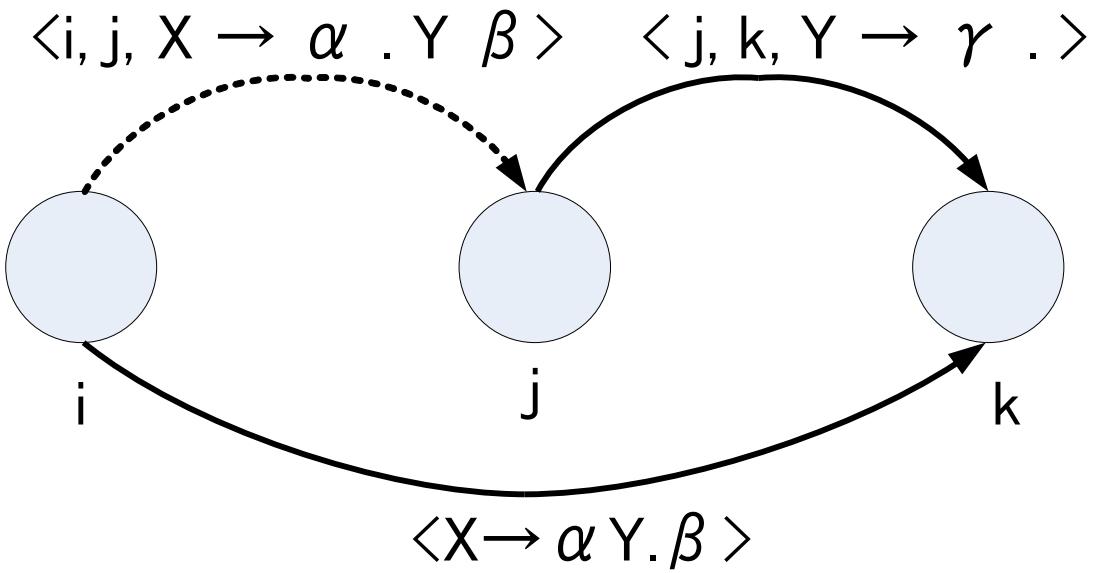


トップダウンチャート法のアルゴリズム(2/2)

- アジェンダが空になるまで以下の操作を繰り返す
  - 弧の選択
    - アジェンダから弧を1個選びチャートに追加（選んだ弧＝arc）
  - 弧の結合
    - arcが活性弧<i,j,X→α.Yβ>のとき，
      - arcの右にある不活性弧<i,k,Y→γ.>を探し，結合する
    - arcが不活性弧<i,j,Y→γ.>のとき，
      - arcの左にある活性弧<k,i,X→α.Yβ>を探し，結合する
    - 結合してできた新しい弧<i,k,X→αY.β>をアジェンダに追加
  - 新しい弧の提案
    - arcが活性弧<i,j,X→α.Yβ>のとき，
      - Yを左辺とする規則Y→γ(辞書規則を除く）があれば，新しい活性弧<j,j,Y→.γ>を作ってアジェンダに追加

トップダウンチャート法のアルゴリズム

- 弧の結合を行う
  - 例えば
  - <i, j, X →α. Y β> + <j, k, Y→γ.>
  - →　<i, k, X → αY. β>



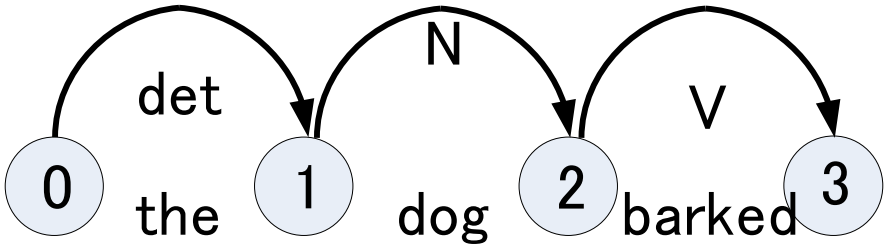
- 不活性弧<0,n,S→α.>が生成できれば解析成功

トップダウンチャート法

- 解析文
- The dog barked.
- 文法
  - S→NP VP
  - NP→det n
  - VP→v
  - VP→v NP
  - det → the
  - n → dog
  - v → barked

The dog barked.

チャート



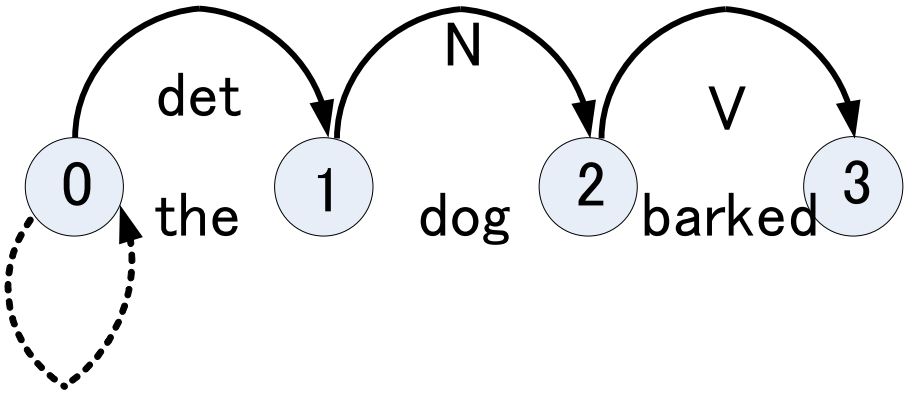
辞書規則をアジェンダにpush

アジェンダ

〈0,1,det→the . 〉
〈1,2,N→dog . 〉
〈2,3,V→barked . 〉

The dog barked.

チャート



〈0,0,S→ . NP VP〉

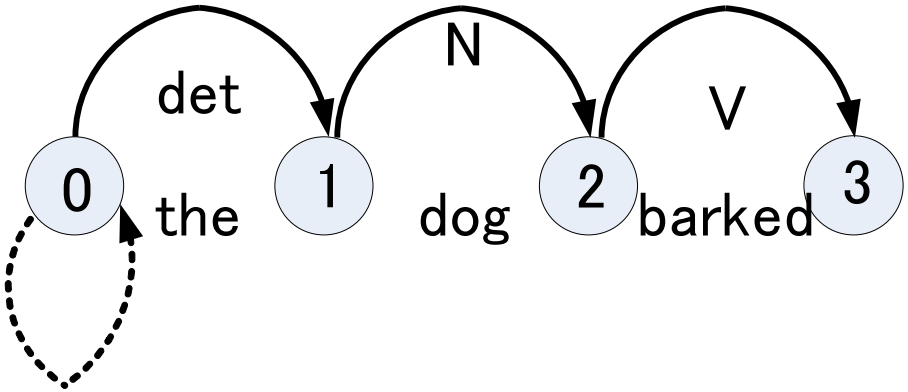
アジェンダ

〈0,0,S→ . NP VP〉
〈0,1,det→the . 〉
〈1,2,N→dog . 〉
〈2,3,V→barked . 〉

〈0,0,S→ . NP VP〉をアジェンダにpush → アジェンダからチャートにpop

The dog barked.

チャート

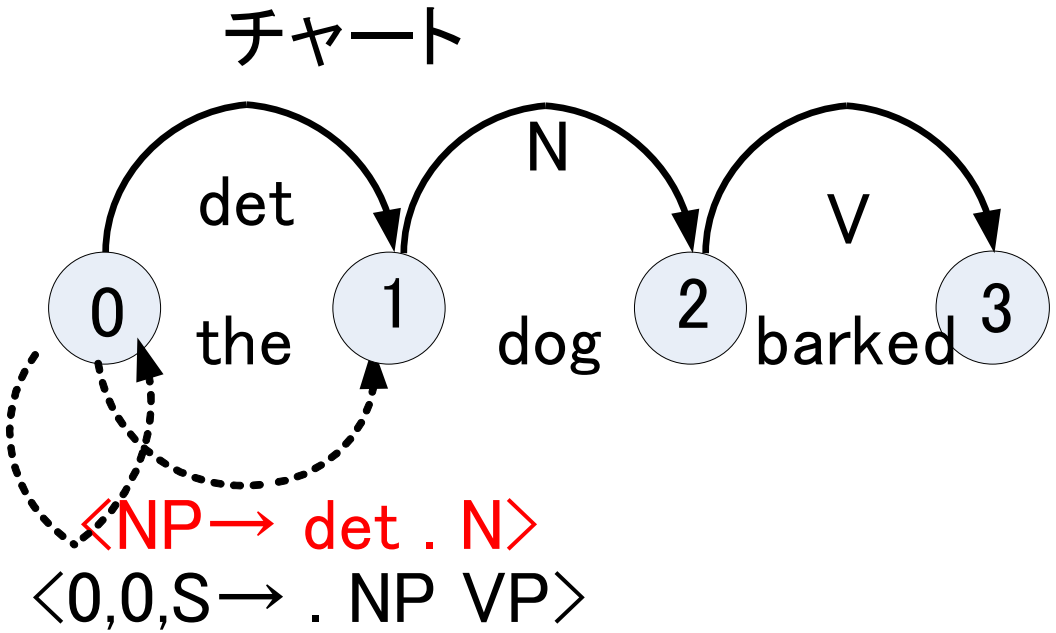


アジェンダ

〈0,0,NP→ . det N〉
〈0,1,det→the . 〉
〈1,2,N→dog . 〉
〈2,3,V→barked . 〉

〈0,0,S→ . NP VP〉
活性弧
〈0,0,NP→ . det N〉
新しい活性弧〈0,0,NP→ . Det N〉をアジェンダにpush → アジェンダからチャートにpop

The dog barked.



アジェンダ

<0,1,NP→ det . N>

<1,2,N→dog . >

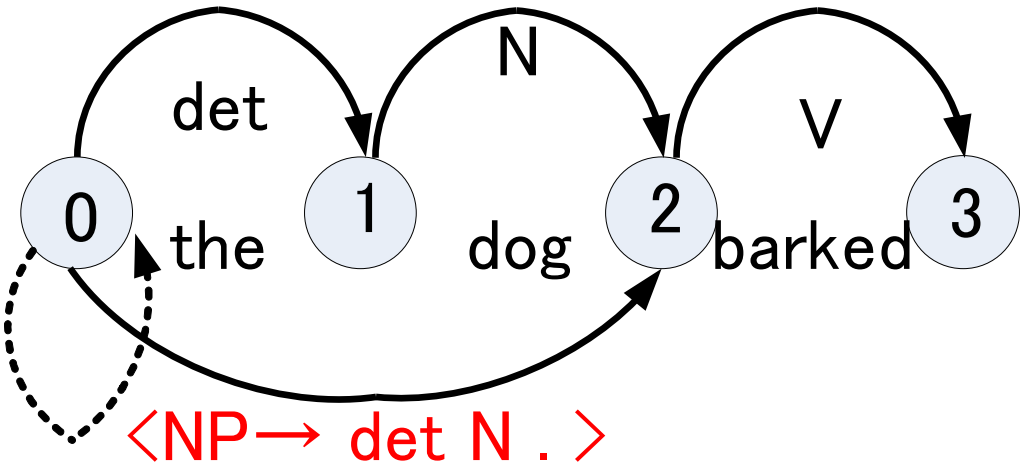
<2,3,V→barked . >

<0,0,NP→ . Det N>と<0,1,det→the .>を結合して<NP→ det . N >を得る.

<NP→ det . N >をアジェンダにpush → アジェンダからチャートにpop

The dog barked.

チャート



アジェンダ

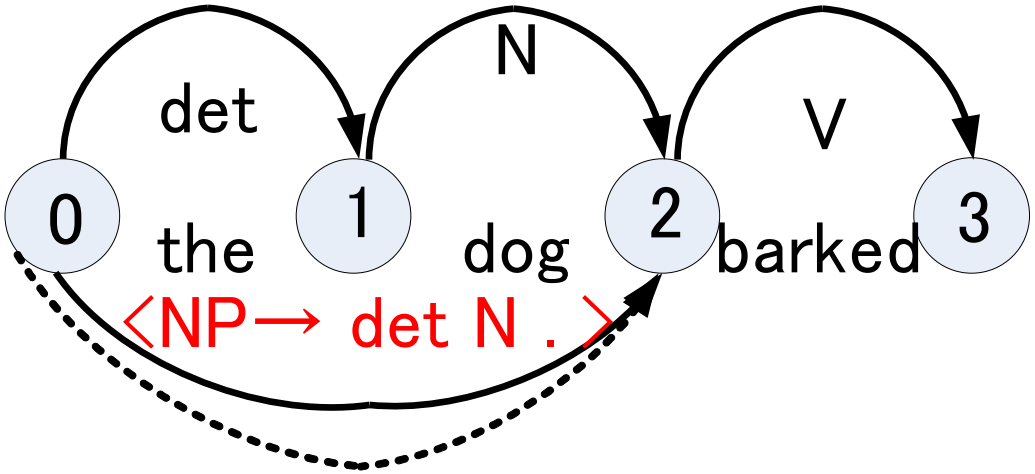
〈0,2,NP→ det N . 〉
〈0,1,det→the . 〉
〈2,3,V→barked . 〉

〈0,0,S→ . NP VP〉

〈NP→ det . N 〉と〈N→ dog . 〉を結合して〈NP→det N . 〉を得る. 〈アジェンダにpush → アジェンダからチャートにpop

The dog barked.

チャート



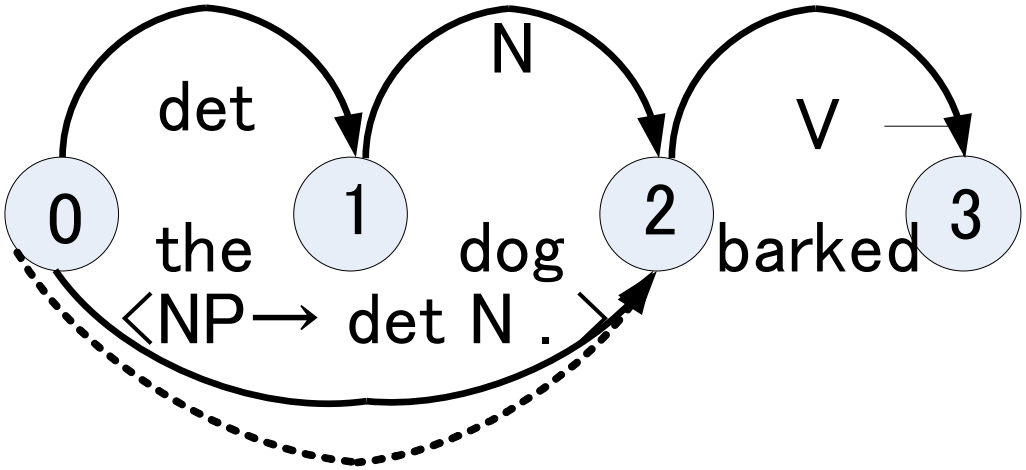
アジェンダ

〈0,2,S→ NP . VP〉
〈0,1,det→the . 〉
〈1,2,N→dog . 〉
〈2,3,V→barked . 〉

〈NP→det N . 〉と〈S→ . NP VP〉を結合して〈S→ NP . VP〉を得る. 〈S→ NP . VP〉をアジェンダにpush → アジェンダからチャートにpop

The dog barked.

チャート



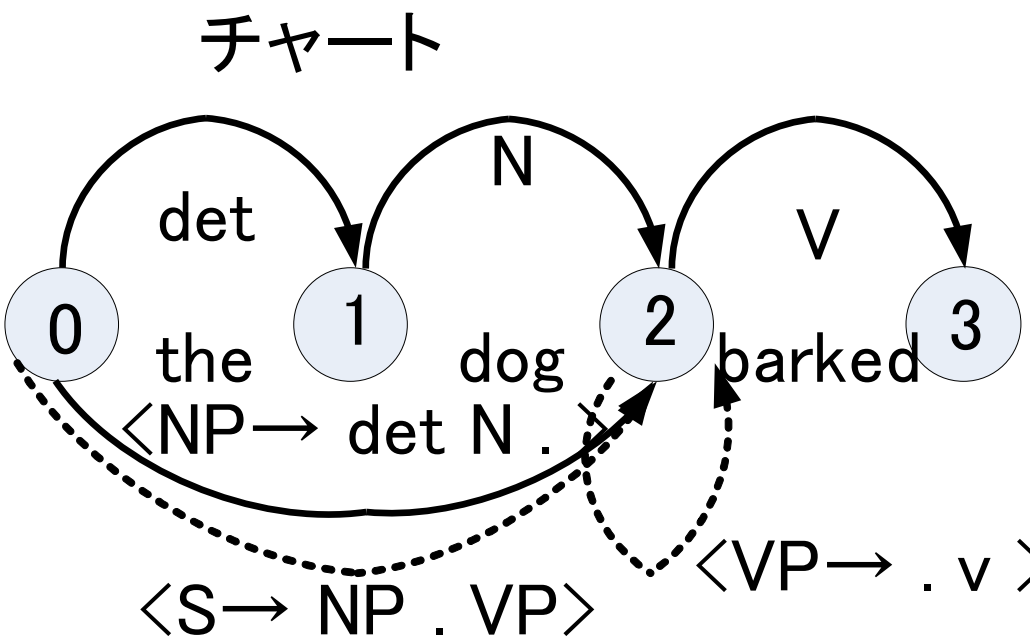
〈S→ NP . VP〉

アジェンダ

〈0,2,S→ NP . VP〉
〈2,3,V→barked . 〉

〈NP→det N . 〉と〈S→ . NP VP〉を結合して〈S→ NP . VP〉を得る. 〈S→ NP . VP〉をアジェンダにpush → アジェンダからチャートにpop

The dog barked.



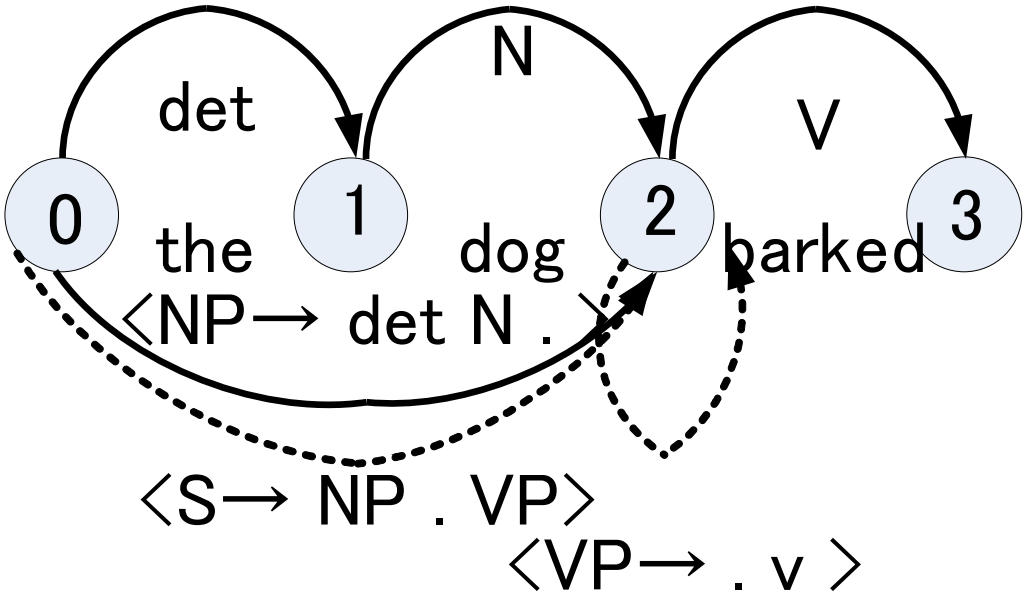
アジェンダ

〈2,2,VP→ . v 〉
〈0,2,S→ NP . VP〉
〈2,3,V→barked . 〉

新しい活性弧〈2,2,VP→ . v〉をアジェンダにpush → アジェンダからチャートにpop

The dog barked.

チャート



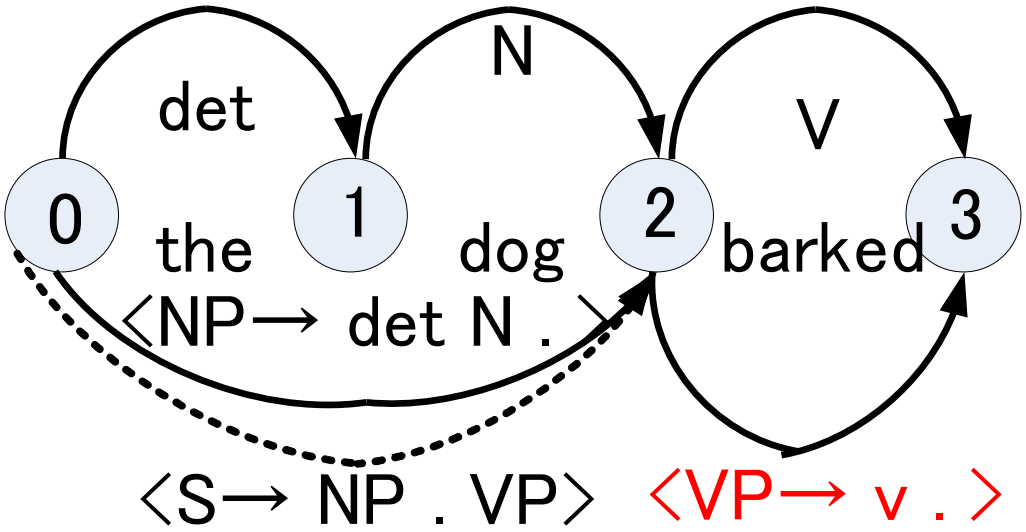
アジェンダ

〈0,2,S→ NP . VP〉
〈2,3,V→barked . 〉

The dog barked.

チャート

アジェンダ



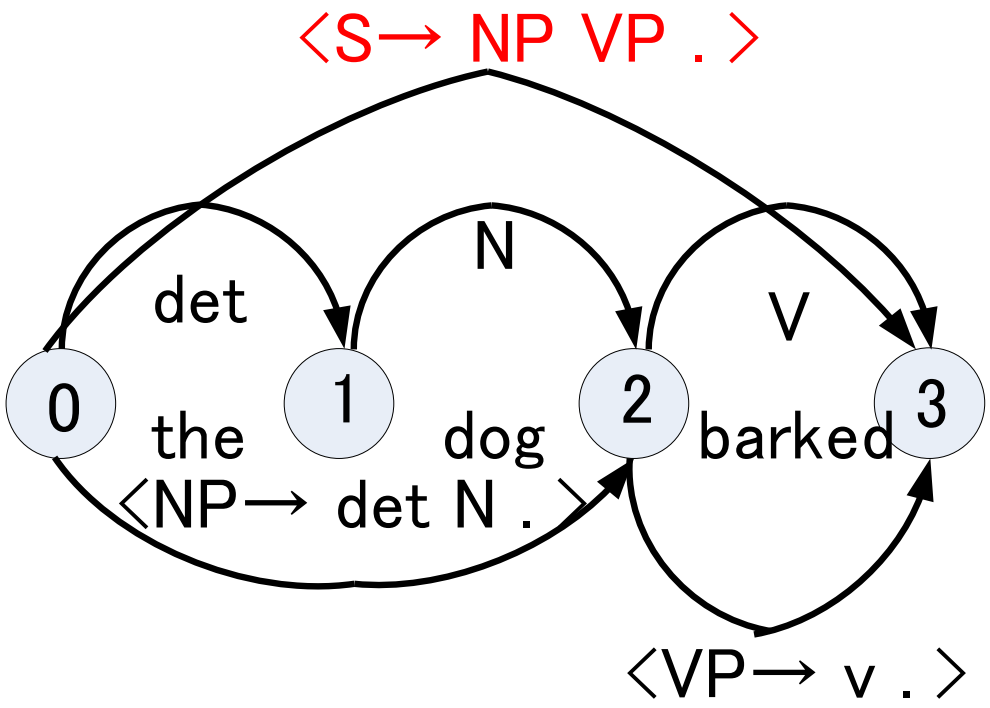
〈2,2,VP→ v . 〉
〈0,2,S→ NP . VP〉
〈2,3,V→barked . 〉

〈VP→ .v 〉と〈v→barked.〉を結合して〈VP→ v.〉を得る

The dog barked.

チャート

アジェンダ

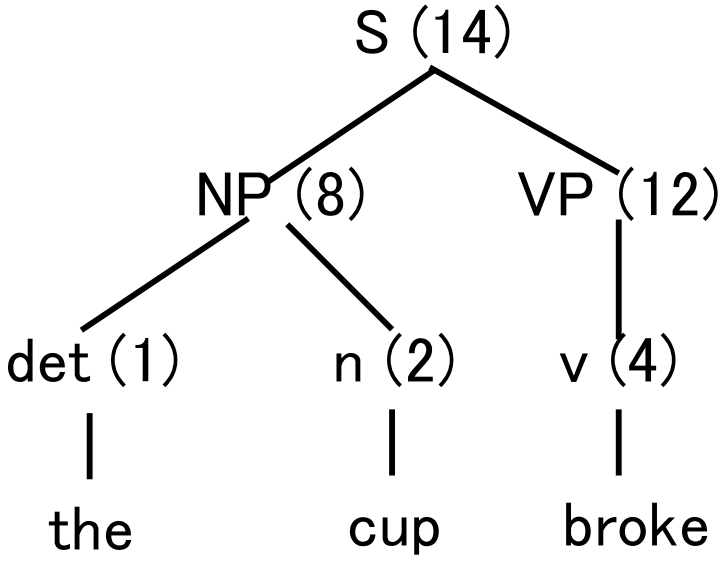


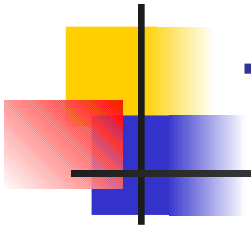
〈0,3,S→ NP VP . 〉

〈S→ NP . VP〉と〈VP→v.〉を結合して〈S→NP VP .〉を得る アジェンダに何もなくなり解析終了

構文木の復元

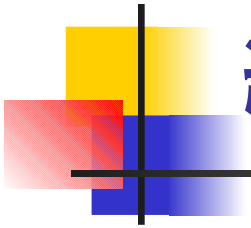
- 弧に履歴を残す.
  - 弧に識別番号をつける
  - 右辺がどの不活性弧によって構成されるかを記録
- 不活性弧の履歴をたどれば構文木が復元できる
- 得られる構文木の例
  - 番号は不活性弧の番号





チャート法の特徴

- 計算量は$O(n^3)$
- 任意の文脈自由文法が扱える
- 4種類の方式
  - トップダウンとボトムアップ
  - 縦型探索と横型探索
- 文法の予測能力が使える
  - 無駄な弧を生成しないので効率が良い
  - トップダウンチャート法のみ
- 広く使われている

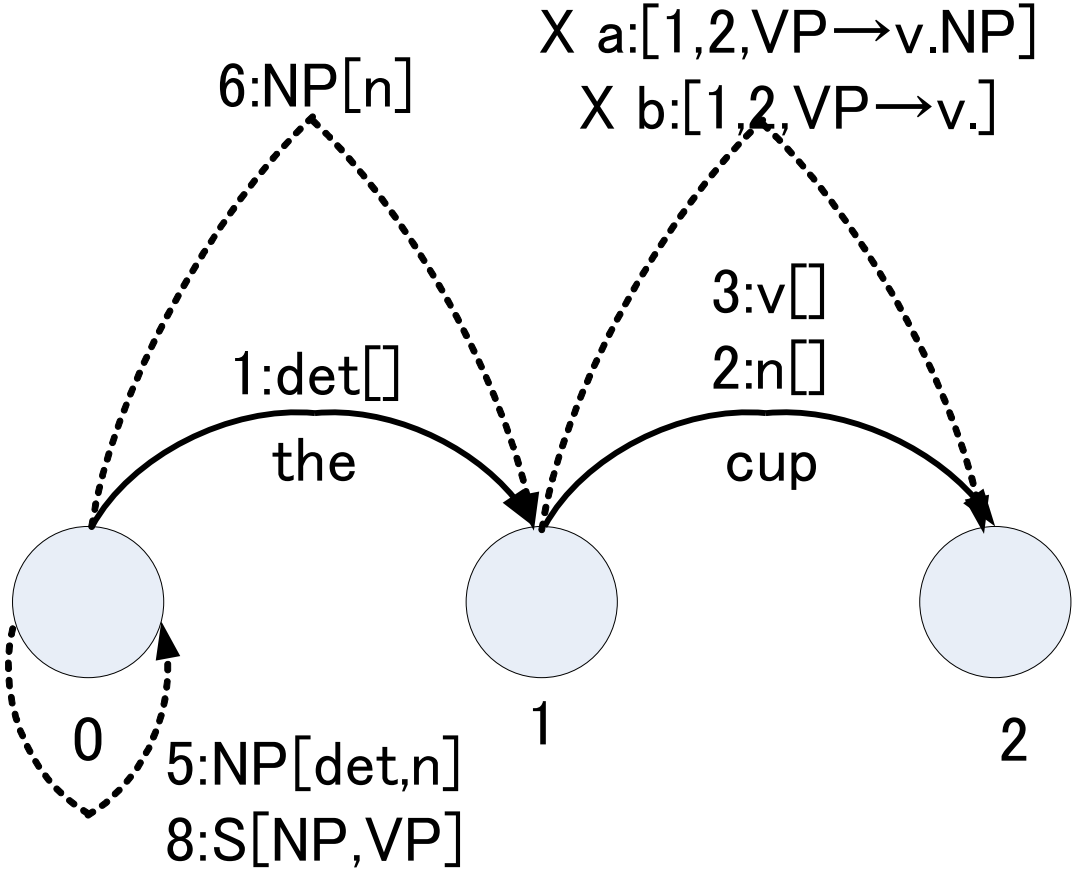


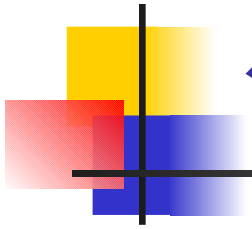
縦型探索と横型探索

- 縦型探索
- 1つの解の候補の解析を優先的に進める
- 文が文法によって生成できるかだけを調べるときに便利
- 横型探索
- 全ての解の候補の解析を並列に進める
- ビームサーチが使える
- チャート法では両方とも可能
- アジェンダをスタック(LIFO)にしたときは縦型探索
- アジェンダをキュー(FIFO)にしたときは横型探索

文法の予測能力

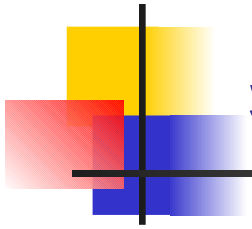
- 無駄な弧は生成されない
- 文法によってdetの後にはvが現れないことが予想されている





グラフ

- ダイクストラ法
- 動的計画法
- DPマッチング



身近な最短経路問題

- 道路の経路探索（カーナビなど）

ダイクストラ法（最短経路問題用アルゴリズム）

- StartノードからGoalノードへ最小コストで移動したい