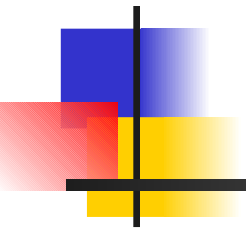


アルゴリズムとデータ構造III
5回目：11月08日

構文解析　CKY法の続き，チャート法

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

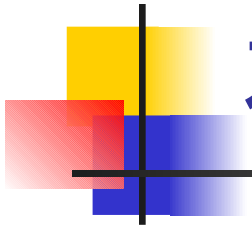
- 12月06日（木）　会社見学
  - 見学場所：ファナック株式会社（忍野村）
  - 12:30 大学発（観光バス）
  - 14:00～16:00 会社見学
  - 17:30 大学着（の予定）
  - ファナック
    - FAとロボット
    - http://www.fanuc.co.jp/

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	グラフ（ダイクストラ法，動的計画法，DPマッチング） グラフ（ビームサーチ，A*アルゴリズム）
11/29	グラフ（トライ構造，トライサーチ）
12/06	中間試験

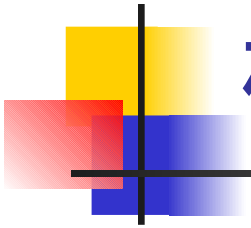
授業の予定(中間試験以降)

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

- CKY法の続き
  - CKYアルゴリズム
  - 解析例（急いで走る一郎を見た）
  - 練習問題（I eat pizza with Nana.）
- (チャート法)

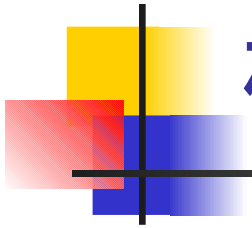


構文解析　CKY法

- 先週勉強した文脈自由文法により，文から自動的に構文木を生成する.

構文解析とは(Wikipediaより)

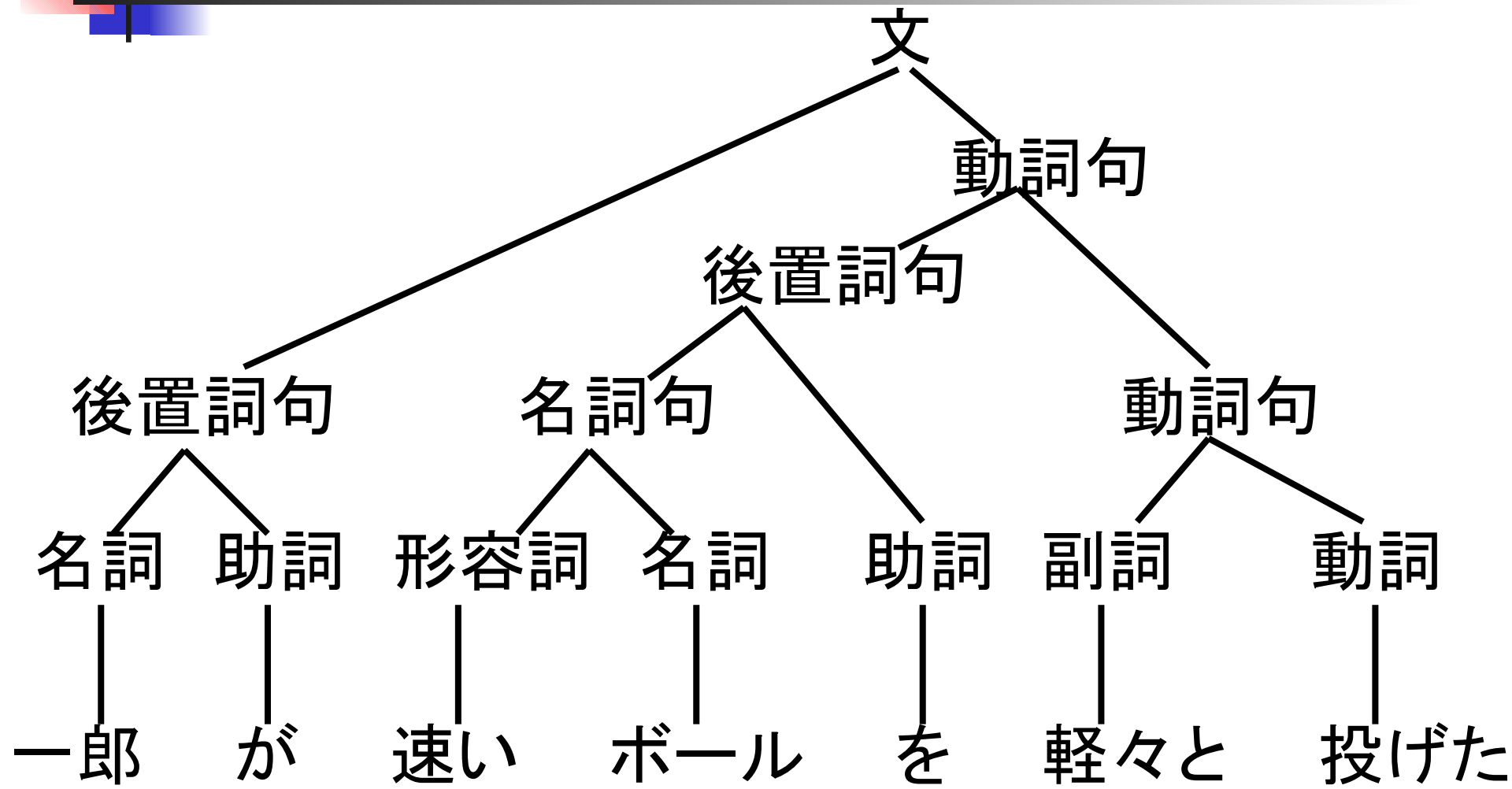
- ある文章の文法的な関係を説明すること(*parse*)。計算機科学の世界では、構文解析は字句解析(*Lexical Analysis*)とともに、おもにプログラミング言語などの形式言語の解析に使用される。また、自然言語処理に応用されることもある。
- コンパイラにおいて構文解析を行う機構を**構文解析器**(Parser)と呼ぶ。
- 構文解析は入力テキストを通常、木構造のデータ構造に変換し、その後の処理に適した形にする。字句解析によって入力文字列から字句を取り出し、それらを構文解析器の入力として、構文木や抽象構文木のようなデータ構造を生成する。



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）



CKY（Cocke-Kasami-Younger）法

- ボトムアップアルゴリズム
- 扱える文法
  - チョムスキーの標準形
    - A→BC
    - A→a
- CKY表
  - 構文解析の途中経過を保持するための表

CKYアルゴリズム

- チョムスキーの標準形の文脈自由文法を対象とした構文解析法
- チョムスキーの標準形
  - A→BC （A,B,C∈Vn)
  - A→a (A∈Vn, a∈Vt）

X→aBはチョムスキーの標準形ではないが
　X→AB, A→aに分解できる
X→ABCはチョムスキーの標準形ではないが
　X→AY, Y→BCに分解できる

チョムスキーの標準形の例「急いで走る一郎を見る」

A→BC型

- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p

A→a型

- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

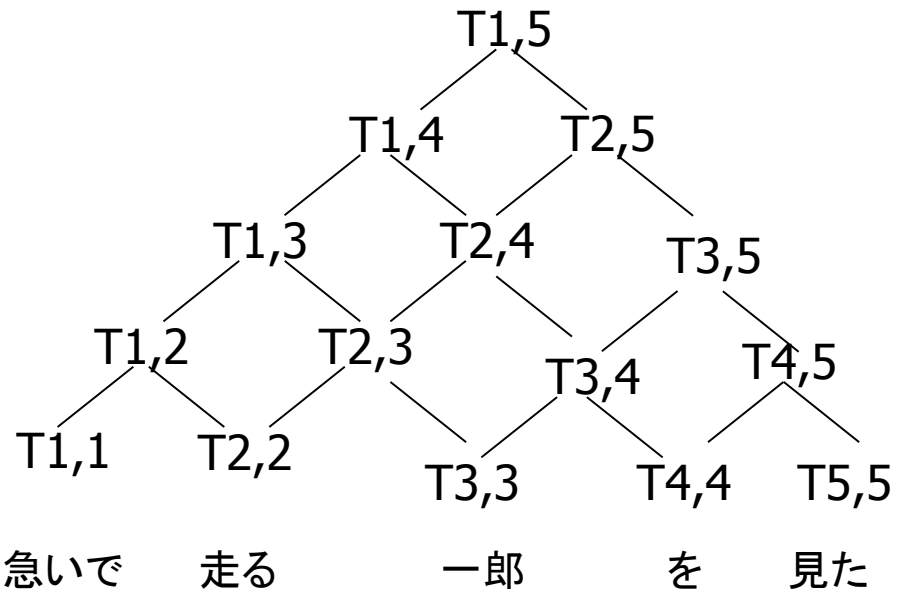
CKY構文解析の概要

T2,5: 走る一郎を見た

T2,2: 走る| T3,5: 一郎を見た
T2,3: 走る一郎| T4,5 を見た
T2,4: 走る一郎を| T5,5 見た

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

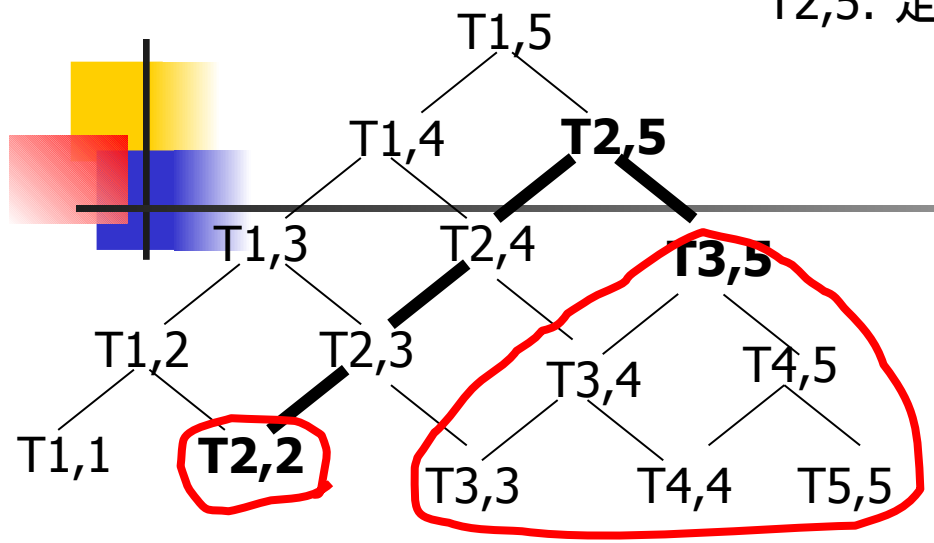
	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで					
2. 走る		T2,2	T2,3	T2,4	T2,5
3. 一郎					T3,5
4. を					T4,5
5. 見た					T5,5



CKY表は構文木を表している

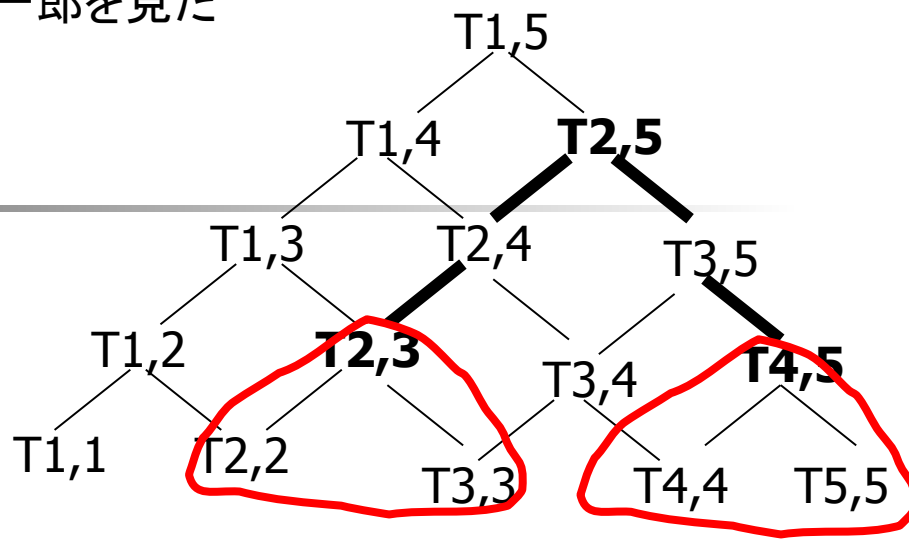
T2,5までの部分木

T2,5: 走る一郎を見た



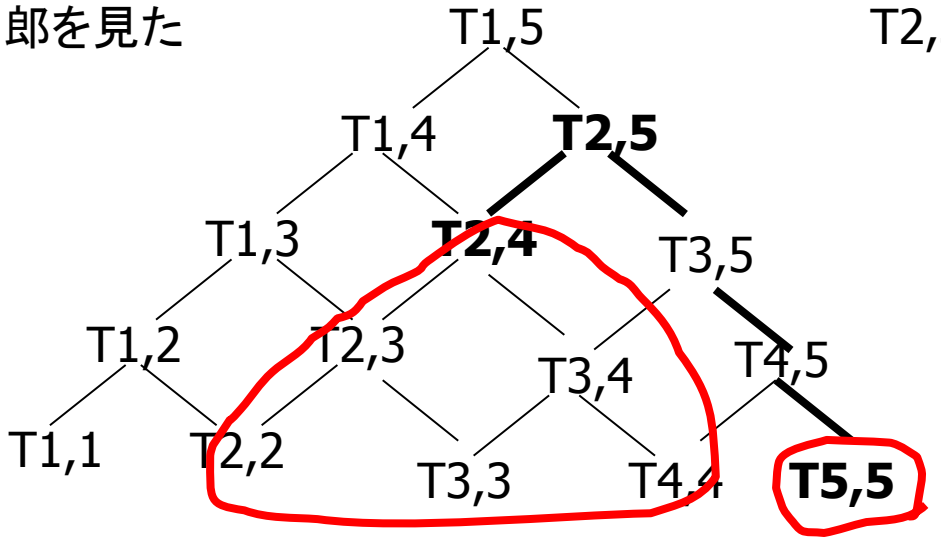
急いで 走る 一郎 を 見た

T2,2: 走る| T3,5: 一郎を見た



急いで 走る 一郎 を 見た

T2,3: 走る一郎| T4,5 を見た



急いで 走る 一郎 を 見た

T2,4: 走る一郎を| T5,5 見た

CKYアルゴリズム

1. $A \rightarrow a$の生成規則を用いて，主対角線上の要素を計算

$for\ i = 1\ to\ N$

$T_{i,j} = \{A \mid A \rightarrow w_i\}$

2. $A \rightarrow BC$の生成規則を用いて,2番目以降の対角線上の要素を計算

$for\ n = 1\ to\ N - 1$

$for\ i = 1\ to\ N - n$

$$T_{i,i+n} = \bigcup_{j=1}^{n} \{A \mid A \rightarrow BC,\ \ B \in T_{i,i+j-1},\ \ C \in T_{i+j,i+n}\}$$

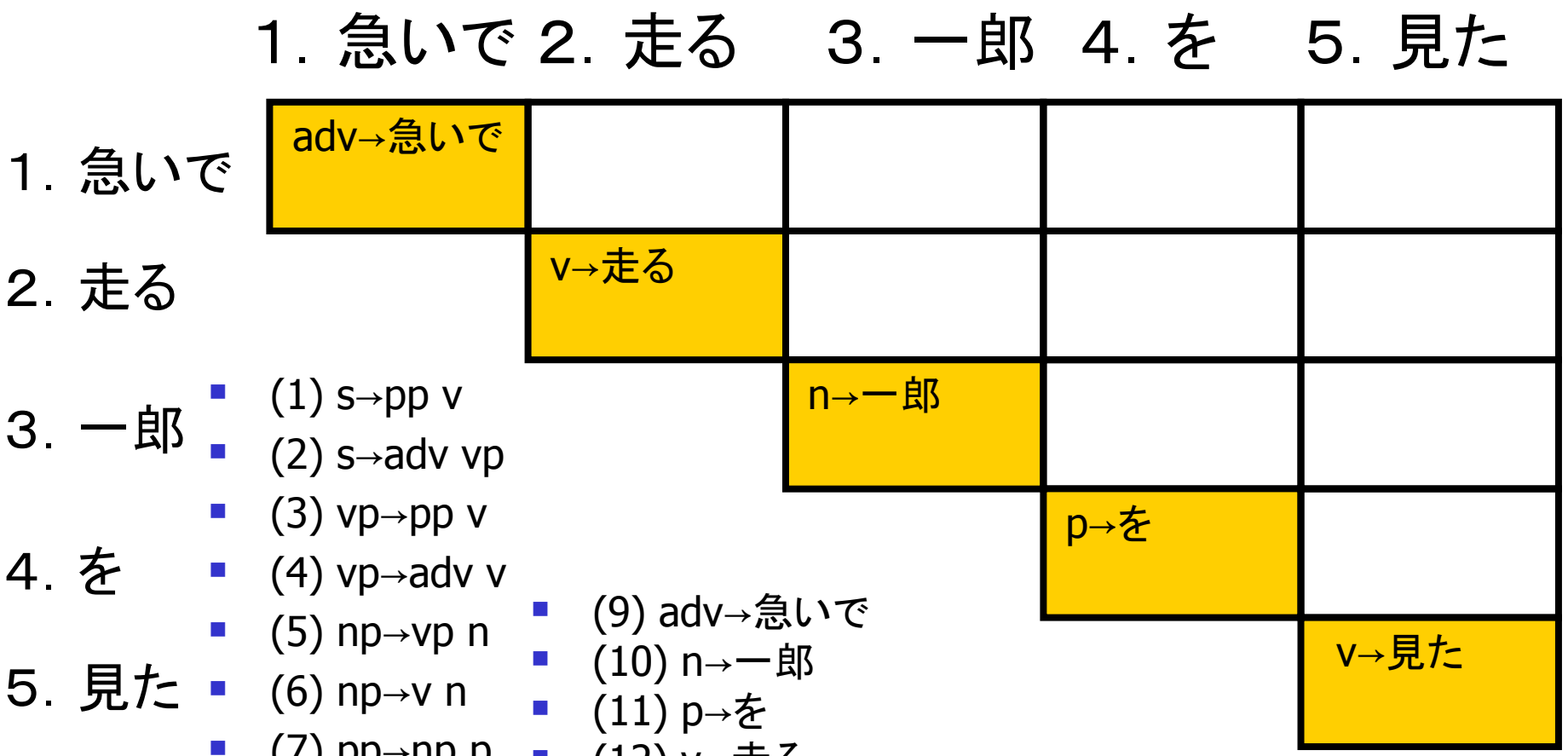
3. $S \in T_{1,N}$であれば，$w_1 \cdots w_N$は開始記号Sから導出可能

CKY構文解析表(完成)

	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

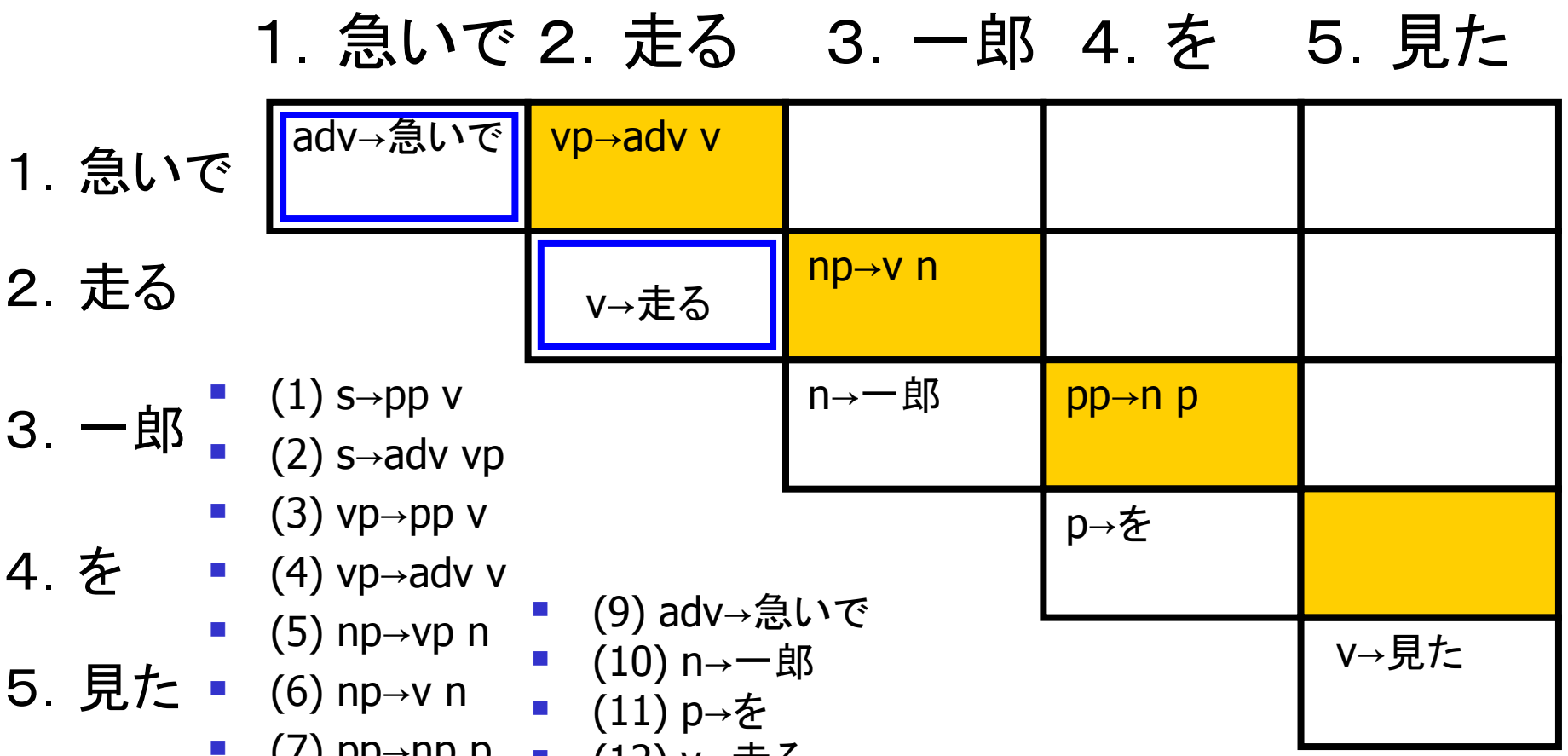
- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

CKY構文解析表(1/5)



- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

CKY構文解析表(2/5)



- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

CKY構文解析表(3/5)

	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n		
2. 走る		v→走る	np→v n	pp→np p	
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

CKY構文解析表(4/5)

	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

CKY構文解析表(5/5)

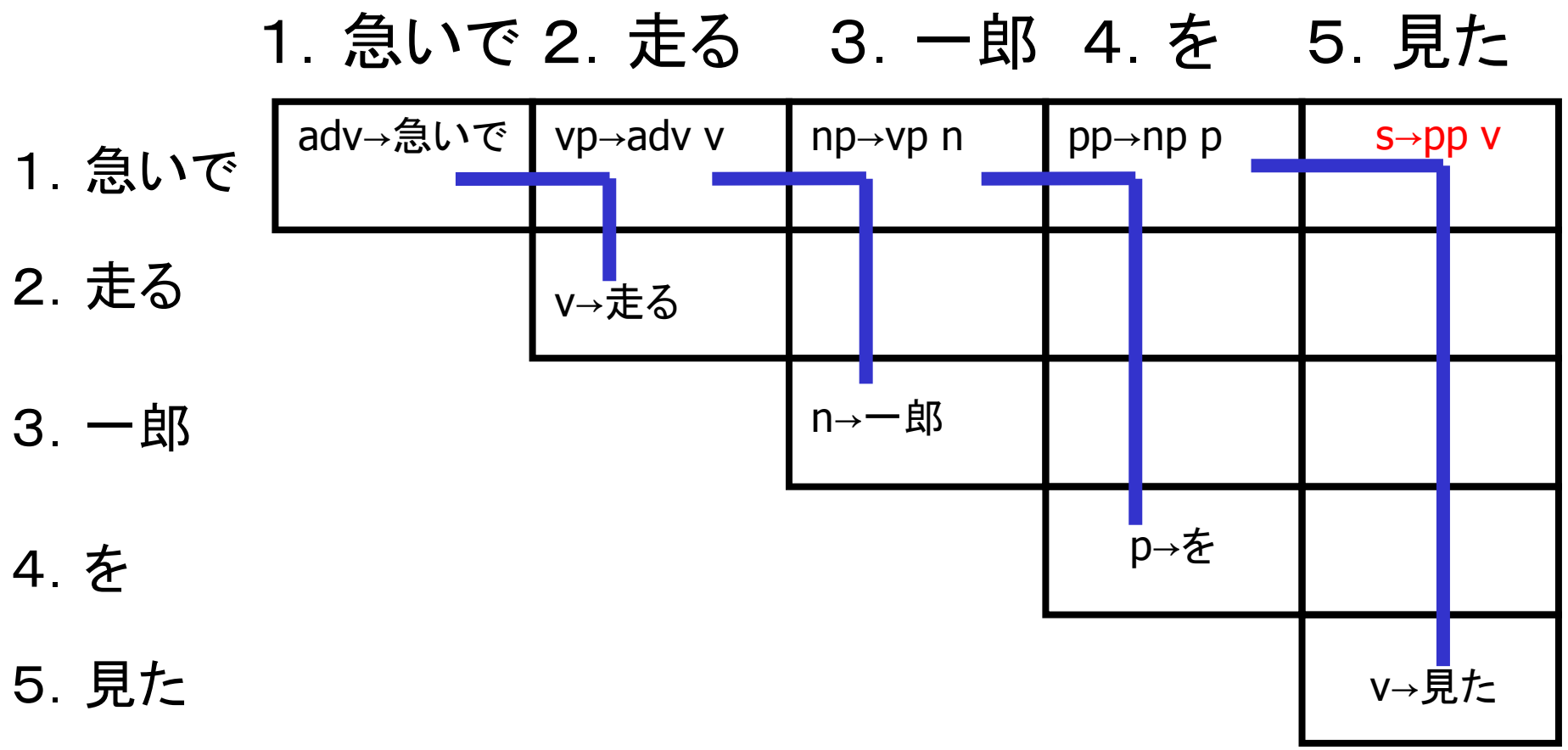
	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

- (1) s→pp v
- (2) s→adv vp
- (3) vp→pp v
- (4) vp→adv v
- (5) np→vp n
- (6) np→v n
- (7) pp→np p
- (8) pp→n p
- (9) adv→急いで
- (10) n→一郎
- (11) p→を
- (12) v→走る
- (13) v→見る

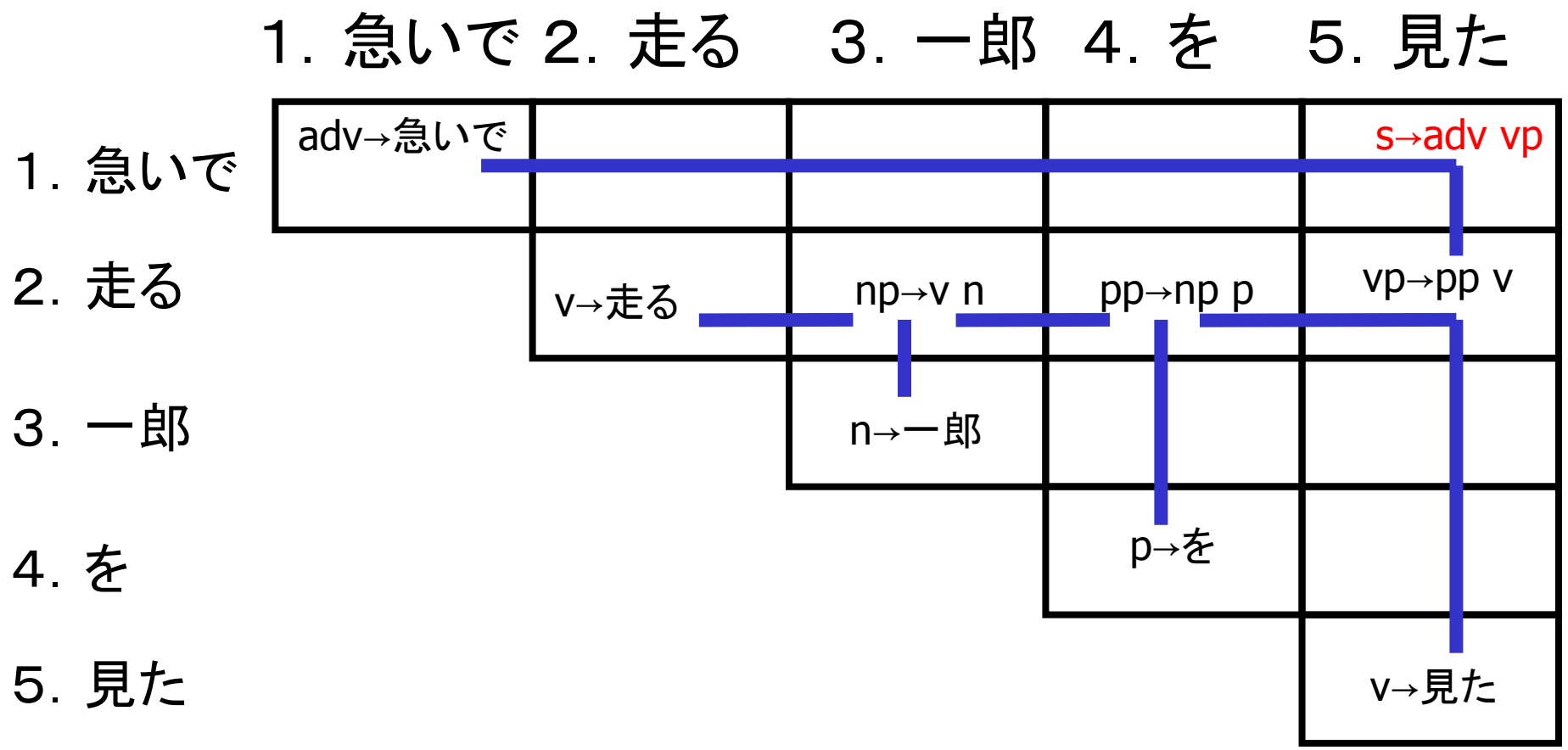
CKY構文解析表(完成！）

	1. 急いで	2. 走る	3. 一郎	4. を	5. 見た
1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

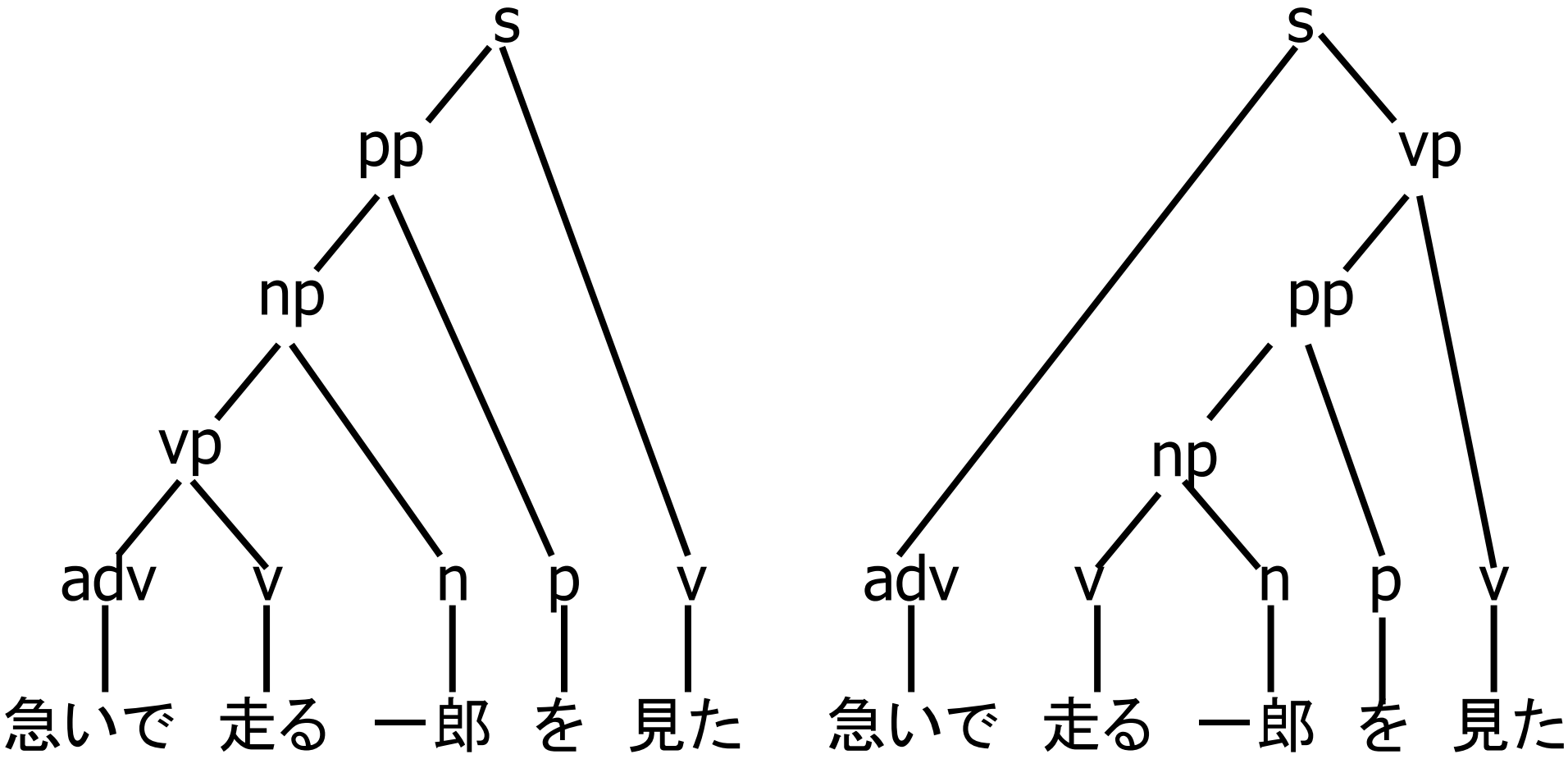
CKY構文解析表 → 構文木（s→pp v の構文木）



CKY構文解析表 → 構文木（s→adv vp の構文木）

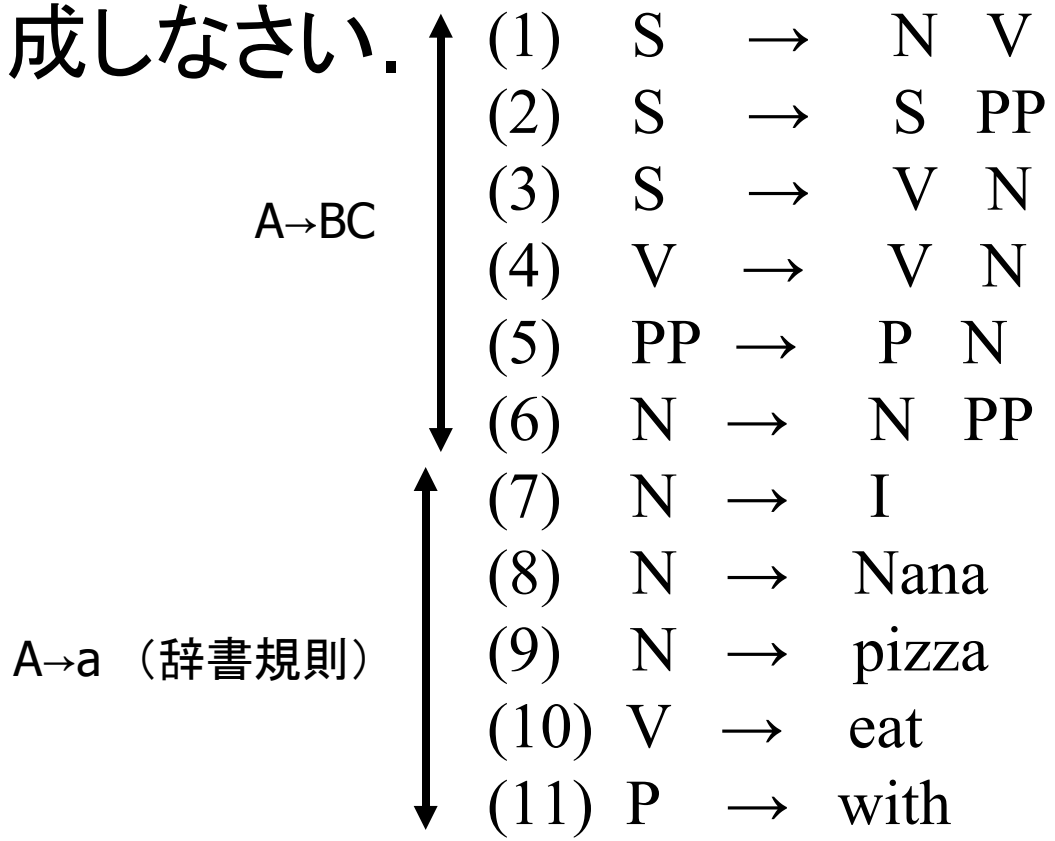


文脈自由文法に基づく構文木

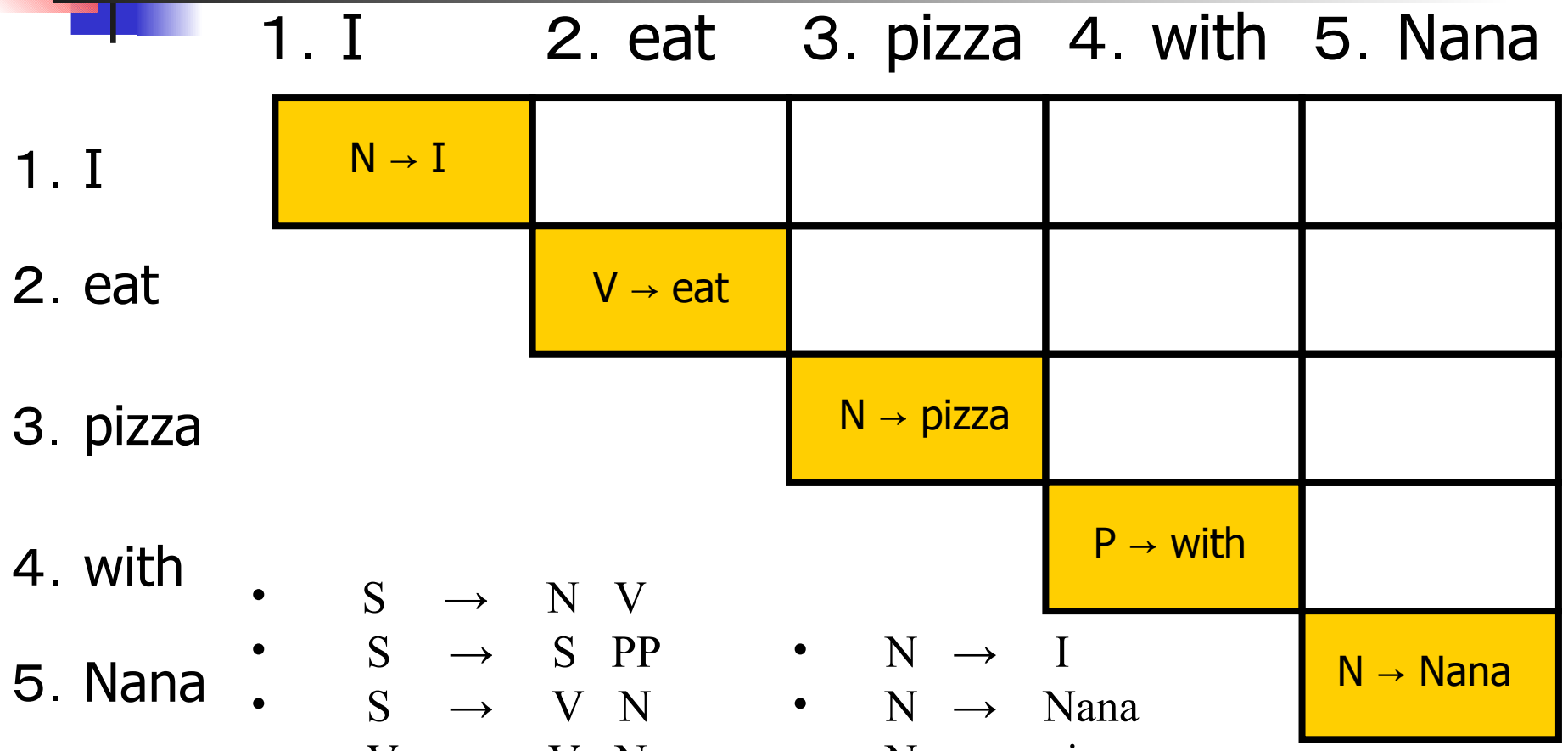


練習問題1

- CKY法を使って“I eat pizza with Nana”の構文解析結果を作成しなさい.



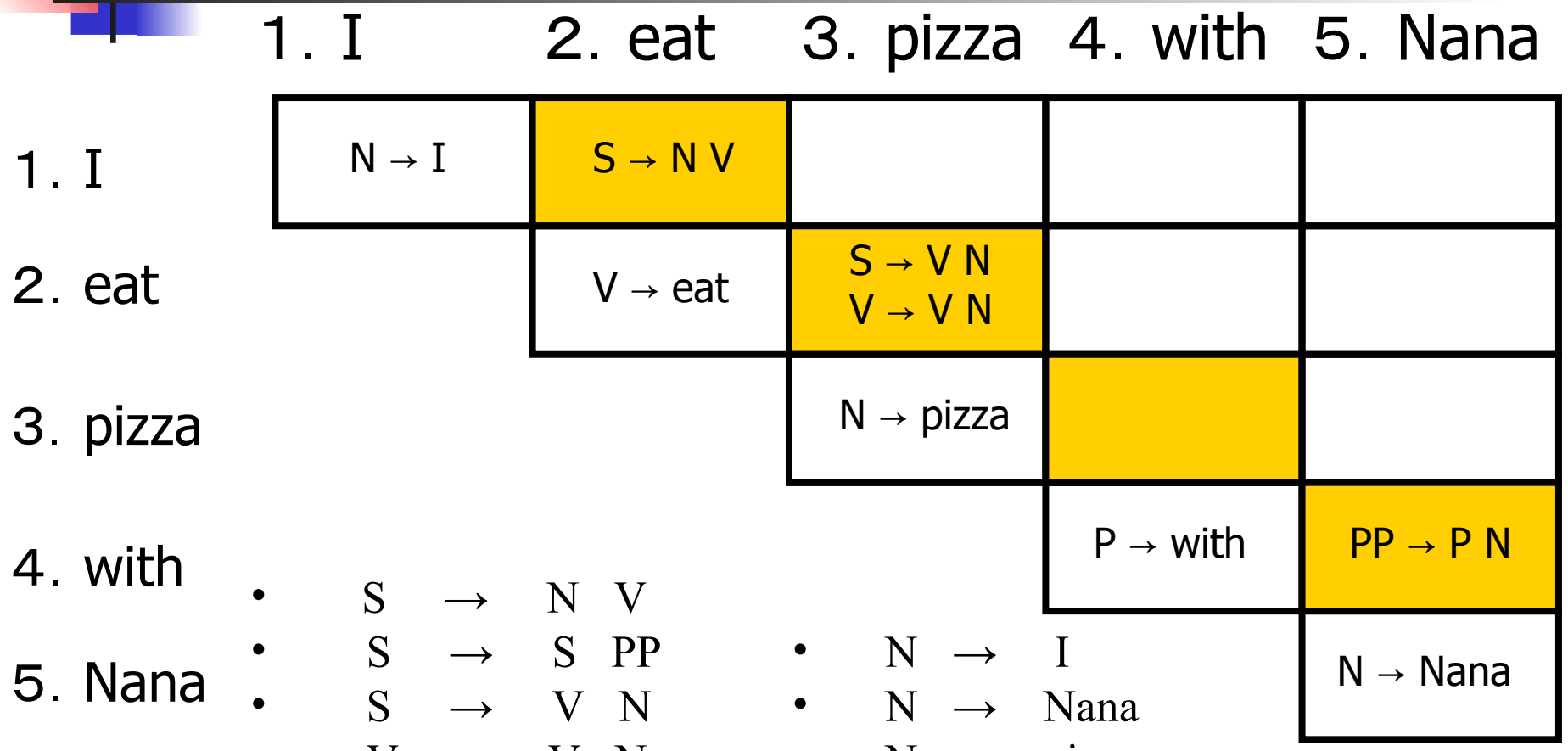
CKY法で構文解析
I eat pizza with Nana.



- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

CKY法で構文解析
I eat pizza with Nana.



- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

CKY法で構文解析
I eat pizza with Nana.

	1. I	2. eat	3. pizza	4. with	5. Nana
1. I	N → I	S → N V	S → N V		
2. eat		V → eat	S → V N V → V N		
3. pizza			N → pizza		N → N PP
4. with				P → with	PP → P N
5. Nana					N → Nana

- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

CKY法で構文解析
I eat pizza with Nana.

	1. I	2. eat	3. pizza	4. with	5. Nana
1. I	N → I	S → N V	S → N V		
2. eat		V → eat	S → V N V → V N		S → S PP S → V N V → V N
3. pizza			N → pizza		N → N PP
4. with				P → with	PP → P N
5. Nana					N → Nana

- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

CKY法で構文解析
I eat pizza with Nana.

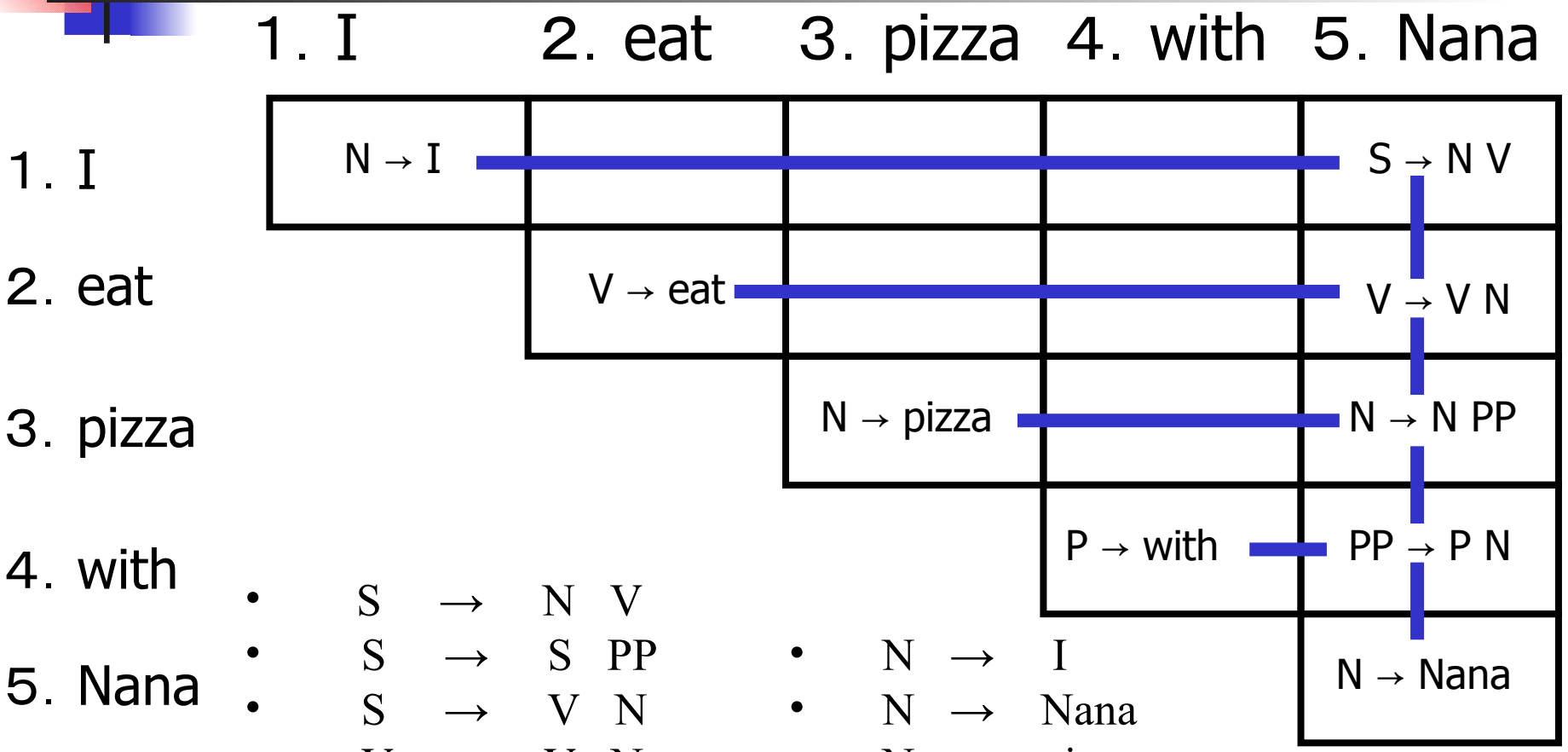
	1. I	2. eat	3. pizza	4. with	5. Nana
1. I	N → I	S → N V	S → N V		S → N V S → S PP
2. eat		V → eat	S → V N V → V N		S → S PP S → V N V → V N
3. pizza			N → pizza		N → N PP
4. with				P → with	PP → P N
5. Nana					N → Nana

- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

CKY法で構文解析　S → N V の構文木

I eat pizza with Nana.

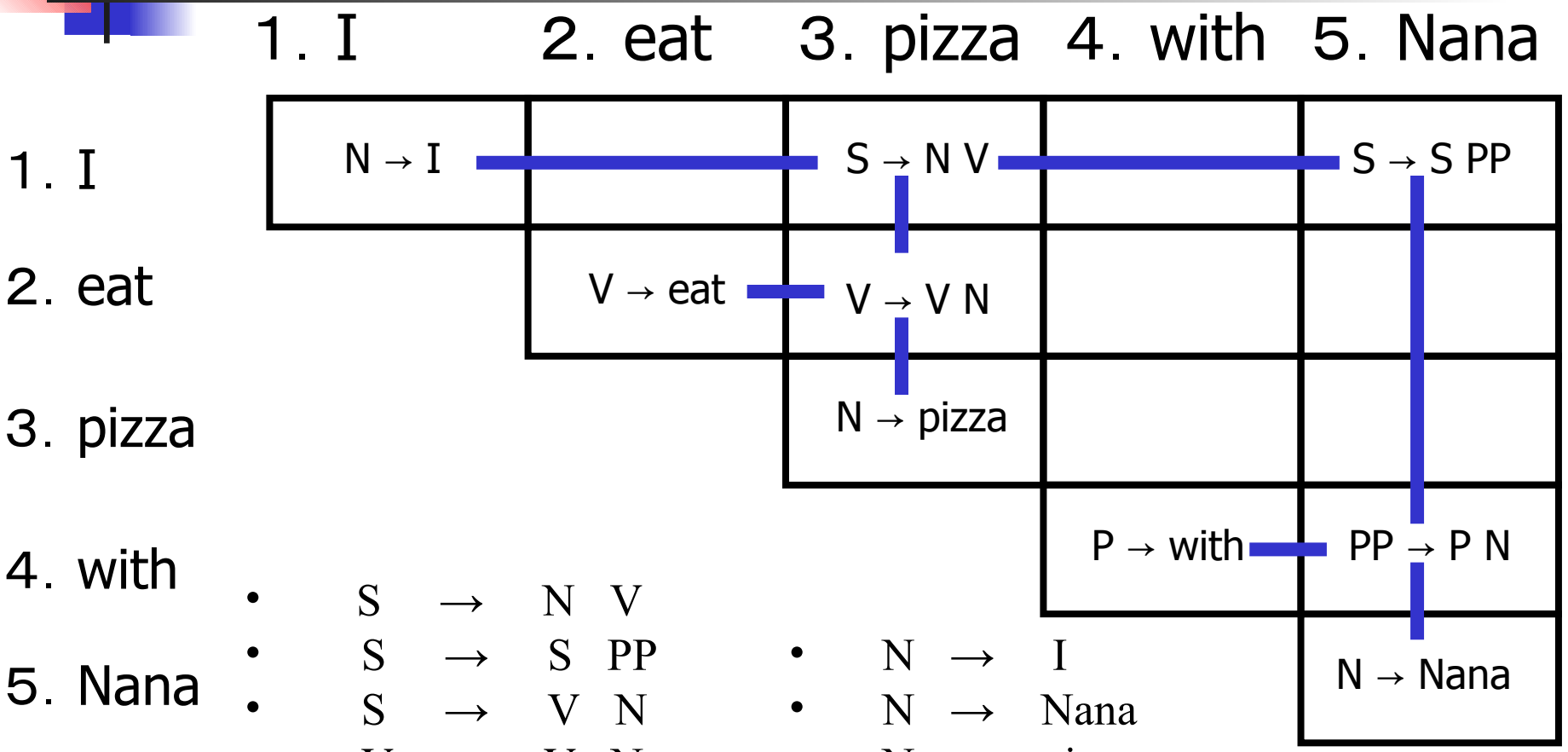


- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP

- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

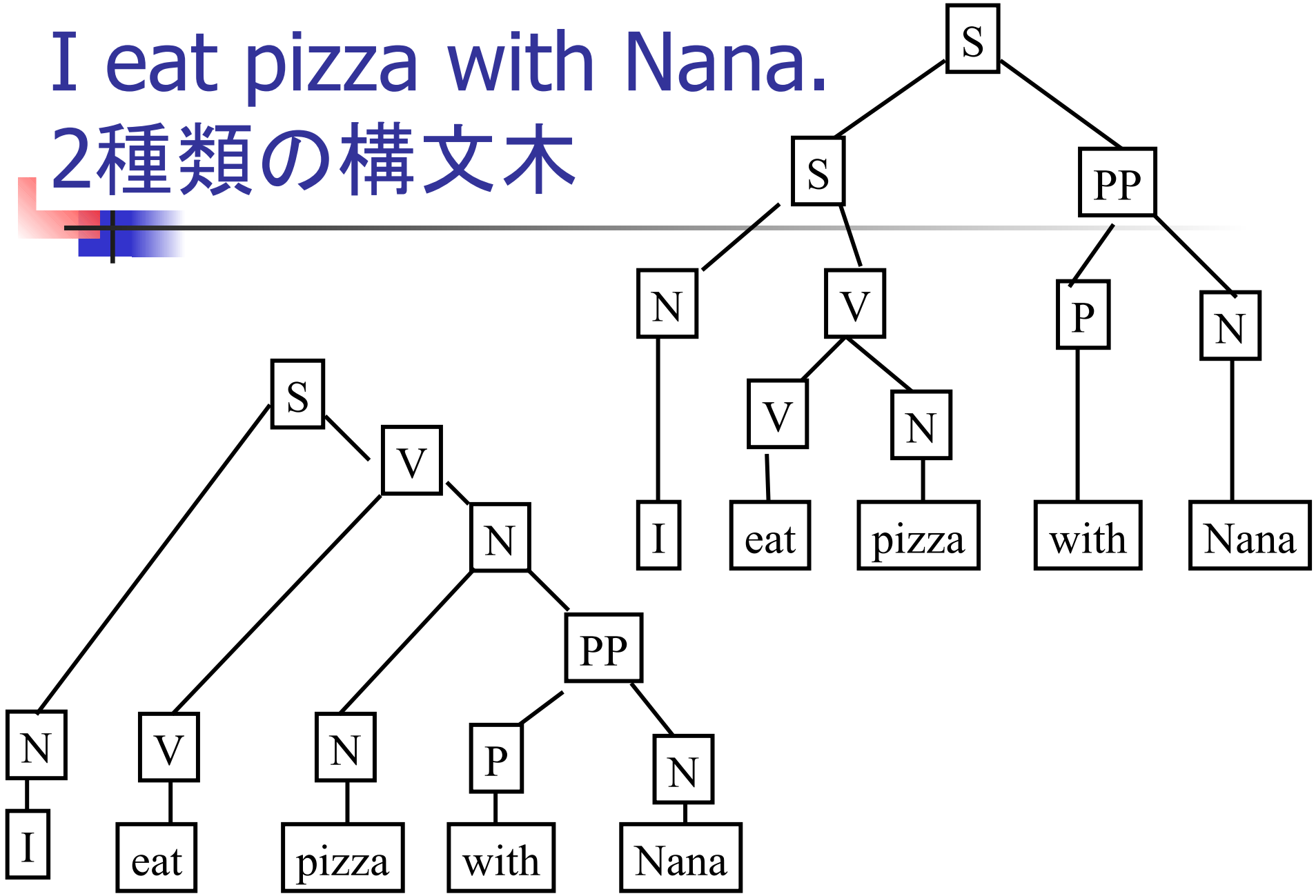
CKY法で構文解析　S → S PP の構文木

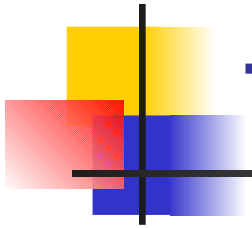
I eat pizza with Nana.



- S → N V
- S → S PP
- S → V N
- V → V N
- PP → P N
- N → N PP
- N → I
- N → Nana
- N → pizza
- V → eat
- P → with

I eat pizza with Nana.
2種類の構文木





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

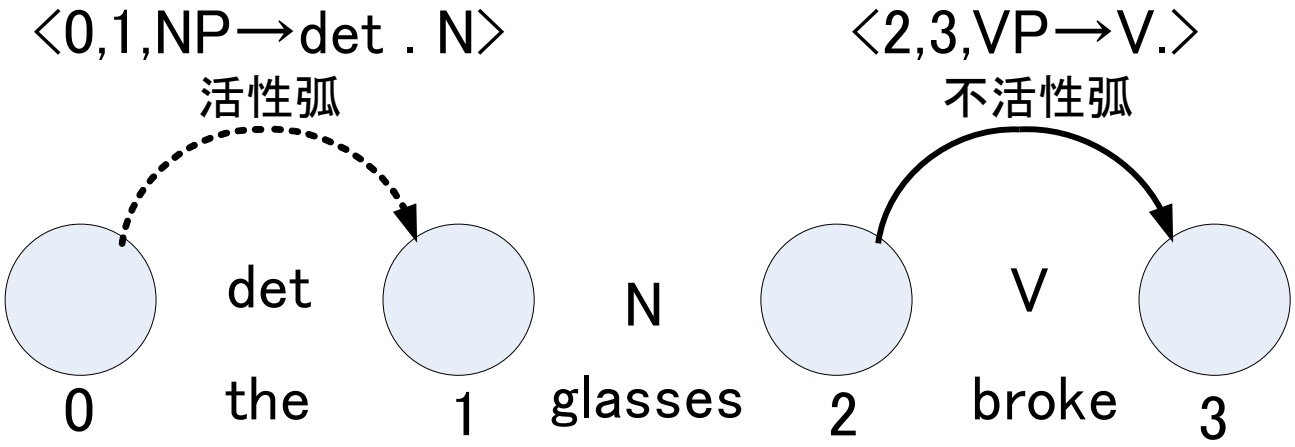
- 節点（ノード）
  - 単語と単語の間に存在する仮想的な点
- 弧（アーク）
  - 節点間を結び，分の部分的な構造を表す
  - <i,j,C → α･β>
  - iは弧の始点，jは弧の終点
  - ･は解析が終了している位置
  - 節点iからjまで解析するとα
  - βまで解析できるとC

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

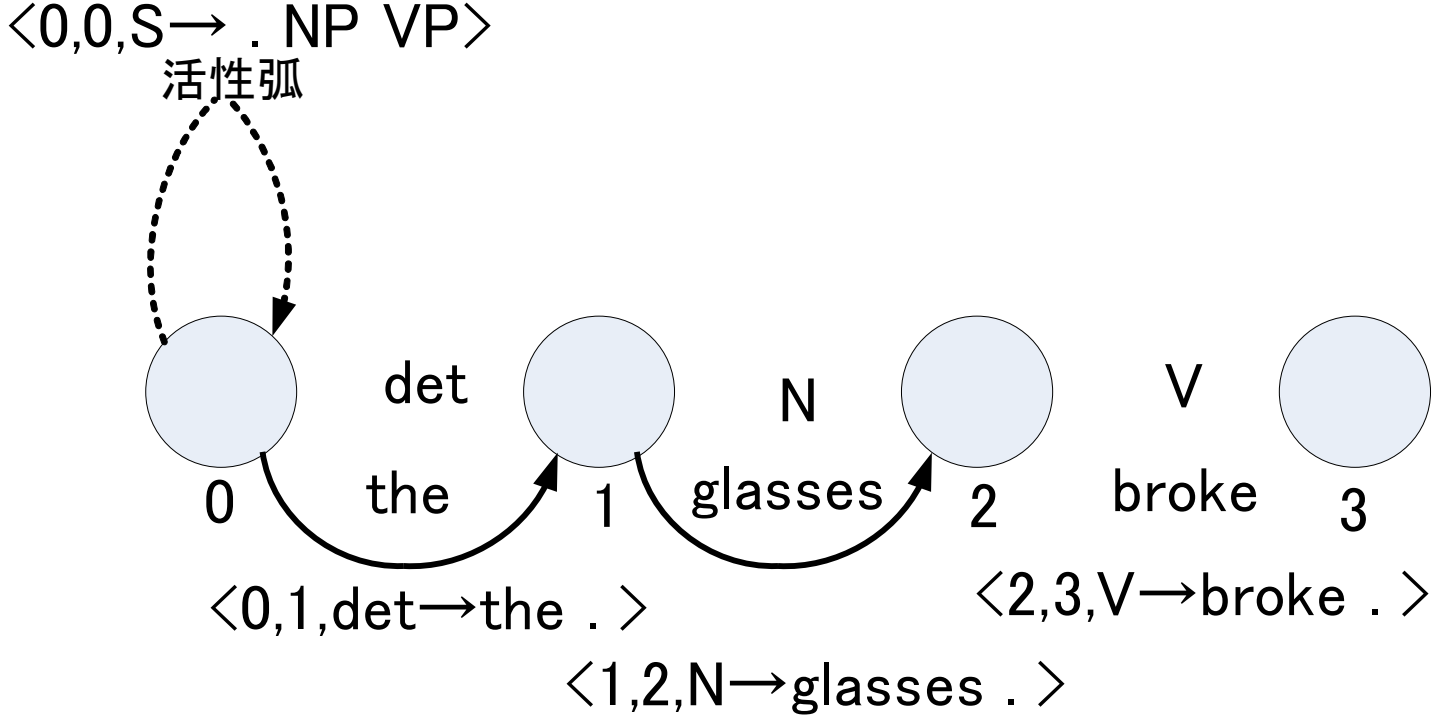
チャート法

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

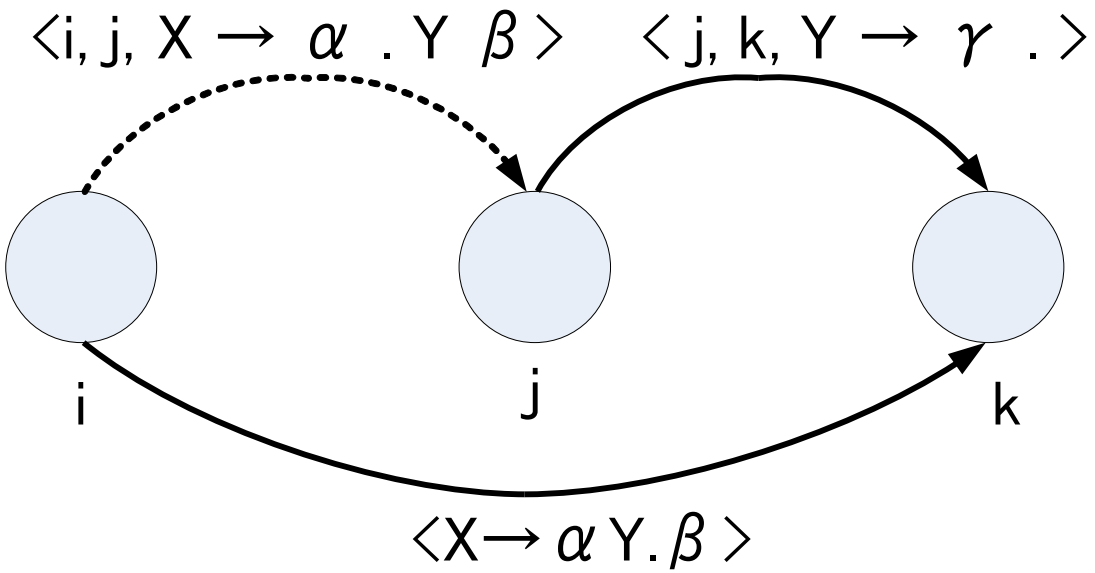


トップダウンチャート法のアルゴリズム(2/2)

- 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
  - 弧の結合
    - チャートに追加された弧が活性弧のとき，その弧の右にある不活性弧を探し，結合する
    - チャートに追加された弧が不活性弧のとき，その弧の左にある活性弧を探し，結合する
    - 結合してできた新しい弧をアジェンダに追加
  - 新しい弧の提案
    - 弧が活性弧のとき，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>



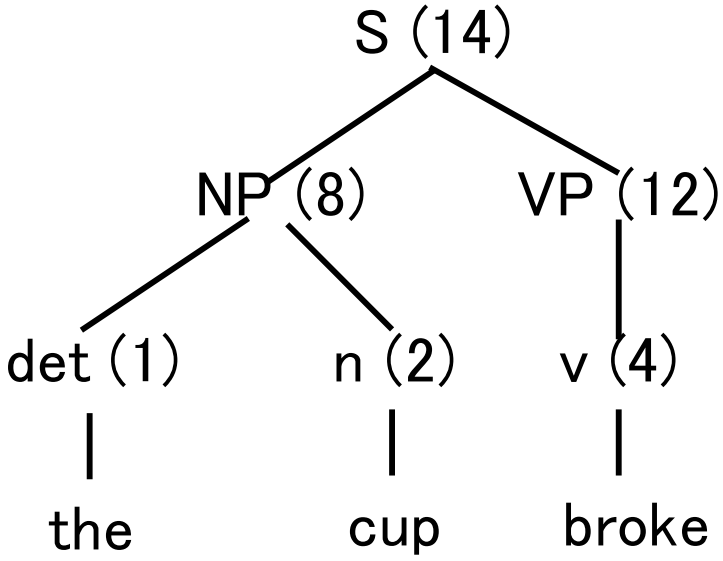
- 不活性弧<0,n,S→α.>が生成できれば解析成功

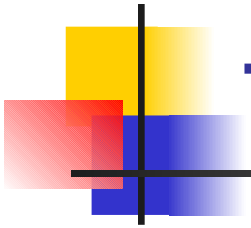
トップダウンチャート法

- 解析文
- The cup broke.
- 文法
  - S→NP VP
  - NP→det n
  - VP→v
  - VP→v NP
  - det → the
  - n → cup
  - v → broke | cup

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

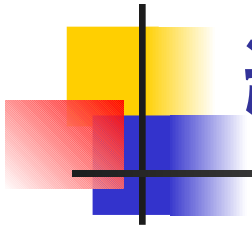




チャート法の特徴

- 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