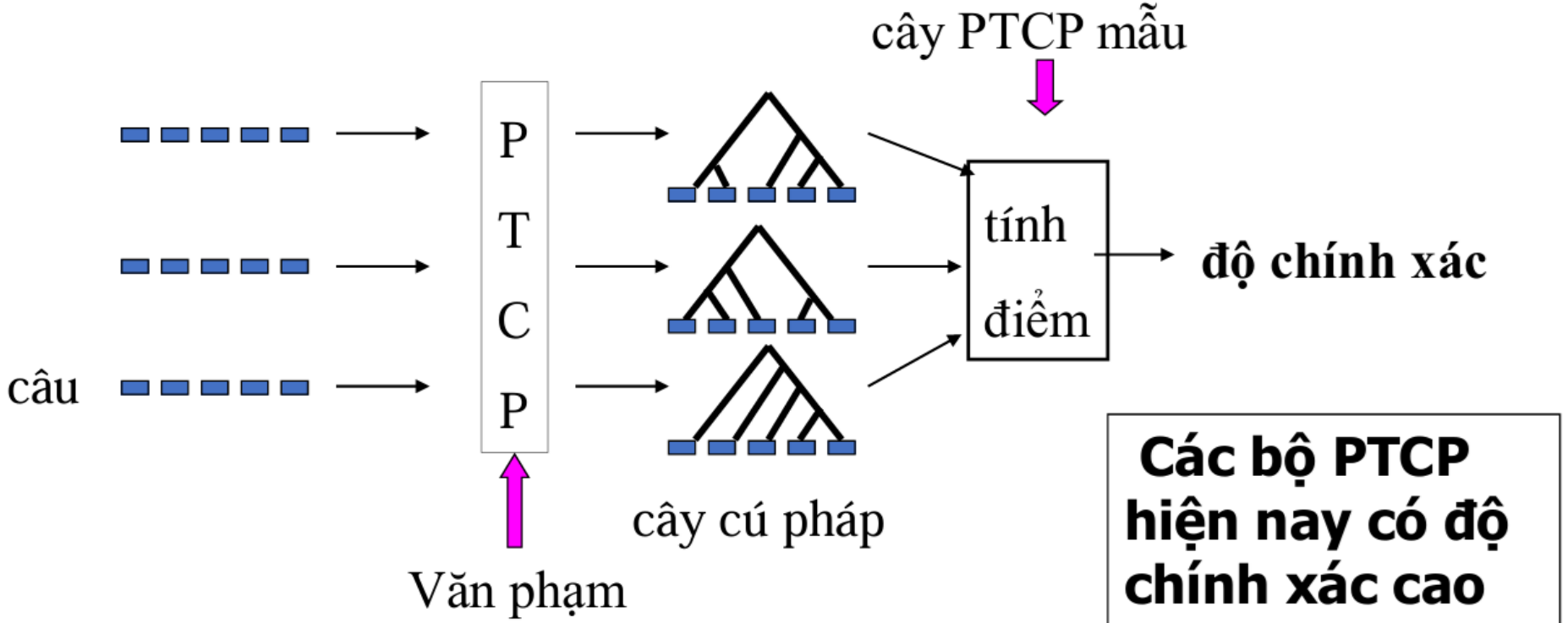
CSE: **Faculty of Computer Science and Engineering**
Thuyloi University

PHÂN TÍCH CÚ PHÁP (PARSING)

TS. Nguyễn Thị Kim Ngân
Email: ngannguyen@tlu.edu.vn

Có tham khảo bài giảng của PGS.TS Nguyễn Thanh Hương, trường đại học Bách khoa Hà Nội

Bài toán phân tích cú pháp (PTCP)

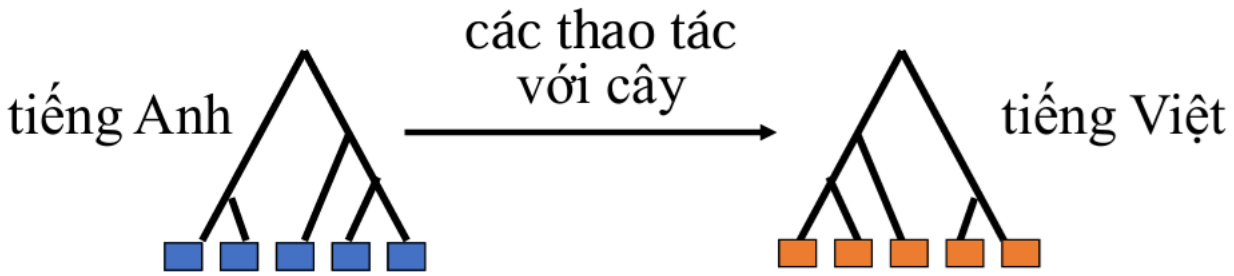


Các bộ PTCP hiện nay có độ chính xác cao

(Eisner, Collins, Charniak, etc.)

Các ứng dụng của PTCP

- Dịch máy

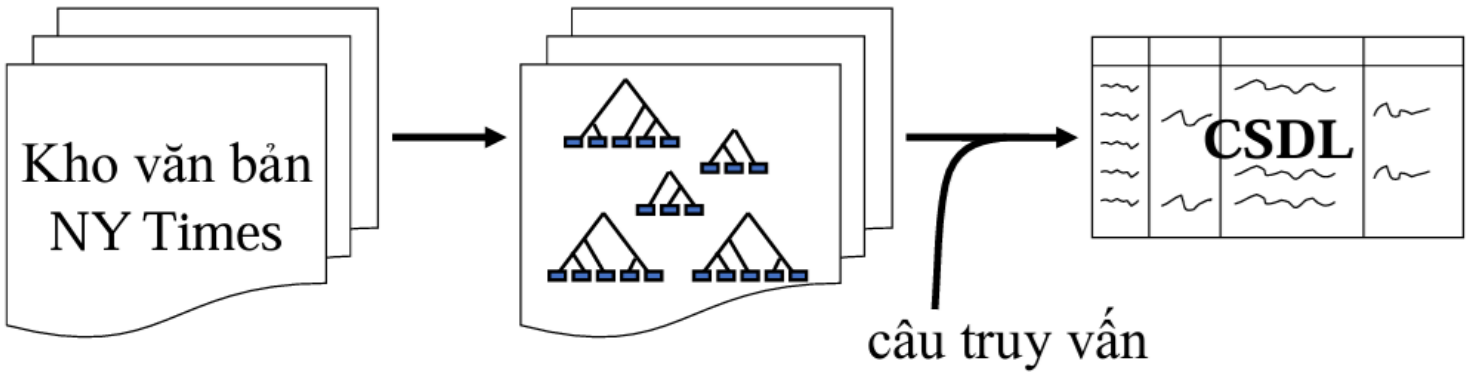


- Nhận dạng tiếng nói

Các ứng dụng của PTCP

- Kiểm tra ngữ pháp
  - Correcting spelling and grammar của Microsoft Office

- Trích rút thông tin



Định nghĩa

- Văn phạm (**grammar**) là dạng biểu diễn hình thức của các cấu trúc được chấp nhận trong 1 ngôn ngữ
- Thuật toán PTCP (**parsing algorithm**) là phương pháp xác định cấu trúc câu trên cơ sở ngữ pháp đã có
- Chương trình PTCP (**parser**) là chương trình xác định cấu trúc ngữ pháp của câu

Ví dụ về văn phạm

- Văn phạm: 1 tập luật viết lại
- Ký hiệu kết thúc: các ký hiệu không thể phân rã được nữa
- Ký hiệu không kết thúc: các ký hiệu có thể phân rã được
- Xét văn phạm G:

  S $\rightarrow$ NP VP

  NP $\rightarrow$ John, garbage

  VP $\rightarrow$ laughed, walks
- G có thể sinh ra các câu sau:

  John laughed.

  Garbage laughed.

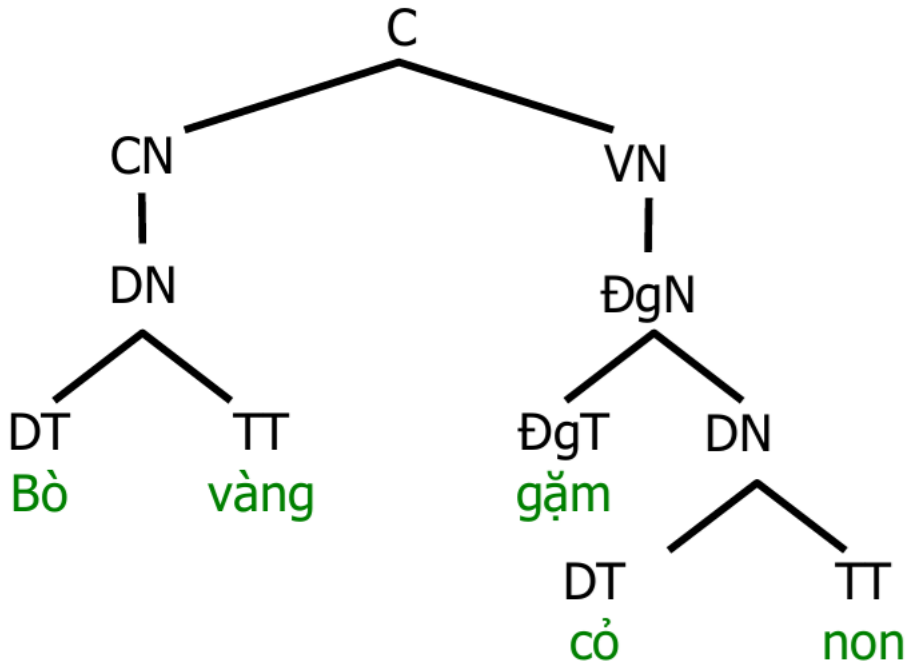
  John walks.

  Garbage walks.

Ví dụ về văn phạm

Phân tích câu "Bò vàng gặm cỏ non"

- Tập luật
  - C → CN VN
  - CN → DN
  - VN → ĐgN
  - ĐgN → ĐgT DN
  - DN → DT TT
- Cây cú pháp



Văn phạm

- Một văn phạm sản sinh là một hệ thống

  $G=(T, N, S, R)$

  Trong đó:

  - T (terminal) – tập ký hiệu kết thúc
  - N (non terminal) – tập ký hiệu không kết thúc
  - S (start) – ký hiệu khởi đầu
  - R (rule) – tập luật

    $R = \{\alpha \rightarrow \beta \mid \alpha,\beta \in (T \cup N)^*\}$

    $\alpha \rightarrow \beta$ gọi là luật sản xuất

Ví dụ

- G1 = ({a,b}, {X}, X, {X→ε, X→aXb})

  Xác định L(G1)

- G2 = ({a,b}, {X}, X, {X→ε, X→b, X→XX})

  Xác định L(G2)

Dạng chuẩn Chomsky

- **Dạng chuẩn Chomsky**:

  Văn phạm phi ngữ cảnh G=(T, N, S, R) ở dạng chuẩn Chomsky là văn phạm mà các luật sản xuất của R ở một trong hai dạng:

  - $A \rightarrow BC$
  - $A \rightarrow a$

  Trong đó: $A, B, C \in N$, $a \in T$

Dạng chuẩn Chomsky

- **Định lý**

Cho văn phạm phi ngữ cảnh G. Ta có thể thành lập một văn phạm G’ ở dạng chuẩn Chomsky sao cho L(G’)=L(G)

Đưa văn phạm về dạng chuẩn Chomsky

Thuật toán: sinh G'=(T, N', S, R') tương đương với G=(T, N, S, R)

- **Bước 1**: Nhặt các sản xuất trong R ở dạng chuẩn Chomsky đưa vào R'
- **Bước 2:** Các sản xuất còn lại đưa về dạng chuẩn Chomsky, theo cách sau:

  Bước 2.1: Nếu có sản xuất $A \rightarrow Y_1 Y_2 \ldots Y_k$ (k>2) thì thay sản xuất đó bằng hai sản xuất $A \rightarrow Y_1 A'$ và $A' \rightarrow Y_2 \ldots Y_k$. Lặp lại cho tới khi độ dài vế phải các sản xuất không lớn hơn 2

  Bước 2.2: Nếu có sản xuất mà vế phải có độ dài lớn hơn 2 và có chứa kí hiệu kết thúc a, ta thêm một kí hiệu không kết thúc C_a, thay sự xuất hiện của a trong sản xuất bằng C_a và thêm sản xuất $C_a \rightarrow a$. Quay lại thực hiện bước 1

  Các bước 2.1, bước 2.2 được thực hiện cho đến khi mọi sản xuất trong R đều được chuyển thành các sản xuất thuộc dạng chuẩn Chomsky trong R'

Đưa văn phạm về dạng chuẩn Chomsky

Ví dụ: Đưa văn phạm G=({a, b}, {A, B, S}, S, R) các luật sản xuất sau về dạng chuẩn Chomsky.
Trong đó:

R={S→aAB| BBBA
A→BAB| a
B→ AS| b}

Giải:

Bước 1: Các sản xuất đã ở dạng chuẩn Chomsky

A→a
B→AS| b

Bước 2: Đưa các sản xuất còn lại về dạng chuẩn Chomsky

S→aAB được thay bằng S→CAB và C→a
S→CAB được thay bằng S→CD và D→AB

S→BBBA được thay bằng S→BE và E→BBA
E→BBA được thay bằng E→BF và F→BA

A→BAB được thay bằng A→BT và T→AB

Đưa văn phạm về dạng chuẩn Chomsky

- Ta thu được văn phạm G'==(T, N', S, R')

  Trong đó:

  N'={S, A, B, C, D, E, F, T}

  R'={ S→CD| BE

  C→a

  D→AB

  E→BE

  F→BA

  A→BT| a

  T→AB

  B→AS| b}

Văn phạm phi ngữ cảnh (Context-Free Grammar)

… còn gọi là văn phạm cấu trúc đoạn

- G = <T,N,P,S,R>
  - T – tập các ký hiệu kết thúc (terminals)
  - N - tập các ký hiệu không kết thúc (non-terminals)
  - P – ký hiệu tiền kết thúc (preterminals), khi viết lại trở thành ký hiệu kết thúc, P⊂N
  - S – ký hiệu bắt đầu
  - R: X→γ, X , X là ký hiệu không kết thúc; γ là chuỗi các ký hiệu kết thúc và không kết thúc (có thể rỗng)
  - Văn phạm G sinh ra ngôn ngữ L
- Bộ nhận dạng: trả về yes hoặc no
- Bộ PTCP: trả về tập các cây cú pháp

Văn phạm phi ngữ cảnh

$$
\begin{array}{lcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & \left\{\begin{array}{l} DT\ NNS \\ DT\ NN \\ NP\ PP \end{array}\right. \\
VP & \rightarrow & \left\{\begin{array}{l} VP\ PP \\ VBD \\ VBD\ NP \end{array}\right. \\
PP & \rightarrow & IN\ NP
\end{array}
\qquad
\begin{array}{lcl}
DT & \rightarrow & the \\
NNS & \rightarrow & \left\{\begin{array}{l} children \\ students \\ mountains \end{array}\right. \\
VBD & \rightarrow & \left\{\begin{array}{l} slept \\ ate \\ saw \end{array}\right. \\
IN & \rightarrow & \left\{\begin{array}{l} in \\ of \end{array}\right. \\
NN & \rightarrow & cake
\end{array}
$$

Áp dụng tập luật ngữ pháp

S	→	NP VP	DT	→	*the*
NP	→	DT NNS DT NN NP PP	NNS	→	*children* *students* *mountains*
VP	→	VP PP VBD VBD NP	VBD	→	*slept* *ate* *saw*
PP	→	IN NP	IN	→	*in* *of*
			NN	→	*cake*

- S → NP VP
  - NP → DT NNS
    - DT → the
    - NNS → children
  - VP →VBD
    - VBD → slept

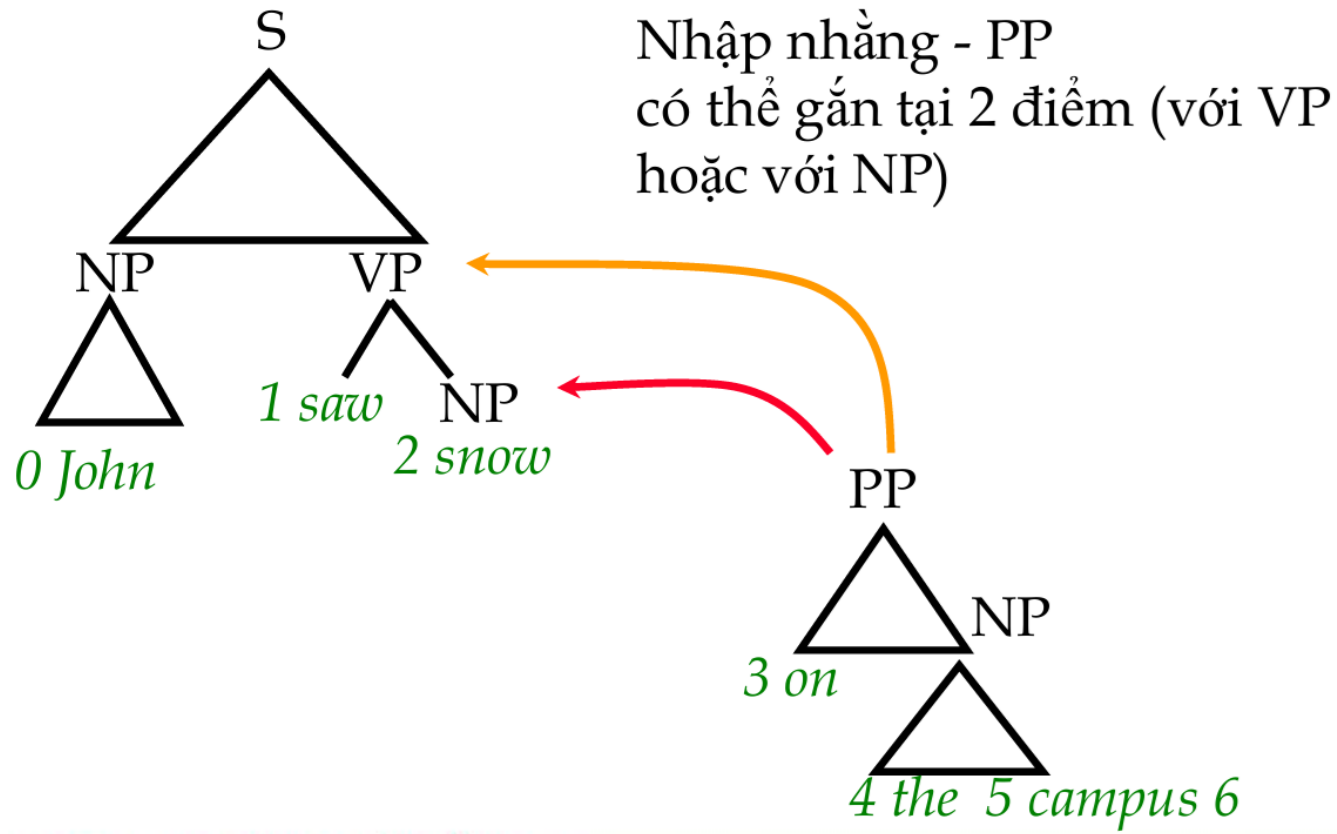
=> The children slept

- S → NP VP
  - NP → DT NNS
    - DT → the
    - NNS → children
  - VP → VBD NP
    - VBD → ate
    - NP → DT NNS
      - DT → the
      - NNS → cake

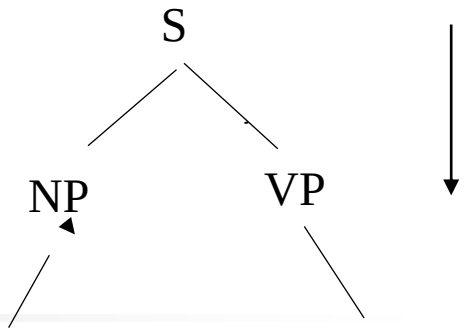
=> The children ate the cake

Văn phạm cho ngôn ngữ tự nhiên có nhập nhằng

John saw snow on the campus



PTCP kiểu trên xuống

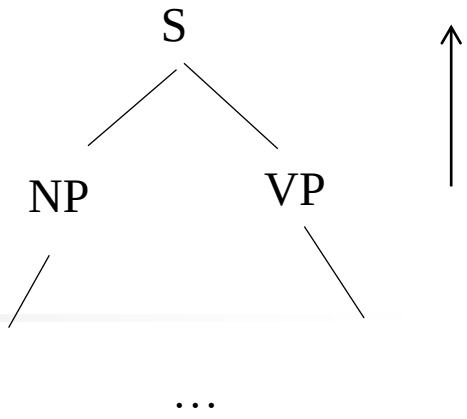


- Hướng đích
- Khởi đầu với 1 danh sách các ký hiệu cần triển khai (S, NP,VP,…)
- Viết lại các đích trong tập đích bằng cách:
  - Tìm luật có vế trái trùng với đích cần triển khai
  - Triểu khai nó với vế phải luật, tìm cách khớp với câu đầu vào
- Nếu 1 đích có nhiều cách viết lại => chọn 1 luật để áp dụng (bài toán tìm kiếm)
- Có thể sử dụng tìm kiếm rộng (breadth-first search) hoặc tìm kiếm sâu (depth-first search)

Khó khăn với PTCP trên xuống

- Các luật đệ qui trái
- PTCP trên xuống rất bất lợi khi có nhiều luật có cùng vế trái
- Nhiều thao tác thừa: triển khai tất cả các nút có thể phân tích trên xuống
- PTCP trên xuống sẽ làm việc tốt khi có chiến lược điều khiển ngữ pháp phù hợp
- PTCP trên xuống không thể triển khai các ký hiệu tiền kết thúc thành các ký hiệu kết thúc. Trên thực tế, người ta thường sử dụng phương pháp dưới lên để làm việc này
- Lặp lại công việc: bất cứ chỗ nào có cấu trúc giống nhau

PTCP kiểu dưới lên



- Hướng dữ liệu
- Khởi tạo với xâu cần phân tích
- Nếu chuỗi trong tập đích phù hợp với vế phải của 1 luật => thay nó bằng vế trái của luật
- Kết thúc khi tập đích = {S}.
- Nếu vế phải của các luật khớp với nhiều luật trong tập đích, cần lựa chọn luật áp dụng (bài toán tìm kiếm)
- Có thể sử dụng tìm kiếm rộng (breadth-first search) hoặc tìm kiếm sâu (depth-first search)

Khó khăn với PTCP dưới lên

- Không hiệu quả khi có nhiều nhập nhằng mức từ vựng
- Lặp lại công việc: bất cứ khi nào có cấu trúc con chung
- Cả PTCP TD (LL) và BU (LR) đều có độ phức tạp là hàm mũ của độ dài câu.

Thuật toán CKY (bộ nhận dạng)

- Vào: xâu n từ, văn phạm phi ngữ cảnh thuộc dạng chuẩn Chormsky
- Ra: yes/no
- Cấu trúc ngữ pháp: bảng nxn (chart table)
  - Hàng đánh số 0 đến n-1
  - Cột đánh số 1 đến n
  - Cell [i,j] liệt kê tất cả các nhãn cú pháp giữa i và j

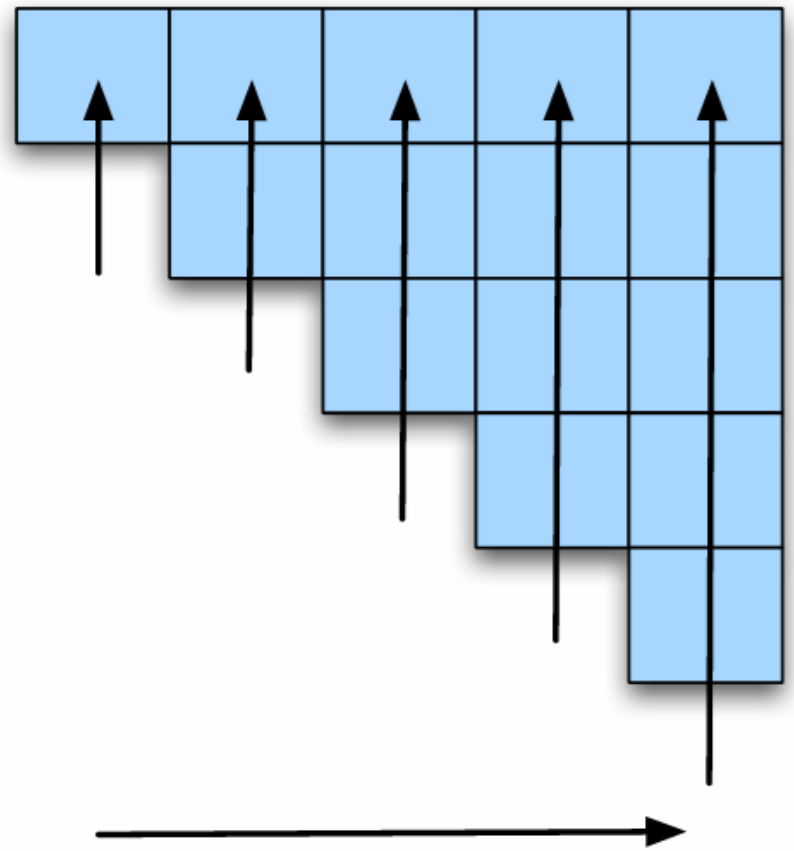
Thuật toán CKY (bottom-up)

function CKY-PARSE(*words, grammar*) **returns** *table*

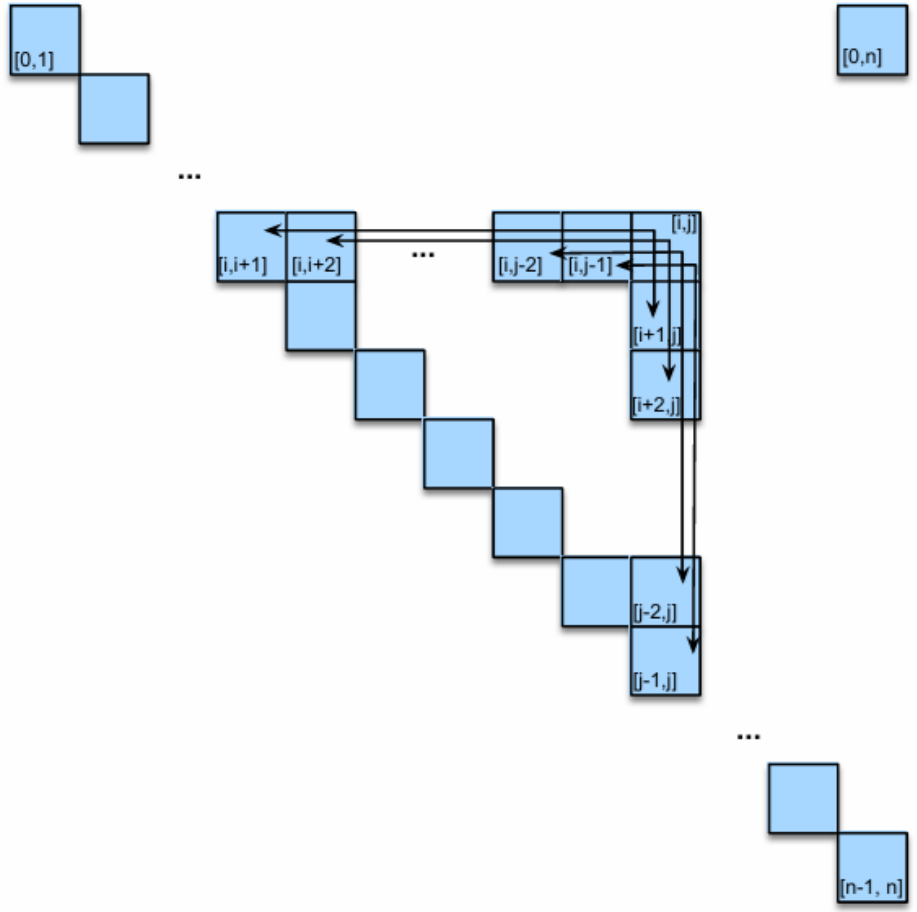
for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$
  for $i \leftarrow$ **from** $j-2$ **down to** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

Thuật toán CKY (bottom-up)

Book	*the*	*flight*	*through*	*Houston*
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]



Thuật toán CKY (bottom-up)

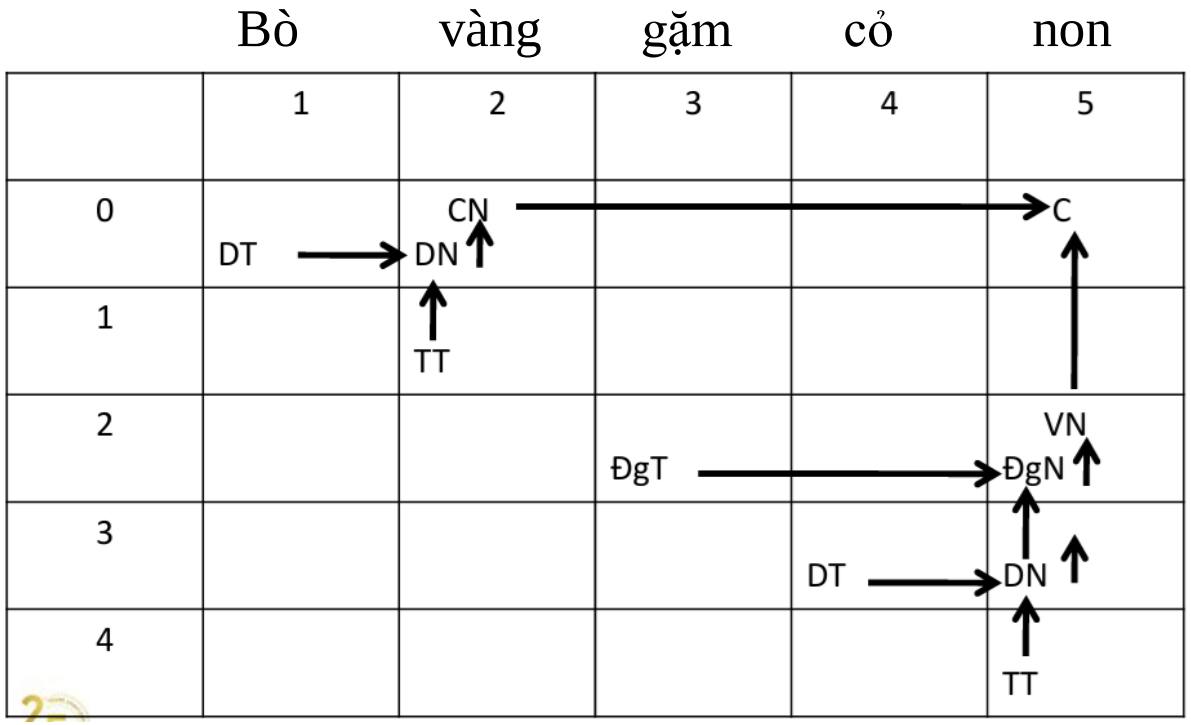


Ví dụ

- C → CN VN
- CN → DN
- VN → ĐgN

ĐgN → ĐgT DN
DN → DT TT

DT → bò | cỏ
TT → vàng | non
ĐgT → gặm



Tính xác suất

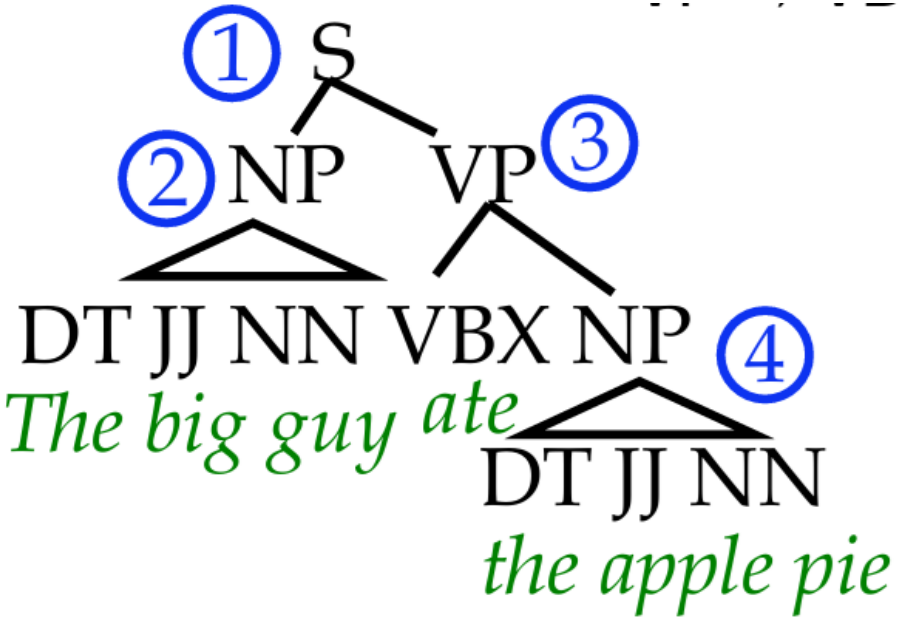
$$\Pr(X \rightarrow Y) = \frac{X \Rightarrow NP}{Y} = \frac{DT\ JJ\ NN}{NP} = \frac{1470}{9711} = 0.1532$$

Tính Pr

S → NP VP; 0.35

NP → DT JJ NN; 0.1532

VP → VBX NP; 0.302



Luật áp dụng	Chuỗi Pr
1 S → NP VP	0.35
2 NP → DT JJ NN	0.1532 x 0.35 = 0.0536
3 VP → VBX NP	0.302 x 0.0536 = 0.0162
4 NP DT → JJ NN	0.1532 x 0.0162 = 0.0025

Pr = 0.0025

Văn phạm phi ngữ cảnh xác suất

- 1 văn phạm phi ngữ cảnh xác suất (Probabilistic Context Free Grammar) gồm các phần thông thường của CFG
- Tập ký hiệu kết thúc $\{w^k\}$, $k = 1, \ldots, V$
- Tập ký hiệu không kết thúc $\{N^i\}$, $i = 1, \ldots, n$
- Ký hiệu khởi đầu N^1
- Tập luật $\{N^i \rightarrow \zeta^j\}$, ζ^j là chuỗi các ký hiệu kết thúc và không kết thúc
- Tập các xác suất của 1 luật là:

$$\forall I \sum_j P(N^i \rightarrow \zeta^j) = 1$$

- Xác suất của 1 cây cú pháp:

$$P(T) = \Pi_{i=1..n} p(r(i))$$

Các giả thiết

- Độc lập vị trí: Xác suất 1 cây con không phụ thuộc vào vị trí của các từ của cây con đó ở trong câu

  $\forall k, P(N_{jk}(k+c) \rightarrow \zeta)$ là giống nhau

- Độc lập ngữ cảnh: Xác suất 1 cây con không phụ thuộc vào các từ ngoài cây con đó

  $P(N_{jkl} \rightarrow \zeta \mid$ các từ ngoài khoảng k đến l) = $P(N_{jkl} \rightarrow \zeta)$

- Độc lập tổ tiên: Xác suất 1 cây con không phụ thuộc vào các nút ngoài cay con đó

  $P(N_{jkl} \rightarrow \zeta \mid$ các từ ngoài cây con $N_{jkl}) = P(N_{jkl} \rightarrow \zeta)$

CKY kết hợp xác suất

- Cấu trúc dữ liệu:
  - Mảng lập trình động π[i,j,a] lưu xác suất lớn nhất của ký hiệu không kết thúc a triển khai thành chuỗi i…j.
  - Backptrs lưu liên kết đến các thành phần trên cây
- Ra: Xác suất lớn nhất của cây

Tính Pr dựa trên suy diễn

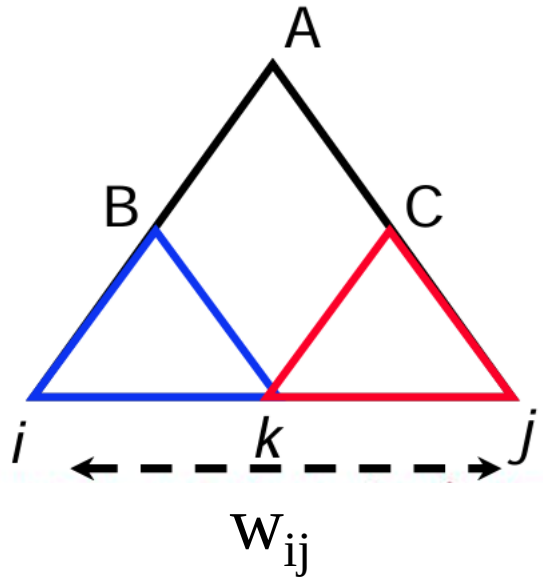
- Trường hợp cơ bản: chỉ có 1 từ đầu vào

  $Pr(tree) = pr(A \rightarrow w_i)$

- Trường hợp đệ qui: Đầu vào là xâu các từ

  $A \rightarrow^* w_{ij}$ if $\exists k$: $A \rightarrow BC$, $B \rightarrow^* w_{ik}$,$C \rightarrow^* w_{kj}$, $i \leq k \leq j$

  $p[i,j] = \max(p(A \rightarrow BC) \times p[i,k] \times p[k,j])$



CKY kết hợp xác suất

function CYK(*words*,*grammar*) **returns** *best_parse*

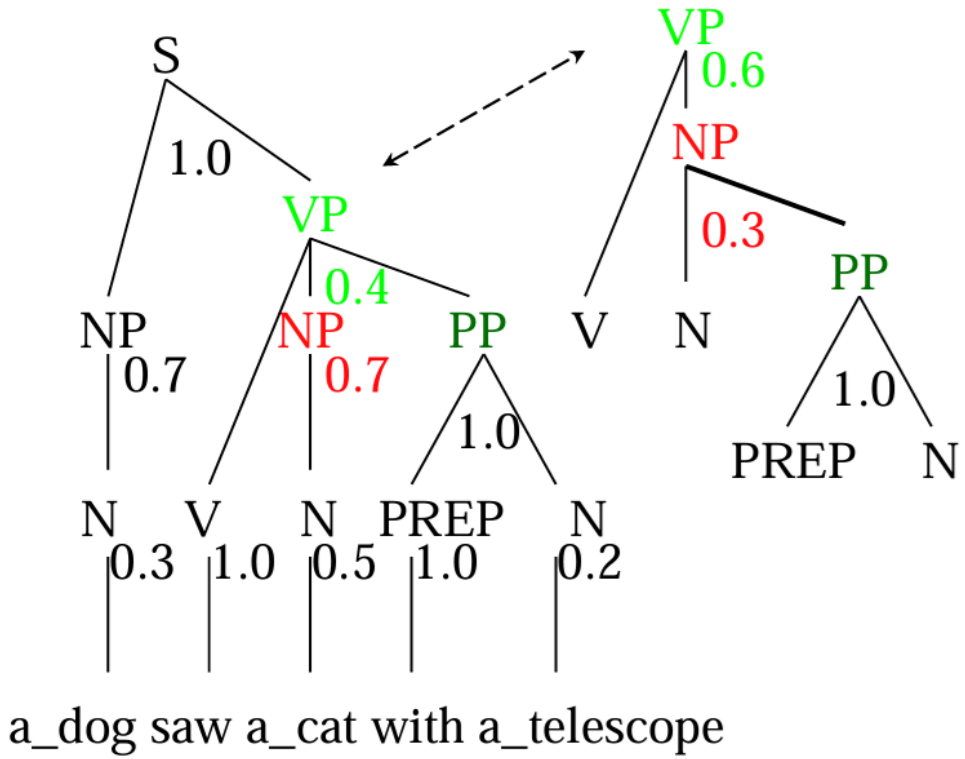
Create and clear *p*[*num_words*,*num_words*,*num_nonterminals*]

base case
for $i = 1$ **to** *num_words*
  for $A = 1$ **to** *num_nonterminals*
    if $A \rightarrow w_i$ is in grammar **then**
      $\pi[i,i,A] = P(A \rightarrow w_i)$

recursive case
for $j = 2$ **to** *num_words*
  for $i = 1$ **to** *num_words*-j+1
    for $k = 1$ **to** j-1
      for $A = 1$ **to** *num_nonterminals*
      for $B = 1$ **to** *num_nonterminals*
      for $C = 1$ **to** *num_nonterminals*
        $prob = \pi[i,k,B] \times p[i+k,j-k,C] \times P(A \rightarrow BC)$
        if $(prob > \pi[i,j,A])$ **then**
          $\pi[i,j,A] = \text{prob}$
          $B[i,j,A] = \{k,A,B\}$

Ví dụ

1.	S → NP VP	1.0
2.	VP → V NP PP	0.4
3.	VP → V NP	0.6
4.	NP → N	0.7
5.	NP → N PP	0.3
6.	PP → PREP N	1.0
7.	N → a_dog	0.3
8.	N → a_cat	0.5
9.	N → a_telescop	0.2
10.	V → saw	1.0
11.	PREP → with	1.0



Pl = 1´.7´.4´.3´.7´1´.5´.1´.1´.2 =.00588

Pr = 1´.7´.6´.3´.3´1´.5´.1´.1´.2=.00378 => P_1 được chọn