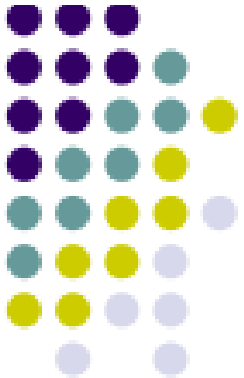
Phân tích cú pháp

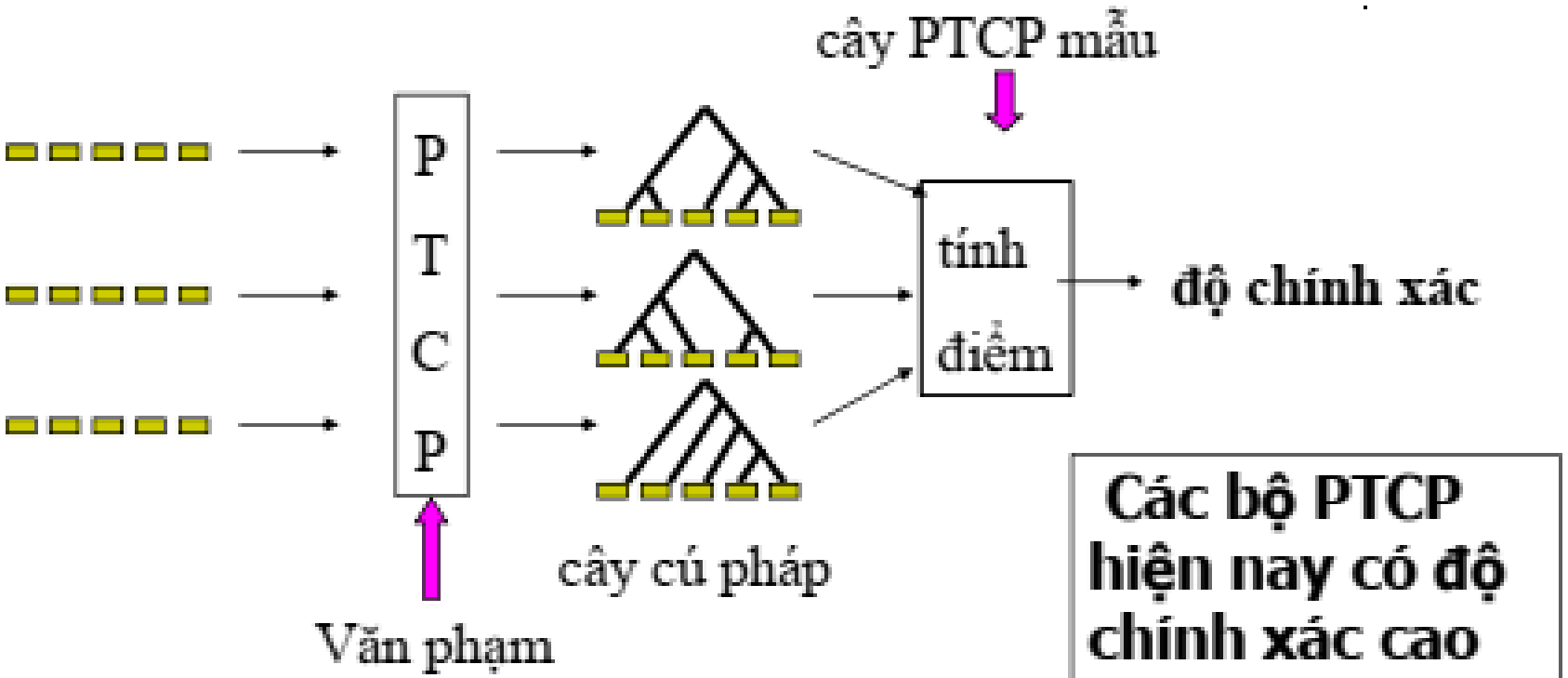
TS. Trần Hồng Việt

Email: thviet79@gmail.com

Phone: 0975486888



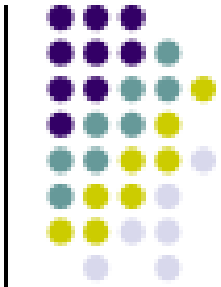
Bài toán PTCP



Các bộ PTCP hiện nay có độ chính xác cao

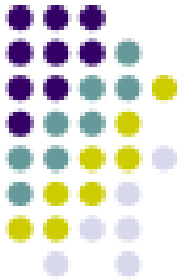
(Eisner, Collins, Charniak, etc.)

Khái niệm về văn phạm



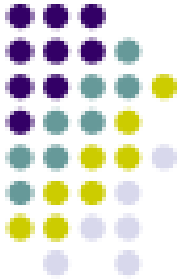
- Phân tích câu "Bò vàng gặm cỏ non"
- Cây cú pháp:
- Tập luật
  - C → CN VN
  - CN → DN
  - VN → ĐgN
  - ĐgN → ĐgT DN
  - DN → DT TT

Văn phạm



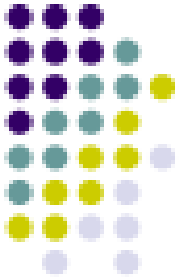
- Một văn phạm sản sinh là một hệ thống
- G = (T, N, S, R), trong đó
- T (terminal) – tập ký hiệu kết thúc
- N (non terminal) – tập ký hiệu không kết thúc
- S (start) – ký hiệu khởi đầu
- R (rule) – tập luật
- $R = \{ \alpha \rightarrow \beta \mid \alpha, \beta \in (T \cup N) \}$
- $\alpha \rightarrow \beta$ gọi là luật sản xuất

Dạng chuẩn Chomsky



- Mọi NNPNC không chứa ε đều có thể sinh từ một văn phạm tnđó mọi sản xuất đều có dạng $A \rightarrow BC$ hoặc $A \rightarrow a$, với $A,B,C \in N$ và $a \in T$
- Ví dụ: Tìm dạng chuẩn Chomsky cho văn phạm G với T = {a,b}, N ={S,A,B}, R như sau:
  - S → bA|aB
  - A →bAA|aS|a
  - B → aBB|bS|b

Nhắc lại 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:

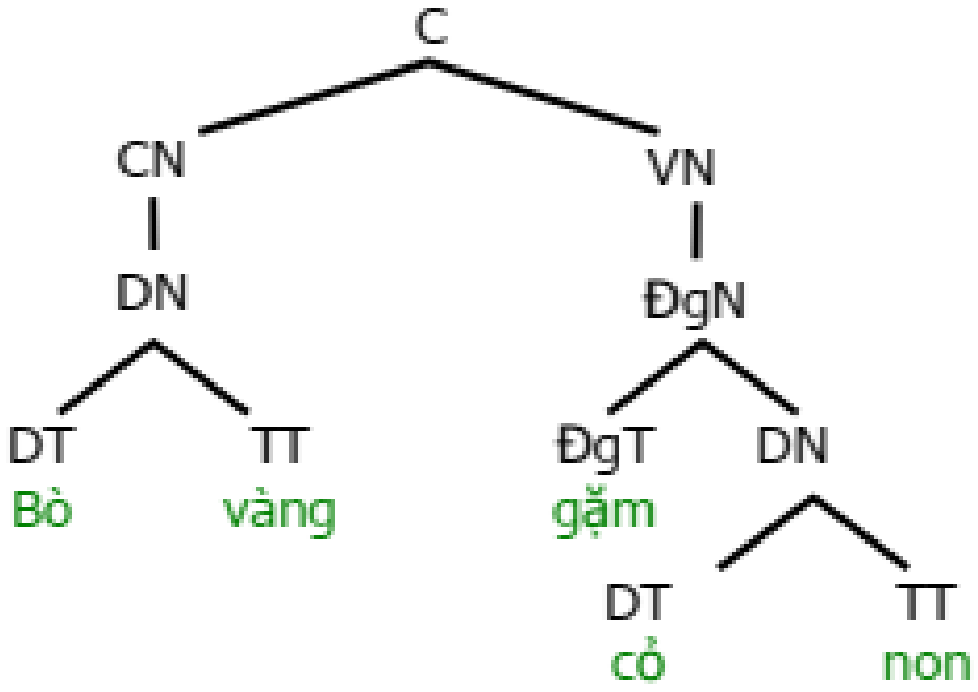
John laughed. John walks.

Garbage laughed. Garbage walks.

Cấu trúc ngữ pháp

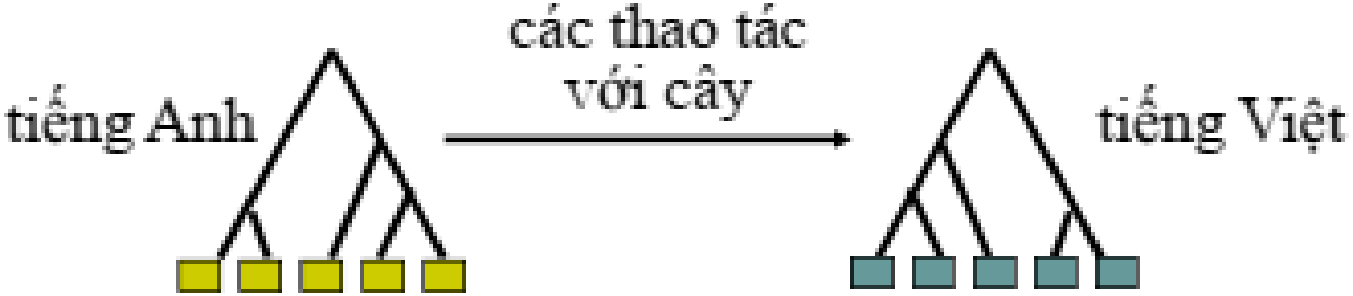
Cây cú pháp biểu diễn cấu trúc ngữ pháp của một câu.

Bò vàng gặm cỏ non.



Các ứng dụng của PTCP

- Dịch máy (Alshawi 1996, Wu 1997, ...)



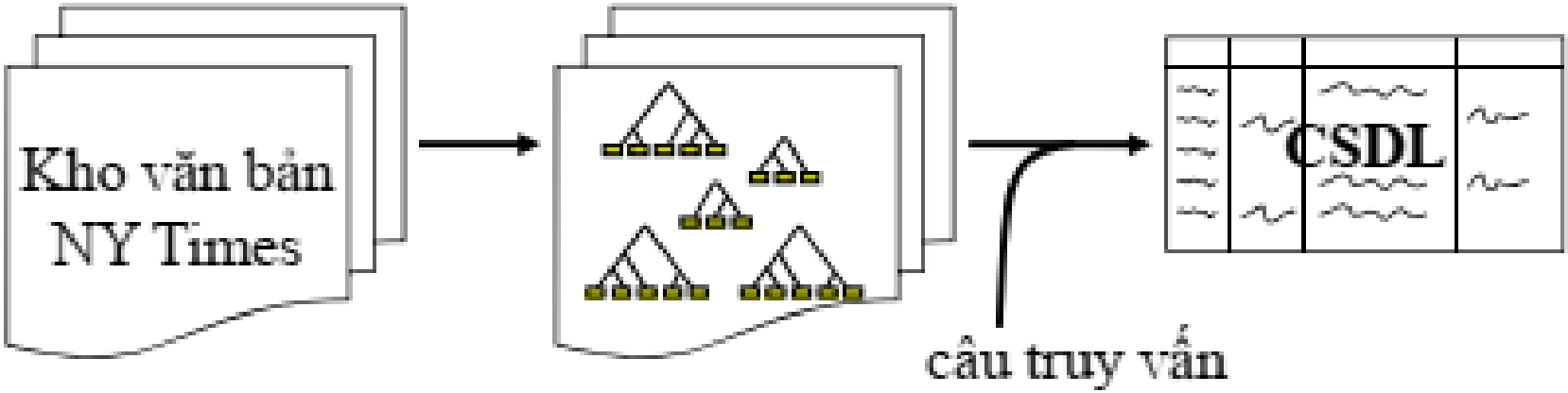
- Nhận dạng tiếng nói sử dụng PTCP (Chelba et al 1998)

  Put the file in the folder.

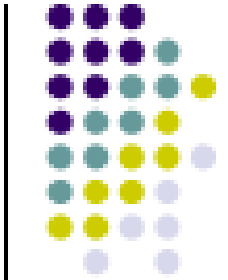
  Put the file and the folder.

Các ứng dụng của PTCP

- Kiểm tra ngữ pháp (Microsoft)
- Trích rút thông tin (Hobbs 1996)



Văn phạm phi ngữ cảnh (Context-Free Grammar)



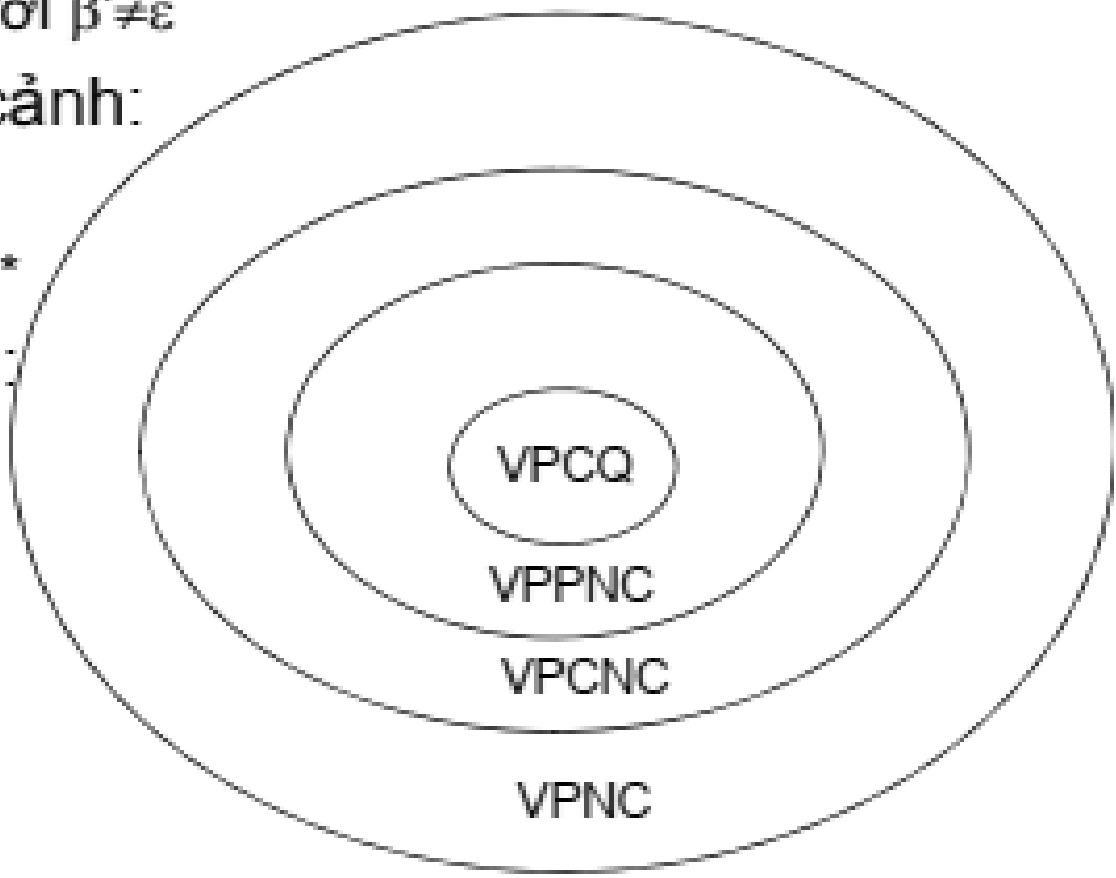
… còn gọi là văn phạm cấu trúc đoạn

- 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, $P \subset N$
  - S – ký hiệu bắt đầu
  - R: $X \rightarrow \gamma$, 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

So với văn phạm cảm ngữ cảnh
R: $\alpha A \gamma \Rightarrow \alpha \beta \gamma$

- Văn phạm ngữ cấu:
  - $\alpha \rightarrow \beta$, với $\alpha \in V+$, $\beta \in V^*$
- Văn phạm cảm ngữ cảnh:
  - $r = \alpha \rightarrow \beta$, với $\alpha \in V+$, $\beta \in V^*$, $|\alpha| \leq |\beta|$
  - và $\alpha 1 A \alpha 2 \rightarrow \alpha 1 \beta' \alpha 2$ với $\beta' \neq \varepsilon$
- Văn phạm phi ngữ cảnh:
  - $A \rightarrow \theta$, $A \in N$,
  - với $\theta \in V^* = (T \cup N)^*$
- Văn phạm chính qui:
  - $A \rightarrow aB$,
  - $A \rightarrow Ba$,
  - $A \rightarrow a$,

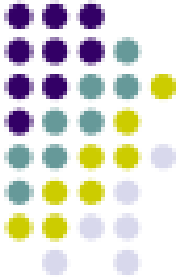
  với $A, B \in N$, $a \in T$.



Văn phạm phi ngữ cảnh

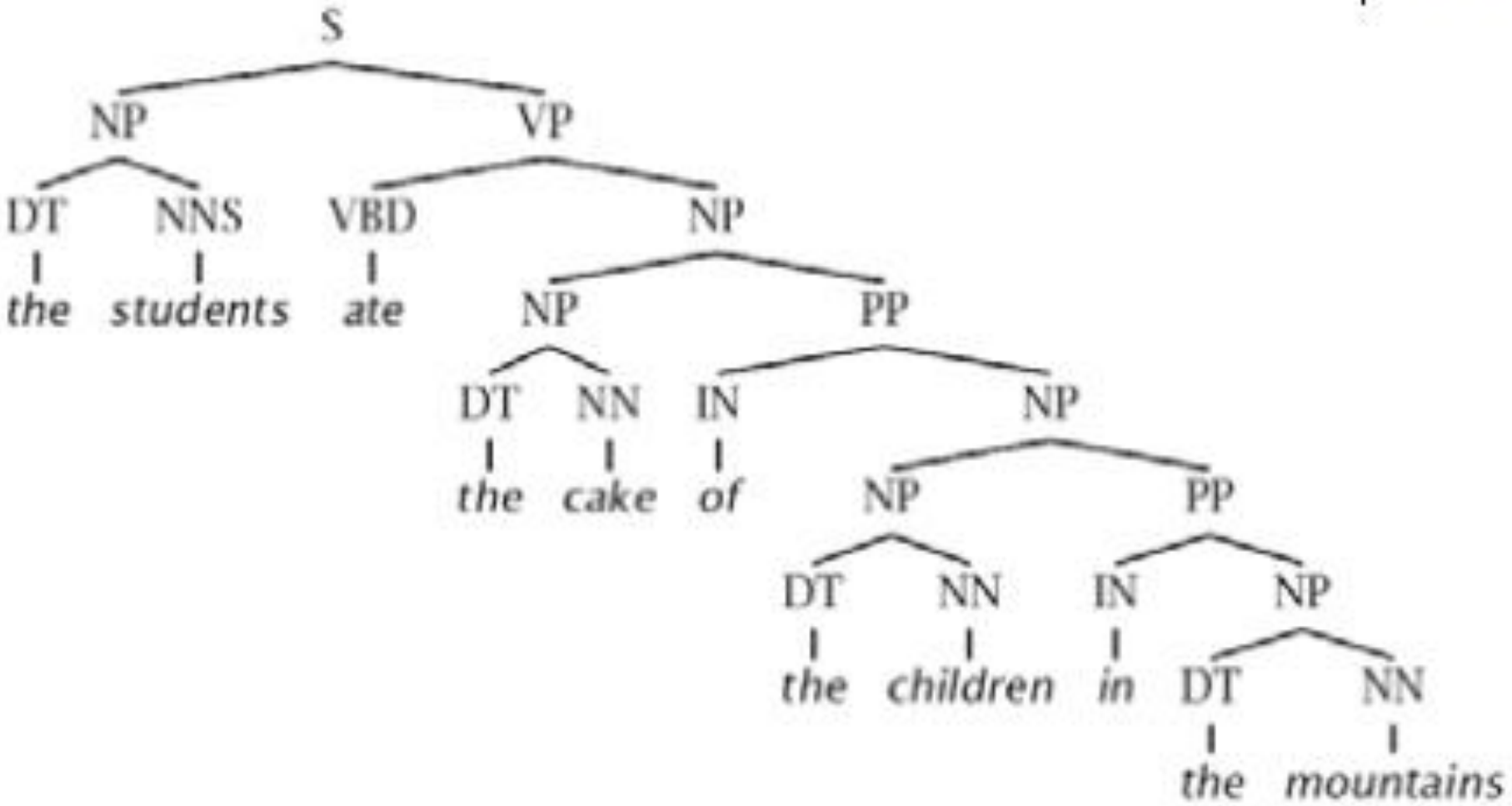
$$
\begin{array}{lcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & \left\{\begin{array}{l} DT\ NNS \\ DT\ NN \\ NP\ PP \end{array}\right. \\
VP & \rightarrow & \left\{\begin{array}{l} VP\ PP \\ VBD \\ VBD\ NP \end{array}\right. \\
PP & \rightarrow & IN\ NP
\end{array}
\qquad
\begin{array}{lcl}
DT &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

Áp dụng tập luật ngữ pháp

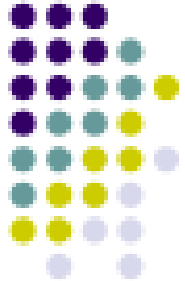


- S
  - → NP VP
  - → DT NNS VBD
  - → *The children slept*
- S
  - → NP VP
  - → DT NNS VBD NP
  - → DT NNS VBD DT NN
  - → *The children ate the cake*

Cấu trúc đoạn đệ qui

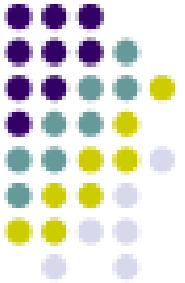


Thuật toán CKY (bộ nhận dạng)



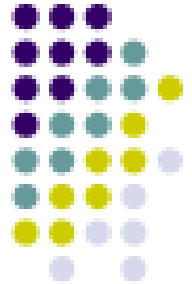
- Vào: xâu n từ
- Ra: yes/no
- Cấu trúc ngữ pháp: bảng n x n (chart table)
  - hàng đánh số 0 đến n-1
  - cột đánh số 1 đến n
  - cell [i,j] liệt kê tất cả các nhãn cú pháp giữa i và j

Thuật toán CKY (bottom-up)



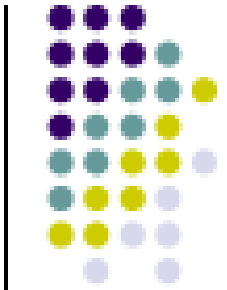
- for i := 1 to n
  - Thêm tất cả từ loại của từ thứ i vào ô [i-1,i]
- for width := 2 to n
  - for start := 0 to n-width
    - end := start + width
    - for mid := start+1 to end-1
      - for mọi nhãn cú pháp X trong [start,mid]
      - for mọi nhãn cú pháp Y trong [mid,end]
      - for mọi cách kết hợp X và Y (nếu có)
      - Thêm nhãn kết quả vào [start,end] nếu chưa có nhãn này

Văn phạm phi ngữ cảnh



1. Start→ S
2. S → NP VP
3. NP → Det Noun
4. NP → Name
5. NP → Name PP
6. PP → Prep NP
7. VP → V NP
8. VP → V NP PP
9. V → *ate*
10. Name → *John*
11. Name → *ice-cream, snow*
12. Noun → *ice-cream, pizza*
13. Noun → *table, guy, campus*
14. Det → *the*
15. Prep → *on*

Luật kết hợp

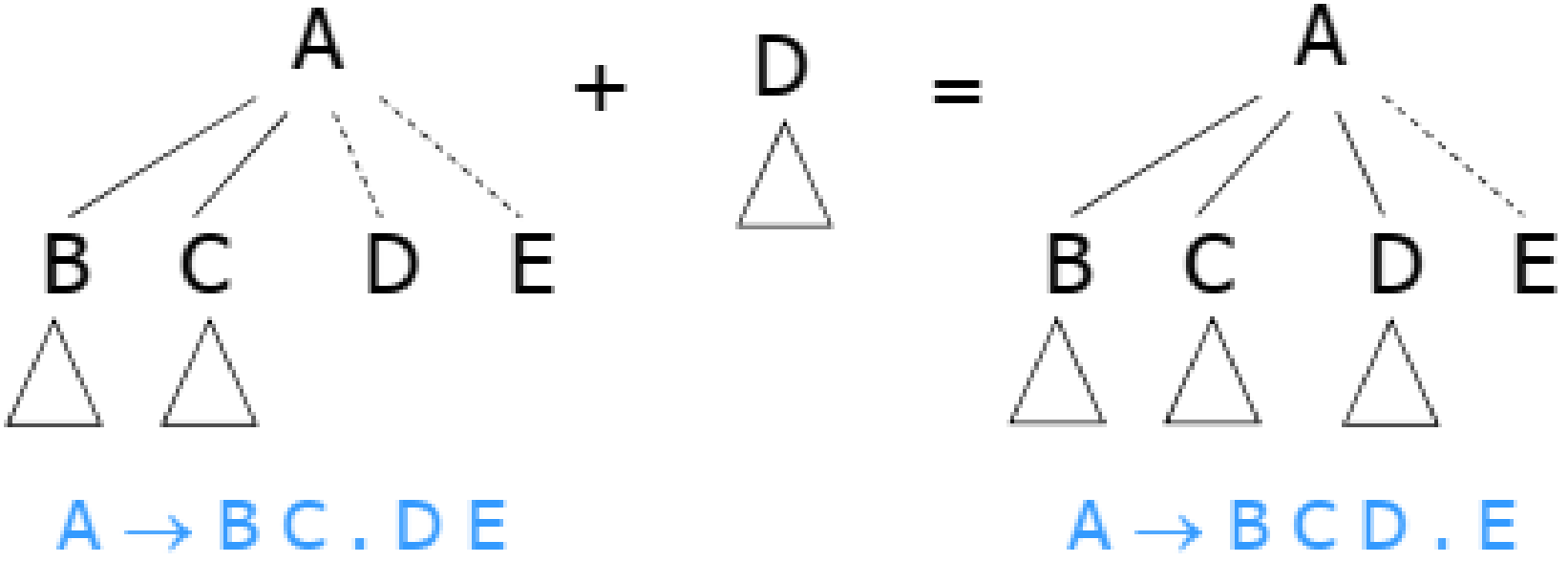


- Ô Cell[i,j] chứa nhãn X nếu
  - Có luật X→YZ;
  - Cell[i,k] chứa nhãn Y và ô Cell[k,j] chứa nhãn Z, với k nằm giữa i và j;

- VD: NP → DT [0,1] NN[1,2]

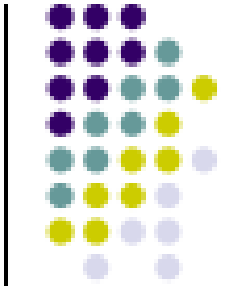
Thuật toán Earley (top-down)

- Tìm các nhãn và các nhãn thiếu (partial constituents) từ đầu vào
  - $A \rightarrow B\ C\ .\ D\ E$ là nhãn thiếu:



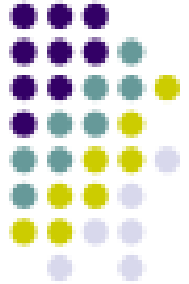
  - Tiến hành dần từ trái sang phải

Ví dụ



ROOT	→ S	NP	→ Papa
S	→ NP VP	N	→ caviar
NP	→ Det N	N	→ spoon
NP	→ NP PP	V	→ ate
VP	→ VP PP	P	→ with
VP	→ V NP	Det	→ the
PP	→ P NP	Det	→ a

Thuật toán Earley



- Thuật toán Earley giống thuật toán đệ qui nói trên, nhưng giải quyết được vấn đề đệ qui trái.
- Sử dụng bảng phân tích giống thuật toán CKY, nhằm lưu lại các thông tin đã tìm thấy → lập trình động **"Dynamic programming."**

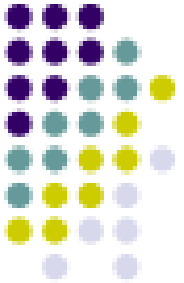
Các thao tác của thuật toán

- Xử lý phần đi sau dấu . theo kiểu đệ qui :
  - Nếu là từ, quét (**scan**) đầu vào để xem có phù hợp không
  - Nếu là ký hiệu không kết thúc, đoán (**predict**) các khả năng để khớp nó (giảm số phép tiên đoán bằng cách nhìn trước k ký hiệu từ đầu vào và chỉ sử dụng các luật phù hợp với k ký hiệu đó)
  - Nếu rỗng, ta đã hoàn thành một thành phần ngữ pháp, gắn (**attach**) nó vào những chỗ liên quan

Ưu điểm

- Thuật toán Earley thực hiện một vài phép lọc *top-down*: bất cứ thành phần nào (state, or triple) được đưa vào tập trạng thái cần tương thích với phần đã được sinh ra ở bên trái. Ví dụ: $S \stackrel{*}{\Rightarrow} w_i$ trong đó w_i là phần của câu đã được duyệt qua

Nhược điểm



- Biểu diễn luật: Explicit representation of rules: wastes time building them.
- Thực hiện phép lọc bên trái nhưng không lọc bên phải

  Phép lọc nhìn trước cho ký hiệu không kết thúc *A:*

  FIRST(A)= {*x*|*A* $\Rightarrow x\delta$}, *x= 1 token*

  v.d., FIRST(S)= who, did, the, etc.