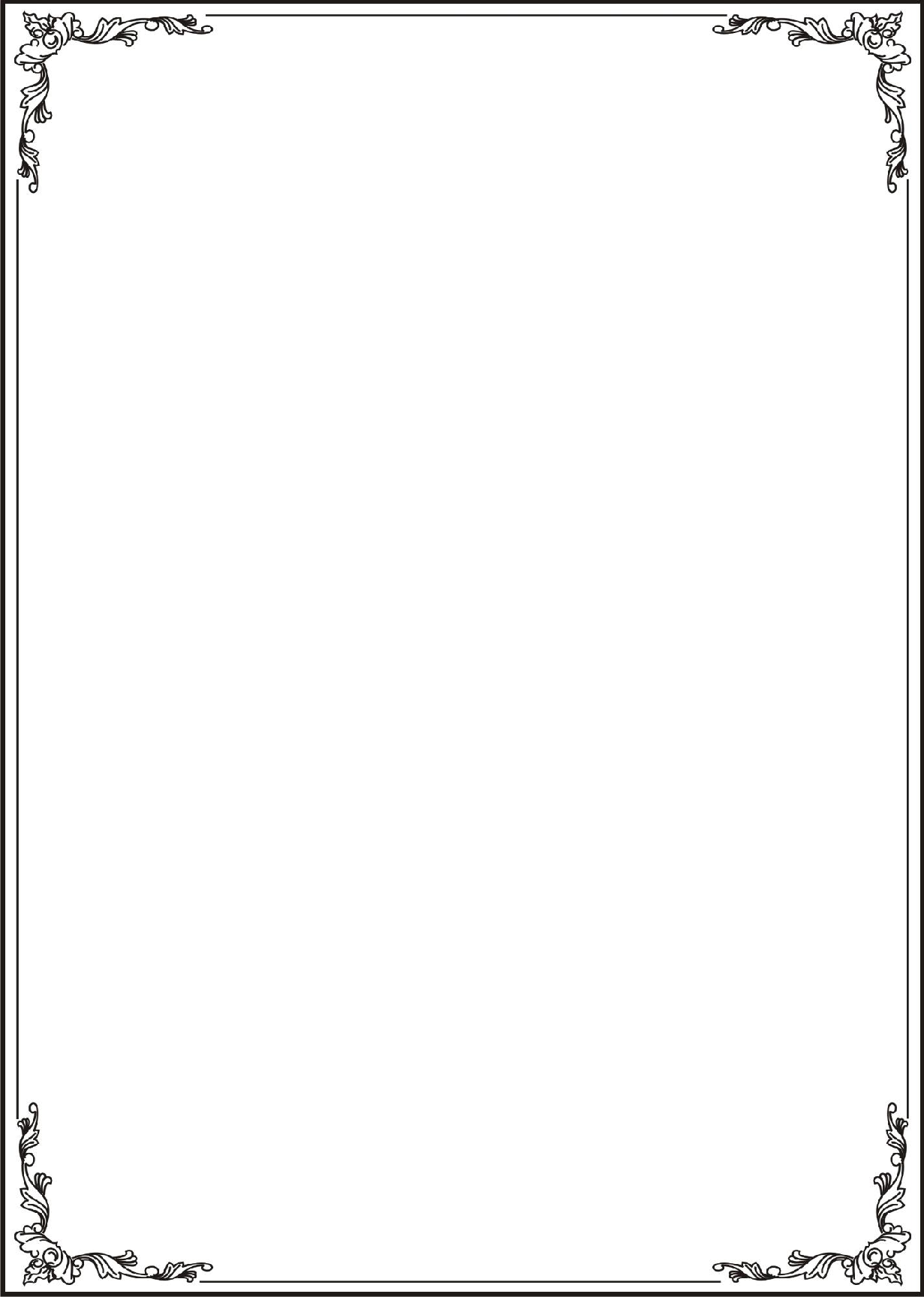
## TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI



----------

**BÁO CÁO BÀI TẬP LẬP TRÌNH LÝ THUYẾT TÍNH TOÁN**

# Bài tập lập trình 4:

# Dạng chuẩn Chomsky và thuật toán CYK

Giảng viên hướng dẫn: **Trần Vĩnh Đức**

Lớp: **108529**

Sinh viên thực hiện: **Vũ Đình Hoàng - 20161728**

Hà Nội 2019

# Dạng chuẩn Chomsky

## Đề bài

* + - Đưa văn phạm về dạng chuẩn Chomsky.
    - Input: File G.jff ở định dạng JFLAP biểu diễn văn phạm G.
    - Output: File Chomsky.jff ở định dạng JFLAP biểu diễn văn phạm G’ ở dạng chuẩn Chomsky tương đương với văn phạm G.

## Thuật toán

* 1. *Xử lý biến bắt đầu*
     + Thêm biến bắt đầu mới để biến bắt đầu không xuất hiện ở vế phải của bất cứ quy tắc nào.
     + Ta thêm biến mới 𝑆0 và quy tắc 𝑆0 → 𝑆, với 𝑆 là biến khởi đầu.
  2. *Loại bỏ quy tắc *
     + Với mỗi quy tắc có dạng 𝐴 → :
       - Loại bỏ quy tắc 𝐴 → 
       - Nếu có quy tắc dạng 𝑅 → 𝑢𝐴𝑣, ta thêm quy tắc 𝑅 → 𝑢𝑣. Thực hiện bước này với mọi sự xuất hiện của A. Ví dụ: Nếu có quy tắc dạng 𝑅 → 𝑢𝐴𝑣𝐴𝑤 thì ta phải thêm quy tắc

𝑆 → 𝑢𝐴𝑣𝑤 | 𝑢𝑣𝐴𝑤 |𝑢𝑣𝑤

* + - * Nếu có quy tắc 𝑅 → 𝐴, ta thêm quy tắc 𝑅 →  trừ khi ở các bước trước ta đã loại bỏ quy tắc R→ .
  1. *Loại bỏ quy tắc đơn*
     + Với mỗi quy tắc dạng 𝐴 → 𝐵, ta thực hiện:
       - Loại bỏ quy tắc 𝐴 → 𝐵
       - Với mỗi quy tắc dạng 𝐵 → u, ta thêm quy tắc 𝐴 → u, trừ khi đây là quy tắc đã bị loại bỏ ở các bước trước đó.
  2. *Chuyển các quy tắc còn lại về dạng đúng*

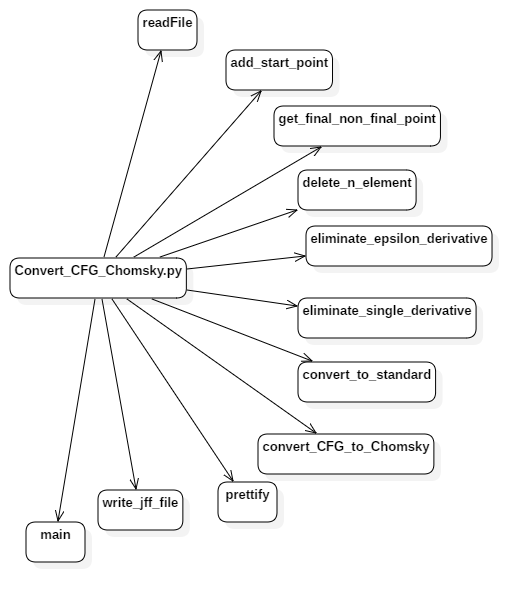
- Thay thế các quy tắc dạng 𝐴 → 𝑢1𝑢2 … 𝑢𝑘 với 𝑘 ≥ 3 𝑣à 𝑢𝑖 ∈ 𝑉 ∪ Σ

bởi các quy tắc 𝐴 → 𝑢1𝐴1, 𝐴1 → 𝑢2𝐴2, 𝐴𝑘−2 → 𝑢𝑘−1𝑢𝑘

* + - Thay thế mọi ký hiệu kết thúc 𝑢𝑖 trong các quy tắc vừa thay thế ở trên bởi biến mới 𝑈𝑖 và thêm quy tắc 𝑈𝑖 → 𝑢𝑖

## Cấu trúc chương trình

Chương trình gồm các hàm:



+ readFile(): đọc file .jff đầu vào và lưu văn phạm thu được vào biến ‘grammar’

Lưu văn phạm dưới dạng list

Mỗi phần tử của list gồm 2 phần tử:

1. Biến không kết thúc
2. Dẫn xuất của nó

+ add\_start\_point(): thêm S0->S vào đầu văn phạm

1. Sao chép grammar sang grammar\_chomsky
2. Thêm S0->S vào đầu văn phạm chomsky
3. Thêm S0 vào tập biến không kết thúc

+ get\_final\_non\_final\_point(): tìm và lưu các điểm kết thúc và không kết thúc vào final\_point và non\_final\_point

**(Ở đây lưu chữ viết hoa là biến không kết thúc, chữ thường là biến kết thúc)**

+ delete\_n\_element(k, n, point, collect, added): xóa các vị trị A mà A-> epsilon trong dẫn xuất

Point: A(A->epsilon)

n: số lần xuất hiện của A trong dẫn xuất collect

k: số A được xóa

1. k>n return
2. xóa 1 A trong dẫn xuất
3. gọi lại delete\_n\_element(k+1,..)

+ eliminate\_epsilon\_derivative(): bước xóa dẫn xuất A->epsilon trong thuật toán

+ eliminate\_single\_derivative(): xóa các dẫn xuất đơn

+ convert\_to\_standard(): đưa dẫn xuất về dạng chuẩn

+ convert\_CFG\_to\_Chomsky(): tổng hợp các hàm ở trên đưa CFG về dạng chuẩn Chomsky

+ prettify(), writte\_jff\_file(): đưa dạng chuẩn Chomsky về dạng file .jff

+ main(): thực hiện chương trình

# Thuật toán CYK

## Đề bài

* + - Viết chương trình mô tả thuật toán CYK
    - Input:
      * File Chomsky.jff mô tả văn phạm G đã ở dạng chuẩn Chomsky
      * Xâu 𝑤 ∈ Σ ∗
    - Output: Thông báo “Yes” nếu xâu 𝑤 được sinh bởi văn phạm G, “No” nếu ngược lại.

## Thuật toán

- Xét xâu vào 𝑤 = 𝑤1𝑤2 … 𝑤𝑛

* + - Ta xây dựng bảng 𝑛 × 𝑛: Phần tử (𝑖, 𝑗) (với𝑖 ≤ 𝑗) của bảng là tập các biến sinh ra xâu

𝑤𝑖𝑤𝑖+1 … 𝑤𝑗 .

- Mã giả: Với xâu vào 𝑤 = 𝑤1𝑤2 … 𝑤𝑛

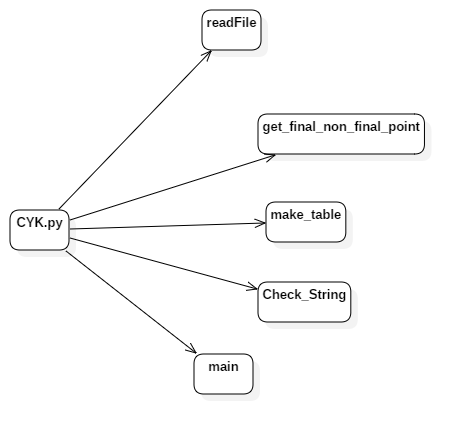
1. Nếu 𝑤 = ** và có luật S → ε, chấp nhận. [[ xử lý ε ]]
2. For i = 1 đến n [[ xử lí xâu con độ dài 1]]
3. For mỗi biến A:
4. Kiểm tra xem có luật 𝐴 → 𝑏 với 𝑤𝑖 = 𝑏
5. Nếu có, đặt A vào 𝑇(𝑖, 𝑖)
6. For l = 2 đến n: [[ l là độ dài xâu con]]
7. For i = 1 đến n – l + 1: 8. j  i + l - 1
8. for k = i đến j – 1:
9. for mỗi sản xuất 𝐴 → 𝐵𝐶

11. Nếu 𝐵 ∈ 𝑇(𝑖, 𝑘) 𝑣à 𝐶 ∈ 𝑇(𝑘 + 1, 𝑗), đặt A vào 𝑇(𝑖, 𝑗).

12. Nếu 𝑆 ∈ 𝑇(1, 𝑛), chấp nhận xâu 𝑤. Ngược lại, bác bỏ.

## Cấu trúc chương trình

Các hàm:



+ readFile(): đọc và lưu trữ văn phạm

+ get\_final\_non\_final\_point()

+ make\_table(): tạo bảng table[n][n]

+ Check\_String(): kiểm tra chuỗi

+ main()