PHENIKAA UNIVERSITY
Realizing your potential

CHƯƠNG TRÌNH DỊCH

Chương 2. Ngôn ngữ hình thức và văn phạm

TS. Phạm Văn Cảnh
Khoa Công nghệ thông tin

Email: canh.phamvan@phenikaa-uni.edu.vn

Nội dung

1. Biểu diễn ngôn ngữ
2. Văn phạm
3. Phân loại văn phạm
4. Văn phạm chính quy
5. Văn phạm phi ngữ cảnh và Automat đẩy xuống
6. Một số văn phạm khác

1. Biểu diễn ngôn ngữ

- Một chương trình đúng → đúng về từ vựng, ngữ pháp và ngữ nghĩa.
- NNHT nghiên cứu:
  - Bản chất ngôn ngữ
  - Ngôn ngữ nói chung -> siêu ngôn ngữ
- Học: nên có sự liên hệ với các ngôn ngữ đã biết như tiếng Việt, Anh, C, Pascal

1. Biểu diễn ngôn ngữ

- Kí hiệu (symbol): khái niệm cơ sở để xây dựng ngôn ngữ, không thể định nghĩa một cách hình thức
  - Các chữ số, các chữ cái, các dấu kí hiệu,...
- Bộ chữ (alphabet): tập hợp hữu hạn các kí hiệu.
  - Bộ chữ cái tiếng Việt (a, ă, â,..., x, y, A, Ă,..., Y).
- Chuỗi (string): dãy hữu hạn các ký hiệu thuộc cùng một bộ ký hiệu nào đó.
  - "2016" là chuỗi gồm 4 ký hiệu thuộc bộ ký hiệu chữ số
  - "2016" còn gọi là chuỗi ***sinh bởi*** bộ ký hiệu chữ số.
  - Chuỗi rỗng (ký hiệu là ϵ).

1. Biểu diễn ngôn ngữ

- Ngôn ngữ (language): tập hợp các chuỗi
  - Ngôn ngữ tiếng Việt là tập một số các chuỗi sinh bởi bộ chữ tiếng Việt.
  - Có những chuỗi sinh từ bộ chữ tiếng Việt những không thuộc ngôn ngữ tiếng Việt (chẳng hạn chuỗi "lẫnh").
  - Chuỗi thuộc ngôn ngữ tiếng Việt đều sinh bởi bộ chữ tiếng Việt.
- Tổng quát:
  - Cho bộ chữ Σ.
  - Σ^* là tập tất cả các chuỗi sinh ra từ Σ (gồm cả ϵ).
  - Ngôn ngữ L sinh bởi Σ là một tập con của Σ^* .

1. Biểu diễn ngôn ngữ

- Định nghĩa ngôn ngữ L như một tập con của Σ^* là quá trừu tượng và không có ý nghĩa thực tế, khó sử dụng với các thuật toán.
- Cần có phương pháp biểu diễn ngôn ngữ có tính hình thức hơn.
- Nếu kích cỡ ngôn ngữ L đủ nhỏ, ta chỉ việc liệt kê mọi chuỗi trong L.
  - Ví dụ: từ điển Anh-Anh, liệt kê mọi từ tiếng Anh, những từ nằm ngoài từ điển coi như không phải tiếng Anh.

1. Biểu diễn ngôn ngữ

- Bài toán biểu diễn ngôn ngữ:
  1. Ngôn ngữ L sinh bởi Σ, cho một chuỗi w thuộc Σ^*, hỏi w có thuộc L hay không?
  2. Nếu w thuộc L, thì w được tạo ra từ các quy tắc nào?
- Bài toán số 2 có sự liên hệ với việc phân tích văn phạm trong CCD
- Hai bài toán trên không giải được trong trường hợp tổng quát, chỉ giải được trong một số tình huống hạn chế, đó chính là lý do tại sao các văn phạm của các ngôn ngữ lập trình thường rất chặt chẽ.

1. Biểu diễn ngôn ngữ

- Đối với ngôn ngữ hữu hạn: có thể liệt kê được các xâu của ngôn ngữ đó.
  - Vd: L1={a,ba, aaba, bbbb}: ngôn ngữ hữu hạn.
- Đối với ngôn ngữ vô hạn (ngôn ngữ tự nhiên, ngôn ngữ lập trình): phải tìm cách *biểu diễn hữu hạn cho một ngôn ngữ vô hạn.*
  - Ví dụ: L2= { a^i| i là các số nguyên tố}

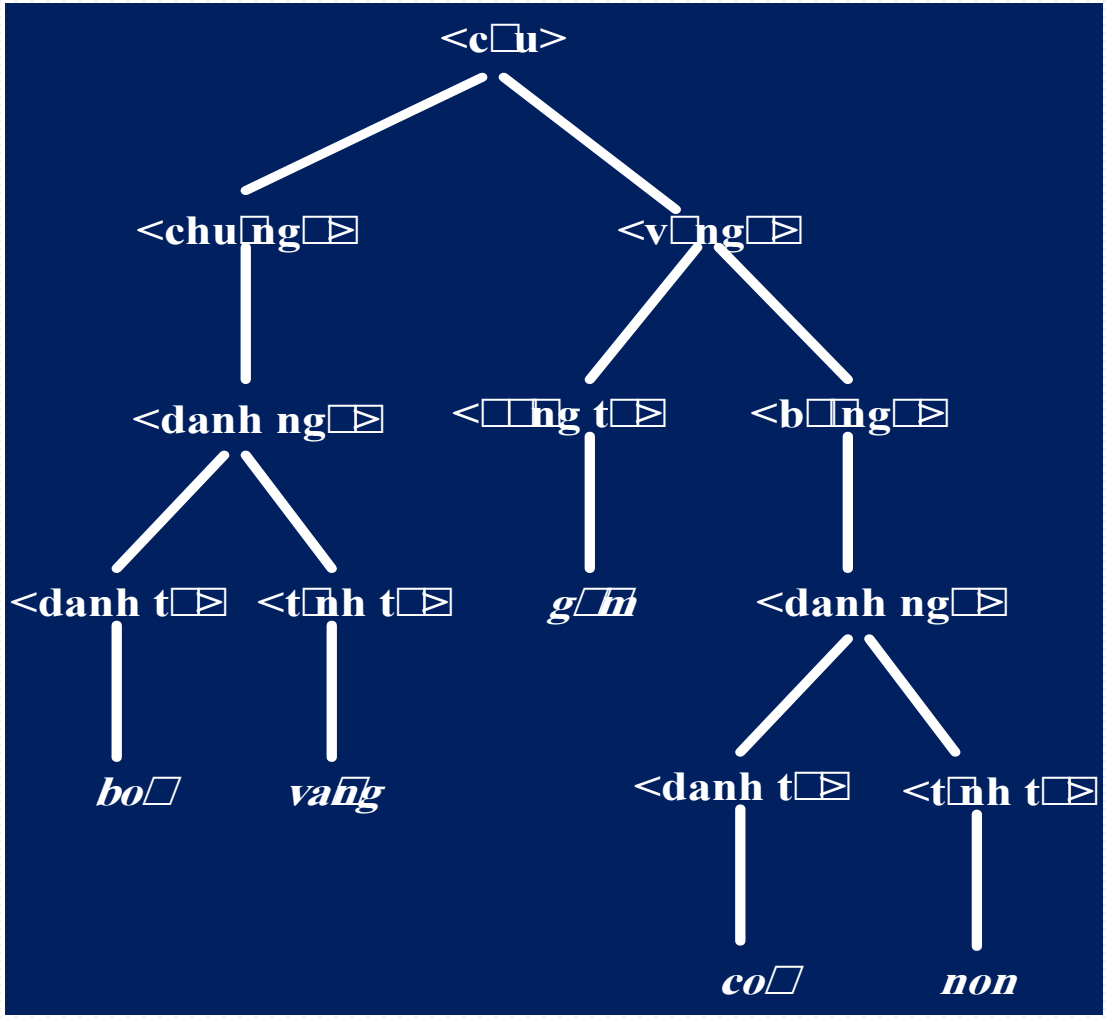
    L3 = { w∈{a,b}*|số lượng a=số lượng b}

1. Biểu diễn ngôn ngữ

- Biểu diễn L bằng văn phạm chỉ là một trong nhiều phương pháp biểu diễn ngôn ngữ, nhưng phương pháp này được ưa thích do có lợi thế:
  - Tính chặt chẽ, vạn năng.
  - Gần gũi với máy stack (kiến trúc máy tính nguyên thủy).
  - Là một công cụ mô tả hữu hạn ngôn ngữ rất hiệu quả
  - Là công cụ có định nghĩa toán học chặt chẽ, đã được nghiên cứu kỹ
- Văn phạm = ngữ pháp = cú pháp

2. Văn phạm

- Ví dụ:
  - Quá trình phân tích câu: Bò vàng gặm cỏ non
  - Bắt đầu từ đỉnh là <câu>
  - Bò, vàng, gặm, cỏ, non: các lá của cây cú pháp
  - Các lá: kí hiệu kết thúc (terminal)
  - Các nút trung gian: kí hiệu không kết thúc (non-terminal)



2. Văn phạm

❑ **Định nghĩa** Một văn phạm là một hệ thống $G = (\Sigma, \Delta, P, S)$ trong đó:

- Σ là tập hữu hạn các ký hiệu, gọi là ký hiệu kết thúc (terminal)
- Δ là tập hữu hạn các ký hiệu, gọi là ký hiệu không kết thúc (nonterminal)
- $S \in \Delta$ gọi là ký hiệu khởi đầu.
- P là tập hữu hạn các cặp xâu (α,β) được gọi là các dẫn xuất hay luật cú pháp $(\alpha \rightarrow \beta)$
  - Chuỗi α có ít nhất một ký hiệu không kết thúc.

2. Văn phạm

Ví dụ:

- Σ = {"bò", "vàng", "gặm", "cỏ", "non"}
- Δ = {<câu>,<chủ ngữ>, <vị ngữ>, <danh ngữ>, <động từ>, <bổ ngữ>, <danh từ>, <tính từ> }
- S =<câu>
- P=

<câu>→<chủ ngữ><vị ngữ>;

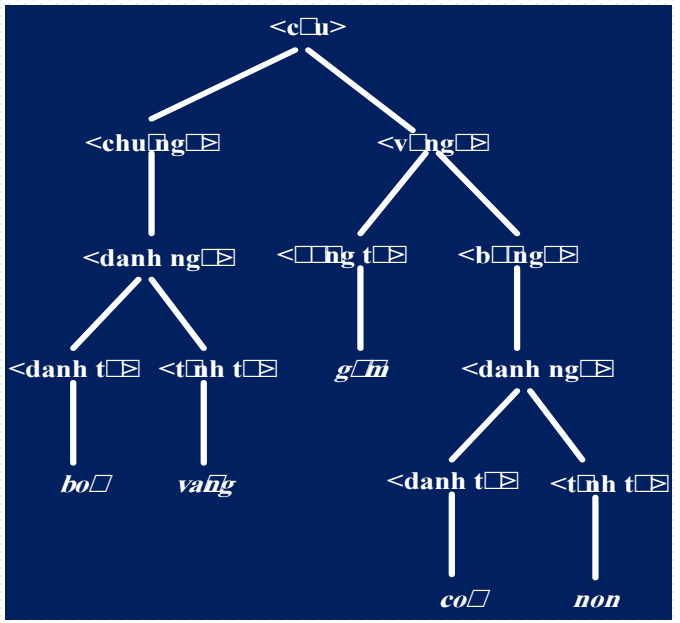
<chủ ngữ>→<danh ngữ>;

< danh ngữ >→<danh từ><tính từ>;

<vị ngữ>→<động từ> <bổ ngữ>

<bổ ngữ>→<danh ngữ>;

<danh từ>→"bò"|"cỏ"; <tính từ>→"vàng"|"non"; <động từ>→"gặm";



2. Văn phạm

Qui ước:

- Dùng các chữ in hoa A, B, C, D, E... hoặc cụm từ trong cặp ngoặc nhọn (như <chủ ngữ>): ký hiệu không kết thúc;
- Dùng các chữ thường a, b, c, d, e... và các con số, các phép toán +, -, *, /, cặp ngoặc đơn để trỏ các ký hiệu kết thúc. Trong một số trường hợp dùng qui ước là một từ được in đậm (như số và chữ) hoặc cụm từ trong cặp ngoặc kép (như "bò");
- Dùng các chữ in hoa X, Y, Z để trỏ các ký hiệu có thể là kết thúc hoặc không kết thúc;
- Dùng các chữ thường u, v, w, x, y, z để trỏ các xâu ký hiệu cuối;
- Dùng các chữ thường Hy lạp α, β, χ để trỏ các xâu gồm các biến và ký hiệu cuối;
- Nếu có các sản xuất cùng vế trái $A \to \alpha$ và $A \to \beta$ thì ta viết gộp lại cho gọn thành $A \to \alpha \mid \beta$. Các sản xuất có cùng một kí hiệu không kết thúc vế trái có thể gọi chung bằng tên kí hiệu vế trái. Ví dụ, sản xuất-A.

2. Văn phạm

❑Qui ước: (tiếp)

- $V = (\Sigma \cup \Delta)$ là một xâu (có thể rỗng) bao gồm cả ký hiệu không kết thúc và ký hiệu kết thúc;
- V^* là tập tất cả các xâu V có thể có;
- V^+ cũng như vậy trừ xâu rỗng;
- | | là ký hiệu độ dài xâu (ví dụ | α | là độ dài của xâu α);
- Ký hiệu ε là một kí hiệu đặc biệt, chỉ xâu rỗng hoặc kí hiệu rỗng

2. Văn phạm

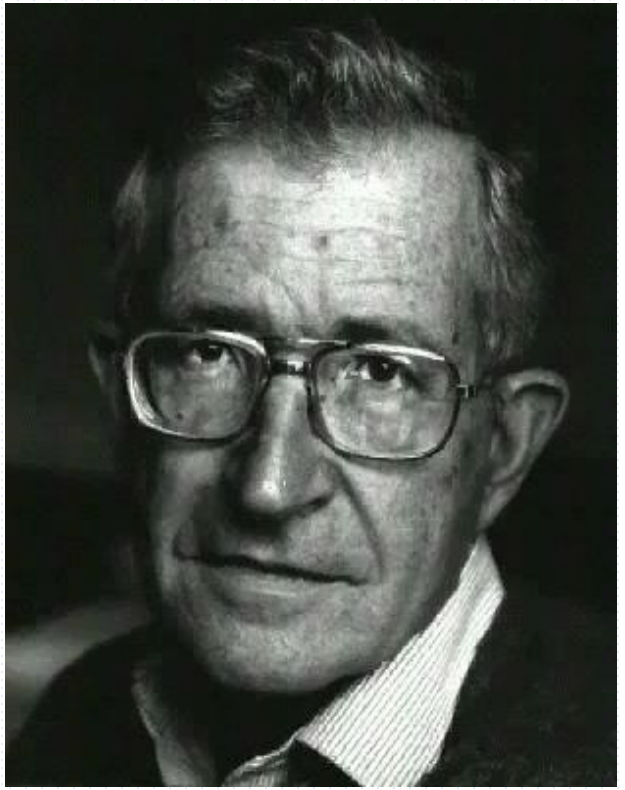
- Suy dẫn (sinh): Cho văn phạm $G = (\Sigma, \Delta, P, S)$
  - Chuỗi $\alpha\delta\gamma$ gọi là suy dẫn trực tiếp từ $\alpha\beta\gamma$ khi áp dụng luật $\beta \to \delta$, ký hiệu $\alpha\beta\gamma \to \alpha\delta\gamma$.
  - Việc áp dụng luật là việc thay thế chuỗi con β trong chuỗi ban đầu bằng vế phải δ của luật.
  - Nếu từ A áp dụng liên tiếp một số suy dẫn được B thì ta gọi B là suy dẫn gián tiếp từ A, kí hiệu $A \Rightarrow^* B$.
  - Suy dẫn k bước.
-

2. Văn phạm

- Định nghĩa ngôn ngữ: Ngôn ngữ của văn phạm G, ký hiệu là $L(G)$ là tập hợp các ký hiệu kết thúc,
  - $L(G) = \{w | w \in \Sigma^* \text{ và } w \Rightarrow^* S\}$.
  - $L(G) = \{ w | w \in \Sigma^* \text{ và } S \Rightarrow w \}$.
- Định nghĩa: Hai văn phạm G_1 và G_2 (sản sinh hoặc đoán nhận) là tương đương khi và chỉ khi $L(G_1) = L(G_2)$.

3. Phân loại văn phạm

- Phân loại văn phạm theo Chomsky.
- Noam Chomsky (1928 – nay) chia văn phạm thành các lớp xét theo các ràng buộc của luật văn phạm
  - Lớp 0: unrestricted grammars (văn phạm tự do)
  - Lớp 1: context-sensitive grammars (văn phạm cảm ngữ cảnh)
  - Lớp 2: context-free grammars (văn phạm phi ngữ cảnh)
  - Lớp 3: regular grammars (văn phạm chính quy)
  - Mô hình này gọi là phân loại chomsky
  - Ngôn ngữ sinh bởi các lớp văn phạm thấp hơn bao gồm hoàn toàn ngôn ngữ sinh bởi các lớp cao hơn



Chomsky, nhà triết học, nhà khoa học nhận thức, nhà logic học, nhà bình luận chính trị, và nhà hoạt động người Mỹ. Ông hiện là giáo sư danh dự hồi hưu (professor emeritus) ở Viện Công nghệ Massachusetts.

3. Phân loại văn phạm

- **Lớp 0**, văn phạm ngữ cấu (phrase - structure) nếu sản xuất có dạng: $\alpha \rightarrow \beta$ **trong đó** $\alpha \in V^+,\ \beta \in V^*$.
  - Không có ràng buộc gì về luật sinh.
  - Tương đương với lớp các ngôn ngữ loại đệ quy đếm được (recursively enumerable languages).
  - Được đoán nhận bởi máy Turing.

3. Phân loại văn phạm

- **Lớp 1**, văn phạm cảm ngữ cảnh (context - sensitive) nếu sản xuất có dạng: $\alpha \rightarrow \beta$ thỏa mãn điều kiện $|\alpha| \leq |\beta|$.
  - Không có ràng buộc về luật sinh
  - Tương đương với lớp các ngôn ngữ cảm ngữ cảnh (context-sensitive languages).
  - Được đoán nhận bởi automat tuyến tính giới nội (LBA –linear bounded automaton).

3. Phân loại văn phạm

- **Lớp 2**, văn phạm phi ngữ cảnh (context free - viết tắt là VPPNC) nếu sản xuất có dạng: $A \rightarrow \alpha$ trong đó $A \in \Delta$, $\alpha \in V^*$
  - Tương đương với lớp các ngôn ngữ phi ngữ cảnh (context-free languages)
  - Được đoán nhận bởi automat đẩy xuống (push down automaton).

3. Phân loại văn phạm

- **Lớp 3**, văn phạm chính quy (regular - viết tắt là VPCQ) nếu sản xuất có dạng:
  - $A \to a, A \to Ba$ trong đó $A, B \in \Delta, a \in \Sigma$.
  - $A \to a, A \to aB$ với $A, B \in \Delta, a \in \Sigma$.
  - Sinh ra các ngôn ngữ chính quy (regular languages)
  - Đoán nhận bởi automat hữu hạn (finite state automaton)

3. Phân loại văn phạm

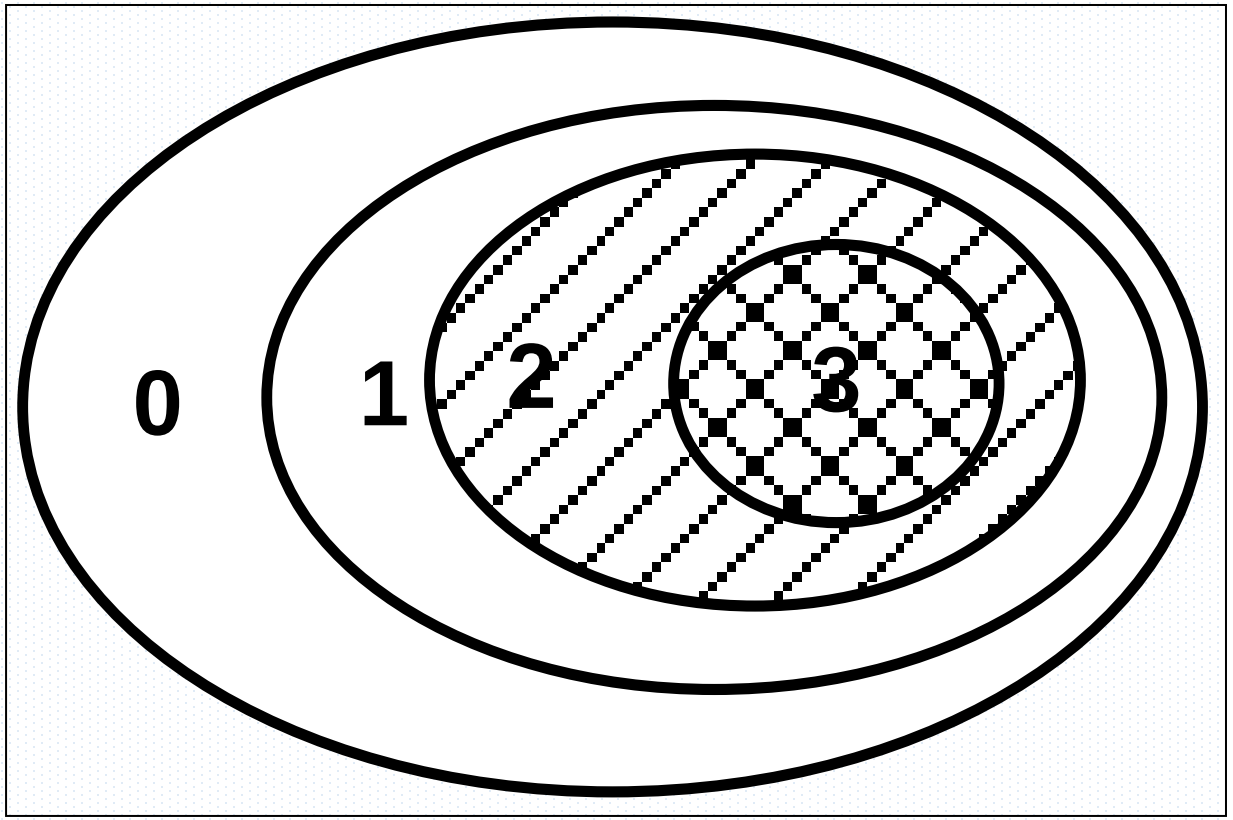
Ví dụ:

- Cho VPPNC $G = (\Sigma, \Delta, P, S)$ với $\Sigma = \{a,b\}$, $\Delta = \{S,A,B\}$ và P có các sản xuất như sau:

$$S \rightarrow aB \mid bA \;,\; A \rightarrow aS \mid bAA \mid a \;,\; B \rightarrow bS \mid aBB \mid$$

3. Phân loại văn phạm

- Ngôn ngữ loại 0 được đoán nhận bởi một máy Turing;
- Ngôn ngữ loại 1 (cảm ngữ cảnh) được đoán nhận bởi một ôtômát tuyến tính giới nội (sai khác xâu rỗng);
- Ngôn ngữ loại 2 (phi ngữ cảnh - viết tắt là NNPNC) đoán nhận bởi một ôtômát đẩy xuống (không đơn định);
- Ngôn ngữ loại 3 (chính qui - viết tắt là NNCQ) được đoán nhận bởi một ôtômát hữu hạn (sai khác xâu rỗng).



3. Phân loại văn phạm

- Ngôn ngữ loại 0 được đoán nhận bởi một máy Turing;
- Ngôn ngữ loại 1 (cảm ngữ cảnh) được đoán nhận bởi một ôtômát tuyến tính giới nội (sai khác xâu rỗng);
- Ngôn ngữ loại 2 (phi ngữ cảnh - viết tắt là NNPNC) đoán nhận bởi một ôtômát đẩy xuống (không đơn định);
- Ngôn ngữ loại 3 (chính qui - viết tắt là NNCQ) được đoán nhận bởi một ôtômát hữu hạn (sai khác xâu rỗng).

4. Văn phạm chính quy

PHENIKAA UNIVERSITY
Realizing your potential

- Văn phạm chính quy giới hạn các luật có dạng:
  - $A \rightarrow a$, $A \rightarrow Ba$ trong đó $A, B \in \Delta$, $a \in \Sigma$.
  - $A \rightarrow a$, $A \rightarrow aB$ với $A, B \in \Delta$, $a \in \Sigma$.
- Người ta ít dụng luật văn phạm mà sử dụng biểu thức chính quy (regular expression).
  - Biểu thức chính quy và văn phạm chính quy là hoàn toàn tương đương
  - Biểu thức chính quy đơn giản, dễ hiểu hơn
  - Biểu thức chính quy sử dụng bộ kí pháp sau:
    - Kí hiệu | có nghĩa là hoặc (or)
    - Kí hiệu ( ) để nhóm các thành phần
    - Kí hiệu * có nghĩa là lặp lại không hoặc nhiều lần

4. Văn phạm chính quy

- Biểu thức chính quy có nhiều biến thể cho phép viết các kí pháp phong phú và tiện lợi hơn.
- BTCQ sử dụng rất nhiều khi phân tích từ vựng.
- Ví dụ: quy cách khai báo tên riêng trong C/C++ .
  - Văn phạm chính quy:

    <tên riêng> → chữ <phần sau>

    <phần sau>→ ϵ

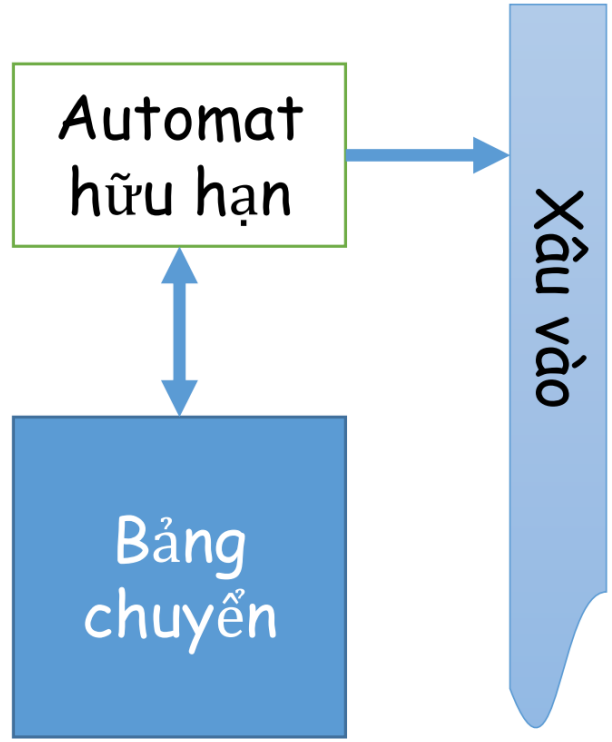
    <phần sau> → chữ <phần sau>

    <phần sau> → số <phần sau>
  - Biểu thức chính quy:

    <tên riêng> → chữ <(chữ|số)*

4. Văn phạm chính quy

- Automat hữu hạn dùng để đoán nhận lớp ngôn ngữ chính quy.
- Cấu trúc của automat hữu hạn gồm:
  - Bảng chuyển
  - Đầu đọc
  - Xâu vào
- Hoạt động của automat:
  - Bắt đầu từ trạng thái xuất phát.
  - Đọc dữ liệu từ xâu vào.
  - Quan sát bảng chuyển để biết sẽ chuyển sang trạng thái nào.
  - Dừng khi kết thúc xâu vào và trả về trạng thái đoán nhận.



4. Văn phạm chính quy

- Ôtômát hữu hạn đơn định (Deterministic Finite Automata - DFA):
  - Nếu ôtômát đang ở một trạng thái nào đó, đọc một ký hiệu vào và chỉ có duy nhất một chuyển đổi.
- Ôtômát này gọi là ôtômát hữu hạn không đơn định (Nodeterministic Finite Automata - NFA):
  - Nếu từ một trạng thái và một ký hiệu vào, có thể có một hoặc nhiều khả năng đổi sang các trạng thái khác.
- DFA có thể coi là một trường hợp đặc biệt của NFA. DFA tương đương NFA về khả năng đoán nhận ngôn ngữ.

5. VPPNC và Automat đẩy xuống

- Văn phạm phi ngữ cảnh giới hạn các luật sinh phải có dạng

  $A \to \alpha$ trong đó $A \in \Delta$ (vế trái của luật chỉ có 1 kí hiệu).

- Văn phạm phi ngữ cảnh sử dụng trong việc biểu diễn và phân tích cú pháp.
- Cú pháp các ngôn ngữ lập trình thường sử dụng **BNF (Backus-Naur Form)** để biểu diễn cú pháp, đây chỉ là cách viết dễ đọc hơn và hoàn toàn tương đương với VPPNC

  <toán hạng> = <tên> | <số> | "(" <biểu thức> ")"

5. VPPNC và Automat đẩy xuống

PHENIKAA UNIVERSITY
Realizing your potential

- Quy ước của BNF (Backus-Naur Form):
  - Các ký hiệu trung gian viết thành một chuỗi đặt trong cặp < >
  - Các ký hiệu kết thúc, các dấu ký hiệu viết trong cặp " "
  - Ký hiệu | thể hiện sự lựa chọn.
  - Ký hiệu = thể hiện ký hiệu ở vế trái được giải thích bởi vế phải
- Bản thân cách viết BNF cũng có một vài biến thể, ở đây sẽ không đề cập đến để tránh nhập nhằng không cần thiết.

5. VPPNC và Automat đẩy xuống

PHENIKAA UNIVERSITY
Realizing your potential

- Ví dụ:

```
// Ví dụ về BNF của một biểu thức thông dụng
// Kí hiệu ::= thay cho kí hiệu giải thích
<expr> ::= <term> "+" <expr>
| <term>
<term> ::= <factor> "*" <term>
| <factor>
<factor> ::= "(" <expr> ")"
| <const>
<const> ::= integer
```

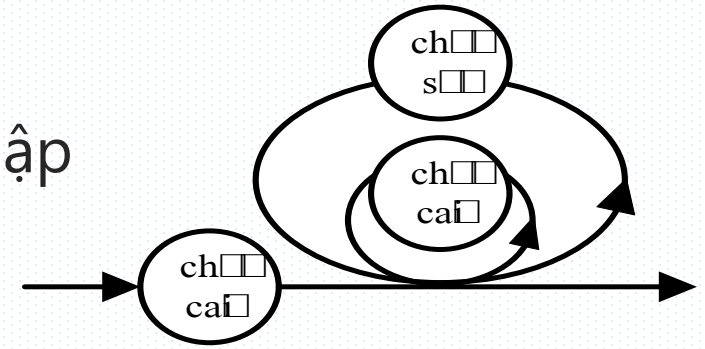
5. VPPNC và Automat đẩy xuống

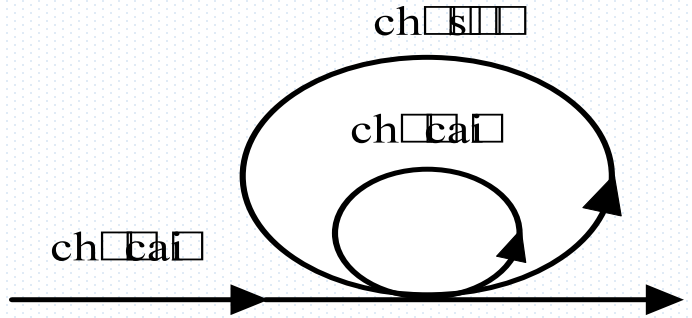
- Có thị dùng đồ thị để biểu diễn dạng BNF
  - Ví dụ: biểu diễn khái niệm Tên trong ngôn ngữ lập trình Pascal

Các trạng thái

Trạng thái kết thúc

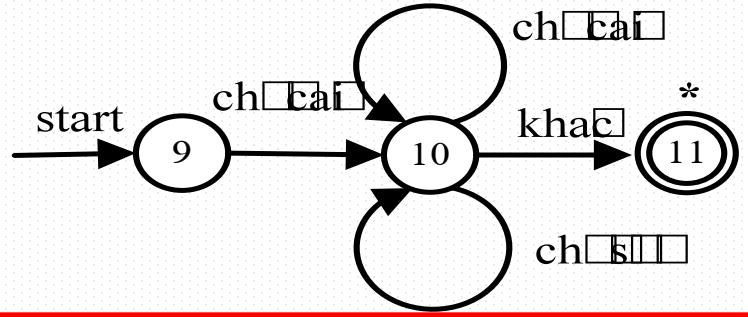
Các cạnh





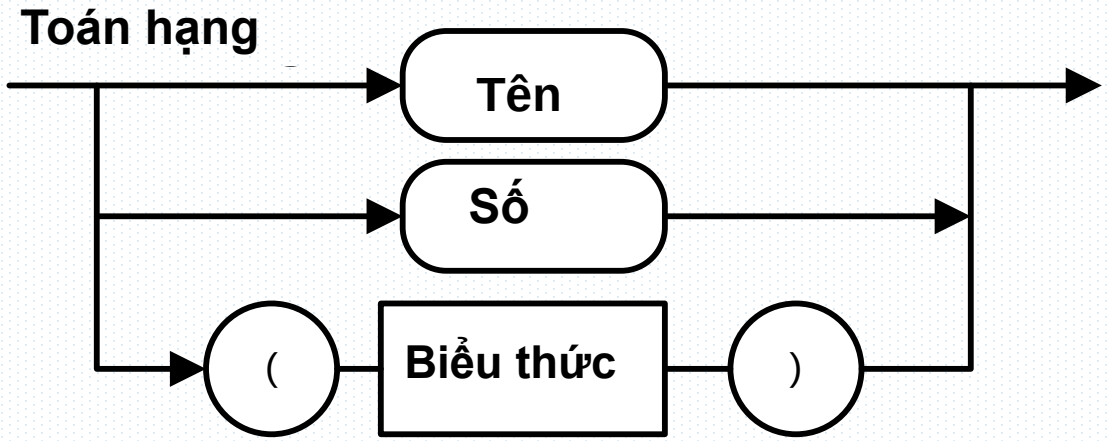
Chữ cái, chữ số: Các nhãn, chỉ mũi tên đi từ trạng thái đó sẽ sinh ra kí hiệu nào

Nhãn *khác*: Mọi kí hiệu khác kí hiệu đã có trên trạng thái.
Nhãn *start*: bắt đầu



5. VPPNC và Automat đẩy xuống

- Có thị dùng đồ thị để biểu diễn dạng NBF
  - Ví dụ: biểu diễn khái niệm toán hạng.

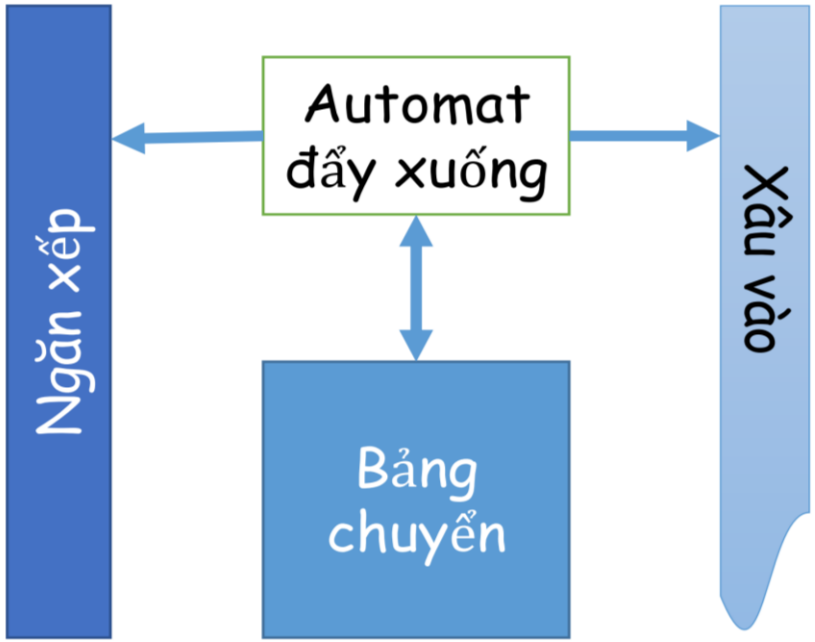


Định nghĩa toán hạng

Cũng có thể kết hợp nhiều đồ thị chuyển với nhau thành một đồ thị chuyển lớn.

5. VPPNC và Automat đẩy xuống

- Automat đẩy xuống chuyên dùng để đoán nhận lớp ngôn ngữ phi ngữ cảnh.
- Cấu trúc của automat gồm:
  - Bảng chuyển.
  - Đầu đọc.
  - Ngăn xếp.
  - Xâu vào.
- Hoạt động của automat:
  - Bắt đầu từ trạng thái xuất phát.
  - Đọc dữ liệu từ xâu vào.
  - Quan sát bảng chuyển và ngăn xếp để biết sẽ xử lý thế nào.
  - Dừng khi kết thúc xâu vào hoặc ở trạng thái kết thúc.



5. VPPNC và Automat đẩy xuống

- Bài toán đoán nhận chuỗi w có thuộc lớp L(G) hay không có nhiều cách tiếp cận
- Những cách tiếp cận tổng quát:
  - Phân tích top-down
  - Phân tích bottom-up
  - Phân tích CYK
  - Phân tích Earley
- Những cách tiếp cận 2 bước: cố gắng sinh automat đẩy xuống để dùng automat này đoán nhận chuỗi
  - Phân tích LL (top-down)
  - Phân tích LR (bottom-up)

6. Một số văn phạm khác

PHENIKAA UNIVERSITY
Realizing your potential

- Văn phạm G gọi là văn phạm có đệ quy trái nếu chứa các luật dạng

$$A \rightarrow A\alpha \mid \beta$$

  - Kí hiệu trung gian A suy dẫn ra chính nó đôi lúc gây ra khó khăn trong việc sinh cây phân tích (đối với một số thuật toán, nhất là những thuật toán ưu tiên chiều sâu)
  - Trường hợp như trên, khi A suy dẫn trực tiếp ra chính nó được gọi là đệ quy trái trực tiếp; nếu A suy dẫn ra chính nó sau một số phép suy dẫn khác thì được gọi là đệ quy trái gián tiếp

6. Một số văn phạm khác

- Văn phạm có đệ quy trái (cả trực tiếp và gián tiếp) có thể được sửa đổi để không còn xuất hiện đệ quy trái nữa bằng cách thêm vào các kí hiệu trung gian mới và sửa đổi các luật văn phạm:
  - Ví dụ với luật trên: $A \rightarrow A\alpha \mid \beta$
  - Ta thêm kí hiệu trung gian mới R
  - Và sửa luật thành:

$$A \rightarrow \beta\ R$$
$$R \rightarrow \alpha\ R \mid \varepsilon$$

6. Một số văn phạm khác

- Văn phạm đơn nghĩa: Một văn phạm bị gọi là nhập nhằng (ambiguity) nếu tồn tại chuỗi w có ít nhất hai cây phân tích tạo ra nó.
- Ngược lại, văn phạm không có nhập nhằng là văn phạm đơn nghĩa:
  - Tính đơn nghĩa đảm bảo cho ngôn ngữ sinh bởi văn phạm chỉ có một cách hiểu duy nhất (không thể hiểu sai)
  - Xây dựng văn phạm chặt chẽ (đơn nghĩa) là cần thiết nhưng cũng làm cho bộ luật văn phạm trở nên phức tạp đáng kể.
  - Bài toán xác định xem văn phạm G có đơn nghĩa hay không là bài toán khó

6. Một số văn phạm khác

- Xét văn phạm sau: $S \rightarrow S + S \mid S * S \mid (S) \mid a$
  - Xây dựng cây phân tích của chuỗi: $a + a * a$
  - Ta có 2 cây phân tích, dẫn đến việc có 2 cách hiểu ngữ nghĩa của chuỗi (nếu thay a bằng số thì có 2 cách tính giá trị của chuỗi)