CHƯƠNG TRÌNH DỊCH

Bài 2: Cơ sở về ngôn ngữ và văn phạm

Nội dung

1. Ngôn ngữ và biểu diễn ngôn ngữ
2. Các lớp văn phạm (phân loại chomsky)
3. Văn phạm chính quy và automat hữu hạn
4. Văn phạm phi ngữ cảnh và automat đẩy xuống
5. Văn phạm có đệ quy trái
6. Văn phạm đơn nghĩa
7. Bài tập

Phần 1

Ngôn ngữ và biểu diễn ngôn ngữ

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 (không có kí hiệu) được kí hiệu là **ε**

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^*

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
  - Trong thực tế: từ điển Anh-Anh, liệt kê mọi từ tiếng Anh, những từ nằm ngoài từ điển coi như không phải tiếng Anh

Biểu diễn ngôn ngữ

- Nếu ngôn ngữ L quá lớn hoặc vô hạn (chẳng hạn như tập số tự nhiên), không thể liệt kê bởi từ điển, lúc này ta cần hình thức hóa các chuỗi w thuộc L bằng cách chỉ ra các đặc điểm của các chuỗi đó
  - Chẳng hạn: $L = \{ w \in \Sigma^* \mid \text{số ký hiệu 0} = \text{số ký hiệu 1} \}$
- 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)

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 chương trình dịch
- 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

Văn phạm

- Văn phạm G là một hệ thống (Σ, Δ, P, S) trong đó:
  - Σ là tập hữu hạn các ký hiệu kết thúc (terminal)
  - Δ là tập hữu hạn các ký hiệu không kết thúc (nonterminal)
    - Còn gọi là ký hiệu trung gian hay biến
    - $\Sigma \cap \Delta = \varnothing$
  - $S \in \Delta$ gọi là ký hiệu khởi đầu (initial)
  - P là tập hữu hạn các cặp chuỗi (α, β) được gọi luật văn phạm (syntax rule) hay luật sinh
    - Thường được viết là $\alpha \rightarrow \beta$
    - Chuỗi α phải có ít nhất một ký hiệu không kết thúc

Ngôn ngữ sinh bởi văn phạm

- Suy dẫn (sinh):
  - Chuỗi $\alpha\delta\gamma$ gọi là suy dẫn trực tiếp từ $\alpha\beta\gamma$ khi áp dụng luật $\beta \rightarrow \delta$, ký hiệu $\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$
    - 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
- Ngôn ngữ của văn phạm G là tập hợp các chuỗi chỉ chứa kí hiệu kết thúc được sinh ra (trực tiếp hoặc gián tiếp) từ S, kí hiệu là L(G)
  - L(G) = { w | w $\in \Sigma^*$ và S $\Rightarrow^*$ w }

Phần 2

Các lớp văn phạm (phân loại chomsky)

Các lớp văn phạm

- 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

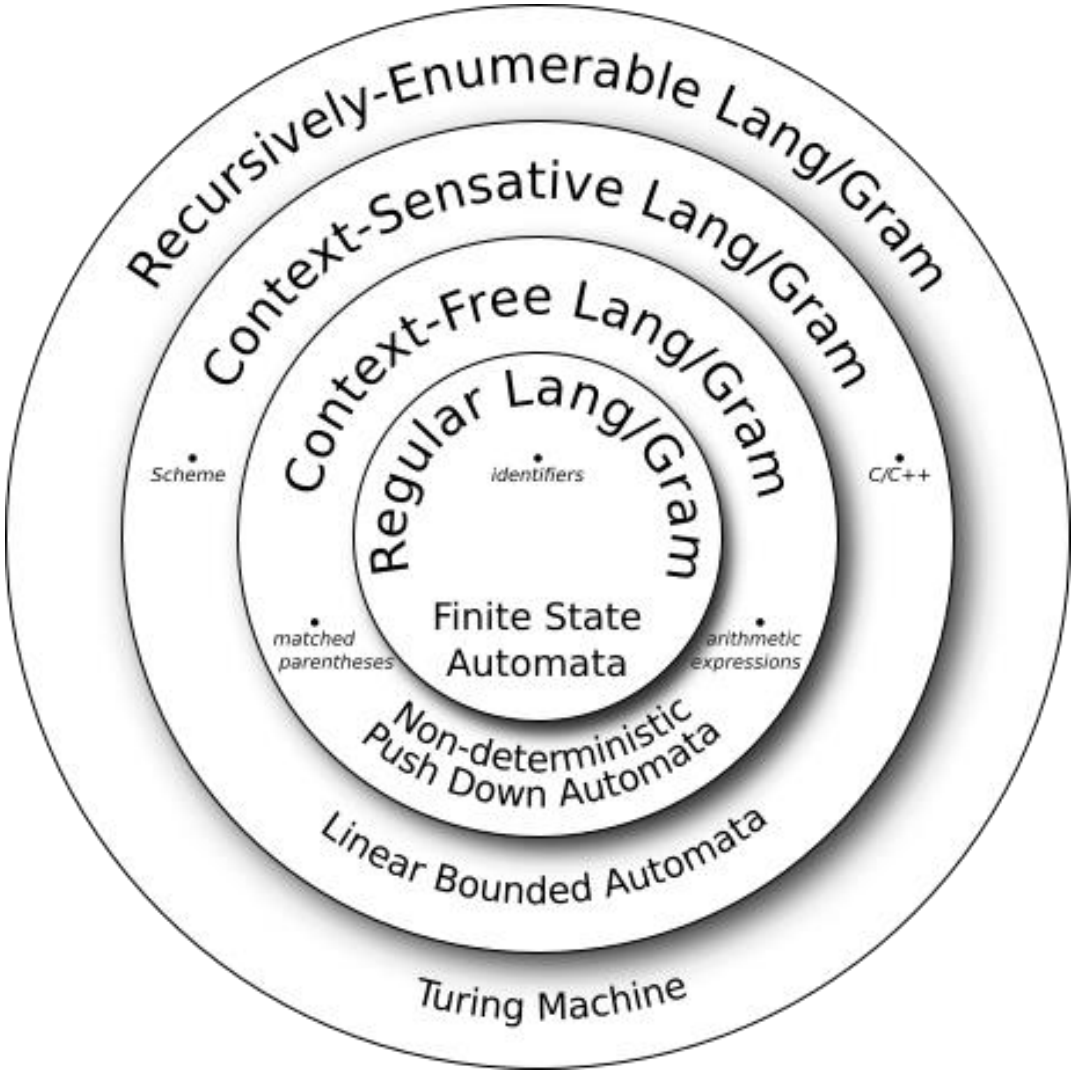
Các lớp văn phạm

- Lớp 0 – văn phạm tự do:
  - 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
- Lớp 1 – văn phạm cảm ngữ cảnh:
  - Các luật sinh $\alpha \rightarrow \beta$ phải thỏa mãn điều kiện $|\alpha| \leq |\beta|$
  - 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)

Các lớp văn phạm

- Lớp 2 – văn phạm phi ngữ cảnh:
  - Các luật sinh phải có dạng $A \rightarrow \alpha$ trong đó $A \in \Delta$
  - Tương đương với lớp các ngôn ngữ phi ngữ cảnh (context-free languages)
  - Được đoán nhận bởi automat đẩy xuống (pushdown automaton)
- Lớp 3 – văn phạm chính quy:
  - Các luật sinh chỉ có thể ở 1 trong 2 loại:
    - $A \rightarrow a$, $A \rightarrow Ba$ trong đó $A, B \in \Delta$, $a \in \Sigma$
    - $A \rightarrow a$, $A \rightarrow aB$ trong đó $A, B \in \Delta$, $a \in \Sigma$
  - Sinh ra các ngôn ngữ chính quy (regular languages)
  - Đoán nhận bởi automat hữu hạn (finite state automaton)

Phân loại chomsky



Phần 3

Văn phạm chính quy và automat hữu hạn

Văn phạm chính quy

- Văn phạm chính quy giới hạn các luật có dạng:

  $A \rightarrow a \mid aB$ với điều kiện $A, B \in \Delta, a \in \Sigma$

- 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

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

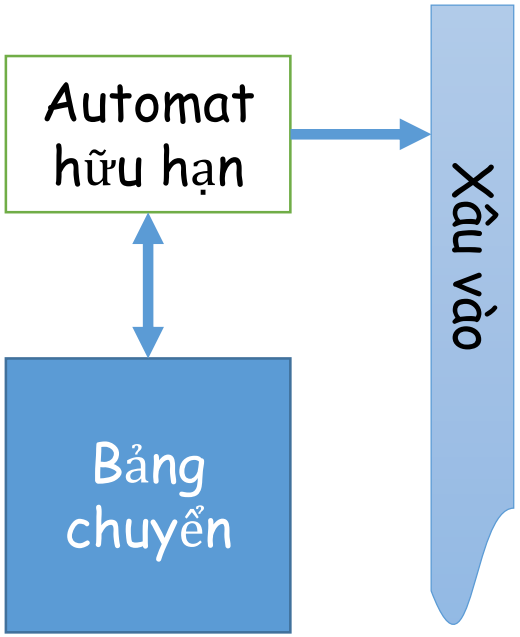
    <phần sau> → chữ <phần sau>

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

    <tên riêng> → chữ (chữ | số) *

Automat hữu hạn

- 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



Phần 4

Văn phạm phi ngữ cảnh và automat đẩy xuống

Văn phạm phi ngữ cảnh

- Văn phạm phi ngữ cảnh giới hạn các luật sinh phải có dạng $A \rightarrow \alpha$ trong đó $A \in \Delta$ (nói một cách vắn tắt là 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> “)”

BNF (Backus-Naur Form)

- 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 sử 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

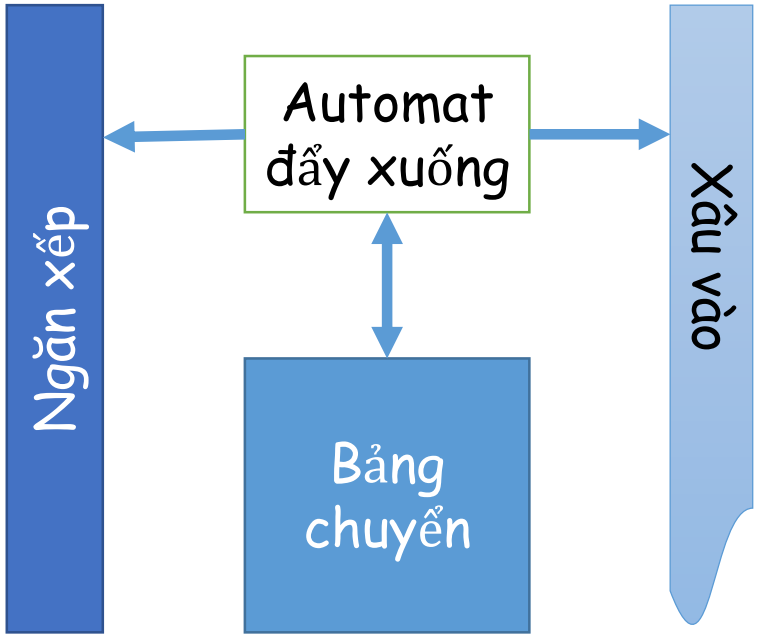
BNF (Backus-Naur Form)

```
// 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
```

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



Sinh 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)

Phần 5

Văn phạm có đệ quy trái

Văn phạm có đệ quy trái

- 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

Văn phạm có đệ quy trái

- 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:

  $A \rightarrow \beta\ R$

  $R \rightarrow \alpha\ R \mid \varepsilon$

Phần 6

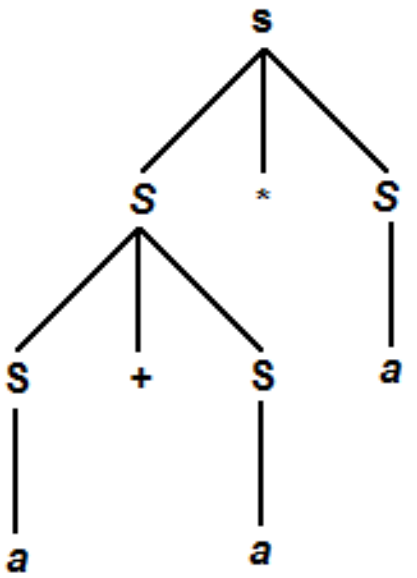
Văn phạm đơn nghĩa

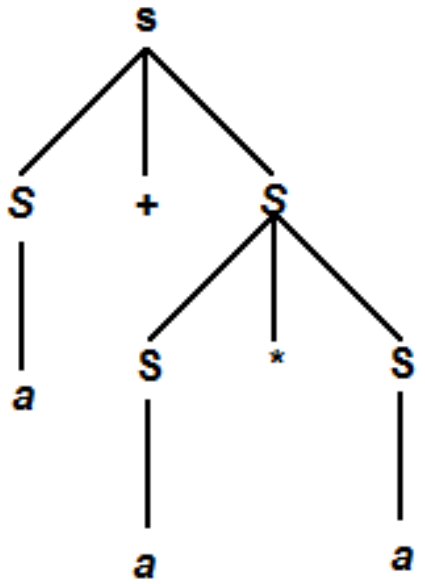
Văn phạm đơn nghĩa

- 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

Văn phạm đơn nghĩa

- 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)





Phần 7

Bài tập

Bài tập

1. Hãy sửa đổi văn phạm ở slide 31 để:
   - Văn phạm này trở thành đơn nghĩa
   - Văn phạm này trở thành đơn nghĩa và phép + thực hiện trước phép *
2. Chứng minh rằng, tất cả các chuỗi nhị phân sinh bởi văn phạm dưới đây đều chia hết cho 3

   $$X \longrightarrow 11 \mid 1001 \mid X0 \mid XX$$

   - Văn phạm trên có sinh ra mọi chuỗi nhị phân chia hết cho 3 hay không?
3. Viết biểu thức chính quy sinh ra tất cả các số nhị phân lớn hơn 101001

Bài tập

4. Cho văn phạm:

$$S \rightarrow S => W \mid W , E$$

   a. Hãy chỉ ra các kí hiệu thuộc văn phạm, kí hiệu nào là terminal, kí hiệu nào là nonterminal

   b. Hãy khử đệ quy trái cho văn phạm trên

5. Khử đệ quy trái cho văn phạm G gồm các luật:

$$A \rightarrow B\ a \mid C\ b$$

$$B \rightarrow C\ b \mid B\ c$$

$$C \rightarrow A\ c \mid b$$