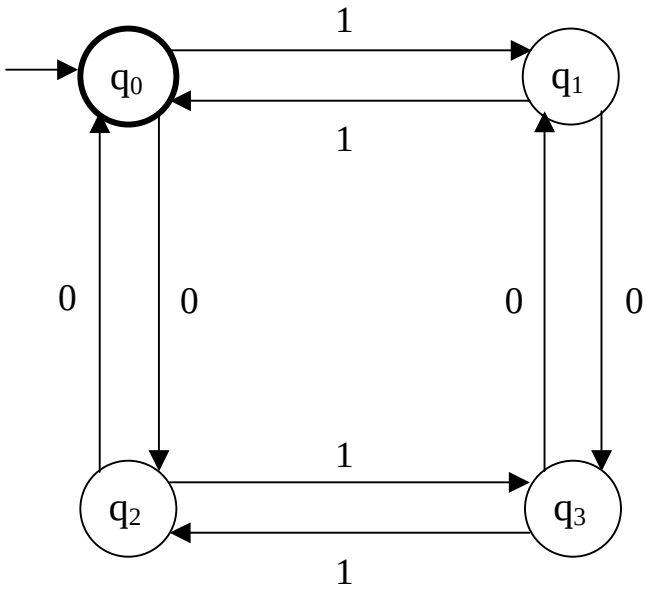
BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC

NGUYỄN GIA ĐỊNH

LÝ THUYẾT NGÔN NGỮ HÌNH THỨC VÀ ÔTÔMAT



HUẾ – 2004

LỜI NÓI ĐẦU

Mấy chục năm gần đây, chúng ta đã chứng kiến sự phát triển mãnh liệt trong các lĩnh vực nghiên cứu toán học liên quan đến máy tính và tin học. Sự phát triển phi thường của các máy tính và những thay đổi sâu sắc trong phương pháp luận khoa học đã mở ra những chân trời mới cho toán học với một tốc độ không thể sánh được trong suốt lịch sử lâu dài của toán học. Những phát triển đa dạng của toán học đã trực tiếp tạo ra "thuở ban đầu" của máy tính và tin học và các tiến bộ trong tin học đã dẫn đến sự phát triển rất mạnh mẽ một số ngành toán học.

Vì vậy, toán học đóng vai trò trung tâm trong các cơ sở của tin học. Có thể kể ra một số lĩnh vực nghiên cứu đáng chú ý trong mối quan hệ này. Thật là thú vị khi nhận thấy rằng các lĩnh vực này cũng phản ánh sự phát triển lịch sử của tin học.

1. Lý thuyết kinh điển về tính toán bắt đầu bằng công trình của Gödel, Tarski, Church, Post, Turing và Kleene chiếm vị trí trung tâm.
2. Trong lý thuyết ôtômat và ngôn ngữ hình thức kinh điển, các khái niệm cơ bản là ôtômat, văn phạm và ngôn ngữ, với các công trình sáng giá của Axel Thue, Chomsky, Post.

Ngoài hai lĩnh vực trên, nhiều lĩnh vực quan trọng khác thuộc về các cơ sở toán học của tin học; chẳng hạn, lý thuyết độ phức tạp, ngữ nghĩa và lý thuyết về tính đúng đắn của các ngôn ngữ lập trình, lý thuyết mật mã, lý thuyết các cấu trúc dữ liệu và lý thuyết các cơ sở dữ liệu.

Lý thuyết ngôn ngữ hình thức và ôtômat đóng một vai trò rất quan trọng trong các cơ sở toán học của tin học. Ngôn ngữ hình thức được sử dụng trong việc xây dựng các ngôn ngữ lập trình, lý thuyết về các chương trình dịch. Các ngôn ngữ hình thức tạo thành một công cụ mô tả đối với các mô hình tính toán cả cho dạng thông tin vào-ra lẫn kiểu thao tác. Lý thuyết ngôn ngữ hình thức, chính vì thực chất của nó là một lĩnh vực khoa học liên ngành; nhu cầu mô tả hình thức văn phạm được phát sinh trong nhiều ngành khoa học khác nhau từ ngôn ngữ học đến sinh vật học. Do đó những khía cạnh thích hợp của lý thuyết ngôn ngữ hình thức sẽ có tầm quan trọng quyết định trong các giáo trình về *Lý thuyết ngôn ngữ hình thức và ôtômat*.

Ngoài ra, một trong các vấn đề cơ bản của lý thuyết tính toán là các bài toán nào có các thuật toán để giải. Sự phát triển có tính chất nền tảng của lôgic toán trong những năm 30 của thế kỷ 20 đã chỉ ra việc tồn tại các bài toán không giải được, đó là các bài toán mà không thể có một thuật toán nào giải được chúng.

Cần phải có một mô hình tính toán để thiết lập tính không giải được. Mô hình tính toán đó là máy Turing, nó đã được đưa ra từ trước khi các máy tính điện tử ra đời khá lâu. Các máy Turing lập thành mô hình tính toán tổng quát được dùng rộng rãi nhất.

Giáo trình này nhằm trình bày về văn phạm hình thức và các ôtômat cũng như máy Turing, là những công cụ sinh ngôn ngữ, đồng thời đề cập đến các tính chất của ngôn ngữ chính quy, ngôn ngữ phi ngữ cảnh, ngôn ngữ đệ quy và ngôn ngữ đệ quy đếm được. Ngoài ra, giáo trình cũng giới thiệu sơ lược về trình biên dịch, một phần quan trọng của học phần *Chương trình dịch*, học phần gắn bó chặt chẽ với *Lý thuyết ngôn ngữ hình thức và ôtômat*. Một phần rất quan trọng trong lý thuyết thuật toán là lớp các ngôn ngữ (hay bài toán) P và NP cũng như lớp các ngôn ngữ NP-đầy đủ được giới thiệu trong phần phụ lục.

Nội dung của tài liệu này được bố trí trong 5 chương, không kể lời nói đầu, mục lục, tài liệu tham khảo và phần phụ lục:

Chương I: Trình bày về các khái niệm cơ bản của ngôn ngữ, cấu trúc của văn phạm sinh ra ngôn ngữ và sự phân cấp Chomsky của ngôn ngữ.

Chương II: Trình bày về ngôn ngữ chính quy, trong đó có các công cụ sinh ra ngôn ngữ chính quy là văn phạm chính quy, ôtômat hữu hạn (đơn định và không đơn định) và biểu thức chính quy.

Chương III: Đi sâu về ngôn ngữ phi ngữ cảnh và ôtômat đẩy xuống là công cụ đoán nhận ngôn ngữ phi ngữ cảnh.

Chương IV: Giới thiệu về máy Turing và vấn đề không giải được của thuật toán.

Chương V: Trình bày sơ lược về các quá trình của sự biên dịch của các ngôn ngữ, đặc biệt là ngôn ngữ lập trình.

Đây là một tài liệu tham khảo, học tập cho sinh viên, học viên cao học và nghiên cứu sinh các chuyên ngành Toán-Tin, Công nghệ thông tin, Tin học và những ai quan tâm về văn phạm, ngôn ngữ hình thức và ôtômat.

Chúng tôi xin chân thành cám ơn các đồng nghiệp đã động viên và góp ý cho công việc viết giáo trình *Lý thuyết ngôn ngữ hình thức và ôtômat* này và lời cám ơn đặc biệt xin dành cho Thầy Lê Mạnh Thạnh và đồng nghiệp Nguyễn Hoàng Sơn về sự cung cấp một số tài liệu quan trọng và động viên kịp thời tạo niềm hưng phấn để tác giả giảng dạy và viết giáo trình cho học phần *Lý thuyết ngôn ngữ hình thức và ôtômat*.

Tác giả mong nhận được sự chỉ giáo của các đồng nghiệp và độc giả về những thiếu sót khó tránh khỏi của cuốn sách.

Trọng Đông năm Giáp Thân (2004)

Nguyễn Gia Định

MỤC LỤC

CHƯƠNG I:
NHẬP MÔN VỀ VĂN PHẠM VÀ NGÔN NGỮ HÌNH THỨC

1.1. KHÁI NIỆM NGÔN NGỮ.

1.1.1. Mở đầu:

Từ ngàn xưa con người muốn giao tiếp với nhau phải dùng ngôn ngữ. Ngôn ngữ để con người có thể giao tiếp với nhau được gọi là ngôn ngữ tự nhiên, chẳng hạn như tiếng Anh, tiếng Nga, tiếng Việt là các ngôn ngữ tự nhiên. Con người muốn giao tiếp với máy tính tất nhiên cũng thông qua ngôn ngữ. Con người muốn máy tính thực hiện công việc, phải viết các yêu cầu đưa cho máy bằng ngôn ngữ máy hiểu được. Việc viết các yêu cầu ta gọi là lập trình. Ngôn ngữ dùng để lập trình được gọi là ngôn ngữ lập trình.

Cả ngôn ngữ lập trình lẫn ngôn ngữ tự nhiên đều có thể xem như những tập các từ, tức là các xâu hữu hạn các phần tử của một bộ chữ cái cơ sở nào đó. Khái niệm ngôn ngữ được đưa vào trong mục này rất tổng quát. Chắc chắn bao hàm cả ngôn ngữ lập trình lẫn tự nhiên, và cả mọi ngôn ngữ vô nghĩa mà ta có thể nghĩ đến. Về mặt truyền thống, lý thuyết ngôn ngữ hình thức liên quan đến các đặc tả cú pháp của ngôn ngữ nhiều hơn là đến những vấn đề ngữ nghĩa. Một đặc tả về cú pháp của một ngôn ngữ có hữu hạn từ, ít nhất về nguyên tắc, có thể được cho bằng cách liệt kê các từ. Điều đó không thể áp dụng đối với các ngôn ngữ có vô hạn từ. Nhiệm vụ chính của lý thuyết ngôn ngữ hình thức là nghiên cứu các cách đặc tả hữu hạn của các ngôn ngữ vô hạn.

Lý thuyết cơ sở của tính toán cũng như của nhiều ngành khác nhau của nó, chẳng hạn mật mã học, có liên quan mật thiết với lý thuyết ngôn ngữ. Các tập vào và ra của một thiết bị tính toán có thể được xem như các ngôn ngữ và nói một sâu sắc hơn thì các mô hình tính toán có thể được đồng nhất với các lớp các đặc tả ngôn ngữ theo nghĩa mà sau này sẽ nêu chính xác hơn. Chẳng hạn, các máy Turing có thể được đồng nhất với các văn phạm cấu trúc câu và các ôtômat hữu hạn có thể đồng nhất với các văn phạm chính quy.

1.1.2. Định nghĩa: Một *bảng chữ cái* là một tập hữu hạn khác rỗng. Các phần tử của một bảng chữ cái Σ được gọi là các *chữ cái* hay các *ký hiệu*.

Thí dụ 1: Dưới đây là các bảng chữ cái:

$\Sigma = \{a, b, c, \ldots, z\}$,

$U = \{\alpha, \beta, \gamma, \delta, \varepsilon, \eta, \varphi, \kappa, \mu, \chi, \nu, \pi, \theta, \rho, \sigma, \tau, \omega, \xi, \psi\}$,

$V = \{0, 1\}$, $W = \{\text{if, then, else, a, b, c, d, e, f, } +, -, *, /, =, \neq\}$.

1.1.3. Định nghĩa: Một *từ* trên bảng chữ cái Σ là một xâu hữu hạn gồm một số lớn hơn hay bằng không các chữ của Σ, trong đó một chữ có thể xuất hiện vài lần. Xâu không có chữ nào được gọi là *từ rỗng* và được ký hiệu là ε.

Như vậy, theo định nghĩa, hai từ $\alpha=a_1a_2\ldots a_n$ và $\beta=b_1b_2\ldots b_m$ là bằng nhau, $\alpha=\beta$, nếu $n=m$ và $a_i=b_i$ với mọi $i=1, 2, \ldots, n$.

Tập mọi từ (t.ư. mọi từ khác rỗng) trên bảng chữ cái Σ được ký hiệu là Σ^* (t.ư. Σ^+). Các tập Σ^* và Σ^+ là vô hạn với bất kỳ Σ nào (thật ra, Σ^* và Σ^+ là vô hạn đếm được như Mệnh đề 1.1.5 dưới đây). Về mặt đại số, Σ^* là một vị nhóm tự do với đơn vị là từ rỗng ε sinh bởi Σ và Σ^+ là một nửa nhóm tự do sinh bởi Σ.

Đối với các từ $\alpha\in\Sigma^*$ và $\alpha'\in\Sigma'^*$, việc đặt α và α'cạnh nhau để có từ mới $\alpha\alpha'\in(\Sigma\cup\Sigma')^*$ được gọi là *phép ghép* α với α'. Từ rỗng là phần tử đơn vị đối với phép ghép: $\omega\varepsilon = \varepsilon\omega = \omega$ đúng với mọi từ ω. Vì phép ghép có tính kết hợp, nghĩa là với mọi từ α, β, γ, ta có $(\alpha\beta)\gamma = \alpha(\beta\gamma)$, nên ký hiệu ω^n, với n là số tự nhiên, được dùng theo nghĩa quen thuộc:

$$\omega^n = \begin{cases} \varepsilon \ khi \ n = 0, \\ \omega \ khi \ n = 1, \\ \omega^{n-1}\omega \ khi \ n > 1. \end{cases}$$

Thí dụ 2: ε, 0, 01, 101, 1010, 110011 là các từ trên bảng chữ cái V = {0,1}.

beautiful là một từ trên bảng chữ cái Σ = {a, b, c, …, z}.

Trên bảng chữ cái W = {if, then, else, a, b, c, d, e, f, +, –, ∗, /, =, ≠}, nếu α là từ if a+b=c then c∗d=e và β là từ else c/d=f thì αβ là từ:

if a + b = c then c ∗ d = e else c / d = f.

1.1.4. Định nghĩa: *Độ dài* của một từ ω, ký hiệu |ω| hay d(ω), là số các chữ có mặt trong ω. Theo định nghĩa, |ε|=0.

Hàm độ dài có một số tính chất hình thức của lôgarit: với mọi từ α, β và mọi số tự nhiên n,

$$|\alpha\beta| = |\alpha| + |\beta|, \ \ |\alpha^n| = n|\alpha|.$$

Đảo của một từ có được bằng cách viết các chữ cái theo thứ tự ngược lại; nếu $\omega=a_1a_2\ldots a_n$ là một từ trên bảng chữ Σ thì đảo ω^R của nó là từ trên bảng chữ Σ:

$$\omega^R = a_n\ldots\ a_2a_1.$$

Từ α được gọi là một *từ con* hay một *nhân tử* của từ β nếu có các từ u và v sao cho $\beta=u\alpha v$. Ngoài ra, nếu u=ε (t.ư. v=ε) thì α được gọi là từ con *đầu* hay *tiền tố* (t.ư. từ con *cuối* hay *hậu tố*) của β.

Thí dụ 3: Từ ω=010111001 trên bảng chữ cái {0, 1} có độ dài 9, trong đó 0101 là tiền tố và 11001 là hậu tố của ω.

Từ if a + b = c then c ∗ d = e else c / d = f trên bảng chữ cái W ở trên có độ dài là 18, trong đó then c ∗ d = e là từ con của nó.

1.1.5. Mệnh đề: Nếu Σ là bảng chữ cái thì Σ^* là tập (vô hạn) đếm được.

***Chứng minh*:** Do mỗi số tự nhiên n đều tồn tại một từ trên Σ có độ dài n nên Σ^* là một tập vô hạn. Giả sử $\Sigma=\{a_1, a_2, \ldots, a_n\}$. Xét ánh xạ f từ Σ^* vào tập hợp **N** các số tự nhiên xác định bởi:

$$f(\varepsilon) = 0,\ f(a_i) = i,\ f(\alpha a_i) = (n+1)f(\alpha)+i,\ \forall \alpha \in \Sigma^*.$$

Với $\alpha = a_{i_0} a_{i_1} \ldots a_{i_k}$, $\beta = b_{j_0} b_{j_1} \ldots b_{j_h}$ và $f(\alpha) = f(\beta)$. Khi đó,

$$(n+1)^k i_0+(n+1)^{k-1} i_1+ \ldots +(n+1)i_{k-1}+i_k = (n+1)^h j_0+(n+1)^{h-1} j_1+ \ldots +(n+1)j_{h-1}+j_h,$$

trong đó 2 vế là hai khai triển của một số nguyên theo cơ số n+1. Do đó, k=h và $i_u=j_u$ với $1 \le u \le k$ hay $\alpha=\beta$. Vì vậy, f là một đơn ánh. Từ đó suy ra Σ^* là một đếm được.

1.1.6. Định nghĩa: Mỗi tập con của Σ^* được gọi là một *ngôn ngữ hình thức* hay ngắn gọn hơn là một *ngôn ngữ* trên Σ. Đặc biệt, tập ∅ là một ngôn ngữ trên Σ, gọi là ngôn ngữ rỗng; tập {ε} cũng là một ngôn ngữ trên Σ, đây là ngôn ngữ chỉ chứa từ rỗng và Σ^* là ngôn ngữ gồm tất cả các từ trên Σ.

Thí dụ 4: $L_1 = \{\varepsilon, a, b, abb, aab, aaa, bbb, abab\}$,

$$L_2 = \{a^n b^n \mid n \in \mathbf{N}\}$$

là hai ngôn ngữ trên bảng chữ $\Sigma = \{a, b\}$, L_1 là ngôn ngữ hữu hạn trong khi L_2 là ngôn ngữ vô hạn. Mỗi từ thuộc ngôn ngữ L_2 có số chữ cái a bằng số chữ cái b với a và b không xen kẻ, a nằm ở phía trái và b ở phía phải của nó.

Các họ ngôn ngữ cụ thể thường được đặc trưng một cách tiện lợi qua các phép toán xác định trên ngôn ngữ, họ đó gồm các ngôn ngữ nhận được bằng việc tổ hợp từ một số ngôn ngữ cho trước bởi một số phép toán nào đó. Vì ngôn ngữ là tập hợp nên ta có các phép toán Boole như là phép giao, phép hợp, phép hiệu, phép lấy bù. Chẳng hạn, với L_1 và L_2 là hai ngôn ngữ trên bảng chữ Σ thì ta có các ngôn ngữ mới sau cũng trên bảng chữ Σ: $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$, $\Sigma^* \setminus L_1$. Ngoài ra, ta còn có các phép toán khác là "phép ghép" và "phép cấu xạ" như dưới đây.

1.1.7. Định nghĩa: Cho hai ngôn ngữ L_1 trên bảng chữ Σ_1 và L_2 trên bảng chữ Σ_2. *Ghép* hay *tích* của hai ngôn ngữ L_1 và L_2 là ngôn ngữ trên bảng chữ $\Sigma_1 \cup \Sigma_2$, ký hiệu L_1L_2, được xác định bởi:

$$L_1L_2 = \{\alpha\beta \mid \alpha \in L_1 \text{ và } \beta \in L_2\}.$$

Dễ dàng thấy rằng phép ghép có tính kết hợp, nghĩa là với mọi ngôn ngữ L_1, L_2 và L_3, ta luôn có:

$$(L_1L_2)L_3 = L_1(L_2L_3).$$

Ngoài ra, với mọi ngôn ngữ L, ta có:

$$\varnothing L = L\varnothing = \varnothing,\ \ \{\varepsilon\}L = L\{\varepsilon\} = L,$$

và phép ghép có tính phân phối đối với phép hợp, nghĩa là

$$L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3,\ (L_2 \cup L_3)L_1 = L_2L_1 \cup L_3L_1.$$

Vì phép ghép ngôn ngữ có tính kết hợp nên ký hiệu L^n được dùng với mọi ngôn ngữ L và số tự nhiên n theo nghĩa quen thuộc sau:

$$L^n = \begin{cases} \{\varepsilon\} \text{ khi } n = 0, \\ L \quad \text{khi } n = 1, \\ L^{n-1}L \text{ khi } n > 1. \end{cases}$$

Lặp hay *bao đóng ghép* của ngôn ngữ L, ký hiệu L^*, được định nghĩa là hợp của mọi luỹ thừa của L:

$$L^* = \bigcup_{n=0}^{\infty} L^n .$$

Lặp không-ε hay *bao đóng ghép không-ε* của L, ký hiệu L^+, được định nghĩa là hợp của mọi luỹ thừa dương của L:

$$L^+ = \bigcup_{n=1}^{\infty} L^n .$$

Thí dụ 5: 1) Xét các ngôn ngữ trên bảng chữ $\Sigma = \{0, 1\}$:

$$L_1 = \{0, 01\},\ L_2 = \{01, 10\},\ L_3 = \{0\}.$$

$L_2L_3 = \{010, 100\}$, $L_1 \cup (L_2L_3) = \{0, 01, 010, 100\}$, $L_1 \cup L_2 = \{0, 01, 10\}$, $L_1 \cup L_3 = \{0, 01\}$, $(L_1 \cup L_2)(L_1 \cup L_3) = \{00, 001, 010, 0101, 100, 1010\}$. Do đó $L_1 \cup (L_2L_3) \neq (L_1 \cup L_2)(L_1 \cup L_3)$ tức là phép hợp không có tính phân phối đối với phép ghép.

$L_2 \cap L_3 = \varnothing$, $L_1(L_2 \cap L_3) = \varnothing$, $L_1L_2 = \{001, 010, 0101, 0110\}$, $L_1L_3 = \{00, 010\}$, $(L_1L_2) \cap (L_1L_3) = \{010\}$. Do đó $L_1(L_2 \cap L_3) \neq (L_1L_2) \cap (L_1L_3)$ tức là phép ghép không có tính phân phối đối với phép giao.

$L_1 \cap (L_2L_3) = \varnothing$, $L_1 \cap L_2 = \{01\}$, $L_1 \cap L_3 = \{0\}$, $(L_1 \cap L_2)(L_1 \cap L_3) = \{010\}$. Do đó $L_1 \cap (L_2L_3) \neq (L_1 \cap L_2)(L_1 \cap L_3)$ tức là phép giao không có tính phân phối đối với phép ghép.

2) Xét ngôn ngữ $L = \{0, 1\}$ trên bảng chữ $\Sigma = \{0, 1\}$. Ta có:

$L^2 = \{00, 01, 10, 11\}$, tập hợp các xâu nhị phân độ dài 2;

$L^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$, tập hợp các xâu nhị phân độ dài 3;

Tương tự, L^n là tập hợp các xâu nhị phân độ dài n.

Vì vậy, L^* là tập hợp tất cả các xâu nhị phân.

3) Xét hai ngôn ngữ trên bảng chữ $\Sigma = \{a\}$:

$$L_1 = \{a^{2n} \mid n \geq 1\},\ L_2 = \{a^{5n+3} \mid n \geq 0\}.$$

Khi đó, ta có $L_1 = \{a^2\}^+$, $L_2 = \{a^5\}^*\{a^3\}$.

Một phép toán có tầm quan trọng cốt yếu trong lý thuyết ngôn ngữ là phép cấu xạ, như được định nghĩa dưới đây.

1.1.8. Định nghĩa: Cho hai bảng chữ Σ và Σ'. Ánh xạ $f: \Sigma^* \longrightarrow \Sigma'^*$ thoả mãn điều kiện

$$f(\alpha\beta) = f(\alpha)f(\beta) \text{ với mọi từ } \alpha, \beta \in \Sigma^* \quad (1)$$

được gọi là một *cấu xạ*. Đối với ngôn ngữ L trên Σ, $f(L) = \{f(\alpha) \mid \alpha \in L\}$ là ngôn ngữ trên Σ'. Theo điều kiện (1), để xác định cấu xạ f, chỉ cần liệt kê mọi từ f(a) trên Σ' với a chạy trên mọi chữ cái của Σ.

Cấu xạ f gọi là *không xoá* (t.ư. *chữ - thành - chữ*) nếu $f(a) \neq \varepsilon$ (t.ư. $f(a) \in \Sigma'$) với mỗi $a \in \Sigma$.

Thí dụ 6: Xét bảng chữ cái tiếng Anh $\Sigma = \{A, B, C, \ldots, Z\}$. Mỗi cấu xạ chữ - thành - chữ

$$f_i: \Sigma^* \longrightarrow \Sigma^*, \ 0 \leq i \leq 25$$

ánh xạ mỗi chữ thành chữ đứng sau nó i vị trí trong bảng chữ cái, trong đó phần cuối của bảng chữ cái được nối tiếp vòng tròn lên phần đầu. Chẳng hạn,

$$f_3(A) = D, \ f_7(Y) = F, \ f_{25}(Z) = Y.$$

Trong mật mã học, mỗi cấu xạ f_i thường được đề cập đến như *cách mã hoá Caesar*. Chẳng hạn,

$$f_{25}(IBM) = HAL, \ f_3(HELP) = KHOS.$$

Dễ dàng thấy rằng các cấu xạ f_i có tính giao hoán:

$$f_i \circ f_j = f_j \circ f_i \text{ với mọi i, j.}$$

Ngoài ra, $f_{26-i} \circ f_i = f_0$ với mọi $i \geq 1$. Như vậy, nếu một bản rõ nào đó được mã hoá bằng cách dùng f_i, chính bản rõ đó có thể tìm lại được bằng cách dùng f_{26-i} để giải mã.

1.2. VĂN PHẠM VÀ NGÔN NGỮ SINH BỞI VĂN PHẠM.

1.2.1. Mở đầu:

Ta có thể hình dung một văn phạm như một "thiết bị tự động" mà nó có khả năng sinh ra một tập hợp các từ trên một bảng chữ cái cho trước. Mỗi từ được sinh ra sau một số hữu hạn bước thực hiện các quy tắc của văn phạm.

Việc xác định một ngôn ngữ trên bảng chữ cái cho trước có thể được thực hiện bằng một trong các cách thức sau:

Cách 1: Đối với mỗi từ thuộc ngôn ngữ đã cho, ta có thể chọn một quy cách hoạt động của "thiết bị tự động" để sau một số hữu hạn bước làm việc nó dừng và sinh ra chính từ đó.

Cách 2: "Thiết bị tự động" có khả năng lần lượt sinh ra tất cả các từ trong ngôn ngữ đã cho.

Cách 3: Với mỗi từ ω cho trước, "thiết bị tự động" có thể cho biết từ đó có thuộc ngôn ngữ đã cho hay không.

Trong lý thuyết văn phạm, người ta đã chứng minh được rằng ba cách thức trên là tương đương nhau hay văn phạm làm việc theo các cách trên là tương đương nhau. Vì vậy, ở đây ta quan tâm đến cách thứ nhất, tức là ta xét văn phạm như là một "thiết bị tự động" sinh ra các từ. Vì lẽ đó mà người ta còn gọi các "thiết bị tự động" đó là văn phạm sinh.

Việc sinh ra các từ có thể được thực hiện bằng nhiều cách khác nhau. Các từ có thể được sinh ra bởi các văn phạm, bởi các Ôtômat, bởi các máy hình thức như máy Turing, …Ở đây ta đề cập đến cách của CHOMSKY đưa ra vào những năm 1956-1957.

1.2.2. Định nghĩa: *Văn phạm* G là một bộ sắp thứ tự gồm 4 thành phần:

$$G = < \Sigma, \Delta, S, P >,$$

trong đó:

a) Σ là một bảng chữ, gọi là *bảng chữ kết thúc* hay *từ điển cơ bản*, mỗi phần tử của nó được gọi là một *ký hiệu kết thúc* hay *ký hiệu cơ bản*;

b) Δ là một bảng chữ, $\Delta \cap \Sigma = \varnothing$, gọi là *bảng chữ không kết thúc* hay *từ điển hỗ trợ*, mỗi phần tử của nó được gọi là một *ký hiệu không kết thúc* hay *ký hiệu hỗ trợ*.

c) $S \in \Delta$ được gọi là *ký hiệu đầu*;

d) P là tập hợp các cặp thứ tự $<\alpha, \beta>$, trong đó $\alpha, \beta \in (\Sigma \cup \Delta)^*$ và trong α chứa ít nhất một ký hiệu không kết thúc; P được gọi là *tập các quy tắc thay thế*, $<\alpha, \beta>$ được gọi là một *quy tắc* hay *sản suất* và thường được viết cho thuận tiện là $\alpha \rightarrow \beta$, α được gọi là *vế trái* và β được gọi là *vế phải* của quy tắc này.

Thí dụ 7: Các bộ bốn sau là các văn phạm:

$G_1 = <\{0, 1\}, \{S\}, S, \{S \rightarrow 0S1, S \rightarrow \varepsilon\}>$,

$G_2 = <\{a, b\}, \{S, A\}, S, \{S \rightarrow Ab, A \rightarrow aAb, A \rightarrow \varepsilon\}>$,

$G_3 = <\{a, b, c\}, \{S, A, B, C\}, S, \{S \rightarrow ABC, A \rightarrow aA, B \rightarrow bB, C \rightarrow cC, A \rightarrow a, B \rightarrow b, C \rightarrow c\}>$

$G_4 = <\Sigma, \Delta, S, P>$, trong đó

Σ={tôi, anh, chị, ăn, uống, cơm, phở, sữa, café},

Δ={<câu>, <chủngữ>, <vịngữ>, <độngtừ1>, <độngtừ2>, <danhtừ1>,<danhtừ2>},

S=<câu>,

P={<câu>→<chủngữ><vịngữ>, <chủngữ>→tôi,<chủngữ>→anh,<chủngữ>→chị,
<vịngữ>→<độngtừ1><danhtừ1>, <vịngữ>→<độngtừ2><danhtừ2>,
<độngtừ1>→ăn, <độngtừ2>→uống, <danhtừ1>→cơm, <danhtừ1>→phở,
<danhtừ2>→sữa, <danhtừ2>→café}.

1.2.3. Định nghĩa: Cho văn phạm $G = < \Sigma, \Delta, S, P >$ và $\eta, \omega \in (\Sigma \cup \Delta)^*$. Ta nói ω được *suy dẫn trực tiếp* từ η trong G, ký hiệu $\eta \overset{G}{\vdash} \omega$ hay ngắn gọn là $\eta \vdash \omega$ (nếu không sợ nhầm lẫn), nếu tồn tại quy tắc $\alpha \to \beta \in P$ và $\gamma, \delta \in (\Sigma \cup \Delta)^*$ sao cho $\eta = \gamma\alpha\delta$, $\omega = \gamma\beta\delta$.

Điều này có nghĩa là nếu η nhận vế trái α của quy tắc $\alpha \to \beta$ như là từ con thì ta thay α bằng β để được từ mới ω.

1.2.4. Định nghĩa: Cho văn phạm $G = < \Sigma, \Delta, S, P >$ và $\eta, \omega \in (\Sigma \cup \Delta)^*$. Ta nói ω được *suy dẫn* từ η trong G, ký hiệu $\eta \overset{G}{\models} \omega$ hay ngắn gọn là $\eta \models \omega$ (nếu không sợ nhầm lẫn), nếu $\eta = \omega$ hoặc tồn tại một dãy $\omega_0, \omega_1, \ldots, \omega_k \in (\Sigma \cup \Delta)^*$ sao cho $\omega_0 = \eta$, $\omega_k = \omega$ và $\omega_{i-1} \overset{G}{\vdash} \omega_i$, với i=1, 2, …, k. Khi đó dãy $\omega_0, \omega_1, \ldots, \omega_k$ được gọi là một *dẫn xuất* của ω từ η trong G và số k được gọi là *độ dài* của dẫn xuất này. Nếu ω_i được suy dẫn trực tiếp từ ω_{i-1} bằng việc áp dụng một quy tắc p nào đó trong G thì ta nói quy tắc p được *áp dụng* ở bước thứ i.

1.2.5. Định nghĩa: Cho văn phạm $G = < \Sigma, \Delta, S, P >$. Từ $\omega \in \Sigma^*$ được gọi là *sinh bởi* văn phạm G nếu tồn tại suy dẫn $S \overset{G}{\models} \omega$. Ngôn ngữ *sinh bởi* văn phạm G, ký hiệu L(G), là tập hợp xác định bởi:

$$L(G) = \{\omega \in \Sigma^* \mid S \overset{G}{\models} \omega\}.$$

Hai văn phạm G_1 và G_2 được gọi là tương đương nếu $L(G_1) = L(G_2)$.

Thí dụ 8: 1) Xét văn phạm G_1 như trong 1) của Thí dụ 7. Từ $0^4 1^4$ được suy dẫn từ S bằng dãy dẫn xuất độ dài 5: $S \vdash 0S1 \vdash 00S11 \vdash 000S111 \vdash 0000S1111 \vdash 0^4 1^4$.

Bằng việc sử dụng n lần ($n \geq 0$) quy tắc 1 rồi quy tắc 2, ta có: $S \models 0^n 1^n$. Do đó $L(G_1) = \{0^n 1^n \mid n \geq 0\}$.

2) Xét văn phạm G_2 như trong 2) của Thí dụ 7. Sử dụng quy tắc 1, rồi n lần ($n \geq 0$) quy tắc 2, sau đó sử dụng quy tắc 3 để kết thúc, ta có:

$$S \vdash Ab \models a^n A b^n b \vdash a^n b^{n+1}.$$

Do đó $L(G_2) = \{a^n b^{n+1} \mid n \geq 0\}$.

3) Xét văn phạm G_3 như trong 3) của Thí dụ 7. Sử dụng quy tắc 1, rồi m-1 lần ($m \geq 1$) quy tắc 2, n-1 lần ($n \geq 1$) quy tắc 3, k-1 lần ($k \geq 1$) quy tắc 4 (có thể xen kẻ), sau đó để kết thúc sử dụng các quy tắc 5,6, 7, ta có:

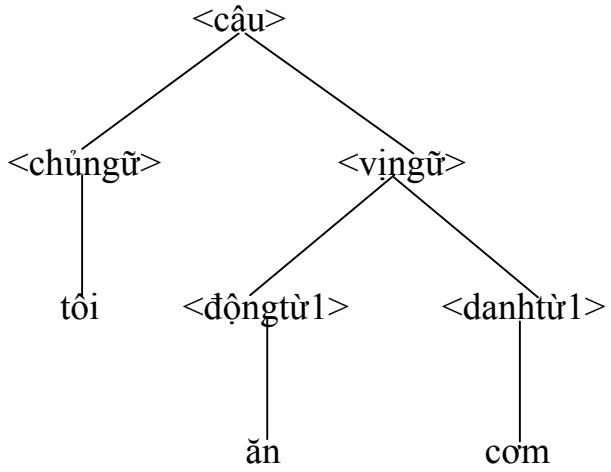
$$S \vdash ABC \models a^m A b^n B c^k C \models a^m b^n c^k.$$

Do đó $L(G_3) = \{a^m b^n c^k \mid m \geq 1, n \geq 1, k \geq 1\}$.

4) $L(G_4)$ = {tôi ăn cơm, anh ăn cơm, chị ăn cơm, tôi ăn phở, anh ăn phở,
chị ăn phở, tôi uống sữa, anh uống sữa, chị uống sữa, tôi uống café,
anh uống café, chị uống café}.

Ta có thể biểu diễn việc dẫn xuất từ <câu> đến một từ trong $L(G_4)$, chẳng hạn "tôi ăn cơm" bằng một cây gọi là *cây dẫn xuất* hay *cây phân tích cú pháp* như dưới đây. Tất nhiên, theo quan điểm phân tích cú pháp thực tế, việc xem xét các

quy tắc theo hướng ngược lại là từ phải qua trái. Điều đó có nghĩa là cây dưới đây được xử lý từ dưới lên trên chứ không phải là từ trên xuống dưới.



Thí dụ 9: 1) Cho hai văn phạm

$$G_1 = <\{a, b\}, \{S\}, S, \{S\to aSb, S\to ab\}>,$$

$$G_2 = <\{a, b\}, \{S, A, B\}, S, \{S\to ASB, A\to a, B\to b, S\to ab\}>.$$

Dễ dàng có được $L(G_1)=L(G_2)=\{a^nb^n \mid n \geq 1\}$ hay G_1 và G_2 là tương đương nhau.

Lưu ý rằng nếu các quy tắc có vế trái giống nhau có thể viết gọn lại. Chẳng hạn, như trong G_1 ta có thể viết hai quy tắc của nó dưới dạng $S\to aSb| ab$.

2) Cho hai văn phạm $G_3 = <\Sigma, \{S\}, S, P_3>$, $G_4 = <\Sigma, \{S\}, S, P_4>$, trong đó:

$\Sigma = \{0, 1, 2, 3, 4, 5 ,6, 7, 8, 9\}$,

$P_3 = \{S\to 1| 2| 3| 4| 5| 6| 7| 8| 9| S0| S1| S2| S3| S4| S5| S6| S7| S8| S9\}$,

$P_4 = \{S\to 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 1S| 2S| 3S| 4S| 5S| 6S| 7S| 8S| 9S\}$.

Sử dụng k-1 lần ($k \geq 1$) các quy tắc trong nhóm 10 quy tắc cuối của G_3, rồi một quy tắc trong nhóm 9 quy tắc của nó, ta có:

$$S\vdash Si_1\vdash Si_2i_1 \vdash\ldots\vdash Si_{k-1}\ldots i_2i_1 \vdash i_ki_{k-1}\ldots i_2i_1,$$

trong đó, $i_1, i_2, \ldots, i_{k-1} \geq 0$ và $i_k \geq 1$. Do đó, $L(G_3) = \{n \mid n \geq 1\} = \mathbf{N} \setminus \{0\}$.

Lập luận như trên, ta nhận được $L(G_4) = \{n\in\mathbf{N} \mid$ n có chữ số hàng đơn vị tuỳ ý và các chữ số khác $n \geq 1\}$. Vì vậy, G_3 và G_4 không tương đương nhau.

1.2.6. Bổ đề: Cho văn phạm $G = <\Sigma, \Delta, S, P>$. Khi đó nếu tồn tại trong P quy tắc chứa ký hiệu đầu S ở vế phải thì tồn tại văn phạm G' tương đương với G mà các quy tắc của nó không chứa ký hiệu đầu ở vế phải.

***Chứng minh*:** Lấy $S'\notin\Sigma \cup \Delta$, xét văn phạm $G' = <\Sigma, \Delta \cup \{S'\}, S', P'>$, trong đó $P'=P \cup \{S'\to\alpha \mid S\to\alpha \in P\}$. Rõ ràng trong P' không chứa quy tắc nào có S' ở vế phải. Ta chứng minh L(G)=L(G').

+ $\omega\in L(G)$ (hay $S \overset{G}{\vdash}\omega$): Giả sử dãy dẫn xuất của ω là $S \overset{G}{\vdash}\alpha \overset{G}{\vdash}\omega_1\overset{G}{\vdash}\ldots \overset{G}{\vdash}\omega$. Vì $S\overset{G}{\vdash}\alpha$ nên có $S\to\alpha\in P$, do đó $S'\to\alpha\in P'$ và vì $P \subset P'$ nên ta có $S'\overset{G'}{\vdash}\alpha\overset{G'}{\vdash}\omega$. Vậy $S'\overset{G'}{\vdash}\omega$ hay $\omega\in L(G')$.

+ $\omega \in L(G')$ (hay $S' \models^{G'} \omega$): Giả sử ta có dãy dẫn xuất là $S' \vdash^{G'} \alpha \models^{G'} \omega$. Vì $S' \vdash^{G'} \alpha$ nên $S' \to \alpha \in P'$, do đó tồn tại $S \to \alpha \in P$. Mặt khác, trong α không chứa S' nên các suy dẫn trực tiếp trong $\alpha \models^{G'} \omega$ chỉ sử dụng các quy tắc của P. Vậy ta có $S \vdash^{G} \omega$ hay $\omega \in L(G)$.

1.2.7. Định nghĩa: Văn phạm $G = < \Sigma, \Delta, S, P >$ mà không có một ràng buộc nào trên các quy tắc của nó được gọi là văn phạm *tổng quát* hay là văn phạm *nhóm 0*.

Như vậy, G là văn phạm nhóm 0 khi và chỉ khi các quy tắc của nó có dạng $\alpha A \beta \to \omega$, trong đó $A \in \Delta$, $\alpha, \beta, \omega \in (\Sigma \cup \Delta)^*$.

1.2.8. Định nghĩa: Văn phạm $G = < \Sigma, \Delta, S, P >$ mà các quy tắc của nó có dạng $\alpha A \beta \to \alpha \omega \beta$, trong đó $A \in \Delta$, $\alpha, \beta, \omega \in (\Sigma \cup \Delta)^*$, $\omega \neq \varepsilon$, được gọi là văn phạm *cảm ngữ cảnh* hay là văn phạm *nhóm 1*.

Các văn phạm mà các quy tắc của chúng có dạng trên, đồng thời chứa thêm quy tắc $S \to \varepsilon$, miễn sao ký hiệu đầu S không xuất hiện ở vế phải của bất kỳ quy tắc nào cũng được xếp vào lớp văn phạm nhóm 1.

Thí dụ 10: Cho văn phạm $G = <\{a, b, c\}, \{S, A, B, C\}, S, P>$, trong đó

$P = \{S \to aSAC, S \to abC, CA \to BA, BA \to BC, BC \to AC, bA \to bb, C \to c\}$.

Khi đó G là văn phạm cảm ngữ cảnh.

Sử dụng n-1 lần ($n \geq 1$) quy tắc 1, rồi quy tắc 2, kế đến sử dụng liên tiếp các quy tắc 3, 4, 5 (để đổi chỗ A và C), sau đó sử dụng n-1 lần quy tắc 6 và n lần quy tắc 7, ta có:

$$S \models a^{n-1}S(AC)^{n-1} \vdash a^n bC(AC)^{n-1} \models a^n bA^{n-1}C^n \models a^n b^n c^n.$$

Từ đó suy ra $L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

1.2.9. Định nghĩa: Văn phạm $G = < \Sigma, \Delta, S, P >$ mà các quy tắc của nó có dạng $A \to \omega$, trong đó $A \in \Delta$, $\omega \in (\Sigma \cup \Delta)^*$, $\omega \neq \varepsilon$, được gọi là văn phạm *phi ngữ cảnh* hay là văn phạm *nhóm 2*.

Các văn phạm mà các quy tắc của chúng có dạng trên, đồng thời chứa thêm quy tắc $S \to \varepsilon$, miễn sao ký hiệu đầu S không xuất hiện ở vế phải của bất kỳ quy tắc nào cũng được xếp vào lớp văn phạm nhóm 2.

Thí dụ 11: 1) Cho văn phạm $G_1 = <\{a, b\}, \{S, A\}, S, P>$, trong đó

$$P = \{S \to Sa, S \to Aa, A \to aAb, A \to ab\}.$$

Khi đó G_1 là văn phạm phi ngữ cảnh.

Sử dụng m-1 lần ($m \geq 1$) quy tắc 1, rồi quy tắc 2, sau đó sử dụng n-1 lần ($n \geq 1$) quy tắc 3, cuối cùng là quy tắc 4, ta có:

$$S \models Sa^{m-1} \vdash Aaa^{m-1} \models a^{n-1}Ab^{n-1}a^m \vdash a^n b^n a^m.$$

Từ đó suy ra $L(G_1) = \{a^n b^n a^m \mid n \geq 1, m \geq 1\}$.

2) Cho văn phạm $G_2 = <\{0, 1\}, \{S\}, S, \{S\to SS, S\to 0S1, S\to 1S0, S\to\varepsilon\}$. Khi đó, G_2 không là văn phạm phi ngữ cảnh vì có quy tắc $S\to\varepsilon$ mà S lại xuất hiện ở vế phải của một quy tắc khác. Theo Bổ đề 1.2.6, G_2 tương đương với văn phạm

$$G_2' = <\{0, 1\}, \{S, S'\}, S', P'>,$$

$$P' = \{S\to SS, S\to 0S1, S\to 1S0, S\to\varepsilon, S'\to SS, S'\to 0S1, S'\to 1S0, S'\to\varepsilon\}.$$

Ở đây, G_2' cũng không là phi ngữ cảnh. Tuy nhiên, văn phạm G_2'' sau là phi ngữ cảnh mà tương đương với văn phạm G_2', nên cũng tương đương với văn phạm G_1.

$$G_2'' = <\{0, 1\}, \{S, S'\}, S', P'>,$$

$$P'' = \{S\to SS, S\to 0S1, S\to 1S0, S\to 01, S\to 10, S'\to SS, S'\to 0S1, S'\to 1S0, S'\to 01, S'\to 10, S'\to\varepsilon\}.$$

Từ các quy tắc của G_2, ta có được:

$L(G_2'')=L(G_2')=L(G_2)=\{\omega\in\{0, 1\}^* \mid$ số các chữ số 0 và 1 trong ω là bằng nhau$\}$.

1.2.10. Định nghĩa: Văn phạm $G = < \Sigma, \Delta, S, P >$ mà các quy tắc của nó có dạng $A\to aB$, $A\to a$, trong đó $A, B\in\Delta$, $a\in\Sigma$, $\omega\neq\varepsilon$, được gọi là văn phạm *chính quy* hay là văn phạm *nhóm 3*.

Các văn phạm mà các quy tắc của chúng có dạng trên, đồng thời chứa thêm quy tắc $S\to\varepsilon$, miễn sao ký hiệu đầu S không xuất hiện ở vế phải của bất kỳ quy tắc nào cũng được xếp vào lớp văn phạm nhóm 3.

Thí dụ 12: 1) Cho văn phạm:

$$G_1 = <\{1\}, \{S, A, B\}, S, \{S\to\varepsilon, S\to 1A, A\to 1B, B\to 1A, A\to 1\}.$$

Khi đó, G_1 là văn phạm chính quy và $L(G_1) = \{1^{2n} \mid n \geq 0\}$.

Thật vậy, sử dụng quy tắc 1, ta có $S\vdash 1^{2n}$, với n=0; sử dụng quy tắc 2, rồi n-1 lần ($n \geq 1$) liên tiếp cặp quy tắc 3 và 4, cuối cùng là quy tắc 5, ta có:

$$S\vdash 1A\vdash 11B\vdash 111A\vdash\ldots\models 1(1^{2n-2})A\vdash 1(1^{2n-2})1=1^{2n}.$$

2) Cho văn phạm $G_2 = <\{0, 1\}, \{S, A\}, S, \{S\to 0A, A\to 0A, A\to 1A, A\to 0\}>$. Khi đó, G_1 là văn phạm chính quy và $L(G_2) = \{0\omega 0 \mid \omega\in\{0, 1\}^*\}$.

Thật vậy, sử dụng quy tắc 1, rồi hữu hạn tuỳ ý có thể xen kẻ các quy tắc 2 và 3, cuối cùng là quy tắc 4, ta có:

$$S\vdash 0A\models 0\omega A\vdash 0\omega 0.$$

1.2.11. Định nghĩa: Từ các định nghĩa trên, ta thấy lớp văn phạm tổng quát là rộng nhất, nó chứa đựng các văn phạm cảm ngữ cảnh, lớp văn phạm cảm ngữ cảnh chứa các văn phạm phi ngữ cảnh và lớp văn phạm phi ngữ cảnh chứa các văn phạm chính quy. Hệ thống các lớp văn phạm này được gọi là *sự phân cấp Chomsky*.

Ngôn ngữ hình thức được gọi là ngôn ngữ tổng quát (t.ư. cảm ngữ cảnh, phi ngữ cảnh, chính quy) nếu tồn tại văn phạm tổng quát (t.ư. cảm ngữ cảnh, phi ngữ cảnh, chính quy) sinh ra nó. Vì vậy, đối với các lớp ngôn ngữ, ta có bao hàm thức:

{ngôn ngữ chính quy} $\subset$ {ngôn ngữ phi ngữ cảnh} $\subset$ {ngôn ngữ cảm ngữ cảnh}
$\subset$ {ngôn ngữ tổng quát}.

Ta cũng thấy về mặt cấu trúc ngữ pháp thì các quy tắc của các văn phạm phi ngữ cảnh và văn phạm chính quy là đơn giản hơn cả và chúng có nhiều ứng dụng trong việc thiết kế các ngôn ngữ lập trình và trong nghiên cứu về chương trình dịch… Vì vậy, trong các phần tiếp theo chúng ta dành thêm sự quan tâm tới hai lớp văn phạm đó.

Thí dụ 13: 1) Cho bảng chữ $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Khi đó các ngôn ngữ

$$L_1 = \{\omega = a_1a_2\ldots a_n\},\ L_2 = \Sigma^+,\ L_3 = \Sigma^*,\ L = \varnothing$$

là các ngôn ngữ chính quy trên bảng chữ Σ.

Thật vậy, $L_1 = L(G_1)$, $L_2 = L(G_2)$, $L_3 = L(G_3)$, $L_4 = L(G_4)$ trong đó

$G_1 = <\Sigma, \{S, A_1, \ldots, A_{n-1}\}, S, \{S\to a_1A_1, A_1\to a_2A_2, \ldots, A_{n-2}\to a_{n-1}A_{n-1}, A_{n-1}\to a_n\}>$,

$G_2 = <\Sigma, \{S\}, S, \{S\to aS, S\to a \mid a\in\Sigma\}>$,

$G_3 = <\Sigma, \{S, A\}, S, \{S\to\varepsilon, S\to a, S\to aA, A\to aA, A\to a \mid a\in\Sigma\}>$

$G_4 = <\Sigma, \{S\}, S, \{S\to aS \mid a\in\Sigma\}>$

là các văn phạm chính quy. (Riêng đối với G_4, nó làm việc không bao giờ dừng, tức là không có $\omega\in\Sigma^*$, $\omega\neq\varepsilon$ mà G_4 sinh ra, hay G_4 sinh ra ngôn ngữ $\varnothing$.)

2) Xét hai ngôn ngữ:

$L_5 = \{\omega\omega^R \mid \omega\in\{a, b\}^*\}$,

$L_6 = \{\omega\omega \mid \omega\in\{a, b\}^*\}$.

(ω^R còn được gọi là ảnh gương của ω và như ta đã biết nó nhận được bằng cách viết ω theo hướng ngược lại.)

Mặc dù xem qua thì L_5 và L_6 phức tạp như nhau, việc xác định L_5 bằng văn phạm đơn giản hơn rất nhiều. Thật vậy, L_5 sinh bởi văn phạm phi ngữ cảnh sau:

$G_5 = <\{a, b\}, \{S, A\}, S, \{S\to\varepsilon, S\to A, A\to aAa, A\to bAb, A\to aa, A\to bb\}>$.

Để sinh L_6, ta xét văn phạm cảm ngữ cảnh G_6 sau đây:

$G_6 = <\{a, b\}, \{S, A, B, C, D, E\}, S, P_6>$, trong đó

$P_6 = \{S\to ABC, AB\to aAD, AB\to bAE, Da\to aD, Db\to bD, Ea\to aE, Eb\to bE,$
$DC\to BaC, EC\to BbC, aB\to Ba, bB\to Bb, aAB\to a, bAB\to b, aC\to a, bC\to b, S\to\varepsilon\}$

và bằng việc sử dụng các quy tắc trên với phương pháp quy nạp, ta có được $L(G_6)=L_6$.

1.2.12. Bổ đề: Nếu L là ngôn ngữ cảm ngữ cảnh (t.ư. phi ngữ cảnh, chính quy) thì $L\cup\{\varepsilon\}$ và $L\setminus\{\varepsilon\}$ cũng vậy.

***Chứng minh*:** a) $L\cup\{\varepsilon\}$: -- $\varepsilon\in L$: $L\cup\{\varepsilon\}=L$.

-- $\varepsilon\notin L$: Giả sử $L=L(G)$, với $G = <\Sigma, \Delta, S, P>$ là văn phạm cảm ngữ cảnh (t.ư. phi ngữ cảnh, chính quy). Theo Bổ đề 1.2.6, tồn tại $G' = <\Sigma, \Delta\cup\{S'\}, S', P'>$ tương

đương và cùng nhóm với G mà S' không xuất hiện ở vế phải của bất kỳ một quy tắc nào trong P'. Khi đó văn phạm:

$$G'' = <\Sigma, \Delta \cup \{S'\}, S', P' \cup \{S' \to \varepsilon\}>$$

là cùng loại với G' và $L(G'')=L(G') \cup \{\varepsilon\}=L(G) \cup \{\varepsilon\}=L \cup \{\varepsilon\}$.

b) $L \setminus \{\varepsilon\}$: -- $\varepsilon \notin L$: $L \setminus \{\varepsilon\}=L$.

-- $\varepsilon \in L$: Giả sử $L=L(G)$, với $G = <\Sigma, \Delta, S, P>$ là văn phạm cảm ngữ cảnh (t.ư. phi ngữ cảnh, chính quy). Khi đó $S \to \varepsilon \in P$ và S không xuất hiện ở vế phải của bất kỳ một quy tắc nào trong P. Khi đó văn phạm $G' = <\Sigma, \Delta, S, P \setminus \{S \to \varepsilon\}>$ cùng nhóm với G và $L(G')=L(G) \setminus \{\varepsilon\}=L \{\varepsilon\} \setminus \{\varepsilon\}$.

1.3. MỘT SỐ TÍNH CHẤT CỦA NGÔN NGỮ.

1.3.1. Định nghĩa: Giả sử L_1 và L_2 là hai ngôn ngữ bất kỳ được sinh bởi văn phạm và $\circ$ là một phép toán nào đó trên lớp các ngôn ngữ. Nếu $L_1 \circ L_2$ là ngôn ngữ cũng được sinh bởi một văn phạm thì ta nói lớp ngôn ngữ do văn phạm sinh ra *đóng* đối với phép toán $\circ$.

1.3.2. Định lý: Lớp ngôn ngữ sinh ra bởi văn phạm là đóng đối với phép hợp.

***Chứng minh*:** Giả sử L_1, L_2 là các ngôn ngữ được sinh bởi văn phạm G_1, G_2 hay $L_1=L(G_1)$, $L_2=L(G_2)$. Ta chứng minh tồn tại văn phạm G sao cho $L(G)=L_1 \cup L_2$.

Giả sử $G_1 = <\Sigma_1, \Delta_1, S_1, P_1>$ và $G_2 = <\Sigma_2, \Delta_2, S_2, P_2>$. Không mất tính chất tổng quát giả thiết rằng $\Delta_1 \cap (\Sigma_2 \cup \Delta_2)=\Delta_2 \cap (\Sigma_1 \cup \Delta_1)=\varnothing$. Lấy $S \notin \Sigma_1 \cup \Delta_1 \cup \Sigma_2 \cup \Delta_2$.

Xét văn phạm $G = <\Sigma_1 \cup \Sigma_2, \Delta_1 \cup \Delta_2 \cup \{S\}, S, P>$, trong đó

$$P = (P_1 \setminus \{S_1 \to \varepsilon\}) \cup (P_2 \setminus \{S_2 \to \varepsilon\}) \cup \{S \to \varepsilon \mid S_1 \to \varepsilon \in P_1 \text{ hoặc } S_2 \to \varepsilon \in P_2\} \cup \cup \{S \to S_1, S \to S_2\}.$$

(Ở đây, ta hiểu rằng nếu $S_1 \to \varepsilon \in P_1$ (t.ư. $S_2 \to \varepsilon \in P_2$) thì S_1 (t.ư. S_2) không xuất hiện ở vế phải của bất kỳ một quy tắc nào trong P_1 (t.ư. P_2).)

a) $\omega \in L(G)$: -- $\omega=\varepsilon$: tồn tại $S \to \varepsilon \in P$, nên có $S_1 \to \varepsilon \in P_1$ hoặc $S_2 \to \varepsilon \in P_2$. Do đó $\omega=\varepsilon \in L_1 \cup L_2$.

-- $\omega \neq \varepsilon$: tồn tại suy dẫn $S \models_G \omega$ và giả sử suy dẫn này là $S \vdash_G \alpha \vdash_G \beta \vdash_G \ldots \vdash_G \omega$. Từ đó ta có $S \to \alpha \in P$ ($\alpha \neq \varepsilon$), nên có $\alpha=S_1$ hoặc $\alpha=S_2$. Nếu $\alpha=S_1$ thì dãy dẫn xuất $\alpha, \beta, \ldots, \omega$ là ở trong G_1, nên ta có $S_1 \vdash_{G_1} \beta \vdash_{G_1} \ldots \vdash_{G_1} \omega$, tức là $\omega \in L(G_1)$. Nếu $\alpha=S_2$ thì dãy dẫn xuất $\alpha, \beta, \ldots, \omega$ là ở trong G_2, nên ta có $S_2 \vdash_{G_2} \beta \vdash_{G_2} \ldots \vdash_{G_2} \omega$, tức là $\omega \in L(G_2)$. Do đó $\omega \in L_1 \cup L_2$.

b) $\omega \in L_1 \cup L_2$: -- $\omega=\varepsilon$: tồn tại $S_1 \to \varepsilon \in P_1$ hoặc $S_2 \to \varepsilon \in P_2$, nên có $S \to \varepsilon \in P$. Do đó $\omega=\varepsilon \in L(G)$.

-- $\omega \neq \varepsilon$: $\omega \in L_1$ hoặc $\omega \in L_2$. Nếu $\omega \in L_1$ (t.ư. $\omega \in L_2$) thì ta có suy dẫn $S_1 \models_{G_1} \omega$ (t.ư. $S_2 \models_{G_2} \omega$) với các quy tắc được sử dụng thuộc $P_1 \setminus \{S_1 \to \varepsilon\}$ (t.ư. $P_2 \setminus \{S_2 \to \varepsilon\}$). Khi đó, ta có $S \vdash_G S_1 \models_G \omega$ (t.ư. $S \vdash_G S_2 \models_G \omega$). Do đó $\omega \in L(G)$.

Vì vậy, $L(G)=L_1 \cup L_2$.

Tập quy tắc P của văn phạm G ở trên có thể được xác định như sau mà ta vẫn có $L(G)=L_1 \cup L_2$:

$P = (P_1 \setminus \{S_1 \to \varepsilon\}) \cup (P_2 \setminus \{S_2 \to \varepsilon\}) \cup \{S \to \alpha \mid S_1 \to \alpha \in P_1 \text{ hoặc } S_2 \to \alpha \in P_2\}$.

1.3.3. Hệ quả: Nếu L_1 và L_2 là hai ngôn ngữ chính quy (t.ư. phi ngữ cảnh, cảm ngữ cảnh) thì $L_1 \cup L_2$ cũng là ngôn ngữ chính quy (t.ư. phi ngữ cảnh, cảm ngữ cảnh).

Thí dụ 14: Cho hai ngôn ngữ $L_1=\{a^n cb^{2n} \mid n \geq 0\}$ và $L_2=\{a^{2n} cb^n \mid n \geq 0\}$ trên bảng chữ $\Sigma = \{a, b, c\}$. Khi đó, $L_1=L(G_1)$ và $L_2=L(G_2)$, trong đó

$G_1 = <\Sigma, \{S_1, A, B\}, S_1, \{S_1 \to AS_1B, S_1 \to c, A \to a, B \to bb\}>$,

$G_2 = <\Sigma, \{S_2, C, D\}, S_2, \{S_2 \to CS_2D, S_2 \to c, C \to aa, D \to b\}>$.

Thật vậy, trong G_1, sử dụng n lần ($n \geq 0$) quy tắc 1, sau đó sử dụng n lần quy tắc 3, n lần quy tắc 4 và quy tắc 2, ta có:

$$S_1 \overset{G_1}{\models} A^n S_1 B^n \overset{G_1}{\models} a^n c(bb)^n = a^n cb^{2n}.$$

Tương tự, trong G_2 ta có $S_2 \overset{G_2}{\models} a^{2n} cb^n$. Từ đó ta có văn phạm G và G':

$G = <\Sigma, \{S_1, A, B, S_2, C, D, S\}, S, P>$, $G' = <\Sigma, \{S_1, A, B, S_2, C, D, S\}, S, P'>$

trong đó $P = \{S_1 \to AS_1B, S_1 \to c, A \to a, B \to bb, S_2 \to CS_2D, S_2 \to c, C \to aa, D \to b, S \to S_1, S \to S_2\}$

và $P' = \{S_1 \to AS_1B, S_1 \to c, A \to a, B \to bb, S_2 \to CS_2D, S_2 \to c, C \to aa, D \to b, S \to AS_1B, S \to CS_2D, S \to c\}$.

Ở đây, G_1, G_2 là hai văn phạm phi ngữ cảnh, G, G' cũng là hai văn phạm phi ngữ cảnh và $L(G)=L(G')=L_1 \cup L_2=\{a^n cb^{2n}, a^{2n} cb^n \mid n \geq 0\}$.

1.3.4. Định lý: Lớp ngôn ngữ sinh ra bởi văn phạm là đóng đối với phép ghép.

***Chứng minh*:** Giả sử L_1, L_2 là các ngôn ngữ được sinh bởi văn phạm G_1, G_2 hay $L_1=L(G_1)$, $L_2=L(G_2)$. Ta chứng minh tồn tại văn phạm G sao cho $L(G)=L_1L_2$.

Giả sử $G_1 = <\Sigma_1, \Delta_1, S_1, P_1>$ và $G_2 = <\Sigma_2, \Delta_2, S_2, P_2>$. Không mất tính chất tổng quát giả thiết rằng $\Delta_1 \cap (\Sigma_2 \cup \Delta_2)=\Delta_2 \cap (\Sigma_1 \cup \Delta_1)=\varnothing$. Lấy $S \notin \Sigma_1 \cup \Delta_1 \cup \Sigma_2 \cup \Delta_2$.

Xét văn phạm $G = <\Sigma_1 \cup \Sigma_2, \Delta_1 \cup \Delta_2 \cup \{S\}, S, P>$, trong đó

$P = (P_1 \setminus \{S_1 \to \varepsilon\}) \cup (P_2 \setminus \{S_2 \to \varepsilon\}) \cup \{S \to S_1 \mid S_2 \to \varepsilon \in P_2\} \cup \{S \to S_2 \mid S_1 \to \varepsilon \in P_1\}$
$\cup \{S \to \varepsilon \mid S_1 \to \varepsilon \in P_1 \text{ và } S_2 \to \varepsilon \in P_2\} \cup \{S \to S_1S_2\}$.

(Ở đây, ta hiểu rằng nếu $S_1 \to \varepsilon \in P_1$ (t.ư. $S_2 \to \varepsilon \in P_2$) thì S_1 (t.ư. S_2) không xuất hiện ở vế phải của bất kỳ một quy tắc nào trong P_1 (t.ư. P_2).)

Với văn phạm G này, dễ dàng có được $L(G) \subset L_1L_2$ và $L_1L_2 \subset L(G)$.

Đặc biệt, khi G_1 và G_2 là hai văn phạm chính quy thì ta có thể xây dựng văn phạm chính quy G' như sau sao cho $L(G')=L_1L_2$.

a) $\varepsilon \notin L_1$ và $\varepsilon \notin L_2$ (tức là $S_1 \to \varepsilon \notin P_1$ và $S_2 \to \varepsilon \notin P_2$): Văn phạm chính quy G' cần tìm là $G' = <\Sigma_1 \cup \Sigma_2, \Delta_1 \cup \Delta_2, S_1, P'>$, trong đó

$$P' = (P_1 \setminus \{A \to a \mid A \to a \in P_1\}) \cup \{A \to aS_2 \mid A \to a \in P_1\} \cup P_2.$$

b) $\varepsilon \in L_1$ và $\varepsilon \notin L_2$: Đặt $L_1' = L_1 \setminus \{\varepsilon\}$ thì theo Bổ đề 1.2.12, ta xây dựng được văn phạm chính quy G_1' sao cho $L(G_1') = L_1'$. Khi đó theo a), ta có văn phạm G' sao cho $L(G') = L_1'L_2$. Từ $L_1L_2 = (L_1' \cup \{\varepsilon\})L_2 = L_1'L_2 \cup L_2$ và từ cách xây dựng văn phạm trong chứng minh của Định lý 1.3.2, ta tìm được văn phạm chính quy G'' sao cho $L(G'') = L_1L_2$.

c) $\varepsilon \notin L_1$ và $\varepsilon \in L_2$: Tương tự trường hợp b).

d) $\varepsilon \in L_1$ và $\varepsilon \in L_2$: Đặt $L_1' = L_1 \setminus \{\varepsilon\}$ và $L_2' = L_2 \setminus \{\varepsilon\}$ thì ta có:

$$L_1L_2 = (L_1' \cup \{\varepsilon\})(L_2' \cup \{\varepsilon\}) = L_1'L_2' \cup L_1' \cup L_2' \cup \{\varepsilon\}$$

và lập luận như trên ta tìm được văn phạm chính quy sinh ra ngôn ngữ L_1L_2.

1.3.5. Hệ quả: Nếu L_1 và L_2 là hai ngôn ngữ chính quy (t.ư. phi ngữ cảnh, cảm ngữ cảnh) thì L_1L_2 cũng là ngôn ngữ chính quy (t.ư. phi ngữ cảnh, cảm ngữ cảnh).

Thí dụ 15: 1) Cho hai ngôn ngữ $L_1 = \{a^n b^n \mid n \geq 1\}$ và $L_2 = \{c^n \mid n \geq 1\}$. Dễ dàng có được $L_1 = L(G_1)$ và $L_2 = L(G_2)$, trong đó

$G_1 = <\{a, b\}, \{S_1\}, S_1, \{S_1 \to aS_1b, S_1 \to ab\}>$,

$G_2 = <\{c\}, \{S_2\}, S_2, \{S_2 \to cS_2, S_2 \to c\}>$.

Từ đó ta nhận được văn phạm phi ngữ cảnh G:

$G = <\{a, b, c\}, \{S_1, S_2, S\}, S, \{S_1 \to aS_1b, S_1 \to ab, S_2 \to cS_2, S_2 \to c, S \to S_1S_2\}>$,

thoả mãn $L(G) = L_1L_2 = \{a^n b^n c^m \mid n \geq 1, m \geq 1\}$.

2) Cho hai ngôn ngữ chính quy $L_3 = \{ba^n \mid n \geq 0\}$ và $L_4 = \{b^n a \mid n \geq 0\}$. Ta có ngay $L_3 = L(G_3)$, $L_4 = L(G_4)$, trong đó G_3 và G_4 là hai văn phạm chính quy:

$G_3 = <\{a, b\}, \{S_1, A\}, S_1, \{S_1 \to b, S_1 \to bA, A \to aA, A \to a\}>$,

$G_4 = <\{a, b\}, \{S_2\}, S_2, \{S_2 \to bS_2, S_2 \to a\}>$.

Từ đó ta nhận được văn phạm chính quy G':

$G' = <\{a, b\}, \{S_1, A, S_2\}, S_1, P'>$,

$P' = \{S_1 \to bA, A \to aA, S_1 \to bS_2, A \to aS_2, S_2 \to bS_2, S_2 \to a\}$

thoả mãn $L(G') = L_3L_4 = \{ba^n b^m a \mid n \geq 0, m \geq 0\}$.

1.3.6. Định lý: Nếu L là ngôn ngữ chính quy thì lặp L^* của L cũng là ngôn ngữ chính quy. Nói một cách khác, lớp các ngôn ngữ chính quy đóng đối với phép toán một ngôi lặp.

***Chứng minh*:** Giả sử L=L(G), với $G = <\Sigma, \Delta, S, P>$ là văn phạm chính quy. Xét văn phạm $G' = <\Sigma, \Delta, S, P'>$, trong đó

$$P' = (P \setminus \{S \to \varepsilon\}) \cup \{A \to aS \mid A \to a \in P\}.$$

Khi đó G' là văn phạm chính quy. Ta chứng minh $L(G') = L^* \setminus \{\varepsilon\}$.

a) $\omega \in L(G')$: Ta có $S \overset{G}{\vDash} \omega$ với $\omega \neq \varepsilon$ vì $S \to \varepsilon \notin P'$. Không mất tính chất tổng quát, ta có thể giả thiết ký hiệu đầu S không xuất hiện ở vế phải của bất kỳ quy tắc nào trong P. Giả sử dãy dẫn xuất của ω có sử dụng n-1 quy tắc của P dạng $A \to aS$. Khi đó ta có:

$$S \models \omega_1{'}a_1S \models \omega_1\omega_2{'}a_2S \models \ldots \models \omega_1\omega_2\ldots\omega_{n-1}{'}a_{n-1}S \models \omega_1\omega_2\ldots\omega_{n-1}\omega_n = \omega,$$

ở đây, $\omega_i = \omega_i{'}a_i$. Như vậy, tồn tại các quy tắc $A_1 \to a_1S$, $A_2 \to a_2S$, …, $A_{n-1} \to a_{n-1}S$ trong P' và do đó tồn tại các quy tắc $A_1 \to a_1$, $A_2 \to a_2$, …, $A_{n-1} \to a_{n-1}$ trong P. Ta có $S \models \omega_1{'}A_1 \vdash \omega_1{'}a_1 = \omega_1$, $S \models \omega_2{'}A_2 \vdash \omega_2{'}a_2 = \omega_2$, …, $S \models \omega_{n-1}{'}A_{n-1} \vdash \omega_{n-1}{'}a_{n-1} = \omega_{n-1}$ là các suy dẫn trong G hay $\omega_1, \omega_2, \ldots, \omega_{n-1} \in L(G)$. Mặt khác suy dẫn $S \models \omega_n$ không sử dụng một quy tắc nào khác ngoài quy tắc của P, nên $\omega_n \in L(G)$. Vậy $\omega \in L^n \subset L^*$.

b) $\omega \in L^* \setminus \{\varepsilon\}$: Tồn tại số nguyên dương n sao cho $\omega \in L^n$ hay $\omega = \omega_1\omega_2\ldots\omega_{n-1}\omega_n$, trong đó $\omega_i \in L \setminus \{\varepsilon\}$, $1 \le i \le n$. Như vậy trong G có các suy dẫn

$$S \models \omega_i{'}A_i \vdash \omega_i{'}a_i = \omega_i$$

và cuối các suy dẫn này có sử dụng các quy tắc $A_i \to a_i$, $1 \le i \le n$, do đó ta có các quy tắc $A_i \to a_iS$ trong G'. Từ đó ta có suy dẫn trong G':

$$S \models \omega_1{'}a_1S \models \omega_1\omega_2{'}a_2S \models \ldots \models \omega_1\omega_2\ldots\omega_{n-1}{'}a_{n-1}S \models \omega_1\omega_2\ldots\omega_{n-1}\omega_n = \omega$$

hay $\omega \in L(G')$.

Cuối cùng, theo Bổ đề 1.2.12, L(G) chính quy kéo theo $L(G) \cup \{\varepsilon\}$ cũng chính quy.

1.3.7. Mệnh đề: Mọi ngôn ngữ hữu hạn đều là ngôn ngữ chính quy.

***Chứng minh*:** Ngôn ngữ hữu hạn là hợp hữu hạn của các ngôn ngữ một từ, nên từ Thí dụ 13 (ngôn ngữ một từ là chính quy) và từ Hệ quả 1.3.3 (hợp hữu hạn của các ngôn ngữ chính quy là chính quy), ta có ngôn ngữ hữu hạn là chính quy.

Thí dụ 16: Cho ngôn ngữ chính quy L={0, 01, 011, 0111}. Khi đó L sinh bởi văn phạm chính quy $G = <\{0, 1\}, \{S, A, B, C\}, S, P>$, trong đó

$$P = \{S \to 0, S \to 0A, A \to 1, A \to 1B, B \to 1, B \to 1C, C \to 1\}.$$

Văn phạm chính quy $G' = <\{0, 1\}, \{S, A, B, C\}, S, P'>$, trong đó

$$P' = \{S \to 0, S \to 0S, S \to 0A, A \to 1, A \to 1S, A \to 1B, B \to 1, B \to 1S, B \to 1C, C \to 1, C \to 1S\}$$

sinh ra ngôn ngữ $L^* \setminus \{\varepsilon\}$. Do đó văn phạm sinh ra ngôn ngữ L^* là:

$G'' = <\{0, 1\}, \{S', S, A, B, C\}, S', P''>$, trong đó

$$P'' = \{S \to 0, S' \to 0, S \to 0S, S' \to 0S, S \to 0A, S' \to 0A, A \to 1, A \to 1S, A \to 1B, B \to 1, B \to 1S, B \to 1C, C \to 1, C \to 1S, S' \to \varepsilon\}.$$

BÀI TẬP CHƯƠNG I:

1. Tìm các ngôn ngữ L_1, L_2, L_3 sao cho:

a) $(L_1L_2)^* \neq L_1^*L_2^*$.

b) $(L_1 \cap L_2)^* \neq L_1^* \cap L_2^*$.

c) $(L_1 \cup L_2)^* \neq L_1^* \cup L_2^*$.

d) $L_1(L_2^* \cap L_3^*) \neq (L_1L_2^*) \cap (L_1L_3^*)$.

2. Cho L_1, L_2, L_3 là các ngôn ngữ trên bảng chữ Σ. Chứng minh rằng:

a) $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$, $(L_2 \cup L_3)L_1 = L_2L_1 \cup L_3L_1$.

b) $(L_1^*L_2^*)^* = (L_1 \cup L_2)^*$.

3. Hãy xác định xem các văn phạm dưới đây sinh ra các ngôn ngữ nào?

a) $G = <\{0, 1\}, \{S, A\}, S, \{S \to 0A, A \to 1S, S \to \varepsilon\}>$.

b) $G = <\{a, b\}, \{S\}, S, \{S \to SaS, S \to b\}>$.

c) $G = <\{a, b, c\}, \{S\}, S, \{S \to aca, S \to bcb, S \to aSa, S \to bSb\}>$.

d) $G = <\{0, 1, 2, \ldots, 9\}, \{S, A\}, S, \{S \to SA \mid A, A \to 0|1|2|3|4|5|6|7|8|9\}>$.

4. Hãy thành lập các văn phạm sinh ra các ngôn ngữ dưới đây:

a) $L = \{\omega \in \{a\}^* \mid |\omega| \bmod 3 = 0\}$.

b) $L = \{a^{2n+1} \mid n \geq 0\}$.

c) $L = \{\omega \in \{a, b\}^* \mid n_a(\omega) > n_b(\omega)\}$ ($n_x(\omega)$ là số chữ cái x có mặt trong ω}.

d) $L = \{a^m b^n \mid n \geq 0, m \geq n\}$.

5. Hãy xây dựng các văn phạm chính quy sinh ra các ngôn ngữ dưới đây trên bảng chữ $\Sigma = \{0, 1\}$:

a) $L = \{0\omega 1 \mid \omega \in \Sigma^*\}$.

b) $L = \{\omega \in \Sigma^* \mid$ trong ω chứa đúng một chữ số 1}.

c) $L = \{\omega \in \Sigma^* \mid$ trong ω chỉ chứa một số lẻ chữ số 0}.

d) $L = \{\omega \in \Sigma^* \mid$ ω không kết thúc bằng hai chữ số 1}.

6. Hãy tìm văn phạm phi ngữ cảnh sinh ra ngôn ngữ sau:

$$L = \{a^n b^m \mid n \geq 0, m \geq 0, n \neq m\}.$$

7. Hãy xây dựng các văn phạm chính quy sinh ra các ngôn ngữ dưới đây:

a) $L = \{(baa)^m (aab)^n \mid m \geq 1, n \geq 1\}$.

b) $L = \{1\}^* \{010\} \{0\}^*$.

c) $L = \{010\}^* \cup \{1100\}^*$.

d) $L = \{a^m b^n c^k \mid m \geq 0, n \geq 0, k \geq 0\}$.

8. Chứng minh rằng lớp ngôn ngữ sinh bởi văn phạm là đóng đối với phép giao.

CHƯƠNG II:
ÔTÔMAT HỮU HẠN
VÀ NGÔN NGỮ CHÍNH QUY

2.1. ÔTÔMAT HỮU HẠN.

2.1.1. Mở đầu:

Một ôtômat hữu hạn là một mô hình tính toán thực sự hữu hạn. Mọi cái liên quan đến nó đều có kích thước hữu hạn cố định và không thể mở rộng trong suốt quá trình tính toán. Các loại ôtômat khác được nghiên cứu sau này có ít nhất một bộ nhớ vô hạn về tiềm năng. Sự phân biệt giữa các loại ôtômat khác nhau chủ yếu dựa trên việc thông tin có thể được đưa vào bộ nhớ như thế nào.

Một ôtômat hữu hạn làm việc theo thời gian rời rạc như tất cả các mô hình tính toán chủ yếu. Như vậy, ta có thể nói về thời điểm "kế tiếp" khi "đặc tả" hoạt động của một ôtômat hữu hạn.

Trường hợp đơn giản nhất là thiết bị không có bộ nhớ mà ở mỗi thời điểm, thông tin ra chỉ phụ thuộc vào thông tin vào lúc đó. Các thiết bị như vậy là mô hình của các mạch tổ hợp.

Tuy nhiên, nói chung, thông tin ra sản sinh bởi một ôtômat hữu hạn phụ thuộc vào cả thông tin vào hiện tại lẫn các thông tin vào trước đó. Như vậy ôtômat có khả năng (với một phạm vi nào đó) ghi nhớ các thông tin vào trong quá khứ của nó. Một cách chi tiết hơn, điều đó có nghĩa như sau.

Ôtômat có một số hữu hạn trạng thái bộ nhớ trong. Tại mỗi thời điểm i, nó ở một trong các trạng thái đó, chẳng hạn q_i. Trạng thái q_{i+1} ở thời điểm sau được xác định bởi q_i và thông tin vào a_i cho ở thời điểm i. Thông tin ra ở thời điểm i được xác định bởi trạng thái q_i (hay bởi cả a_i và q_i).

2.1.2. Định nghĩa: Một ôtômat hữu hạn đơn định hay một DFA (Deteministic Finite Automata) là một bộ năm

$$A = <Q, \Sigma, \delta, q_0, F>,$$

trong đó:

– Q là một tập hữu hạn khác rỗng, được gọi là tập các trạng thái;

– Σ là một bảng chữ, được gọi là bảng chữ vào;

– δ: $D \longrightarrow Q$, trong đó $D \subset Q \times \Sigma$, được gọi là ánh xạ chuyển;

– $q_0 \in Q$, được gọi là trạng thái đầu;

– $F \subset Q$, được gọi là tập các trạng thái kết thúc.

Trong trường hợp $D=Q \times \Sigma$, ta nói A là đầy đủ. Về sau ta sẽ thấy rằng mọi ôtômat hữu hạn đều đưa về được ôtômat hữu hạn đầy đủ tương đương.

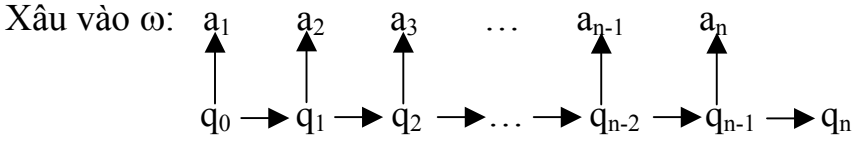
Hoạt động của ôtômat hữu hạn đơn định A = <Q, Σ, δ, q_0, F> khi cho xâu vào ω=$a_1a_2\ldots a_n$ có thể được mô tả như sau:

Khi bắt đầu làm việc, máy ở trạng thái đầu q_0 và đầu đọc đang nhìn vào ô có ký hiệu a_1. Tiếp theo máy chuyển từ trạng thái q_0 dưới tác động của ký hiệu vào a_1 về trạng thái mới $\delta(q_0, a_1)=q_1\in Q$ và đầu đọc chuyển sang phải một ô, tức là nhìn vào ô có ký hiệu a_2. Sau đó ôtômat A có thể lại tiếp tục chuyển từ trạng thái q_1 nhờ ánh xạ chuyển δ về trạng thái mới $q_2=\delta(q_1, a_2)\in Q$. Quá trình đó sẽ tiếp tục cho tới khi gặp một trong các tình huống sau:

– Trong trường hợp ôtômat A đọc hết xâu vào ω và $\delta(q_{n-1},a_n)=q_n\in F$, ta nói rằng A đoán nhận ω.

– Trong trường hợp ôtômat A đọc hết xâu vào ω và $\delta(q_{n-1},a_n)=q_n\notin F$ hoặc tồn tại chỉ số j ($j\le n$) sao cho $\delta(q_{j-1},a_j)$ không xác định, ta nói rằng A không đoán nhận ω.

Q. Khi đó ôtômat dừng lại. Nếu $q_n\in F$ thì ta nói rằng ôtômat đã đoán nhận xâu ω.



2.1.3. Phương pháp biểu diễn ôtômat hữu hạn đơn định:

Ánh xạ chuyển là một bộ phận quan trọng của một ôtômat hữu hạn đơn định. Nó có thể cho dưới dạng bảng chuyển hoặc cho dưới dạng đồ thị.

1) Phương pháp cho bảng chuyển:

Trạng thái	Ký hiệu vào			
	a_1	a_2	……..	a_n
q_1	$\delta(q_1,a_1)$	$\delta(q_1,a_2)$	……	$\delta(q_1,a_2)$
q_2	$\delta(q_2,a_1)$	$\delta(q_2,a_2)$	……	$\delta(q_2,a_2)$
q_3	$\delta(q_3,a_1)$	$\delta(q_3,a_2)$	……	$\delta(q_3,a_2)$
…	……...			
q_m	$\delta(q_m,a_1)$	$\delta(q_m,a_2)$	……	$\delta(q_m,a_2)$

trong đó dòng i cột j của bảng là ô trống nếu $(q_i,a_j)\notin D$, tức là $\delta(q_i,a_j)$ không xác định.

2) Phương pháp cho bằng đồ thị chuyển:

Cho ôtômat A = <Q, Σ, δ, q_0, F>. Ánh xạ chuyển δ có thể cho bằng một đa đồ thị có hướng, có khuyên G sau đây, được gọi là đồ thị chuyển của ôtômat A. Tập đỉnh của G là Q. Nếu $a\in\Sigma$ và từ trạng thái q chuyển sang trạng thái p do đẳng thức $\delta(q, a)=p$ thì sẽ có một cung từ q tới p được gán nhãn a.

Đỉnh vào của đồ thị chuyển là đỉnh ứng với trạng thái ban đầu q_0. Các đỉnh sẽ được khoanh bởi các vòng tròn, tại đỉnh q_0 có mũi tên đi vào, riêng đỉnh với trạng thái kết thúc được khoanh bởi vòng tròn đậm.

Thí dụ 1: Cho hai ôtômat hữu hạn đơn định

$$A_1 = <\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\}>,$$

trong đó $\delta(q_0, a)=q_0$, $\delta(q_0, b)=q_1$, $\delta(q_1, a)=q_0$, $\delta(q_1, b)=q_2$, $\delta(q_2, a)=q_2$, $\delta(q_2, b)=q_2$ và

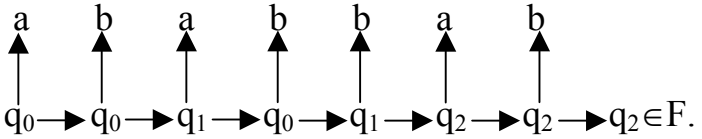
$$A_2 = <\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\}>,$$

trong đó $\delta(q_0, 0)=q_2$, $\delta(q_0, 1)=q_1$, $\delta(q_1, 0)=q_3$, $\delta(q_1, 1)=q_0$, $\delta(q_2, 0)=q_0$, $\delta(q_2, 1)=q_3$, $\delta(q_3, 0)=q_1$, $\delta(q_3, 1)=q_2$. Khi đó các bảng chuyển của A_1 và A_2 là:

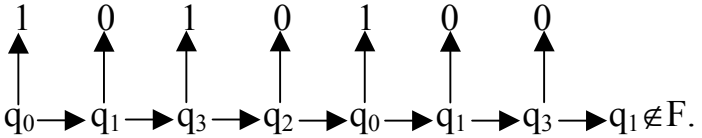
Trạng thái	Ký hiệu vào a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_2

Trạng thái	Ký hiệu vào 0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

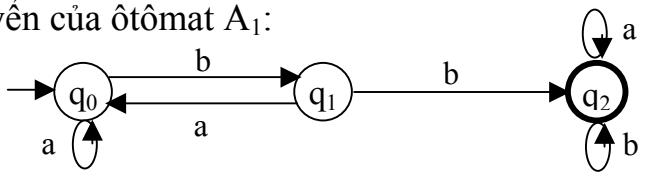
Dãy trạng thái của ôtômat A_1 khi cho xâu α=ababbab vào là:



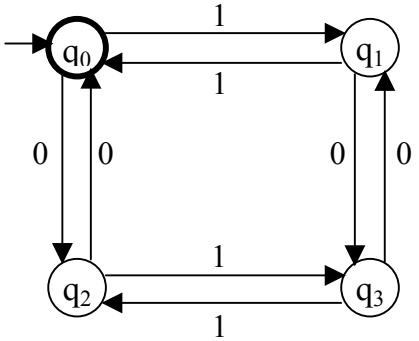
Dãy trạng thái của ôtômat A_2 khi cho xâu β=1010100 vào là:



Đồ thị chuyển của ôtômat A_1:



Đồ thị chuyển của ôtômat A_2:



Ta có thể mô tả quá trình đoán nhận xâu vào của ôtômat hữu hạn đơn định đầy đủ A bằng thuật toán mô phỏng sau:

Đầu vào:

– Một xâu ω, kết thúc bởi ký hiệu hết File là eof.

– Một ôtômat hữu hạn đơn định đầy đủ A với trạng thái đầu q_0 và tập trạng thái kết thúc là F.

Đầu ra: "Đúng" nếu A đoán nhận xâu ω.

"Sai" nếu A không đoán nhận xâu ω.

Thuật toán:

```
Begin
    S:=q0;
    C:=ký hiệu tiếp theo;
    While C < > eof do
        begin
            S:=δ(S, C);
            C:=ký hiệu tiếp theo;
        end;
    if S in F return (True)
    else return (False);
End.
```

Để mô tả hình thức quá trình đoán nhận một từ (xâu vào), người ta đưa vào ánh xạ mở rộng δ' từ tập con của $Q \times \Sigma^*$ vào Q như trong định nghĩa sau.

2.1.4. Định nghĩa: Cho ôtômat hữu hạn đơn định $A = <Q, \Sigma, \delta, q_0, F>$. Mở rộng δ' của δ là một ánh xạ từ tập con của $Q \times \Sigma^*$ vào Q được xác định như sau:

1) $\delta'(q, \varepsilon)=q, \forall q \in Q$,

2) $\delta'(q, \omega a)=\delta(\delta'(q, \omega), a), \forall a \in \Sigma, \forall q \in Q, \forall \omega \in \Sigma^*$ sao cho $\delta'(q, \omega)$ được xác định.

Ta có $\delta'(q, a)=\delta'(q, \varepsilon a) = \delta(\delta'(q, \varepsilon), a) = \delta(q, a), \forall a \in \Sigma, \forall q \in Q$. Do đó trên $Q \times \Sigma$, ta có thể đồng nhất δ' với δ. Nếu không cần phân biệt, từ đây về sau ta viết δ thay cho δ'.

2.1.5. Định nghĩa: Cho ôtômat hữu hạn đơn định $A = <Q, \Sigma, \delta, q_0, F>$, $\omega \in \Sigma^*$ và L là một ngôn ngữ trên Σ. Ta nói:

– ω được đoán nhận bởi A nếu $\delta(q_0, \omega) \in F$;

– L được đoán nhận bởi A nếu $L=\{\omega \in \Sigma^* \mid \delta(q_0, \omega) \in F\}$ và ký hiệu L là T(A).

Lưu ý rằng trong đồ thị chuyển của A, $\omega \in \Sigma^*$ được đoán nhận bởi A khi và chỉ khi ω ứng với một đường đi từ đỉnh q_0 đến một trong các đỉnh kết thúc. Cụ thể là nếu $\omega=a_1a_2\ldots a_n$ thì đường đi là $(q_0, q_1, \ldots, q_n)$ với cung (q_{i-1}, q_i) có nhãn a_i

($1 \le i \le n$) và $q_n \in F$. Như vậy, T(A) là tập hợp tất cả các đường đi từ q_0 đến các đỉnh kết thúc.

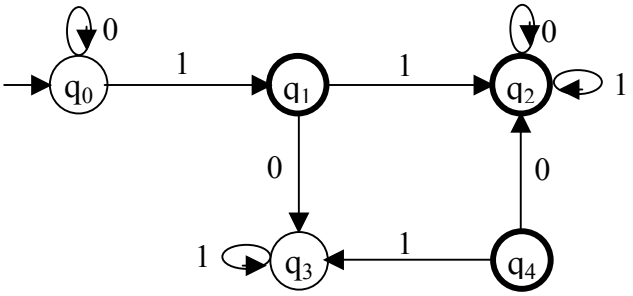
2.1.6. Định nghĩa: Hai ôtômat hữu hạn đơn định A và A' được gọi là tương đương nếu T(A)=T(A').

Thí dụ 2: Cho ôtômat hữu hạn đơn định:

$$A = <\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_1, q_2, q_4\}>,$$

trong đó $\delta(q_0,0)=q_0$, $\delta(q_0,1)=q_1$, $\delta(q_1,0)=q_3$, $\delta(q_1,1)=q_2$, $\delta(q_2,0)=q_2$, $\delta(q_2,1)=q_2$, $\delta(q_3,1)=q_3$, $\delta(q_4,0)=q_2$, $\delta(q_4,1)=q_3$.

Đồ thị chuyển của A là:

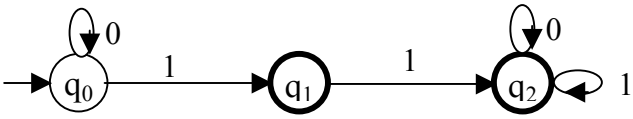


Trước hết, ta nhận thấy rằng không có đường đi từ q_0 đến đỉnh kết thúc q_4, do đó ôtômat A tương đương với ôtômat A' sau:

$$A' = <\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1, q_2\}>,$$

trong đó $\delta(q_0,0)=q_0$, $\delta(q_0,1)=q_1$, $\delta(q_1,1)=q_2$, $\delta(q_2,0)=q_2$, $\delta(q_2,1)=q_2$.

Đồ thị chuyển của A' là:



Các đường đi từ q_0 đến đỉnh kết thúc q_1 ứng với các xâu 0^n1, $n \ge 0$. Các đường đi từ q_0 đến đỉnh kết thúc q_2 ứng với các xâu $0^n11\omega$, $n \ge 0$, $\omega \in \{0, 1\}^*$. Vậy

$$T(A)=T(A')=\{0^n1, 0^n11\omega \mid n \ge 0, \omega \in \{0, 1\}^*\}.$$

2.1.7. Bổ đề: Cho ôtômat hữu hạn đơn định $A = <Q, \Sigma, \delta, q_0, F>$. Khi đó $\forall \omega_1, \omega_2 \in \Sigma^*$, $\forall q \in Q$ sao cho $\delta(q, \omega_1\omega_2)$ xác định, ta có:

$$\delta(q, \omega_1\omega_2) = \delta(\delta(q, \omega_1), \omega_2).$$

Chứng minh: Ta chứng minh đẳng thức trên bằng quy nạp theo độ dài của ω_2. Khi $d(\omega_2)=0$ hay $\omega_2=\varepsilon$, ta có $\delta(\delta(q, \omega_1), \varepsilon)=\delta(q, \omega_1)=\delta(q, \omega_1\varepsilon)$. Giả sử đẳng thức đúng với mọi ω_2 có độ dài $\le n$. Với ω'_2 có độ dài n+1, ta có $\omega'_2=\omega_2 a$, với $\omega_2 \in \Sigma^*$, $d(\omega_2)=n$, $a \in \Sigma$ và $\delta(q, \omega_1\omega'_2)=\delta(q, \omega_1\omega_2 a)=\delta(\delta(q, \omega_1\omega_2), a)=\delta(\delta(\delta(q, \omega_1), \omega_2), a)=\delta(\delta(q, \omega_1), \omega_2 a)=\delta(\delta(q, \omega_1), \omega'_2)$. Do đó đẳng thức đúng đến n+1.

2.1.8. Định lý: Nếu L là ngôn ngữ được đoán nhận bởi ôtômat hữu hạn đơn định thì tồn tại số tự nhiên n sao cho với mọi $\alpha \in L$ có $d(\alpha) \ge n$ đều có thể phân tích dưới dạng $\alpha=uvw$, trong đó $d(uv) \le n$, $d(v) \ge 1$ và với mọi $i \in \mathbf{N}$, ta có $uv^iw \in L$.

Chứng minh: Giả sử L=T(A) , với A = <Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định. Gọi n=|Q|. Ta chứng minh n là số tự nhiên cần tìm.

Cho $\alpha=a_1a_2\ldots a_m\in L$ với m≥n. Khi đó $\exists q_1, \ldots, q_m\in Q$ sao cho $\delta(q_{i-1}, a_i)=q_i$, $1\le i\le m$ và $q_m\in F$. Do m≥n nên trong dãy $q_0, q_1, \ldots, q_m$ có ít nhất hai trạng thái trùng nhau. Gọi k là số nhỏ nhất sao cho tồn tại i (i<k≤n) để $q_i=q_k$.

Đặt $u=a_1\ldots a_i$, $v=a_{i+1}\ldots a_k$, $w=a_{k+1}\ldots a_m$. Ta có $\alpha=uvw$, d(uv)=k≤n, d(v)≥1 (do i<k). Ngoài ra,

$\delta(q_0, u)=q_i=q_k=\delta(q_0, uv)$,

$\delta(q_0, u)=\delta(q_0, uv)=\delta(\delta(q_0, u), v)=\delta(\delta(q_0, uv), v)=\delta(q_0, uv^2)$,

$\delta(q_0, u)=\delta(q_0, uv^2)=\delta(\delta(q_0, u), v^2)=\delta(\delta(q_0, uv), v^2)=\delta(q_0, uv^3)$,

tiếp tục ta được $\delta(q_0, u)=\delta(q_0, uv^i)$, $\forall i\in\mathbf{N}$. Cuối cùng ta có:

$\delta(q_0, uv^iw)=\delta(\delta(q_0, uv^i), w)=\delta(\delta(q_0, uv), w)=\delta(q_0, uvw)\in F$.

Vậy $uv^iw\in L$, $\forall i\in\mathbf{N}$.

2.1.9. Hệ quả: Cho A là ôtômat hữu hạn đơn định có n trạng thái và L là ngôn ngữ được đoán nhận bởi A. Khi đó L≠∅ khi và chỉ khi $\exists\omega\in L$ sao d(ω)<n.

Chứng minh: Điều kiện đủ là hiển nhiên. Bây giờ cho L≠∅. Giả sử mọi từ trong L đều có độ dài ≥n. Gọi α là từ có độ dài nhỏ nhất trong L (d(α)≥n). Theo Định lý 2.1.8, ta có α=uvw, trong đó d(uv)≤n, d(v)≥1 và với mọi $i\in\mathbf{N}$, ta có $uv^iw\in L$. Với i=0, $\alpha=uw\in L$ mà d(uw)<d(α). Điều này mâu thuẫn với tính nhỏ nhất củad(α). Vậy tồn tại $\omega\in L$ sao d(ω)<n.

2.1.10. Hệ quả: Tồn tại một ngôn ngữ phi ngữ cảnh mà không được đoán nhận bởi bất kỳ một ôtômat hữu hạn đơn định nào.

Chứng minh: Cho ngôn ngữ $L=\{a^nb^n \mid n\ge1\}$ trên bảng chữ Σ={a, b}. Ta có L=L(G), trong đó G=<Σ, {S}, S, {S→aSb, S→ab}> là văn phạm phi ngữ cảnh. Giả sử L=T(A) với A=<Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định. Với n đủ lớn, $\alpha=a^nb^n$ có d(α)≥|Q|. Theo Định lý 2.1.8, $a^nb^n=uvw$, trong đó d(uv)≤|Q|, d(v)≥1, $uv^iw\in L$, $\forall i\in\mathbf{N}$.

– Nếu $n_a(v)>0$ và $n_b(v)=0$ thì với i đủ lớn $n_a(v)>n_b(v)$.

– Nếu $n_b(v)>0$ và $n_a(v)=0$ thì với i đủ lớn $n_b(v)>n_a(v)$.

– Nếu $n_a(v)>0$ và $n_b(v)>0$ thì với i=2 ta có a và b xen kẻ nhau trong uv^iw.

Cả ba trường hợp đều mâu thuẫn với $uv^iw\in L$. Vậy không tồn tại một ôtômat hữu hạn đơn định nào đoán nhận A.

Về sau, ta sẽ thấy rằng điều kiện cần và đủ để một ngôn ngữ được đoán nhận bởi một ôtômat hữu hạn đơn định là chính quy. Do đó hệ quả trên cho biết tồn tại một ngôn ngữ phi ngữ cảnh mà không là ngôn ngữ chính quy, tức là lớp các ngôn ngữ chính quy là con thực sự của lớp các ngôn ngữ phi ngữ cảnh.

2.1.11. Chú ý: Với ôtômat hữu hạn đơn định A=<Q, Σ, δ, q_0, F> bất kỳ, ta luôn có thể xây dựng một ôtômat hữu hạn đơn định đầy đủ A' tương đương với A.

Thật vậy, lấy S∉Q (do đó S∉F), đặt Q'=Q∪{S} và δ': Q' x Σ ⟶ Q' xác định bởi: ∀q∈Q, ∀a∈Σ, δ'(q, a)=δ(q, a) nếu δ(q, a) được xác định, δ'(q, a)=S nếu δ(q, a) không được xác định và δ'(S, a)=S. Khi đó A'=<Q',Σ, δ', q_0, F> là ôtômat hữu hạn đơn định đầy đủ mà T(A')=T(A).

2.1.12. Định nghĩa: Một ôtômat hữu hạn không đơn định hay một NDFA (Nondeteministic Finite Automata) là một bộ năm

$$A = <Q, \Sigma, \delta, q_0, F>,$$

trong đó Q, Σ, q_0, F như trong Định nghĩa 2.1.2 và δ: Q x Σ ⟶ $\mathcal{P}$(Q) ($\mathcal{P}$(Q) là tập hợp các tập con của Q) gọi là ánh xạ chuyển.

Trong trường hợp δ(q, a)≠∅, ∀q∈Q, ∀a∈Σ, ta nói A là đầy đủ.

Nếu δ(q, a)={p_1, p_2, …, p_k} thì ta nói rằng ôtômat A ở trạng thái q gặp ký hiệu a thì có thể chuyển đến một trong các trạng thái p_1, p_2, …, p_k. Nếu δ(q, a)={p} thì ở trạng thái q gặp ký hiệu a, ôtômat A chỉ chuyển đến một trạng thái duy nhất p. Nếu δ(q, a)=∅ thì ở trạng thái q gặp ký hiệu a, ôtômat A không thể chuyển đến trạng thái nào. Trường hợp này tương tự như δ(q, a) không xác định của ôtômat hữu hạn đơn định. Như vậy, ta thấy rằng một ôtômat hữu hạn đơn định là một trường hợp đặc biệt của một ôtômat hữu hạn không đơn định.

Hoạt động của ôtômat hữu hạn không đơn định A = <Q, Σ, δ, q_0, F> khi cho xâu vào ω=a_1a_2… a_n có thể được mô tả như sau:

Khi bắt đầu làm việc, máy ở trạng thái đầu q_0 và đầu đọc đang nhìn vào ô có ký hiệu a_1. Từ trạng thái q_0, dưới tác động của ký hiệu vào a_1, δ(q_0, a_1)={p_1,…, p_k}, máy xác định các trạng thái có thể tiếp theo là p_1, …, p_k và đầu đọc chuyển sang phải một ô, tức là nhìn vào ô có ký hiệu a_2. Tiếp tục với mỗi p_i (1≤i≤k) và ký hiệu tiếp theo là a_2, các trạng thái tiếp theo có thể đến được là δ(p_1, a_2)∪…∪δ(p_k, a_2). Quá trình đó sẽ tiếp tục cho tới khi gặp một trong các tình huống sau:

– Trong trường hợp tập trạng thái tiếp theo sau khi đọc a_j nào đó là rỗng hoặc sau khi đọc ký hiệu a_n là Q' mà Q'∩F=∅, ta nói rằng A không đoán nhận ω.

– Trong trường hợp tập trạng thái tiếp theo sau khi đọc ký hiệu a_n là Q' mà Q'∩F≠∅, ta nói rằng A đoán nhận ω.

Một ôtômat hữu hạn không đơn định có thể biểu diễn dưới dạng bảng chuyển hoặc đồ thị chuyển như trong trường hợp ôtômat hữu hạn đơn định.

Nếu δ(q, a)={p_1, p_2, …, p_k} thì trong đồ thị chuyển có k cung từ q sang p_1, …, p_k cùng một nhãn a.

Quá trình đoán nhận một xâu vào ω của một ôtômat hữu hạn không đơn định A có thể biểu diễn bằng một cây có gốc mà gốc là trạng thái đầu q_0. Trong cây này,

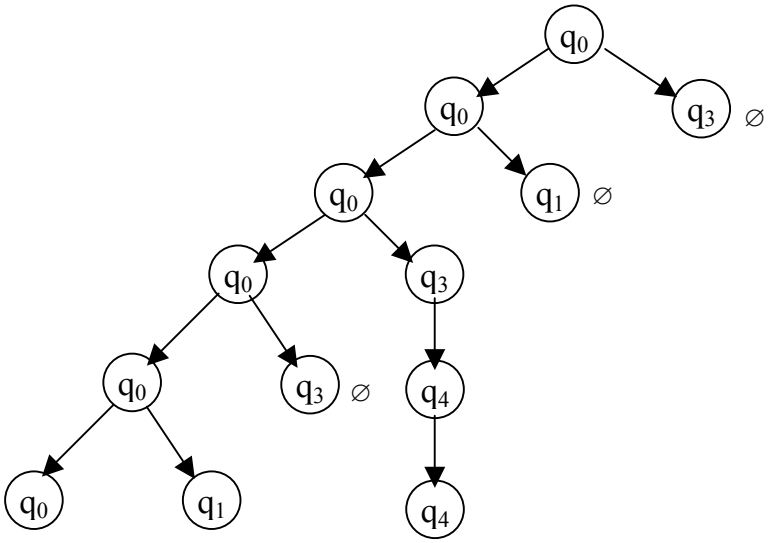
nếu có một đường đi qua một dãy các trạng thái ứng với xâu vào ω từ q_0 đến một lá chứa trạng thái kết thúc thì xâu vào này được đoán nhận bởi ôtômat A. Ngược lại, nếu không có một lá nào trong cây chứa trạng thái kết thúc thì ω không được đoán nhận bởi A.

Thí dụ 3: Cho ôtômat hữu hạn không đơn định:

$$A = <\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\}>,$$

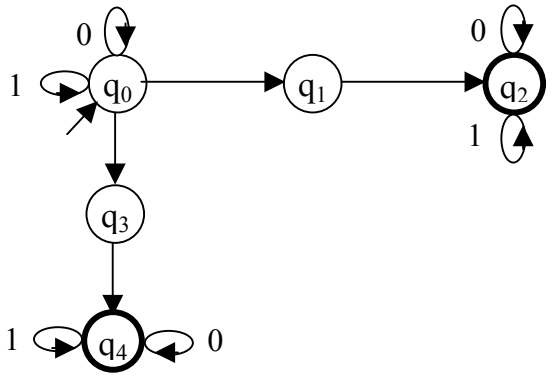
trong đó $\delta(q_0,0)=\{q_0,q_3\}$, $\delta(q_0, 1)=\{q_0,q_1\}$, $\delta(q_1, 0)=\varnothing$, $\delta(q_1, 1)=\{q_2\}$, $\delta(q_2, 0)=\{q_2\}$, $\delta(q_2, 1)=\{q_2\}$, $\delta(q_3, 0)=\{q_4\}$, $\delta(q_3, 1)=\varnothing$, $\delta(q_4, 0)=\{q_4\}$, $\delta(q_4, 1)=\{q_4\}$.

Cho xâu vào ω=01001. Ta có cây đoán nhận ω như sau:



Trong cây trên có một đường đi từ q_0 đến $q_4 \in F$ nên xâu ω=01001 là xâu được đoán nhận bởi ôtômat A.

Đồ thị chuyển của ôtômat A là:



2.1.13. Định nghĩa: Cho ôtômat hữu hạn không đơn định $A = <Q, \Sigma, \delta, q_0, F>$. Mở rộng của δ là ánh xạ δ' từ tập $Q \times \Sigma^*$ vào $\mathcal{P}(Q)$ được xác định như sau:

1) $\delta'(q, \varepsilon)=\{q\}$, $\forall q \in Q$,

2) $\delta'(q, \omega a)= \bigcup_{p \in \delta'(q,\omega)} \delta(p,a)$, $\forall q \in Q$, $\forall \omega \in \Sigma^*$, $\forall a \in \Sigma$.

Ta có $\delta'(q, a)=\delta'(q, \varepsilon a)=\bigcup_{p\in\delta'(q,\varepsilon)}\delta(p,a)=\delta(q, a)$, $\forall q\in Q$, $\forall a\in\Sigma$. Vì vậy, cũng như trường hợp ôtômat hữu hạn đơn định, ta có thể sử dụng ký hiệu δ thay cho δ'.

2.1.14. Định nghĩa: Cho ôtômat hữu hạn không đơn định $A = <Q, \Sigma, \delta, q_0, F>$, $\omega\in\Sigma^*$ và L là một ngôn ngữ trên Σ. Ta nói:

– ω được đoán nhận bởi A nếu $\delta(q_0, \omega)\cap F\neq\varnothing$;

– L được đoán nhận bởi A nếu $L=\{\omega\in\Sigma^* \mid \delta(q_0, \omega)\cap F\neq\varnothing\}$ và ký hiệu L là T(A).

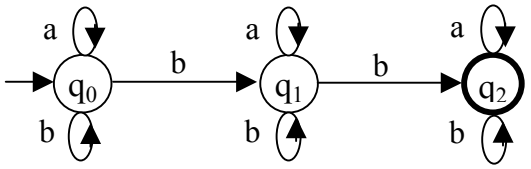
Hai ôtômat hữu hạn không đơn định (hoặc một đơn định một không đơn định) A và A' được gọi là tương đương nếu T(A)=T(A').

Thí dụ 4: Cho ôtômat hữu hạn không đơn định:

$$A = <\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\}>,$$

trong đó $\delta(q_0, a)=\{q_0\}$, $\delta(q_0, b)=\{q_0, q_1\}$, $\delta(q_1, a)=\{q_1\}$, $\delta(q_1, b)=\{q_1, q_2\}$,
$\delta(q_2, a)=\{q_2\}$, $\delta(q_2, b)=\{q_2\}$.

Đồ thị chuyển của A là:



$T(A)=\{\omega_1 b\omega_2 b\omega_3 \mid \omega_1, \omega_2, \omega_3\in\{a, b\}^*\}$.

2.2. QUAN HỆ GIỮA ÔTÔMAT HỮU HẠN VÀ NGÔN NGỮ CHÍNH QUY.

2.2.1. Định lý: Nếu ngôn ngữ L được đoán nhận bởi một ôtômat hữu hạn không đơn định thì tồn tại một ôtômat hữu hạn đơn định đoán nhận L.

Chứng minh: Giả sử L=T(A), với $A = <Q, \Sigma, \delta, q_0, F>$ là một ôtômat hữu hạn không đơn định. Xét ôtômat hữu hạn đơn định $A' = <Q', \Sigma, \delta', t_0, F'>$, trong đó

– Q' là tập trạng thái mới mà $|Q'|=|\mathcal{P}(Q)|$ và ta có thể đồng nhất mỗi phần tử của $\mathcal{P}(Q)$ với mỗi phần tử của Q' như sau: mỗi tập con $\{p_1, p_2, \ldots, p_n\}\in\mathcal{P}(Q)$ được đặt tương ứng với phần tử của Q' ký hiệu $t[p_1, p_2, \ldots, p_n]$;

– $\forall a\in\Sigma$, $\forall t[p_1, p_2, \ldots, p_n]\in Q'$, $\delta'(t[p_1, p_2, \ldots, p_n], a)=t[r_1, r_2, \ldots, r_k]$ nếu $\{r_1, r_2, \ldots, r_k\}=\delta(p_1, a)\cup\delta(p_2, a)\cup\ldots\cup\delta(p_n, a)$;

– $t_0=t[q_0]$;

– $F'=\{t[p_1, p_2, \ldots, p_n]\in Q' \mid \{p_1, p_2, \ldots, p_n\}\cap F\neq\varnothing\}$.

Ta chứng minh T(A')=L. Để có điều này, ta chứng minh mệnh đề sau:

$$\forall\omega\in\Sigma^*, \delta'(t[q_0], \omega)=t[p_1, p_2, \ldots, p_n]\Leftrightarrow\delta(q_0, \omega)=\{p_1, p_2, \ldots, p_n\}$$

bằng quy nạp theo độ dài của ω.

Nếu $d(\omega)=0$ thì $\omega=\varepsilon$, khi đó $\delta(q_0, \varepsilon)=\{q_0\}$ và theo định nghĩa $\delta'(t[q_0], \varepsilon)=t[q_0]$.

Giả sử mệnh đề đúng với mọi từ ω_1 có $d(\omega_1)\le m$ ($m\ge 1$). Cho $\omega\in\Sigma^*$ có $d(\omega)=m+1$. Đặt $\omega=\omega_1 a$, với $a\in\Sigma$, $\omega_1\in\Sigma^*$, $d(\omega_1)=m$. Giả sử $\delta'(t[q_0], \omega_1)=t[p_1,\ldots, p_n]$. Khi đó theo định nghĩa của δ', ta có

$$\delta'(t[q_0], \omega_1 a)=\delta'(\delta'(t[q_0], \omega_1), a)=\delta'(t[p_1, \ldots, p_n], a).$$

Theo giả thiết quy nạp,

$$\delta'(t[q_0], \omega_1)=t[p_1, \ldots, p_n]]\Leftrightarrow\delta(q_0, \omega_1)=\{p_1, \ldots, p_n\}.$$

Mặt khác,

$$\delta'(t[p_1, \ldots, p_n], a)=t[r_1, r_2, \ldots, r_k]\Leftrightarrow\{r_1, r_2, \ldots, r_k\}=\bigcup_{i=1}^{n}\delta(p_i, a)=\delta(q_0, \omega_1 a).$$

Vậy mệnh đề đúng đến $m+1$. Cuối cùng ta có:

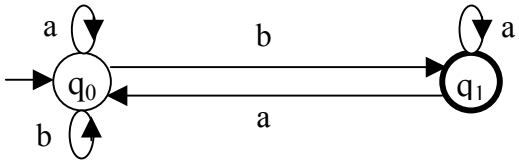
$$\omega\in T(A')\Leftrightarrow\delta'(t[q_0], \omega)=t[p_1, \ldots, p_n]\in F'\Leftrightarrow\{p_1, \ldots, p_n\}\cap F\neq\varnothing\Leftrightarrow\delta(q_0, \omega)\cap F\neq\varnothing$$
$$\Leftrightarrow\omega\in T(A).$$

Thí dụ 5: Cho ôtômat hữu hạn không đơn định:

$$A = <\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\}>,$$

trong đó $\delta(q_0, a)=\{q_0\}$, $\delta(q_0, b)=\{q_0, q_1\}$, $\delta(q_1, a)=\{q_0, q_1\}$, $\delta(q_1, b)=\varnothing$.

Đồ thị chuyển của A là:



Ta xây dựng ôtômat $A'=<Q', \{a, b\}, \delta', t_0, F'\}$ tương đương với A, trong đó:

– $Q'=\{t_\varnothing, t[q_0], t[q_1], t[q_0, q_1]\}$

– $t_0=t[q_0]$,

– $F'=\{t[q_1], t[q_0, q_1]\}$,

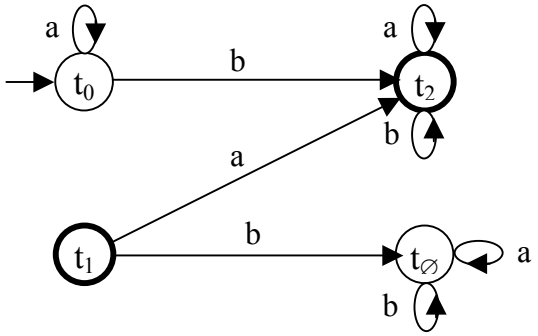
– δ' được xác định như sau: $\delta'(t[q_0], a)=t[q_0]$, $\delta'(t[q_0], b)=t[q_0, q_1]$,

$\delta'(t[q_1], a)=t[q_0, q_1]$, $\delta'(t[q_1], b)=t_\varnothing$,

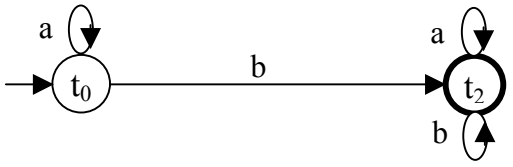
$\delta'(t_\varnothing, a)=t_\varnothing$, $\delta'(t_\varnothing, b)=t_\varnothing$,

$\delta'(t[q_0, q_1], a)=t[q_0, q_1]$, $\delta'(t[q_0, q_1], b)=t[q_0, q_1]$.

Đặt $t_1=t[q_1]$, $t_2=t[q_0, q_1]$, $t_3=t_\varnothing$, ta có đồ thị chuyển của A' là:



Rõ ràng A’ tương đương với ôtômat A’’ có đồ thị chuyển sau:



và T(A)=T(A’)=T(A’’)={a^nbω | n≥0, ω∈{a, b}*}.

2.2.2. Định lý: Nếu L là một ngôn ngữ chính quy thì tồn tại một ôtômat hữu hạn không đơn định đoán nhận L.

Chứng minh: Giả sử L=L(G), với G=<Σ, Δ, S, P> là văn phạm chính quy. Xét ôtômat hữu hạn không đơn định A=<Q, Σ, δ, q_0, F>, trong đó

– Q=Δ∪{E}, E∉Σ∪Δ;

– q_0=S;

– F={E} nếu S→ε∉P và F={E, S} nếu S→ε∈P;

– δ(A, a)={B | A→aB∈P}∪{E | A→a∈P} và δ(E, a)=∅, ∀A∈Δ, ∀a∈Σ.

Ta chứng minh L=T(A).

1) ω∈L: a) ω=ε: S→ε∈P, do đó S∈F. Trong trường hợp này δ(S, ε)={S} nên ε∈T(A).

b) ω=a_1a_2 …a_n ≠ε: Ta có suy dẫn S⊢a_1A_1⊢$a_1a_2A_2$⊢…⊢$a_1a_2…a_{n-1}A_{n-1}$⊢$a_1…a_{n-1}a_n$ Do đó tồn tại dãy quy tắc S→a_1A_1, A_1→a_2A_2, …, A_{n-1}→a_n trong P. Từ định nghĩa của δ, ta có A_1∈δ(S, a_1), A_2∈δ(A_1, a_2), …, A_{n-1}∈δ(A_{n-2}, a_{n-1}), E∈δ(A_{n-1}, a_n). Như vậy, E∈δ(S, a_1a_2… a_n) hay ω∈T(A).

2) ω∈T(A): a) ω=ε: δ(S, ε)∩F≠∅ hay S∈F hay S→ε∈P, do đó ε∈L.

b) ω=a_1a_2 …a_n ≠ε: δ(S, ω)∩F≠∅ với ω≠ε hay E∈δ(S, ω), do đó tồn tại các trạng thái A_1, A_2, …, A_{n-1}∈Δ sao cho A_1∈δ(S, a_1), A_2∈δ(A_1, a_2), …, A_{n-1}∈δ(A_{n-2}, a_{n-1}), E∈δ(A_{n-1}, a_n). Từ đó ta có S→a_1A_1, A_1→a_2A_2, …, A_{n-1}→a_n∈P hay trong G có một suy dẫn là S⊢ a_1A_1⊢ $a_1a_2A_2$⊢…⊢$a_1a_2…a_{n-1}A_{n-1}$⊢$a_1…a_{n-1}a_n$=ω. Vì vậy ω∈L.

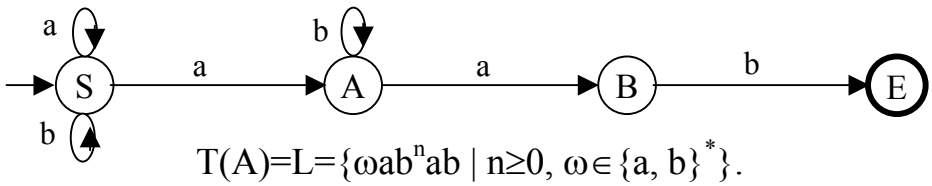
Thí dụ 6: Cho ngôn ngữ L={ωab^nab | n≥0, ω∈{a, b}*}. Ta có L=L(G) trong đó

G=<{a, b}, {S, A, B}, S, {S→aS, S→bS, S→aA, A→bA, A→aB, B→b}>

là văn phạm chính quy.

Xét ôtômat hữu hạn không đơn định A=<{S, A, B, E}, {a, b}, δ, S, {E}>, trong đó δ(S, a)={S, A}, δ(S, b)={S}, δ(A, a)={B}, δ(A, b)={A}, δ(B, a)=∅, δ(B, b)={E}, δ(E, a)=∅, δ(E, b)=∅.

Đồ thị chuyển của A là:



T(A)=L={ωab^nab | n≥0, ω∈{a, b}*}.

2.2.3. Định lý: Nếu L là ngôn ngữ được đoán nhận bởi một ôtômat hữu hạn đơn định thì L là một ngôn ngữ chính quy.

Chứng minh: Giả sử L=T(A), với A = <Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định. Xét văn phạm

$$G=<\Sigma, Q, q_0, P>,$$

trong đó $P=\{q\to ap \mid \delta(q, a)=p\}\cup\{q\to a \mid \delta(q, a)=p\in F\}$. Khi đó G là một văn phạm chính quy. Ta chứng minh $L(G)=L \setminus \{\varepsilon\}$.

1) $\omega=a_1a_2 \ldots a_n\in L(G)$: $\omega\neq\varepsilon$ và tồn tại suy dẫn:

$$q_0\vdash a_1p_1\vdash a_1a_2p_2\vdash \ldots\vdash a_1a_2\ldots a_{n-1}p_{n-1}\vdash a_1\ldots a_{n-1}a_n=\omega.$$

Do đó $q_0\to a_1p_1$, $p_1\to a_2p_2$, …, $p_{n-1}\to a_{n-1}p_{n-1}$, $p_{n-1}\to a_n\in P$ hay ta có

$$p_1=\delta(q_0, a_1),\ p_2=\delta(p_1, a_2),\ \ldots,\ p_{n-1}=\delta(p_{n-2}, a_{n-1}),\ p_n\in F$$

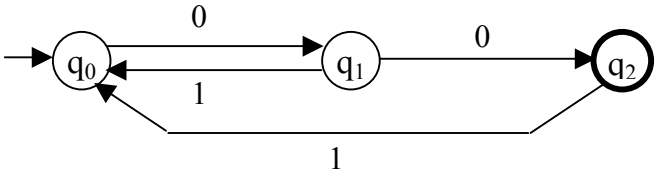
tức là $\delta(q_0, \omega)=p_n\in F$ hay $\omega\in T(A) \setminus \{\varepsilon\}=L \setminus \{\varepsilon\}$.

2) $\omega=a_1a_2 \ldots a_n\in L \setminus \{\varepsilon\}$: Tồn tại dãy trạng thái $p_1, p_2, \ldots, p_n$ sao cho $\delta(q_0, a_1)=p_1$, $\delta(p_1, a_2)=p_2$, …, $\delta(p_{n-2}, a_{n-1})=p_{n-1}$, $p_n\in F$. Do đó $q_0\to a_1p_1$, $p_1\to a_2p_2$, …, $p_{n-1}\to a_{n-1}p_{n-1}$, $p_{n-1}\to a_n\in P$ hay $q_0\vdash a_1p_1\vdash a_1a_2p_2\vdash \ldots\vdash a_1a_2\ldots a_{n-1}p_{n-1}\vdash a_1\ldots a_{n-1}a_n=\omega$ hay $\omega\in L(G)$.

Trong trường hợp $\varepsilon\in L$, ta xây dựng G' tương đương với G trong đó ký hiệu đầu không xuất hiện trong bất kỳ vế phải của quy tắc nào, đồng thời thêm vào G' quy tắc $q'_0\to\varepsilon$ (q'_0 là ký hiệu đầu của G') để nhận được văn phạm chính quy G'' sao cho $L(G'')=L(G')\cup\{\varepsilon\}=L(G)\cup\{\varepsilon\}$.

Thí dụ 7: Cho ôtômat hữu hạn đơn định $A=<\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\}>$, trong đó $\delta(q_0, 0)=q_1$, $\delta(q_1, 0)=q_2$, $\delta(q_1, 1)=q_0$, $\delta(q_2, 1)=q_0$.

Đồ thị chuyển của A là:



$T(A)=\{\omega00 \mid \omega\in\{01, 001\}^*\}$ là ngôn ngữ chính quy.

2.2.4. Chú ý: Từ các định lý trên với chú ý là mỗi ôtômat hữu hạn đơn định có thể xem như là một ôtômat hữu hạn không đơn định, ta có thể rút ra kết luận sau.

Gọi $\mathcal{D}$ là lớp các ngôn ngữ được đoán nhận bởi ôtômat hữu hạn đơn định, $\mathcal{N}$ là lớp các ngôn ngữ được đoán nhận bởi ôtômat hữu hạn không đơn định và $\mathcal{R}$ là lớp các ngôn ngữ chính quy.

Định lý 2.2.1 cho biết $\mathcal{N}\subset\mathcal{D}$.

Định lý 2.2.2 cho biết $\mathcal{R}\subset\mathcal{N}$.

Định lý 2.2.3 cho biết $\mathcal{D}\subset\mathcal{R}$.

Vậy $\mathcal{D}=\mathcal{N}=\mathcal{R}$.

2.3. BIỂU THỨC CHÍNH QUY.

2.3.1. Định nghĩa: Trên bảng chữ Σ, ta định nghĩa biểu thức chính quy theo các bước đệ quy sau đây:

1) ∅ là biểu thức chính quy, nó biểu diễn ngôn ngữ rỗng.

2) **ε** là biểu thức chính quy, nó biểu diễn ngôn ngữ {ε}.

3) Nếu a∈Σ thì a là biểu thức chính quy, nó biểu diễn ngôn ngữ {a}.

4) Nếu r, s tương ứng là biểu thức chính quy trên Σ biểu diễn ngôn ngữ R, S thì (**r+s**) là biểu thức chính quy biểu diễn ngôn ngữ R∪S, (**rs**) là biểu thức chính quy biểu diễn ngôn ngữ R.S và (**r*****)** là biểu thức chính quy biểu diễn ngôn ngữ R*.

Trong biểu thức chính quy, ta có thể bỏ các dấu ngoặc và quy ước thứ tự thực hiện các phép tính là phép lặp, phép ghép, phép hợp. Chẳng hạn, biểu thức chính quy **ab*****a+ba** thay cho biểu thức (((**a**(**b***))**a**)+(**ba**). Ngoài ra, nếu không sợ nhầm lẫn, ta có thể sử dụng ký hiệu a thay cho biểu thức chính quy **a** với mỗi a∈Σ.

2.3.2. Định nghĩa: Hai biểu thức chính quy **r** và **s** được gọi là tương đương, ký hiệu **r=s**, nếu chúng biểu diễn cùng một ngôn ngữ.

Với **r**, **s**, **t** là các biểu thức chính quy trên bảng chữ Σ, dễ dàng có được các tương sau:

1) **r+s** = **s+r**,

2) (**r+s**)**+t** = **r+**(**s+t**),

3) **r+r** = **r**,

4) (**rs**)**t** = **r**(**st**),

5) **r**(**s+t**) = **rs+rt**, (**s+t**)**r** = **sr+tr**,

6) ∅* = **ε**,

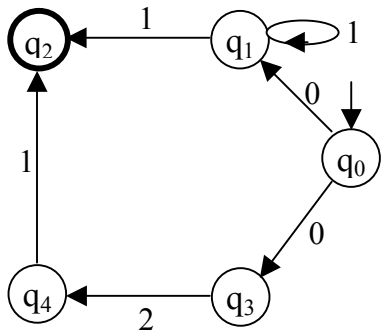
7) (**r***)* = **r***,

8) **rr*****+ε=r***,

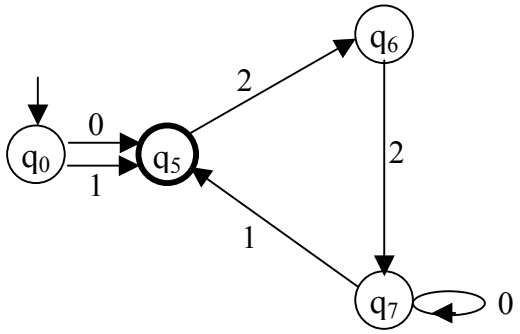
9) (**r*****s***)* = (**r+s**)*,

Thí dụ 8: Cho biểu thức chính quy (**01*****+02**)**1+**(**0+1**)(**220*****1**)*.

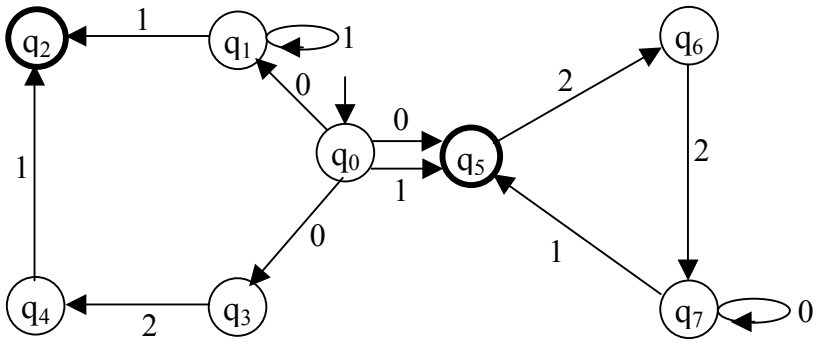
a) (**01*****+02**)**1** = **01*1+021** là biểu thức chính quy biểu diễn ngôn ngữ được đoán nhận bởi ôtômat hữu hạn có đồ thị chuyển là:



b) (**0+1**)(**220*****1**)* là biểu thức chính quy biểu diễn ngôn ngữ được đoán nhận bởi ôtômat hữu hạn có đồ thị chuyển là:



Vì vậy, biểu thức chính quy đã cho biểu diễn ngôn ngữ được đoán nhận bởi ôtômat hữu hạn có đồ thị chuyển là:



2.3.3. Định lý: Cho L là một ngôn ngữ trên bảng chữ Σ. Khi đó L là một ngôn ngữ chính quy khi và chỉ khi tồn tại một biểu thức chính quy trên Σ biểu diễn L.

Chứng minh: Giả sử tồn tại một biểu thức chính quy trên bảng chữ $\Sigma=\{a_1, \ldots, a_n\}$ biểu diễn ngôn ngữ L. Theo định nghĩa của biểu thức chính quy thì L là tập hợp được tạo thành từ các tập cơ sở $\varnothing$, $\{\varepsilon\}$, $\{a_1\}$, …, $\{a_n\}$ bằng việc áp dụng một số hữu hạn các phép hợp, ghép, lặp. Vì các tập cơ sở trên là ngôn ngữ chính quy và hợp, ghép, lặp của một số hữu hạn của chúng cũng là ngôn ngữ chính quy. Do đó L là một ngôn ngữ chính quy.

Giả sử L là một ngôn ngữ chính quy trên bảng chữ Σ. Khi đó theo Mục 2.2, L=T(A), với A=<Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định.

Giả sử $Q=\{q_0, q_1, \ldots, q_m\}$. Ký hiệu R_{ij}^k là tập hợp tất cả các từ mà dưới tác động của chúng, ôtômat A chuyển từ trạng thái q_i đến q_j, thêm vào đó các trạng thái mà A đi qua có chỉ số không vượt quá k. Trên đồ thị chuyển của A, R_{ij}^k là tập hợp các từ ứng với các đường đi từ q_i đến q_j không đi qua $q_{k+1}, \ldots, q_m$.

Một cách hình thức, ta có thể định nghĩa R_{ij}^k bằng đệ quy như sau:

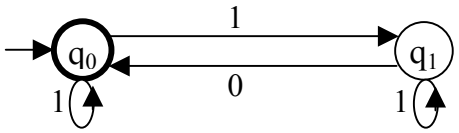
$R_{ij}^{-1}=\{a\in\Sigma \mid \delta(q_i, a)=q_j\}$, $i\neq j$; $R_{ii}^{-1}=\{a\in\Sigma \mid \delta(q_i, a)=q_i\}\cup\{\varepsilon\}$;

$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$, ∀i, j, k=0, 1, …, m.

Bằng quy nạp theo k, tồn tại một biểu thức chính quy biểu diễn ngôn ngữ R_{ij}^k, ∀i, j=0, 1, …, m.

Giả sử F={$q_{i_1}, q_{i_2}, ..., q_{i_k}$}. Khi đó L=T(A)=$R_{0i_1}^m \cup R_{0i_2}^m \cup ... \cup R_{0i_k}^m$ (nếu tồn tại j sao cho i_j=0 thì thành phần thứ j của hợp là $(R_{00}^m)^*$. Do đó tồn tại biểu thức chính quy biểu diễn ngôn ngữ L.

Thí dụ 9: Cho ôtômat hữu hạn không đơn định A có đồ thị chuyển là:



Theo chứng minh của Định lý 2.3.3, ta có: T(A)=$(R_{00}^1)^*$=$(R_{00}^0 \cup R_{01}^0 (R_{11}^0)^* R_{10}^0)^*$.
Từ đồ thị chuyển ta thấy rằng:

– Biểu thức chính quy **1*** biểu diễn R_{00}^0.
– Biểu thức chính quy **1*****1** biểu diễn R_{01}^0.
– Biểu thức chính quy **1*****01** biểu diễn R_{11}^0.
– Biểu thức chính quy **1*****0** biểu diễn R_{10}^0.

Vậy T(A) biểu diễn bởi biểu thức chính quy (**1*****+1*****1(1*****01)1*****0)***=(**1*****+11*****0)***.

2.4. CỰC TIỂU HOÁ ÔTÔMAT HỮU HẠN.

Cùng một ngôn ngữ chính quy L, có thể có nhiều ôtômat hữu hạn đoán nhận nó. Nhưng trước hết người ta phải quan tâm đến các ôtômat có số trạng thái ít nhất cùng đoán nhận ngôn ngữ L. Từ đó ta có khái niệm ôtômat tối tiểu.

2.4.1. Định nghĩa: Ôtômat có số trạng thái ít nhất trong các ôtômat hữu hạn đơn định đầy đủ cùng đoán nhận ngôn ngữ L được gọi là ôtômat tối tiểu của ngôn ngữ L. Việc tìm ôtômat tối tiểu M sao cho T(M)=T(A) với A là ôtômat hữu hạn đơn định đầy đủ cho trước gọi là cực tiểu hoá ôtômat hữu hạn A.

2.4.2. Định nghĩa: Cho A = <Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định đầy đủ. Trên Σ* có quan hệ R_A được định nghĩa như sau:

$$\forall \alpha, \beta \in \Sigma^*, \alpha R_A \beta \Leftrightarrow \delta(q_0, \alpha) = \delta(q_0, \beta).$$

Dễ dàng thấy rằng R_A có các tính chất sau:

– R_A có tính phản xạ vì $\forall \alpha \in \Sigma^*$, $\alpha R_A \alpha$.
– R_A có tính đối xứng vì $\forall \alpha, \beta \in \Sigma^*$, $\alpha R_A \beta$ kéo theo $\beta R_A \alpha$.
– R_A có tính bắc cầu vì $\forall \alpha, \beta, \gamma \in \Sigma^*$, $\alpha R_A \beta$ và $\beta R_A \gamma$ kéo theo $\alpha R_A \gamma$.

Do đó R_A là một quan hệ tương đương, nên quan hệ R_A phân hoạch Σ* thành các lớp tương đương. Từ định nghĩa của R_A, ta thấy rằng mỗi lớp tương đương ứng với một trạng thái. Vì vậy số các lớp tương đương theo R_A không lớn hơn số các trạng thái của A.

2.4.3. Định nghĩa: Quan hệ tương đương R trên Σ^* được gọi là bất biến phải nếu

$$\forall \alpha, \beta, \gamma \in \Sigma^*, \alpha R \beta \Rightarrow \alpha\gamma R \beta\gamma.$$

Quan hệ R_A như trên là bất biến phải. Thật vậy,

$\forall \alpha, \beta, \gamma \in \Sigma^*, \alpha R_A \beta \Rightarrow \delta(q_0, \alpha) = \delta(q_0, \beta) \Rightarrow \delta(q_0, \alpha\gamma) = \delta(\delta(q_0, \alpha), \gamma) = \delta(\delta(q_0, \beta), \gamma) = \delta(q_0, \beta\gamma) \Rightarrow \alpha\gamma R_A \beta\gamma.$

Bây giờ ta xét một quan hệ tương đương bất biến phải khác trên Σ^* mà nó được xác định bởi một ngôn ngữ L cho trước như dưới đây.

2.4.4. Định nghĩa: Cho L là một ngôn ngữ trên bảng chữ Σ. Trên Σ^* có quan hệ R_L được định nghĩa như sau:

$$\forall \alpha, \beta \in \Sigma^*, \alpha R_L \beta \Leftrightarrow (\forall \gamma \in \Sigma^*, \alpha\gamma \in L \Leftrightarrow \beta\gamma \in L).$$

Kiểm tra dễ dàng rằng R_L thoả mãn các tính chất phản xạ, đối xứng và bắc cầu, do đó R_L là một quan hệ tương đương. Hơn thế nữa, R_L cũng bất biến phải.

2.4.5. Định lý: Cho L là một ngôn ngữ trên bảng chữ Σ. Khi đó L là ngôn ngữ chính quy khi và chỉ khi R_L phân hoạch Σ^* thành một số hữu hạn các lớp tương đương.

Chứng minh: Giả sử L là một ngôn ngữ chính quy. Khi đó tồn tại một ôtômat hữu hạn đơn định đầy đủ $A = <Q, \Sigma, \delta, q_0, F>$ sao cho $L=T(A)$.

Xét quan hệ R_A trên Σ^*. Với $\alpha, \beta \in \Sigma^*$ sao cho $\alpha R_A \beta$ thì $\forall \gamma \in \Sigma^*$ ta có

$$\alpha\gamma \in L \Leftrightarrow \delta(q_0, \alpha\gamma) \in F \Leftrightarrow \delta(q_0, \beta\gamma) \in F \Leftrightarrow \beta\gamma \in L.$$

Điều này có nghĩa là $\alpha R_L \beta$. Như vậy, $\alpha R_A \beta$ kéo theo $\alpha R_L \beta$ hay quan hệ R_A mịn hơn quan hệ R_L. Do đó số lớp tương đương theo quan hệ R_L không lớn hơn số lớp tương đương theo quan hệ R_A, mà số này là hữu hạn nên số lớp tương đương theo quan hệ R_L là hữu hạn.

Bây giờ giả sử L là một ngôn ngữ trên bảng chữ Σ mà số lớp tương đương theo quan hệ R_L là hữu hạn. Ta xây dựng một ôtômat hữu hạn đơn định đầy đủ là $A = <Q, \Sigma, \delta, q_0, F>$, trong đó:

– $Q=\{[\alpha] \mid \alpha \in \Sigma^*\}$ ($[\alpha]=\{\beta \in \Sigma^* \mid \alpha R_L \beta\}$ là lớp tương đương theo quan hệ R_L);

– $q_0=[\varepsilon]$;

– $\delta: Q \times \Sigma \longrightarrow Q$ cho bởi $\delta([\alpha], a)=[\alpha a]$ (do R_L là bất biến phải nên nếu $\alpha R_L \beta$ thì $\alpha a R_L \beta a$, do đó định nghĩa của δ là tốt).

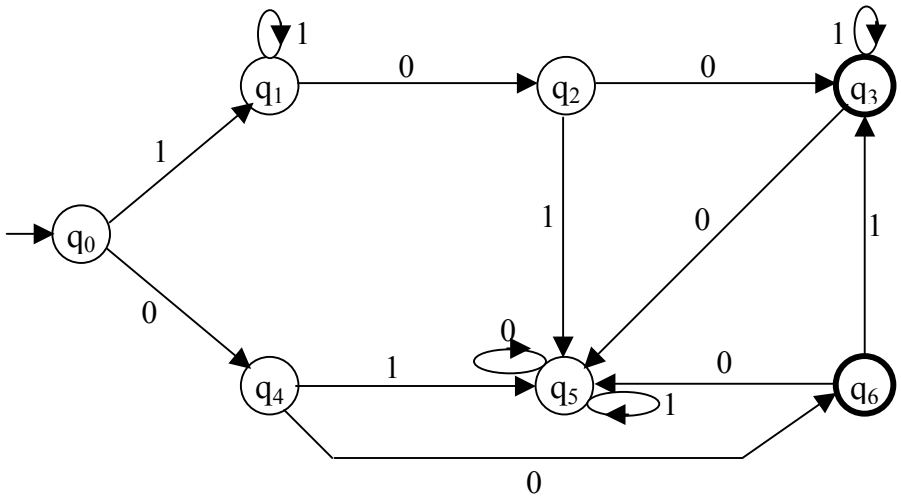
– $F=\{[\alpha] \mid \exists \beta \in [\alpha], \beta \in L\}$.

Khi đó nếu $[\alpha] \in F$ thì $\forall \gamma \in [\alpha]$, ta có $\gamma \in L$. Thật vậy, với $\beta \in [\alpha]$, $\beta \in L$, ta có $\beta R_L \gamma$, từ đó do $\beta = \beta\varepsilon \in L$ nên $\gamma = \gamma\varepsilon \in L$.

Ngoài ra, ta có $T(A)=L$. Thật vậy,

$$\omega \in T(A) \Leftrightarrow \delta([\varepsilon], \omega) = [\varepsilon\omega] = [\omega] \in F \Leftrightarrow \omega \in L.$$

Thí dụ 10: Cho A là một ôtômat hữu hạn đơn định đầy đủ có đồ thị chuyển được cho dưới đây:



Ký hiệu C_i là lớp tương đương xác định bởi q_i, nghĩa là $C_i = \{\alpha \in \Sigma^* \mid \delta(q_0, \alpha)=q_i\}$.

$C_0 = \{\varepsilon\}$,

$C_1 = \mathbf{11}^*$,

$C_2 = \mathbf{11}^*\mathbf{0}$,

$C_3 = \mathbf{11}^*\mathbf{001}^*\mathbf{+0011}^*$,

$C_4 = \mathbf{0}$,

$C_5 = (\mathbf{01+000+11}^*\mathbf{01+11}^*\mathbf{001}^*\mathbf{0+0011}^*\mathbf{0})(\mathbf{0+1})^*$,

$C_6 = \mathbf{00}$.

Dễ dàng thấy rằng $T(A) = C_3 \cup C_6 = \mathbf{0011}^*\mathbf{+00+11}^*\mathbf{001}^* = \mathbf{1}^*\mathbf{001}^*$.

Với $L=T(A)=\mathbf{1}^*\mathbf{001}^*$, ta tìm các lớp tương đương theo quan hệ R_L. Với α, $\beta \in \{0, 1\}^*$, $\alpha R_L \beta$ khi và chỉ khi một trong các mệnh đề dưới đây thoả mãn:

a) α và β không chứa số 0.

b) α và β chứa đúng một chữ số 0 ở cuối.

c) α và β chứa đúng hai chữ số 0 liên tiếp nhau.

d) α và β hoặc là chỉ chứa một chữ số 0 không phải ở cuối câu, hoặc là hai chữ số 0 không đứng cạnh nhau, hoặc chứa nhiều hơn 2 chữ số 0. Khi đó các lớp tương đương theo quan hệ R_L là:

$[\varepsilon] = C_0 \cup C_1 = \boldsymbol{\varepsilon + 11}^*$**,**

$[0] = C_2 \cup C_4 = \mathbf{11}^*\mathbf{0+0}$,

$[00] = C_3 \cup C_6 = \mathbf{11}^*\mathbf{001}^*\mathbf{+0011}^*\mathbf{+00}$,

$[000] = C_5 = (\mathbf{01+000+11}^*\mathbf{01+11}^*\mathbf{001}^*\mathbf{0+0011}^*\mathbf{0})(\mathbf{0+1})^*$.

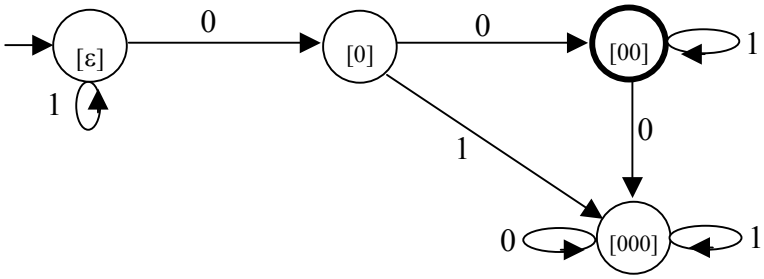
Trên cơ sở chứng minh của Định lý 2.4.5, ta có ôtômat hữu hạn đơn định:

$$M = <\{[\varepsilon], [0], [00], [000]\}, \{0, 1\}, \delta, [\varepsilon], \{[00]\},$$

trong đó, $\delta([\varepsilon], 0)=[0]$, $\delta([\varepsilon], 1)=[\varepsilon]$, $\delta([0], 0)=[00]$, $\delta([0], 1)=[000]$,

$\delta([00], 0)=[000]$, $\delta([00], 1)=[00]$, $\delta([000], 0)=[000]$, $\delta([000], 1)=[000]$.

Đồ thị chuyển của M là:



T(M) = T(A) = $\mathbf{1^*001^*}$.

Lưu ý rằng hai ôtômat được xem là như nhau nếu có một phép đổi tên các trạng thái sao cho hai đồ thị chuyển của chúng là giống nhau.

2.4.6. Định lý: Nếu L là một ngôn ngữ chính quy thì tồn tại duy nhất một ôtômat hữu hạn đơn định tối tiểu đoán nhận L.

Chứng minh: Gọi M = <Q, Σ, δ, q_0, F> là ôtômat hữu hạn đơn định đoán nhận ngôn ngữ L được xây dựng như trong Định lý 2.4.5. Cho A là ôtômat hữu hạn đơn định đầy đủ tuỳ ý đoán nhận L. Khi đó số trạng thái của A không ít hơn số lớp tương đương theo quan hệ R_A và do đó không ít hơn số lớp tương đương theo quan hệ R_L, tức là số trạng thái của M.

Giả sử A' = <Q', Σ, δ', q'$_0$, F'> là ôtômat hữu hạn đơn định đoán nhận L có số trạng thái bằng số trạng thái của M. $\forall q' \in Q'$, do số lớp tương đương theo quan hệ $R_{A'}$ bằng số trạng thái của A', nên $\exists \omega \in \Sigma^*$ sao cho $\delta'(q'_0, \omega)=q'$. Đặt $q=\delta(q_0, \omega)$. Nếu $\exists \alpha \in \Sigma^*$ sao cho $\delta'(q'_0, \omega)=\delta'(q'_0, \alpha)=q'$ thì $\omega R_{A'} \alpha$. Từ sự bằng nhau giữa số lớp tương đương theo $R_{A'}$ và theo R_L, ta suy ra $\omega R_L \alpha$. Do đó ta có:

$$\delta([\varepsilon], \omega) = [\omega] = [\alpha] = \delta([\varepsilon], \alpha) \text{ (ở đây } q_0=[\varepsilon]).$$

Bằng phương pháp như vậy, ta có thể đồng nhất các trạng thái của M với các trạng thái của A'. Chính xác hơn, tồn tại song ánh f từ Q' lên Q thoả mãn:

$$f(\delta'(q'_0, \alpha)) = \delta(q_0, \alpha), \forall \alpha \in \Sigma^*.$$

Điều này cho biết hai ôtômat M và A' được xem là như nhau và M là ôtômat hữu hạn đơn định tối tiểu duy nhất đoán nhận L.

2.4.7. Định nghĩa: Trạng thái q của ôtômat hữu hạn đơn định đầy đủ A=<Q, Σ, δ, q_0, F> được gọi là đến được nếu $\exists \omega \in \Sigma^*$ sao cho $\delta(q_0, \omega)=q$.

Khi đó mỗi lớp tương đương theo quan hệ R_A có thể được đồng nhất với một trạng thái đến được.

2.4.8. Chú ý: Cho A=<Q, Σ, δ, q_0, F> là một ôtômat hữu hạn đơn định đầy đủ đoán nhận ngôn ngữ chính quy L trên bảng chữ Σ. Chúng ta sẽ chỉ ra phương pháp tìm ôtômat hữu hạn đơn định tối tiểu M đoán nhận L. Để tìm các trạng thái của M, ta cần tìm các lớp tương đương theo quan hệ R_L, mà mỗi lớp tương đương theo R_L lại là hợp của một số lớp tương đương theo quan hệ R_A. Do ta đồng nhất mỗi lớp tương đương theo R_A với một trạng thái đến được của A nên ta sẽ xây dựng một

quan hệ tương đương trên tập hợp K tất cả các trạng thái đến được mà quan hệ này có thể đồng nhất với R_L.

Trên tập hợp K xét quan hệ ≡ sau:

$$\forall p,q\in K,\ p\equiv q \Leftrightarrow (\forall \gamma\in\Sigma^*,\ \delta(p,\gamma)\in F \Leftrightarrow \delta(q,\gamma)\in F).$$

Rõ ràng ≡ là một quan hệ tương đương trên K.

$\forall p,q\in K$, $\exists \alpha, \beta\in\Sigma^*$ sao cho $p=\delta(q_0,\alpha)$ và $q=\delta(q_0,\beta)$. Khi đó ta có:

$$\begin{aligned} p\equiv q &\Leftrightarrow (\forall \gamma\in\Sigma^*,\ \delta(\delta(q_0,\alpha),\gamma)\in F \Leftrightarrow \delta(\delta(q_0,\beta),\gamma)\in F)\\ &\Leftrightarrow (\forall \gamma\in\Sigma^*,\ \delta(q_0,\alpha\gamma)\in F \Leftrightarrow \delta(q_0,\beta\gamma)\in F)\\ &\Leftrightarrow (\forall \gamma\in\Sigma^*,\ \alpha\gamma\in L \Leftrightarrow \beta\gamma\in L)\\ &\Leftrightarrow \alpha R_L \beta. \end{aligned}$$

Tập K các trạng thái đến được có thể nhận được như sau. Đặt $K_0=\{q_0\}$, $K_1=K_0\cup\{\delta(q_0, a) \mid a\in\Sigma\}$. Nếu $K_1\neq K_0$, đặt $K_2=K_1\cup\{\delta(p, a) \mid a\in\Sigma,\ p\in K_1 \setminus K_0\}$. Tương tự như vậy, ta có thể tìm được K_{j+1} thông qua K_j ($j\geq 0$). Nếu tồn tại i sao cho $K_{i+1}=K_i$ thì ta có $K=K_i$.

Bây giờ ta tìm các lớp tương đương theo quan hệ ≡ trên K bằng thuật toán như dưới đây.

1. Chọn ra các cặp trạng thái (p, q) mà $p\in F$ và $q\notin F$. Gán số 0 cho các cặp này và đánh dấu mỗi cặp bằng dấu *.

2. Chọn ra các cặp (p, q) mà nó chưa được đánh dấu và xét các cặp $(\delta(p, a), \delta(q, a))$ với mỗi $a\in\Sigma$. Nếu trong số các cặp này tìm thấy một cặp đã được đánh dấu thì ta đánh dấu cặp (p, q). Thêm vào đó, nếu đối với cặp đã chọn $(\delta(p, a), \delta(q, a))$ đã được gán số k thì ta gán số k+1 cho cặp (p, q). Ngược lại, nếu không tìm thấy cặp được đánh dấu * thì ta xếp cặp (p, q) cùng với các cặp $(\delta(p, a), \delta(q, a))$ thành danh sách riêng biệt.

3. Nếu một cặp (p, q) đã được đánh dấu * và được gán số k thì ta đánh dấu * tất cả các cặp mà trước đó ta đã xếp chúng vào danh sách riêng đối với (p, q) và chưa được gán bởi một số nào, ta gán số k−1 cho mỗi cặp đã ký hiệu.

4. Lặp lại Bước **2** cho đến khi không còn các cặp (p, q) không được đánh dấu * mà chưa được đưa ra xét.

Ta sẽ chứng tỏ rằng các trạng thái $p, q\in K$ là không tương đương (theo quan hệ ≡) khi và chỉ khi trong quá trình trên, cặp (p, q) bị đánh dấu *.

Trước hết, giả sử p và q không tương đương và $\gamma\in\Sigma^*$ sao cho $\delta(p, \gamma)\in F$ và $\delta(q, \gamma)\notin F$. Bằng quy nạp theo độ dài của γ, ta chỉ ra rằng cặp (p, q) được đánh dấu *. Nếu $\gamma=\varepsilon$ thì $\delta(p, \varepsilon)=p$, $\delta(q, \varepsilon)=q$, nên cặp (p, q) được đánh dấu *. Giả sử mệnh đề đúng đối với mọi cặp trạng thái mà chúng được chỉ ra là không tương đương với nhau bằng một từ có độ dài nhỏ hơn n. Lấy $\gamma\in\Sigma^*$ mà $d(\gamma)=n$, ta có $\gamma=a\gamma'$, với $a\in\Sigma$ và $\gamma'\in\Sigma^*$ mà $d(\gamma')<n$. Giả sử $\delta(p, a)=r$, $\delta(q, a)=s$. Khi đó r, s được chỉ ra là không

tương đương bằng từ γ', nên theo giả thiết quy nạp (r, s) đã được đánh dấu *, do đó cặp (p, q) được đánh dấu *.

Bây giờ giả sử cặp (p, q) đã được đánh dấu *. Ta cần chứng tỏ các trạng thái p và q là không tương đương. Bằng quy nạp theo số k gán cho cặp (p, q), ta có thể chứng minh điều này.

Khi k=0, ta có $p \in F$ và $q \notin F$, điều này có nghĩa là các trạng thái p và q là không tương đương. Giả sử mệnh đề đúng đối với mọi cặp được gán với số h<k. Cho (p, q) là cặp được gán bởi số k. Trên cơ sở Bước **2** và **3**, tồn tại $a \in \Sigma$ sao cho cặp ($r=\delta(p, a)$, $s=\delta(q, a)$) được gán bởi số k–1. Theo giả thiết quy nạp, r và s không tương đương; nghĩa là, $\exists \alpha \in \Sigma^*$ sao cho $\delta(r, \alpha) \in F$ và $\delta(s, \alpha) \notin F$. Do đó $\delta(p, a\alpha) \in F$ và $\delta(q, a\alpha) \notin F$ hay p và q là không tương đương.

Xét ôtômat hữu hạn đơn định $M=<Q', \Sigma, \delta', [q_0], F'>$, trong đó:

– $Q'=\{[q] \mid q \in K\}$ ([q] là lớp tương đương theo quan hệ ≡),

– $F'=\{[q] \mid q \in F\}$,

– δ': $Q' \times \Sigma \longrightarrow Q'$ cho bởi $\delta'([q], a)=[\delta(q, a)]$.

Bằng quy nạp theo độ dài của từ ω, dễ dàng có được $\delta'([q], \omega)=[\delta(q, \omega)]$. Từ đó suy ra T(M)=T(A)=L.

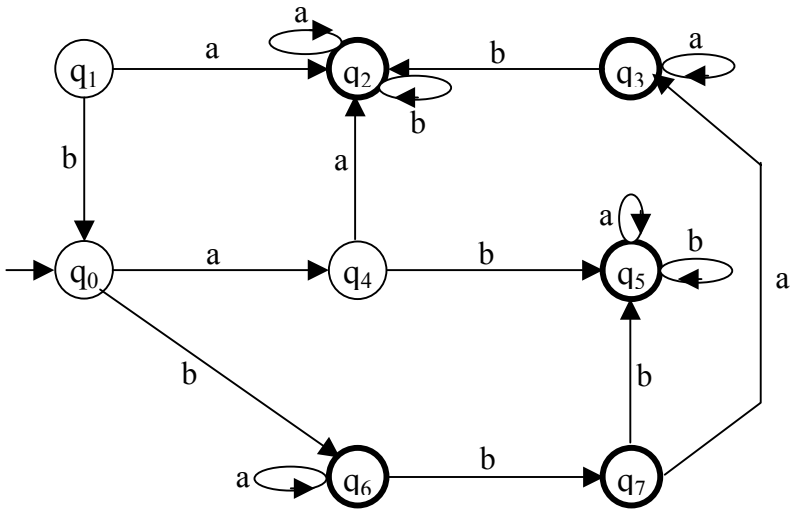
Vậy M là ôtômat hữu hạn đơn định tối tiểu đoán nhận L.

Thí dụ 11: Cho ôtômat hữu hạn đơn định đầy đủ

$$A = <\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a, b\}, \delta, q_0, \{q_2, q_3, q_5, q_6, q_7\}>,$$

trong đó $\delta(q_0, a)=q_4$, $\delta(q_0, b)=q_6$, $\delta(q_1, a)=q_2$, $\delta(q_1, b)=q_0$, $\delta(q_2, a)=q_2$, $\delta(q_2, b)=q_2$, $\delta(q_3, a)=q_3$, $\delta(q_3, b)=q_2$, $\delta(q_4, a)=q_2$, $\delta(q_4, b)=q_5$, $\delta(q_5, a)=q_5$, $\delta(q_5, b)=q_5$, $\delta(q_6, a)=q_6$, $\delta(q_6, b)=q_7$, $\delta(q_7, a)=q_3$, $\delta(q_7, b)=q_5$.

Đồ thị chuyển của A là:



Trước hết, ta tìm các trạng thái đến được của A. Ta có:

$K_0=\{q_0\}, K_1=\{q_0, q_4, q_6\}, K_2=\{q_0, q_4, q_6, q_2, q_5, q_7\}, K_4=K_3=\{q_0, q_4, q_6, q_2, q_5, q_7, q_3\}$.

Trên cơ sở Bước **1**, ta đánh dấu các cặp trạng thái sau:

$(q_0, q_2), (q_0, q_3), (q_0, q_5), (q_0, q_6), (q_0, q_7), (q_4, q_2), (q_4, q_3), (q_4, q_5), (q_4, q_6), (q_4, q_7)$.

Vì $\delta(q_0, a)=q_4$, $\delta(q_4, a)=q_2$ và (q_4, q_2) đã được đánh dấu nên ta đánh dấu (q_0, q_4). Tương tự như vậy đối với các cặp khác. Kết quả có được trong bảng sau:

q_7						
q_6						
q_5	*	*	*			
q_4	*					
q_3	*					
q_2	*					
q_0	*	*	*	*	*	*

Các trạng thái đến được chia thành các lớp tương đương sau:

$[q_0] = \{q_0\}, [q_2] = \{q_2, q_3, q_5, q_6, q_7\}, [q_4] = \{q_4\}$.

Vậy ôtômat hữu hạn đơn định tối tiểu đoán nhận L=T(A) là:

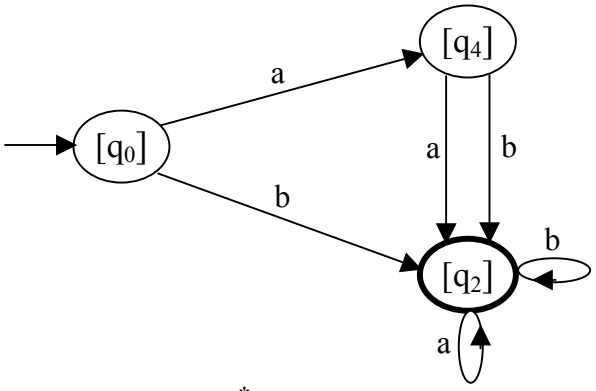
$M = <\{[q_0], [q_2], [q_4]\}, \{a, b\}, \delta', [q_0], \{[q_2]\}>$,

trong đó, $\delta'([q_0], a)=[\delta(q_0, a)]=[q_4]$, $\delta'([q_0], b)=[\delta(q_0, b)]=[q_2]$,

$\delta'([q_2], a)=[\delta(q_2, a)]=[q_2]$, $\delta'([q_2], b)=[\delta(q_2, b)]=[q_2]$,

$\delta'([q_4], a)=[\delta(q_4, a)]=[q_2]$, $\delta'([q_4], b)=[\delta(q_4, b)]=[q_2]$.

Đồ thị chuyển của M là:

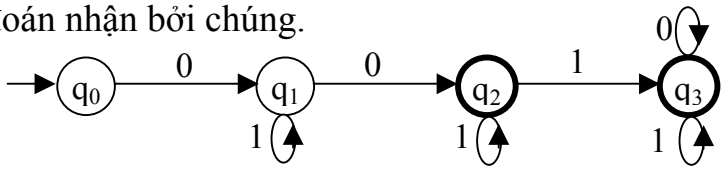


$T(M) = T(A) = (\mathbf{b}+\mathbf{aa}+\mathbf{ab})(\mathbf{a}+\mathbf{b})^*$.

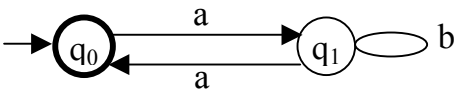
BÀI TẬP CHƯƠNG II:

1. Hãy xây dựng các ôtômat hữu hạn có đồ thị chuyển sau và xác định các ngôn ngữ được đoán nhận bởi chúng.

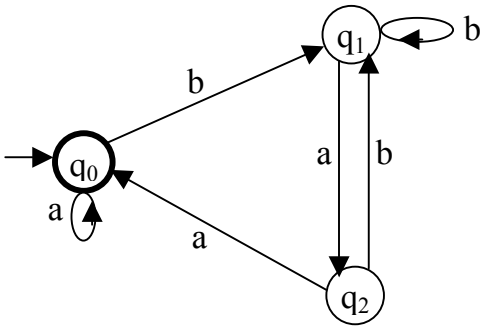
a)



b)



c)



2. Hãy xây dựng các ôtômat hữu hạn đơn định đoán nhận các ngôn ngữ sau:

a) $L = \{a^n b^m \mid n \geq 1, m \geq 1\}$.

b) $L = \{\omega \in \{0, 1\}^* \mid \omega$ bắt đầu đúng 3 chữ số $0\}$.

c) $L = \{\omega \in \{0, 1\}^* \mid \omega$ không bắt đầu bởi hai chữ số 1 liên tiếp$\}$.

d) $L = \{(01)^n, (101)^n \mid n \geq 0\}$.

e) $L = \{(aab)^n (baa)^m \mid n \geq 1, m \geq 1\}$.

3. Chứng minh rằng không tồn tại ôtômat hữu hạn đơn định nào đoán nhận các ngôn ngữ sau:

a) $L = \{\omega\omega \mid \omega \in \{a, b\}^*\}$.

b) $L = \{a^p \mid p$ là một số nguyên tố$\}$.

4. Hãy xây dựng các ôtômat hữu hạn không đơn định đoán nhận các ngôn ngữ sau:

a) $L = \{\omega_1 aba \omega_2 \mid \omega_1, \omega_2 \in \{a, b\}^*\}$.

b) $L = \{\omega \in \{0, 1\}^* \mid \omega$ bắt đầu bằng luỹ thừa dương của $101\}$.

c) $L = \{(1111)^n \omega \mid \omega \in \{0, 1\}^*, n \geq 0\}$.

5. Hãy thành lập các văn phạm chính quy sinh ra các ngôn ngữ mà được đoán nhận bởi các ôtômat hữu hạn không đơn định sau:

a) $A = <\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_2, q_4\}>$, trong đó
$\delta(q_0, a)=\{q_0, q_1\}$, $\delta(q_0, b)=\{q_0, q_1\}$, $\delta(q_1, a)=\varnothing$, $\delta(q_1, b)=\{q_2\}$, $\delta(q_2, a)=\{q_2\}$, $\delta(q_2, b)=\{q_2\}$, $\delta(q_3, a)=\{q_4\}$, $\delta(q_3, b)=\varnothing$, $\delta(q_4, a)=\{q_4\}$, $\delta(q_4, b)=\{q_4\}$.

b) $A = <\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\}>$, trong đó

$\delta(q_0, 0)=\{q_0, q_1\}$, $\delta(q_0, 1)=\{q_1\}$, $\delta(q_1, 0)=\varnothing$, $\delta(q_1, 1)=\{q_0, q_1\}$.

6. Hãy xây dựng các ôtômat hữu hạn đơn định đoán nhận các ngôn ngữ mà được sinh bởi các văn phạm chính quy sau:

a) G=<{a, b, c}, {S, A, B, C, D}, S, {S→aB, S→aC, B→aA, C→bD, D→bC, D→bA, A→cA, A→c}>.

b) G=<{0, 1}, {S, A, B, C, D, E}, S,{S→ε, S→1A, B→0C, B→0D, A→1B, B→1D, D→0E, C→1B, C→0A, E→1A, D→1E, E→0B, A→0D, E→1, C→1}>.

7. Cho k là một số nguyên dương và $L_k = \{\omega \in \{1\}^* \mid d(\omega)=nk, n=1, 2, \ldots\}$. Chứng minh rằng L_k có thể được biểu diễn bởi một biểu thức chính quy. Hãy cho các ôtômat hữu hạn đơn định đoán nhận L_2, L_3, L_5.

8. Cho $L = \{0^n1^n \mid n\geq 0\}$. Hãy xác định các lớp tương đương theo quan hệ R_L và từ đó suy ra L không là ngôn ngữ chính quy.

9. Hãy thay các biểu thức chính quy sau bằng biểu thức tương đương, đơn giản hơn, trong đó không chứa phép cộng:

a) E = $\mathbf{00+0011^*+1^*1001^*}$.

b) E = $\mathbf{100^*100^*+1100^*+100^*1+11}$.

10. Hãy xây dựng các ôtômat hữu hạn đơn định đoán nhận các ngôn ngữ được biểu diễn bởi các biểu thức chính quy sau:

a) $\mathbf{bba(a+b)^*}$.

b) $\mathbf{(a+b)^*bab}$.

c) $\mathbf{(bb+a)^*(aa+b)^*}$.

d) $\mathbf{(\varepsilon+1+11)(01)^*}$.

e) $\mathbf{(0+1)(0+1)(0+1)^*}$.

11. Hãy cực tiểu hoá ôtômat hữu hạn đơn định sau:

$A = <\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}, \{0, 1\}, \delta, q_0, \{q_7, q_9\}>$,

trong đó: $\delta(q_0, 0)=q_1$, $\delta(q_0, 1)=q_3$, $\delta(q_1, 0)=q_7$, $\delta(q_1, 1)=q_1$, $\delta(q_2, 0)=q_0$, $\delta(q_2, 1)=q_5$, $\delta(q_3, 0)=q_5$, $\delta(q_3, 1)=q_9$, $\delta(q_4, 0)=q_1$, $\delta(q_4, 1)=q_7$, $\delta(q_5, 0)=q_8$, $\delta(q_5, 1)=q_4$, $\delta(q_6, 0)=q_3$, $\delta(q_6, 1)=q_8$, $\delta(q_7, 0)=q_8$, $\delta(q_7, 1)=q_7$, $\delta(q_8, 0)=q_7$, $\delta(q_8, 1)=q_8$, $\delta(q_9, 0)=q_8$, $\delta(q_9, 1)=q_9$.

CHƯƠNG III:
ÔTÔMAT ĐẨY XUỐNG VÀ NGÔN NGỮ PHI NGỮ CẢNH

Đối với các lớp văn phạm được phân loại theo Chomsky, lớp văn phạm phi ngữ cảnh có vai trò quan trọng nhất trong việc ứng dụng để xây dựng các ngôn ngữ lập trình và chương trình dịch.

Trong quá trình dịch từ chương trình nguồn ra chương trình đích, người ta sử dụng cấu trúc cú pháp của văn phạm phi ngữ cảnh để phân tích các xâu vào. Cấu trúc cú pháp của một xâu vào được xác định từ dãy các quy tắc suy từ xâu đó. Dựa vào dãy các quy tắc đó, bộ phân tích cú pháp của chương trình dịch sẽ cho biết xâu vào đang xét có thuộc vào xâu do văn phạm phi ngữ cảnh sinh ra hay không. Nói cách khác là với xâu vào ω và một văn phạm phi ngữ cảnh G, hỏi xem $\omega \in L(G)$ hay không? Nếu có thì hãy tìm cách biểu diễn ω bằng văn phạm, tức là tìm các quy tắc sinh của văn phạm G để sinh ra xâu ω.

3.1. VĂN PHẠM PHI NGỮ CẢNH VÀ CÂY SUY DẪN CỦA NÓ.

3.1.1. Định nghĩa: Cho văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$. Cây suy dẫn trong văn phạm G là một cây có gốc thoả mãn bốn yêu cầu sau:

1. Ở mỗi đỉnh được gán một nhãn. Nhãn gán ở đỉnh là các ký hiệu trong tập $\Sigma \cup \Delta$. Gốc của cây được gán nhãn là S.

2. Mỗi đỉnh trong được gán nhãn là một ký hiệu nào đó trong Δ.

3. Mỗi đỉnh ngoài (lá của cây) được gán nhãn là một ký hiệu trong tập Σ.

4. Nếu đỉnh m được gán nhãn là $A \in \Delta$, còn các đỉnh $n_1, n_2, \ldots, n_k$ là các con của đỉnh m theo thứ tự từ trái sang phải và được gán nhãn $B_1, B_2, \ldots, B_k$ tương ứng thì $A \rightarrow B_1B_2\ldots B_k$ là một quy tắc trong P của văn phạm G.

Nếu đọc tất cả nhãn ở các lá theo thứ tự từ trái sang phải, ta sẽ nhận được một từ nào đó. Từ đó sẽ là một phần tử trong L(G) và được gọi là kết quả của cây suy dẫn trong G.

Thí dụ 1: Cho các văn phạm phi ngữ cảnh:

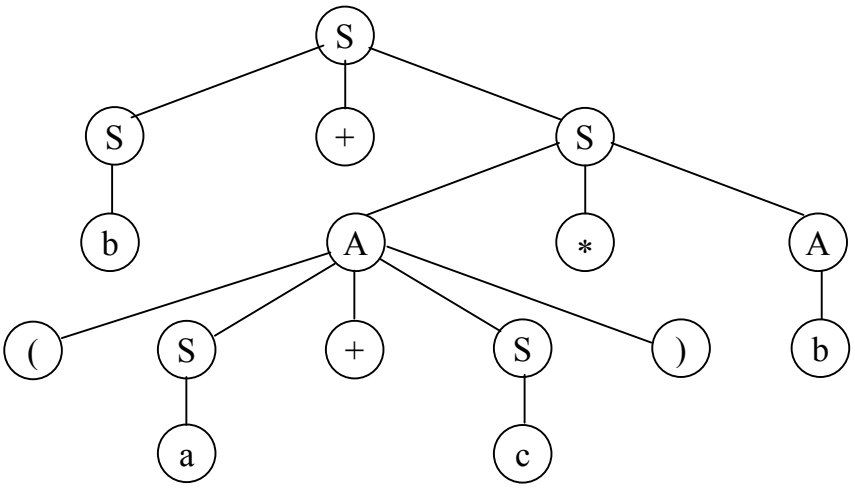
$G_1=<\{a, b, c, +, *, (,)\}, \{S, A\}, S, \{S \rightarrow S+S \mid A*A \mid a \mid b \mid c, A \rightarrow (S+S) \mid a \mid b \mid c\}>$,

$G_2=<\{a, b\}, \{S, A\}, S, \{S \rightarrow Sa \mid Aa, A \rightarrow aAb \mid ab\}>$,

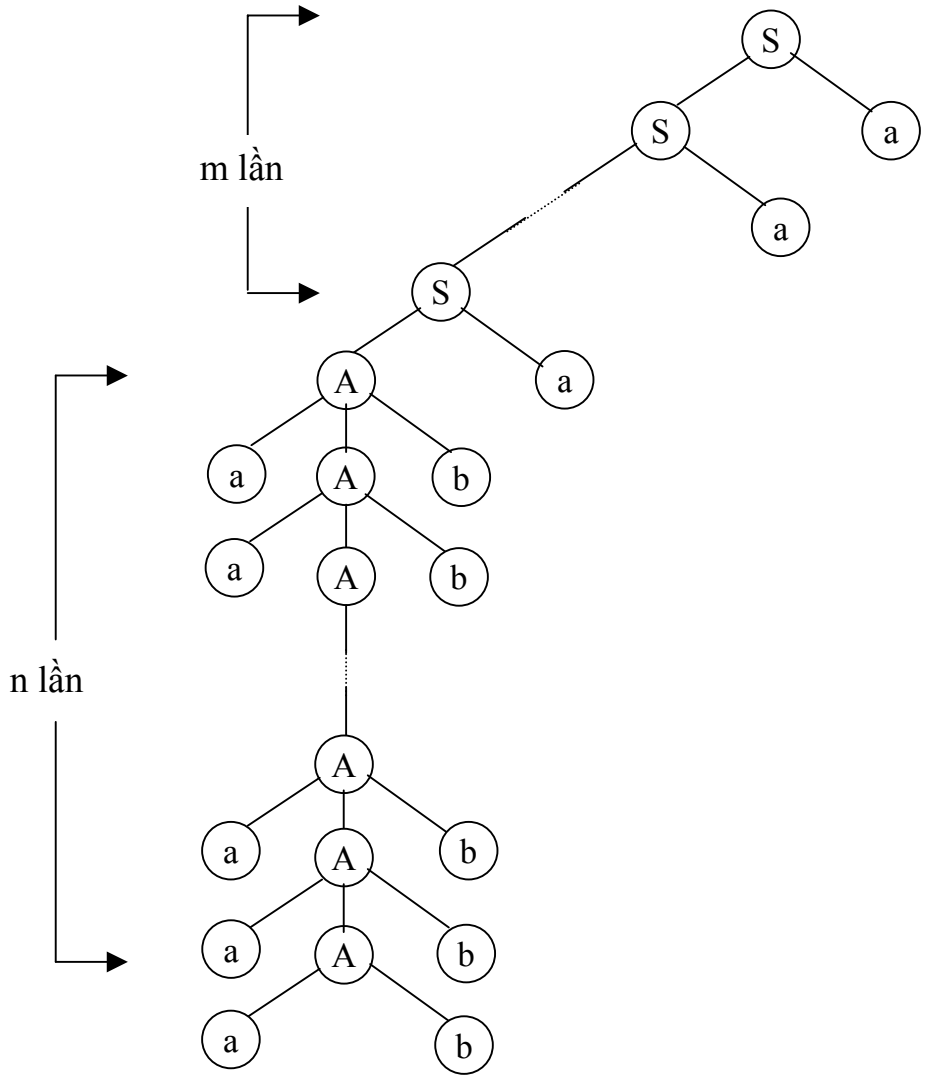
$G_3=<\{\Sigma=\{a_1, a_2, \ldots, a_n\}, \{S\}, S, \{S \rightarrow aSa, S \rightarrow aa \mid a \in \Sigma\}>$,

$G_4=<\{$if, then, else, for, do, a, b, c$\}, \{S, A\}, S, \{S \rightarrow$ if b then A | if b then A else S,
$S \rightarrow a, A \rightarrow$ for c do S | a$\}>$.

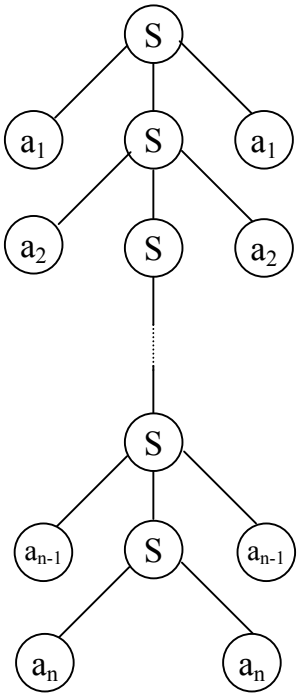
Cây suy dẫn của từ $b+(a+c)*b$ trong G_1 là:



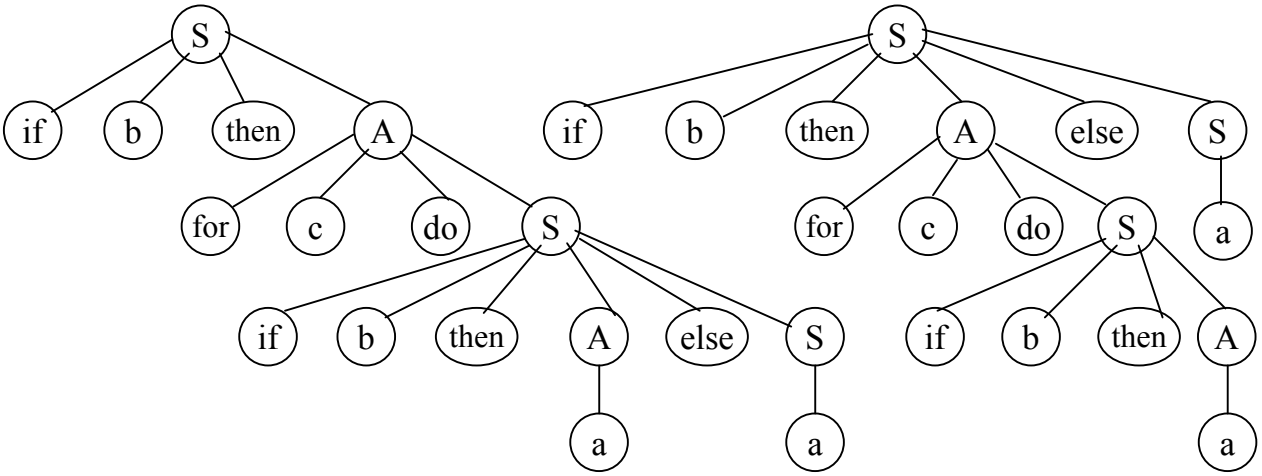
Cây suy dẫn của từ $a^n b^n a^m$ trong G_2 là:



Cây suy dẫn của từ $\omega\omega^R = a_1 a_2 \ldots a_n a_n \ldots a_2 a_1$ trong G_3 là:



Hai cây suy dẫn của từ ω=**if b then for c do if b then a else a** trong G_3 là:



3.1.2. Định lý: Cho G=<Σ, Δ, S, P> là văn phạm phi ngữ cảnh và $\omega \in \Sigma^* \setminus \{\varepsilon\}$. Khi đó $\omega \in L(G)$ khi và chỉ khi tồn tại một cây suy dẫn trong G có kết quả là ω.

Chứng minh: Do $\omega \neq \varepsilon$ nên ta có thể giả thiết rằng $S \rightarrow \varepsilon \notin P$. Bây giờ với mọi $A \in \Delta$, đặt G_A=<Σ, Δ, A, P>, ta có G_A là văn phạm phi ngữ cảnh. Ta sẽ chứng tỏ rằng $\omega \in L(G_A)$ khi và chỉ khi tồn tại một cây suy dẫn trong G_A có kết quả là ω.

Giả sử ω là kết quả của một cây suy dẫn trong G_A và n là số ký hiệu không kết thúc trong cây. Bằng quy nạp theo n, ta sẽ chỉ ra rằng $\omega \in L(G_A)$.

Nếu tổng số ký hiệu không kết thúc trong cây là 1, ký hiệu này phải là A và là gốc của cây, do đó các con của A phải là các đỉnh được gán bởi các ký hiệu kết

thúc, chẳng hạn b_1, b_2, …, b_k. Theo định nghĩa của cây suy dẫn, ta có $A \rightarrow b_1 b_2 \ldots b_k$ hay $A \models \omega$.

Giả sử mệnh đề đúng với mọi cây suy dẫn có số ký hiệu không kết thúc là n−1. Xét một cây suy dẫn trong G_A có kết quả là ω và trong cây có n ký hiệu không kết thúc. Gọi các con của A theo thứ tự từ trái sang phải là B_1, B_2, …, B_k. Nếu các đỉnh này đều là lá thì cây gốc A chỉ có một đỉnh có ký hiệu không kết thúc. Giả sử trong các đỉnh này có các đỉnh trong là C_1, C_2, …, C_m. Xét các cây con mà gốc của nó là C_1, C_2, …,C_m. Gọi α_i là kết quả của cây suy dẫn gốc C_i. Theo giả thiết quy nạp, $\alpha_i \in L(G_i)$. Vì tập các quy tắc trong G_{Ci} chứa trong tập các quy tắc trong G_A nên ta có các suy dẫn trong G_A là $C_1 \models \alpha_1$, $C_2 \models \alpha_2$, …, $C_m \models \alpha_m$. Sử dụng các suy dẫn này và quy tắc $A \rightarrow B_1 B_2 \ldots B_k$, ta nhận được:

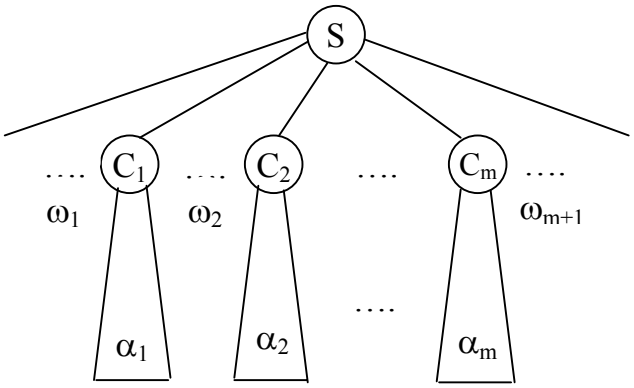
$$A \models B_1 B_2 \ldots B_k \models \omega_1 C_1 \omega_2 C_2 \ldots \omega_m C_m \omega_{m+1} \models \ldots \models \omega_1 \alpha_1 \omega_2 \alpha_2 \ldots \omega_m \alpha_m \omega_{m+1}.$$

Do kết quả của cây suy dẫn trong G_A là ω nên $\omega = \omega_1 \alpha_1 \omega_2 \alpha_2 \ldots \omega_m \alpha_m \omega_{m+1}$ hay $\omega \in L(G_A)$.

Đảo lại, ta cần chứng minh rằng nếu có suy dẫn $A \models \omega$ ($\omega \neq \varepsilon$) trong G_A thì có thể xây dựng một cây suy dẫn trong G_A có kết quả là ω. Mệnh đề này được chứng minh bằng quy nạp theo độ dài của suy dẫn.

Trước hết, nếu $A \models \omega = b_1 b_2 \ldots b_k$ (suy dẫn một bước) thì có thể xây dựng một cây có gốc là A và các con từ trái sang phải lần lượt là b_1, b_2, …, b_k.

Giả sử mệnh đề đúng với mọi suy dẫn có độ dài không lớn hơn n. Cho suy dẫn trong G_A là $A \models \omega$ có độ dài n+1. Giả sử quy tắc đầu tiên trong suy dẫn này là $A \rightarrow B_1 B_2 \ldots B_k$ và C_1, C_2, …, C_m là các ký hiệu không kết thúc trong các B_i ($1 \leq i \leq k$), có nghĩa là $B_1 B_2 \ldots B_k = \omega_1 C_1 \omega_2 C_2 \ldots \omega_m C_m \omega_{m+1}$, ở đây $C_i \models \alpha_i$ có độ dài không vượt quá n. Theo giả thiết quy nạp, tồn tại các cây T_i của G_{Ci} mà kết quả của nó là α_i và do đó ta có thể xây dựng trong G_A cây suy dẫn có kết quả là ω như sau:



3.1.3. Định nghĩa: Cho văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$. Ta nói văn phạm G là nhập nhằng hay đa nghĩa nếu tồn tại một xâu ω là kết quả của hai cây suy dẫn khác nhau trong G.

Trong trường hợp ngược lại, ta nói G là không nhập nhằng hay đơn nghĩa.

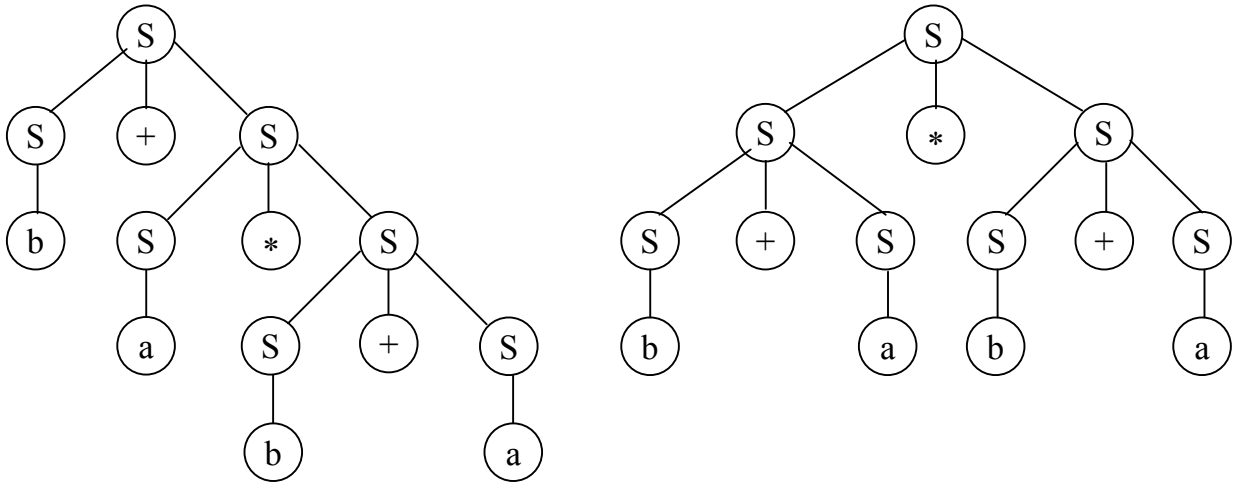
Một văn phạm phi ngữ cảnh được gọi là nhập nhằng vĩnh cửu nếu không tồn tại văn phạm phi ngữ cảnh đơn nghĩa nào tương đương với nó.

Ngôn ngữ do văn phạm G sinh ra gọi là ngôn ngữ nhập nhằng nếu G là văn phạm nhập nhằng.

Thí dụ 2: Văn phạm phi ngữ cảnh sau là nhập nhằng:

$$G = <\{a, b, +, *\}, \{S\}, S, \{S\to S+S, S\to S*S, S\to a, S\to b\}>,$$

vì xâu ω=b+a∗b+a có hai suy dẫn trái khác nhau trong G.



Cùng với văn phạm G ở trên, văn phạm

$$G' = <\{a, b, +, *\}, \{S, A\}, S, \{S\to A, S\to A+S, A\to A*A, A\to a, A\to b\}>$$

là đơn nghĩa và L(G')=L(G).

Trong một văn phạm phi ngữ cảnh có thể có nhiều yếu tố thừa, chẳng hạn có những ký hiệu không hề tham gia vào quá trình sinh các ngôn ngữ, hoặc có những quy tắc dạng A→B chỉ làm mất thời gian trong quá trình hình thành các xâu của ngôn ngữ. Vì lẽ đó cần loại bỏ những yếu tố dư thừa không có ích trong việc sinh ngôn ngữ, sao cho việc loại bỏ đó không làm ảnh hưởng tới quá trình sinh ngôn ngữ. Điều đó có nghĩa là chỉ cần giữ lại các ký hiệu và các quy tắc có ích trong văn phạm G mà chúng thực sự là cần thiết trong quá trình sinh ngôn ngữ mà thôi.

3.1.4. Định nghĩa: Cho văn phạm phi ngữ cảnh G=<Σ, Δ, S, P>. X được gọi là ký hiệu có ích nếu tồn tại suy dẫn $S \models \alpha X\beta \models \omega$, trong đó $\alpha, \beta \in (\Sigma\cup\Delta)^*$, $X\in\Sigma\cup\Delta$ và $\omega\in\Sigma^*$.

Nếu ký hiệu X không thoả mãn điều kiện trên thì X được gọi là ký hiệu thừa. Như vậy X là ký hiệu thừa nếu từ X không thể dẫn ra một xâu $\omega\in\Sigma^*$. Ký hiệu X có tính chất như thế còn được gọi là ký hiệu vô sinh. Nếu từ ký hiệu đầu S không dẫn được một xâu nào có chứa ký hiệu X thì ta nói ký hiệu X là ký hiệu không đến

được. Như vậy một ký hiệu là thừa nếu nó là ký hiệu vô sinh hoặc là không đến được.

3.1.5. Bổ đề (loại ký hiệu vô sinh)**:** Cho văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$ với $L(G)\neq\varnothing$. Khi đó tồn tại văn phạm phi ngữ cảnh $G'=<\Sigma, \Delta', S, P'>$ tương đương với G sao cho mỗi $A\in\Delta'$ có một xâu $\omega\in\Sigma^*$ để $A \models \omega$.

Chứng minh: Từ tập quy tắc P của G, ta xây dựng Δ' như sau:

– Nếu trong P có quy tắc dạng $A\to\omega$ với $A\in\Delta$, $\omega\in\Sigma^*$ thì kết nạp A vào Δ'.

– Nếu $A\to X_1X_2\ldots X_n$ là quy tắc trong P mà $X_i\in\Sigma$ hoặc X_i là biến đã được kết nạp vào Δ' thì đưa A vào Δ'.

Cứ tiếp tục xét các quy tắc trong P, ta sẽ xây dựng các ký hiệu cho tập Δ'. Vì P là hữu hạn nên quá trình sẽ được dừng lại sau một số hữu hạn bước. Khi đó ta xây dựng được tập Δ'.

Ta xây dựng tiếp tập quy tắc P' gồm các quy tắc trong P mà các ký hiệu có mặt trong đó đều thuộc tập $\Sigma\cup\Delta'$.

3.1.6. Bổ đề (loại ký hiệu không đến được)**:** Cho văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$. Khi đó tồn tại văn phạm phi ngữ cảnh $G'=<\Sigma', \Delta', S, P'>$ tương đương với G sao cho mỗi $X\in\Sigma'\cup\Delta'$ có $\alpha, \beta\in(\Sigma'\cup\Delta')^*$ để cho $S \models \alpha X\beta$.

Chứng minh: Xây dựng tập Σ' và Δ' như sau:

Đưa ký hiệu S vào Δ'. Nếu một biến A đã được kết nạp vào Δ' và $A\to\alpha$, ở đây $\alpha\in(\Sigma'\cup\Delta')^*$ thì ta kết nạp các ký hiệu biến trong α vào Δ', còn các ký hiệu kết thúc trong α thì kết nạp vào Σ'.

Thủ tục kết nạp trên sẽ ngừng khi không còn bổ sung thêm được bất kỳ ký hiệu nào nữa vào các tập Σ' và Δ'.

Tập quy tắc P' được xây dựng như sau:

P' bao gồm mọi quy tắc trong P mà các ký hiệu thuộc tập $\Sigma'\cup\Delta'$. Với cách xây dựng đó, ta có $L(G)=L(G')$, trong đó G' gồm các ký hiệu đến được. Từ hai bổ đề trên ta có:

3.1.7. Định lý: Mọi ngôn ngữ phi ngữ cảnh khác rỗng đều có thể được sinh ra từ một văn phạm phi ngữ cảnh không có ký hiệu thừa.

3.1.8. Định nghĩa: Cho văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$. Quy tắc trong P có dạng $A\to B$, ở đây $A, B\in\Delta$, được gọi là quy tắc đơn hay phép đổi tên.

Quy tắc đơn có tác dụng làm kéo dài quá trình sinh ra ngôn ngữ, vì vậy ta sẽ tìm cách loại quy tắc đơn mà không làm ảnh hưởng tới quá trình sinh ra ngôn ngữ của văn phạm đã cho.

Lưu ý rằng quy tắc $A\to a$, với $A\in\Delta$ và $a\in\Sigma$ không phải là quy tắc đơn.

3.1.9. Định lý: Đối với mọi văn phạm phi ngữ cảnh mà trong tập các quy tắc của nó có quy tắc đơn thì tồn tại một văn phạm phi ngữ cảnh tương đương với nó mà trong tập các quy tắc của nó không chứa quy tắc đơn.

Chứng minh: Giả sử $G=<\Sigma, \Delta, S, P>$ là văn phạm phi ngữ cảnh có chứa quy tắc đơn (và không chứa ký hiệu thừa). Ta xây dựng văn phạm phi ngữ cảnh $G'=<\Sigma, \Delta, S, P'>$ tương đương với G và không chứa quy tắc đơn.

Đưa tất cả các quy tắc không đơn của P vào P'. Nếu trong P có quy tắc $A\to B$, với $A, B\in\Delta$, thì tồn tại suy dẫn $S\models\alpha A\beta\models\alpha B\beta\models\alpha\omega\beta$, ở đây $\alpha,\beta\in(\Sigma\cup\Delta)^*$, $\omega\in\Sigma^*$ do Δ gồm các ký hiệu không thừa.

Vậy thay cho $A\to B$, ta đưa vào P' quy tắc $S\to\alpha A\beta$ và $A\to\omega$ đều là các quy tắc không đơn nhưng chức năng sinh ngôn ngữ tương đương với quy tắc $A\to B$.

Thí dụ 3: Văn phạm phi ngữ cảnh

$G= <\{a, +, *\}, \{S, A, B\}, S, \{S\to S+A, S\to A, A\to A*B, A\to B, B\to a\}>$

tương đương với văn phạm phi ngữ cảnh sau không còn các quy tắc đơn:

$G' = <\{a, +, *\}, \{S, A, B\}, S, \{S\to S+A, A\to A*B, B\to a, S\to A*B, A\to a, S\to a\}>$.

3.1.10. Định lý: Cho $G=<\Sigma, \Delta, S, P>$ là văn phạm mà các quy tắc của nó có dạng $A\to\alpha$, ở đây $A\in\Delta$, $\alpha\in(\Sigma\cup\Delta)^*$. Khi đó L(G) là ngôn ngữ phi ngữ cảnh.

Chứng minh: Đối với văn phạm G như ở trên, ta cũng có thể định nghĩa cây suy dẫn bằng cách sử dụng từ ε và như vậy ta có thể xác định được rằng $A\models\varepsilon$ hay không? Để làm việc này, ta chỉ cần xét các cây suy dẫn trong G trong đó không có con đường nào dài hơn số phần tử của Δ. Giả sử $A_1, A_2, \ldots, A_n$ là các ký hiệu không kết thúc mà đối với chúng ta có $A_i\models\varepsilon$. Cuối cùng ta giả thiết rằng S không có mặt trong vế phải của bất kỳ quy tắc nào của P. Ta xây dựng văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P'>$, trong đó P' chứa $S\to\varepsilon$ nếu $S\models\varepsilon$ và mọi quy tắc dạng $A\to\alpha_1\ldots\alpha_k$ nếu $A\to C_1\ldots C_k$ $(k\geq 1)$ là một quy tắc trong P và $\alpha_i=C_i$, trong đó $C_i\in\Sigma\cup\Delta\setminus\{A_1, \ldots, A_n\}$ hoặc $\alpha_i\in\{C_i, \varepsilon\}$, trong đó $C_i\in\{A_1, \ldots, A_n\}$, ở đây ta ràng buộc mỗi một α_i không thể bằng ε.

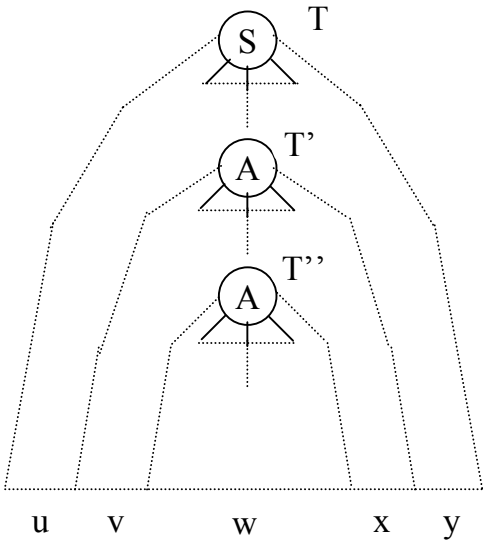
Bằng quy nạp theo độ dài suy dẫn, ta có thể chỉ ra rằng với $\omega\in\Sigma^*$, $S\overset{G}{\models}\omega$ khi và chỉ khi $S\overset{G'}{\models}\omega$.

3.1.11. Định lý: Cho L là một ngôn ngữ phi ngữ cảnh. Khi đó tồn tại số tự nhiên n thoả mãn điều kiện với mỗi $\omega\in L$ có $d(\omega)>n$, tồn tại các từ u, v, w, x và y sao cho $\omega=uvwxy$, ở đây $d(w)\geq 1$, $d(vx)\geq 1$, $d(vwx)\leq n$ và với mọi i $(i\geq 0)$, $uv^iwx^iy\in L$.

Chứng minh: Giả sử $G=<\Sigma, \Delta, S, P>$ là văn phạm phi ngữ cảnh sinh ra L và P không chứa các quy tắc đơn. Gọi m là số phần tử của Δ và $n=l^{m+1}$, với l là độ dài cực đại của tất cả các vế phải của các quy tắc trong P.

Lấy $\omega\in L(G)$ sao cho $d(\omega)>n=l^{m+1}$. Khi đó tồn tại cây suy dẫn mà kết quả là ω. Giả sử T là cây suy dẫn có kết quả ω mà chiều cao của nó nhỏ nhất. Dễ dàng

thấy rằng đối với cây suy dẫn chiều cao h thì độ dài của từ nhận được không vượt quá l^h. Vì ω là kết quả của cây T nên $d(\omega) \le l^h$. Mặt khác từ $d(\omega) > l^{m+1}$ nên suy ra $h > m+1$. Điều này có nghĩa là trong T tồn tại một đường đi từ gốc đến lá có nhiều hơn m+1 nút và trên đường này có ít nhất m+1 ký hiệu không kết thúc. Vì rằng số phần tử của Δ là m nên trên đường này có ít nhất hai nút có cùng một tên A. Gọi T' và T'' là hai cây con lớn nhất có cùng gốc ký hiệu A nằm trên đường đi đã chỉ ra.



Bây giờ ta xét kết quả ω của cây T và phân tích nó như trên hình vẽ, ω=uvwxy. Vì rằng trong G không có quy tắc đơn, nên từ nút A có ít nhất là hai nhánh đi ra, do đó $d(vx) \ge 1$ và $d(w) \ge 1$.

Độ cao của cây T' cùng lắm là m+1, ta đã chứng minh các nút được ký hiệu bởi A sao cho phần đầu tiên trong T' của con đường đã xét mỗi ký hiệu chỉ xuất hiện một lần, do đó ta có $d(vwx) \le l^{m+1} = n$. Từ cây T, ta có $S \models uAy$. Tiếp theo ta xét cây T' mà nó là một cây suy dẫn của văn phạm $G_A = <\Sigma, \Delta, A, P>$, ta có $A \models vAx$ trong G_A, do đó $A \models vAx$ trong G. Tiếp đến xét cây T'', ta có $A \models w$ trong G_A, do đó $A \models w$ trong G. Cuối cùng, $S \models uAy$, $A \models vAx$ và $A \models w$. Từ đó, ta có

$S \models uAy \models uwy$,

$S \models uAy \models uvAxy \models uvwxy$,

$S \models uAy \models uvAxy \models \ldots \models uv^iwx^iy$.

3.1.12. Hệ quả: Tồn tại ngôn ngữ cảm ngữ cảnh mà nó không phải là phi ngữ cảnh.

Chứng minh: Trong Thí dụ 10 của Chương I, ta đã biết ngôn ngữ $L=\{a^mb^mc^m \mid m \ge 1\}$ được sinh bởi văn phạm cảm ngữ cảnh $G = <\{a, b, c\}, \{S, A, B, C\}, S, P>$, trong đó $P = \{S \to aSAC, S \to abC, CA \to BA, BA \to BC, BC \to AC, bA \to bb, C \to c\}$. Ta sẽ chứng minh rằng không tồn tại văn phạm phi ngữ cảnh nào sinh ra L.

Giả sử tồn tại một văn phạm như vậy. Theo định lý trên, tồn tại số tự nhiên n sao cho với mọi từ ω có d(ω)>n đều được phân tích thành ω=uvwxy, d(vx)≥1 và $uv^iwx^iy \in L$ với mọi i=0, 1, 2, …

Lấy từ $\omega=a^nb^nc^n$, với d(ω)=3n>n. Do đó ta có $\omega=a^nb^nc^n=uvwxy$, d(vx)≥1.

Trước hết ta thấy rằng nếu trong v hoặc trong x chứa hai trong ba ký hiệu a, b, c thì $uv^iwx^iy \notin L$, do đó x chỉ chứa một loại ký hiệu và trong trường hợp này thì với i đủ lớn, số ký hiệu a, b, c trong uv^iwx^iy không bằng nhau và vì vậy $uv^iwx^iy \notin L$. Điều này mâu thuẫn với kết quả trên. Vậy không tồn tại văn phạm phi ngữ cảnh nào sinh ra L.

3.2. ÔTÔMAT ĐẨY XUỐNG.

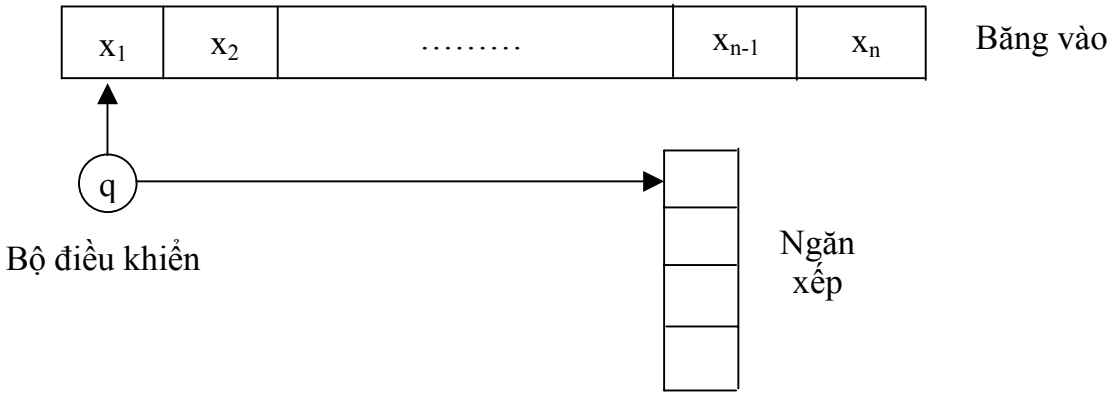
Như ta đã biết, lớp các ngôn ngữ chính quy do văn phạm chính quy sinh ra cũng trùng với lớp các ngôn ngữ được đoán nhận bởi ôtômat hữu hạn (đơn định hoặc không đơn định).

Tương tự, ta sẽ thấy trong phần này là lớp các ngôn ngữ phi ngữ cảnh do các văn phạm phi ngữ cảnh sinh ra sẽ trùng với lớp các ngôn ngữ được đoán nhận bởi ôtômat đẩy xuống không đơn định (Nondeteministic Pushdown Automata).

Đáng lưu ý, chỉ có lớp ôtômat đẩy xuống không đơn định mới có thể đoán nhận hết lớp ngôn ngữ phi ngữ cảnh. Còn ôtômat đẩy xuống đơn định chỉ có khả năng đoán nhận được lớp con thực sự của lớp ngôn ngữ phi ngữ cảnh mà thôi. Tuy vậy, lớp ngôn ngữ được đoán nhận bởi lớp các ôtômat đẩy xuống đơn định là khá rộng, nó bao gồm phần lớn các ngôn ngữ lập trình hiện nay. Ở đây, ta chỉ đề cập tới ôtômat đẩy xuống không đơn định mà người ta thường gọi tắt là NDPA hay gọn hơn là PA.

3.2.1. Mở đầu:

Một ôtômat đẩy xuống bao gồm một băng vào, một ngăn xếp và một bộ điều khiển như hình dưới đây:



Ôtômat đẩy xuống cũng như ôtômat hữu hạn có bộ điều khiển là tập hữu hạn các trạng thái. Đầu đọc của ôtômat cho phép đọc lần lượt các ký hiệu trên băng vào

từ trái sang phải. Ngoài ra, ôtômat đẩy xuống còn có thêm băng làm việc (ngăn xếp) hay stack, nhờ có nó mà bộ nhớ của ôtômat đẩy xuống được tăng lên so với khả năng nhớ của ôtômat hữu hạn. Ngăn xếp được tổ chức theo nguyên tắc ký hiệu vào sau thì ra trước, giống như ổ nạp đạn. Khi đưa ký hiệu vào ngăn xếp thì ký hiệu đó được trở thành ký hiệu đầu của ngăn xếp. Khi ngăn xếp đọc thì ký hiệu đó là ký hiệu trên cùng và khi ký hiệu đó được xong thì nó sẽ bị loại khỏi ngăn xếp. Một ngăn xếp như vậy còn được gọi là một danh sách đẩy xuống.

Căn cứ vào trạng thái hiện tại của bộ điều khiển, ký hiệu vào mà đầu đọc đang quan sát và ký hiệu đầu của ngăn xếp, ôtômat đẩy xuống sẽ chuyển sang một trạng thái mới nào đó và đồng thời đầu đọc có thể được chuyển sang ô bên phải. Nếu đầu đọc chuyển sang ô bên phải thì ta gọi quá trình trên là một bước chuyển. Ngược lại, nếu ký hiệu vào không ảnh hưởng tới bước chuyển thì ta gọi đó là bước chuyển "nhắm mắt" và trong bước chuyển đó đầu đọc vẫn đứng yên tại chỗ. Thực chất của bước chuyển "nhắm mắt" là sự tạm ngừng quan sát băng vào để chấn chỉnh lại ngăn xếp.

Có hai cách đoán nhận xâu vào của ôtômat đẩy xuống:

– Cách 1: Xâu vào được đọc xong và ôtômat đẩy xuống chuyển được về một trạng thái kết thúc nào đó.

– Cách 2: Xâu vào được đọc xong và ngăn xếp trở thành rỗng.

Sau này ta sẽ chỉ ra hai cách đoán nhận trên là tương đương.

Thí dụ 4: Cho văn phạm phi ngữ cảnh:

$$G = <\{0, 1, c\}, \{S\}, S, \{S\rightarrow 0S0, S\rightarrow 1S1, S\rightarrow c\}>.$$

Dễ dàng thấy rằng $L(G)=\{\omega c\omega^R \mid \omega\in\{0, 1\}^*\}$ (ω^R là xâu viết các ký hiệu của xâu ω theo một thứ tự ngược lại). Chẳng hạn, đối với xâu ω=010011 thì xâu $\omega c\omega^R$=010011c110010 sẽ có dẫn xuất sau đây: (S, 0S0, 01S10, 010S010, 0100S0010, 01001S10010, 010011S110010, 010011c110010).

Mặt khác xâu $\omega c\omega^R$ có thể được đoán nhận bởi ôtômat đẩy xuống như sau:

Trước hết đặt xâu $x=\omega c\omega^R$ lên băng vào. Đầu đọc dịch chuyển từ trái sang phải. Ban đầu ngăn xếp rỗng. Ôtômat đẩy xuống có hai trạng thái p, q trong đó p là trạng thái đầu. Khi ở trạng thái đầu p, đầu đọc đọc ký hiệu trên băng vào là 0 hoặc 1 và nó đưa ký hiệu đó vào ngăn xếp. Trạng thái của ôtômat đẩy xuống lúc đó vẫn là p. Đầu đọc dịch chuyển sang bên phải một ô và đọc ký hiệu trên ô mới này. Nếu ký hiệu này là 0 hoặc 1 thì ôtômat đẩy xuống đưa ký hiệu đó vào ngăn xếp, trạng thái của ôtômat vẫn là p và đầu đọc lại được chuyển sang phải một ô.

Quá trình đó vẫn tiếp tục cho tới khi đầu đọc gặp ký hiệu c. Khi đó ôtômat sẽ chuyển trạng thái p sang trạng thái q và đầu đọc chuyển sang phải một ô. Tại thời điểm này ngăn xếp xuất hiện xâu ω. Ký hiệu bên phải nhất của ω nằm trên cùng

của ngăn xếp, còn trạng thái của ôtômat là q. Đầu đọc đang chỉ ô bên phải ký hiệu c. Nếu ký hiệu của ô này là ký hiệu của ô trên cùng của ngăn xếp thì ôtômat đẩy xuống loại ký hiệu trên cùng ra khỏi ngăn xếp, ký hiệu dưới nó lại lên vị trí đầu của ngăn xếp, ôtômat đẩy xuống chuyển đầu đọc sang phải một ô, còn trạng thái vẫn là q. Quá trình đó cứ tiếp tục và có hai khả năng xảy ra:

1) Ôtômat đẩy xuống đọc hết xâu x và ngăn xếp trở thành rỗng thì ôtômat dừng lại và đoán nhận được xâu $x=\omega c\omega^R$.

2) Các ký hiệu ở ngăn xếp chưa bị loại hết thì đầu đọc gặp ký hiệu trên xâu vào khác ký hiệu trên cùng của ngăn xếp. Trong trường hợp này ôtômat đẩy xuống không đoán nhận xâu x.

Nhờ có ngăn xếp mà ôtômat đẩy xuống có khả năng nhớ được nửa đầu của xâu $x=\omega c\omega^R$ với ω có độ dài tuỳ ý và sau đó nó so sánh dần với nửa cuối ω^R của x. Ôtômat hữu hạn không có khả năng này.

Bây giờ ta định nghĩa một cách hình thức ôtômat đẩy xuống như sau:

3.2.2. Định nghĩa: Một ôtômat đẩy xuống là một bộ bảy:

$$M = <Q, \Sigma, \Delta, \delta, q_0, z_0, F>,$$

trong đó,

– Σ là một bảng chữ, gọi là bảng chữ vào, mỗi ký hiệu trong Σ gọi là ký hiệu vào;

– Q là một tập hữu hạn, khác rỗng các trạng thái sao cho $\Sigma \cap Q = \varnothing$;

– Δ là một bảng chữ mà các ký hiệu của nó gọi là các ký hiệu của ngăn xếp;

– $z_0 \in \Delta$ là ký hiệu đặc biệt, gọi là ký hiệu đáy của ngăn xếp (còn gọi là ký hiệu đầu của danh sách đẩy xuống);

– $q_0 \in Q$ gọi là trạng thái đầu;

– $F \subset Q$ gọi là tập các trạng thái kết thúc;

– δ: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Delta \longrightarrow \mathcal{P}(Q \times \Delta^*)$ gọi là hàm chuyển của M.

Một đẳng thức dạng:

$$\delta(q, a, z)=\{<q_1, \gamma_1>, <q_2, \gamma_2>, \ldots, <q_m, \gamma_m>\},$$

trong đó $q, q_i \in Q$, $a \in \Sigma$, $z \in \Delta$, $\gamma_i \in \Delta^*$, i=1, …, m được gọi là một bước chuyển của ôtômat đẩy xuống M. Nó đang ở trạng thái q, đọc ký hiệu a ở băng vào và z là ký hiệu ở đỉnh ngăn xếp. Khi đó nó chuyển sang trạng thái q_i, thay ký hiệu z ở đỉnh ngăn xếp bằng xâu γ_i, i=1, …, m, đồng thời chuyển đầu đọc sang bên phải một ô. Quy ước rằng khi đưa γ_i vào ngăn xếp, ký hiệu bên trái nhất của γ_i sẽ nàm ở phần dưới, còn ký hiệu bên phải nhất của γ_i sẽ nằm ở ô đầu ngăn xếp.

Một đẳng thức dạng:

$$\delta(q, \varepsilon, z)=\{<q_1, \gamma_1>, <q_2, \gamma_2>, \ldots, <q_m, \gamma_m>\}$$

diễn tả một bước chuyển "nhắm mắt" của ôtômat đẩy xuống M: Ôtômat ở trạng thái q, ký hiệu z ở đỉnh ngăn xếp. Khi đó ôtômat đẩy xuống chuyển trạng thái q về

q_i và thay $z \in \Delta$ ở đỉnh ngăn xếp bởi xâu γ_i ($1 \le i \le m$), còn đầu đọc thì không dịch chuyển.

Ở một thời điểm, tình huống tức thời của ôtômat đẩy xuống xác định bởi ba yếu tố:

– Xâu $\gamma \in \Delta^*$ trong ngăn xếp (với quy ước là ký hiệu bên trái nhất của γ nằm ở đáy ngăn xếp).

– Trạng thái $q \in Q$.

– Phần xâu vào $x \in \Sigma^*$ chưa được đọc đến trên băng vào (với quy ước ký hiệu bên trái nhất của x là ký hiệu sẽ được đọc tiếp).

3.2.3. Định nghĩa: Cho ôtômat đẩy xuống $M = <Q, \Sigma, \Delta, \delta, q_0, z_0, F>$. Một hình trạng (hay cấu hình) của M là một bộ ba $<q, \alpha, \beta>$, trong đó $q \in Q$, $\alpha \in \Sigma^*$, $\beta \in \Delta^*$.

Giả sử $\alpha = a_1a_2\ldots a_k \in \Sigma^*$, $\beta = x_1x_2\ldots x_m \in \Delta^*$. Dưới tác động của ánh xạ chuyển trạng thái, M có thể chuyển từ hình trạng này sang hình trạng khác theo nguyên tắc sau:

1) Nếu $<p, \gamma> \in \delta(q, a_1, x_m)$ thì hình trạng $<q, a_1a_2\ldots a_k, x_1x_2\ldots x_m>$ có thể chuyển sang hình trạng $<p, a_2\ldots a_k, x_1x_2\ldots x_{m-1}\gamma>$ và ký hiệu:

$$<q, a_1a_2\ldots a_k, x_1x_2\ldots x_m> \vdash <p, a_2\ldots a_k, x_1x_2\ldots x_{m-1}\gamma>.$$

2) Nếu $<p, \gamma> \in \delta(q, \varepsilon, x_m)$ thì hình trạng $<q, a_1a_2\ldots a_k, x_1x_2\ldots x_m>$ có thể chuyển sang hình trạng $<p, a_1a_2\ldots a_k, x_1x_2\ldots x_{m-1}\gamma>$ và ký hiệu:

$$<q, a_1a_2\ldots a_k, x_1x_2\ldots x_m> \vdash <p, a_1a_2\ldots a_k, x_1x_2\ldots x_{m-1}\gamma>.$$

3.2.4. Định nghĩa: Cho ôtômat đẩy xuống $M = <Q, \Sigma, \Delta, \delta, q_0, z_0, F>$ và $\omega \in \Sigma^*$. Ta nói rằng ôtômat M đoán nhận từ ω theo tập trạng thái kết thúc nếu tồn tại một dãy hữu hạn các hình trạng $K_0, K_1, \ldots, K_n$ sao cho:

1) $K_0 = <q_0, \omega, z_0>$, gọi là hình trạng đầu;

2) $K_n = <p, \varepsilon, \gamma>$, $p \in F$, gọi là hình trạng kết thúc;

3) $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_n$.

Ký hiệu $T(M) = \{\omega \in \Sigma^* | \omega$ được đoán nhận bởi M theo tập trạng thái kết thúc$\}$.

T(M) được gọi là ngôn ngữ được đoán nhận bởi ôtômat đẩy xuống M theo tập trạng thái kết thúc.

3.2.5. Định nghĩa: Cho ôtômat đẩy xuống $M = <Q, \Sigma, \Delta, \delta, q_0, z_0, F>$ và $\omega \in \Sigma^*$. Ta nói rằng ôtômat M đoán nhận từ ω theo ngăn xếp rỗng nếu tồn tại một dãy hữu hạn các hình trạng $K_0, K_1, \ldots, K_n$ sao cho:

1) $K_0 = <q_0, \omega, z_0>$, gọi là hình trạng đầu;

2) $K_n = <p, \varepsilon, \varepsilon>$ gọi là hình trạng kết thúc;

3) $K_0 \vdash K_1 \vdash K_2 \vdash \ldots \vdash K_n$.

Ký hiệu $N(M) = \{\omega \in \Sigma^* | \omega$ được đoán nhận bởi M theo ngăn xếp rỗng$\}$.

N(M) được gọi là ngôn ngữ được đoán nhận bởi ôtômat đẩy xuống M theo ngăn xếp rỗng.

Thí dụ 5: Cho ôtômat đẩy xuống $M=<\{q_0, q_1, q_2\}, \{a, b\}, \{z_0, z_1\}, \delta, q_0, z_0, \{q_2\}>$, trong đó $\delta(q_0, \varepsilon, z_0)=\{<q_0, \varepsilon>\}$, $\delta(q_0, a, z_0)=\{<q_1, z_0z_1>\}$, $\delta(q_1, a, z_1)=\{<q_1, z_1z_1>\}$, $\delta(q_1, b, z_1)=\{<q_2, \varepsilon>\}$, $\delta(q_2, b, z_1)=\{<q_2, \varepsilon>\}$, $\delta(q_2, \varepsilon, z_0)=\{<q_0, \varepsilon>\}$ (ở đây, ảnh của các bộ ba khác qua δ được hiểu là $\varnothing$).

Xét các từ α=aabb và β=abaab. Ta có:

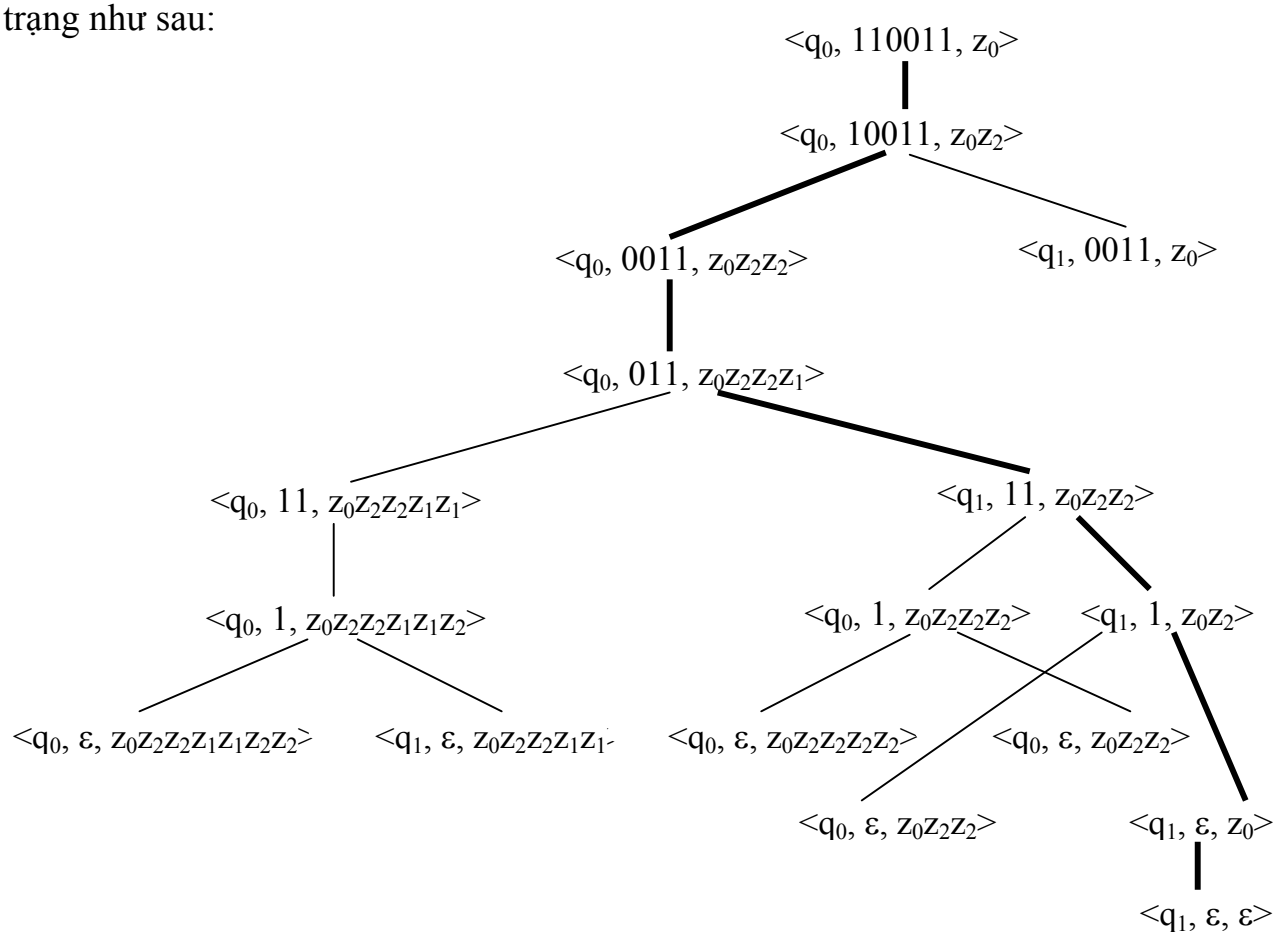
$<q_0, aabb, z_0>\vdash <q_1, abb, z_0z_1> \vdash <q_1, bb, z_0z_1z_1> \vdash <q_2, b, z_0z_1> \vdash <q_2, \varepsilon, z_0>$
$\vdash <q_0, \varepsilon, \varepsilon>$.

$<q_0, abaab, z_0>\vdash <q_1, baab, z_0z_1> \vdash <q_2, aab, z_0>$.

Do đó $\alpha \in N(M)$, $\beta \notin N(M)$, $\beta \notin T(M)$. Tổng quát, ta có thể chứng minh được rằng $N(M)=T(M)=\{a^nb^n \mid n\geq 0\}$.

Thí dụ 6: Cho ôtômat đẩy xuống $M=<\{q_0, q_1\}, \{0, 1\}, \{z_0, z_1, z_2\}, \delta, q_0, z_0>$, trong đó $\delta(q_0, 0, z_0)=\{<q_0, z_0z_1>\}$, $\delta(q_0, 0, z_1)=\{<q_0, z_1z_1>, <q_1, \varepsilon>\}$,
$\delta(q_0, 0, z_2)=\{<q_0, z_2z_1>\}$, $\delta(q_0, 1, z_0)=\{<q_0, z_0z_2>\}$, $\delta(q_0, 1, z_1)=\{<q_0, z_1z_2>\}$,
$\delta(q_0, 1, z_2)=\{<q_0, z_2z_2>, <q_1, \varepsilon>\}$, $\delta(q_1, 0, z_1)=\{<q_1, \varepsilon>\}$,
$\delta(q_1, 1, z_2)=\{<q_0, z_2z_2>, <q_1, \varepsilon>\}$, $\delta(q_0, \varepsilon, z_0)=\{<q_1, \varepsilon>\}$, $\delta(q_1, \varepsilon, z_0)=\{<q_1, \varepsilon>\}$.

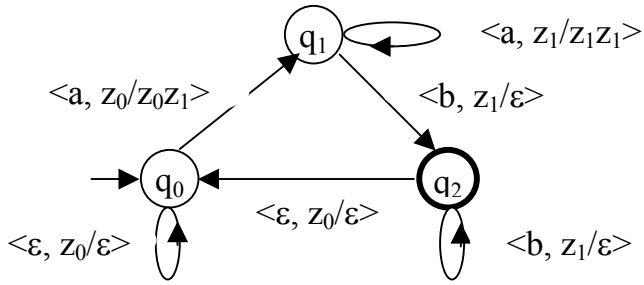
Xét ω=110011, ta có thể vẽ cây biểu diễn các khả năng biến đổi của các hình trạng như sau:



Đường in đậm là dãy dịch chuyển từ hình trạng đầu $<q_0, \omega, z_0>$ đến hình trạng cuối $<q_1, \varepsilon, \varepsilon>$ theo ngăn xếp rỗng.

3.2.6. Chú ý: Cũng như đối với ôtômat hữu hạn, ta có thể mô tả ôtômat đẩy xuống bằng đồ thị chuyển. Đó là một đa đồ thị có hướng, có khuyên G với tập đỉnh của G là Q. Với $a\in\Sigma\cup\{\varepsilon\}$, $p, q\in Q$, $z\in\Delta$, $\gamma\in\Delta^*$, nếu $<p, \gamma>\in\delta(q, a, z)$ thì sẽ có một cung từ q tới p được gán nhãn là $(a, z/\gamma)$.

Chẳng hạn, đồ thị chuyển của ôtômat đẩy xuống M trong Thí dụ 5 là:



3.2.7. Định lý: Cho L là một ngôn ngữ phi ngữ cảnh. Khi đó tồn tại một ôtômat đẩy xuống M đoán nhận L theo tập trạng thái kết thúc.

Chứng minh: Giả sử $G=<\Sigma, \Delta, S, P>$ là văn phạm phi ngữ cảnh sinh ra ngôn ngữ L. Ta xây dựng ôtômat đẩy xuống $M=<Q, \Sigma, \Gamma, \delta, q_0, z_0, F>$ đoán nhận L với:

– $Q=\{q_0, q_1, q_2\}$ là tập các trạng thái,

– $\Gamma=\Sigma\cup\Delta\cup\{\%\}$ là tập các ký hiệu ngăn xếp, ký hiệu % là không thuộc $\Sigma\cup\Delta$,

– $z_0=S$ là ký hiệu đầu của ngăn xếp,

– $q_0\in Q$ là trạng thái đầu,

– $F=\{q_2\}$ là tập các trạng thái kết thúc,

– Hàm chuyển δ: $Q \times (\Sigma\cup\{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma^*)$ được định nghĩa qua các biểu thức sau:

1) $\delta(q_1, \varepsilon, z)=\{<q_1, w^R> \mid z\to w\in P, z\in\Delta, w\in\Gamma^*\}$.

2) $\delta(q_1, a, a)=\{<q_1, \varepsilon>\}$, với mọi $a\in\Sigma$.

3) $\delta(q_1, \varepsilon, \%)=\{<q_2, \%>\}$.

4) $\delta(q_0, \varepsilon, S)=\{<q_1, \%S>\}$.

Giả sử $w\in L(G)$. Khi đó tồn tại dãy suy dẫn đầy đủ trong G là

$$D=(S, u_1z_1v_1, u_1u_2z_2v_2, \ldots, u_1\ldots u_{n-1}z_{n-1}v_{n-1}, u_1u_2\ldots u_n=w),$$

ở đây $z_i\in\Delta$, $u_i\in\Sigma$, $v_i\in\Sigma\cup\Delta$ $(1\le i\le n-1)$ và $u_n\in\Sigma^*$.

Dựa vào các quy tắc của G sử dụng trong dãy suy dẫn đầy đủ D của xâu w, ôtômat đẩy xuống M đoán nhận w theo nguyên tắc sau:

Áp dụng hàm chuyển trong các biểu thức 4, 1, 2, ta có:

$$<q_0, w, z_0>\vdash <q_1, w, \%S>\vdash <q_1, u_1u_2\ldots u_n, \%v_1^Rz_1u_1^R>\vdash <q_1, u_2u_3\ldots u_n, \%v_1^Rz_1>.$$

Giả sử các quy tắc tiếp theo (sau quy tắc $S\to u_1z_1v_1$) trong D là $z_i\to x_i\in P$ $(i=1, 2, \ldots, n-2)$ với $z_i\in\Delta$, $x_i\in\Sigma\cup\Delta$ sao cho $x_iv_i=u_{i+1}z_{i+1}v_{i+1}$. Khi đó M sau thời

điểm thực hiện quy tắc $S\to u_1z_1v_1$ nó có hình trạng $<q_1, u_2u_3\ldots u_n, \%v_1^Rz_1>$. Hình trạng của M khi áp dụng quy tắc $z_i\to x_i$ biến đổi như sau:

$<q_1, u_2u_3\ldots u_n, \%v_1^Rz_1>\vdash <q_1, u_3\ldots u_n, \%v_1^Rx_1^R>=<q_1, u_3\ldots u_n, \%v_2^Rz_2u_2^R>\vdash$

$\vdash <q_1, u_3\ldots u_n, \%v_2^Rz_2>\vdash \ldots \vdash <q_1, u_n, \%v_{n-1}^Rz_{n-1}>.$

Trong D, sử dụng quy tắc $z_{n-1}\to x_{n-1}\in P$ với $x_{n-1}v_{n-1}=u_n$, ta có:

$<q_1, u_n, \%v_{n-1}^Rz_{n-1}>\vdash <q_1, u_n, \%v_{n-1}^Rx_{n-1}^R>\vdash <q_1, \varepsilon, \%>\vdash <q_2, \varepsilon, \%>.$

Từ đó ta có dãy suy dẫn các hình trạng trong M: $<q_0, w, z_0>\vdash <q_2, \varepsilon, \%>$ mà $q_2\in F$ nên M đoán nhận được xâu w theo tập trạng thái kết thúc.

Thí dụ 7: Cho văn phạm phi ngữ cảnh

$G=<\{a, b\}, \{S, A\}, S, \{S\to a, S\to bSA, A\to b, S\to bA, A\to aS\}>.$

Theo chứng minh của Định lý 3.2.7, ta có thể xây dựng ôtômat đẩy xuống đoán nhận L(G) như sau:

$M = <\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, S, A, \%\}, \delta, q_0, S, \{q_2\}>,$

trong đó $\delta(q_1, \varepsilon, S)=\{<q_1, a>, <q_1, Asb>, <q_1, Ab>\}$, $\delta(q_1, \varepsilon, A)=\{<q_1, b>, <q_1, Sa>\}$,

$\delta(q_1, a, a)=\{<q_1, \varepsilon>\}$, $\delta(q_1, b, b)=\{<q_1, \varepsilon>\}$, $\delta(q_1, \varepsilon, \%)=\{<q_2, \%>\}$,

$\delta(q_0, \varepsilon, S)=\{<q_1, \%S>\}$.

3.2.8. Định lý: Cho ôtômat đẩy xuống M. Khi đó tồn tại ôtômat đẩy xuống M' sao cho N(M')=T(M).

Chứng minh: Giả sử $M=<Q, \Sigma, \Gamma, \delta, q_0, z_0, F>$ là ôtômat đẩy xuống nào đó. Ta xây dựng ôtômat đẩy xuống $M'=<Q', \Sigma', \Gamma', \delta', q'_0, z'_0, F'>$ sao cho N(M')=T(M).

Muốn vậy ta đưa thêm vào ký hiệu trạng thái mới $q_1, q_2\notin Q$ và ký hiệu ngăn xếp mới $\%\notin\Gamma$ và đặt:

$\Sigma'=\Sigma, Q'=Q\cup\{q_1, q_2\}, \Gamma'=\Gamma\cup\{\%\}, q'_0=q_1, z'_0=\%, F'=\varnothing,$

$\delta': Q' \times (\Sigma\cup\{\varepsilon\}) \times \Gamma' \longrightarrow \mathcal{P}(Q' \times \Gamma'^*)$ được định nghĩa như sau:

1) $\delta'(q_1, \varepsilon, \%)=\{<q_0, \%z_0>\}$,

2) $\delta'(q, x, z)=\delta(q, x, z)$ với $x\in\Sigma, q\in Q, z\in\Gamma$,

3) $\delta'(q, \varepsilon, z)=\delta(q, \varepsilon, z)$ nếu $q\in Q\setminus F, z\in\Gamma$ và $\delta'(q, \varepsilon, z)=\delta(q, \varepsilon, z)\cup\{<q_2, \varepsilon>\}$ nếu $q\in F, z\in\Gamma$,

4) $\delta'(q, \varepsilon, \%)=\{<q_2, \varepsilon>\}$ với $q\in F$,

5) $\delta'(q_2, \varepsilon, z)=\{<q_2, \varepsilon>\}$ với $z\in\Gamma\cup\{\%\}$.

Bây giờ ta chỉ ra N(M')=T(M).

Giả sử $w\in N(M')$. Khi đó theo cách làm việc của M' thì ta có một dãy các hình trạng sau:

$<q'_0, w, z'_0>=<q_1, w, \%>\vdash K_1\vdash K_2\vdash \ldots\vdash K_t=<q, \varepsilon, \varepsilon>,$

với $t\geq 2$, $K_i=<q_i, w_i, z_i>$, $w_i\in\Sigma^*$, $q_i\in Q'$, $z_i\in\Gamma'^*$ (i=1, 2, …, t).

Theo cách xây dựng của M' thì

$K_1=<q_0, w, \%z_0>$, $K_t=<q_2, \varepsilon, \varepsilon>$ và $w\in N(M')$.

Từ đó $\exists i<t$ sao cho $K_i=<q_i, \varepsilon, \%z'_i> \vdash <q_2, \varepsilon, \%z'_{i+1}>$, ở đây $q_i \in F$, $z'_i, z'_{i+1} \in \Gamma^*$ và $K_j=<q_j, w_j, \%z'_j> \vdash <q_{j+1}, w_{j+1}, \%z'_{j+1}>$, ở đây $w_j \in \Sigma^*$, $q_j \in Q$, $z'_j \in \Gamma^*$ $(1 \le j \le i)$. Cũng như $K_j=<q_2, \varepsilon, \%z'_i>$, ở đây $z'_j \in \Gamma^*$ $(1<j<t)$. Từ đó $<q_0, w, z_0> \vdash <q_i, \varepsilon, \%z_i>$ và do $q_i \in F$ suy ra $w \in T(M)$.

Tóm lại ta có bao hàm thức $N(M') \subset T(M)$. Bao hàm thức ngược lại $T(M) \subset N(M')$ được suy trực tiếp từ cách xây dựng M' từ M.

3.2.9. Định lý: Cho M là một ôtômat đẩy xuống. Khi đó tồn tại một văn phạm phi ngữ cảnh G sao cho $L(G)=N(M)$.

Chứng minh: Giả sử $M=<Q, \Sigma, \Gamma, \delta, q_0, z_0, F>$ là một ôtômat đẩy xuống. Theo Định lý 3.2.8, ta cần chỉ ra có văn phạm phi ngữ cảnh $G=<\Sigma, \Delta, S, P>$ sao cho $L(G)=N(M)$. Không mất tính chất tổng quát, giả sử $\varepsilon \notin N(M)$. Ta xây dựng văn phạm G ở trên như sau:

– $\Delta=\{S\} \cup \{[q_1, z, q_2] \mid q_1, q_2 \in Q, z \in \Gamma \cup \Sigma\}$,

– $P=P_1 \cup P_2 \cup P_3$, ở đây $P_1=\{S \to [q_0, z, q] \mid q \in Q, z \in \Gamma\}$,

$P_2=\{[t_1, z, t_2] \to x[t_3, z_n, q_{n-1}][q_{n-1}, z_{n-1}, q_{n-2}]\ldots[q_2, z_2, q_1][q_1, z_1, t_2] \mid n \ge 1, q_i \in Q$ $(1 \le i \le n)$, $t_1, t_2, t_3 \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $<t_3, z_1 z_2 \ldots z_n> \in \delta(t_1, x, z)\}$,

$P_3=\{[t_1, z, t_2] \to x \mid <t_2, \varepsilon> \in \delta(t_1, x, z)$ với $z \in \Gamma$, $t_1, t_2 \in Q$, $x \in \Sigma \cup \{\varepsilon\}\}$.

Người ta chỉ ra được với G định nghĩa như trên là văn phạm phi ngữ cảnh mà $L(G)=N(M)$.

Lưu ý rằng gọi $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ lần lượt là lớp các ngôn ngữ phi ngữ cảnh, lớp các ngôn ngữ được đoán nhận bởi ôtômat đẩy xuống theo tập trạng thái kết thúc, lớp các ngôn ngữ được đoán nhận bởi ôtômat đẩy xuống theo ngăn xếp rỗng, ta có:

$\mathcal{P}_1 \subset \mathcal{P}_2$ (theo Định lý 3.2.7),

$\mathcal{P}_2 \subset \mathcal{P}_3$ (theo Định lý 3.2.8),

$\mathcal{P}_3 \subset \mathcal{P}_1$ (theo Định lý 3.2.9).

Vì vậy, $\mathcal{P}_1=\mathcal{P}_2=\mathcal{P}_3$.

BÀI TẬP CHƯƠNG III:

1. Cho văn phạm phi ngữ cảnh:

$G=<\{x, +, *, (,)\}, \{S, A, B\}, S, \{S\rightarrow A \mid S+A, A\rightarrow A*B \mid B, B\rightarrow x \mid (S)\}>$

và $\omega=(x+x*x)*(x+x*x*x)$. Hãy tìm một suy dẫn từ S của ω và vẽ cây suy dẫn có kết quả là ω.

2. Chứng tỏ các văn phạm phi ngữ cảnh sau là nhập nhằng:

a) $G = <\{a, b\}, \{S, A\}, S, \{S\rightarrow A, A\rightarrow AbA, A\rightarrow a\}>$.

b) $G = <\{a, b\}, \{S, A, B\}, S, \{S\rightarrow A, A\rightarrow Bb, A\rightarrow Ab, B\rightarrow Bb, A\rightarrow a, B\rightarrow b\}>$.

c) $G = <\{a, b, c, +, *\}, \{S, A\}, S, \{S\rightarrow S+S \mid A*A \mid a \mid b \mid c, A\rightarrow S+S \mid a \mid b \mid c\}>$.

3. Hãy chứng minh các ngôn ngữ sau đây không phải là phi ngữ cảnh:

a) $L = \{a^{2n} \mid n\geq 0\}$.

b) $L = \{a^p \mid p$ là một số nguyên tố$\}$.

4. Hãy cho ôtômat đẩy xuống đoán nhận các ngôn ngữ sau:

a) $L = \{a^n b^{2n} \mid n\geq 0\}$.

b) $L = \{a^{2n} c b^n \mid n\geq 0\}$.

c) $L = \{\omega\in\{0, 1\}^* \mid n_0(\omega)>n_1(\omega)\}$.

5. Vẽ đồ thị chuyển của ôtômat đẩy xuống được cho dưới đây và xác định ngôn ngữ được đoán nhận bởi nó.

$$M = <\{q_0, q_1\}, \{a, b, c\}, \{z_0, z_1, z_2\}, \delta, q_0, z_0, \varnothing\},$$

trong đó $\delta(q_0, a, z_0)=\{<q_0, z_0z_1>\}$, $\delta(q_0, a, z_1)=\{<q_0, z_1z_1>\}$,

$\delta(q_0, a, z_2)=\{<q_0, z_0z_1>\}$, $\delta(q_0, b, z_0)=\{<q_0, z_0z_1>\}$,

$\delta(q_0, b, z_1)=\{<q_0, z_1z_2>\}$, $\delta(q_0, b, z_2)=\{<q_0, z_2z_2>\}$,

$\delta(q_0, c, z_0)=\{<q_0, z_0>\}$, $\delta(q_0, c, z_1)=\{<q_1, z_1>\}$,

$\delta(q_0, c, z_2)=\{<q_1, z_2>\}$, $\delta(q_1, b, z_2)=\{<q_1, \varepsilon>\}$,

$\delta(q_1, a, z_1)=\{<q_1, \varepsilon>\}$, $\delta(q_1, \varepsilon, z_0)=\{<q_1, \varepsilon>\}$.

6. Hãy xây dựng các ôtômat đẩy xuống đoán nhận các ngôn ngữ phi ngữ cảnh được sinh bởi các văn phạm sau:

a) $G = <\{a, b\}, \{S\}, S, \{S\rightarrow aSb, S\rightarrow ab\}>$.

b) $G = <\{a, b\}, \{S, A, B\}, S, \{S\rightarrow AB, A\rightarrow aAa, B\rightarrow bBb, A\rightarrow a, B\rightarrow b\}>$.

c) $G = <\{a, b, c\}, \{S, A, B, C\}, S, \{S\rightarrow ABC, A\rightarrow aA \mid a, B\rightarrow bB \mid b, C\rightarrow cC \mid c\}>$.

d) $G = <\{a, b, c\}, \{S, A, B\}, S, \{S\rightarrow AB, A\rightarrow aAb, B\rightarrow Bc, A\rightarrow ab, B\rightarrow c\}>$.

CHƯƠNG IV:
MÁY TURING

Khi thiết kế và cài đặt một phần mềm tin học cho một vấn đề nào đó, ta cần phải đưa ra phương pháp giải quyết mà thực chất đó là thuật toán giải quyết vấn đề này. Rõ ràng rằng, nếu không tìm được một phương pháp giải quyết thì không thể lập trình được. Chính vì thế, thuật toán là khái niệm nền tảng của hầu hết các lĩnh vực của tin học. Ta có thể hiểu khái niệm thuật toán như sau. Nếu cho trước một bài toán thì một cách giải bài toán được phân định ra thành một số hữu hạn bước, có kết thúc cuối cùng và đạt được kết quả mong muốn gọi là thuật toán.

Về mặt lịch sử, trong những năm 30 của thế kỷ trước, khi khoa học và công nghệ phát triển, nhân loại đã nêu ra nhiều bài toán mà không tìm thấy lời giải. Có nghĩa là không tìm được thuật toán để giải chúng. Người ta đã phải tìm cách định nghĩa chính xác khái niệm thuật toán. Năm 1936 A. Turing đã đưa ra một công cụ rất tốt để mô tả thuật toán mà ngày nay người ta gọi là máy Turing. Để ghi nhớ công lao này, Hội Tin học Mỹ (ACM) đã đặt ra giải thưởng Turing trong tin học. Cho đến nay, giải thưởng Turing là giải thưởng tin học lớn nhất thế giới. Tiếp theo Turing, một số nhà khoa học khác đã đưa ra các công cụ chính xác hoá khái niệm thuật toán. Đó là các khái niệm hàm đệ quy, thuật toán Marcop, văn phạm sinh của N. Chomsky. Những khái niệm này là cơ sở phát triển của việc nghiên cứu và ứng dụng thuật toán. Mặt khác chính nhờ các khái niệm này, người ta cũng xác định được những bài toán không thể giải được bằng thuật toán.

A. Turing đã đề xuất khái niệm máy Turing nhằm chính xác hoá khái niệm thuật toán. Thực tế đã chứng tỏ rằng máy Turing là một công cụ rất tốt để mô tả thuật toán. Trải qua nhiều thập niên, lý thuyết về máy Turing đã phát triển không ngừng bởi sự đóng góp công sức của nhiều nhà khoa học, trong đó có những công trình nền tảng của Hartmanis, Lewis, Stearns, Minsky, Blum, Hopcroft, Ullman.

Thực chất, máy Turing là một mô hình máy. Nó phân rã toàn bộ quá trình hoạt động ra thành các bước thao tác rất đơn giản. Bản thân máy Turing là một mô hình khái quát và đơn giản có thể mô hình hoá một quá trình tính toán bất kỳ.

Máy Turing có thể xem là một máy với bộ nhớ ngoài có dung lượng được xem như vô hạn. Trong bộ nhớ ngoài, các giá trị được bố trí sao cho có thể truy cập, đọc và sửa đổi được.

Ta có thể xem máy Turing như là một máy đoán nhận một ngôn ngữ gọi là ngôn ngữ đếm được đệ quy. Đồng thời được sử dụng để mô tả một lớp hàm quan trọng, gọi là các hàm có thể tính được. Chương này cũng mô tả một máy Turing

phổ dụng mà nó có thể bắt chước hoạt động của tất cả các máy Turing khác. Từ đó ta đi đến khái niệm bài toán không giải được bằng thuật toán.

4.1. MÁY TURING VÀ LỚP CÁC HÀM CÓ THỂ TÍNH ĐƯỢC.

4.1.1. Định nghĩa: Máy Turing đơn định là một bộ bảy

$$M = <Q, \Sigma, \Delta, \delta, s_0, B, F>,$$

trong đó,

– Q là tập hữu hạn khác rỗng, gọi là tập các trạng thái;

– Σ là một bảng chữ, gọi là bảng chữ vào hay bảng chữ trong;

– Δ là một bảng chữ, $\Delta \supset \Sigma$, gọi là bảng chữ ngoài hay tập các ký hiệu có thể ghi được lên băng;

– $\delta: D \longrightarrow Q \times \Delta \times \{R, L\}$, với $D \subset Q \times \Delta$ và $R, L \notin Q \times \Delta$, gọi là ánh xạ chuyển;

– $s_0 \in Q$, gọi là trạng thái đầu;

– $B \in \Delta \setminus \Sigma$, gọi là ký hiệu trắng;

– $F \subset Q$, gọi là tập các trạng thái kết thúc.

Trong trường hợp miền giá trị của δ là $\mathcal{P}(Q \times \Delta \times \{R, L\})$ thì máy Turing được gọi là không đơn định và lớp các ngôn ngữ được đoán nhận bởi máy Turing đơn định và không đơn định sẽ trùng nhau. Ngoài ra, có nhiều dạng máy Turing; chẳng hạn, máy Turing với băng vô hạn một đầu hoặc băng vô hạn hai đầu. Ở đây, ta chỉ xét lớp các máy Turing đơn định với băng vô hạn hai đầu.

4.1.2. Định nghĩa: Cho máy Turing $M = <Q, \Sigma, \Delta, \delta, s_0, B, F>$. Bộ ba $<\varphi, s, a\psi>$, trong đó $\varphi, \psi \in \Delta^*$, $s \in Q$, $a \in \Delta$, φ không được bắt đầu và ψ không được kết thúc bởi B, được gọi là một hình trạng của M. φaψ được gọi là từ ứng với hình trạng đã cho.

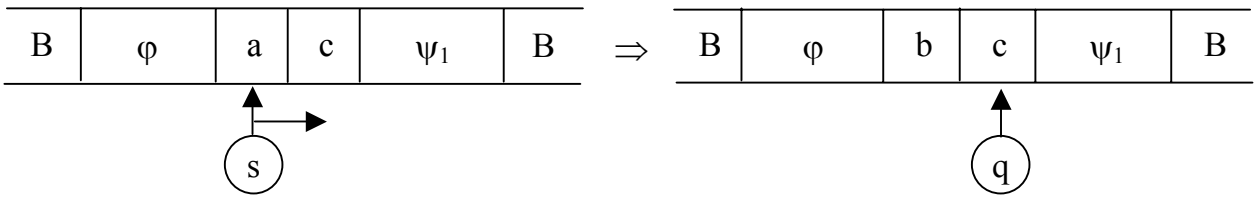
Bộ ba $<\varepsilon, s_0, a\psi>$, trong đó $a \in \Delta$, $\psi \in \Delta^*$, được gọi là hình trạng đầu (có từ ứng với nó là aψ).

4.1.3. Định nghĩa: Cho máy Turing $M = <Q, \Sigma, \Delta, \delta, s_0, B, F>$. Ta nói hình trạng $\alpha = <\varphi, s, a\psi>$ chuyển đến hình trạng β của M, ký hiệu $\alpha \vdash_M \beta$ hay đơn giản là $\alpha \vdash \beta$, nếu thoả mãn một trong các điều sau:

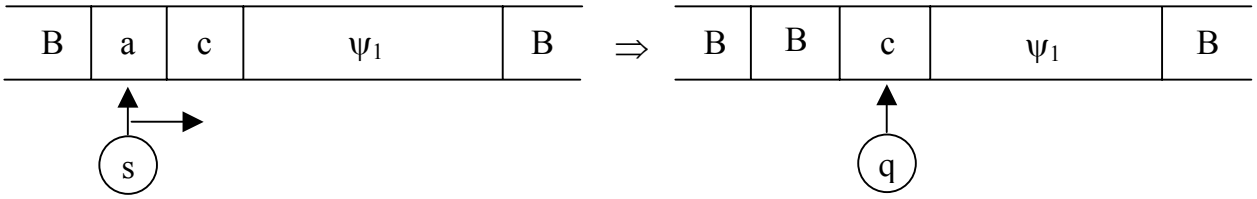
1) $\delta(s, a) = <q, b, R>$:

a) $\psi = c\psi_1 \neq \varepsilon$, $c \in \Delta$, $\psi_1 \in \Delta^*$:

+ $\alpha \vdash <\varphi b, q, c\psi_1>$, nếu $\varphi b \notin \{B\}^*$,

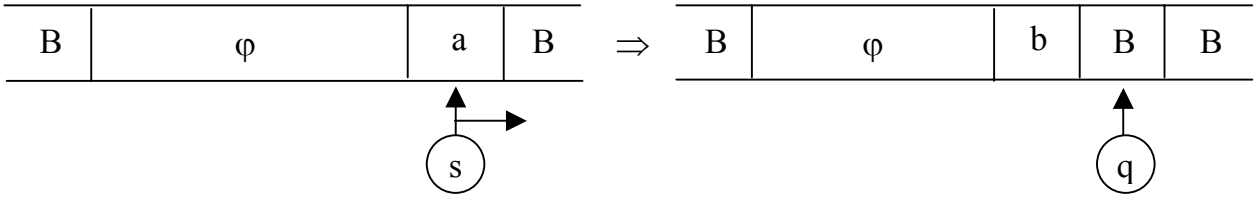


+ $\alpha \vdash <\varepsilon, q, c\psi_1>$, nếu $\varphi b \in \{B\}^*$,

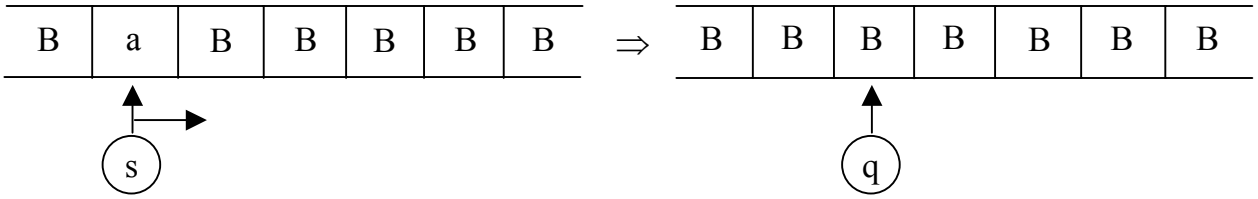


b) ψ=ε:

+ α=<φ, s, a> ⊢ <φb, q, B>, nếu φb ∉ {B}*,



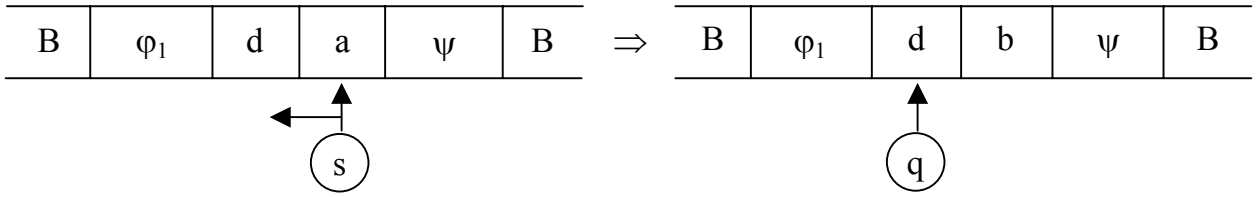
+ α=<φ, s, a> ⊢ <ε, q, B>, nếu φb ∈ {B}*,



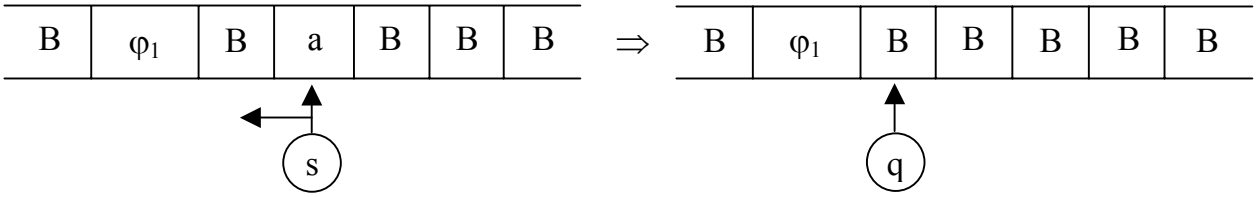
2) δ(s, a)=<q, b, L>:

a) φ=φ1d≠ε, d∈Δ, φ1∈Δ*:

+ α ⊢ <φ1, q, dbψ>, nếu dbψ ∉ {B}*,

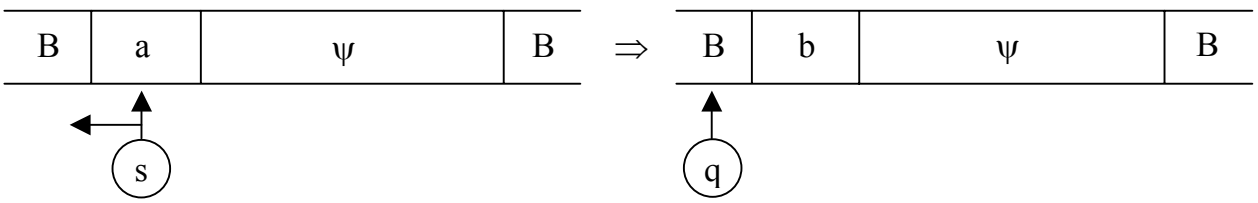


+ α ⊢ <φ1, q, B>, nếu dbψ ∈ {B}*,

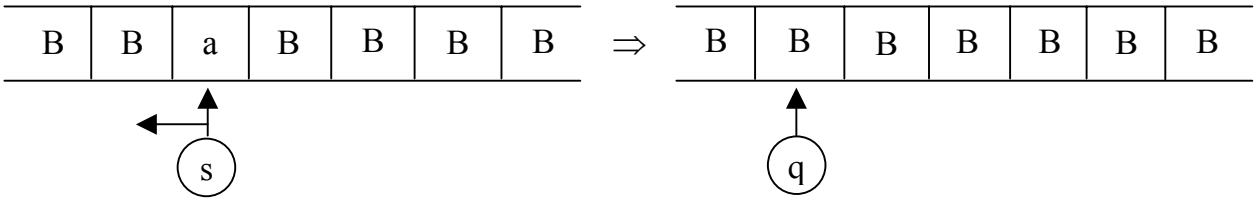


b) φ=ε:

+ α=<ε, s, aψ> ⊢ <ε, q, Bbψ>, nếu Bbψ ∉ {B}*,



+ $\alpha=<\varepsilon, s, a\psi> \vdash <\varepsilon, q, B>$, nếu $Bb\psi \in \{B\}^*$,



Dãy hình trạng α_i $(1 \le i \le n)$ của máy Turing M sao cho $\alpha_i \vdash \alpha_{i+1}$ $(1 \le i \le n-1)$ được gọi là quá trình tính toán trong M, ký hiệu $\alpha_1 \overset{M}{\models} \alpha_n$ hay đơn giản là $\alpha_1 \models \alpha_n$.

Các hình trạng không thể chuyển đến hình trạng mới được gọi là hình trạng cuối. Quá trình tính toán được bắt đầu bởi hình trạng đầu và kết thúc bởi hình trạng cuối được gọi là một quá trình tính toán hoàn chỉnh.

4.1.4. Định nghĩa: Cho máy Turing $M = <Q, \Sigma, \Delta, \delta, s_0, B, F>$ và $\omega \in \Sigma^*$. Ta nói M đoán nhận ω nếu tồn tại quá trình tính toán hoàn chỉnh $<\varepsilon, s_0, \omega> \models <\varphi, q, a\psi>$ với $q \in F$. Tập hợp các từ được đoán nhận bởi máy Turing M được gọi là ngôn ngữ được đoán nhận bởi M, ký hiệu T(M).

Ngôn ngữ được đoán nhận bởi máy Turing còn được gọi là ngôn ngữ đệ quy đếm được (Recursively Enumerable). Ngôn ngữ được đoán nhận bởi máy Turing mà nó sẽ dừng sau một số hữu hạn bước đối với mọi từ vào được gọi là ngôn ngữ đệ quy. Từ định nghĩa suy ra rằng mọi ngôn ngữ đệ quy đều là ngôn ngữ đệ quy đếm được.

4.1.5. Chú ý: Sự hoạt động của máy Turing được thể hiện ở ánh xạ chuyển. Ánh xạ này có thể được mô tả bằng bảng hoặc đồ thị chuyển.

Bảng gồm các cột được đánh dấu bằng các ký hiệu của Δ và các dòng được đánh dấu bằng các trạng thái. Nếu $\delta(s, a)=<q, b, C>$, với $a, b \in \Delta$, $s, q \in Q$, $C \in \{R, L\}$ thì bộ ba <b, C, q> được ghi vào ô ứng với dòng s cột a.

Đồ thị chuyển là một đa đồ thị có hướng, có khuyên G với tập đỉnh của G là Q. Với $a, b \in \Delta$, $s, q \in Q$, $C \in \{R, L\}$, nếu $\delta(s, a)=<q, b, c>$ thì có một cung từ s đến q với nhãn là <a/b, C>.

Thí dụ 1: Cho máy Turing:

$$M = <\{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}, \{0, 1\}, \{B, 0, 1, X\}, \delta, s_0, B, \{s_0\}>,$$

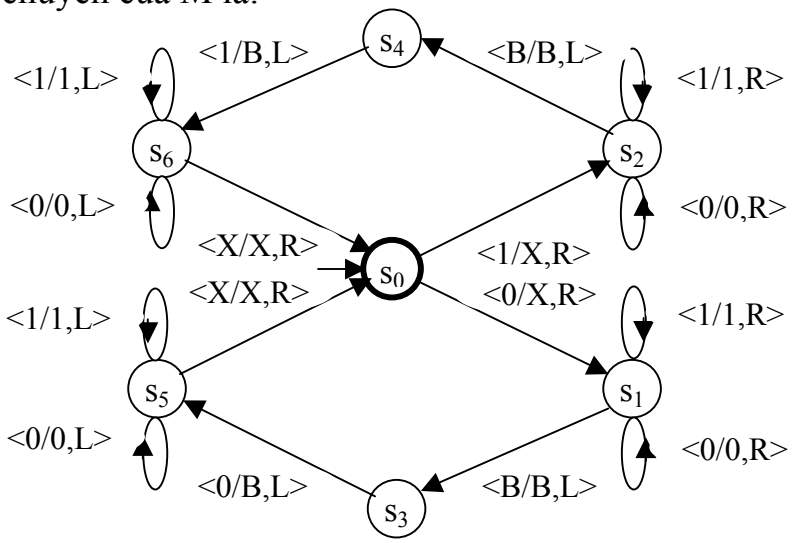
trong đó

$\delta(s_0, 0)=<s_1, X, R>$, $\delta(s_0, 1)=<s_2, X, R>$, $\delta(s_1, 0)=<s_1, 0, R>$,
$\delta(s_1, 1)=<s_1, 1, R>$, $\delta(s_1, B)=<s_3, B, L>$, $\delta(s_2, 0)=<s_2, 0, R>$,
$\delta(s_2, 1)=<s_2, 1, R>$, $\delta(s_2, B)=<s_4, B, L>$, $\delta(s_3, 0)=<s_5, B, L>$,
$\delta(s_4, 1)=<s_6, B, L>$, $\delta(s_5, 0)=<s_5, 0, L>$, $\delta(s_5, 1)=<s_5, 1, L>$,
$\delta(s_5, X)=<s_0, X, R>$, $\delta(s_6, 0)=<s_6, 0, L>$, $\delta(s_6, 1)=<s_6, 1, L>$,
$\delta(s_6, X)=<s_0, X, R>$.

Ánh xạ chuyển có thể cho bằng bảng sau:

	B	0	1	X
S_0		<X, R, s_1>	<X, R, s_2>	
S_1	<B, L, s_3>	<0, R, s_1>	<1, R, s_1>	
S_2	<B, L, s_4>	<0, R, s_2>	<1, R, s_2>	
S_3		<B, L, s_5>		
S_4			<B, L, s_6>	
S_5		<0, L, s_5>	<1, L, s_5>	<X, R, s_0>
S_6		<0, L, s_6>	<1, L, s_6>	<X, R, s_0>

Đồ thị chuyển của M là:



Ta hãy xem máy Turing M hoạt động như thế nào đối với các từ 001 và 1001.

Đối với từ 001, ta có dãy hình trạng:

<ε, s_0, 001>⊢<X, s_1, 01>⊢<X0, s_1, 1>⊢<X01, s_1, B>⊢<X0, s_3, 1>.

Rõ ràng <X0, s_3, 1> hình trạng cuối, nhưng s_3 không phải là trạng thái kết thúc, do đó M không đoán nhận từ 001.

Đối với từ 1001, ta có dãy hình trạng:

<ε, s_0, 1001>⊢<X, s_2, 001>⊢<X0, s_2, 01>⊢<X00, s_2, 1>⊢<X001, s_2, B>⊢
⊢<X00, s_4, 1>⊢<X0, s_6, 0>⊢<X, s_6, 00>⊢<B, s_6, X00>⊢<X, s_0, 00>⊢
⊢<XX, s_1, 0>⊢<XX0, s_1, B>⊢<XX, s_3, 0>⊢<X, s_5, X>⊢<XX, s_0, B>.

<XX, s_0, B> là hình trạng cuối và s_0 là trạng thái kết thúc nên từ 1001 được đoán nhận bởi máy Turing M.

Từ đồ thị chuyển dễ dàng thấy rằng M hoạt động với xâu vào ω như sau: M đọc xâu ω từ trái sang phải. Bắt đầu từ trạng thái s_0, thay ký hiệu đã đọc bởi ký hiệu X, đồng thời nếu ký hiệu vừa đọc là 0 thì chuyển sang trạng thái s_1 và nếu

ngược lại thì chuyển sang trạng thái s_2. Tại các trạng thái s_1 hoặc s_2, máy M chuyển đầu đọc qua phải mà không thay đổi ký hiệu được đọc cho đến khi gặp ký hiệu B. Từ s_1 máy chuyển sang s_3 và từ s_2 máy chuyển sang s_4. Từ s_3 nếu gặp 0 thì xoá 0 và sang s_5, từ s_4 nếu gặp 1 thì xoá 1 và sang s_6. Ở đây, ta cần lưu ý rằng xoá 0 trong trường hợp xuất phát từ s_0, máy thay 0 bởi X và xoá 1 trong trường hợp xuất phát từ s_0, máy thay 1 bởi X. Tại các trạng thái s_5 và s_6, máy dịch chuyển qua trái mà không làm thay đổi các ký hiệu trên băng cho đến khi gặp ký hiệu X, máy quay trở lại s_0 và tiếp tục quá trình trên cho đến khi máy dừng ở các trường hợp sau:

– Máy ở trạng thái s_3 gặp 1 hoặc ở trạng thái s_4 gặp 0. Trong trường hợp này rõ ràng ω ban đầu không có dạng $\alpha\alpha^R$ và máy không đoán nhận từ này.

– Máy ở trạng thái s_0 và gặp ký hiệu B. Điều này có nghĩa là các ký hiệu 0, 1 trên băng đã được thay bằng X hoặc B. Điều này chỉ xảy ra khi xâu vào ω có dạng $\alpha\alpha^R$.

Vậy $T(M)=\{\alpha\alpha^R \mid \alpha\in\{0, 1\}^*\}$.

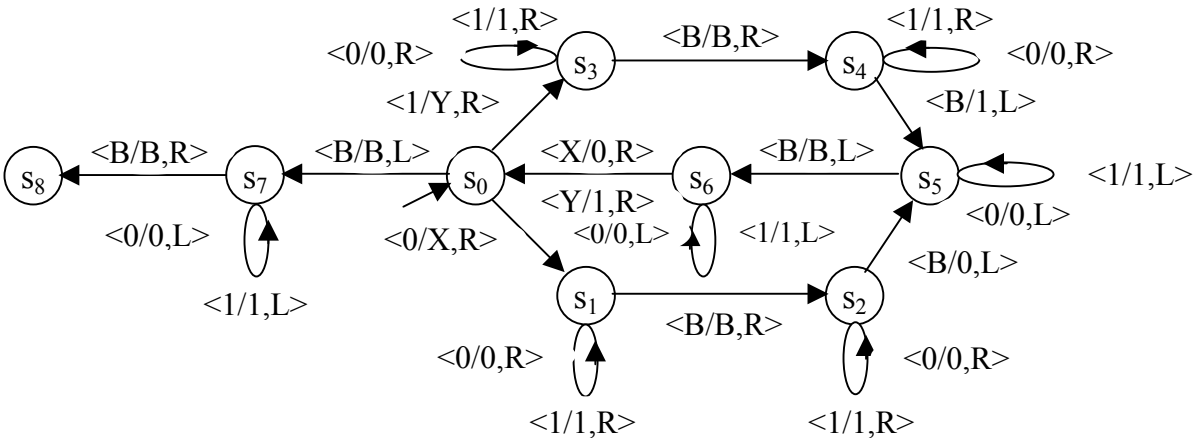
4.1.6. Định nghĩa: Cho máy Turing M = <Q, Σ, Δ, δ, s_0, B, F>. Hàm được xác định bởi máy Turing M là hàm:

$$f_M(\omega)=\begin{cases}\psi \text{ khi} < \varepsilon, s_0, a\omega' > \models < \psi', q, b\psi'' > & \text{là một quá trình tính toán hoàn chỉnh}\\ \text{ở đây } a\omega' = \omega, \psi' b \psi'' = \psi; & \\ \text{Không xác định khi không tồn tại quá trình như vậy.} & \end{cases}$$

Thí dụ 2: Cho hàm $f(\omega)=\omega B\omega$ ($\omega\in\{0, 1\}^*$). Ta xây dựng máy Turing M xác định hàm f như sau:

M = <{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8}, {0, 1}, {0, 1, X, Y, B}, δ, s_0, B, ∅>,

trong đó ánh xạ chuyển δ được chỉ ra trong đồ thị chuyển dưới đây:



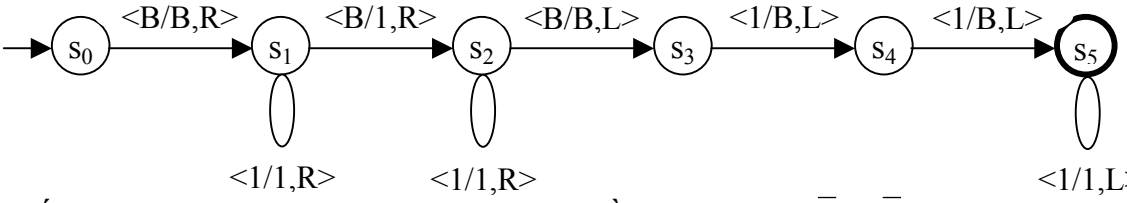
M hoạt động như sau: ký hiệu đầu tiên của ω được thay bởi X hoặc là Y tuỳ thuộc vào ký hiệu đó là 0 hay 1, sau đó đầu đọc/ghi chuyển sang phải để tìm ký hiệu B, thay ký hiệu B tiếp theo bằng 0 hoặc 1 tuỳ thuộc trước đó đã ghi x hay Y. Sau đó chạy ngược lại để tìm ký hiệu X hay Y và thay nó bởi 0 hoặc 1 tương ứng và lại chuyển sang phải. Nếu ký hiệu này là B thì tính toán kết thúc, ngược lại thì

lặp lại quá trình trên. Dễ dàng thấy rằng, sau mỗi vòng thực hiện một ký hiệu của ω được ghi sang bên phải và khi quá trình tính toán kết thúc trên băng là ωBω hay f_M=ωBω.

4.1.7. Định nghĩa: Cho hàm f: D $\longrightarrow$ **N**, với **N** là tập số tự nhiên, D⊂$\mathbf{N}^m$ và m là một số nguyên dương. Ở đây, với mỗi số tự nhiên n, ký hiệu $\overline{n}=1^{n+1}$. Ta nói hàm f có thể tính được bằng máy Turing nếu tồn tại máy Turing M xác định hàm sau:

$$h(B\overline{n}_1B\overline{n}_2\dots\overline{n}_{m-1}B\overline{n}_m)=\begin{cases}\varphi B\overline{f(n_1,n_2,...,n_m)} & \text{nếu } f(n_1, n_2, \dots, n_m) \text{ tồn tại và}\\ <\varepsilon, s_0, B\overline{n}_1B\overline{n}_2\dots\overline{n}_{m-1}B\overline{n}_m> \models <\varphi, q, B\overline{f(n_1,n_2,...,n_m)} & \\ \text{là quá trình tính toán hoàn chỉnh;} & \\ \text{Không xác định nếu } f(n_1, n_2, \dots, n_m) \text{ không tồn tại.} & \end{cases}$$

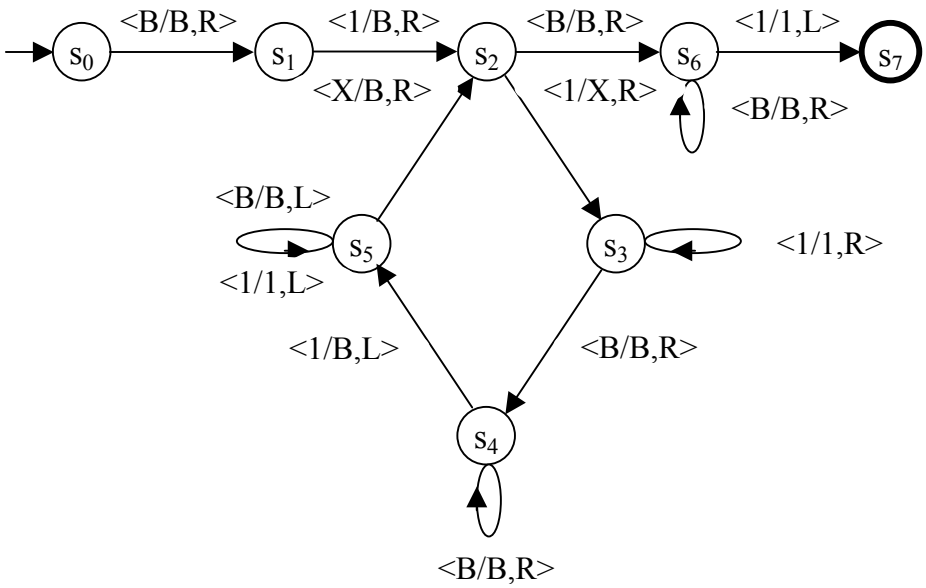
Thí dụ 3: Cho hàm $f(n_1, n_2)=n_1+n_2$. Ta xây dựng máy Turing M với đồ thị chuyển như sau:



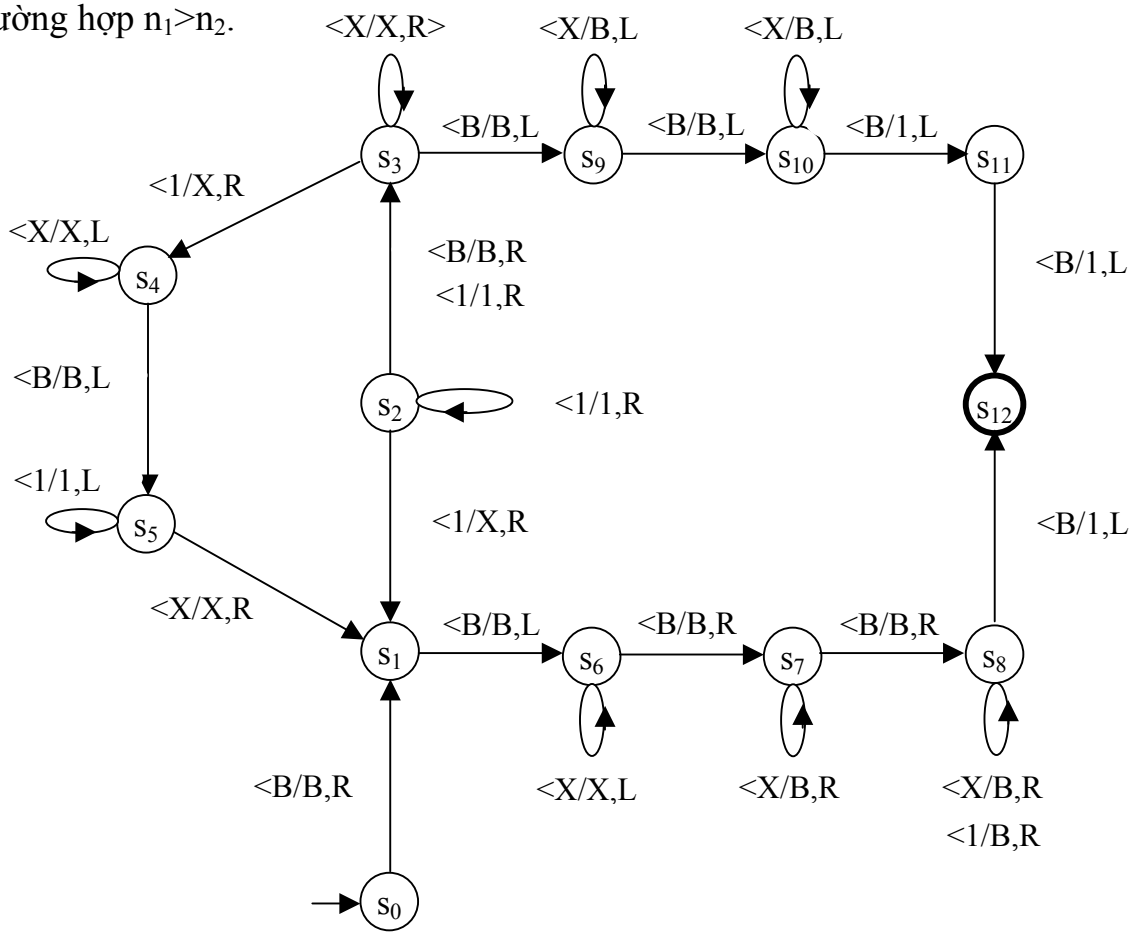
Đối với máy Turing M, với hình trạng đầu là <ε, s_0, $B\overline{n}_1B\overline{n}_2$> chỉ có các quá trình tính toán hoàn chỉnh với hình trạng kết thúc <ε, s_5, $B\overline{n_1+n_2}$>. Do đó f là hàm có thể tính được bằng máy Turing.

Thí dụ 4: Cho hàm $f(n_1, n_2)=\begin{cases}n_2-n_1 & \text{nếu } n_2\geq n_1;\\ \text{Không xác định nếu ngược lại.}\end{cases}$

Máy Turing M có đồ thị chuyển dưới đây với hình trạng đầu <ε, s_0, $B\overline{n}_1B\overline{n}_2$> và trong trường hợp xác định, hình trạng cuối là <φ, s_7, $B\overline{n_2-n_1}$> sẽ xác định hàm f.



Thí dụ 5: Cho hàm $f(n_1, n_2)=\begin{cases}0 \; khi \; n_2 \geq n_1; \\ 1 \; khi \; n_2 < n_1.\end{cases}$ Khi đó hàm f có thể tính được bằng máy Turing M có đồ thị chuyển dưới đây. Hình trạng đầu là <ε, s_0, $B\bar{n}_1B\bar{n}_2$>, hình trạng cuối là sẽ là <ε, s_{12}, B1> trong trường hợp $n_2 \geq n_1$ và <ε, s_{12}, B11> trong trường hợp $n_1 > n_2$.



Thí dụ 6: Chúng ta đưa ra một ngôn ngữ chương trình thích hợp cho việc mô tả hoạt động của máy Turing. Ngôn ngữ này dựa trên các chỉ thị cơ sở sau:

k: print A,
k: move right,
k: move left,
k: if A then go to h,
k: stop,

ở đây, k và h là các số tự nhiên ký hiệu dòng chỉ thị, còn A là một ký hiệu trên băng.

Chương trình là một dãy chỉ thị. Việc thực hiện chương trình thông thường là theo thứ tự tự nhiên của cách viết trừ khi gặp lệnh if A then go to h (nếu ký hiệu hiện hành trên băng là A thì nhảy đến thực hiện lệnh có nhãn h). Sau đây là chương trình mô tả hoạt động của một máy Turing thực hiện phép toán nhân hai:

1: print B,
2: move left,
3: if 1 then go to 2,
4: print 1,
5: move left,
6: if 1 then go to 5,
7: print 1,
8: move right,
9: if 1 then go to 1,
10: stop,

Các máy Turing và các hàm có thể tính được bằng máy Turing đóng vai trò quan trọng trong lý thuyết thuật toán. Cơ sở cho việc xây dựng một số lý thuyết thuật toán từ máy Turing là định đề Church sau đây.

Định đề Church: Lớp các hàm có thể tính được bằng thuật toán trùng với lớp các hàm có thể tính được bằng máy Turring.

4.2. MÁY TURING PHỔ DỤNG.

4.2.1. Mở đầu:

Máy Turing phổ dụng là một máy Turing có thể bắt chước sự hoạt động của bất kỳ máy Turring nào. Máy Turing phổ dụng U có thể được hiểu như sau: với một cách mã hoá thích hợp ánh xạ chuyển trạng thái của máy Turing bất kỳ và từ ω trên bảng chữ vào. Với từ đã mã hoá này, máy Turing phổ dụng U dừng khi và chỉ khi máy Turing M dừng với từ ω. Ta có thể xem máy Turing phổ dụng như là mô hình toán học của máy tính điện tử ngày nay. Các máy này thực hiện công việc bằng cách mã hoá chương trình theo ngôn ngữ bên trong được gọi là ngôn ngữ máy.

Tập các ký hiệu ghi lên băng là hữu hạn nên ta có thể ký hiệu chúng như sau: S_0=B, S_1, S_2, …, S_m và có thể mã hoá bằng bộ các chữ số 1. Chẳng hạn, B−1, S_1−11, S_2−111, …, S_m−1^{m+1}.

Tương tự, tập các trạng thái là hữu hạn và ta cũng có thể mã hoá chúng: q_0−1, q_1−11, q_2−111, …, q_n−1^{n+1}.

Cuối cùng hai ký hiệu L và R cũng có thể mã hoá: L−1, R−11.

Bây giờ để mã hoá ánh xạ chuyển trạng thái của máy Turing, ta sử dụng bảng biểu diễn ánh xạ này. Trong bảng, các cột được ký hiệu bởi các ký hiệu có thể ghi lên băng, các dòng được ký hiệu bởi các trạng thái. Tiếp theo liệt kê các phần tử của bảng theo dòng cùng với các chỉ số dòng và cột tương ứng của chúng, thứ tự là các phần tử của dòng 1, dòng 2, …từ trái sang phải. Chẳng hạn, trên một dòng xuất hiện bộ $<q_i, S_j, S_k, C, q_h>$ có nghĩa là $\delta(q_i, S_j)=<q_h, S_k, C>$. Giữa các dãy mã

của các bộ năm $<q_i, S_j, S_k, C, q_h>$ có chèn hai chữ số 0 và giữa mã các ký hiệu trong cùng một bộ năm được chèn bởi một chữ số 0. Máy Turing M được mã hoá như vậy ký hiệu là [M].

Ta chấp nhận mà không chứng minh ở đây là với máy Turing M bất kỳ tồn tại một máy Turing tương đương chỉ có một trạng thái kết thúc, vì vậy ta có thể xem q_0 là trạng thái đầu và q_1 là trạng thái kết thúc duy nhất của máy Turing M.

Thí dụ 7: Cho máy Turing M với ánh xạ chuyển được cho bởi bảng sau:

	B	S_1
q_0	$<S_1, R, q_1>$	$<S_1, R, q_0>$
q_1		$<S_1, L, q_2>$
q_2		$<S_1, L, q_2>$

Các phần tử của bảng được sắp xếp thành dãy dưới đây:

$<q_0, B, S_1, R, q_1>$, $<q_0, S_1, S_1, R, q_0>$, $<q_1, S_1, S_1, L, q_2>$, $<q_2, S_1, S_1, L, q_2>$.

Dãy này sẽ được mã hoá dưới dạng xâu nhị phân:

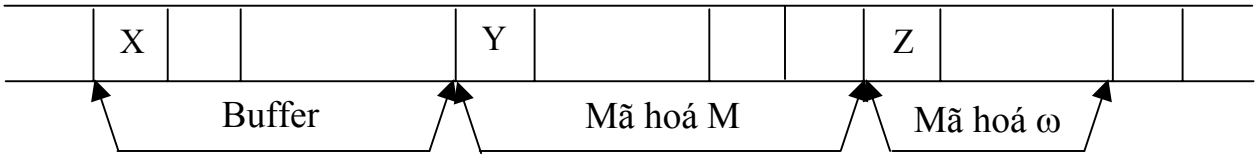
[M] = 10101101101100101101101101001101101101011100111011011010111.

Nếu trên băng vào có $\omega=S_1S_1BS_1$ thì mã tương ứng của nó là:

$$[\omega] = 1101101011.$$

4.2.2. Hoạt động của máy Turing phổ dụng:

Bây giờ giả sử máy Turing M có n trạng thái và bảng chữ ghi lên băng có m ký hiệu, thêm vào đó các ký hiệu, các trạng thái và ánh xạ chuyển của M được mã hoá như đã nói ở trên. Mô hình hoá hoạt động của máy Turing M bằng một máy Turing phổ dụng U có thể mô tả khái quát như sau: Trước hết [M] và [ω] cần phải được ghi lên băng của máy Turing phổ dụng U theo quy cách sau đây. Ký hiệu X được ghi lên băng chia băng thành hai nửa vô hạn. Nửa băng bên phải được dành ra ba đoạn kề nhau kể từ vị trí ký hiệu ngay sau X: Đoạn đầu tiên được gọi là Buffer gồm n+m+2 vị trí ký hiệu và tất cả được nhận ký hiệu 0; đoạn tiếp theo được gọi là vùng mã hoá của M, bắt đầu bởi ký hiệu Y, tiếp sau Y là [M] và được kết thúc bởi ba chữ số 0; đoạn sau cùng được gọi là đoạn mã của ω, bắt đầu bởi ký hiệu Z và tiếp theo là [ω]. Hình ảnh của băng lúc đầu là như sau:



Buffer phục vụ cho việc ghi nhận hình trạng của M trong từng bước. Ta có thể sao chép vào vùng này trạng thái bên trong và mã hoá của ký hiệu đang đọc. Ký hiệu Y thường đứng trước bộ năm xác định trạng thái hiện hành của M, ký hiệu

hiện hành trên băng, hướng chuyển động của đầu đọc trên băng. Z đánh dấu ký hiệu đang đọc trên băng của M.

Quá trình tính toán trong U mô phỏng hoạt động của máy Turing M với xâu vào ω được chia ra các pha thích hợp với việc dịch chuyển các hình trạng của M.

Một giai đoạn (pha) hoạt động của máy Turing phổ dụng U có thể tóm tắt như sau. Đầu tiên sao chép vào Buffer một khối các ký hiệu 1 nằm ngay sau Y (gọi là khối Y), sau đó ghi vào cuối khối vừa được chép một ký hiệu X, tiếp theo xoá ký hiệu Y, chạy sang phải tìm ký hiệu Z và sao chép khối ký hiệu 1 ngay sau Z (gọi là khối Z) vào Buffer ngay sau ký hiệu X rồi ghi lại ký hiệu Y trước [M]. Như vậy sau giai đoạn này trong Bufffer chứa mã của trạng thái và ký hiệu hiện hành của máy Turing M. Bước tiếp theo, máy Turing phổ dụng U so sánh hai khối ký hiệu 1 liên tiếp nhau sau Y với nội dung Buffer. Nếu trùng nhau thì tìm được bộ năm cần tìm. Nếu ngược lại thì tìm đến mã hoá của bộ năm tiếp theo sau Y và lại tiếp tục so sánh. Trong trường hợp giữa các bộ năm mô tả M không tìm thấy bộ nào thích hợp thì U dừng. Ngược lại, nếu tìm được bộ năm cần tìm thì xoá nội dung buffer rồi chuyển Y đến trước phần tử thứ ba trong bộ năm đó. Đổi nội dung của khối sau Z bởi nội dung của khối sau Y và chuyển Y đến trước phần tử thứ tư của bộ năm. Sau khi đã đọc xong phần tử thứ tư mà nó xác định hướng chuyển động của đầu đọc/ghi của M và U chuyển ký hiệu Y đến sau phần tử trước phần tử thứ năm. Tuỳ thuộc vào nội dung của khối thứ tư (một ký hiệu 1 hay hai ký hiệu 1) U chuyển Z qua phải hay qua trái một khối. Nếu Z lúc đầu nằm ở tận cùng trái của băng ghi và M cần dịch chuyển sang phải thì U đẩy mã của từ sang phải và ghi mã hoá của ký hiệu trắng vào sau Z. Nếu Z nằm tận cùng phải của băng và cần chuyển sang phải, khi đó U ghi mã của ký hiệu trắng vào cuối từ. Khi hoàn thành các công việc trên khối ký hiệu 1 đứng sau Y, ký hiệu trạng thái hiện hành của M, còn khối sau Z xác định ký hiệu M cần đọc tiếp theo. Như vậy, giai đoạn tiếp theo của việc mô phỏng bước tiếp theo của M có thể bắt đầu.

Các giai đoạn hoạt động của máy Turing phổ dụng U mô hình hoá hoạt động từng bước của máy Turing M như dã chỉ ở trên. Ngoài ra, U còn thực hiện công việc sau đây. Đầu tiên U thay tất cả các ký hiệu 0 trên ba đoạn của băng vào bằng các khoảng trắng, cuối công việc, khi M dừng máy U còn kiểm tra liệu trạng thái cuối của M có phải là trạng thái kết thúc hay không.

Các pha của một máy Turing phổ dụng U được chia thành chín phần:

– Phần 1: Thay các ký hiệu 0 bởi ký hiệu B và đầu đọc/ghi chuyển đến trước Y.

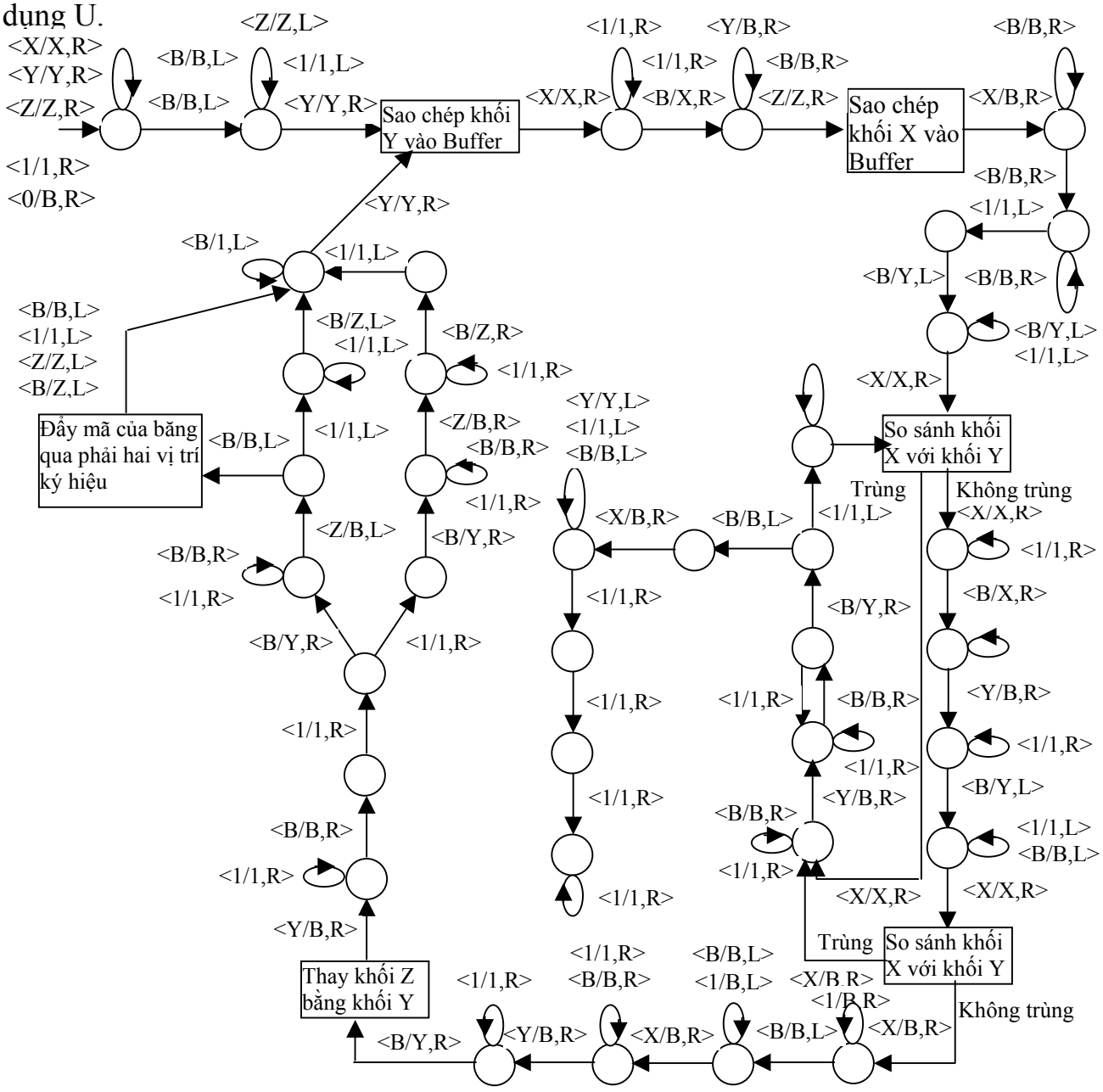
– Phần 2: Sao chép mã của trạng thái hiện hành vào buffer.

– Phần 3: Sao chép mã của ký hiệu cần đọc trên băng của M vào buffer.

– Phần 4: Đặt X và Y vào trước buffer và trước ký hiệu của [M].

– Phần 5: Tìm bộ năm có mã của trạng thái và ký hiệu trên băng trùng với buffer.
– Phần 6: Xoá buffer.
– Phần 7: Thay mã ký hiệu đã đọc bằng mã ký hiệu mới của M.
– Phần 8: Đẩy Z sang phải hay sang trái một khối mà mã ký hiệu của khối đó sẽ được đọc trong pha tiếp theo. Nếu cần thì ghi mã một khoảng trắng vào phải hoặc trái từ trên băng của M.
– Phần 9: Máy Turing phổ dụng U dừng ở trạng thái kết thúc khi và chỉ khi M dừng ở trạng thái kết thúc. Đồng thời trong vùng mã hoá của từ trên băng sẽ chứa mã của từ đáng ra còn lại trên băng của M, còn mã của trạng thái cuối của M có thể thấy trên buffer.

Dưới đây là đồ thị chuyển trạng thái phù hợp cho việc mô tả máy Turing phổ dụng U.



4.3. VẤN ĐỀ KHÔNG GIẢI ĐƯỢC BẰNG THUẬT TOÁN.

4.3.1. Mở đầu:

Trong toán học thường gặp vấn đề cần xác định liệu một phần tử của một lớp vô hạn nào đó có một số tính chất đã cho hay không. Chẳng hạn, ta có thể hỏi liệu hai số tự nhiên cho trước có nguyên tố cùng nhau hay không, hoặc là một máy Turing có dừng sau một số hữu hạn bước hay không, … Các vấn đề trên được gọi là vấn đề thuật toán và có thể quy về vấn đề là liệu một từ trên bảng chữ nào đó có thuộc vào một ngôn ngữ hình thức đã cho hay không.

Một bài toán được gọi là không giải được bằng thuật toán nếu không tồn tại một máy Turing nào sau một số hữu hạn bước xác định được liệu một sự mã hoá cụ thể nào đó của bài toán có thuộc ngôn ngữ hình thức đã cho hay không. Trong trường hợp ngược lại thì bài toán được gọi là giải được bằng thuật toán. Như vậy, một vấn đề giải được bằng thuật toán nếu và chỉ nếu ngôn ngữ hình thức mô tả nó là đệ quy.

Sau đây ta sẽ nghiên cứu một vài tính chất của ngôn ngữ đệ quy và đệ quy đếm được. Việc chứng minh các tính chất này khá phức tạp nên ta không đi sâu vào chi tiết.

4.3.2. Định lý: Phần bù của một ngôn ngữ đệ quy là ngôn ngữ đệ quy.

Chứng minh: Giả sử L là một ngôn ngữ đệ quy trên bảng chữ Σ. Điều này có nghĩa là tồn tại một máy Turing M mà với từ $\omega \in \Sigma^*$ tuỳ ý nó dừng sau một số hữu hạn bước và T(M)=L. Thay tập trạng thái kết thúc của M bằng phần bù của nó, ta được một máy Turing mới M'. Dưới tác động cùng một từ, M' dừng với trạng thái kết thúc khi và chỉ khi dưới tác động của từ đó M dừng với trạng thái không kết thúc. Điều này dẫn đến $T(M')=\overline{L}$. Do M' dừng sau một số hữu hạn bước với mọi từ vào nên $T(M')=\overline{L}$ cũng là ngôn ngữ đệ quy.

4.3.3. Định lý: Nếu L là ngôn ngữ đệ quy đếm được trên bảng chữ Σ và phần bù $\overline{L}$ của nó cũng là đệ quy đếm được thì L là đệ quy.

Chứng minh: Giả sử M_1, M_2 là các máy Turing sao cho $T(M_1)=L$ và $T(M_2)=\overline{L}$. Ta cần xây dựng máy Turing M' mà nó mô hình hoá đồng thời sự hoạt động của M_1, M_2. Điều ta cần có là dưới tác động của từ ω, máy Turing M' ngừng hoạt động khi và chỉ khi M_1 hoặc M_2 đoán nhận từ ω. Ta muốn M_1 dừng với trạng thái kết thúc, còn M_2 dừng với trạng thái không kết thúc. Việc xây dựng này là có thể thực hiện được và với mọi từ ω máy Turing M' sau một số hữu hạn bước sẽ dừng. Nếu $\omega \in \Sigma^*$ thì $\omega \in L$ hoặc $\omega \in \overline{L}$. Nếu $\omega \in \overline{L}$ thì M_2 sẽ đoán nhận ω sau một số hữu hạn bước. Từ đó nếu M_1 không dừng sau một số hữu hạn bước thì M' phải hoàn thành công việc của mình trong thời gian hữu hạn. Như vậy $T(M_1)=L$ là đệ quy.

4.3.4. Định lý: Tồn tại một ngôn ngữ đệ quy đếm được nhưng không đệ quy.

Chứng minh: Xét ngôn ngữ T(U) được đoán nhận bởi máy Turing U. Khi đó T(U) là đệ quy đếm được.

Giả sử T(U) là đệ quy. Điều này có nghĩa là tồn tại một máy Turing M (không đòi hỏi là trùng với U) sao cho T(M)=T(U) và sẽ dừng sau một số hữu hạn bước dưới tác động của mọi từ trên bảng chữ vào. Ta xây dựng máy Turing M' như sau:

Giả sử ω là một từ trên bảng chữ vào của M'. Trước hết M' cho mã [ω] của ω, đồng thời trên cơ sở sắp xếp các từ trên bảng chữ vào tìm chỉ số i của ω (số thứ tự của ω trong dãy là i). Tiếp theo ứng với các chỉ số i, máy Turing M' thiết lập sự mô tả mã của máy Turing thứ i trong dãy các máy Turing M_1, M_2, …Cuối cùng M thiết lập sự mô tả chuẩn của từ ω_i và M_i, $<M_i\omega_i>$. M' bắt chước sự hoạt động của máy Turing M với từ $<M_i\omega_i>$ mà theo giả thiết nó tồn tại, đoán nhận ngôn ngữ T(U) và dừng sau một số hữu hạn bước đối với mọi từ trên bảng chữ vào. Nếu M kết thúc sự hoạt động với trạng thái không kết thúc thì khi đó M' chuyển sang một trạng thái kết thúc riêng và dừng. Nếu M dừng ở trạng thái kết thúc thì M' bắt đầu bắt chước sự hoạt động của máy Turing mà nó không dừng với mọi từ của bảng chữ vào.

Rõ ràng rằng máy Turing M' đoán nhận từ $\omega=\omega_i$ khi và chỉ khi không đoán nhận $<M_i\omega_i>$. Ta biết rằng $T(M)=T(U)=\{<M_i\varphi> \mid \varphi\in T(M_i)\}$. Đồng thời mọi máy Turing đều có mặt trong dãy M_1, M_2, … và do đó M' cũng nằm trong dãy này, có nghĩa là tồn tại một số tự nhiên h sao cho $M'=M_h$.

Bây giờ ta xem máy Turing M_1 hoạt động như thế nào với từ ω_1. Ta nhận được $\omega_1\in T(M')$ khi và chỉ khi $\omega_1\in T(M_1)$. Điều này mâu thuẫn với giả thiết. Vậy T(U) không đệ quy.

4.3.5. Hệ quả: Tồn tại một ngôn ngữ hình thức nhưng không đệ quy đếm được.

Chứng minh: Như trong chứng minh của Định lý 4.3.4, T(U) là đệ quy đếm được nhưng không đệ quy. Do đó theo Định lý 4.3.3, phần bù của T(U) là không đệ quy đếm được.

4.3.6. Định lý: Một ngôn ngữ hình thức là loại 0 khi và chỉ khi nó là đệ quy đếm được. Điều này có nghĩa là lớp ngôn ngữ hình thức loại 0 chính là lớp ngôn ngữ đệ quy đếm được.

4.3.7. Chú ý: Nhờ vào Định lý 4.3.4, ta thấy rằng có những ngôn ngữ đệ quy đếm được nhưng không đệ quy. Với việc mã hoá thích hợp, ta chỉ ra rằng nhiều vấn đề không giải quyết được bằng thuật toán. Ta sẽ khảo sát một số vấn đề liên quan đến lớp các ngôn ngữ đệ quy đếm được. Chẳng hạn như một ngôn ngữ đệ quy đếm được có rỗng hay không, có hữu hạn hay không, có chính quy hay không, có là phi ngữ cảnh hay không và có là cảm ngữ cảnh hay không, … Tất cả các ngôn ngữ có

tính chất này hình thành lên một lớp con của lớp các ngôn ngữ đệ quy đếm được. Khi ta khảo sát một ngôn ngữ có hay không một tính chất đã cho thì thực tế ta cần giải quyết vấn đề ngôn ngữ này có thuộc hay không lớp con đã cho của lớp các ngôn ngữ đệ quy đếm được.

Ta nói rằng một tính chất của các ngôn ngữ đệ quy đếm được là tầm thường nếu hoặc mọi ngôn ngữ đệ quy đếm được đều có tính chất này hoặc ngược lại mọi ngôn ngữ đệ quy đếm được không có tính chất này. Điều này có nghĩa là lớp con các ngôn ngữ xác định bởi tính chất này hoặc bằng rỗng hoặc bằng chính lớp các ngôn ngữ đệ quy đếm được. Như vậy các tính chất: một ngôn ngữ đã cho có rỗng hay không, có hữu hạn hay không, có chính quy hay không, … là không tầm thường. Có những ngôn ngữ đệ quy đếm được có những tính chất trên và có những ngôn ngữ đệ quy đếm được không có tính chất trên.

Từ Định lý 4.3.3, ta biết rằng muốn xác định bằng thuật toán việc thực hiện một tính chất nào đó thì vấn đề này được giải quyết bằng thuật toán khi và chỉ khi việc phủ định của tính chất này cũng được giải quyết bằng thuật toán.

4.3.8. Định lý: Cho trước một tính chất không tầm thường của lớp các ngôn ngữ đệ quy đếm được. Khi đó vấn đề xác định rằng một ngôn ngữ có tính chất này hay không là không giải quyết được bằng thuật toán.

4.3.9. Hệ quả: Việc xác định rằng một ngôn ngữ đệ quy đếm được có là:

a) Rỗng,

b) Hữu hạn,

c) Chính quy,

d) Phi ngữ cảnh,

e) Cảm ngữ cảnh,

f) Đệ quy

hay không là vấn đề không giải được bằng thuật toán.

4.3.10. Định lý: Mọi ngôn ngữ cảm ngữ cảnh đều là ngôn ngữ đệ quy.

4.3.11. Định lý: Tồn tại một ngôn ngữ đệ quy mà không là ngôn ngữ cảm ngữ cảnh.

BÀI TẬP CHƯƠNG IV:

1. Hãy xây dựng một máy Turring M sao cho T(M)={$a^n b^n c^n$ | n≥1}.

2. Cho máy Turing M = <{s_0, s_1, s_2, s_3, s_4}, {0, 1}, {0, 1, X, Y, B}, δ, s_0, B, {s_4}>, trong đó:

δ(s_0, 0)=<s_1, X, R>, δ(s_0, Y)=<s_3, Y, R>, δ(s_1, 0)=<s_1, 0, R>,
δ(s_1, 1)=<s_2, Y, L>, δ(s_1, Y)=<s_1, Y, R>, δ(s_2, 0)=<s_2, 0, L>,
δ(s_2, X)=<s_0, X, R>, δ(s_2, Y)=<s_2, Y, L>, δ(s_3, Y)=<s_3, Y, R>,
δ(s_3, B)=<s_4, B, R>.

a) Hãy vẽ đồ thị chuyển của M.

b) Hãy trình bày các quá trình tính toán hoàn chỉnh của máy M xuất phát từ các hình trạng đầu sau: <ε, s_0, 000111>, <ε, s_0, 00111>, <ε, s_0, 0101>, <ε, s_0, 00011>.

c) Hãy xác định ngôn ngữ T(M).

3. Chứng tỏ rằng các hàm dưới đây có thể được xác định bằng máy Turing.

a) $f(\omega) = \underbrace{\omega B \omega B \ldots B \omega}_{n \text{ lần } \omega}, \quad \omega \in \{1\}^+$.

b) $f(\omega) = \omega B \omega^R, \omega \in \{0,1\}^+$.

c) $f(\omega B \omega \omega), \omega \in \{0,1\}^+$.

d) $f(1^n B 1^m) = 1^{nm}, n, m \geq 0$.

e) $f(1^n) = (0011)^n$.

f) $f(\varphi) = \begin{cases} 1 \; khi \; \varphi = \varphi^R, \\ 0 \; khi \; \varphi \neq \varphi^R; \end{cases} \quad \varphi \in \{0,1\}^*$.

4. Chứng tỏ rằng các hàm dưới đây có thể tính được bằng máy Turing.

a) f(n_1)=3n_1.

b) f(n_1)=n_1+3.

c) f(n_1, n_2)=$n_1 n_2$.

d) f(n_1, n_2)=|n_1−n_2|.

e) f(n_1, n_2)=$\begin{cases} n_1 - n_2 \; khi \; n_1 \geq n_2, \\ 0 \; khi \; n_1 < n_2. \end{cases}$

f) f(n_1, n_2)=$\begin{cases} 0 \; khi \; n_1 = n_2, \\ 1 \; khi \; n_1 \neq n_2. \end{cases}$

5. Cho f_1, f_2, …, f_n là các hàm m biến và g là hàm n biến. Chứng minh rằng nếu các hàm trên đều có thể tính được bằng máy Turing thì hàm

h(x_1, x_2, …x_m)=g(f_1(x_1, x_2, …x_m), f_2(x_1, x_2, …x_m), …, f_n(x_1, x_2, …x_m))

cũng có thể tính được bằng máy Turing.

CHƯƠNG V:
GIỚI THIỆU VỀ TRÌNH BIÊN DỊCH

5.1. NGÔN NGỮ LẬP TRÌNH.

5.1.1. Mở đầu:

Từ ngàn xưa con người muốn giao tiếp với nhau phải dùng ngôn ngữ. Vậy người giao tiếp với máy tính tất nhiên cũng thông qua ngôn ngữ. Con người muốn máy tính thực hiện công việc, phải viết các yêu cầu đưa cho máy bằng ngôn ngữ máy hiểu được. Việc viết các yêu cầu, ta gọi là lập trình (programming). Ngôn ngữ dùng để lập trình được gọi là ngôn ngữ lập trình (programming language).

Viết chương trình để giải quyết vấn đề sẽ dễ dàng và tự nhiên hơn nếu ngôn ngữ lập trình gần với vấn đề cần giải quyết. Có nghĩa là ngôn ngữ phải chứa đựng các cấu trúc thuật ngữ, phần tử dùng để miêu tả vấn đề và không phụ thuộc vào máy tính cụ thể.

Các ngôn ngữ lập trình có tính chất như trên được gọi là ngôn ngữ cấp cao.

Nhưng máy tính chỉ hiểu, chỉ chấp nhận ngôn ngữ cấp thấp riêng của mình, đó là chuỗi các số 0 và 1, chuỗi số đó lại không gần gũi chút nào đối với con người.

Việc phân cấp ngôn ngữ lập trình được dựa trên cơ sở của tính không phụ thuộc với máy tính ngày càng cao của các ngôn ngữ.

Phân loại: 1) Ngôn ngữ máy (machine language),
2) Hợp ngữ (assembly language),
3) Ngôn ngữ cấp cao (higher-level language).

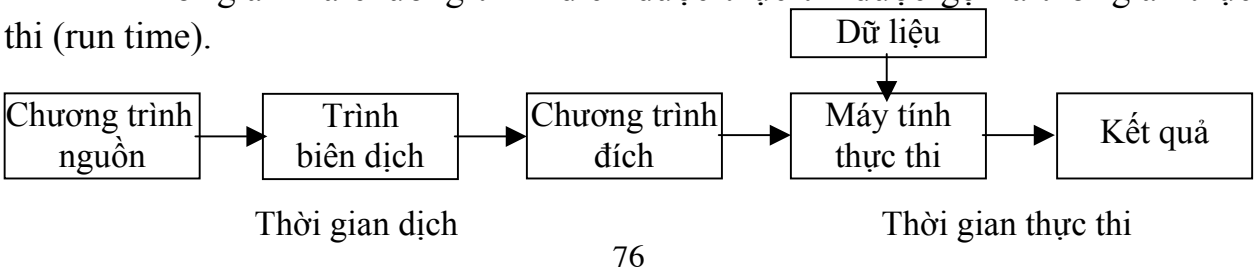
Bởi vì máy tính chỉ có thể hiểu ngôn ngữ máy cho nên một chương trình viết trong ngôn ngữ cấp cao cuối cùng rồi cũng được dịch sang ngôn ngữ máy. Công cụ thực hiện việc dịch đó được gọi là chương trình dịch (translator).

Chương trình dịch được chia làm hai loại: trình biên dịch (compiler) và trình thông dịch (interpreter).

– Trình biên dịch: chuyển một chương trình viết trong ngôn ngữ cấp cao – chương trình nguồn sang chương trình trong ngôn ngữ cấp cao khác hoặc ngôn ngữ máy – chương trình đích.

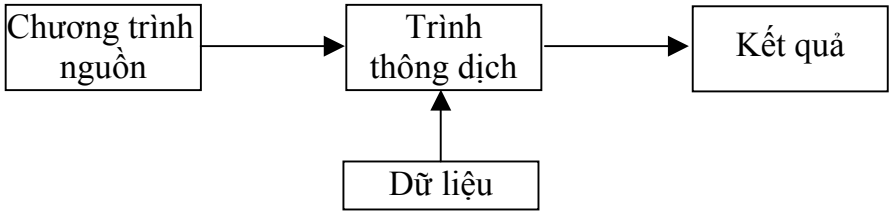
+ Thời gian chuyển một chương trình nguồn sang chương trình đích được gọi là thời gian dịch (compile time).

+ Thời gian mà chương trình đích được thực thi được gọi là thời gian thực thi (run time).



Như vậy, đối với trình biên dịch, chương trình nguồn và dữ liệu được xử lý trong thời gian khác nhau, đó là thời gian dịch và thời gian thực thi.

– Trình thông dịch: quá trình xử lý dạng bên trong của chương trình nguồn và dữ liệu cùng một thời gian.



Một số trình thông dịch làm việc như sau: phân tích từng phát biểu và thực thi luôn.

Hiện nay trình thông dịch đa phần áp dụng kỹ thuật của trình biên dịch là biên dịch chương trình nguồn sang dạng mã trung gian. Từ mã trung gian sẽ được thực thi bằng trình thông dịch.

Đặc tả ngôn ngữ lập trình: để đặc tả ngôn ngữ lập trình, tối thiểu ta cần định nghĩa:

1. Tập các ký hiệu cần dùng trong các chương trình hợp lệ.
2. Tập các chương trình hợp lệ.
3. Nghĩa của từng chương trình hợp lệ.

Việc định nghĩa tập các ký hiệu của một ngôn ngữ thật dễ dàng, ta chỉ cần liệt kê chúng. Song định nghĩa tập chương trình, gọi là hợp lệ thì quả là một công việc khó khăn hơn nhiều. Bởi vì thật là khó để xác định thế nào là một chương trình hợp lệ.

Khi đặc tả ngôn ngữ lập trình, ta thường định nghĩa lớp các chương trình bằng tập các luật văn phạm, ta có thể tạo nên cả những chương trình còn nghi vấn về tính hợp lệ.

Chẳng hạn, trong Fortran cho phép phát biểu sau: *L GOTO L* là hợp lệ. Mặc dù nếu phát biểu này mà được thực hiện thì sẽ là vòng lặp vô tận, không giải quyết được gì cả. Vì thế chương trình mà ta coi là hợp lệ phải được hiểu trong nghĩa hẹp.

Khó khăn thứ ba cũng là vấn đề khó nhất của đặc tả ngôn ngữ là việc định nghĩa ý nghĩa của một chương trình hợp lệ. Có ba phương pháp để xác định nghĩa của chương trình hợp lệ.

Phương pháp thứ nhất là định nghĩa bằng phép ánh xạ: ánh xạ mỗi chương trình vào một câu trong ngôn ngữ mà nghĩa của nó ta hiểu được.

Phương pháp thứ hai, xác định ý nghĩa của chương trình bằng một máy lý tưởng (idealized machine). Nghĩa của chương trình có thể được đặc tả trong ngôn

từ của máy lý tưởng này. Như vậy máy lý tưởng trở thành bộ thông dịch cho ngôn ngữ.

Phương pháp thứ ba, nghĩa của một chương trình nguồn chính là sản phẩm xuất ra của trình biên dịch, khi nó dịch chương trình nguồn. Trình biên dịch được dặc tả như là tập các cặp (x, y), với x là chương trình nguồn và y là chương trình đích, là chương trình mà x sẽ được dịch sang y. Giả sử cặp (x, y) đã có trước, bây giờ ta quan tâm đến cấu trúc của thiết bị để khi nhận x là đầu vào thì sinh ra y ở đầu ra.

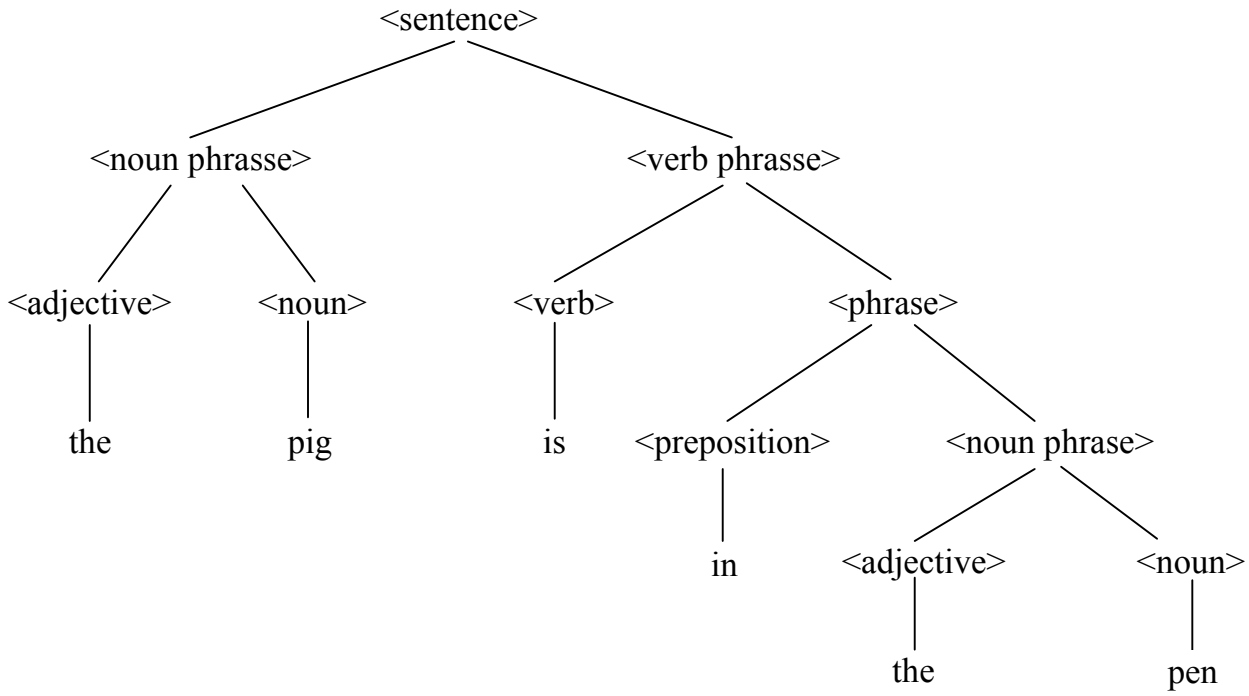
Ta coi tập (x, y) là sự biên dịch. Nếu mỗi chuỗi x được định nghĩa trên bảng chữ Σ và chuỗi y được định nghĩa trên bảng chữ Δ thì biên dịch là phép ánh xạ từ Σ^* đến Δ^*.

5.1.2. Cú pháp và ngữ nghĩa:

Để tiện lợi hơn trong đặc tả và hiện thực sự biên dịch, ta coi sự biên dịch bao gồm hai phép chiếu đơn giản hơn.

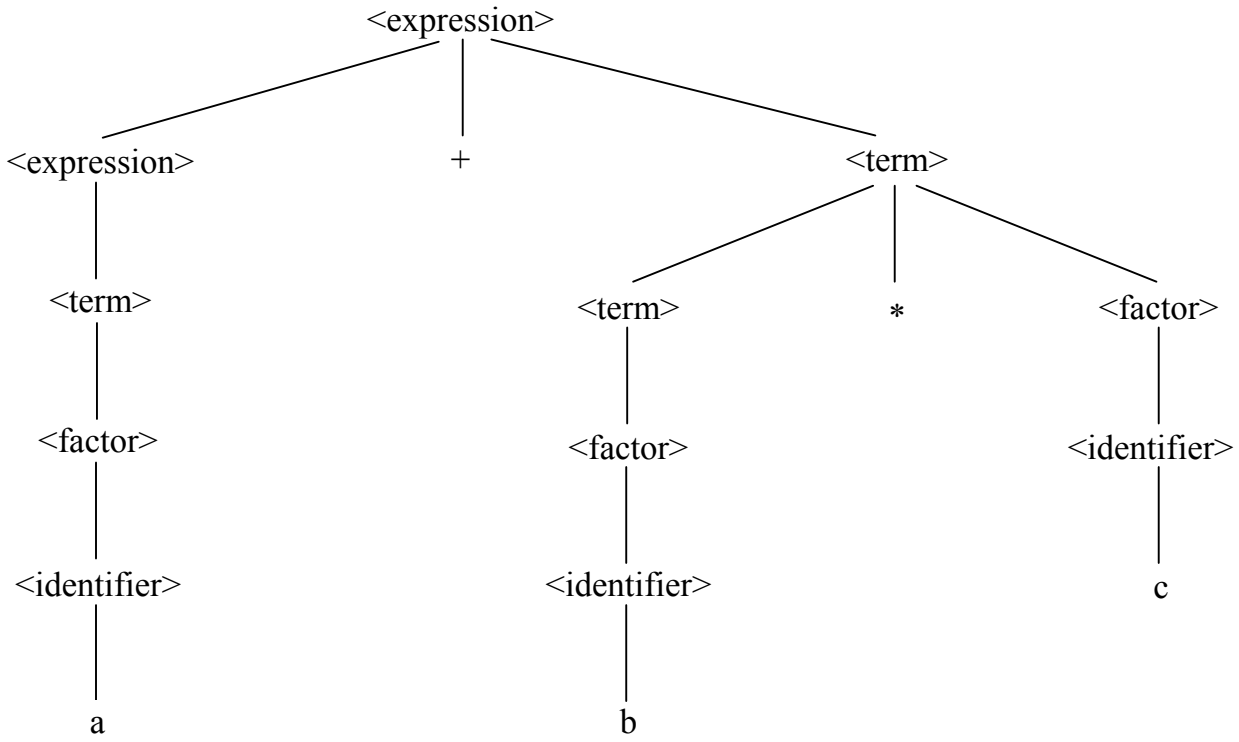
Thứ nhất là phép ánh xạ cú pháp (syntactic mapping), nó ánh xạ một chương trình viết trong ngôn ngữ nguồn sang cấu trúc là đối số của phép ánh xạ tiếp theo, đó là phép ánh xạ ngữ nghĩa (semantic mapping). Cấu trúc của phép ánh xạ cú pháp là cây cú pháp (syntactic tree). Sau đây là thí dụ cây cú pháp được xây dựng như thế nào trên chuỗi nhập vào là một câu tiếng Anh. Mỗi câu tiếng Anh được bẻ ra thành những ký hiệu cú pháp nhờ vào các luật văn phạm.

Thí dụ 1: *The pig is in the pen* có cấu trúc văn phạm được biểu diễn bằng cây cú pháp ở hình sau. Cây cú pháp với các nút có tên là ký hiệu cú pháp (ký hiệu không kết thúc) và lá có tên là ký hiệu kết thúc.



Tương tự, một chương trình được viết trong ngôn ngữ lập trình có thể bẻ nhỏ thành các phần tử cú pháp, quan hệ với nhau bởi luật cú pháp, điều khiển ngôn ngữ lập trình.

Thí dụ 2: Chuỗi ký tự a + b ∗ c có cây cú pháp sau:



Qua hai thí dụ trên ta thấy rằng với mỗi câu của ngôn ngữ đêu tồn tại cây cú háp của nó. Quá trình tìm ra cây cú pháp của một câu, được gọi là quá trình phân tích cú pháp câu đó (syntactic analysis parsing). Quá trình phân tích cú pháp được thực hiện dựa trên cơ sở các luật cú pháp của ngôn ngữ (đó chính là các quy tắc hay luật sinh). Cú pháp của ngôn ngữ là tập luật sinh, nó cung cấp cho ta mối quan hệ giữa mỗi câu của ngôn ngữ với cấu trúc cú pháp.

Thứ hai là phép chiếu ngữ nghĩa, trong đó cấu trúc cú pháp của câu trong ngôn ngữ nguồn sẽ được ánh xạ đến vùng xuất – đó là ngôn ngữ đích (cuối cùng cũng là ngôn ngữ máy). Ngữ nghĩa của ngôn ngữ là phép ánh xạ nó kết hợp cấu trúc cú pháp của mỗi câu nhập vào với chuỗi ký hiệu trong ngôn ngữ nào đó, mà ta gọi là nghĩa của câu nguyên thuỷ (câu nhập vào). Đặc tả ngữ nghĩa của ngôn ngữ là vấn đề rất khó, nó chưa được giải quyết đầy đủ, đặc biệt với ngôn ngữ tự nhiên.

Mặc dù việc đặc tả cú pháp và ngữ nghĩa của ngôn ngữ lập trình hoàn toàn không phải là công việc đơn giản, cũng như cho đến bây giờ chưa tồn tại phương pháp tổng quát để đặc tả, song đã tồn tại hai khái niệm của lý thuyết ngôn ngữ, chúng được dùng để xây dựng sự đặc tả ngôn ngữ.

Thứ nhất là khái niệm về văn phạm phi ngữ cảnh. Hầu hết các luật miêu tả cấu trúc cú pháp đều có thể hình thức hoá như là văn phạm phi ngữ cảnh. Hơn nữa van phạm phi ngữ cảnh còn cung cấp sự miêu tả được coi là một phần của đặc tả trình biên dịch.

Thứ hai là khái niệm sơ đồ dịch trực tiếp cú pháp, nó được dùng để đặc tả phép ánh xạ từ ngôn ngữ này sang ngôn ngữ khác.

5.2. TRÌNH BIÊN DỊCH.

Trình biên dịch là chương trình dùng để đọc một chương trình được viết trong một ngôn ngữ lập trình được gọi là ngôn ngữ nguồn (source language) và dịch chương trình đó sang chương trình tương ứng trong ngôn ngữ khác, ngôn ngữ đích (target language).

Như vậy ta sẽ có rất nhiều trình biên dịch, vì ta có hàng ngàn các ngôn ngữ nguồn từ những ngôn ngữ lập trình họ cổ điển (Fortran, Pascal) đến các ngôn ngữ đặc biệt đã xuất hiện rất nhiều trong lĩnh vực ứng dụng máy tính.

Ngôn ngữ đích cũng rất đa dạng vì có thể là ngôn ngữ lập trình bất kỳ hoặc ngôn ngữ máy. Trình biên dịch đầu tiên xuất hiện vào những năm 50 của thế kỷ trước. Lúc đó viết trình biên dịch quả là một công việc hết sức khó khăn. Trình biên dịch ngô ngữ Fortran đầu tiên đã phải viết trong 18 năm/công người.

5.2.1. Các phần của trình biên dịch:

Chương trình nguồn trong ngôn ngữ lập trình không gì khác là chuỗi các ký tự. Trình biên dịch có nhiệm vụ chuyển chuỗi ký tự này sang chuỗi ký tự khác - đó là mã đối tượng. Quá trình này bao gồm các quá trình nhỏ hơn và được đặt tên như sau:

1) Phân tích từ vựng.
2) Bảng danh biểu và thông báo lỗi.
3) Phân tích cú pháp.
4) Phân tích ngữ nghĩa.
5) Sinh mã trung gian.
6) Tối ưu mã trung gian.
7) Sinh mã đối tượng.

Đối với một trình biên dịch tồn tại trong thực tế, thứ tự các quá trình nhỏ có thể hơi khác so với thứ tự ở trên. Có thể một số quá trình nhỏ kết hợp lại với nhau thành một quá trình duy nhất. Trình biên dịch phải có khả năng nhận biết chuỗi nhập vào có phải là một chương trình hợp lệ cú pháp không. Nếu không, trình biên dịch phải thông báo lỗi.

5.2.2. Phân tích từ vựng (lexical analysis):

Giai đoạn phân tích từ vựng là giai đoạn đầu của quá trình biên dịch.

Dòng nhập vào trình biên dịch là chuỗi các ký tự cho phép của một ngôn ngữ lập trình, cũng là chuỗi nhập vào bộ phân tích từ vựng.

Chẳng hạn, đối với ngôn ngữ Pascal, các ký tự alphabet là các ký tự sau:

A…Z, a…z, $ @ 0 1 2…9 dấu trống – + = := / * ( ), & >=…

Trong chương trình nguồn, sự kết hợp một số ký tự alphabet sẽ tạo nên một thực thể của ngôn ngữ. Chẳng hạn, BEGIN là sự kết hợp 5 ký tự B, E, G, I, N tạo nên thực thể là từ khoá BEGIN.

Các thực thể

1. Ký hiệu trống, dấu tab, dấu xuống hàng,
2. Từ khoá: begin, end, goto, while, do, integer…,
3. Chuỗi các ký tự số tượng trưng cho hằng số,
4. Danh biểu, dùng để đặt tên cho các biến, hàm thủ tục, nhãn,
5. Các ký hiệu đặc biệt: +, –, /, *, :=, ;, =, >, >=, <, <=,

được gọi là các token.

Nhiệm vụ của bộ phân tích từ vựng là khi tiếp nhận chuỗi ký tự nhập phải biết nhóm các ký tự thành các thực thể cú pháp token. Token là ký hiệu kết thúc, mỗi token sẽ có một cấu trúc từ vựng. Cấu trúc từ vựng này là một cặp (loại token, dữ liệu), gồm hai thành phần:

Thành phần thứ nhất là phạm trù cú pháp: hằng, biến.

Thành phần thứ hai là con trỏ, chỉ đến thông tin của token, được cất giữ trong bảng, được gọi là bảng danh biểu (symbol table).

Với ngôn ngữ lập trình cho trước, số lượng loại token là hữu hạn. Tóm lại bộ phân tích từ vựng là bộ dịch (translator) mà đầu nhập của nó là chuỗi các ký tự, tượng trưng cho chương trình nguồn, đầu ra là các token. Dạng đầu ra này là đầu nhập của bộ phân tích cú pháp.

Thí dụ 3: Chương trình nguồn là phát biểu gán trong ngôn ngữ Pascal:

COST:=(PRICE+TAX)*65

Bộ phân tích từ vựng có nhiệm vụ nhận biết: COST, PRICE, TAX là nhóm token thuộc loại danh biểu, 65 là token thuộc loại hằng. Các ký tự :=, (,), +, * tự bản thân là token.

Giả sử tất cả các hằng, danh biểu là các token có loại <num> và <id>. Thành phần thứ hai là dữ liệu, ở đây chính là con trỏ chỉ đến vị trí của các token đó trong bảng danh biểu, chứa đựng trị từ vựng (lexeme) của token và các thuộc tính khác của token. Thành phần thứ nhất của token sẽ được dùng trong giai đoạn phân tích cú pháp. Thành phần thứ hai của token được dùng trong giai đoạn xử lý ngữ nghĩa và sinh mã đối tượng.

Trở lại với Thí dụ 3, ta thấy đầu ra của bộ phân tích từ vựng sẽ có các token:

(<id>, 1):=(<id>, 2)+(<id>, 3)∗(<num>, 4)

Để đơn giản ta viết lại như sau:

$$id_1:=(id_2+id_3)*num_4$$

Các chỉ số 1, 2, 3, 4 là con trỏ của token tương ứng trong bảng danh biểu. Các ký hiệu :=, (,), +, ∗ là các token, loại token sẽ được hiểu là chính nó, không có dữ liệu, nên chúng không có thành phần thứ hai là con trỏ.

Sự phân tích từ vựng sẽ đơn giản nếu token có nhiều hơn một ký tự, được cô lập bởi các ký tự mà tự chúng là token. Ở thí dụ trên, ta thấy COST, PRICE, TAX, 65 là các token, được tạo bởi nhiều hơn một ký tự được cô lập bởi các token :=, (, +,), ∗. Rõ ràng :=, (, +,), ∗ không thể là một thành phần trong COST, PRICE, TAX, 65. Song trong các trường hợp khác thì phân tích từ vựng không phải đơn giản như vậy.

Thí dụ 4: Trong Fortran ta có hai phát biểu sau:

(1) DO 10 I=1.15

(2) DO 10 I=1,15

Giả sử dấu trống được bỏ qua khi bộ phân tích từ vựng xét các ký tự. Như vậy ở trường hợp (1) DO10I là biến và 1.15 là hằng và (1) là phát biểu gán. Trong trường hợp (2) DO là từ khoá, I là biến vòng DO, 1 và 15 là hằng. Trong hai trường hợp trên, bộ phân tích từ vựng chỉ xác định được token kế tiếp khi gặp dấu '.' trong (1) và dấu ',' trong (2).

Như vậy bộ phân tích từ vựng cần phải nhìn thấy trước một token. Nhưng nhìn thấy trước một token nhiều khi cũng chưa đủ.

Chẳng hạn, DECLARE(X_1, X_2, …, X_n) trong ngôn ngữ PL/I. Bộ phân tích từ vựng sẽ không thể nói được DECLARE là tên hàm và X_1, X_2, …, X_n là các đối số của hàm hay DECLARE là từ khoá khai báo biến, X_1, X_2, …, X_n là các danh biểu mà kiểu dữ liệu của chúng sẽ đứng ngay sau dấu đóng ngoặc ')'. Muốn kết luận được một trong hai trường hợp là đúng, bộ phân tích từ vựng phải có khả năng nhìn trước một khoảng cách bất kỳ. Tuy nhiên, ta có thể nghĩ đến cách phân tích từ vựng khác, tránh được vấn đề nhìn trước một khoảng bất kỳ. Từ những suy nghĩ trên mà ta có hai cách tiếp cận với việc phân tích từ vựng. Hầu hết các kỹ thuật được sử dụng để phân tích từ vựng cũng nằm một trong hai phạm trù trên.

Bộ phân tích từ vựng thao tác trực tiếp: Nếu có một chuỗi ký tự của văn bản nhập và có con trỏ trong văn bản, đánh dấu vị trí bắt đầu tìm token thì bộ phân tích từ vựng sẽ xác định được ngay token nằm phía bên phải của vị trí con trỏ. Sau đó nó sẽ chuyển vị trí con trỏ về bên phải, ở vị trí ký tự đầu tiên, đứng sau ký tự cuối cùng của token vừa được xác định.

Bộ phân tích từ vựng thao tác không trực tiếp: Nếu với chuỗi ký tự của văn bản nhập cho trước, có con trỏ trong văn bản và loại token cho trước, bộ phân tích từ vựng sẽ xác định được token nếu các ký tự đứng ngay sau con trỏ tạo nên token của loại token cho trước. Nếu đúng, con trỏ sẽ được chuyển đến ký tự đứng ngay sau token vừa được xác định.

5.2.3. Bảng danh biểu:

Các token được bộ phân tích từ vựng nhận biết và các thông tin của từng token sẽ được lưu chứa trong bảng danh biểu. Xét phát biểu trong Thí dụ 3. Sau hi phát biểu được đi qua bộ phân tích từ vựng, bảng danh biểu sẽ chứa các thông tin sau:

Chỉ số	Token	Lexeme	Các thông tin khác
1	id	COST	biến thực
2	id	PRICE	biến thực
3	id	TAX	biến thực
4	Num	65	hằng số nguyên

Bảng danh biểu

Nếu bộ phân tích từ vựng nhận tiếp các chuỗi ký tự của chương trình nhập, để nhận dạng token, thì bảng danh biểu cũng thường xuyên được truy xuất. Hành vi truy xuất nhằm hai mục đích: nếu danh biểu vừa được nhận dạng đã được lưu chứa trong bảng danh biểu thì phần thứ hai của token là dữ liệu sẽ được cập nhật bằng chỉ số của danh biểu đó trong bảng danh biểu.

Thí dụ 5: Với phát biểu trong Thí dụ 3, COST có chỉ số là 1 trong bảng danh biểu, COST lại xuất hiện trong chuỗi nhập sau:

$$y:=COST*2.0$$

Chuỗi xuất ra của bộ phân tích từ vựng là:

$$id_5:=id_1*num_6 \Leftrightarrow (id, 5):=(id, 1)*(num, 6)$$

Trong trường hợp này COST không cất vào bảng danh biểu nữa, nhưng bộ phân tích từ vựng sẽ đẩy ra token (id, 1), 1 là vị trí COST đã được cất trong bảng danh biểu trước đó.

Bảng danh biểu thường xuyên được truy xuất để thêm hoặc truy xuất các token, do đó phải thoả mãn hai điều kiện:

1. Thực hiện nhanh các thao tác thêm token, hoặc các thông tin của token.
2. Có khả năng truy xuất nhanh các thông tin của một token.

5.2.4. Phát hiện và thông báo lỗi:

Ở mỗi giai đoạn của quá trình biên dịch một chương trình nguồn đều có thể có lỗi. Như vậy sau khi phát hiện một lỗi, trình biên dịch xem xét lỗi đó xem có

tiếp tục quá trình dịch hay không. Tất nhiên, nếu một trình biên dịch mà ngay khi phát hiện lỗi đầu tiên đã dừng chương trình thì không hữu hiệu.

Trong giai đoạn phân tích từ vựng và cú pháp thường xuất hiện nhiều lỗi do trình biên dịch phát hiện. Trong lúc phân tích từ vựng, lỗi được phát hiện khi phần còn lại trên băng nhập không thể tạo nên token. Lỗi xảy ra khi bộ phân tích cú pháp không thể xây dựng cấu trúc cú pháp cho chuỗi token cho trước. Lỗi cũng có thể được phát hiện trong quá trình phân tích ngữ nghĩa, khi trình biên dịch kiểm tra kiểu dữ liệu của hai toán hạng thuộc một phép toán không phù hợp. Chẳng hạn, một toán hạng thuộc kiểu dãy, cộng với một toán hạng là tên của chương trình con.

5.2.5. Phân tích cú pháp (Syntactic analysis parsing)**:**

Chuỗi xuất ra từ bộ phân tích từ vựng là các token có dạng (loại token, dữ liệu), sẽ là chuỗi nhập vào bộ phân tích cú pháp. Bộ phân tích cú pháp chỉ xét thành phần thứ nhất của token là loại token.

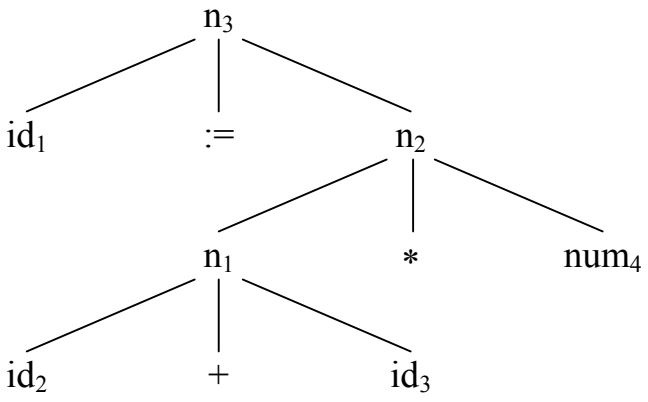
Sự phân tích cú pháp là một quá trình, trong quá trình này chuỗi các token sẽ được kiểm tra xem có thể được biểu diễn bằng cấu trúc cú pháp của ngôn ngữ lập trình cho trước hay không? Nếu tồn tại một cấu trúc cú pháp cho chuỗi nhập thì cấu trúc được sinh ra đó chính là kết quả của quá trình phân tích cú pháp. Ở giai đoạn sinh mã, cấu trúc cú pháp sẽ được xem xét để từ đó sinh ra mã cho chuỗi ký tự của chương trình nguồn.

Thí dụ 6: Với phát biểu trong Thí dụ 3, kết quả của quá trình phân tích từ vựng:

COST:=(PRICE+TAX)∗65 $\xrightarrow{\text{Phân tích từ vựng}}$ id_1:=(id_2+id_3)∗num_4

và kết quả của quá trình phân tích cú pháp là:

id_1:=(id_2+id_3)∗num_4 $\xrightarrow{\text{Phân tích cú pháp}}$



Cây cú pháp của phát biểu COST:=(PRICE+TAX)∗65

Vậy, kết quả của quá trình phân tích cú pháp của một chuỗi nhập là cấu trúc cú pháp được biểu diễn bằng cấu trúc cây. Cây để biểu diễn cấu trúc cú pháp của một chuỗi nhập được gọi là cây cú pháp hay cây phân tích. Với một chuỗi token là chuỗi nhập và tập luật sinh cho trước, bộ phân tích cú pháp sẽ tự động tìm ra cây

cú pháp cho chuỗi nhập. Khi cây cú pháp được xây dựng xong thì quá trình phân tích cú pháp của chuỗi nhập cũng kết thúc thành công. Ngược lại, nếu bộ phân tích cú pháp áp dụng tất cả các luật sinh hiện có, nhưng không thể xây dựng được cây cú pháp của chuỗi nhập cho trước thì bộ phân tích cú pháp sẽ ra thông báo rằng chuỗi nhập không được viết đúng cú pháp của ngôn ngữ lập trình. Nhìn vào cây cú pháp ở trên với các nhãn của các nút n_1, n_2, n_3 ta thấy được trình tự thực hiện:

(1) n_1 là nút miêu tả phép toán:

$id_2 + id_3$ (PRICE+TAX)

(2) n_2 miêu tả phép toán:

$n_1 * num_4$ (kết quả ở (1) * 65)

(3) là phép toán:

$id_1 := n_2$ (tức là gán kết quả của phép (1) * 65 vào biến COST)

Ta thấy rằng dấu '(' và ')' không hiện diện trên cây cú pháp, song việc thực hiện phép toán ở n_1: $id_2 + id_3$ trước phép nhân với num_4 đã chứng tỏ sự có mặt của chúng.

5.2.6. Phân tích ngữ nghĩa:

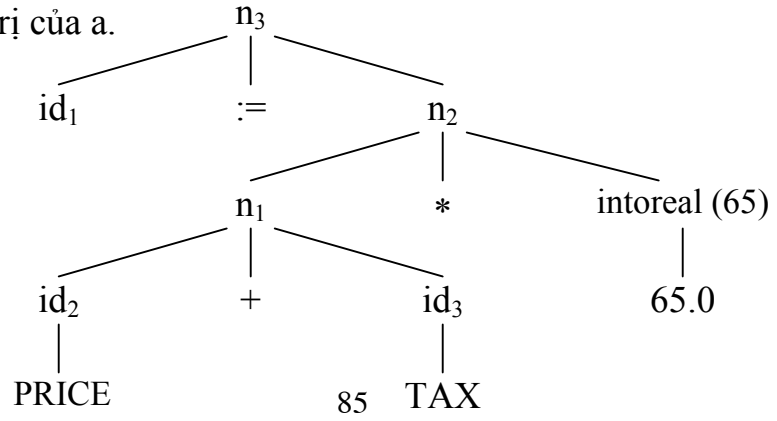
Sau giai đoạn phân tích cú pháp, cấu trúc cú pháp của chuỗi nhập sẽ được bộ phân tích ngữ nghĩa xử lý. Bộ phân tích ngữ nghĩa sẽ kiểm tra lỗi về ngữ nghĩa.. Một nhiệm vụ quan trọng mà bộ phân tích ngữ nghĩa thực hiện là kiểm tra loại dữ liệu. Dựa trên cây cú pháp, bộ phân tích ngữ nghĩa sẽ xử lý từng phép toán. Với mỗi phép toán, nó sẽ xét các toán hạng xem loại dữ liệu của chúng có cho phép để tham gia vào phép tính đó không (nói cách khác loại dữ liệu của các toán hạng trong phép toán cụ thể, có được ngôn ngữ lập trình định nghĩa không).

Thí dụ 7: a + 1 với a là biến thuộc loại dữ liệu số thực, 1 là thuộc loại luận lý.

Vậy phép cộng không thể thực hiện với hai toán hạng loại số thực và loại luận lý.

Hoặc: a + n với a là số thực và n là số nguyên

Khi kiểm tra thấy hai toán hạng của phép cộng một có trị thực, một có trị nguyên thì hầu hết các trình biên dịch sẽ chuyển trị của toán hạng n sang biểu thức số thực, cụ thể nếu n có trị là 10 thì trị 10 sẽ được chuyển sang trị thuộc loại thực 10.0 để cộng với trị của a.



Với phát biểu trong Thí dụ 3, trị 65 sẽ được chuyển sang số thực. Cây cú pháp khi xử lý ngữ nghĩa sẽ có dạng như trên.

5.2.7. Sinh mã trung gian:

Sau giai đoạn phân tích cú pháp và ngữ nghĩa, một số trình biên dịch đã tạo ra sự biểu diễn trung gian của chương trình nguồn. Sự biểu diễn trung gian của chương trình nguồn được hiểu như là chương trình của máy tính trừu tượng (abstract machine).

Ngôn ngữ được dùng cho máy trừu tượng là mã trung gian. Mã trung gian có hai đặc điểm quan trọng: dễ được sinh ra và dễ chuyển sang mã đối tượng của chương trình đích. Với Thí dụ 3, kết quả của giai đoạn sinh mã trung gian có dạng:

temp p_1 := intoreal (65)
temp p_2 := id_2 + id_3 (1)
temp p_3 := temp p_2 * temp p_1
id_1 := temp p_3

5.2.8. Tối ưu mã trung gian:

Giai đoạn này sẽ thu giảm một số bước trong mã trung gian nhằm làm cho khi sinh ra mã đối tượng thì thời gian thực thi mã đối tượng sẽ ngắn hơn.

Bước sinh mã sẽ dùng cây cú pháp đã được xử lý ngữ nghĩa (đã qua bước phân tích ngữ nghĩa) để sinh mã trung gian.

Cách làm thông thường như sau:cứ ứng với nút là toán tử sẽ sinh ra mã trung gian như ở (1). Tuy vậy, có cách tốt hơn là với (1) chỉ cần hai mã trung gian mà thôi.

temp p_1 := id_2 + id_3 (2)
id_1 := temp p_1 + 65.0

Việc thu giảm như trên sẽ được thực hiện ở bước tối ưu mã. Bước chuyển số nguyên sang số thực sẽ được thực hiện ngay trong thời gian dịch, do đó phép toán intoreal sẽ được bỏ đi. Xem lại (1), ta thấy ở mã thứ tư id_1 := temp p_3, với temp p_3 chỉ dùng có một lần là gán trị vào id_1, do đó có thể ghép mã thứ 3 và thứ 4 thành mã thứ 2 của (2).

Còn rất nhiều trường hợp khác mà trình biên dịch thực hiện tối ưu mã. Ở đây một vấn đề nảy sinh là thực hiện tối ưu mã trong thời gian biên dịch sẽ làm thời gian dịch tăng lên trong giai đoạn này. Tuy nhiên một số trường hợp tối ưu mã cho phép nếu thời gian thực thi của chương trình đích được rút ngắn mà không làm sự biên dịch quá lâu.

5.2.9. Sinh mã đối tượng:

Giai đoạn cuối của trình biên dịch là sinh mã đối tượng. Mã đối tượng có thể là mã máy định vị lại địa chỉ hoặc mã hợp ngữ.

Thí dụ 8: Ta sử dụng hai thanh ghi 1 và 2, để dịch mã trung gian (2) sang mã hợp ngữ:

movF id_2, R_1
movF id_3, R_2
addF R_2, R_1 (3)
mulF # 65.0, R_1
movF R_1, id_1

Lưu ý rằng movF, addF, mulF là phép tính với số thực. Chỉ thị đầu thực hiện chuyển trị từ vị trí nhớ có tên PRICE, thuộc loại token id_2 vào thanh ghi R_1. Chỉ thị thứ hai thực hiện chuyển trị ở vị trí nhớ có tên TAX thuộc loại token id_3 vào thanh ghi R_2. Chỉ thị thứ ba thực hiện phép cộng nội dung hai thanh ghi R_1 và R_2, kết quả phép toán được cất trong R_1. Chỉ thị thứ tư thực hiện phép nhân hằng có trị số thực 65.0 với trị nằm trong thanh ghi R_1. Chỉ thị cuối cùng chuyển nội dung trong thanh ghi R_1 vào vị trí nhớ có tên COST thuộc loại token id_1.

5.3. CÁC MỐI LIÊN QUAN VỚI TRÌNH BIÊN DỊCH.

Các phần trước của chương này ta có nói chuỗi ký tự nhập vào trình biên dịch là văn bản của chương trình nguồn. Đúng vậy, song văn bản đó lại có thể là sản phẩm đầu ra của một hoặc nhiều bộ tiền xử lý (preprocessor) và sản phẩm đầu ra của trình biên dịch có thể lại tiếp tục được xử lý trước khi trở thành mã máy của máy tính thật. Trong phần này ta sẽ nói tới các mối liên quan đó.

5.3.1. Bộ tiền xử lý:

Bộ tiền xử lý sẽ tạo ra chuỗi nhập vào trình biên dịch. Bộ tiền xử lý thực hiện các chức năng sau:

1. Xử lý macro (macro processing). Bộ tiền xử lý có thể cho phép người sử dụng định nghĩa các macro. Macro được hiểu là cách viết ngắn gọn cho cấu trúc dài hơn.

2. Chêm tập tin (file inclusion). Bộ tiền xử lý có thể "nhét" các tập tin vào chương trình văn bản. Chẳng hạn, tiền xử lý ngôn ngữ C sẽ "nhét" nội dung của tập tin <global.h> vào thay thế cho phát biểu # include <global.h> khi nó xử lý một tập tin có chứa phát biểu trên.

3. Bộ xử lý hoà hợp (Rational processor). Bộ tiền xử lý loại này sẽ tạo nên sự hoà hợp giữa ngôn ngữ cổ điển với những cấu trúc điều khiển, cấu trúc dữ liệu hiện đại hơn.. Chẳng hạn, bộ tiền xử lý giúp cho người sư dụng có thể dùng các phát biểu có cấu trúc như while, if trong ngôn ngữ lập trình, mà tự bản thân ngôn ngữ đó không có các phát biểu trên. Thực tế các phát biểu while, if chính là các macro, khi người sử dụng viết một chương trình trong ngôn ngữ cổ điển có dùng tới hai loại phát biểu có cấu trúc trên và cần biên dịch ra ngôn ngữ máy thì bộ tiền xử lý sẽ làm

việc trước. Tất cả nơi nào có hai phát biểu while, if sẽ được thay thế bởi chuỗi các phát biểu mà ngôn ngữ lập trình cổ điển có.

4. Mở rộng ngôn ngữ (language extension). Bộ tiền xử lý tăng khả năng cho ngôn ngữ bằng một số các macro nội tại của nó. Thí dụ ngôn ngữ Equel là ngôn ngữ hỏi đáp với cơ sở dữ liệu được nhúng vào ngôn ngữ C. Các phát biểu được bắt đầu bằng hai dấu # # ở trong C được bộ tiền xử lý, xử lý, là các phát biểu truy xuất cơ sở dữ liệu, không liên quan đến C, được dịch thành các phát biểu gọi thủ tục, sẽ gọi các trình con đặc nhiệm trong mã máy để thực hiện việc truy xuất cơ sở dữ liệu.

Bây giờ ta sẽ nói kỹ hơn về bộ xử lý macro. Bộ xử lý này thường làm việc với hai loại phát biểu: định nghĩa macro và sử dụng macro.

Định nghĩa macro bao gồm: từ khoá define hoặc macro, tiếp theo là tên macro. Theo sau là thân (body) của macro.

Chẳng hạn, \define <macroname> {<body>}.

Thông thường bộ xử lý macro cho phép các thông số hình thức (formal parameter) trong định nghĩa, chúng là các ký hiệu sẽ bị thay thế bởi các trị (chuỗi các ký tự) sau này khi macro được dùng.

Phát biểu dùng macro bao gồm: tên macro và các thông số thực (actual parameter), là trị của các thông số hình thức trong thân của macro.

Thí dụ 9: Hệ thóng đánh máy typesetting có phương tiện macro với phát biểu định nghĩa macro như sau:

\define <macro name><template> {<body>}

<macrro name>: tên macro

<template> : danh sách thông số hình thức

<body> : thân macro

Macro định nghĩa ve sự trích dẫn của tạp chí ACM như sau:

\define\JACM #1; #2; #3

{{\S1 J.ACM}{\bf #1}: #2, pp. #3}

Tên macro là \JACM. Các thông số hình thức là #1, #2, #3 được ngăn cách nhau bởi dấu ';' và được kết thúc bằng dấu '.'.

Khi dùng macro, người sử dụng sẽ viết như sau: \JACM 17; 4; 715 – 728 sẽ được hiểu như sau: J.ACM 17: 4, pp. 715 – 728.

5.3.2. Trình biên dịch hợp ngữ:

Một số trình biên dịch cho sản phẩm ở đầu ra là mã hợp ngữ, chuỗi mã hợp ngữ này sẽ được đưa sang trình biên dịch hợp ngữ xử lý tiếp. Một số trình biên dịch khác thực hiện luôn công việc của assembler, nghĩa là nó dịch ra luôn mã máy khả định vị (relocatable machine code), mã máy sẽ được chuyển trực tiếp đến bộ phận "loader/link editor.

Mã hợp ngữ là phiên bản gợi nhớ của mã máy, trong đó các tên sẽ được dùng thay thế cho các mã nhị phân của các tác vụ và tên cũng được đại diện cho các địa chỉ của vị trí nhớ. Chẳng hạn, chuỗi chỉ thị trong mã hợp ngữ của phát biểu gán b := a+2.

mov a, R_1
add #2, R_1 (4)
mov R_1, b

Ba chỉ thị thực hiện việc chuyển nội dung ở địa chỉ a vào thanh ghi R_1, sau đó cộng hằng số 2 với nội dung của R_1 và kết quả được giữ lại trong thanh ghi R_1, cuối cùng là chuyển nội dung của R_1 vào địa chỉ b. Sau khi thực hiện ba chỉ thị thì máy thực sự đã thực hiện phát biểu gán b:=a+2. Thông thường hợp ngữ cũng có các phương tiện macro và bộ tiền xử lý macro.

5.3.3. Trình biên dịch hợp ngữ hai chuyến (two pass assembler):

Trình biên dịch hợp ngữ đơn giản nhất là biên dịch hai chuyến trên dữ liệu nhập vào. Chuyến ở đây được coi là lần đọc tập tin nhập trọn vẹn. Ở chuyến đầu, toàn bộ danh biểu, đại diện cho vị trí nhớ sẽ được nhặt ra, cất vào bảng danh biểu.

Danh biểu	Địa chỉ tương đối
a	0
b	4

Theo bảng bên, ta giả sử địa chỉ được đánh cho từng từ (một từ là 4 byte). a là danh biểu đại diện cho địa chỉ bắt đầu ở byte 0. b ở thứ 4. Ở chuyến thứ hai, trình biên dịch hợp ngữ sẽ rà lại tập tin nhập một lần nữa. Lần này nó sẽ dịch mã gợi nhớ (được đặt bằng tên) của tác vụ sang chuỗi mã máy – mã nhị phân và phần tên danh biểu đại diện cho vị trí nhớ sẽ được thay thế bằng địa chỉ tương đối của danh biểu đó trong bảng danh biểu.

Thí dụ 10: Đoạn chỉ thị (4) được dịch sang mã máy là:

0001 010000000000*
0011 011000000010* (5)
0100 010000000100*

4 bit đầu là mã tác vụ 0001, 0011, 0100 là mã load, add, store. Hai bit tiếp theo 01 ở ba chỉ thị là mã của thanh ghi 1. 2 bit tiếp theo là mã thông báo cho biết 8 bit theo sau là địa chỉ hay toán hạng. Hai bit này được gọi là mode địa chỉ nếu là 00 và mode trực tiếp – toán hạng nếu là 10. Vì vậy 8 bit của chỉ thị 1 và 3 là địa chỉ, ngược lại ở chỉ thị 2, 00000010 là toán hạng, hằng nguyên có trị 2.

Đầu ra chuyến thứ hai của trình biên dịch hợp ngữ là mã máy khả định vị, nghĩa là chương trình trong dạng này có thể được chứa vào bộ nhớ ở bất kỳ vị trí L nào. Như vậy địa chỉ tương đối trong bảng danh biểu sẽ được tính lại, trở thành địa chỉ tuyệt đối, bằng cách lấy L cộng với địa chỉ tương đối, việc này được thực hiện

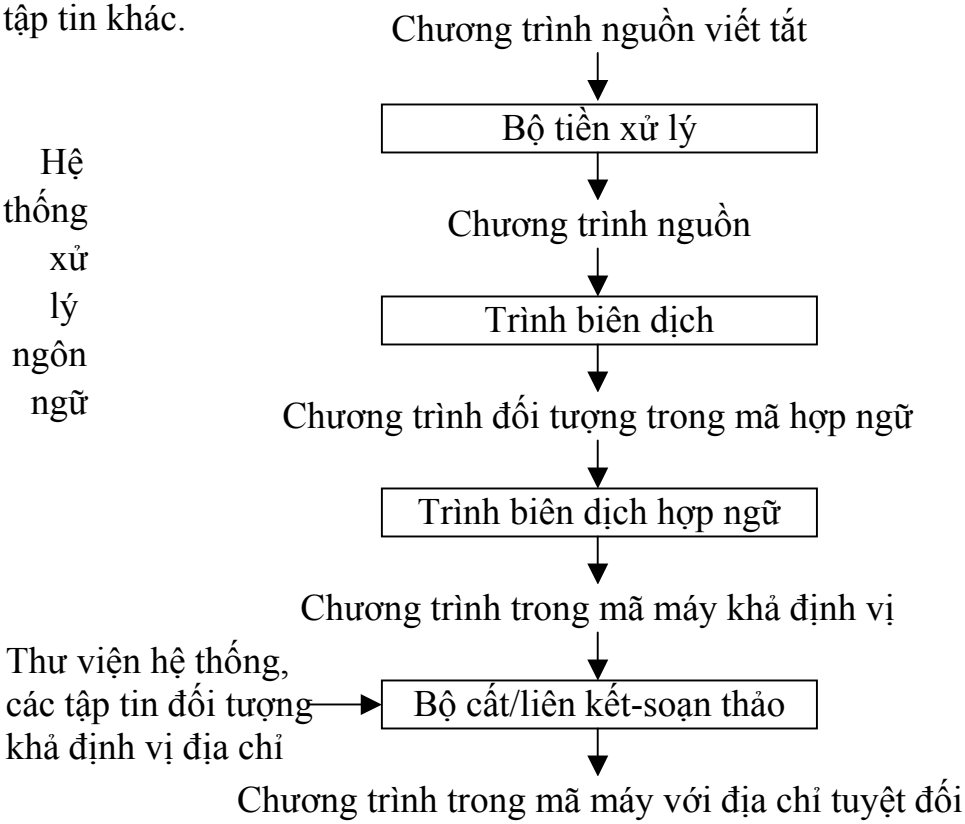
cho tất cả các danh biểu trong bảng danh biểu. Quay lại (5), ta thấy ở chỉ thị 1 và 3 thì 8 bit sau cùng là địa chỉ tương đối của danh biểu a, b. Giả sử L=00001111, địa chỉ tuyệt đối của a, b là 00001111, 00010011. Ba chỉ thị (5) được viết lại dưới dạng mã máy tuyệt đối:

0001010000001111
0011011000000010 (6)
0010010000010011

5.3.4. Bộ cất liên kết soạn thảo (loader/link editor):

Loader là chương trình, thực hiện hai nhiệm vụ sau: cất và soạn thảo liên kết. Quá trình cất bao gồm lấy mã máy khả định vị tính lại địa chỉ thành địa chỉ tuyệt đối như ở thí dụ trên. Sau đó ta đem cất tất cả chỉ thị với các địa chỉ tuyệt đối của danh biểu và dữ liệu vào trong bộ nhớ tại vị trí tương ứng như ở (6).

Link editor cho phép ta tạo một chương trình duy nhất từ nhiều tập tin ở dạng mã máy khả định vị của những lần biên dịch riêng biệt và từ những tập tin thư viện, do hệ thống cung cấp. Sự liên kết này tạo điều kiện thuận lợi cho bất kỳ chương trình nào cần tới chúng khi thực thi. Nếu có một số tập tin được chương trình khác tham chiếu, chúng sẽ được tham chiếu ngoài (external reference). Trong đó mã của tập tin này có thể tham chiếu đến một vị trí nhớ trong tập tin khác. Có nghĩa là vị trí nhớ chứa dữ liệu được khai báo trong một tập tin lại có thể được truy xuất ở tập tin khác. Hoặc thủ tục được khai báo trong tập tin này lại được gọi trong tập tin khác.



Mã khả định vị phải lưu giữ thông tin trong bảng danh biểu cho danh biểu và tên các thủ tục. Vì ta không thể biết được toàn bộ chương trình trong dạng mã khả định vị sẽ được chứa ở đâu trong bộ nhớ trong khi nó còn ở bộ nhớ ngoài, do đó toàn bộ bảng danh biểu phải được lưu giữ đầy đủ như là một phần của chương trình trong mã khả định vị.

Ở bảng trong 5.3.3, ta thấy: khi có một tập tin được thực thi, nó tham chiếu đến b thì vị trí nhớ của b + địa chỉ bắt đầu vùng dữ liệu của tập tin (6), được cất trong bộ nhớ trong.

5.4. NHÓM CÁC GIAI ĐOẠN CỦA TRÌNH BIÊN DỊCH.

Như trong phần trước ta đã thấy tổ chức luận lý của trình biên dịch gồm nhiều giai đoạn. Song thực tế một số các giai đoạn thường được gộp lại thành một giai đoạn lớn hơn.

5.4.1. Giai đoạn trước và giai đoạn sau (front end and back end):

Thông thường các giai đoạn được nhóm lại trong hai giai đoạn bao trùm hơn là giai đoạn trước (front end) và giai đoạn sau (back end). Giai đoạn trước bao gồm các giai đoạn, hoặc các phần của các giai đoạn mà chúng chỉ phụ thuộc vào ngôn ngữ nguồn mà hầu như không phụ thuộc vào máy đích. Giai đoạn đầu này bao gồm phân tích từ vựng, phân tích cú pháp, tạo bảng danh biểu, phân tích ngữ nghĩa, thông báo lỗi và sinh mã trung gian. Phần lớn tối ưu mã trung gian cũng nằm trong giai đoạn đầu. Giai đoạn sau bao gồm những phần phụ thuộc vào máy đích, mà không (về tổng quát) phụ thuộc vào ngôn ngữ nguồn. Giai đoạn này bao gồm giai đoạn sinh mã đối tượng, tối ưu mã đối tượng và tất nhiên nó cần các tác vụ của thông báo lỗi và bảng danh biểu.

Với ý niệm như vậy có thể xuất hiện một thủ tục đặc biệt, sẽ lấy giai đoạn đầu của trình biên dịch kết nối với các phần sau để tạo ra một trình biên dịch cho cùng một ngôn ngữ nguồn trên các máy khác nhau. Hoặc ngược lại, có thể các trình biên dịch cho nhiều ngôn ngữ nguồn khác nhau, có chung một ngôn ngữ trung gian và dùng chung giai đoạn cuối, sẽ cho ta nhiều trình biên dịch trên một máy.

5.4.2. Các chuyến:

Thông thường một số giai đoạn có thể hiện thực trong một chuyến. Chẳng hạn, phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa và sinh mã trung gian có thể được gom lại, hiện thực trong một chuyến. Nếu như vậy thì chuỗi token được nhận dạng sẽ được dịch thẳng sang mã trung gian. Nói chi tiết hơn, ta sẽ thấy vai trò bộ phân tích cú pháp là bao trùm, nó trông coi toàn bộ hoạt động của chuyến. Nó có nhiệm vụ phải phát hiện cấu trúc văn phạm của các token đưa đến cho nó. Nó lại phải biết lúc nào cần lấy tiếp token và nó sẽ gọi bộ phân tích từ vựng nhận dạng token kế tiếp. Khi đã phát hiện xong một cấu trúc văn phạm, bộ

phân tích cú pháp sẽ gọi bộ sinh mã trung gian, để thực hiện phân tích ngữ nghĩa và tạo mã trung gian.

5.4.3. Thu giảm số lượng các chuyến:

Nếu một quá trình dịch được chia thành nhiều chuyến, nó sẽ làm tăng thời gian để đọc và ghi lên bộ nhớ ngoài mã trung gian. Ngược lại, nếu ta gom một số giai đoạn thành một chuyến thì buộc phải giữ toàn bộ chương trình trong bộ nhớ. Bởi vì giai đoạn đoạn này sẽ cần những thông tin theo thứ tự khác với thứ tự mà giai đoạn trước tạo ra, như vậy vấn đề bộ nhớ không phải là đơn giản.

Việc thực hiện gom một số giai đoạn trong một chuyến phải giải quyết được một số vấn đề sau. Việc giao tiếp giữa bộ phân tích từ vựng và phân tích cú pháp có thể được giới hạn ở một token vì khi nào bộ phân tích cú pháp cần tới một token sẽ gọi bộ phân tích từ vựng cung cấp. Nhưng thật khó để thực hiện việc sinh mã đối tượng khi toàn bộ mã trung gian của chương trình nguồn chưa được tạo xong. Chẳng hạn, trong PL/I, Algol 68 cho phép dùng biến trước khi nó được khai báo, như vậy ta không thể tạo mã đối tượng cho một cấu trúc mã mà ta chưa biết loại của biến, được xuất hiện trong nó. Rõ ràng trong trường hợp này phải có sự phân tích ngữ nghĩa, kiểm tra kiểu dữ liệu tại cấu trúc cụ thể để quyết định xem biến đó thuộc kiểu nào, khi đã kết luận được thì bộ sinh mã trung gian mới sinh mã được. Cũng tương tự trong trường hợp phát biểu goto tham khảo trước, các phát biểu goto L có thứ tự đứng trước phát biểu có nhãn L.

Khi dịch ra mã đối tượng, bộ sinh mã sẽ sinh mã cho tác vụ goto còn địa chỉ có tên L thì chưa được thay thế vì tại thời điểm đó, nó chưa nhìn thấy chỉ thị có nhãn L, nên không biết chỉ thị đó nằm ở địa chỉ nào. Bộ sinh mã sẽ tạo ra một danh sách liên kết, ghi nhớ địa chỉ của các chỉ thị goto L. Khi gặp chỉ thị có nhãn L, bộ sinh mã đã xác định được địa chỉ có tên là L, nó sẽ lần theo danh sách liên kết để điền vào các chỉ thị goto địa chỉ của L.

PHỤ LỤC:
CÁC LỚP P VÀ NP
VÀ LỚP CÁC BÀI TOÁN NP-ĐẦY ĐỦ

Có những bài toán thực tế mà cho đến nay vẫn chưa xây dựng được thuật toán hiệu quả để giải (đó là thuật toán có độ phức tạp tính toán là đa thức) và chứng minh được mức độ khó thực chất của nó. Trong số các bài toán như vậy, có thể kể ra các bài toán nổi tiếng sau: Bài toán người du lịch, Bài toán chu trình Hamilton, Bài toán tô màu đồ thị, Bài toán tìm đường đi đơn dài nhất của đồ thị. Ta có thể quy lỗi cho việc thiết kế và phân tích thuật toán hay lý thuyết độ phức tạp hay không? Liệu trên thực tế có thuật toán hiệu quả để giải quyết các bài toán này không?

Trong phần này, ta sẽ có một kết quả nổi tiếng: mỗi thuật toán hiệu quả để giải một trong số các bài toán vừa kể trên sẽ cũng cho ta thuật toán hiệu quả để giải tất cả các bài toán còn lại. Ta chưa biết những bài toán này là dễ hay khó giải, nhưng ta biết rằng tất cả chúng có độ phức tạp như nhau. Ý nghĩa thực tế quan trọng của các bài toán này là đảm bảo rằng mỗi một bài toán này là đối tượng của những cố gắng tìm thuật toán hiệu quả để giải.

1. LỚP P VÀ LỚP NP.

1.1. Định nghĩa: Cho M là một máy Turing. Hàm T(n) được gọi là độ phức tạp tính toán của M nếu với mọi xâu vào ω có độ dài n thì đều tồn tại một dãy hình trạng có nhiều nhất là T(n) bước đoán nhận ω (ở đây T(n) là một số nguyên dương). Nếu có một xâu nào đó có độ dài n mà máy Turing không dừng thì đối với n đó, T(n) không xác định.

1.2. Định nghĩa: Lớp P là lớp các ngôn ngữ được đoán nhận bởi máy Turing đơn định và độ phức tạp tính toán là đa thức.

Có thể phát biểu một cách khác là: một bài toán được coi là thuộc lớp P nếu tồn tại một thuật toán đa thức để giải nó. Người ta nói rằng những bài toán thuộc lớp P là dễ.

1.3. Chú ý: Theo quan điểm toán học, lớp P là rất tự nhiên. Điều này thấy được từ việc nó là bất biến cao đối với mô hình tính toán được dùng. Chẳng hạn, các máy Turing M_1 với nhiều băng là nhanh hơn các máy Turing thông thường, tức là độ phức tạp tính toán của chúng nhận các giá trị nhỏ hơn. Tuy nhiên, nếu độ phức tạp tính toán của một máy Turing M_1 như vậy bị chặn trên bởi một đa thức $T_1(n)$, ta có thể xây dựng một máy Turing thông thường M với giới hạn thời gian đa thức T(n)

đoán nhận chính ngôn ngữ như M_1. (Nói chung, T(n) nhận giá trị lớn hơn $T_1(n)$ nhưng vẫn là đa thức). Tương tự, mỗi ngôn ngữ là trong giới hạn đa thức đối với một mô hình máy Turing chuẩn mực bất kỳ hay đối với một mô hình tính toán hợp lý bất kỳ đều thuộc vào lớp P được định nghĩa như ở trên đối với các máy Turing thông thường.

Lớp P cũng có tầm quan trọng quyết định vì các ngôn ngữ nằm ngoài P có thể xem là không thể tính được. Trên thực tế ta nói rằng một ngôn ngữ đệ quy là bất trị nếu nó không thuộc P.

Rõ ràng rằng các ngôn ngữ nằm ngoài P là bất trị theo quan điểm thực hành. Ta cũng có thể nói như vậy đối với các ngôn ngữ trong P có cận là một đa thức khổng lồ. Tuy nhiên, việc vạch ra một ranh giới giữa tính bất trị và tính không bất trị bên trong P là không tự nhiên lắm. Một định nghĩa như vậy sẽ thay đổi theo thời gian: sự phát triển kỳ diệu trong lĩnh vực máy tính có thể làm thay đổi ranh giới này. Mặt khác, lớp P cho ta một cách đặc trưng rất tự nhiên cho tính không bất trị.

Thí dụ 1: Bài toán tìm đường đi ngắn nhất giữa hai thành phố A và B là bài toán dễ vì độ phức tạp của thuật toán để giải nó là $O(n^2)$ (tức là một thuật toán đa thức).

Ta xét dưới đây các bài toán thuộc lớp sẽ được gọi là lớp NP.

Theo định nghĩa trên, ta nêu ra thí dụ về bài toán "khó". Giả sử người ta đòi hỏi xác định tất cả các con đường nối đỉnh S với đỉnh T trong một mạng nào đó và có độ dài nhỏ hơn $(1+\varepsilon)$ lần so với độ dài của đường đi ngắn nhất. Người ta có thể không có khả năng lập nên danh mục này trong thời gian đa thức (cách nói này có ý nghĩa tương tự với số các phép toán) vì một nguyên nhân đơn giản là danh mục này chứa một số các phần tử không đa thức (nghĩa là nó không bị chặn bởi một đa thức theo số các dữ liệu).

1.4. Định nghĩa: Một bài toán được gọi là "nhận biết" nếu đó là bài toán mà các kết quả chỉ có thể lập một trong hai giá trị tại ĐÚNG hay SAI.

Thí dụ 2: 1) Bài toán về việc tìm phân bố phù hợp.

Cho tập hợp $X=\{x_1, x_2, \ldots, x_n\}$ gồm các biến Boole và một biểu thức Boole đối với các số hạng của các biến này: $E=C_1 \wedge C_2 \wedge \ldots \wedge C_m$, trong đó C_i (i=1,…, m) là biểu thức $C_i=u_{j1} \vee u_{j2} \vee \ldots \vee u_{jk(i)}$, trong đó mỗi u_{jq} là một trong các biến của X.

Bài toán đặt ra là thử tìm xem có một phân bố các biến x_k (k=1,…,n) bằng 0 hay 1 sao cho E=1.

Đối với $E=(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge x_3$ có câu trả lời là ĐÚNG khi lấy x_1=0 hay 1, x_2=1, x_3=1. Tuy nhiên, câu trả lời là SAI trong trường hợp này đối với $E=(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2) \wedge x_3 \wedge (\overline{x_2} \vee \overline{x_3})$.

2) Bài toán về chu trình Hamilton. Vấn đề đặt ra là xác định xem trong một đồ thị G đã cho có một chu trình sơ cấp đi qua tất cả các đỉnh hay không?

Nghiệm của bài toán nhận biết chỉ là ĐÚNG hoặc SAI. Người ta không đòi hỏi gì hơn. Điều này phân biệt một cách cơ bản các bài toán nhận biết với các bài toán tồn tại cũng như đối với bài toán về sự tìm phân bố phù hợp, nếu câu trả lời là ĐÚNG, người ta không đòi hỏi cho một phân bố các biến của X cho E giá trị 1. Đối với bài toán chu trình Hamilton, người ta không đòi hỏi diễn tả chu trình.

1.5. Định nghĩa: Cho bài toán tối ưu hoá tổ hợp $\min_{s\in S}(f(s))$ (tương ứng $\max_{s\in S}(f(s))$) và một số a. Người ta định nghĩa "bài toán nhận biết liên hợp" là bài toán: liệu có tồn tại $s\in S$ sao cho $f(s)\le a$ (tương ứng $f(s)\ge a$).

Thí dụ 3: 1) Cho một tập n thành phố, các khoảng cách giữa các thành phố và một số a. Bài toán với nội dung là xác định xem có tồn tại một vòng đi với chi phí nhỏ hơn hoặc bằng a là bài toán nhận biết liên hợp của bài toán người du lịch.

2) Cho một ma trận $\overline{A}$ và vectơ $\overline{b}$ với các hệ số nguyên. Bài toán có nội dung là xác định xem có tồn tại vectơ x có các thành phần nguyên sao cho $\overline{A}x\le\overline{b}$ là một bài toán nhận biết.

Nếu đặt $\overline{A}=\begin{pmatrix}C\\A\end{pmatrix}$, $\overline{b}=\begin{pmatrix}a\\b\end{pmatrix}$, ta có thể coi bài toán nhận biết là liên hợp với bài toán quy hoạch tuyến tính nguyên:

$Cx=z(\min)$

$$\begin{cases}Ax\le b\\ x_j\in N,\ j=\overline{1,n}.\end{cases}$$

1.6. Định lý: Nếu bài toán nhận biết liên hợp của một bài toán tối ưu hoá tổ hợp đã cho là "khó" thì bài toán tối ưu hoá tổ hợp cũng là "khó".

Định lý 1.6 chỉ ra rằng bài toán tối ưu hoá tổ hợp ít nhất là "khó" như bài toán nhận biết liên hợp. Trong thực tế người ta luôn luôn có thể chứng minh rằng bài toán nhận biết (chẳng hạn bài toán người du lịch) không phải là "dễ hơn" bài toán tối ưu hoá tổ hợp mà nó liên hợp.

1.7. Nhận xét: Ký hiệu NP đặc trưng cho lớp các bài toán mà ta sẽ nghiên cứu bây giờ trở nên như là "lường gạt". Vấn đề là nó không phải thuộc các bài toán "không phải là đa thức" như người ta tưởng.

Giả sử rằng ta biết câu trả lời của một bài toán nhận biết là ĐÚNG. Nếu ta có thể chia sẻ sự tin chắc của ta cho một người "siêu quan sát" bằng thời gian đa thức thì bài toán thuộc lớp NP, ngay cả khi ta không biết tìm bằng thời gian đa thức một nghiệm s mà đối với nó câu trả lời là ĐÚNG. Người ta chỉ đòi hỏi rằng nếu nghiệm s được đề xuất thì người ta có thể thử lại bằng thời gian đa thức rằng câu trả lời tương ứng là ĐÚNG.

Các bài toán về sự tìm phân bố phù hợp, về chu trình Hamilton, về nhận biết liên hợp với bài toán người du lịch và bài toán nhận biết liên hợp của quy hoạch tuyến tính nguyên là các bài toán thuộc lớp NP.

Bây giờ ta xét các máy Turing không đơn định: khi đọc mỗi ký hiệu bất kỳ ở một trạng thái bất kỳ, máy được phép có một số khả năng hành động. Còn về các yếu tố khác, một máy Turing không đơn định được định nghĩa như một máy đơn định. Một từ ω được đoán nhận nếu nó sinh ra một tính toán đoán nhận được, độc lập với việc nó cũng có thể sinh ra các tính toán khác dẫn đến thất bại. Như vậy, khi quan hệ với các máy không đơn định, ta không quan tâm đến mọi con đường dẫn đến thất bại nếu có một con đường có thể có dẫn đến thành công.

Thời gian cần thiết để máy Turring không đơn định M đoán nhận một từ $\omega \in T(M)$ được định nghĩa bằng số bước trong tính toán ngắn nhất của M dùng để đoán nhận ω.

1.8. Định nghĩa: Lớp NP là lớp các ngôn ngữ được đoán nhận bởi các máy Turing không đơn định trong giới hạn đa thức.

1.9. Chú ý: Các bài toán trong lớp P là trị liệu được, trong khi đó, các bài toán trong lớp NP có tính chất là việc kiểm chứng xem một phỏng đoán tốt không đối với việc giải bài toán có là đúng đắn không là trị liệu được. Một máy Turing không đơn định có thể được hình dung như một thiết bị kiểm chứng xem một phỏng đoán có đúng hay không: nó tiến hành một (hay một số) phỏng đoán ở từng bước trong suốt quá trình tính toán và chung cuộc là việc đoán nhận chỉ trong trường hợp (các) phỏng đoán này là đúng đắn. Như vậy, trong thực tế một giới hạn thời gian đối với một máy Turing không đơn định là một giới hạn thời gian để kiểm chứng xem một phỏng đoán đối với lời giải có là đúng đắn không.

Dễ thấy lớp P là một lớp con của lớp NP. Tuy nhiên, ta không biết liệu bao hàm này có là thực sự hay không. Vấn đề "P có bằng NP hay không" có thể xem là vấn đề tồn tại nổi tiếng nhất trong lý thuyết tính toán. Vấn đề này có ý nghĩa vì nhiều bài toán quan trọng trong thực tế được biết là thuộc NP, trong khi đó ta không biết nó có thuộc P hay không. Thực ra, về mặt thời gian, mọi thuật toán đơn định được biết đối với các bài toán này đều là mũ. Như vậy, một chứng minh cho P=NP sẽ làm cho mọi bài toán này trị liệu được.

Các máy Turing không đơn định và việc đoán chừng vốn không được dự định để mô hình hoá việc tính toán. Tính không đơn định chỉ là một khái niệm bổ trợ và như ta sẽ thấy, nó rất tiện lợi. Thực vậy, nếu ta muốn giải quyết vấn đề có hay không đẳng thức P=NP, các định nghĩa và kết quả sau này chứng tỏ rằng chỉ cần xét một ngôn ngữ đặc biệt (có thể là một ngôn ngữ ta ưa thích!) và xác định

xem nó có thuộc P hay không. Có một số lớn và rất đa dạng các ngôn ngữ mà ta sẽ gọi là các ngôn ngữ NP-đầy đủ nhận được thực tế từ mọi lĩnh vực của toán học.

1.10. Định nghĩa: Ngôn ngữ $L_1 \subset \Sigma_1^*$ được gọi là dẫn được trong thời gian đa thức về ngôn ngữ $L_2 \subset \Sigma_2^*$, ký hiệu $L_1 \leq_P L_2$, nếu có một hàm xác định bởi máy Turing đơn định trong thời gian đa thức $f: \Sigma_1^* \longrightarrow \Sigma_2^*$ thoả mãn:

$$\forall \omega \in \Sigma_1^*,\ \omega \in L_1 \Leftrightarrow f(\omega) \in L_2.$$

Ta nhận thấy rằng máy Turing M được đưa vào trong định nghĩa trên phải dừng với mọi dữ liệu vào, đó là một hệ quả của việc M là đơn định và trong thời gian đa thức.

Kết quả tiếp theo là một hệ quả trực tiếp của định nghĩa.

1.11. Mệnh đề: Nếu $L_1 \leq_P L_2$ và $L_2 \in P$ thì $L_1 \in P$.

2. LỚP NP-ĐẦY ĐỦ.

Đối với phần lớn các bài toán thuộc lớp NP, người ta không nói được là chúng có thể giải được hay không bằng một thuật toán đa thức. Chỉ biết rằng người ta chưa tìm được một thuật toán đa thức để giải chúng.

Để chứng minh P=NP, ta phải chứng tỏ rằng trong lớp NP tất cả các bài toán có thể giải với thời gian đa thức bằng các thuật toán đơn định.. Để chứng minh P≠NP, ta phải chỉ ra một bài toán trong NP mà không thể giải được một cách tiền định với thời gian đa thức. Cách giải quyết hiện nay là xây dựng lớp các bài toán tương đương.

2.1. Định nghĩa: Một ngôn ngữ L được gọi là NP-khó nếu với mọi ngôn ngữ L' trong NP, ta có $L' \leq_P L$.

Ngôn ngữ L được gọi là NP-đầy đủ nếu nó là NP-khó và $L \in NP$.

2.2. Chú ý: Các ngôn ngữ NP-đầy đủ có thể hình dung như đại diện cho các bài toán khó nhất trong NP. Hơn nữa, để giải quyết vấn đề có P=NP không, chỉ cần quyết định xem một ngôn ngữ NP-đầy đủ L nào đó có thuộc P hay không. Thật vậy, xét một ngôn ngữ L như vậy. Nếu L không thuộc P thì rõ ràng P≠NP. Nếu L thuộc P thì định nghĩa của tính NP-đầy đủ và Mệnh đề 1.11 chứng tỏ rằng mỗi ngôn ngữ thuộc NP cũng thuộc P. Nhưng điều đó có nghĩa là P=NP.

Ta có thể xây dựng cho mỗi bài toán trong lớp NP một thuật toán làm việc trong thời gian đa thức miễn là ta biết một thuật toán (đơn định) trong giới hạn thời gian đa thức đối với một bài toán NP-đầy đủ nào đó. (Hiện thời ta nói về các bài toán thay cho các ngôn ngữ để nhắc nhở rằng có thể thay đổi qua lại giữa các khái niệm này). Như vậy, một khi chúng ta có được một thuật toán trong giới hạn thời gian đa thức cho một trong số rất nhiều bài toán NP-đầy đủ, ta sẽ có được thuật toán trong giới hạn thời gian đa thức cho mỗi bài toán trong lớp NP! Do những nỗ lực cực kỳ lớn dành cho dự định cải tiến các thuật toán đã được biết cho một số

trong các bài toán như vậy (do tầm quan trọng thực tế lớn lao của chúng) và do chưa một nỗ lực nào như vậy dẫn đến thành công, bây giờ nói chung người ta tin rằng P≠NP.

Mệnh đề sau đây là một hệ quả trực tiếp của tính bắc cầu của quan hệ $\leq_P$.

2.3. Mệnh đề: Nếu L_1 là NP-đầy đủ và L_2 là một ngôn ngữ trong lớp NP thoả mãn $L_1 \leq_P L_2$ thì ngôn ngữ L_2 cũng là NP-đầy đủ.

Thí dụ 4: Xét bảng chữ:

$$\Sigma = \{1, 2, \vee, \wedge, \bar{}, (,)\}.$$

Một từ ω trên bảng chữ Σ được gọi là một công thức được thiết lập đúng của phép tính mệnh đề, viết tắt là **wffpc**, nếu hoặc (1) hoặc (2) đúng.

(1) ω là một từ khác rỗng trên bảng chữ {1, 2}.

(2) Có các **wffpc** u và v sao cho:

$$\omega=(u \vee v) \text{ hay } \omega=(u \wedge v) \text{ hay } \omega=\bar{u}.$$

Về mặt trực giác, $\vee$, $\wedge$ và $\bar{}$ chỉ phép tuyển, phép hội và phép phủ định. Trong các **wffpc**, ta có thể có nhiều không hạn chế các biến x_i mà i là một số nguyên theo cách viết 2-adic. Chẳng hạn, thay vì x_9 thì ta viết 121. Điều kiện (1) nói rằng mỗi biến đơn lẻ là một **wffpc**.

Một cách hình thức, mọi từ con $\alpha \in \{1, 2\}^+$ của một **wffpc** ω thoả mãn các điều kiện:

$$\omega=\omega_1\alpha\omega_2,\ \omega_1 \notin \Sigma^*\{1, 2\},\ \omega_2 \notin \{1, 2\}\Sigma^*$$

được gọi là một biến.

Giả sử $\alpha_1, \ldots, \alpha_n$ là tất cả các biến có mặt trong một **wffpc** ω. Một ánh xạ T từ tập $\{\alpha_1, \ldots, \alpha_n\}$ đến tập {0, 1} được gọi là một phép gán giá trị chân lý cho ω. Giá trị chân lý của một biến α_i bằng $T(\alpha_i)$. Giá trị chân lý của $(u \vee v)$ (tương ứng của $(u \wedge v)$) bằng $\max(u_1, v_1)$ (tương ứng $\min(u_1, v_1)$), trong đó u_1 và v_1 tương ứng là các giá trị chân lý của u và v. Giá trị chân lý của $\bar{u}$ bằng $1-u_1$.

Một **wffpc** ω được gọi là thoả được nếu nó nhận giá trị chân lý 1 đối với một cách gán giá trị chân lý T nào đó. Ta ký hiệu ngôn ngữ trên Σ gồm mọi **wffpc** thoả được là SAT.

Về mặt trực giác, 1 và 0 ký hiệu tương ứng các giá trị chân lý đúng và sai.. Một **wffpc** là thoả được nếu nó không là đồng nhất sai theo kỹ thuật bảng chân lý quen thuộc. Tất nhiên, trong mỗi cách gán giá trị chân lý, mọi xuất hiện của mỗi biến cá biệt α_i nhận cùng một giá trị chân lý.

Sau đây, các quy tắc nghiêm ngặt về định nghĩa của một **wffpc** được giảm nhẹ đôi chút. Ta dùng các chữ thường ở cuối bảng chữ cái để ký hiệu các biến. Như vậy, một biến cá biệt có thể được ký hiệu là x_9 thay cho ký hiệu 121 đã chỉ ra trong định nghĩa. Các dấu ngoặc không cần thiết được bỏ đi. Quy ước này cũng áp

dụng đối với các dấu ngoặc không cần thiết do tính kết hợp của $\wedge$ và $\vee$. (Ta chỉ quan tâm đến các giá trị chân lý và rõ ràng rằng các hàm min và max là kết hợp).

Xét hai **wffpc** sau đây:

$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge x_3$ (1)

$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge \overline{x_3}$ (2)

Cả hai (1) và (2) đều là hội của **wffpc** mà mỗi một trong số chúng là tuyển của các ký hiệu chữ, trong đó các biến và các phủ định của chúng được gọi là các ký hiệu chữ. Ta nói rằng các **wffpc** thuộc loại này là ở dạng chuẩn hội. Hơn nữa, nếu mỗi tuyển chứa nhiều nhất ba (tương ứng hai) ký hiệu chữ, ta nói rằng **wffpc** này là ở dạng chuẩn 3-hội (tương ứng 2-hội). Như vậy, (2) là ở dạng chuẩn 3-hội và (1) ở dạng chuẩn 2-hội (đồng thời cũng ở dạng chuẩn 3-hội).

Ta ký hiệu ngôn ngữ trên Σ gồm mọi **wffpc** thoả được ở dạng chuẩn hội là CONSAT. Các ký hiệu 3-CONSAT và 2-CONSAT được định nghĩa tương tự. **wffpc** (1) thuộc 2-CONSAT nhưng **wffpc** (2) không thuộc 3-CONSAT vì tuyệt nhiên nó không là thoả được. Ta có thể thấy điều đó nhờ lý luận sau. Câu cuối cùng của (2) buộc ta phải gán trị 0 cho x_3. Do đó, các câu thứ hai và thứ ba buộc ta phải gán trị 1 và 0 tương ứng cho x_2 và x_1. Nhưng với cách gán trị này, câu thứ nhất nhận giá trị 0.

Rõ ràng tính thoả được là một tính chất có thể quyết định được. Ta chỉ cần kiểm tra qua tất cả 2^n cách gán trị chân lý có thể có đối với n biến. (Thực ra điều này cũng chẳng khác gì so với kỹ thuật bảng chân lý quen thuộc). Một cách kiểm tra vét cạn như thế dùng một lượng thời gian mũ (theo số biến hay độ dài của **wffpc** cho trước). Bây giờ ta mô tả một cách ngắn gọn một thuật toán để kiểm tra tính thoả được dựa trên việc rút gọn số biến. Ta giả thiết rằng dữ liệu vào được cho dưới dạng chuẩn hội. Thuật toán này bộc lộ sự khác nhau đáng kể giữa các dạng chuẩn 2-hội và 3-hội.

Giả sử rằng α là một **wffpc** ở dạng chuẩn hội. Như vậy

$$\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_k,$$

trong đó mỗi α_i là một tuyển của các ký hiệu chữ. Ta gọi các tuyển α_i là các mệnh đề.

Bước 1: Bảo đảm cho mỗi biến xuất hiện (hoặc bị phủ định hoặc không) nhiều nhất một lần trong mỗi mệnh đề. Điều này được thực hiện bằng cách biến đổi α như sau. Mỗi mệnh đề chứa cả x và $\overline{x}$ với một biến x nào đó bị bỏ đi khỏi α. Nếu x (tương ứng $\overline{x}$) xuất hiện một số lần trong một mệnh đề nào đó, nhưng xuất hiện này được thay bằng một xuất hiện duy nhất của x (tương ứng $\overline{x}$). Nếu tất cả bị bỏ đi, α là thoả được. (Thực tế là nó đồng dạng đúng). Trái lại, giả sử α' là **wffpc** thu được.

Bước 2: Thay α' bằng một **wffpc** α'' không chứa một mệnh đề nào chỉ có một ký hiệu chữ (và cũng thoả mãn điều kiện được đòi hỏi đối với α' sau Bước 1). Thực vậy, nếu x (tương ứng $\bar{x}$) xuất hiện đơn độc trong một mệnh đề nào đó, ta bỏ đi mọi mệnh đề chứa x (tương ứng $\bar{x}$) và tiếp đó loại bỏ $\bar{x}$ (tương ứng x) khỏi mọi mệnh đề mà trong đó nó xuất hiện cùng với một biến khác nào đó; nếu $\bar{x}$ (tương ứng x) xuất hiện một mình trong một mệnh đề khác nào đó, ta kết luận rằng α không là thoả được. Lặp lại thủ tục này cho tới khi thu được α'' như mô tả ở trên.
Bước 3: Nếu không có biến nào xuất hiện trong α'' vừa bị phủ định và vừa không bị phủ định, ta kết luận rằng α'' là thoả được. Nếu trái lại, ta chọn một biến x nào đó mà cả x và $\bar{x}$ đều xuất hiện trong α''. Ta tìm mọi mệnh đề

$$(x \vee \beta_1),..., (x \vee \beta_m), (\bar{x} \vee \gamma_1),..., (\bar{x} \vee \gamma_n),$$

trong đó x hay $\bar{x}$ xuất hiện. Giả sử δ là hội của mọi mệnh đề khác (nếu còn). Khi đó α'' là thoả được nếu **wffpc**

$$((\beta_1 \wedge ... \wedge \beta_m) \vee (\gamma_1 \wedge ... \wedge \gamma_n)) \wedge \delta$$

là thoả được. Ta nhận thấy rằng mỗi một trong số các β và γ chứa ít nhất một ký hiệu chữ và chứa đúng một ký hiệu chữ nếu α nguyên bản là ở dạng chuẩn 2-hội.

Nếu một trong số các β hay γ chứa hơn một ký hiệu chữ ta thay α'' bằng hai **wffpc**

$$\bar{\alpha} = \beta_1 \wedge ... \wedge \beta_m \wedge \delta \text{ và } \bar{\bar{\alpha}} = \gamma_1 \wedge ... \wedge \gamma_n \wedge \delta,$$

bảo đảm cả $\bar{\alpha}$ lẫn $\bar{\bar{\alpha}}$ đều không chứa cùng một mệnh đề hai lần (bằng cách bỏ đi những xuất hiện không cần thiết) và quay về Bước 1. **wffpc** ban đầu α là thoả được nếu $\bar{\alpha}$ hay $\bar{\bar{\alpha}}$ là thoả được.

Nếu mọi β và γ chứa đúng một ký hiệu chữ, ta thay α'' bằng **wffpc**

$$\alpha''' = (\beta_1 \vee \gamma_1) \wedge ... \wedge (\beta_1 \vee \gamma_n) \wedge ... \wedge (\beta_m \vee \gamma_n) \wedge \delta,$$

loại bỏ những xuất hiện bị lặp lại của cùng một mệnh đề và quay về Bước 1. α ban đầu là thoả được nếu α''' là thoả được.

Đến đây ta kết thúc việc mô tả thuật toán. Chúng ta có thể dễ dàng kiểm nghiệm rằng phương pháp này có hiệu lực. Một số giải thích đã được cho ở trên. Điều cốt yếu là số lượng biến thực sự giảm trước mỗi lần quay về Bước 1.

Xét các từ có dạng:

$$\omega_0 \# \omega_1 \# \ldots \# \omega_k$$

trên bộ chữ cái {1, 2, #} sao cho k≥1, mỗi ω là một từ không rỗng trên bộ chữ cái {1, 2} và hơn nữa, ω_0 bằng tổng của một số ω khác nào đó khi các từ được xem như là các số nguyên 2-adic. Ta ký hiệu KNAPSACK là ngôn ngữ gồm mọi từ như vậy.

2.4. Định lý: Ngôn ngữ 2-CONSAT thuộc lớp P.

2.5. Định lý: Ngôn ngữ SAT là NP-đầy đủ.

2.6. Định lý: Ngôn ngữ CONSAT là NP-đầy đủ.

2.7. Định lý: Ngôn ngữ 3-CONSAT là NP-đầy đủ.

2.8. Định lý: Ngôn ngữ KNAPSACK là NP-đầy đủ.

2.9. Định lý (J. Demetrovics – V.Đ. Thi, 1999)**:** Cho s = ($\mathcal{U}$, F) là một sơ đồ quan hệ trên $\mathcal{U}$. Giả sử $\mathcal{U}$=$\{a_1, \ldots, a_n\}$ và F=$\{A_1 \to B_1, \ldots, A_t \to B_t\}$. Ký hiệu V_s=$\{A \mid A \subset \mathcal{U}, A^+ \neq \mathcal{U}\}$ (nghĩa là V_s là tập các tập con của $\mathcal{U}$ mà không phải là khoá) và m là số nguyên dương, m ≤ $|\mathcal{U}|$. Khi đó bài toán xác định xem có tồn tại một phần tử $A \in V_s$ mà m ≤ $|\mathcal{U}|$ hay không là NP-đầy đủ.

Chứng minh: Chọn tuỳ ý một tập A sao cho m ≤ |A|. Kiểm tra xem $A^+ \neq \mathcal{U}$ hay không. Việc kiểm tra này là thực hiện trong thời gian đa thức, vì thuật toán xây dựng bao đóng của một tập thuộc tính bất kỳ của s có thời gian tính đa thức. Như vậy thuật toán của chúng ta là bất định và có độ phức tạp tính toán đa thức. Vậy bài toán của ta thuộc lớp NP.

Bài toán tập độc lập sau của Garey và Johnson (1979) là bài toán NP-đầy đủ:

Cho trước số nguyên dương m và đồ thị G=(V, E), với V là tập các đỉnh và E là tập các cung, E=$\{(a_i, a_j) \mid a_i, a_j \in V\}$. Ta gọi A là tập độc lập của đồ thị G nếu A là tập con của V và với mọi a, b∈A thì (a, b)∉E. Kiểm tra xem có tồn tại tập độc lập A của G mà m ≤ |A| hay không.

Ta sẽ chứng minh rằng bài toán độc lập trên là được chuyển đa thức về bài toán của chúng ta.

Cho G=(V, E) là đồ thị mà m ≤ |V|. Xây dựng sơ đồ quan hệ s = ($\mathcal{U}$, F) với $\mathcal{U}$=V và F=$\{\{a_i, a_j\} \to \{a\} \mid (a_i, a_j) \in E$ và $a \in V \setminus \{a_i, a_j\}\}$. Rõ ràng s được xây dựng trong thời gian đa thức theo kích thước của G.

Theo định nghĩa tập cạnh, rõ ràng E là một siêu đồ thị đơn trên V (định nghĩa về siêu đồ thị và các khái niệm, tính chất liên quan có thể tìm đọc ở bài báo của Vũ Đức Thi "Some results about hypergraph" trong Tạp chí Tin học và Điều khiển học, tập 13, số 2, năm 1997). Từ điều này, ta thấy rằng s là dạng chuẩn BCNF. Do định nghĩa khoá tối thiểu và định nghĩa tập E nên có thể thấy nếu $(a_i, a_j) \in E$ thì $\{a_i, a_j\}$ là một khoá tối thiểu của s. Ngược lại, nếu $B \in K_s$ thì có $\{a_i, a_j\}$ sao cho $\{a_i, a_j\} \subset B$. Vì B là một khoá tối thiểu nên ta có $\{a_i, a_j\}$=B. Do đó K_s=E.

Như vậy A không phải là khoá của s khi và chỉ khi $\{a_i, a_j\} \not\subset A$ với mọi $(a_i, a_j) \in E$. Do đó A không phải là khoá của s khi và chỉ khi A là một tập độc lập của đồ thị G.

2.10. Định nghĩa: Cho s = ($\mathcal{U}$, F) là một sơ đồ quan hệ. Phụ thuộc hàm $A \to \{a\} \in F^+$ được gọi là phụ thuộc hàm cực đại của s nếu $a \notin A$ và với mọi $A' \subset A$, $A' \to \{a\} \in F^+$ kéo theo A'=A.

Đặt T_a={A | A→{a} là phụ thuộc hàm cực đại của s}. Ta có thể thấy {a} và $\mathcal{U} \notin T_a$ và T_a là một hệ Sperner trên $\mathcal{U}$ (hệ Sperner chính là siêu đồ thị đơn).

2.11. Định lý (J. Demetrovics – V.Đ. Thi, 1994)**:** Bài toán sau là NP-đầy đủ: Cho một sơ đồ quan hệ s = ($\mathcal{U}$, F) và hai thuộc tính a, b, quyết định xem có hay không phụ thuộc hàm cực đại A→{a} sao cho b∈A.

Chứng minh: Với b, ta chọn bất định tuỳ ý một tập con A của $\mathcal{U}$ sao cho b∈A. Vì thuật toán tính bao đóng của A có độ phức tạp tính toán đa thức và theo định nghĩa của phụ thuộc hàm cực đại, ta xác định được A∈T_a hay không. Rõ ràng rằng thuật toán này là bất định có độ phức tạp tính toán đa thức. Vậy bài toán này thuộc lớp NP.

Bây giờ ta cần chỉ ra rằng bài toán trên là NP-khó, có nghĩa là có một bài toán NP-đầy đủ chuyển về bài toán của ta nhờ một thuật toán có độ phức tạp tính toán đa thức. Có thể thấy rằng bài toán dưới đây về việc xác định thuộc tính cơ bản của sơ đồ quan hệ là NP-đầy đủ.

Cho sơ đồ quan hệ s = ($\mathcal{U}$, F) và thuộc tính a. Xác định có tồn tại hay không một khoá tối thiểu của s chứa a (a gọi là thuộc tính cơ bản của s).

Bài toán này được đưa về bài toán của ta nhờ một thuật toán có độ phức tạp tính toán đa thức như được chứng minh dưới đây.

Giả sử s' = ($\mathcal{P}$, F') là một sơ đồ quan hệ trên $\mathcal{P}$. Không mất tính chất tổng quát, ta giả thiết rằng $\mathcal{P}$ không là một khoá tối thiểu của s', có nghĩa là nếu A∈K_s thì A⊂$\mathcal{P}$. Vì việc tìm một khoá tối thiểu của một sơ đồ quan hệ cho trước được giải quyết bằng một thuật toán đa thức, ta có thể tìm một khoá tối thiểu C của s'. Bây giờ ta xây dựng sơ đồ quan hệ s = ($\mathcal{U}$, F) như sau:

$$\mathcal{U} = \mathcal{P} \cup \{a\}, \text{ ở đây } a \notin \mathcal{P} \text{ và } F = F' \cup C \to \{a\}.$$

Hiển nhiên s được xây dựng trong thời gian đa thức theo kích thước của $\mathcal{P}$ và F'. Rõ ràng C∈K_s. Trên cơ sở kiến trúc s và định nghĩa của khoá tối thiểu, ta thấy nếu A∈$K_{s'}$ thì A∈K_s. Ngược lại, nếu B là một khoá tối thiểu của s thì do C→{a}∈F, ta có a∉B. Mặt khác, do định nghĩa của khoá tối thiểu, ta có B∈$K_{s'}$. Như vậy ta có $K_{s'}$=K_s. Vì C∈K_s và a∉$\mathcal{U}$, nên nếu B→{a} là một phụ thuộc hàm cực đại của s thì B∈K_s. Có thể thấy rằng nếu A∈$K_{s'}$ thì A→{a}∈F^+. Phù hợp với định nghĩa của phụ thuộc hàm cực đại, ta có A→{a} là một phụ thuộc hàm cực đại của s. Do đó b là một thuộc tính cơ bản của s' khi và chỉ khi tồn tại một phụ thuộc hàm cực đại A→{a} của s để b∈A.

2.12. Định nghĩa: Bài toán A được gọi là co-NP-đầy đủ nếu bài toán phủ định của A là NP-đầy đủ.

Những khái niệm khoá của file dữ liệu và khoá của sơ đồ quan hệ đóng vai trò rất quan trọng trong việc xử lý dữ liệu. Chúng dùng để tìm kiếm các bản ghi và

nhờ có chúng người ta mới tìm cách tiến hành xử lý dữ liệu được. Dưới đây là một bài toán co-NP-đầy đủ liên quan đến việc so sánh giữa hai tập khoá của sơ đồ quan hệ và file dữ liệu.

Gottlob và Libkin đã chỉ ra rằng bài toán phần bù giới hạn các tập con (SDC-Subset delimiter complementarity) sau là co-NP-đầy đủ.

2.13. Định lý (G. Gottlob – L. Libkin, 1990)**:** Bài toán sau là co-NP-đầy đủ: Cho một tập hữu hạn T, hai họ P={P_1, …, P_n} và Q={Q_1, …, Q_m} các tập con của T. Kiểm tra xem với mọi A⊂T có tồn tại P_i để P_i⊂A hoặc có Q_j để A⊂Q_j, với 1≤i≤n, 1≤j≤m.

Gottlob và Libkin cũng chứng minh rằng nếu Q_1, …, Q_m là một hệ Sperner trên T thì bài toán trên vẫn là co-NP-đầy đủ.

Bài toán SDC này sẽ được chứng tỏ chuyển đa thức về bài toán dưới đây.

Ký hiệu L_r và L_s tương ứng là tập tất cả các khoá của quan hệ r và sơ đồ quan hệ s. Bài toán kiểm tra L_r⊂L_s hay không cũng là co-NP-đầy đủ.

2.14. Định lý (J. Demetrovics – V.Đ. Thi, 1993)**:** Bài toán sau là co-NP-đầy đủ: Cho quan hệ r và sơ đồ quan hệ s = ($\mathcal{U}$, F), kiểm tra xem L_r có là tập con của L_s hay không.

Chứng minh: Đối với mỗi A⊂$\mathcal{U}$, ta kiểm tra rằng A là hoặc không là một khoá của r bằng một thuật toán đa thức. Từ thuật toán tìm bao đóng A^+ và định nghĩa khoá của sơ đồ quan hệ, ta cũng có thể kiểm tra trong thời gian đa thức A là hoặc không là khoá của s. Do đó ta chọn tuỳ ý một tập con A⊂$\mathcal{U}$ sao cho A là khoá của r nhưng không là khoá của s. Như vậy, vấn đề của ta thuộc co-NP.

Xét bài toán SDC với tập hữu hạn T và hai họ P={P_1, …, P_n}, Q={Q_1, …, Q_m}, ở đây Q là một hệ Sperner trên T. Ký hiệu

$$P' = \{P_i \in P \mid \text{không tồn tại } P_j \text{ để } P_j \subset P_i,\ 1 \le i, j \le n\}.$$

Rõ ràng P' là tập các phần tử nhỏ nhất của P và P' là một hệ Sperner trên T. Từ P ta có thể tính P' trong thời gian đa thức theo |P| và |T|. Dễ thấy {T, P', Q} là một thể hiện tương đương của {T, P, Q}. Từ đó ta có thể giả thiết rằng P là một hệ Sperner trên T. Ta sẽ chứng minh rằng bài toán SDC được dẫn về bài toán của ta bằng một thuật toán thời gian đa thức.

Đặt $\mathcal{U}$ = T, s = ($\mathcal{U}$, F), ở đây F = {$P_1 \to \mathcal{U}$, …, $P_n \to \mathcal{U}$}.

Đặt M = {$Q_i \setminus \{a\}$ | i=1, 2, …, m và a∈$\mathcal{U}$} = {M_1, …, M_t}. Xây dựng quan hệ r = {h_0, h_1, …, h_t} như sau:

Với mỗi a∈$\mathcal{U}$, h_0(a)=0; h_i(a)=0 nếu a∈M_i và h_i(a)=i trong trường hợp ngược lại, với i=1, 2, …, t.

Rõ ràng r và s được xây dựng trong thời gian đa thức theo kích thước |T|, |P| và |Q|. Có thể thấy rằng F_r và s = ($\mathcal{U}$, F) là dạng chuẩn BCNF.

Do s là BCNF nên với mỗi $A \subset \mathcal{U}$, ta có $A^+=A$ hoặc $A^+=\mathcal{U}$. Từ định nghĩa của khoá, đối với mỗi khoá A của s đều có một P_i sao cho $P_i \subset A$, ở đây $1 \le i \le n$.

Có thể thấy rằng Q là tập phản khoá của r. Do F_r là BCNF nên với mỗi $A \subset \mathcal{U}$, $H_{Fr}(A)=\mathcal{U}$ hoặc $H_{Fr}(A)=A$, ở đây $H_{Fr}(A)=\{a \in \mathcal{U} \mid (A, \{a\}) \in F_r\}$. Từ định nghĩa phản khoá của r, ta thấy A là khoá của r khi và chỉ khi với mọi i=1, 2, …, m, $A \not\subset Q_i$.

Vậy $L_r \subset L_s$ khi và chỉ khi với mỗi $A \subset T$, với mỗi i=1, 2, …, m, $A \not\subset Q_i$ thì có một P_j để $P_j \subset A$. Từ đây ta thấy rằng bài toán SDC được dẫn về bài toán của ta bằng một thuật toán thời gian đa thức.

TÀI LIỆU THAM KHẢO

[1] Phan Đình Diệu, *Lý thuyết ôtômat và thuật toán*, NXB Đại học và Trung học chuyên nghiệp, Hà Nội, 1977.
[2] Đỗ Đức Giáo, Đặng Huy Ruận, *Văn phạm và ngôn ngữ hình thức*, NXB Khoa học và Kỹ thuật, Hà Nội, 1991.
[3] Đỗ Đức Giáo, *Toán rời rạc*, NXB Đại học Quốc Gia Hà Nội, Hà Nội, 2000.
[4] Lê Mạnh Thạnh, *Nhập môn ngôn ngữ hình thức và ôtômat*, NXB Giáo dục, Đà Nẵng, 1998.
[5] Vũ Đức Thi, *Thuật toán trong tin học*, NXB Khoa học và Kỹ thuật, Hà Nội, 1999.
[6] Bùi Minh Trí, *Tối ưu hoá tổ hợp*, NXB Khoa học và Kỹ thuật, Hà Nội, 2003.
[7] Phan Thị Tươi, *Trình Biên Dịch*, NXB Đại học Quốc Gia TP. Hồ Chí Minh, TP. Hồ Chí Minh, 2001.
[8] A.V. Aho, J.D. Ullman, *The theory of parsing, Translation and compiling*, Vol. 1, 2, Prentice-Hall, Englewood Cliffs, 1972.
[9] J.E. Hopcroft, J.D. Ullman, *Formal languages and their ralation to automata*, Addison Wesley, Reading Mass. London, 1969.
[10] J.E. Hopcroft, J.D. Ullman, *Introduction to formal language theory*, Addison Wesley, Reading Mass. London, 1979.
[11] K.H. Rosen, *Discrete mathematics and its applications*, Mc Graw-Hill, New York, 1994.
[12] A. Salomaa, *Formal languages*, Academic Press, New York, 1973.