



CHƯƠNG TRÌNH DỊCH

Bài 8: Phân tích cú pháp bằng thuật toán **bottom-up**



Nội dung

1. Ý tưởng & thuật toán
2. Ví dụ minh họa
3. Cài đặt bottom-up đơn giản
 - Cấu trúc một luật văn phạm
 - Cấu trúc một suy diễn trực tiếp
 - Máy phân tích: các hàm hỗ trợ
 - Máy phân tích: các hàm chính
4. Đánh giá về bottom-up
5. Bài tập



Phần 1

Ý tưởng & thuật toán



Bottom-up: ý tưởng

- Cho văn phạm G với các luật sinh:

$$S \rightarrow E + S \mid E \qquad E \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid (S)$$

- Xâu vào: $W = (1 + 2 + (3 + 4)) + 5$

- Thu gọn W thành S :

$$\begin{aligned} (1 + 2 + (3 + 4)) + 5 &\leftarrow (E + 2 + (3 + 4)) + 5 \leftarrow \\ (E + E + (3 + 4)) + 5 &\leftarrow (E + E + (E + 4)) + 5 \leftarrow \\ (E + E + (E + E)) + 5 &\leftarrow (E + E + (E + S)) + 5 \leftarrow \\ (E + E + (S)) + 5 &\leftarrow (E + E + E) + 5 \leftarrow \\ (E + E + S) + 5 &\leftarrow (E + S) + 5 \leftarrow \\ (S) + 5 &\leftarrow E + 5 \leftarrow E + E \leftarrow E + S \leftarrow S \end{aligned}$$



Bottom-up: mục tiêu & ý tưởng

- **Mục tiêu:** trong số nhiều suy dẫn dạng $S \Rightarrow^* w$, thuật toán sẽ tìm suy dẫn phải (?)
- **Ý tưởng chính:**
 - Thử sai và quay lui bằng năng lực tính toán của máy tính
 - Dò ngược quá trình suy dẫn $w \Leftarrow w_{n-1} \Leftarrow \dots \Leftarrow w_1 \Leftarrow S$ bằng kỹ thuật thu-gọn: tìm xem w_i có chứa vế phải của luật hay không, nếu có thì thay thế phần vế phải đó bằng vế trái tương ứng
 - Nếu một $w_i \neq S$ thì chắc chắn nó cần phải được thu-gọn, nếu w_i không chứa vế phải của luật nào đó thì nhánh thử sai này cần quay lui, ngược lại thì thu-gọn và thử tiếp



Bottom-up: thuật toán

1. $A = w$
2. Với chuỗi A đạt được trong quá trình lần ngược:
 - Nếu $A = "S"$:
 - Kết luận: quá trình tìm kiếm thành công
 - Lưu lại kết quả (chuỗi biến đổi từ đầu để được A)
 - Kết thúc ngay lập tức quá trình tìm kiếm
 - * Duyệt tất cả các luật sinh dạng $x \rightarrow \alpha$, nếu α là một chuỗi con trong A thì:
 - Áp dụng thu-gọn: thế α trong A bằng x , ta được A'
 - Thử bước 2 với chuỗi $A = A'$
 - Nếu không có phương án thu gọn nào thì quay lui



Phần 2

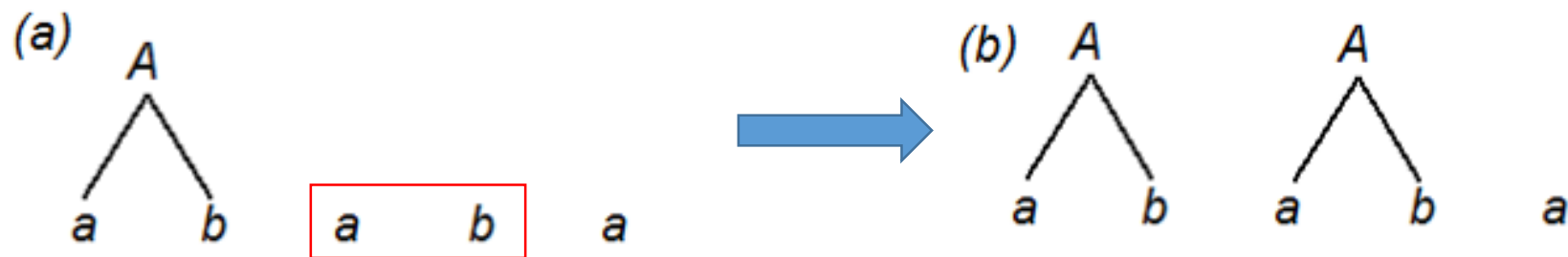
Ví dụ minh họa



Bottom-up: ví dụ

Cho tập luật $S \rightarrow AB$, $A \rightarrow ab$, $B \rightarrow aba$

Chỉ ra quá trình thu gọn chuỗi $w = ababa$

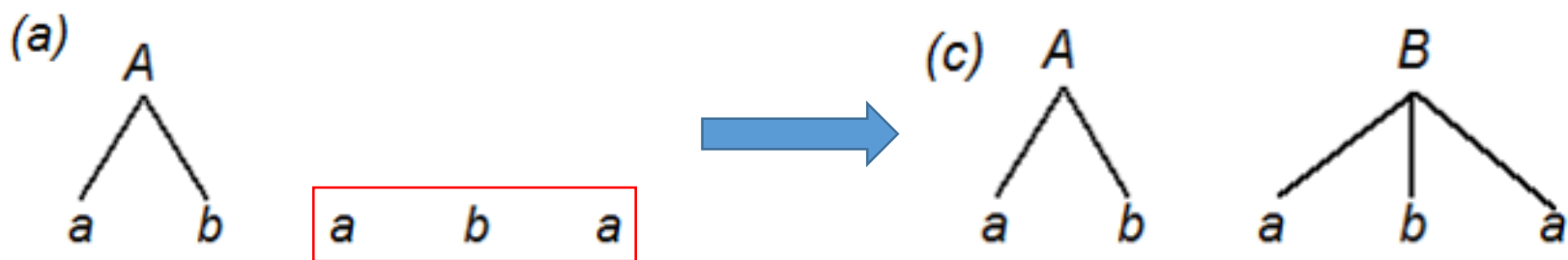


- ...
- Áp dụng luật $A \rightarrow ab$, thu gọn $ababa \Leftarrow Aaba$
- Chuỗi $Aaba$ có 2 phương án thu gọn: $Aaba \Leftarrow AAa$ và $Aaba \Leftarrow AB$



Bottom-up: ví dụ

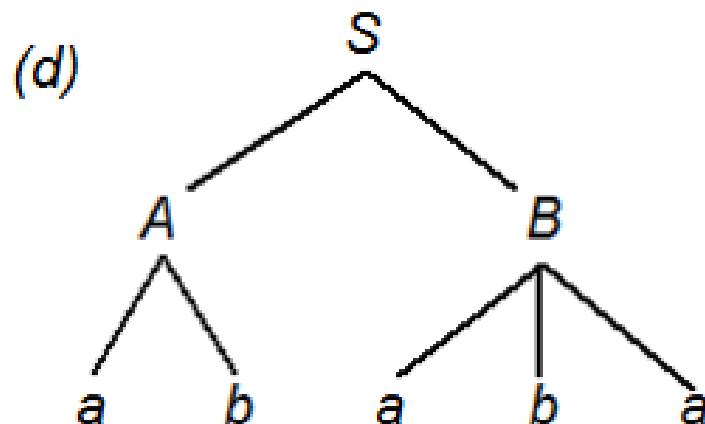
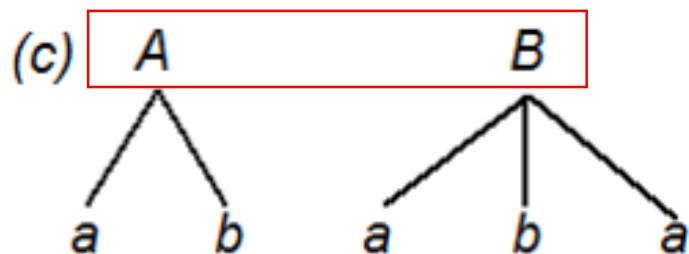
- Áp dụng luật $A \rightarrow ab$, thu gọn $Aaba \Leftarrow AAa$
- Đến đây nhánh này ngưng vì không thu gọn tiếp được nữa
- Áp dụng luật $B \rightarrow aba$, thu gọn $Aaba \Leftarrow AB$





Bottom-up: ví dụ

- Áp dụng luật $S \rightarrow AB$, thu gọn $AB \Leftarrow S$
- Đến đây điều kiện $A = "S"$ xảy ra:
 - Thuật toán dừng, kết luận thu gọn thành công
 - Lưu lại quá trình suy dẫn ngược



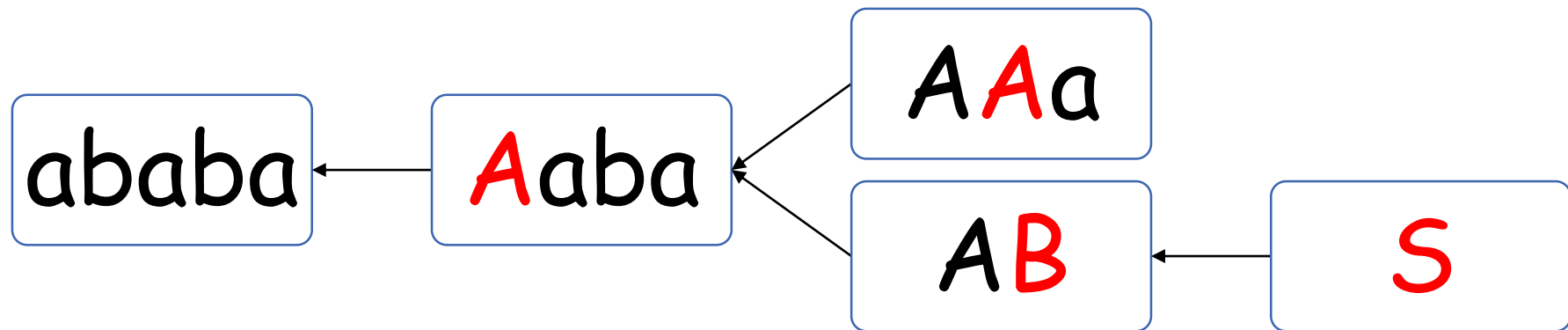


Bottom-up: ví dụ

Với tập luật $S \rightarrow AB$, $A \rightarrow ab$, $B \rightarrow aba$

Phân tích $W = ababa$

Cây rút gọn minh họa hoạt động của bottom-up:





Phần 3

Cài đặt bottom-up đơn giản



Cấu trúc một luật văn phạm

```
// Lớp chứa luật văn phạm, dạng left->right
```

```
class Rule {
```

```
    public string left, right;
```

```
    public Rule(string l, string r) {
```

```
        left = l; right = r;
```

```
    }
```

```
// chuyển đổi luật về dạng string (để in cho dễ nhìn)
```

```
public string ToFineString() {
```

```
    string s = left + " -->";
```

```
    for (int i = 0; i < right.Length; i++) s += " " + right[i];
```

```
    return s;
```

```
}
```

```
}
```

Cấu trúc một suy diễn trực tiếp



```
// Lớp chứa một bước áp dụng luật suy diễn  
// + ruleNumber: số thứ tự của luật sẽ được dùng  
// + position: vị trí sẽ áp dụng luật đó
```

```
class Step {  
    public int ruleNumber, position;  
    public Step(int r, int p) {  
        ruleNumber = r;  
        position = p;  
    }  
}
```



Máy phân tích: các hàm hỗ trợ

```
class PTBU {  
    public List<Rule> rules = new List<Rule>();  
    public List<Step> steps;  
    string word = null;  
    public void AddRule(string left, string right) {  
        rules.Add(new Rule(left, right));  
    }  
    public void PrintAllRules() {  
        Console.WriteLine("<bo luat van pham>");  
        foreach (Rule r in rules)  
            Console.WriteLine("  " + r.ToFineString());  
    }  
}
```



Máy phân tích: các hàm hỗ trợ

```
public void PrintSteps() {
    Console.WriteLine("Doan nhan thanh cong sau...");
    string w = word;
    foreach (Step s in steps) {
        string x = DoBackStep(w, s);
        ...
    }
}

string DoBackStep(string w, Step s) {
    string w1 = w.Substring(0, s.position);
    string w2 = w.Substring(s.position +
        rules[s.ruleNumber].right.Length);
    return w1 + rules[s.ruleNumber].left + w2;
}
```




Máy phân tích: các hàm chính

```
public bool Process(string x) {  
    steps = new List<Step>();  
    word = x;  
    return BottomUp(x);  
}
```

// áp dụng được luật k ở vị trí i của chuỗi w?

```
bool CanApplyHere(string w, int i, int k) {  
    string s = w.Substring(i);  
    if (s.Length > rules[k].right.Length)  
        s = s.Substring(0, rules[k].right.Length);  
    return (s == rules[k].right);  
}
```



Máy phân tích: các hàm chính

```
public bool BottomUp(string w) {  
    if ("S" == w) return true;  
    for (int i = 0; i < w.Length; i++)  
        for (int k = 0; k < rules.Count; k++)  
            if (CanApplyHere(w, i, k)) {  
                Step st = new Step(k, i);  
                steps.Add(st);  
                if (BottomUp(DoBackStep(w, st))) return true;  
                steps.RemoveAt(steps.Count - 1);  
            }  
    return false;  
}
```



Thử nghiệm

```
class Program {  
    public static void Main() {  
        PTBU parser = new PTBU();  
        parser.AddRule("S", "AB");  
        parser.AddRule("A", "ab");  
        parser.AddRule("B", "aba");  
        parser.PrintAllRules();  
        if (parser.Process("ababa"))  
            parser.PrintSteps();  
    }  
}
```



Phần 4

Đánh giá về bottom-up



Đánh giá về bottom-up

- Đặc trưng:
 - Dễ hiểu: cài đặt đơn giản
 - Chậm: duyệt toàn bộ, không có các bước cắt nhánh
 - Không vạn năng: không làm việc với văn phạm có suy dẫn rỗng ($A \rightarrow \varepsilon$) hoặc đệ quy ($A \Rightarrow^+ A$)
 - Không dễ loại bỏ những kết quả trùng lặp (trường hợp muốn tìm mọi phương án suy dẫn)
- Ý tưởng cải tiến:
 - * Quy hoạch động: sử dụng lại những kết quả duyệt cũ
 - * Cắt nhánh sớm: dựa trên đặc trưng của một số luật để loại bỏ các phương án không có tương lai



Phần 5

Bài tập



Bài tập

1. Chỉ ra quá trình thu gọn theo phân tích bottom-up của chuỗi **raid** thuộc văn phạm G sau:
 - $S \rightarrow r X d \mid r Z d$
 - $X \rightarrow o a \mid e a$
 - $Z \rightarrow a i$
2. Chỉ ra quá trình thu gọn theo phân tích bottom-up của chuỗi **(a=(b+a))** thuộc văn phạm G sau:
 - $S \rightarrow B$
 - $B \rightarrow R \mid (B)$
 - $R \rightarrow E = E$
 - $E \rightarrow a \mid b \mid (E + E)$



Bài tập

3. Chỉ ra quá trình thu gọn theo phân tích bottom-up cho chuỗi $(5+7)^*3$ thuộc văn phạm G sau:
- $E \rightarrow E + T \mid T$
 - $T \rightarrow T * F \mid F$
 - $F \rightarrow (E) \mid \text{số}$
4. Chỉ ra quá trình thu gọn theo phân tích bottom-up của chuỗi **true and not false** với tập luật:
- $E \rightarrow E \text{ and } T \mid T$
 - $T \rightarrow T \text{ or } F \mid F$
 - $F \rightarrow \text{not } F \mid (E) \mid \text{true} \mid \text{false}$



Bài tập

5. Chỉ ra quá trình thu gọn theo phân tích bottom-up của chuỗi **abcbcd** thuộc văn phạm G có tập luật:
- $S \rightarrow a A \mid b A$
 - $A \rightarrow c A \mid b A \mid d$
6. Có thể thực hiện phân tích bottom-up chuỗi **aaab** thuộc văn phạm G dưới đây hay không? Chỉ ra phương án xử lý nếu có.
- $S \rightarrow A B$
 - $A \rightarrow a A \mid \varepsilon$
 - $B \rightarrow b \mid b B$



Bài tập (lập trình)

1. Hãy chỉnh sửa thuật toán top-down và bottom-up để chúng có thể chỉ ra mọi phương án suy dẫn từ kí hiệu bắt đầu S thành chuỗi đích w
2. * Mã nguồn minh họa cả hai thuật toán top-down và bottom-up đều dựa trên đệ quy, hãy chuyển đổi chúng thành dạng không đệ quy (gợi ý: sử dụng một stack lưu lại trạng thái của các chuỗi trung gian trong quá trình thử-sai các luật sinh)
3. Hãy xây dựng thuật toán chuyển đổi từ suy dẫn trả về bởi thuật toán top-down (bottom-up) thành cây phân tích cú pháp tương ứng



Bài tập (lập trình)

4. * Hãy điều chỉnh thuật toán top-down (bottom-up) để chúng trả về mọi cây phân tích cú pháp khác nhau (dùng cho văn phạm có nhập nhằng)
5. Nếu văn phạm G đệ quy phải, thì thuật toán top-down hay bottom-up sẽ trả về kết quả nhanh hơn trong các tính huống:
 - Chuỗi w không thuộc văn phạm G
 - Chuỗi w có nhiều cây phân tích cú pháp