PHENIKAA UNIVERSITY
Realizing your potential

CHƯƠNG TRÌNH DỊCH

Chương 4. Phân tích cú pháp
Thuật toán Bottom-up

TS. Phạm Văn Cảnh
Khoa Công nghệ thông tin

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

Nội dung

1. Ý tưởng & thuật toán
2. Cài đặt Bottom-up đơn giản
3. Đánh giá về bottom-up
4. Bài tập

1. Ý tưởng và thuật toán

- Cho văn phạm G với các luật sinh:

$$S \rightarrow E + S \mid E$$

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

- Xâu vào: $W = (1 + 2 + (3 + 4)) + 5$
- Thu gọn W thành S:

$$(1 + 2 + (3 + 4)) + 5 \leftarrow (E + 2 + (3 + 4)) + 5 \leftarrow$$
$$(E + E + (3 + 4)) + 5 \leftarrow (E + E + (E + 4)) + 5 \leftarrow$$
$$(E + E + (E + E)) + 5 \leftarrow (E + E + (E + S)) + 5 \leftarrow$$
$$(E + E + (S)) + 5 \leftarrow (E + E + E) + 5 \leftarrow$$
$$(E + E + S) + 5 \leftarrow (E + S) + 5 \leftarrow$$
$$(S) + 5 \leftarrow E + 5 \leftarrow E + E \leftarrow E + S \leftarrow S$$

1. Ý tưởng và thuật toán

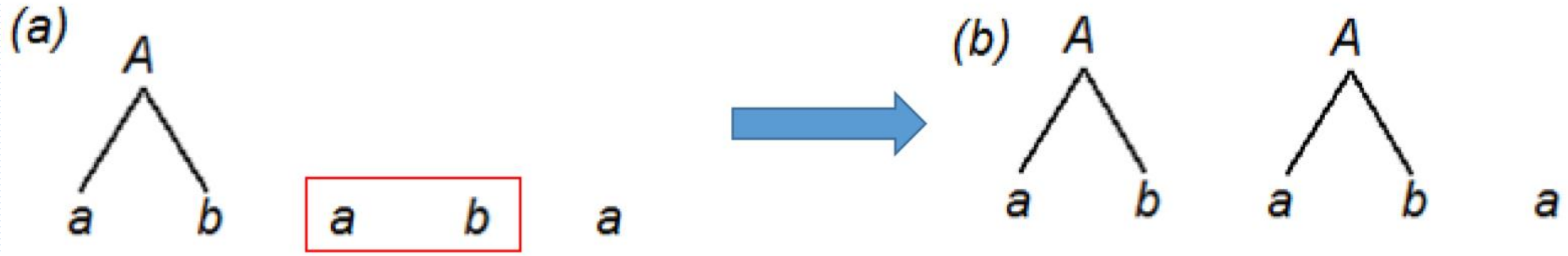
- **Mục tiêu**: trong số nhiều suy dẫn dạng $S \rightarrow^* w$, thuật toán sẽ tìm suy dẫn phải (?)
- **Ý tưởng chính**:
  - Thử sai và quay lui bằng năng lực tính toán của máy tính
  - Dò ngược quá trình suy dẫn $w \leftarrow w_{n-1} \leftarrow \cdots \leftarrow w_1 \leftarrow S$ bằng kĩ thuật thu-gọn: 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.

Thuật toán Top-down

- Bước 1: A = w
- Bước 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

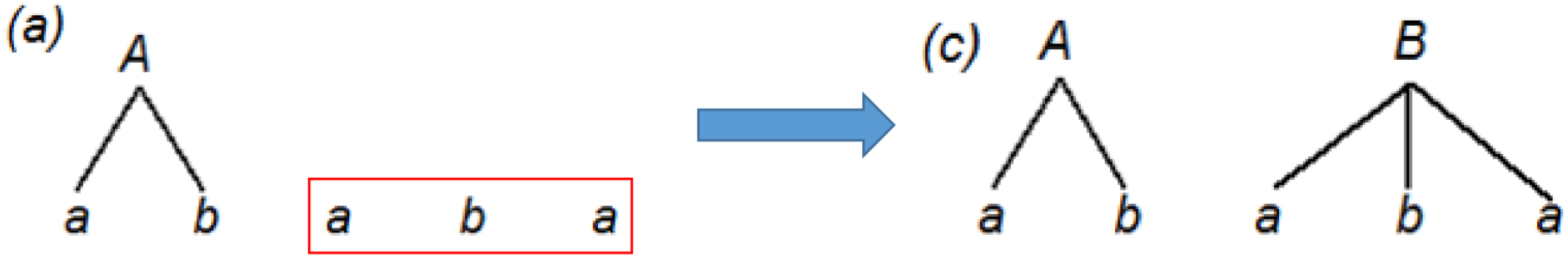
Thuật toán Top-down

- Ví dụ: Cho tập luật S → AB, A → ab, B → 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 $Aaba \leftarrow AB$



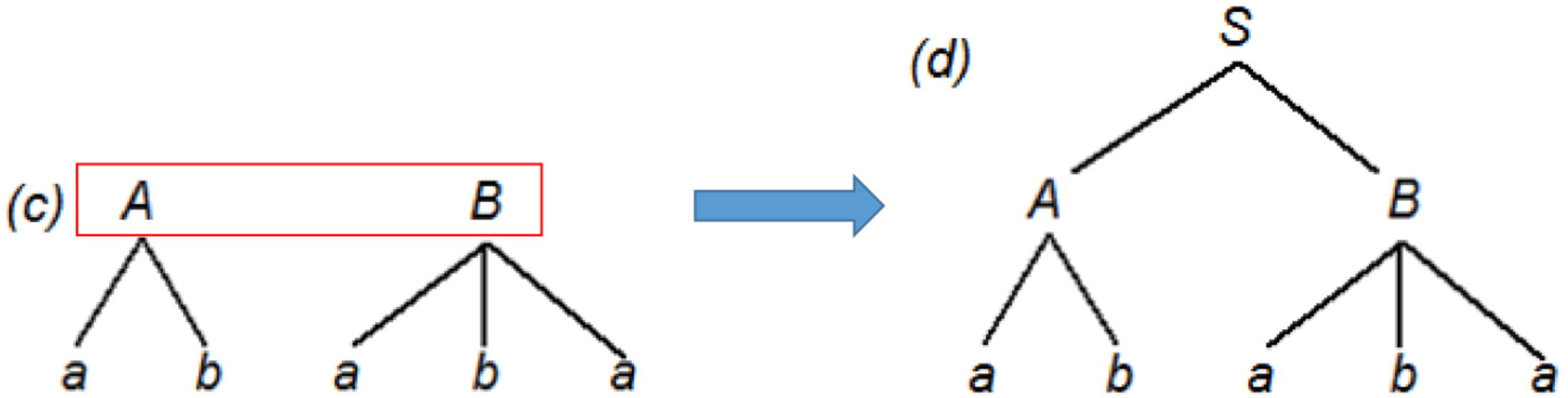
Thuật toán Top-down

- 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 $Aaba \rightarrow AB$



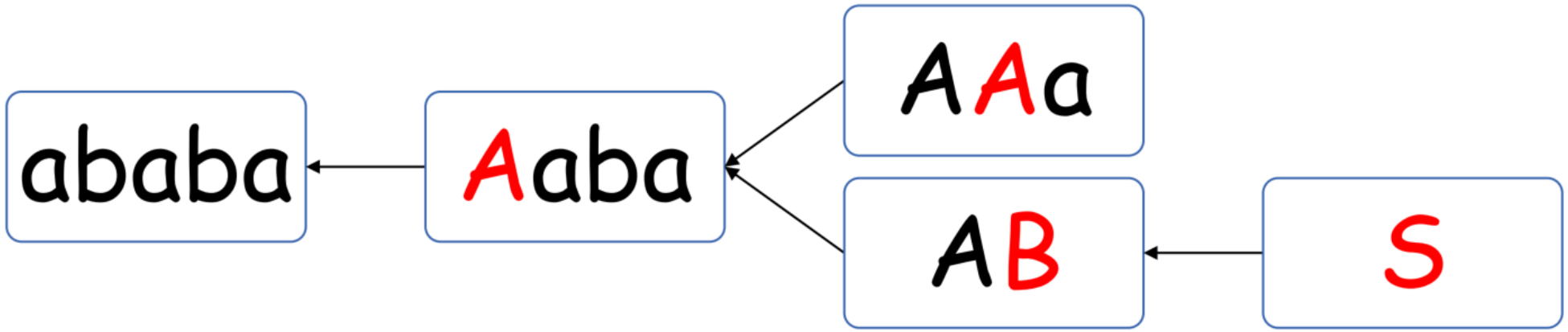
Thuật toán Top-down

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



Thuật toán bottom-up

- Cho tập luật S → AB, A → ab, B → aba. Chỉ ra quá trình thu gọn chuỗi w = ababa
- Cây rút gọn của quá trình là:



Cài đặt thuật toán bottom-up: Cấu trúc 1 luật

```
// 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;
  }
}
```

Các hàm hỗ trợ

```
// 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;
    }
}
```

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());
    }
```

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;
}
```

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);
}
```

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;
}
```

Các hàm chính

```
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();
    }
}
```

3. Đánh giá thuật toán bottom-up

PHENIKAA UNIVERSITY
Realizing your potential

- 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 ($A \rightarrow \epsilon$) 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

4. Bài tập

1. Chỉ ra quá trình thực hiện phân tích bottom-up của chuỗi **raid** thuộc văn phạm G có tập luật:

S → r X d | r Z d

X → o a | e a

Z → a i

2. Chỉ ra quá trình thực hiện phân tích bottom-up của chuỗi **((x+y)=(y+x))** thuộc văn phạm G có tập luật:

S → B

B → R | (B)

R → E = E

E → x | y | (E + E)

4. Bài tập

3. Có thể áp dụng thuật toán phân tích bottom-up cho chuỗi **(a+a)*a** thuộc văn phạm G dưới đây hay không? Chỉ ra quá trình thực hiện nếu có thể

E → E + T | T

T → T * F | F

F → (E) | a

4. Tương tự câu trên, chỉ ra quá trình phân tích topdown của chuỗi **true and not false** với tập luật:

E → E and T | T

T → T or F | F

F → not F | (E) | true | false

4. Bài tập

5. Chỉ ra quá trình thực hiện phân tích top-down của chuỗi **abbcbd** thuộc văn phạm G có tập luật:

S → a A | b A

A → c A | b A | d

6. Chỉ ra quá trình thực hiện phân tích top-down của chuỗi **aaab** thuộc văn phạm G có tập luật:

S → A B

A → a A | ϵ

B → b | b B