CHƯƠNG TRÌNH DỊCH

Bài 7: Phân tích cú pháp bằng thuật toán top-down

Nội dung

1. Ý tưởng & thuật toán
2. Ví dụ minh họa
3. Cài đặt top-down 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
   - Thử nghiệm
4. Đánh giá về top-down
5. Bài tập

Phần 1

Ý tưởng & thuật toán

Top-down: 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
- Suy dẫn trái từ S thành W như sau:

  $S \Rightarrow E + S \Rightarrow (S) + S \Rightarrow (E + S) + S \Rightarrow (1 + S) + S$

  $\Rightarrow (1 + E + S) + S \Rightarrow (1 + 2 + S) + S$

  $\Rightarrow (1 + 2 + E) + S \Rightarrow (1 + 2 + (S)) + S$

  $\Rightarrow (1 + 2 + (E + S)) + S \Rightarrow (1 + 2 + (3 + S)) + S$

  $\Rightarrow (1 + 2 + (3 + E)) + S \Rightarrow (1 + 2 + (3 + 4)) + S$

  $\Rightarrow (1 + 2 + (3 + 4)) + E \Rightarrow (1 + 2 + (3 + 4)) + 5$

Top-down: ý tưởng

- Xét quá trình suy dẫn $S \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \ldots \Rightarrow W$
- W_i luôn chứa ít nhất một non-terminal
- Xét X là non-terminal trái nhất của W_i:
  - W không chứa non-terminal nên X sẽ phải "biến mất"
  - Cách làm "biến mất" X chỉ có thể do sử dụng luật văn phạm mà vế trái là X
- Nhận xét: trước sau gì X cũng sẽ "biến mất" bởi một luật văn phạm có dạng $X \rightarrow \alpha$
  - Top-down sử dụng năng lực tính toán của máy tính để thử các khả năng có thể (phương pháp thử-sai-quay-lui)

Top-down: ý tưởng

- Dò tìm quá trình suy dẫn $S \Rightarrow W_1 \Rightarrow \ldots \Rightarrow W$:
  - Với W_i, tìm non-terminal X
  - Tìm luật dạng $X \rightarrow \alpha$, áp dụng để suy diễn $W_i \Rightarrow W_{i+1}$
  - Dừng nếu $W_{i+1} = W$ (tìm được phương án suy dẫn)
  - Thử với W_{i+1} hoặc quay lui nếu đã xét hết phương án
- Đặc điểm của Top-down:
  - Nếu W_i có chứa nhiều non-terminal thì chỉ cần thử với non-terminal trái nhất
  - Trong số nhiều suy dẫn dạng $S \Rightarrow^* W$, thuật toán sẽ tìm suy dẫn trái

Top-down: thuật toán

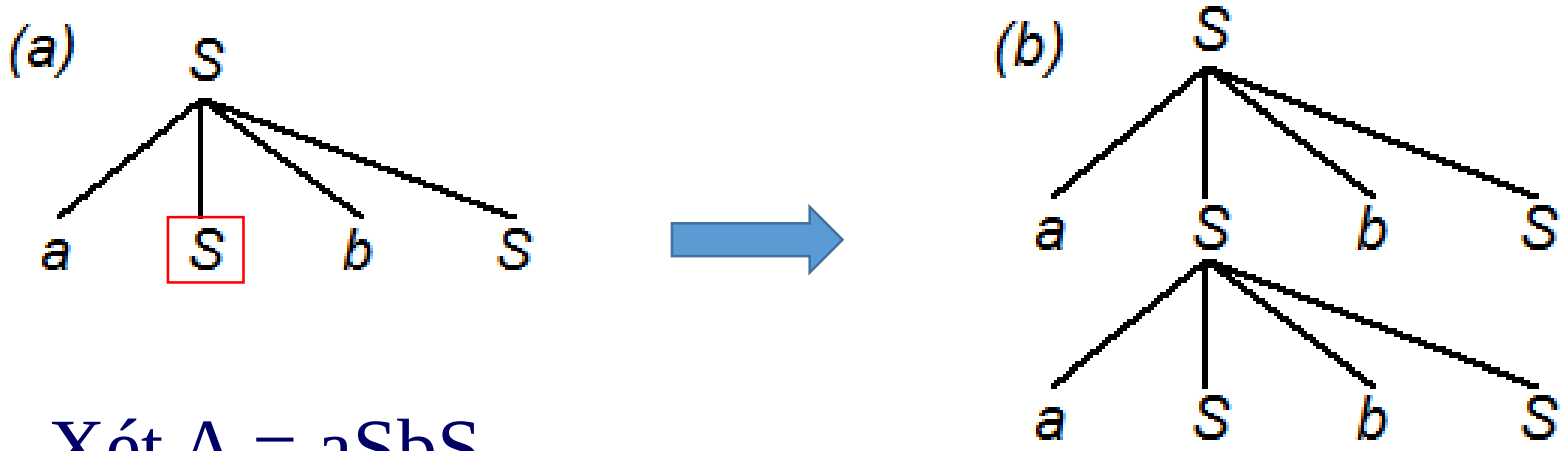
1. A = S
2. Với một chuỗi A đạt được trong quá trình suy dẫn:
   - Nếu A = W:
     - Kết luận: quá trình tìm kiếm thành công
     - Lưu lại quá trình biến đổi từ đầu để được A
     - Kết thúc ngay lập tức quá trình tìm kiếm
   - Nếu A ≠ W: tìm kí hiệu trung gian trái nhất X
   - Không tìm được X thì dừng, quay lui lại hàm gọi
   - Duyệt tất cả các luật sinh dạng X → α
     - Áp dụng luật đó trên A (ở vị trí X), ta được A'
     - Thử bước 2 với chuỗi A = A'

Phần 2

Ví dụ minh họa

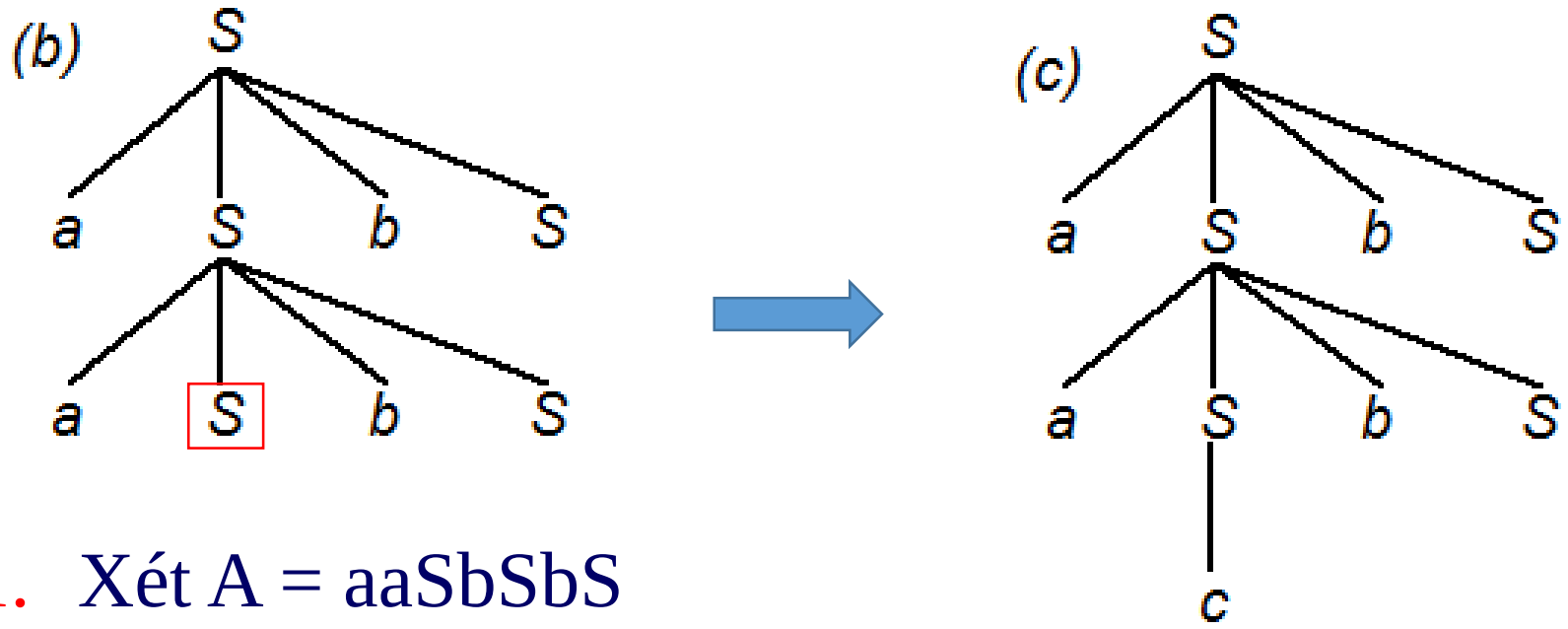
Top-down: ví dụ

Phân tích W = aacbc với tập luật S → aSbS | aS | c



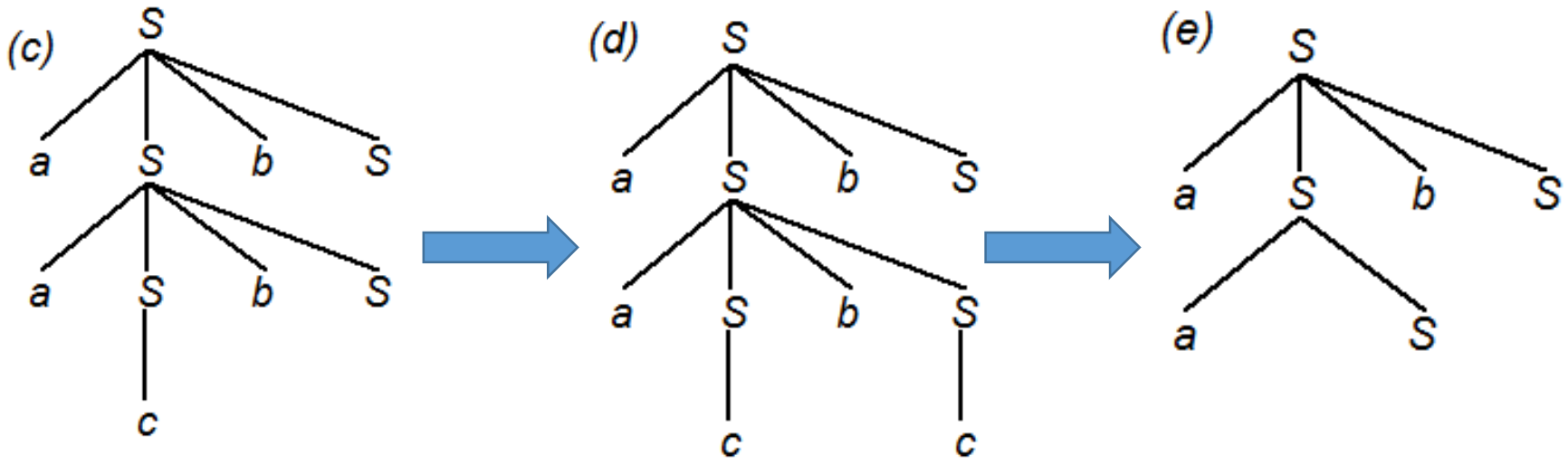
1. Xét A = aSbS
2. Tìm được kí hiệu S thứ 2 trong A là non-terminal
3. Thử áp dụng luật S → aSbS được A’ = aaSbSbS

Top-down: ví dụ



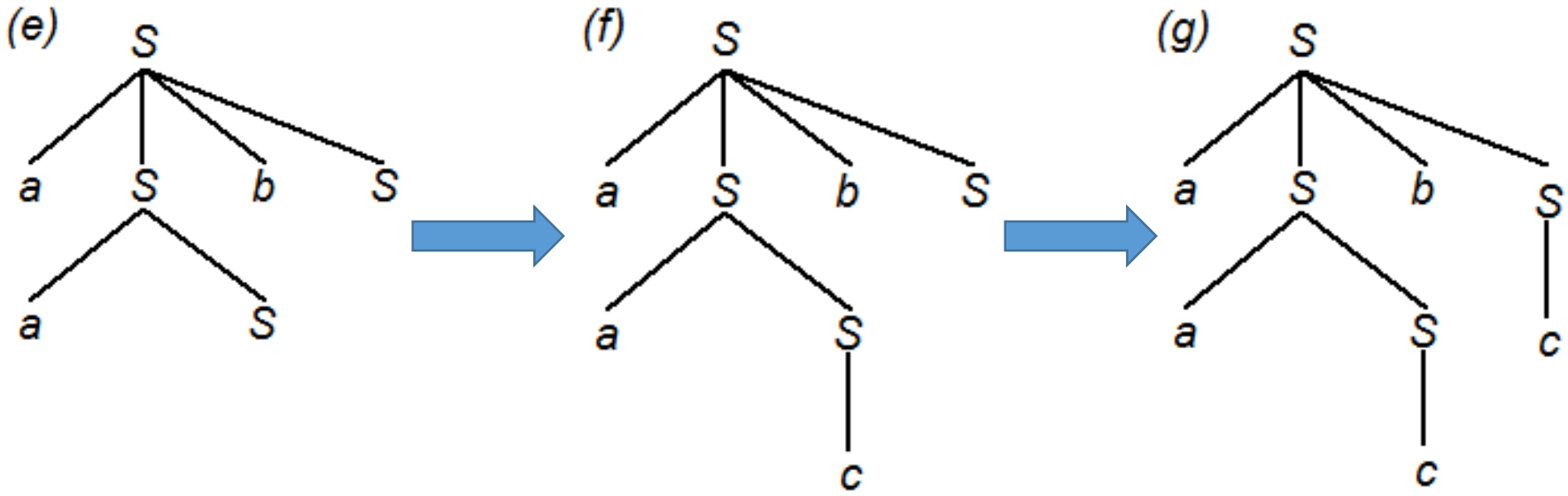
1. Xét A = aaSbSbS
2. Tìm được kí hiệu S thứ 3 trong A là non-terminal
   1. Thử áp dụng luật S → aSbS được A’ = aaaSbSbSbS
   2. Thử áp dụng luật S → aS được A’ = aaaSbSbS
   3. Thử áp dụng luật S → c được A’ = aacbSbS

Top-down: ví dụ



- Quá trình thử sai kết luận rằng A = aSbS không thể áp dụng luật S → aSbS
- Quay lui về đến tình huống ban đầu ở hình (a)
- Thử phương án tiếp theo S → aS, được A' = aaSbS

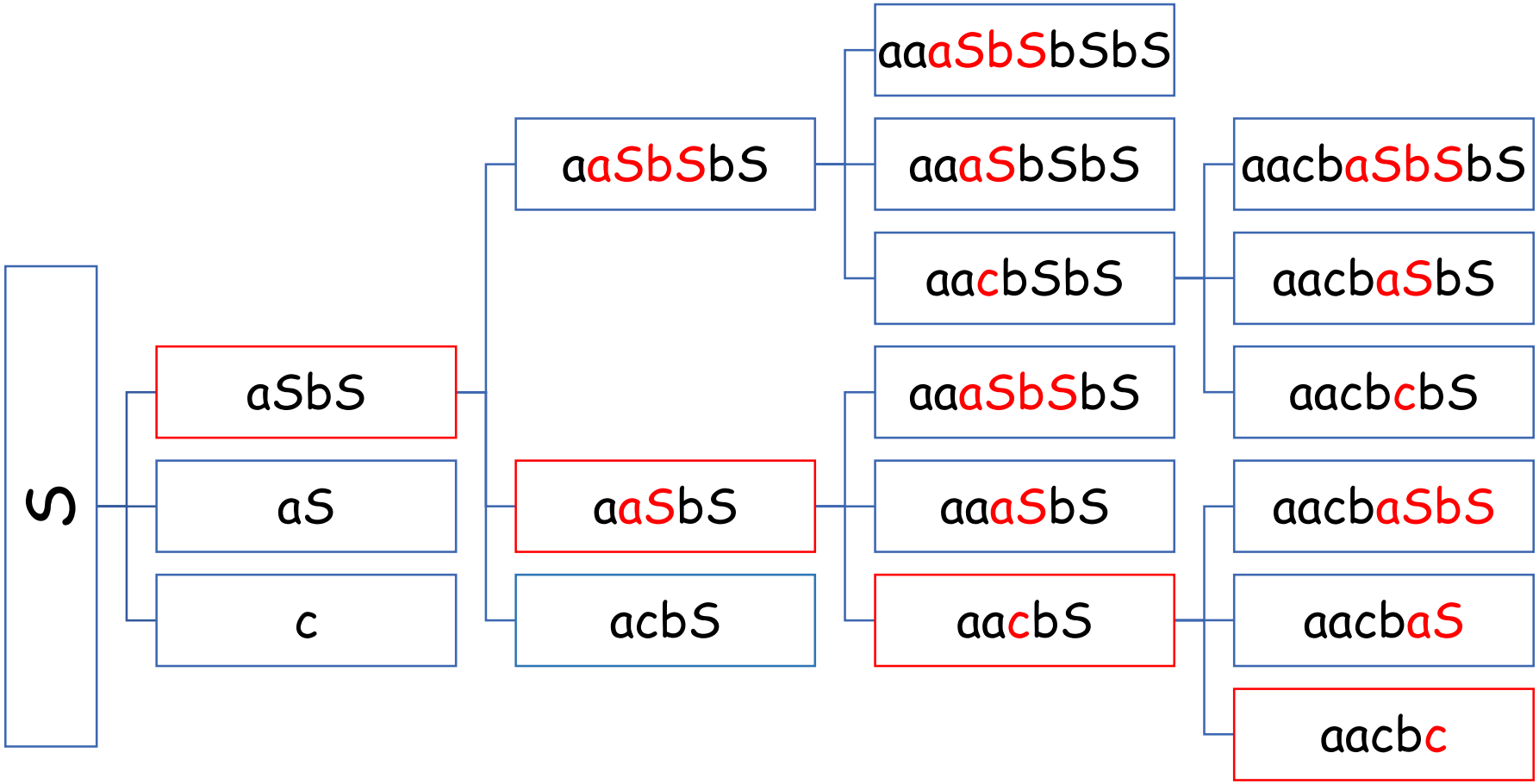
Top-down: ví dụ



- Quá trình thử sai tiếp tục và cuối cùng dừng ở phương án được thể hiện ở hình (g)
- Khi nhận được chuỗi A = W = aacbc, ngay lập tức thuật toán dừng và trả về quá trình áp dụng luật

Top-down: ví dụ

Phân tích W = aacbc với tập luật S → aSbS | aS | c



Phần 3

Cài đặt top-down 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 PTTD {
  public List<Rule> rules = new List<Rule>();    // bộ luật
  public List<Step> steps;                       // các bước suy diễn
  string w = null;                               // chuỗi W đích
  // thêm luật left --> right vào tập luật
  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 = "S";
  foreach (Step s in steps) {
    string w0 = DoStep(w, s);
    Console.WriteLine("  {0} => {1} (vi tri...
    w = w0;
  }
}
string DoStep(string w, Step s) {
  string w1 = w.Substring(0, s.position);
  string w2 = w.Substring(s.position + 1);
  return w1 + rules[s.ruleNumber].right + w2;
}
```

Máy phân tích: các hàm chính

```
public bool Process(string x) {
  steps = new List<Step>();
  w = x;
  return Try("S");
}
// tìm vị trí non-terminal trái nhất trong s
// trả về -1 nếu không tìm được
public int FindNonterminal(string s) {
  for (int i = 0; i < s.Length; i++) {
    if (i >= w.Length) return i;
    if (s[i] != w[i]) return i;
  }
  return -1;
}
```

Máy phân tích: các hàm chính

```
// hàm thử-sai-quay-lui với chuỗi s
public bool Try(string s) {
  if (s == w) return true;
  int n = FindNonterminal(s);
  if (n != -1)
    for (int i = 0; i < rules.Count; i++)
      if (rules[i].left[0] == s[n]) {
        Step st = new Step(i, n);
        steps.Add(st);
        if (Try(DoStep(s, st))) return true;
        steps.RemoveAt(steps.Count - 1);
      }
  return false;
}
```

Thử nghiệm

```
class Program {
  public static void Main() {
    PTTD parser = new PTTD();
    // nạp thử bộ luật
    parser.AddRule("S", "B");
    parser.AddRule("B", "R");
    parser.AddRule("B", "(B)");
    parser.AddRule("R", "E=E");
    ...
    parser.PrintAllRules();
    if (parser.Process("(x=(x+y))"))
      parser.PrintSteps();
  }
}
```

Phần 4

Đánh giá về top-down

Đánh giá về top-down

- Thuật toán đơn giản, sử dụng sức mạnh của máy tính để tìm kiếm lời giải
- Thuật toán dạng thử-sai-quay-lui, không cắt nhánh, độ phức tạp tính toán là hàm mũ (~ chậm)
- Thuật toán không vạn năng, không làm việc được với các văn phạm có đệ quy trái
  - Lý do: vì không có cắt nhánh phù hợp, dẫn đến việc đi mãi theo chiều sâu mà không quay lui

Câu hỏi: có thể sửa đổi thuật toán như thế nào để làm việc được với văn phạm có đệ quy trái?

Cải tiến top-down thế nào?

- Tăng tính vạn năng của thuật toán:
  - Xử lý tình huống đệ quy trái bằng ràng buộc phù hợp
  - Biến đổi văn phạm trước khi bắt đầu thử-sai-quay-lui
- Tăng tốc độ tính toán:
  - Tập trung vào việc cài đặt cắt nhánh (nhiều ý tưởng)
    - Cắt nhánh khi trong A có terminal không có trong w
    - Cắt nhánh khi số terminal trong A nhiều hơn trong w
  - Tính trước các bước không có "cơ hội về đích" để loại bỏ bớt những tình huống thử-sai không cần thiết
  - Sử dụng lại những kết quả đã duyệt cũ

Phần 5

Bài tập

Bài tập

1. Chỉ ra quá trình thực hiện phân tích top-down 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 top-down 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)

Bài tập

3. Có thể áp dụng thuật toán phân tích top-down 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 top-down 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

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