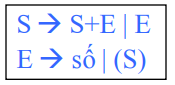
**Phân tích từ dưới lên (Bottom-up parsing)**

1 Đặc điểm

* Có kĩ thuật phân tích mạnh hơn
* Văn phạm lớp LR có khả năng mô tả mạnh hơn văn phạm lớp LL, có thể mô tả văn phạm đệ quy trái
* Dễ dàng mô tả các ngôn ngữ lập trình thông thường
* Bộ phân tích cú pháp **Gạt – Thu gọn :**
  + Xây dự cây suy dẫn phải
  + Tự động xây dụng bộ phận cú pháp
  + Phát hiện lỗi ngay khi xuất hiện
  + Cho phép phục hồi khi lỗi xảy ra

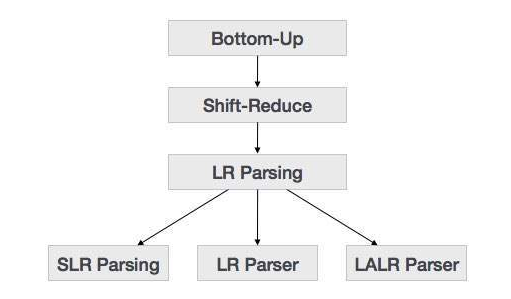
So sánh đặc điểm 2 loại phân tích cú pháp

Ví dụ 

Phân tích (1+2+(3+4))+5

|  |  |
| --- | --- |
| Phân tích từ trên xuống | Phân tích từ dưới lên |
| * Xây dựng cây phân tích cú pháp từ trên xuống. * Suy dẫn trái. * Toàn bộ cây phía trên một kí hiệu được sinh ra (Xác định gốc của cây phân tích trước, sau đó di chuyển xuống các cây con cho đến khi tìm thấy kí tự kết thúc). * Phải có khả năng đoán trước được các node sinh ra(thiết lập trước trình duyệt của cây cú pháp). | * Xây dựng cây phân tích cp từ dưới lên * Suy dẫn phải. * Cây suy dẫn được xây dựng ngược lại   + Bắt đầu từ kí hiệu kết thúc.   + Kết thúc tại kí tự bắt đầu. * Phân tích từ dưới lên có nhiều thông tin hơn khi phân tích. * Phân tích dưới lên không cân sinh ra toàn bộ cây suy dẫn trong quá trình phần tích(duyệt theo thứ tự cảu cây cú pháp). |

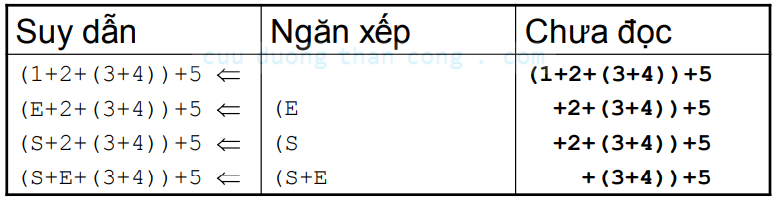
Mô hinh phân tích của phân tích từ dưới lên



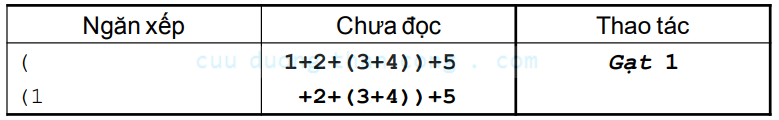
Phân tích **Gạt – thu gọn (Shift - Reduce):**

* Phân tích bằng một dãy thao tác: gạt và thu gọn
* Mỗi thời điểm, trạng thái của bộ phân tích là ngăn xếp các kí hiệu kết thúc và không kết thúc
* Cấu hình tại mỗi thời điểm gồm :

**ngăn xếp + xâu các kí hiệu chưa đọc**

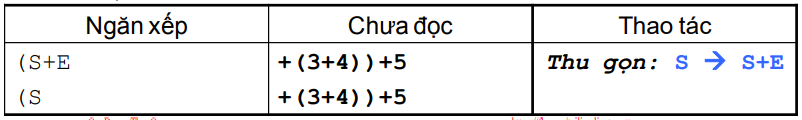


**Gạt :** Đọc và đưa một kí tự kết thúc của xâu vào stack

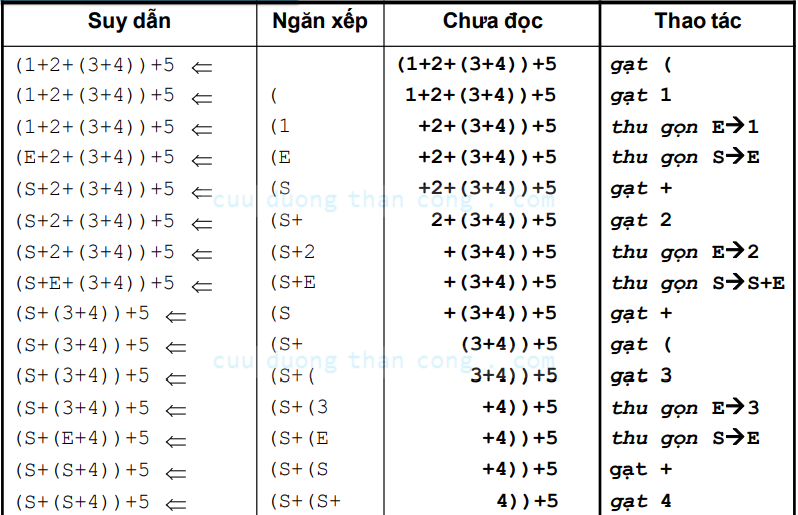


**Thu gọn:** Thay thế một xâu **Y** ở đình của ngăn xếp băng kí hiệu không kết thúc **X** với **X -> Y**

(pop **Y**, push **X**)



Sau các thao tác



Các thao tác xử lý

* Sử lý sản xuất rỗng

X -> e

* Thu gọn

S - > E hay S - > S + E

* Tại mỗi thời điểm, từ cấu hình

<S- ngăn xếp, a – từ tố nhìn trước >

* Xác định
  + **Gạt** a, ngăn xếp trở thành <Sa>
  + **Thu gọn X-Y,** nếu S = aY,   
    ngăn xếp trở thành <aX>
* Nếu X = aY, cân lự chọn **Gạt** a hoặc **thu gọn** X -> Ydựa vào tiền tố a
  + Với mỗi khả năng thu gọn X -> Y có một a
  + Cần tìm cách đánh dấu các khả năng thu gọn

Trnaj thái bộ phân tích gạt

**Mục tiêu:** Xác định khả năng thu gọn hợp lệ tại từng thời điểm.

**Ý tưởng:** gộp các khả năng có thể có của tiền tố a thành trạng thái của bộ phân tích.

**Bộ phân tích cú pháp LR(LR parsing)**

Trình phân tích cú pháp LR là trình phân tích cú pháp không đệ quy, Shift – reduce, từ dưới lên.

SLR(1) - Trình phân tích cú pháp LR đơn giản:

* Hoat động trên lớp ngữ pháp nhỏ nhất.
* Rất ít trang thái, do đó bảng rất nhỏ.
* Triển khai đơn giản và nhanh chóng.

LR(1) Trình phân tích cú pháp LR:

* Làm viêc trên bộ ngữ pháp LR(1) hoàn chỉnh.
* Tạo bảng lớn và ső lượng lớn các trạng thái.
* Triên khai chậm.

LALR(1) Trình phân tích cú pháp LR nhìn trước:

* Hoạt động trên kích thước trung bình của ngữ pháp.
* Số lượng trang thái giống như trong SLR(1).

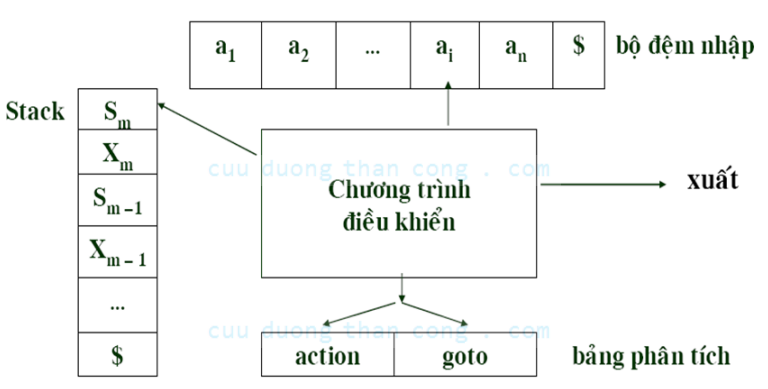
Phân tích cú pháp LR(k):

* L (left - to - right): Duyệt chuỗi nhập từ trái sang phải.
* R (rightmost derivation): Xây dựng chuỗi dẫn xuất phải nhất đảo ngược.
* k : Số lượng ký hiệu nhập được xét tại mỗi thời điểm dùng để đưa ra quyết định phân tích. Khi không đề cập đến k, hiểu ngầm là k = 1.

Các tính chất của phương pháp phân tích LR(k):

* Bộ phân tích LR có thể nhận dạng được cấu trúc cú pháp của các ngôn ngữ lập trình do văn phạm phi ngữ cảnh tạo ra.
* Phương pháp LR là phương pháp tổng quát nhất của phương pháp phân tích gạt và thu gọn, không bị quay lui.
* Lớp văn phạm có thể dùng phương pháp LR là một lớp rộng lớn hơn lớp văn phạm có thể sử dụng phương pháp dự đoán.
* Bộ phân tích cú pháp LR cũng có thể xác định lỗi cú pháp nhanh ngay trong khi duyệt dòng nhập từ trái sang phải

Cấu tạo



* **Stack** được dùng để chứa chuỗi ký hiệu có dạng s0X1 s1X2…Xmsm, với sm nằm trên đỉnh stack, Xi được gọi là ký hiệu văn phạm, si là trạng thái tóm tắt thông tin bên dưới stack. Cặp(si, Xi) sẽ xác định một trị được lưu chứa trong bảng phân tích.
* **Cấu hình (configuration)** của một bộ phân tích cú pháp LR là một cặp, trong đó thành phần đầu là nội dung của Stack, phần sau là chuỗi nhập chưa phân tích:

(s0X1 s1X2 s2 ... Xmsm, aiai+1... an$)

* **Bảng phân tích** : gồm 2 hàm **action** và **goto**
  + **action[sm, ai ]** có thể có một trong 4 giá trị :   
    1. shift s: đẩy s, trong đó s là một trạng thái.   
    2. reduce A→ β: thu gọn bằng luật sinh A→ β.   
    3. accept: Chấp nhận   
    4. error: Báo lỗi
  + **Goto** lấy 2 tham số là một trạng thái và một ký hiệu văn phạm, nó sinh ra một trạng thái.

**Giải thật LR**

* **Nhập**: chuỗi nhập w, bảng phân tích action goto của văn phạm G (giả sử đã có).
* **Xuất**: nếu w thuộc L (G), nó tạo ra sự phân tích từ dưới lên. Ngược lại, bộ phân tích sẽ báo lỗi.

|  |
| --- |
| **token = next\_token()**  **repeat forever**  **s = top of stack**  **if action[s, token] = “shift si” then**  **PUSH token**  **PUSH si**  **token = next\_token()**  **else if action[s, token] = “reduce A::= β“ then**  **POP 2 \* |β| symbols**  **s = top of stack**  **PUSH A**  **PUSH goto[s,A]**  **else if action[s, token] = “accept” then**  **return**  **else**  **error()** |