**CHƯƠNG TRÌNH DỊCH**

# **Lý Thuyết**

1. **Chương trình dịch là gì? Các giai đoạn của chương trình dịch**

* Là một chương trình thực hiện việc chuyện đổi một chương trình hay 1 đoạn chương trình con từ ngôn ngữ nguồn một cách tương đương sang ngôn ngữ đích.
* Ngôn ngữ nguồn là ngôn ngữ lập trình bậc cao (Pascal, Java, C…)
* Ngôn ngữ đích là ngôn ngữ lập trình bậc thấp như Assembly, hoặc ngôn ngữ máy.
* là phần mềm hệ thống chuyển đổi đoạn văn viết trong ngôn ngữ A sang đoạn văn tương đương viết trong ngôn ngữ B
* 2 pha:
* Kỳ trước – font-end còn gọi là giai đoạn phân tích
* Kỳ sau – back-end còn gọi là giai đoạn tổng hợp
* 6 giai đoạn:
* Phân tích từ vựng
* Phân tích cú pháp
* Phân tích ngữ nghĩa
* Sinh mã trung gian
* Tối ưu mã trung gian
* Sinh mã đích

1. **Định nghĩa biên dịch và thông dịch**

* Trong hệ thống biên dịch, toàn bộ chương trình ngồn được chuyển sang chương trình đích ở dạng mã máy
* Khí thực hiện chương trình chạy độc lập trên máy mà không cần hệ thống biên dịch nữa
* Ưu điểm: chương chình chạy nhanh
* Nhược điểm: việc thiết kế cài đặt phức tạp

1. **Phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa**

* Phân tích từ vựng: sẽ đọc chương trình nguồn từ trái sang phải để tách thành các thẻ từ (token)

Vd: position := initial + rate \* 60 sẽ tách thành các token nào?

* Từ tố định danh: position, initial, rate
* Từ tố phép toán quan hệ: :=
* Từ tố phép toán cộng: +
* Từ tố phép toán sao: \*
* Từ tố số nguyên: 60
* Phân tích cú pháp: thực hiện công việc nhóm các thẻ từ của chương trình nguồn thành các ngữ đoạn văn phạm, mà sau đó sẽ được trình biên dịch tổng hợp ra thành phẩm
* Thông thường các ngữ đoạn văn phạm này được biểu diễn bằng cây phân tích cú pháp với:

+ Ngôn ngữ được đặc tả bởi các luật sinh

+ Phân tích cú pháp dựa vào luật sinh để xây dựng cây phân tích cú pháp

+ Cấu trúc phân cấp của 1 chương trình thường được diễn tả bởi quy luật đệ qui

* Phân tích ngữ nghĩa: sẽ được thực hiện việc kiểm tra xem chương trình nguồn có chứa lỗi về ngữ nghĩa hay không và tập hợp thông tin về kiểu cho giai đoạn sinh mã về sau. Một phần quan trọng trong giai đọan phân tích ngữ nghiã là kiểm tra kiểu và ép kiểu chuyển đổi.

1. **Nêu định nghĩa văn phạm. Ý nghĩa các ký hiệu**

* Văn phạm G là một hệ thống (Σ, Δ, S, P) trong đó:

+ Σ là tập hữu hạn các ký hiệu kết thúc (terminal)

+ Δ là tập hữu hạn các ký hiệu không kết thúc (nonterminal)

* Còn gọi là ký hiệu trung gian hay biến
* Σ ∩ Δ = ∅

+ S ∈ Δ gọi là ký hiệu khởi đầu (initial) , đôi khi còn gọi là kí hiệu mục tiêu.

+ P là tập hữu hạn các luật sinh

* Thường được viết là α → β
* Chuỗi α phải có ít nhất một ký hiệu không kết thúc

Ví dụ: Cho văn phạm sau:

G = ({S, A, B}, {a, b}, S, P)

S🡪aAS| bBS|

A🡪aaA| b

B🡪bbB| a

1. **Phân loại văn phạm**

* Lớp 0: Văn phạm tự do – Văn phạm ngữ cấu

+ Không có ràng buộc gì về luật sinh

+ Tương đương với lớp các ngôn ngữ loại đệ quy đếm

+ Được đoán nhận bởi máy Turing

* Lớp 1: Văn phạm cảm ngữ cảnh

+ Các luật sinh α→β phải thỏa mãn điều kiện | α | ≤ | β |

+ Tương đương với lớp các ngôn ngữ cảm ngữ cảnh

+ Được đoán nhận bởi automat tuyến tính giới nội

* Lớp 2: Văn phạm phi ngữ cảnh

+ Các luật sinh phải có dạng A → α trong đó A ∈ Δ (ký hiệu không kết thúc). Nói 1 cách khác, vế trái của luật sinh chỉ có 1 ký hiệu.

+ Tương đương với lớp các ngôn ngữ phi ngữ cảnh

+ Đoán nhận bởi automat đẩy xuống

* Lớp 3: Văn phạm chính quy

+ Các luật sinh chỉ có thể ở 1 trong 2 loại:

A → a, A → Ba trong đó A, B ∈ Δ, a ∈ Σ (ký hiệu kết thúc)

A → a, A → aB trong đó A, B ∈ Δ, a ∈ Σ

+ Sinh ra các ngôn ngữ chính quy

+ Đoán nhận bởi automat hữu hạn

1. **Định nghĩa otomat? Các otomat hữu hạn trạng thái (FSA) có vai trò gì trong ctd?**

* Otomat, dịch nghĩa là máy tự động, là thiết bị có thể tự thực hiện công việc mà không cần sự can thiệp của con người. Nó hoạt động dựa trên 1 số quy tắc và dựa vào các quy tắc này con người lập trình cho nó hoạt động theo ý muốn của mình.
* Cung cấp kiến thức nền tảng cho việc xây dựng các ngôn ngữ lập trình, các trình dịch.
* Dùng văn phạm để định nghĩa các ngôn ngữ lập trình
* Dùng accepter để định nghĩa một vài thành phần của ngôn ngữ lập trình. Xây dựng các bộ phân tích từ vựng, phân tích cú pháp cho ngôn ngữ lập trình.
* Automat hữu hạn dùng để đoán nhận lớp ngôn ngữ chính qui

1. **Đồ thị chuyển**

* Là phương pháp thường sử dụng để mô tả 1 cách trực quan sơ đồ hoạt động của các automat hữu hạn.
* Trạng thái:

+ Trạng thái kết thúc là vòng tròn kép

+ Trạng thái kết thúc có đánh dấu \*: Ký tự cuối cùng không thuộc vào từ tố được đoán nhận.

1. **Biểu thức chính quy**

* Cho 1 bảng chữ cái = {a1, a2, …, an} khi đó biểu thức chính qui được định nghĩa đệ qui:
* và a với a là biểu thức chính qui trên biểu diễn ngôn ngữ và {a}
* Nếu r và s là hai biểu thức chính qui biểu diễn các ngôn ngữ chính qui R và S trên

+ r + s là một biểu thức chính quy trên biểu diễn ngôn ngữ R S

+ r.s là một biểu thức chính quy trên biểu diễn ngôn ngữ R.S

+ r\* hay (s\*) là một biểu thức chính quy trên biểu diễn ngôn ngữ R\* (hay S\*)

* Không còn biểu thức chính qui nào khác trên nằm ngoài định nghĩa trên.

Ví dụ: Cho ∑ = {a, b}

* Biểu thức chính quy a | b đặc tả {a, b}
* Biểu thức chính quy (a | b) (a | b) đặc tả tập hợp {aa, ab, ba, bb}. Tập hợp này có thể được đặc tả bởi biểu thức chính quy tương đương sau: aa | ab | ba | bb.
* Biểu thức chính quy a\* đặc tả {ε, a, aa, aaa, ...}
* Biểu thức chính quy (a | b) \* đặc tả {a, b, aa, bb, ...}. Tập này có thể đặc tả bởi (a\*b\*) \*.
* Biểu thức chính quy a | a\* b đặc tả {a, b, ab, aab, ...} Hai biểu thức chính quy cùng đặc tả một tập hợp ta nói rằng chúng tương đương và viết r = s.

1. **Văn phạm chính qui là gì?**

* Văn phạm G = (Σ, Δ, S, P) mà qui tắc của nó chỉ có dạng:

+ A🡪aB, A🡪a (Văn phạm chính qui phải) hoặc

+ A🡪Ba, A🡪a (Văn phạm chính qui trái) – đệ qui trái

Trong đó:

+ G được gọi là văn phạm chính qui

+ Ngôn ngữ được sinh bởi G là ngôn ngữ chính qui

1. **Văn phạm phi ngữ cảnh**

Gồm 4 thành phần: G = (Σ, Δ, S, P) trong đó qui tắc sinh đều có dạng:

A🡪w| A Δ, w (ΣΔ)\*

Khi đó: G được gọi là văn phạm phi ngữ cảnh

Ngôn ngữ được sinh ra gọi là ngôn ngữ phi ngữ cảnh.

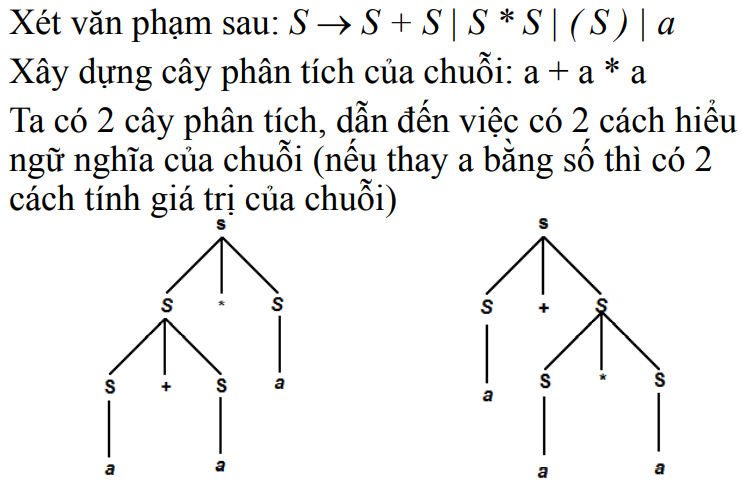
* Văn phạm phi ngữ cảnh sử dụng trong việc biểu diễn và phân tích cú pháp.

1. **Văn phạm mơ hồ**

Cho văn phạm phi ngữ cảnh: G = (Σ, Δ, S, P) và chuỗi:

w (Σ\*\ {}

* Nếu tồn tại chuỗi w là kết quả của 2 cây suy dẫn khác nhau trong G, khi đó G được gọi là văn phạm nhập nhằng. Ngược lại G được gọi là đơn nghĩa.
* tạo ra nhiều hơn 1 cây phân tích cú pháp cho cùng 1 chuỗi nhập thì gọi là Văn phạm nhập nhằng hay văn phạm mơ hồ



1. **Biểu diễn từ tố bằng biểu thức chính quy**

* Các ký hiệu quy ước:

+ |: hoặc

+ (): nhóm các thành phần

+ \* lặp đi lặp lại không hoặc nhiều lần

+ +: lặp lại 1 hoặc nhiều lần

+ ?: lặp lại không hoặc một lần

1. **Dẫn xuất của văn phạm đệ qui trái, phải**

* Một văn phạm là đệ qui trái nếu ó có 1 ký hiệu chưa kết thúc A sao cho có 1 dẫn xuất A =>+A, với là 1 chuỗi nào đó.

1. **Văn phạm tương đương**

* Cho văn phạm G với ký hiệu S bắt đầu. Ta dùng quan hệ =>+ để định nghĩa L(G) một ngôn ngữ được sinh ra bởi G.
* Chuỗi L(G) có thể chỉ chứa 1 ký hiệu kết thúc của G. Chuỗi các ký hiệu kết thúc w thuộc L(G) nếu và chỉ nếu S=>+w, chuỗi w được gọi là 1 câu của G.
* Một ngôn ngữ được sinh ra bởi 1 văn phạm gọi là ngôn ngữ phi ngữ cảnh.
* Nếu 2 văn phạm cùng sinh ra cùng một ngôn ngữ thì chúng được gọi là hai văn phạm tương đương.

1. **Cây phân tích cú pháp**

* Một cách hình thức, cho 1 văn phạm phi ngữ cảnh thì cây phân tích cú pháp có tính chất:

+ Nút gốc có nhãn là ký hiệu bắt đầu

+ Mỗi nút có nhãn là 1 ký hiệu kết thúc hoặc epsilon

+ Mỗi nút trong có nhãn là ký hiệu chưa kết thúc

+ Nếu A là 1 ký hiệu chưa kết thúc được dùng làm nhán cho 1 nút trong nào đó X1 … Xn là nhãn của các con của nó theo thứ tự từ trái sang phải thì A🡪X1X2 … Xn là 1 luật sinh. Ở đây, X1, … Xn có thể là ký hiệu kết thúc hoặc chưa kết thúc. Đặc biệt, nếu A🡪 thì nút có nhãn A có thể có 1 con có nhãn epsilon

1. **Đầu ra của bộ phân tích từ vựng là gì?**

* Tập các từ tố (tokens)
* Phân tích từ vựng giai đoạn thứ mấy: Thứ nhất

1. **Bộ phân tích cú pháp Shift-reduce**

* Cố gắng xây dựng 1 cây phân tích cú pháp cho chuỗi nhập bắt đầu từ nút lá và đi lên hướng về nút gốc. Đây có thể xem là 1 quá trình thu gọn 1 chuỗi w thành 1 ký hiệu bắt đầu của văn phạm. Tại mỗi bước thu gọn, mỗi chuỗi con cụ thể đối sánh được với vế phải của 1 luật sinh nào đó thì chuỗi con này được thay thế bởi ký hiệu vế trái của luật sinh đó. Và nếu chuỗi co được chọn đúng tại mỗi bước, mỗi dẫn xuất phải đảo ngược sẽ được xây dựng.
* Vd: cho văn phạm

S🡪aABe

A🡪Abc| b

B🡪d

Câu abbcde có thể thu gọn thành S theo các bước sau:

abbcde

aAbcde

aAde

aABe

S

Thực chấp đây là 1 dẫn xuất phải nhất đảo ngược như sau:

S =>rm aABe =>rm aAde =>rm aAbcde =>rm abbcde

* Dẫn xuất phải nhất là chuỗi các thay thế ký hiệu chưa kết thúc phải nhất.

1. **Phương pháp shift reduce là phương pháp gì?**

* Là phương pháp phân tích thu gọn

1. **Trong 1 trình biên dịch, giai đoạn phân tích từ vựng thực hiện như nào?**

* Đọc chương trình nguồn từ trái sang phải để tách ra thành các thẻ từ token

1. **Trong 1 trình biên dịch, giai đoạn phân tích cú pháp cho cùng 1 chuỗi nhập thì được** **gọi là văn phạm gì?**
2. **Giai đoạn cuối cùng của 1 trình biên dịch có tên gọi là gì? Nó thực hiện công việc gì?**

* Sinh mã đích
* tạo ra mã đích (thường là dạng mã máy hoặc mã assembly) từ các mã TAC hoặc SSA đã được tối ưu

1. **Định nghĩa First**

* Giả sử là 1 chuỗi các kí hiệu văn phạm, First () là tập hợp các ký hiệu kết thúc mà nó bắt đầu 1 chuỗi dẫn xuất trừ

1. **Trình bày các tính First**

* Nếu X là kí hiệu kết thúc thì FIRST(X) là {X}
* Nếu X → ε là một luật sinh thì thêm ε vào FIRST(X)
* Nếu X → Y1Y2Y3 ...Yk là một luật sinh thì:

+ Thêm tất cả các ký hiệu kết thúc khác ε của FIRST (Y1) vào FIRST(X)

+ Nếu ε ∈ FIRST (Y1) thì tiếp tục thêm vào FIRST(X) tất cả các ký hiệu kết thúc khác ε của FIRST (Y2)

+ Nếu ε ∈ FIRST (Y1) ∩ FIRST (Y2) thì thêm tất cả các ký hiệu kết thúc khác ε ∈ FIRST (Y3)

+ Tiếp tục như vậy cho tới Yk

+ Thêm ε vào FIRST(X) nếu ε ∈ ∩­­i=1🡪k FIRST (Yi)

1. **Định nghĩa Follow**

* Follow(A): Với A là ký hiệu của kết thúc
* Là tập hợp các ký hiệu kết thúc a mà nó xuất hiện ngay sau A (bên phải của A) trong 1 dạng câu nào đó. Tức là 1 tập ký hiệu kết thúc a, sao cho tồn tại 1 dẫn xuất dạng S=>\* Aa

1. **Trình bày cách tính Follow**

* Tính FOLLOW (A): áp dụng các quy tắc sau cho đến khi không thể thêm gì vào mọi tập FOLLOW được nữa

+ Ðặt $ vào follow(S), trong đó S là ký hiệu bắt đầu của văn phạm và $ là ký hiệu kết thúc chuỗi nhập

+ Nếu có một luật sinh A→ αBβ thì thêm mọi phần tử khác ε của FIRST(β)vào trong FOLLOW(B)

+ Nếu có luật sinh A→ αB hoặc A→ αBβ mà ε ∈ FIRST(β) thì thêm tất cả các phần tử trong FOLLOW(A) vào FOLLOW(B)

1. **Thế nào là văn phạm tăng cường (Augmented Grammar)**

* G là văn phạm với ký hiệu bắt đầu S, ta thêm 1 ký hiệu bắt đầu mới S’, và luật sinh S’🡪S để được văn phạm mới G’ => văn phạm tăng cường

1. **Mô tả thuật toán phân tích cú pháp bottom-up? Thuật toán phân tích thành công khi nào?**

* 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 → α, 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 phân tích thành công khi gặp một lỗi hoặc Stack chứa ký hiệu bắt đầu và bộ đệm input rỗng.

1. **Thuật toán bottom-up được xây dựng dựa trên ý tưởng nào?**

* Ý tưởng chính:

Thử sai và quay lui bằng năng lực tính toán của máy tính

+ Dò ngược quá trình suy dẫn w <= wn-1 <= … <= w1 <= S bằng kĩ thuật thu-gọn: tìm xem wi có chứa vế phải của luật hay không, nếu có thì thay thế phần vế phải đó bằng vế trái tương ứng

+ Nếu một wi S thì chắc chắn nó cần phải được thu-gọn, nếu wi không chứa vế phải của luật nào đó thì nhánh thử sai này cần quay lui, ngược lại thì thu-gọn và thử tiếp

# **Bài tập**

1. Biểu thức chính quy? Văn phạm
2. Tìm ký hiệu kết thúc, k kết thúc
3. Tìm các mục

Luật sinh A → BCDEFXY có thể tạo thành:

+) A → \*BCDEFXY

+) A → B\*CDEFXY

+) A → BC\*DEFXY

+) A → BCD\*EFXY

+) A → BCDE\*FXY

+) A → BCDEF\*XY

+) A → BCDEFX\*Y

+) A → BCDEFXY\*

* Tạo thành 8 mục

1. Tìm quá trình sinh ra 1 câu khi cho 1 văn phạm

Cho văn phạm G gồm có các luật sinh: E->E+E; E->E\*E; E->(E); E->id. Hãy viết dẫn xuất sinh ra xâu: id+id\*id

E→E+E

E→id => E→id+E

E→E\*E => E→id+E\*E

E→id => E→id+id\*E

E→id => E→id+id\*id

1. Tính follow
2. Tính first
3. Tính closure, tính goto
4. Ngôn ngữ hình thức

**BG**

**Câu 23: Khi thực hiện phân tích ngữ nghĩa cho câu lệnh position = initial + rate \* 10.5 thì trong chương trình nguồn của nó?**

* **position, initial, rate phải khai báo là các biến số thực**

**Câu 18: Văn phạm với các luật sinh E->EAE; E->(E); E->-E; E->id; A->+ Có thể sinh ra chuỗi nhập nào?**

* –(id+id)

=> Làm nào để sinh ra câu đấy

E 🡪 -E và E 🡪 (E) => E 🡪 -(E)

E🡪EAE => E🡪-(EAE)

E 🡪 id và A 🡪 + => E🡪-(id+id)

**Câu 6<DE>: Văn phạm gồm các luật sinh S🡪AA; A🡪aa; A🡪bb mô tả ngôn ngữ nào? Đối xưng của** aa **và** bb

* {aaaa, aabb, bbaa, bbbb}

BG

(a/b)2n (a/b)2n n thuộc N – tập số tự nhiên

**Câu 7<DE>: Biểu thức chính quy (x/y)(x/y) kí hiệu bởi tập?**

* B: {xx, xy, yx, yy}

BG

(x/ y) (x/ y) -> Bao giờ cũng có 2 kí tự, ở đây là hoặc

* xx, xy, yx, yy

**Câu 9<DE>: Biểu thức chính quy (x/y) kí hiệu bởi tập?**

* A: {x, y}

**Câu 20<TB>: Văn phạm gồm các luật sinh S**🡪**bA; S**🡪**aB; A**🡪**a B**🡪**b A**🡪**aS B**🡪**bS; A**🡪**bAA B**🡪**aBB sinh ra được chuỗi nào?**

C: aabbab

BG

S->aB

B->aBB => S -> aaBB

B->b, B->bS => S -> aabbS

S -> aB => S -> aabbaB

B -> b => S -> aabbab

* Có bao nhiêu loại từ tố:

+ Từ tố số nguyên

+ Từ số số thực

+ Từ tố định danh: max, a, b, ….

+ Từ tố từ khóa: printf, int

+ Từ tố phép toán (quan hệ =, cộng, sao)

+ Từ tố xâu: “max của 2 số %d**” -> luôn trong “”**

**11. Xét văn phạm tăng cường**

E’ 🡪 E

E 🡪 E+T| T

T 🡪 T \* F| F

F 🡪 (E)| id

BG

Nếu I là tập hợp chỉ gồm văn phạm {E’ 🡪 .E} được thêm vào closure(I) gồm:

E’ 🡪 .E (Luật 1)

E 🡪 .E + T (Luật 2)

E 🡪 .T (Luật 2)

T 🡪 .T \* F (Luật 2)

T 🡪 .F (Luật 2)

F 🡪 .(E) (Luật 2)

F 🡪 .id (Luật 2)

**12. Tính goto**

VD1: I = {E’ 🡪 E., E 🡪 E. + T}

* Tính goto (I, +)

BG

Ta có: I’ = {E’ 🡪 E., E 🡪 E. + T}

* Goto (I, +) = closure(I’)

closure(I’) bao gồm các mục:

+) E 🡪 E + .T (Luật 1)

+) T 🡪 .T \* F (Luật 2)

+) T 🡪 .F (Luật 2)

+) F 🡪 .(E) (Luật 2)

+) F 🡪 .id (Luật 2)

**VD2: Xét văn phạm tăng cương của văn phạm sau:**

E’ 🡪 E

E 🡪 E + T| T

T 🡪 T\*F| F

F 🡪 (E)| id

Nếu I là tập chỉ gồm văn phạm {E’ 🡪 .E}. Xây dựng họ tập mục C của văn phạm:

BG

Closure({E’🡪.E}):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I0 = E’🡪.E  E🡪.E + T  E🡪.T  T🡪.T\*F  T🡪.F  F🡪.(E)  F🡪.id | Goto(I0, E)  I1 = E’🡪E.  E🡪E. +T  Goto(I0, T)  I2 = E🡪T.  T🡪T. \*F  Goto(I0, F)  I3 = T🡪 F. | Goto(I0, ()  I4 = F 🡪 (.E)  E🡪.E + T  E🡪.T  T🡪.T\*F  T🡪.F  F🡪.(E)  F🡪.id | Goto(I0, id)  I5 = F🡪id.  Goto(I1, +)  I6 = E🡪E +. T  T🡪.T\*F  T🡪.F  F🡪.(E)  F🡪.id | Goto(I2, \*)  I7 = T🡪T\*.F  F🡪.(E)  F🡪.id |
| Goto(I4, E)  I8 = F 🡪 (E.)  E🡪.E + T | Goto(I6, T)  I9 = E🡪E +T.  T🡪.T\*F | Goto(I7, F)  I10 = T🡪T\*F. | Goto(I8, ))  I11 = F 🡪 (E). |  |

13. Bảng phân tích cú pháp SLR

