

NGÔN NGỮ' và PHƯƠNG PHÁP DỊCH

Phạm Đăng Hải

haipd@soict.hust.edu.vn

Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Sự cần thiết của ngôn ngữ lập trình bậc cao



- Nhiều loại máy tính
 - Mỗi loại nhiều kiểu
 - Mỗi kiểu có ngôn ngữ máy riêng
 - Ngôn ngữ máy là dãy nhị phân
 - Dùng ngôn ngữ máy
 - Không phải dịch
 - Phức tạp
 - Không khả chuyển
- Cần ngôn ngữ
 - Độc lập với máy
 - Gần với ngữ tự nhiên
- Ví dụ: C, Pascal, basic..

Ngôn ngữ
bậc cao

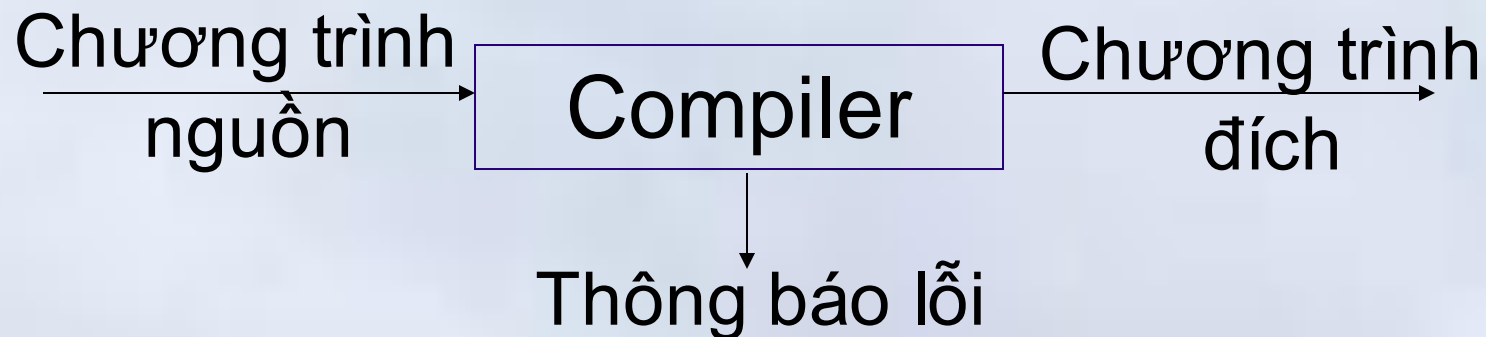
Ngôn ngữ lập trình cấp cao (NNLTCC)

Chương trình viết bằng NNLTCC

- Độc lập với máy tính
- Gần với ngôn ngữ tự nhiên
- Chương trình dễ đọc, viết và bảo trì
- Muốn thực hiện phải chuyển sang ngôn ngữ
 - Máy hiểu được (*ngôn ngữ máy*)
 - Ngôn ngữ trung gian mà máy hiểu được
- Được chuyển đổi bởi **Chương trình dịch**
- Chương trình thực hiện chậm hơn

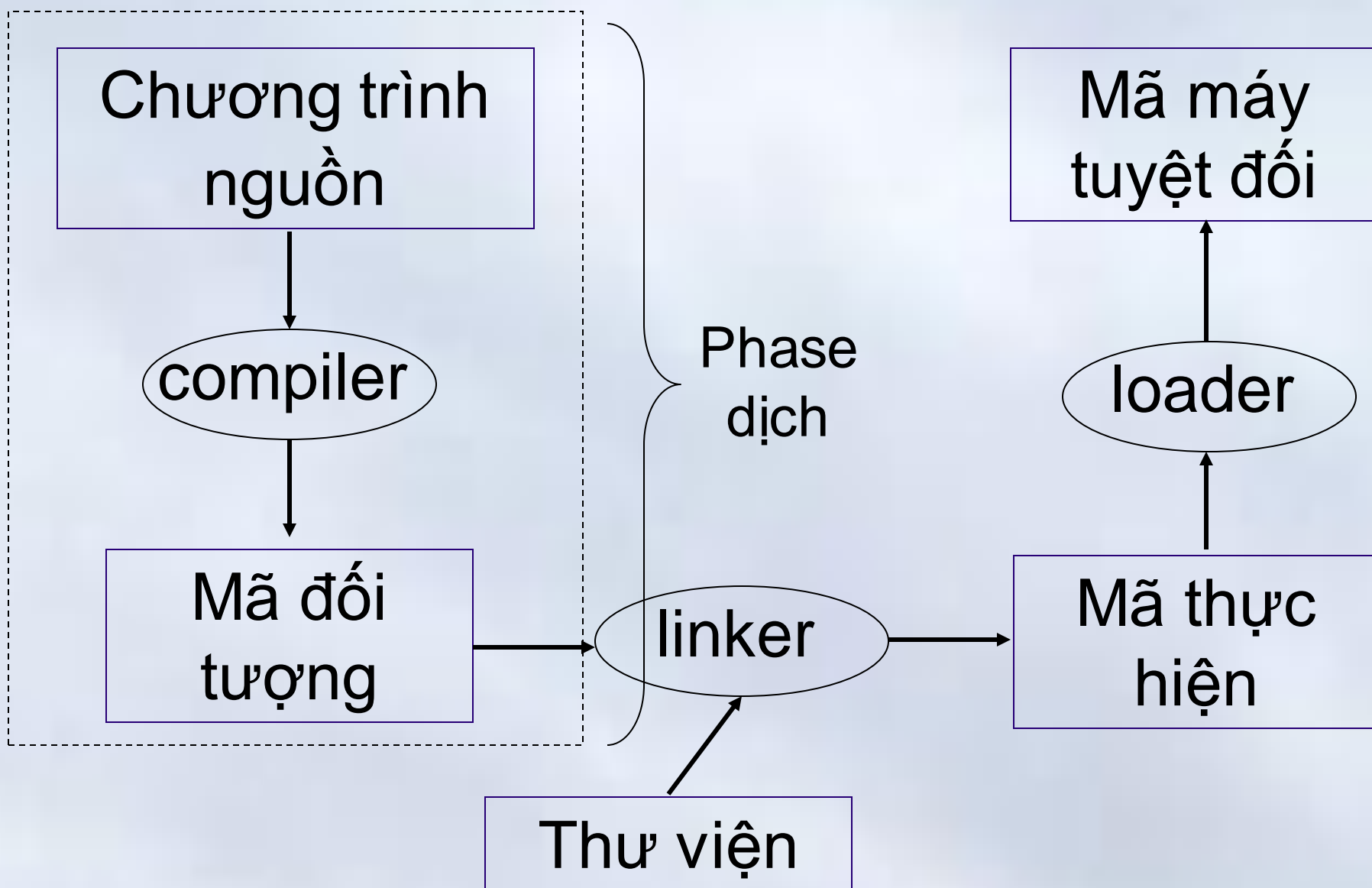
Chương trình biên dịch (compiler)

- Chương trình dịch làm nhiệm vụ dịch chương trình nguồn (*thường được viết bằng ngôn ngữ lập trình bậc cao*) sang các chương trình đối tượng (*chương trình đích*)

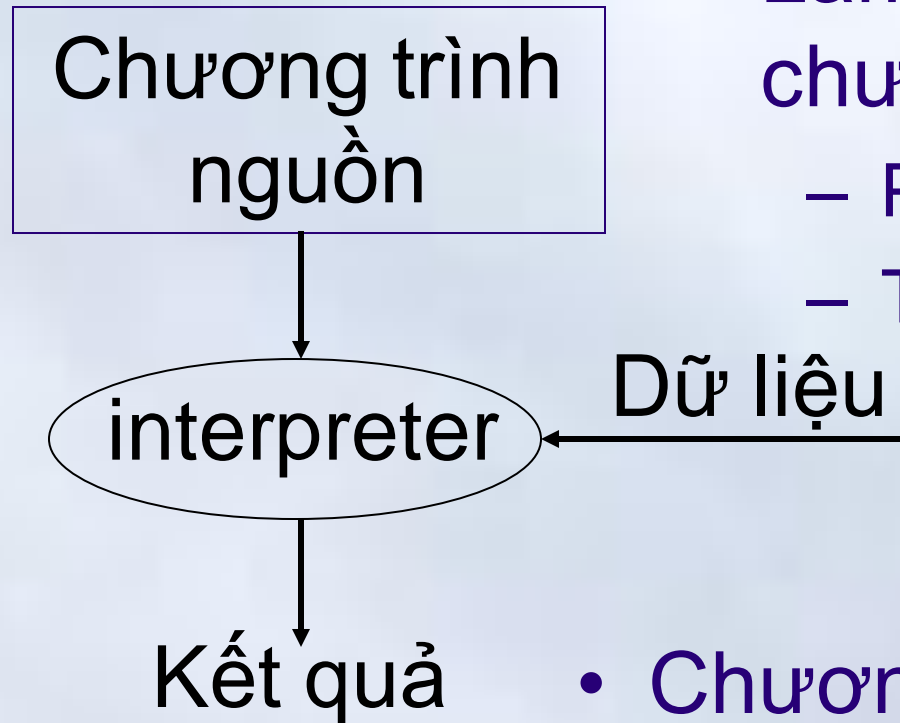


- Chương trình đích có thể không thực hiện được ngay mà cần liên kết (link) đến thư viện để được chương trình thực hiện

Các bước xử lý chương trình

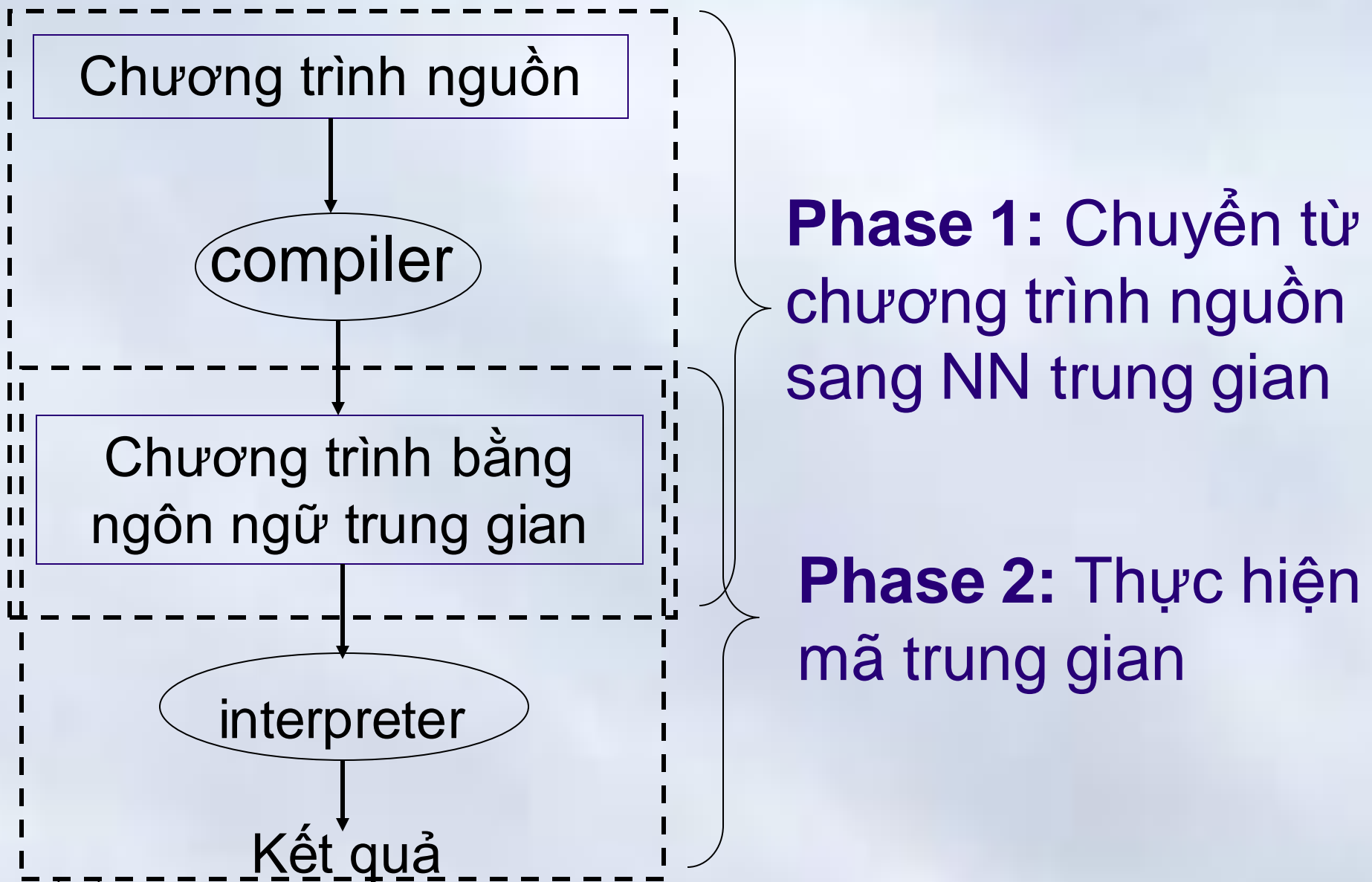


Thông dịch (interpreter)



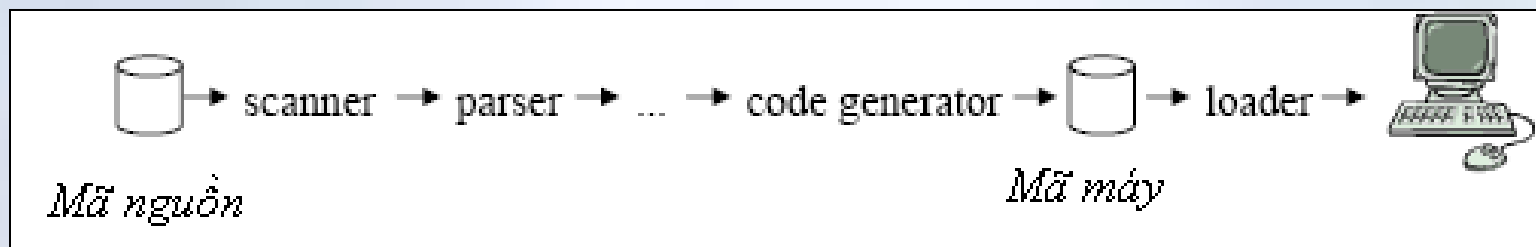
- Làm nhiệm vụ “*giải thích*” chương trình nguồn
 - Phân tích câu lệnh tiếp
 - Thực hiện câu lệnh
- Chương trình thông dịch có kích thước nhỏ hơn, nhưng chạy chậm hơn

Dịch và thực hiện chương trình nguồn

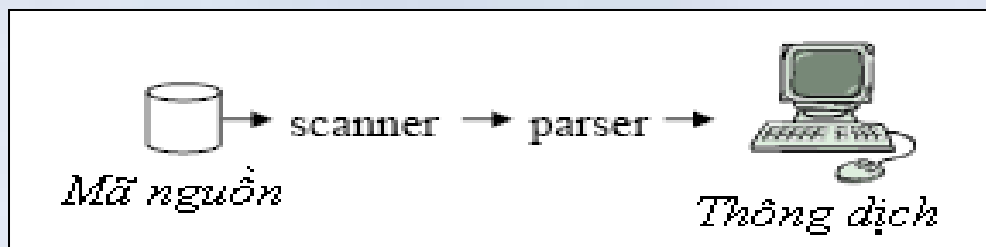


Compiler <> interpreter

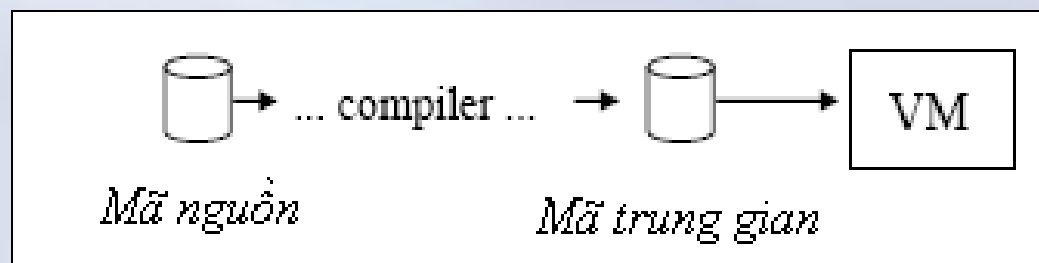
- **Compiler** : Dịch trực tiếp ra mã máy



- **Interpreter** : Trực tiếp thực hiện từng lệnh mã nguồn

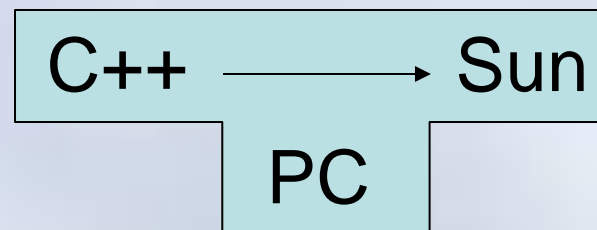
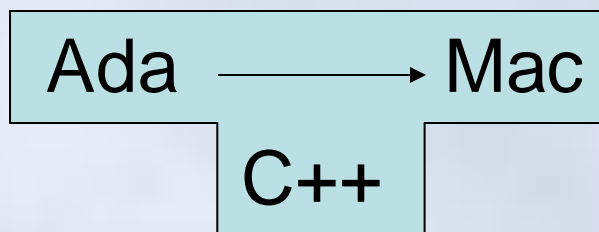
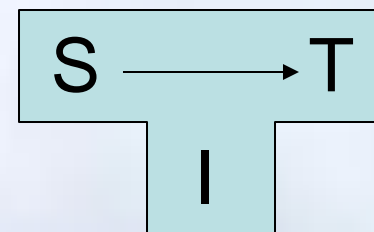


- Biến thể của Interpreter : thông dịch mã trung gian



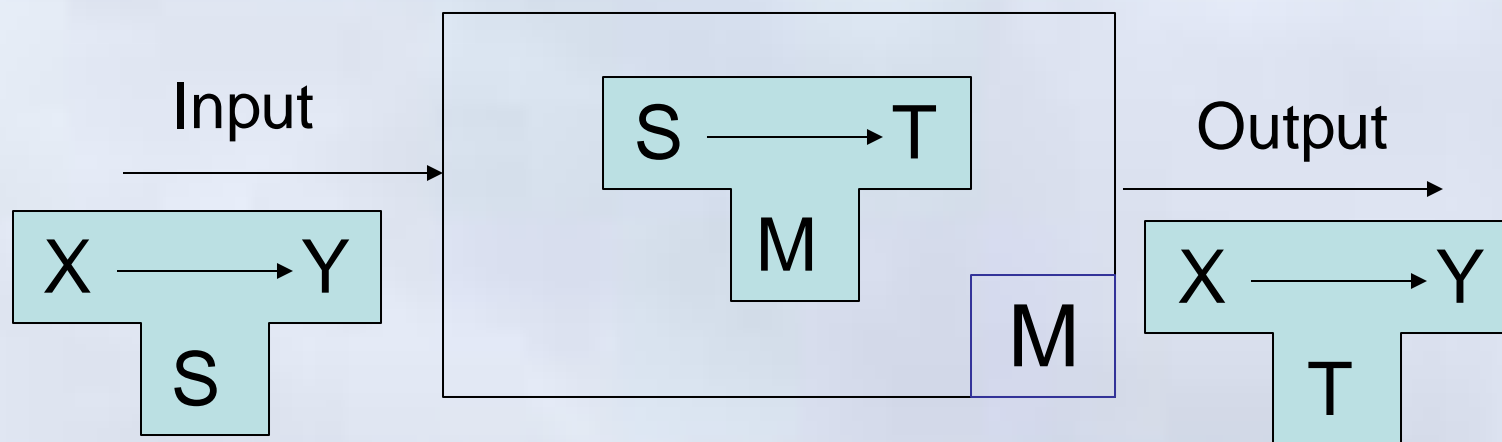
T-Diagram

- Chương dịch được đặc trưng bởi ($T, S \rightarrow T$)
 - Ngôn ngữ nguồn được dịch **S** (*input*)
 - Ngôn ngữ đích **T** (*output*)
 - Ngôn ngữ cài đặt **I** (*Đã tồn tại*)
 - Có thể là ngôn ngữ máy



T-Diagram

- M là máy tính, có ngôn ngữ máy M
- $T_M^{S \rightarrow T}$: Chương trình dịch trên M, dịch từ $S \rightarrow T$
 - **Input:** Chương trình viết bằng ngôn ngữ S
 - **Output:** Chương trình viết bằng ngôn ngữ T



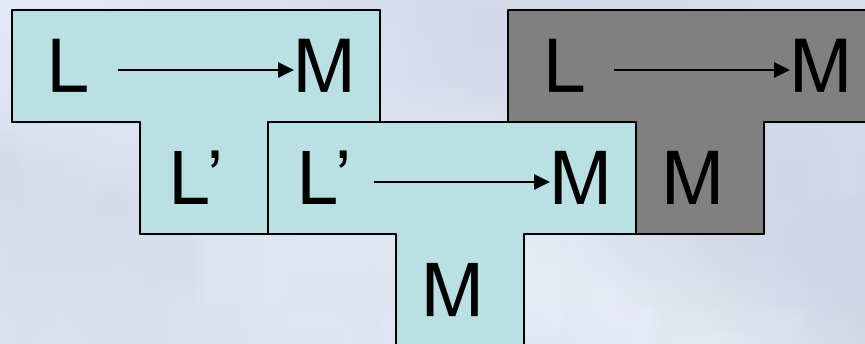
Xây dựng chương trình dịch

- Viết trực tiếp từ ngôn ngữ máy
 - Khó khăn
- Sử dụng ngôn ngữ bậc cao
 - Dễ dàng thực hiện, gọn rỗi,
 - Hiệu quả
 - Tăng tính khả chuyển,...
- **Chương trình bậc cao đầu tiên?**
 - Chiến lược mồi (*Bootstrapping*) : Đầu vào của một chương trình là chính nó

Xây dựng chương trình dịch: *Bootstrapping*

Yêu cầu: Viết $T_M^{L \rightarrow M}$

- CTD cho ngôn ngữ L , dịch ra mã máy M
- CTD thực hiện trên máy M
- Dùng mã máy M viết L' là tập con của L
 - Viết ra $T_M^{L' \rightarrow M}$
- Dùng L' để viết CTD $L \rightarrow M$ ($T_{L'}^{L \rightarrow M}$)
- Dùng $T_M^{L' \rightarrow M}$ để dịch $T_{L'}^{L \rightarrow M}$

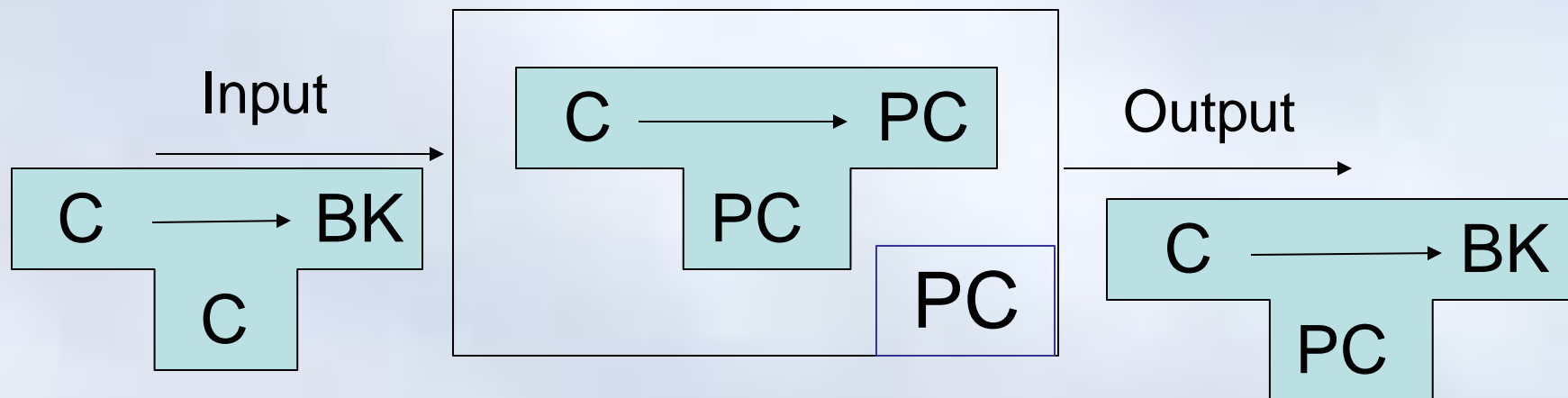


Xây dựng chương trình dịch: Cross Compiling

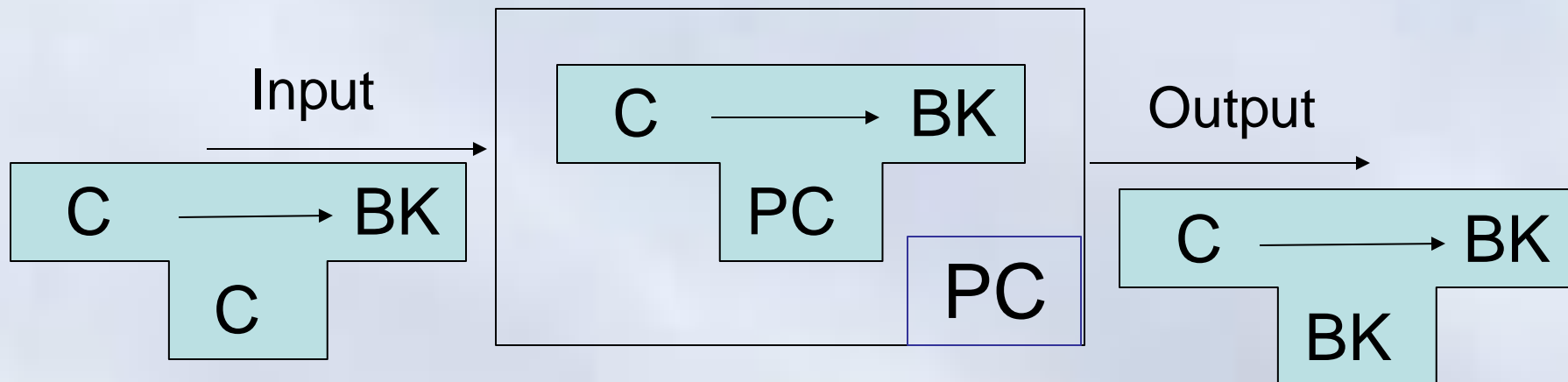
- Mục đích :
 - Xây dựng chương trình dịch cho ngôn ngữ đã tồn tại, trên một loại máy tính mới
 - Nếu máy mới đang được thiết kế: Cần phải biến kiến trúc tập lệnh, dạng lệnh, kiểu địa chỉ,..
 - Không sử dụng mã máy
- Ví dụ:
 - Yêu cầu xây dựng CTD $T_{BK}^{C \rightarrow BK}$
 - BK: là một loại máy tính mới, chưa được sản xuất
 - Đã tồn tại $T_{PC}^{C \rightarrow PC}$
 - Thực hiện xây dựng $T_{PC}^{C \rightarrow BK}$

Xây dựng chương trình dịch: Cross Compiling

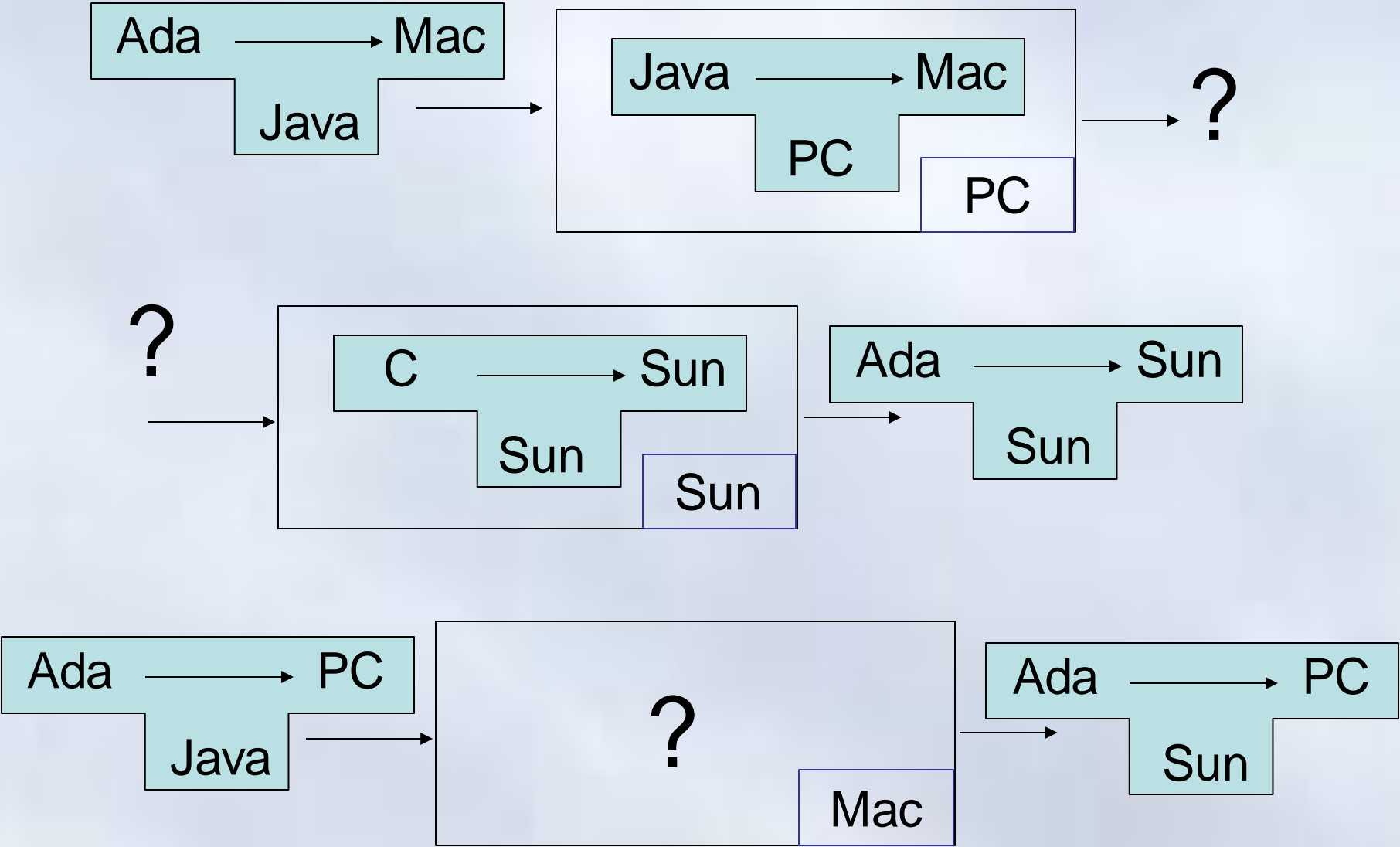
- Bước 1



- Bước 2



Xây dựng chương trình dịch: Cross Compiling



Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Các thể hệ ngôn ngữ lập trình

- Được chia thành 5 thể hệ.
- Việc phân chia cấp cao hay thấp phụ thuộc mức độ trừu tượng của ngôn ngữ
 - Cấp thấp : gần với máy
 - Cấp cao : gần với ngôn ngữ tự nhiên

Các thể hệ ngôn ngữ lập trình

- Các ngôn ngữ lập trình bậc thấp
 - Thể hệ thứ nhất : ngôn ngữ máy
 - Thể hệ thứ hai : Assembly
- Thể hệ thứ ba ← **Ngôn ngữ bậc cao**
 - Dễ hiểu hơn
 - Câu lệnh gần ngôn ngữ tự nhiên
 - Cho phép thực hiện các khai báo, Ví dụ biến
 - Phần lớn các NNLT cho phép lập trình cấu trúc
 - Ví dụ: Fortran, Cobol, C, C++, Basic .

Các thể hệ ngôn ngữ lập trình

- Ngôn ngữ lập trình thể hệ thứ tư
 - Thường được sử dụng trong một lĩnh vực cụ thể
 - Dễ lập trình, xây dựng phần mềm
 - Có thể kèm công cụ tạo form, báo cáo
 - Ví dụ :SQL, Visual Basic, Oracle . . .
- Ngôn ngữ lập trình thể hệ thứ năm
 - Giải quyết bài toán dựa trên các ràng buộc đưa ra cho chương trình (*không phải giải thuật của người lập trình*)
 - Việc giải quyết bài toán do máy tính thực hiện
 - Phần lớn các ngôn ngữ dùng để lập trình logic
 - Giải quyết các bài toán trong lĩnh vực trí tuệ nhân tạo

Các thành phần của NNLTCC

1. Từ vựng và cú pháp
2. Kiểu dữ liệu
3. Các đại lượng
4. Các toán tử và biểu thức
5. Các câu lệnh
6. Chương trình con

Từ vựng và cú pháp

❖ Từ vựng

- Chữ cái: A..Z, a..z
- Chữ số: 0..9
- Dấu: dấu chức năng, dấu toán tử
 - Dấu đơn: +, -, ; {, }
 - Dấu kép: >=, <=, /*, */
- Từ khóa : từ dành riêng cho ngôn ngữ
 - Được dùng để khai báo, ra lệnh cho chương trình

❖ Cú pháp

- Là các quy tắc để
 - Viết ra các đại lượng
 - Viết các câu lệnh

Kiểu dữ liệu

❖ Các kiểu cơ bản

- | | | |
|------------------|-------|---------|
| – Kiểu số nguyên | int | Integer |
| – Kiểu số thực | float | Real |
| – Kiểu ký tự | char | Char |
| – Kiểu logic | 0, 1 | Boolean |

❖ Kiểu dữ liệu có cấu trúc

- | | | |
|-----------------|--------|--------|
| – Kiểu mảng | [] | Array |
| – Kiểu chuỗi | * | String |
| – Kiểu bản ghi | struct | Record |
| – Kiểu con trỏ, | & | ^ |
| – Kiểu file | FILE | File |

Các đại lượng

❖ Hằng

- Số nguyên
 - Cách biểu diễn (decimal, octal, hexadecimal,...)
- Số thực
 - Dấu phẩy tĩnh
 - Dấu phẩy động
- Hằng chuỗi
 - C: “Hello”
 - Pascal: ‘Hello’

❖ Tên

- Nguyên tắc đặt tên ?
- Độ dài?

Toán tử và biểu thức

❖ Toán tử

- Số học: cộng (+), chia, chia dư(%, MOD),...
- Logic
 - Quan hệ: bằng (=, ==) khác (!=, <>), lớn hơn, (>),...
 - Logic: Và (AND, &&), hoặc (OR, ||),...
- Xâu: ghép xâu ←Pascal

❖ Biểu thức ← *Kết hợp các toán hạng bởi toán tử*

- Số học: *Trả về một con số*
- Logic: *Trả về một giá trị luận lý*
- Xâu: *Trả về một chuỗi ký hiệu*

Các câu lệnh

❖ Câu lệnh tuần tự

- Câu lệnh gán: `:=`, `=`
- Câu lệnh vào/ra, gọi hàm
- Câu lệnh ghép, gộp: `begin..end`, `{ }`

❖ Câu lệnh rẽ nhánh

- 1 vào 1 ra: `if...then`, `if()`
- 1 vào 2 ra: `if ...then...else`, `if()...else...`
- 1 vào, nhiều ra: `caseof`, `switch() { }`

❖ Câu lệnh lặp

- Số lần lặp xác định: `For ..to/downto..do`, `for(; ;)`
- Kiểm tra điều kiện trước: `While.. Do`, `while ()`
- Kiểm tra điều kiện sau: `Repeat..Until`, `do..while()`

Chương trình con

Các dạng chương trình con

- Thủ tục → Không trả về giá trị (void)
- Hàm → Trả về một giá trị

Vấn đề truyền tham số

- Truyền theo trị: Không thay đổi giá trị
- Truyền theo biến (địa chỉ): thay đổi giá trị

Vấn đề Địa phương /toàn cục

- Địa phương: chỉ tồn tại trong chương trình con
- Toàn cục: Tồn tại trong toàn bộ chương trình

Nhận xét

Ngôn ngữ lập trình bậc cao có
nguyên tắc giống nhau, cách thể
hiện có thể khác nhau

Bài tập thực hành số 1

Viết một chương trình bằng C. Chương trình có thể nhận 1 hoặc 2 tham số dòng lệnh

- Nếu 0 có tham số:
 - Đưa ra một thông báo lỗi
- Nếu có 1 tham số
 - Hiện thị xâu Hello + Tham số
- Nếu có 2 tham số
 - Hiện thị xâu Hello + tham số 1 vào file có tên là tham số 2
- Nếu có nhiều hơn 3 tham số.
 - Đưa ra một thông báo lỗi

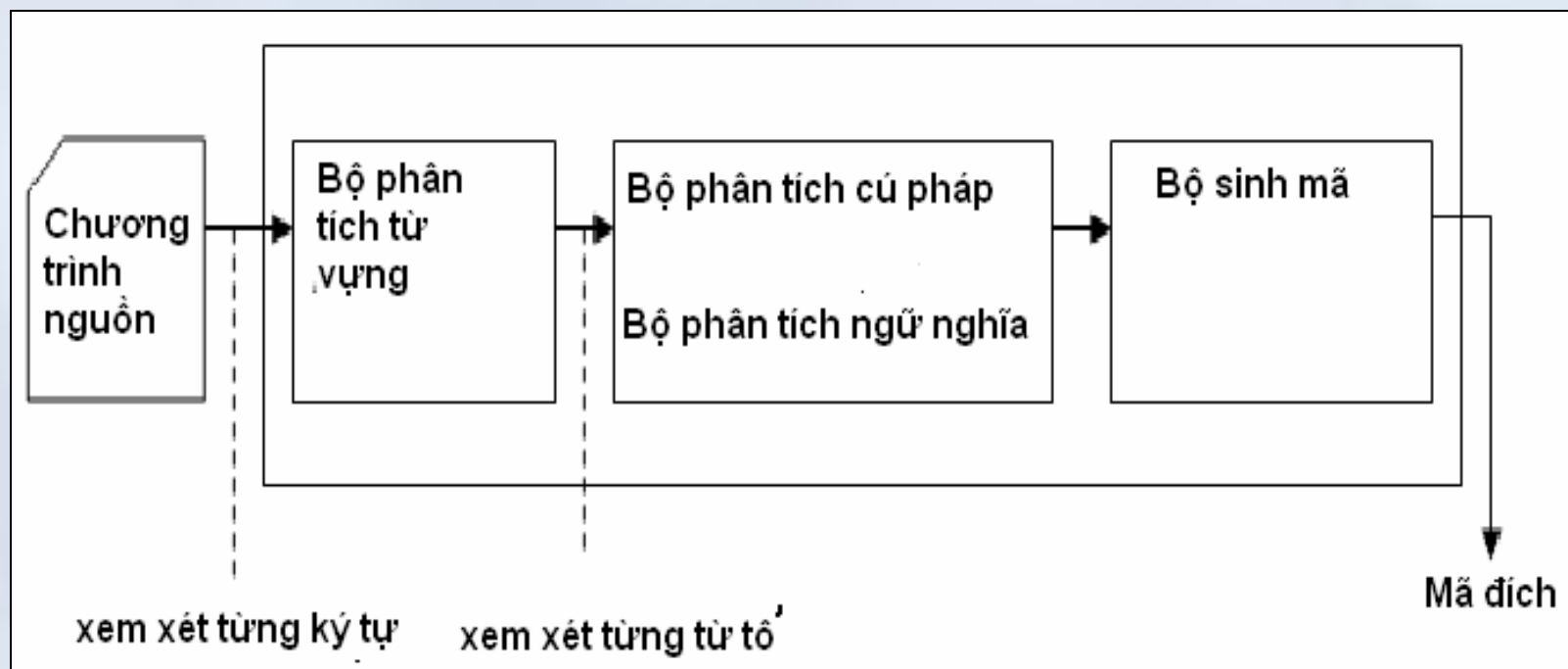
Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Các phase của chương trình dịch

Chương trình dịch gồm 3 phase chính

1. Phân tích từ vựng
2. Phân tích cú pháp
 - Phân tích ngữ nghĩa
3. Sinh mã



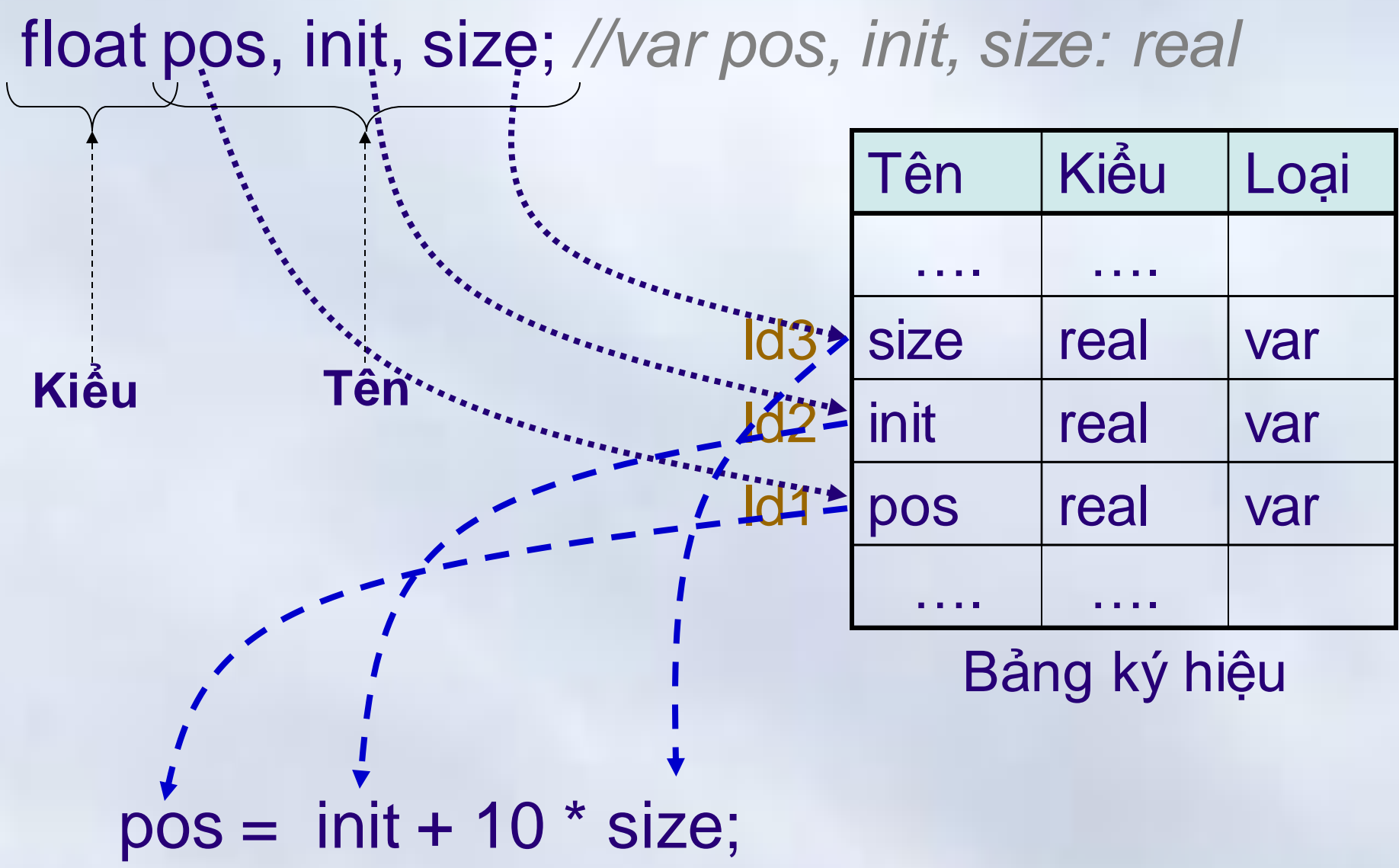
Cấu trúc chương trình dịch



Bảng ký hiệu

- Là cấu trúc dữ liệu dùng chứa tên và thuộc tính cần thiết của chúng
 - Thuộc tính cung cấp thông tin
 - Vị trí, kiểu, phạm vi hoạt động...
 - Nếu là tên chương trình con: số tham số, kiểu trả về
 - Tên được xác định bởi bộ phân tích từ vựng
- Khi chỉ ra được một tên, tùy thuộc vào vị trí của tên trong chương trình
 - Đưa tên và thuộc tính vào bảng ký hiệu
 - Lấy thông tin của tên trong bảng ký hiệu

Bảng ký hiệu→Ví dụ



Phân tích từ vựng (Lexical Analysis - Scanner)

Là pha đầu tiên của chương trình dịch

- Duyệt từng ký tự của chương trình nguồn
 - Loại bỏ các ký tự thừa
 - Dấu tab, khoảng trắng, chú thích..
- Xây dựng các từ vựng từ các ký tự đọc được
- Nhận dạng các *từ tổ* từ các từ vựng
 - **Từ tổ (*token*)** là đơn vị cú pháp được xử lý trong quá trình dịch như một thực thể không thể chia nhỏ hơn nữa
- Chuyển các từ tổ cho pha tiếp

Phân tích từ vựng → Ví dụ

```
pos := init + 10 * size;
```

Bộ PTTV thực hiện

- Đọc từng ký tự: bắt đầu từ chữ cái **p**
 - Nhận dạng từ vựng thuộc dạng tên, hoặc từ khóa (*vì bắt đầu bởi 1 chữ cái*)
 - Đọc tiếp (**o**, **s**) tới khi gặp ký tự khác chữ cái, chữ số,
- Gặp dấu trắng → xây dựng xong từ vựng **pos**
 - Do **pos** không trùng với từ khóa. Vậy **pos** là tên (*ident*)
 - Trả lại cho bộ phân tích cú pháp từ tổ *ident*
- Đọc tiếp được dấu **:** rồi dấu **=** và sau đó dấu cách
 - Nhận dạng được từ vựng **:=** và trả về từ tổ gán (*assign*)

Phân tích từ vựng → Ví dụ

```
pos := init + 10 * size;
```

Kết quả thực hiện PTTV :

	Từ vựng	Từ tổ	Ý nghĩa	
1	pos	ident	Tên	<div>- Các từ tổ: <i>assign, ident, plus, times...</i> do người viết CTD tự đặt ra để dễ dàng mã hóa c/trình. - Pttv có thể coi là giai đoạn số hóa chương trình nguồn</div>
2	:=	assign	Phép gán	
3	init	ident	Tên	
4	+	plus	Dấu cộng	
5	10	number	Con số	
6	*	times	Dấu nhân	
7	size	ident	Tên	
8	;	semicolon	Chấm phẩy	

Trả về: ident, assign, ident, plus, number, times, ident, semicolon

Phân tích cú pháp (Syntax Analysis)

- Bộ ptcp phân tích chương trình nguồn
 - Dựa vào các từ tổ nhận được từ pha pttv
- Kiểm tra những từ tổ có tuân theo quy tắc cú pháp của ngôn ngữ được dịch không
 - Cú pháp thể hiện cấu trúc văn phạm của ngôn ngữ, được mô tả dạng: đệ quy, BNF, sơ đồ..
- Kết quả của bộ phân tích cú pháp:
 - Cây phân tích cú pháp (*nếu có*)
 - Có cây phân tích → chương trình đúng cú pháp
 - Thông báo lỗi nếu ngược lại

Phân tích cú pháp → Ví dụ 1

- Quy tắc định nghĩa một biểu thức
 - 1. Tên là một biểu thức
 - 2. Con số là biểu thức
 - 3. Nếu E, E_1, E_2 là biểu thức thì
 - $E_1 + E_2, E_1 * E_2, (E)$ là biểu thức

Luật cơ sở

Luật đệ quy
- Kiểm định: ***init + 10 * size*** là biểu thức?
 - Theo (2): **10** là biểu thức
 - Theo (1): **size** là biểu thức
 - Theo (3) **10 * size** là biểu thức
 - Theo (1): **init** là biểu thức
 - Theo (3): **init + 10 * size** là biểu thức

Phân tích cú pháp → Ví dụ 2

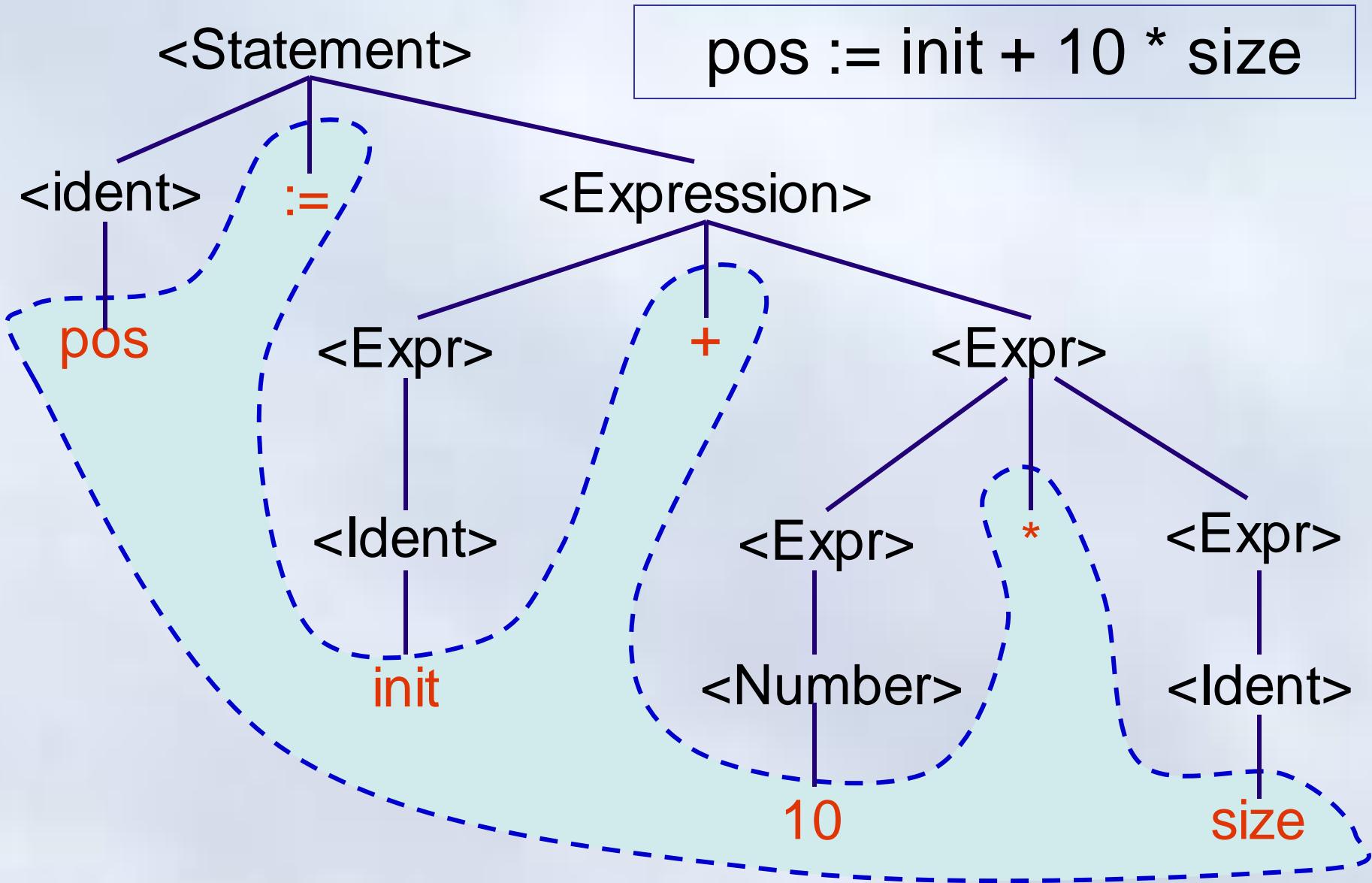
- Quy tắc định nghĩa một lệnh
 1. Nếu Id là một Tên, E là một biểu thức thì
 $Id := E$ là một câu lệnh
 2. Nếu E là một biểu thức, S là một lệnh thì
 $If\ E\ Then\ S, While\ E\ Do\ S$ là câu lệnh

Kiểm định: **$pos := init + 10 * size$** là câu lệnh?

- Theo bộ phân tích từ vựng: pos là một tên
- Theo ví dụ 1: $init + 10 * size$ là một biểu thức
- Vậy theo luật (1) $pos := init + 10 * size$ là một lệnh

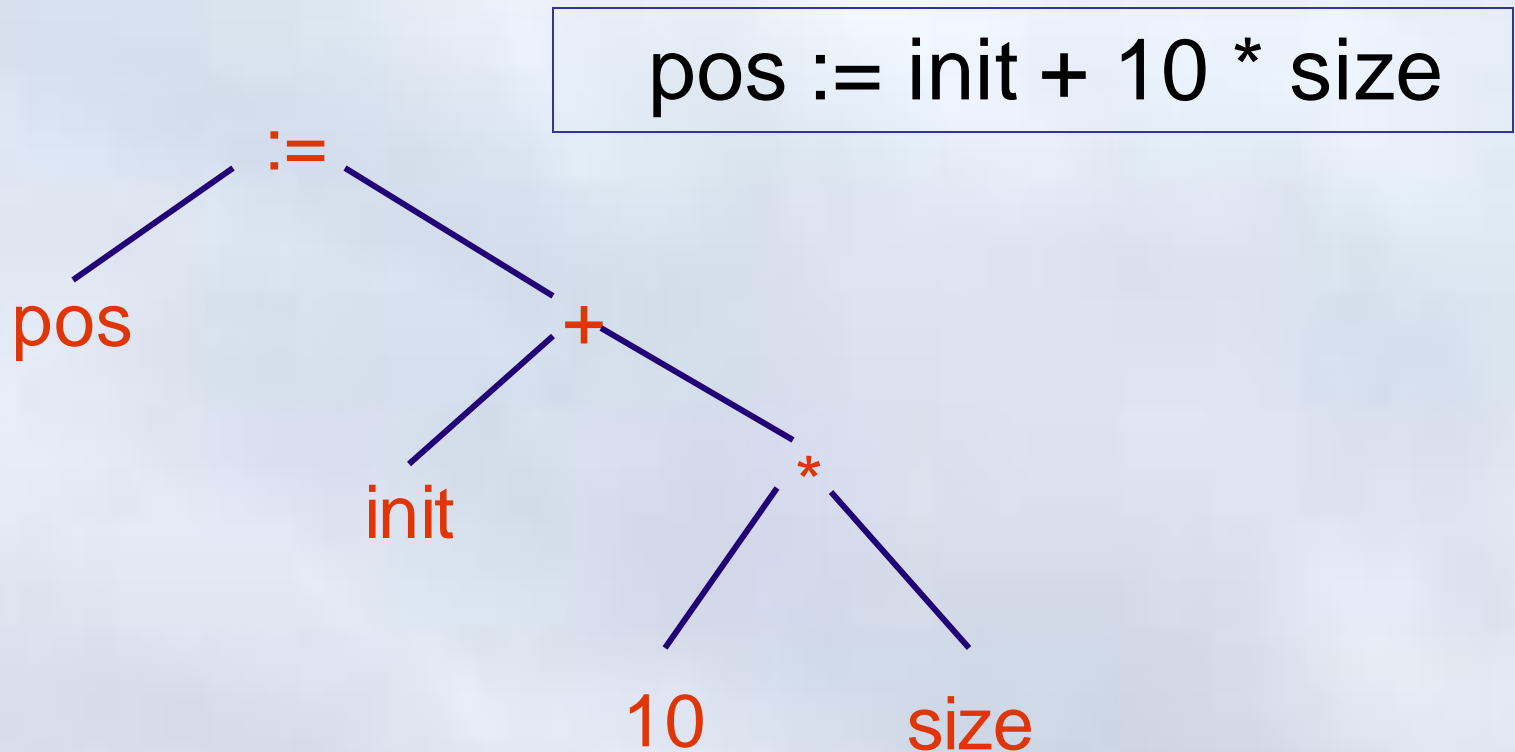
Ptcp thường được biểu diễn bởi cây phân tích/cây cú pháp

Phân tích cú pháp → Cây phân tích (parse tree)



Phân tích cú pháp → Cây cú pháp (syntax tree)

- Các nút trong là các toán tử
- Các toán hạng là các nút con của toán tử



Phân tích ngữ nghĩa (Semantic Analysis)

- Kiểm tra lỗi ngữ nghĩa của chương trình
 - Lệnh *pos := init+10*size* đúng cú pháp
 - Nếu pos là hằng số
 - *pos := init+10*size* sai ngữ nghĩa
- Kiểm tra kiểu toán hạng có phù hợp toán tử
 - % (MOD) đòi hỏi toán hạng nguyên
 - Một số toán tử chấp nhận toán hạng khác kiểu
 - Vấn đề chuyển kiểu tự động (*nguyên* → *thực*)
 - Ví dụ *pos := init+int2Real(10) * size*
- Lấy thông tin về kiểu của danh biểu (tên)
 - Thông tin dùng cho giai đoạn sinh mã

Sinh mã (Code Generation)

- Được thực hiện khi chương trình nguồn đúng cả về cú pháp và ngữ nghĩa
- Thường gồm 3 giai đoạn
 - Sinh mã trung gian
 - Tối ưu mã
 - Sinh mã đích

Sinh mã → Sinh mã trung gian

- Mã nguồn được chuyển sang chương trình tương đương trong ngôn ngữ trung gian
 - Mã trung gian là mã máy độc lập, tương tự với tập lệnh trong máy.
- Ưu điểm của mã trung gian
 - Thuận lợi khi cần thay đổi cách biểu diễn chương trình đích
 - Có thể tối ưu hóa mã độc lập với máy đích cho dạng biểu diễn trung gian.
 - Giảm thời gian thực thi chương trình đích vì mã trung gian có thể được tối ưu
- Ngôn ngữ trung gian
 - Mã 3 địa chỉ (*thường được dùng*)
 - Cây cú pháp, Ký pháp Ba Lan sau (hậu tố),...

Sinh mã → Sinh mã trung gian → Mã 3 địa chỉ

- Mỗi câu lệnh có nhiều nhất
 - 3 toán hạng
 - 2 toán tử, trong đó có 1 toán tử gán
- Chương trình dịch phải sinh ra biến tạm để chứa giá trị tính toán sau mỗi lệnh

Ví dụ: $pos := init + \text{Int2Real}(10) * size$

Sinh ra mã 3 địa chỉ

Temp1 := Int2Real(10)

Temp2 := Temp1 * Id 3

Temp3 := Id2 + Temp2

Id1 := Temp3

Id1, Id2, Id 3 là các con trỏ, trỏ tới các phần tử size, init, pos trong bảng ký hiệu

Sinh mã → Tối ưu mã trung gian

- Mục đích:
 - Tối ưu về kích thước, tốc độ

Ví dụ

- *Int2Real()* được thay bằng số thực tại thời điểm dịch
- *Temp3* chỉ dùng 1 lần làm nơi lưu tạm thời dữ liệu trước khi chuyển cho *Id1*
 - Có thể thay *Id1* trực tiếp cho *Temp3*
- Kết quả
$$\begin{array}{ll} \text{Temp1} & := 10.0 * \text{Id } 3 \\ \text{Id1} & := \text{Id2} + \text{Temp1} \end{array}$$

Sinh mã → Sinh mã đích

- **Mục đích:** Tạo chương trình thực thi
- **Thực hiện:**
 - Các biến được cấp ô nhớ cụ thể
 - Các câu lệnh trung gian được thay bằng chuỗi mã máy tương đương
 - Các biến được ấn định cho các thanh ghi

Ví dụ

MOVF Id3, R2

MULF #10.0, R2

MOVF Id2, R1

ADDF R2, R1

MOVF R1, Id1

Temp1 := 10.0 * Id 3

ID1 := Id2 + Temp1

Xử lý lỗi

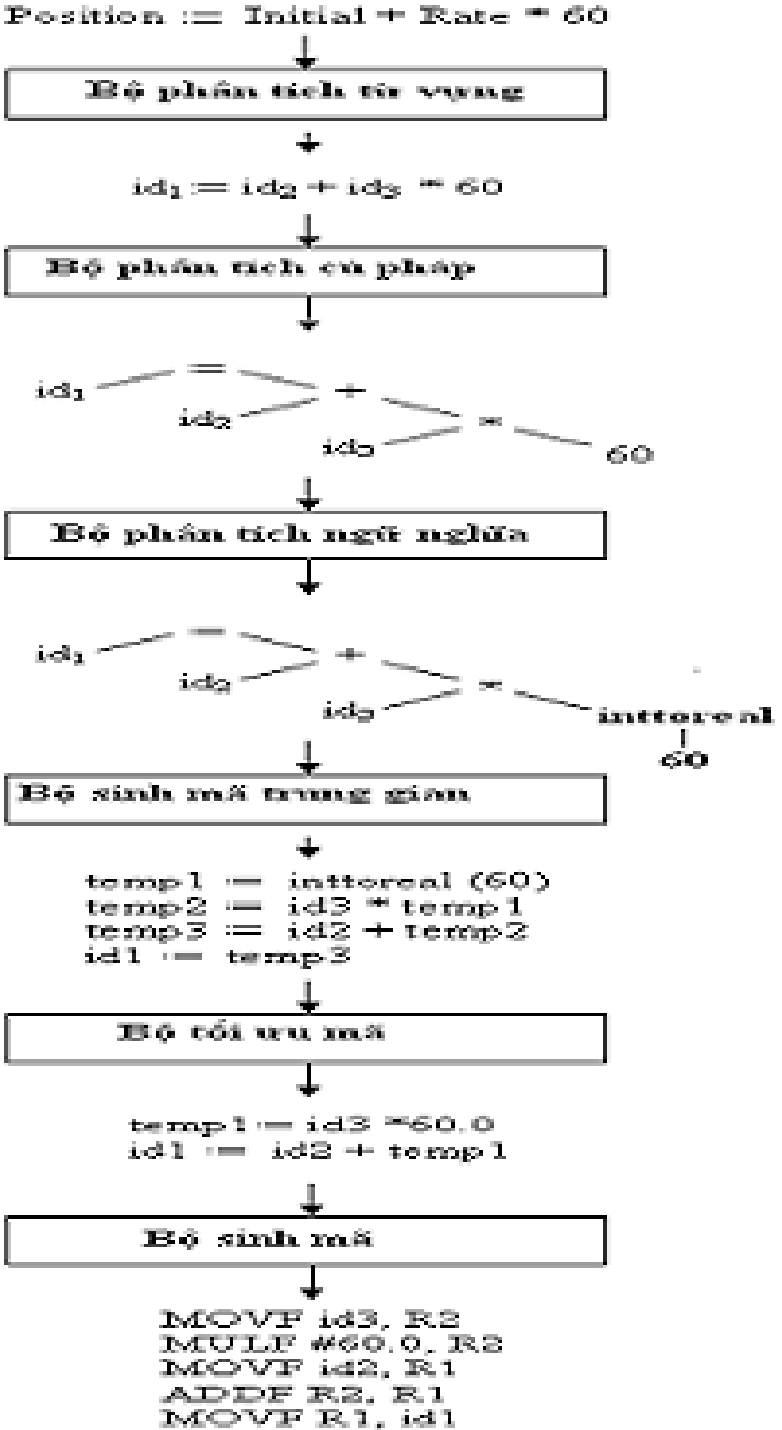
Lỗi có thể gặp trong mọi pha của CTD

- Phân tích từ vựng
 - Gặp ký tự lạ. Ví dụ: @, \$,... trong NNLT C
- Phân tích cú pháp
 - Không tuân theo cú pháp: VD *while* (*a* <> *b*)
- Phân tích ngữ nghĩa
 - Gán giá trị cho hằng, tính toán với tên thủ tục
- Sinh mã
 - Kích thước quá lớn

Ví dụ: `int Arr[1000][1000];` → Lỗi tràn ô nhớ

Xử lý lỗi

- Gặp lỗi, chương trình dịch cần thông báo
 - Ghi nhận kiểu lỗi
 - Ghi nhận vị trí gây ra lỗi (dòng, cột,..)
- Có thể cần hồi phục sau khi gặp lỗi
 - Mục đích: cho phép tiến hành phân tích tiếp tục, tránh lãng phí
 - Có thể thông báo không chính xác



Quá trình dịch một câu lệnh

Position := Initial + Rate * 60

[Phan Thi Tươi, 1998]

Ghi chú

- Phân chia thành từng pha nhằm mục đích nghiên cứu để chọn giải pháp thích hợp
- Trong cài đặt, kết hợp các pha thành các modul chương trình. Thường gồm 2 phần
 - **Phần đầu:** Modul liên quan tới phân tích từ vựng, phân tích cú pháp, phân tích ngữ nghĩa và sinh mã trung gian
 - **Phần sau:** Tối ưu mã, sinh mã đích, xử lý lỗi
- Cơ sở đề ra ngôn ngữ lập trình là **lý thuyết ngôn ngữ và văn phạm**

Bài tập thực hành số 2

- Sử dụng gcc để dịch file nguồn trong chế độ dòng lệnh
 - \>gcc sourcefile.c
- Sử dụng makefile để dịch project dịch
 - \>make
- Viết chương trình thực hiện
 - **Input:** 01 tham số C:\> TenFile TextFile.txt
 - **Output:** Hiện thị nội dung TextFile, mỗi từ trên một dòng

Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Giới thiệu

- Ngôn ngữ (*tự nhiên/nhân tạo*):
 - Tập hợp các câu có cấu trúc quy định
- Câu:
 - Tập hợp các từ
- Từ:
 - Ký hiệu nào đó
- Ngữ pháp/cú pháp
 - **Cấu trúc quy định** tạo câu của trong ngôn ngữ
 - Ngôn ngữ lập trình → cú pháp

Khái niệm văn phạm và ngôn ngữ

1. Kiến thức toán học liên quan
2. Bộ chữ và xâu ký hiệu
3. Văn phạm
4. Suy dẫn
5. Câu
6. Ngôn ngữ
7. Độ quy
8. Phân loại văn phạm

Kiến thức toán học liên quan → Tập hợp

- Khái niệm:
 - Tập không có thứ tự và không lặp lại của các p/tử
 - **Ví dụ:** $A_5 = \{1, 3, 5\}$ tập các số nguyên dương lẻ nhỏ hơn 6
 - Tập không có phần tử nào được gọi là **tập rỗng**
 - Tập rỗng được ký hiệu: \emptyset
- Biểu diễn: $A = \{x \mid x \text{ thỏa mãn tính chất } Q\}$
 - Chỉ ra tập các phần tử thỏa mãn tính chất Q
 - Sử dụng biểu đồ Venn
 - Mỗi phần tử biểu diễn là một điểm
 - Mỗi tập là một hình bao quanh các điểm
- Lực lượng của một tập: Số phần tử của tập
 - Tập có thể hữu hạn hoặc vô hạn phần tử
 - Lực lượng của tập A, được ký hiệu $|A|$

Kiến thức toán học liên quan → Tập hợp

- Ký hiệu :
 - $a \in A$ a là phần tử trong tập A
 - $a \notin A$ a không phải phần tử trong tập A
 - $A \subset B$ A là tập con thực thụ của B
 - $A \subseteq B$ A là tập con của B
- Tập lũy thừa của A : Tập các tập con của A
 - Tập lũy thừa của A ký hiệu 2^A
 - A là một tập thì $2^A = \{S \mid S \subseteq A\}$
 - Ví dụ $2^{\{1,3,5\}} = \{\emptyset, \{1\}, \{3\}, \{5\}, \{1,3\}, \{1,5\}, \{5,3\}, \{1,3,5\}\}$
 - Nhận xét $|2^A| = 2^{|A|}$
 - $S \in 2^A \rightarrow \exists$ số nhị phân $e = e_1 \dots e_n$ ($n = |A|$) thỏa mãn : $e_i = 1$ khi phần tử i của A có mặt trong S ; Ngược lại $e_i = 0$

Kiến thức toán học liên quan → Tập hợp

- Các phép toán trên tập: A, B là 2 tập


– **Hợp:** $A \cup B = \{a \mid a \in A \text{ hoặc } a \in B\}$

– **Giao:** $A \cap B = \{a \mid a \in A \text{ và } a \in B\}$

- A, B được gọi là phân biệt nếu $A \cap B = \emptyset$

– **Hiệu:** $A - B = \{a \mid a \in A \text{ và } a \notin B\}$

– **Tích Đề-các:** $A \times B = \{\langle a, b \rangle \mid a \in A \text{ và } b \in B\}$

• $A = \{1, 2\}$
 • $B = \{a, b\}$  $A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$

• $A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) \mid a_i \in A_i\}$

• Tích Đề-các lũy thừa bậc n:

$$A^n = A \times A \times \dots \times A = \{(a_1, \dots, a_n) \mid a_i \in A\}$$

Kiến thức toán học liên quan → Quan hệ

- Khái niệm:
 - Tập con \mathfrak{R} của tích Đề-các của các tập
- Quan hệ 2 ngôi:
 - Tập con của tích Đề-các 2 tập
- Ví dụ:
 - $a \in A$ và $b \in B \Rightarrow (a,b) \in A \times B$ là một quan hệ 2 ngôi
 - Nếu $A \equiv B \Rightarrow$ Quan hệ 2 ngôi trên A
- Nếu \mathfrak{R} là một quan hệ và $(a,b) \in \mathfrak{R}$, ký hiệu $a \mathfrak{R} b$

Kiến thức toán học liên quan → Hàm

- Hàm là một quan hệ trên tích Đề-các $\mathcal{D} \times \mathcal{R}$ thỏa mãn
 - $\forall d \in \mathcal{D}, \exists$ nhiều nhất 1 cặp $(d, r) \in \mathcal{D} \times \mathcal{R}$
 - \mathcal{D} được gọi là **miền xác định**
 - \mathcal{R} được gọi là **miền giá trị**
- Ký hiệu $f: \mathcal{D} \rightarrow \mathcal{R}$ (f : là một hàm/ ánh xạ)
 - Nếu $(d, r) \in \mathcal{D} \times \mathcal{R}$ thì
 - Giá trị của hàm f tại d là r và ký hiệu **$f(d) = r$**
- Hàm xác định: Khi \mathcal{D} và \mathcal{R} là những tập xác định
 - Có thể biểu diễn bằng bảng các cặp $\{(d, r)\}$
- Hàm n biến vào, m biến ra
 - Miền xác định là tích Đề-các của n tập,
 - Miền giá trị là tích Đề-các của m tập

Kiến thức toán học liên quan → Đồ thị

Đồ thị $G = (V, E)$ bao gồm

- Tập hữu hạn V các nút - đỉnh
- Tập hữu hạn E các cặp nút – cạnh
 - $E = \{(v_1, v_2) \mid v_1, v_2 \in V\} \Leftrightarrow E \subseteq V \times V$
- Đồ thị vô hướng: $\exists (v_1, v_2) \in E \Rightarrow \exists (v_2, v_1) \in E$
- Đồ thị có hướng: Không phải là đồ thị vô hướng
 - Nếu $(v_1, v_2) \in E$: $\mathbf{v_1}$ là đỉnh đứng trước $\mathbf{v_2}$ trong đồ thị
- Đường đi trên đồ thị có hướng/vô hướng
 - Dãy đỉnh (v_1, v_2, \dots, v_n) thỏa mãn $(v_i, v_{i+1}) \in E$ ($i: 1..n-1$)
- Đồ thị có chu trình:
 - \exists đường đi đỉnh (v_1, v_2, \dots, v_n) , trong đó $v_1 \equiv v_n$

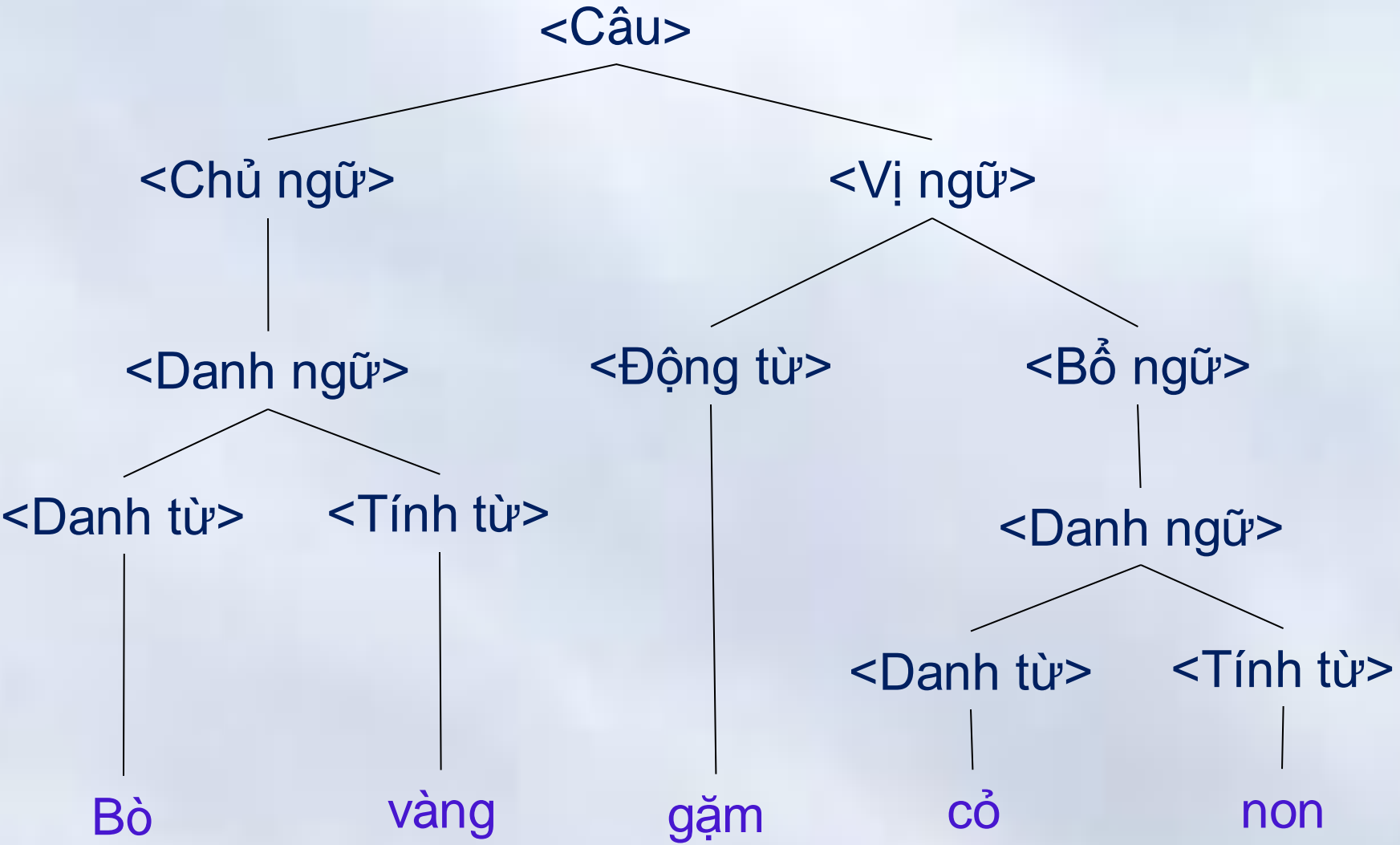
Kiến thức toán học liên quan→Cây

Là một đồ thị định hướng với các tính chất sau

- Có một nút gọi là gốc, có các tính chất
 - Không có nút đứng trước gốc
 - Có một đường đi từ gốc đến mọi nút khác
- Nút khác với nút gốc có đúng một nút đứng trước
- Các nút sau của một nút: Sắp thứ tự từ trái qua phải
- Quy ước biểu diễn cây
 - Vẽ nút gốc ở trên, cung hướng xuống dưới (bỏ mũi tên)
 - Các nút sau vẽ theo thứ tự từ trái qua phải
- Nút lá: Không có nút sau (không có con)
 - Nút trong: Mọi nút khác nút lá

Kiến thức toán học liên quan→Cây

Cấu trúc ngữ pháp của một câu tiếng việt



Bộ chữ

- Tập các thành phần được gọi là các ký hiệu
 - Trong thực tế: Tập là hữu hạn và khác rỗng.
 - Thường được ký hiệu **V** (**V**ocabulary) hoặc Σ

Ví dụ

- Bảng chữ cái a,b,c...:
 - Bộ chữ cái của một ngôn ngữ tự nhiên
- Bộ từ điển tiếng Anh
 - Bộ chữ cái trong ngôn ngữ tiếng Anh
- Các từ khóa, tên, ký hiệu..
 - Bộ chữ của một ngôn ngữ lập trình

Xâu ký hiệu

Khái niệm

- Là dãy liên tiếp các ký hiệu của một bộ chữ
 - Được gọi là xâu trên bộ chữ đó
 - Thường được ký hiệu α , β , γ ,...
- Xâu rỗng: xâu không chứa ký hiệu nào
 - Thường được ký hiệu ε

Ví dụ

- α : **Madam** \rightarrow Xâu trên bảng chữ cái a, b, c,...
- β : **Hôm_nay trời đẹp** \rightarrow Xâu trong tiếng Việt
- γ : **while (a > b) a = a - b;** \rightarrow Xâu trong NNLT

Xâu ký hiệu

- Độ dài chuỗi
 - Số ký hiệu trong chuỗi đó
 - Thường được ký hiệu $|\alpha|$ hay $l(\alpha)$
 - Ví dụ:

$l(\alpha) = 5$	$l(\beta) = 3$
$l(\gamma) = 12$	$l(\epsilon) = 0$
- Đảo ngược chuỗi
 - Đảo ngược chuỗi α , (ký hiệu là α^R)
 - Viết các ký hiệu của chuỗi α theo chiều ngược lại
- Chuỗi con
 - Chuỗi v là chuỗi con của w nếu chuỗi v được tạo nên từ dãy các ký hiệu liên tiếp của chuỗi w
 - v là tiền tố w nếu v nằm ở đầu, nếu nằm ở cuối, v là hậu tố
 - Ví dụ: ada là chuỗi con của chuỗi $madam$

Xâu ký hiệu

- Ghép chuỗi
 - Ghép 2 chuỗi α , β là việc viết các ký hiệu của chuỗi α rồi viết tiếp các ký hiệu của chuỗi β
- Chuỗi lũy thừa:
 - $\alpha^n = \alpha \alpha \dots \alpha$ (ghép n chuỗi α)
- Ví dụ: $\alpha: abc$ và $\beta: ba$
 - $\alpha\beta = abcba$ và $\beta\alpha = bacab$
 - $\alpha^3 = abcabcabc$
- Nhận xét:
 - $\alpha\beta \neq \beta\alpha$; $\alpha\varepsilon \equiv \varepsilon\alpha \equiv \alpha$; $\varepsilon\varepsilon \equiv \varepsilon$;
 - $l(\alpha\beta) = l(\alpha) + l(\beta)$

Xâu ký hiệu \rightarrow Tính toán trên tập xâu

- *A, B là 2 tập xâu trên một bộ chữ*
 - **Hợp:** $A \cup B = \{\alpha \mid \alpha \in A \text{ hoặc } \alpha \in B\}$
 - **Giao:** $A \cap B = \{\alpha \mid \alpha \in A \text{ và } \alpha \in B\}$
 - **Tích/Ghép:** $AB = \{x = \alpha\beta \mid \alpha \in A \text{ và } \beta \in B\}$
 - **Tích Descarter:** $A \times B = \{\langle \alpha, \beta \rangle \mid \alpha \in A \text{ và } \beta \in B\}$
- **Ví dụ:** $A = \{0, 01\}$ $B = \{\varepsilon, 1, 01\}$
 - $A \cup B = \{\varepsilon, 0, 1, 01\}$
 - $A \cap B = \{01\}$
 - $AB = \{0, 01, 001, 011, 0101\}$
 - $A \times B = \{\langle 0, \varepsilon \rangle, \langle 0, 1 \rangle, \langle 0, 01 \rangle, \langle 01, \varepsilon \rangle, \langle 01, 1 \rangle, \langle 01, 01 \rangle\}$

Xâu ký hiệu \rightarrow Tính toán trên tập xâu

- *A là tập xâu trên một bộ chữ*
 - **Lũy thừa:** $A^n = \{\epsilon\}$ nếu $n = 0$
 $A^n = AA^{n-1} = A^{n-1}A$ nếu $n > 0$
 - **Bao đóng:** $A^* = \lim(A^0 \cup A^1 \cup \dots \cup A^n), n \rightarrow \infty$
 - **Bao đóng dương:** $A^+ = \lim(A^1 \cup A^2 \cup \dots \cup A^n)$
- Nhận xét
 - **A^* :** Tập tất cả các xâu có thể được tạo nên từ các ký hiệu trong A , *kể cả xâu rỗng*.
 - **A^+ :** Tập tất cả các xâu có thể được tạo nên từ các ký hiệu trong A , *không kể xâu rỗng*.

Xâu ký hiệu \rightarrow Tính toán trên tập xâu

- **Ví dụ :** Cho $A = \{0, 1\}$, $B = \{a, b\}$
 - $A^0 = \{\varepsilon\}$ $A^1 = \{0, 1\}$
 - $A^2 = \{00, 01, 10, 11\}$
 - $A^n = \{\text{Tập các số nhị phân độ dài } n\}$
 - $A^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \dots\}$
 - $A^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$
 - $BA^2 = \{a, b\}\{00, 01, 10, 11\}$
 $= \{a00, a01, a10, a11, b00, b01, b10, b11\}$
 - $BA^+ = ?$
 - $A^+BA^+ = ?$
 - $ABA^* = ?$

Ngôn ngữ (Languages)

- Cho V Là một bộ chữ
 - V^* Tập tất cả các xâu trên bộ chữ, kể cả xâu ϵ
 - V^+ Tập tất cả các xâu trên bộ chữ, không kể xâu ϵ
 - V^* là đếm được: Liệt kê lần lượt các xâu của V^* theo thứ tự độ dài
Cùng độ dài, liệt kê theo thứ tự từ điển
- Cho V Là một bộ chữ,
 $L \subseteq V^*$ được gọi là một ngôn ngữ trên bộ chữ V
 - Mỗi p/tử của L được gọi là một câu/từ/xâu ngôn ngữ
 - Tập $\{\epsilon\}$ và \emptyset là ngôn ngữ trên mọi bộ chữ
- Nhận xét
 - Ngôn ngữ là một tập (các xâu) \Rightarrow Có thể tạo ngôn ngữ mới từ các ngôn ngữ đã có bởi sử dụng toán tử tập

Ngôn ngữ (Languages) → Các phép toán

- Ví dụ:
 - Nếu L, L_1, L_2 : Ngôn ngữ trên cùng một bộ chữ V
 - $L_1 \cap L_2, L_1 \cup L_2, L_1 - L_2, L_1 L_2$: Ngôn ngữ trên V
 - $V^* - L, L^+, L^*$: Ngôn ngữ trên V
- Tính chất
 - $L^+ = L L^* = L^* L$
 - $L^* = L^+ \cup \{\epsilon\}$
- Câu hỏi: Các phát biểu sau đúng hay sai?
 - L là ngôn ngữ thì $L^+ = L^* - \{\epsilon\}$
 - $|L_1 L_2| = |L_1| \times |L_2|$

Văn phạm (Grammar)

$$G = (V_T, V_N, P, S)$$

- **V_T** : Tập các ký hiệu kết thúc của một bảng chữ
 - Các phần tử của V_T thường được ký hiệu: a, b, c,...
- **V_N** : Tập các k/hiệu không kết thúc của một bảng chữ
 - Các phần tử của V_N thường được ký hiệu: A, B, C,...
 - $V_T \cap V_N = \emptyset$; $V = V_T \cup V_N$: Bộ chữ của văn phạm
- **S** : Ký hiệu bắt đầu. $S \in V_N$
- **P** : Tập hữu hạn các cặp (α, β) được gọi là các quy tắc hay các sản xuất. Thường được viết $\alpha \rightarrow \beta$
 - $\alpha \in V^* V_N V^* // \rightarrow$ Phải có ít nhất một ký hiệu không k/thúc
 - $\beta \in V^* // \rightarrow$ Có thể chứa xâu rỗng

Văn phạm \rightarrow Ví dụ

Cho văn phạm

$$G_1 = (V_T, V_N, P, S)$$

Trong đó

- $V_T = \{a, b, c\}$
- $V_N = \{S, A\}$
- $P = \{S \rightarrow aSA, S \rightarrow b, A \rightarrow bA, A \rightarrow c\}$

Ghi chú

- Chỉ cần nhắc tới tập sản xuất khi giới thiệu văn phạm

Suy dẫn (Derivation)

Cho văn phạm $G = (V_T, V_N, P, S)$

- Gọi γ viết ra δ hay δ được suy dẫn trực tiếp ra từ γ , và được ký hiệu $\gamma \Rightarrow \delta$ nếu tồn tại các xâu α, β, v, w thỏa mãn các điều kiện
 - $\gamma = \alpha v \beta$
 - $\delta = \alpha w \beta$

Do $v \rightarrow w$ là một sản xuất của P .
Thay v trong γ bằng w , sẽ được δ

 - $(v, w) \in P$ {có thể viết $v \rightarrow w \in P$ }
 - $\alpha, \beta \in V^*, v \in V^* V_N V^*, w \in V^*$
- Ví dụ với văn phạm G_1
 - $aSc \Rightarrow abc$ $\{\alpha=a, \beta=c, v=S, w=b, S \rightarrow b \in P\}$

Suy dẫn

Cho văn phạm $G = (V_T, V_N, P, S)$

- Gọi δ được suy dẫn ra từ γ , nếu tồn tại dãy các xâu $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$ thỏa mãn điều kiện

$$\gamma = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_n = \delta$$

- Nếu $n \geq 1$, áp dụng ít nhất 1 bước suy dẫn, ký hiệu: $\gamma \Rightarrow^+ \delta$
- Nếu $n \geq 0$, có thể không áp dụng bước nào, ký hiệu: $\gamma \Rightarrow^* \delta$

Suy dẫn \rightarrow Ví dụ

Xét văn phạm G_1

- $abA \Rightarrow abbA \{ \alpha=ab, \beta=\varepsilon, v=A, w=bA, A \rightarrow bA \in P \}$
- $abA \Rightarrow abbA \Rightarrow abbbA \Rightarrow abbbbA \Rightarrow abbbbc$
 $\{ \alpha_0=abA, \alpha_1=abbA, \dots, \alpha_4=abbbbc \}$

Vậy

$$abA \Rightarrow^* abbbbc$$

$$abA \Rightarrow^+ abbbbc$$

$$abA \Rightarrow^* abA // \leftarrow \text{Áp dụng 0 bước suy dẫn}$$

$$abA \Rightarrow^+ abA // \leftarrow \text{Sai, do áp dụng ít nhất 1 suy dẫn}$$

Câu

Cho văn phạm $G = (V_T, V_N, P, S)$

- Một xâu δ được suy ra từ ký hiệu khởi đầu S , được gọi là *dạng câu* hay *dạng cú pháp*
 - δ là một dạng câu nếu $S \Rightarrow^* \delta$
- Câu là một dạng cú pháp chỉ bao gồm toàn ký hiệu kết thúc
 - δ là một câu nếu $S \Rightarrow^* \delta$ và $\delta \in V_T^*$

Câu \rightarrow Ví dụ

Ví dụ văn phạm G_1

$S \Rightarrow aSA \Rightarrow abA \Rightarrow abc$

$S \Rightarrow aSA \Rightarrow abA \Rightarrow abbA \Rightarrow abbc$

$S \Rightarrow aSA \Rightarrow abA \Rightarrow abbA \Rightarrow abbbA \Rightarrow abbbbc$

Dạng câu: $aSA, abA, abc, abbA, abbc, \dots$

Câu: $abc, abbc, abbbbc, abbbbcb, \dots$

$aabbbcc$ có phải là một câu?

Ngôn ngữ sản sinh

Cho văn phạm $G = (V_T, V_N, P, S)$

- Ngôn ngữ L được *sinh ra* từ văn phạm G , ký hiệu $L(G)$ là tập tất cả các câu của văn phạm
 - $L(G) = \{ \delta \mid \delta \in V_T^* \text{ và } S \Rightarrow^* \delta \}$
- Ví dụ văn phạm G_1
 - $L(G_1) = \{b, abc, abbc, abbbc, \dots aabbcc, \dots\}$
- $G_2 = (\{a, b\}, \{S, A\}, \{S \rightarrow aA, A \rightarrow a, A \rightarrow b\}, S)$
 - $L(G_2) = \{aa, ab\}$

Quy ước

Nếu $A \rightarrow \alpha_1$

$A \rightarrow \alpha_2$

.....

$A \rightarrow \alpha_n$

Được viết gọn lại thành $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$

Ví dụ:

$G_1 = (\{a, b, c\}, \{S, A\}, \{S \rightarrow aSA \mid b, A \rightarrow bA \mid c\}, S)$

$G_2 = (\{a, b\}, \{S, A\}, \{S \rightarrow aA, A \rightarrow a \mid b\}, S)$

Ngôn ngữ → Ví dụ

Cho văn phạm $G = (V_T, V_N, P, S)$

- $V_T = \{\text{Mơ, Mận, thì, uống, nhanh, đẹp}\}$
- $V_N = \{\langle \text{Câu} \rangle, \langle \text{Danh từ} \rangle, \langle \text{Động từ} \rangle, \langle \text{Tính từ} \rangle\}$
- $S : \text{Câu}$
- $P = \{ \begin{array}{l} \langle \text{Câu} \rangle \rightarrow \langle \text{Danh từ} \rangle \langle \text{Động từ} \rangle \langle \text{Tính từ} \rangle, \\ \langle \text{Danh từ} \rangle \rightarrow \text{Mơ} \mid \text{Mận}, \\ \langle \text{Động từ} \rangle \rightarrow \text{uống} \mid \text{thì}, \\ \langle \text{Tính từ} \rangle \rightarrow \text{nhanh} \mid \text{đẹp} \end{array} \}$

$L(G) = \{\text{Mơ uống nhanh, Mơ thì nhanh, Mơ thì đẹp,}$
Mơ uống đẹp, Mận thì nhanh, ...}

Đệ quy

Cho văn phạm $G = (V_T, V_N, P, S)$

$A \in V_N // \leftarrow A$ là một ký hiệu không kết thúc

- A được gọi là đệ quy nếu tồn tại

$$A \Rightarrow^+ \alpha A \beta, \text{ với } \alpha, \beta \in V^*$$

- Nếu $\alpha = \varepsilon$, $(A \Rightarrow^+ A \beta)$ thì gọi là *đệ quy trái*
 - Nếu $A \Rightarrow A \beta$: Đệ quy **trái** trực tiếp (*Sản xuất: $A \rightarrow A \beta$*)
- Nếu $\beta = \varepsilon$, $(A \Rightarrow^+ \alpha A)$ thì gọi là *đệ quy phải*
 - Nếu $A \Rightarrow \alpha A$: Đệ quy **phải** trực tiếp (*Sản xuất: $A \rightarrow \alpha A$*)
- Nếu $\alpha, \beta \neq \varepsilon$, $(A \Rightarrow^+ \alpha A \beta)$ thì gọi là *đệ quy trong*
 - Nếu $A \Rightarrow \alpha A \beta$: Đệ quy trong trực tiếp (*Sản xuất: $A \rightarrow \alpha A \beta$*)

Ví dụ, định nghĩa Tên

$$\langle \text{Tên} \rangle \rightarrow \langle \text{Chữ cái} \rangle | \langle \text{Tên} \rangle \langle \text{Chữ cái} \rangle | \langle \text{Tên} \rangle \langle \text{Chữ số} \rangle$$

Văn phạm tương đương

Hai văn phạm là tương đương, nếu chúng cùng sinh ra một ngôn ngữ

$$G_1 = (^1V_T, ^1V_N, P_1, S_1)$$

$$G_2 = (^2V_T, ^2V_N, P_2, S_2)$$

$$L(G_1) = L(G_2) \Rightarrow G_1 \Leftrightarrow G_2$$

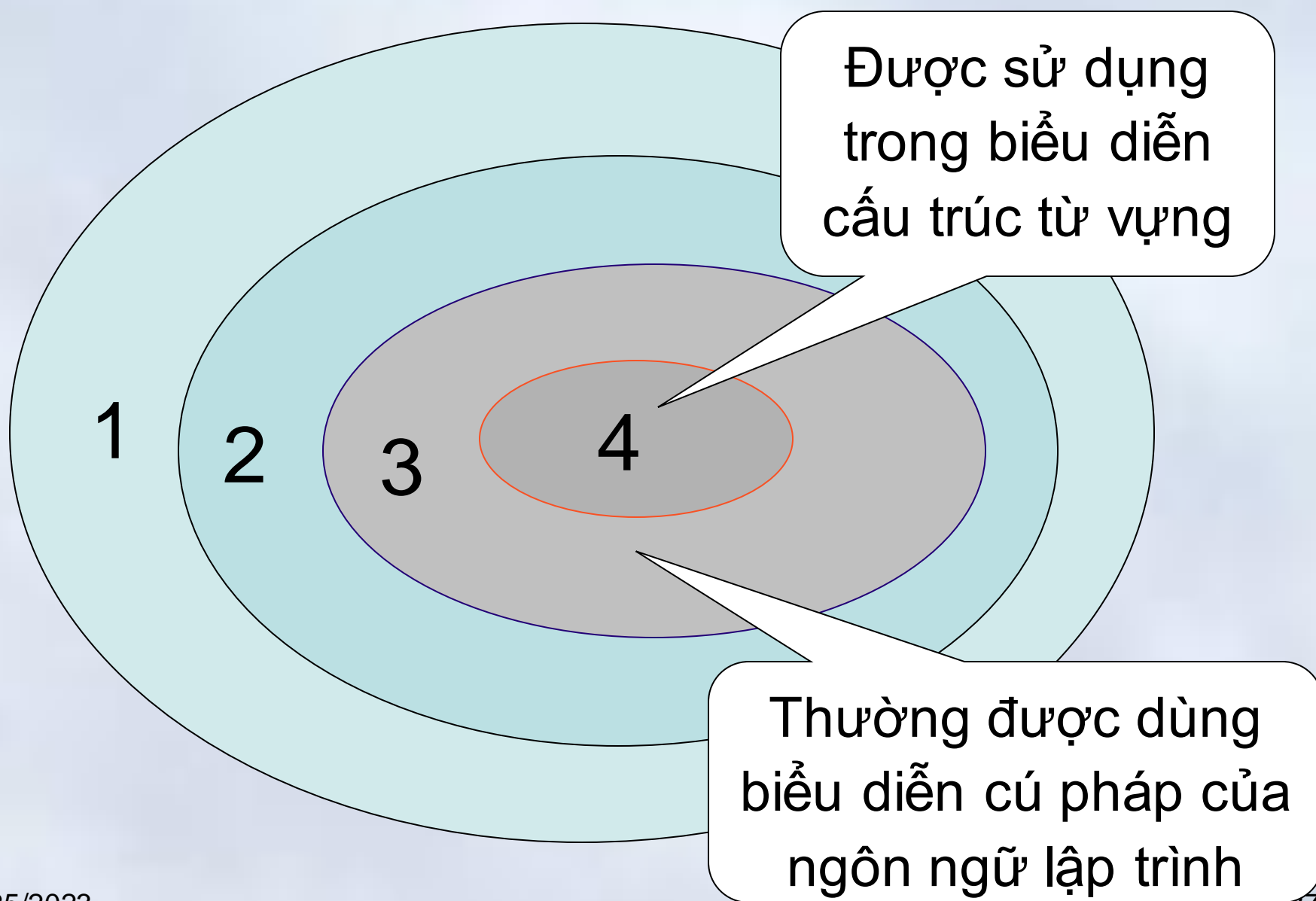
Phân loại văn phạm

Phân thành 4 loại [Chomsky, 1957]

1. Văn phạm ngữ cấu (*phrase structure grammar*)
 1. **Ràng buộc:** Không có ràng buộc ngoài đ/nghĩa
2. Văn Phạm cảm ngữ cảnh (*context sensitive*)
 - **Ràng buộc:** Nếu $\alpha \rightarrow \beta$ thì $l(\alpha) \leq l(\beta)$; $S \rightarrow \varepsilon$ được chấp nhận khi S không là vế phải của SX bất kỳ
3. Văn phạm phi ngữ cảnh (*context free grammar*)
 - **Ràng buộc:** Vế trái của các sản xuất là một ký hiệu không kết thúc
4. Văn phạm chính quy (*regular grammar*)

Ràng buộc: SX có dạng $A \rightarrow a|aB$ hoặc $A \rightarrow a|Ba$

Phân loại văn phạm



Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Văn phạm phi ngữ cảnh

- Cây suy dẫn
- Suy dẫn trái, suy dẫn phải
- Văn phạm đơn nghĩa
- Sản xuất đệ quy
- Ký hiệu vô ích
- Vấn đề phân tích cú pháp

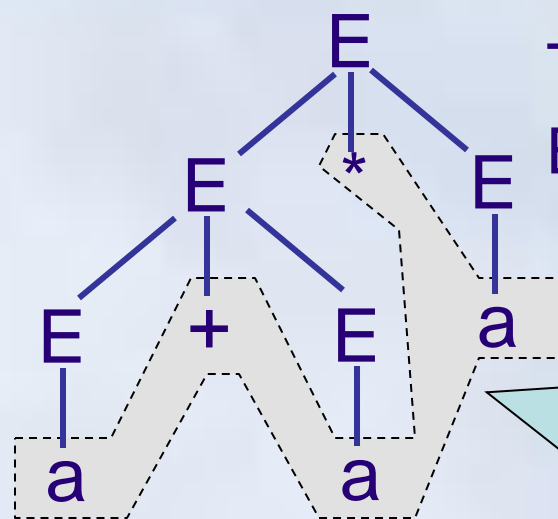
Cây suy dẫn

- $G = (V_T, V_N, P, S)$ là văn phạm phi ngữ cảnh
- Cây suy dẫn D cho VP G là cây có thứ tự, trong đó
 - Mỗi một nút có nhãn là ký hiệu trong tập $V_T \cup V_N \cup \{\epsilon\}$
 - Nhãn của nút gốc là S (*Start Symbol*)
 - Các nút lá có nhãn là *một ký hiệu kết thúc* hoặc ϵ
 - Nếu A là nút trong của D và X_1, X_2, \dots, X_n là các hậu duệ trực tiếp của A theo tự trái qua phải thì
 - $A \rightarrow X_1 X_2 \dots X_n$ là một sản xuất của P
 - Nếu $n=0$ ($X=\epsilon$) thì X là hậu duệ đơn và duy nhất của A và $A \rightarrow \epsilon$ là một sản xuất trong P ,
- **Biên** (*kết quả*) của cây suy dẫn: Từ thu được bằng cách nối các nút lá lại theo thứ tự từ trái qua phải
 - Rõ ràng, nếu α là biên của D thì $S \Rightarrow^* \alpha$

Cây suy dẫn→Ví dụ

Văn phạm: $E \rightarrow E + E \mid E^* E \mid (E) \mid a$

Xét xâu: $\omega = a + a^* a$

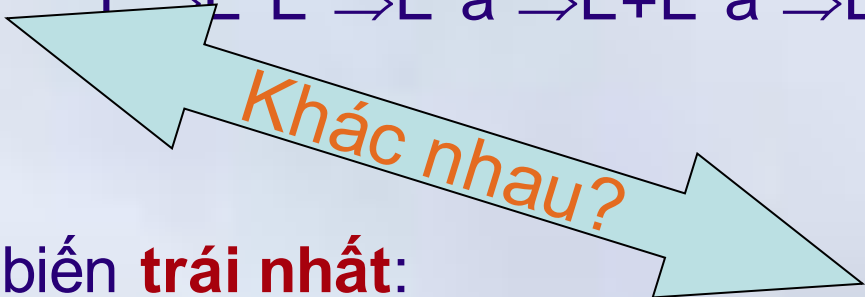


Thay thế các biến **trái nhất**:

$E \Rightarrow E^* E \Rightarrow E + E^* E \Rightarrow a + E^* E \Rightarrow a + a^* E \Rightarrow a + a^* a$

Thay thế các biến **phải nhất**:

$E \Rightarrow E^* E \Rightarrow E^* a \Rightarrow E + E^* a \Rightarrow E + a^* a \Rightarrow a + a^* a$

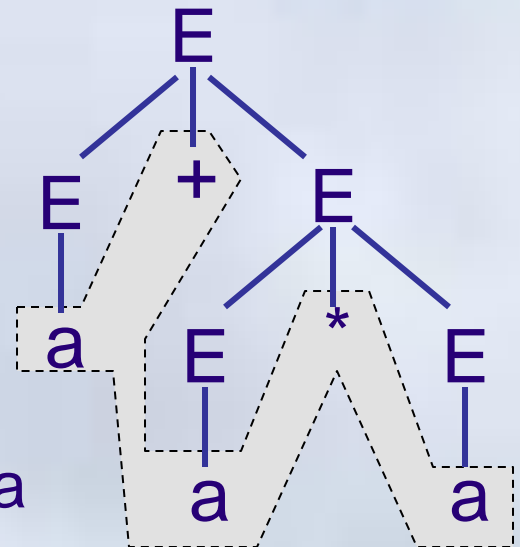


Thay thế các biến **trái nhất**:

$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E^* E \Rightarrow a + a^* E \Rightarrow a + a^* a$

Thay thế các biến **phải nhất**:

$E \Rightarrow E + E \Rightarrow E + E^* E \Rightarrow E + E^* a \Rightarrow E + a^* a \Rightarrow a + a^* a$



Suy dẫn trái

Cho văn phạm $G = (V_T, V_N, P, S)$

- Gọi δ được suy dẫn *trực tiếp* ra từ bên *trái* của γ , và được ký hiệu $\gamma \Rightarrow_L \delta$ nếu tồn tại các xâu x, α, β thỏa mãn các điều kiện
 - $\gamma = xA\alpha \Rightarrow x\beta\alpha = \delta$
 - $A \rightarrow \beta \in P$, và
 - $\alpha, \beta \in V^*, x \in V_T^*$
 Luôn thay ký hiệu không kết thúc *bên trái nhất* của xâu
- Gọi δ được suy dẫn ra từ bên trái của γ nếu tồn tại dãy các xâu $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$ thỏa mãn

$$\gamma = \alpha_0 \Rightarrow_L \alpha_1 \Rightarrow_L \alpha_2 \Rightarrow_L \dots \Rightarrow_L \alpha_n = \delta$$

- Nếu $n \geq 1$, dùng ít nhất một suy dẫn, ký hiệu: $\gamma \Rightarrow_L^+ \delta$
- Nếu $n \geq 0$, có thể không suy dẫn nào, ký hiệu: $\gamma \Rightarrow_L^* \delta$

Suy dẫn phải

Cho văn phạm $G = (V_T, V_N, P, S)$

- Gọi δ được suy dẫn *trực tiếp* ra từ bên *phải* của γ , và được ký hiệu $\gamma \Rightarrow_R \delta$ nếu tồn tại các xâu y, α, β thỏa mãn các điều kiện
 - $\gamma = \alpha A y \Rightarrow \alpha \beta y = \delta$
 - $A \rightarrow \beta \in P$, và
 - $\alpha, \beta \in V^*, y \in V_T^*$
 Luôn thay ký hiệu không kết thúc *bên phải nhất* của xâu
- Gọi δ được suy dẫn ra từ bên phải của γ nếu tồn tại dãy các xâu $\alpha_0, \alpha_1, \alpha_2, \dots, \alpha_n$ thỏa mãn:

$$\gamma = \alpha_0 \Rightarrow_R \alpha_1 \Rightarrow_R \alpha_2 \Rightarrow_R \dots \Rightarrow_R \alpha_n = \delta$$

- Nếu $n \geq 1$, dùng ít nhất một suy dẫn, ký hiệu: $\gamma \Rightarrow_R^+ \delta$
- Nếu $n \geq 0$, có thể không suy dẫn nào, ký hiệu: $\gamma \Rightarrow_R^* \delta$

Văn phạm đơn nghĩa

Văn phạm PNC $G=(V_T, V_N, P, S)$ đơn nghĩa nếu

- Với mỗi câu $\omega \in L(G)$ chỉ có đúng một suy dẫn trái (*hoặc một suy dẫn phải*) \Leftrightarrow **Chỉ có đúng một cây suy dẫn**

Ngược lại, *nếu có nhiều hơn một suy dẫn trái, hoặc một suy dẫn phải hoặc có nhiều cây suy dẫn \Rightarrow Văn phạm nhập nhằng*

Nếu $\omega \in L(G)$ có nhiều hơn 2 cây suy dẫn

\Rightarrow Có thể phân tích cú pháp theo nhiều hơn 2 cách

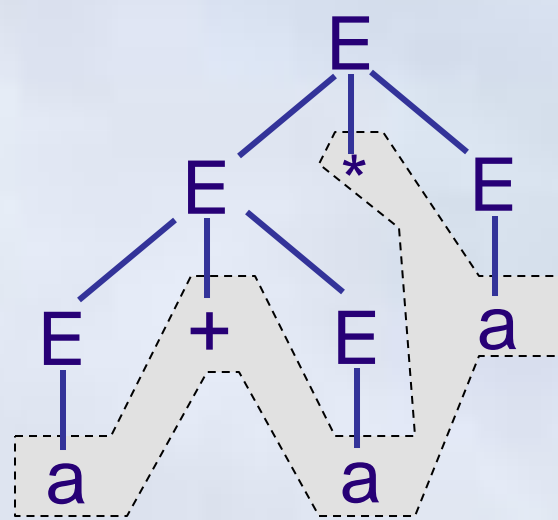
\Rightarrow Nhiều hơn 2 cách hiểu

Văn phạm đơn nghĩa→Ví dụ 1

Văn phạm: $E \rightarrow E + E \mid E * E \mid (E) \mid a$

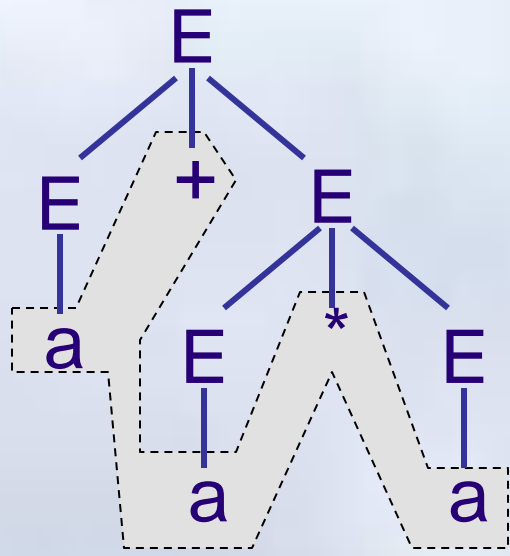
Xâu: $\omega = a + a * a$ là biên của 2 cây suy dẫn

⇒ Văn phạm nhập nhằng



$E = 8$

$a = 2$



$E = 6$

Văn phạm đơn nghĩa \rightarrow Ví dụ 2

$$G=(V_T, V_N, P, S)$$

$$V_T = \{\text{if, then, else}\}$$

$$V_N = \{\langle \text{Cond} \rangle, \langle \text{Stmt} \rangle\}$$

$$S: \langle \text{Stmt} \rangle$$

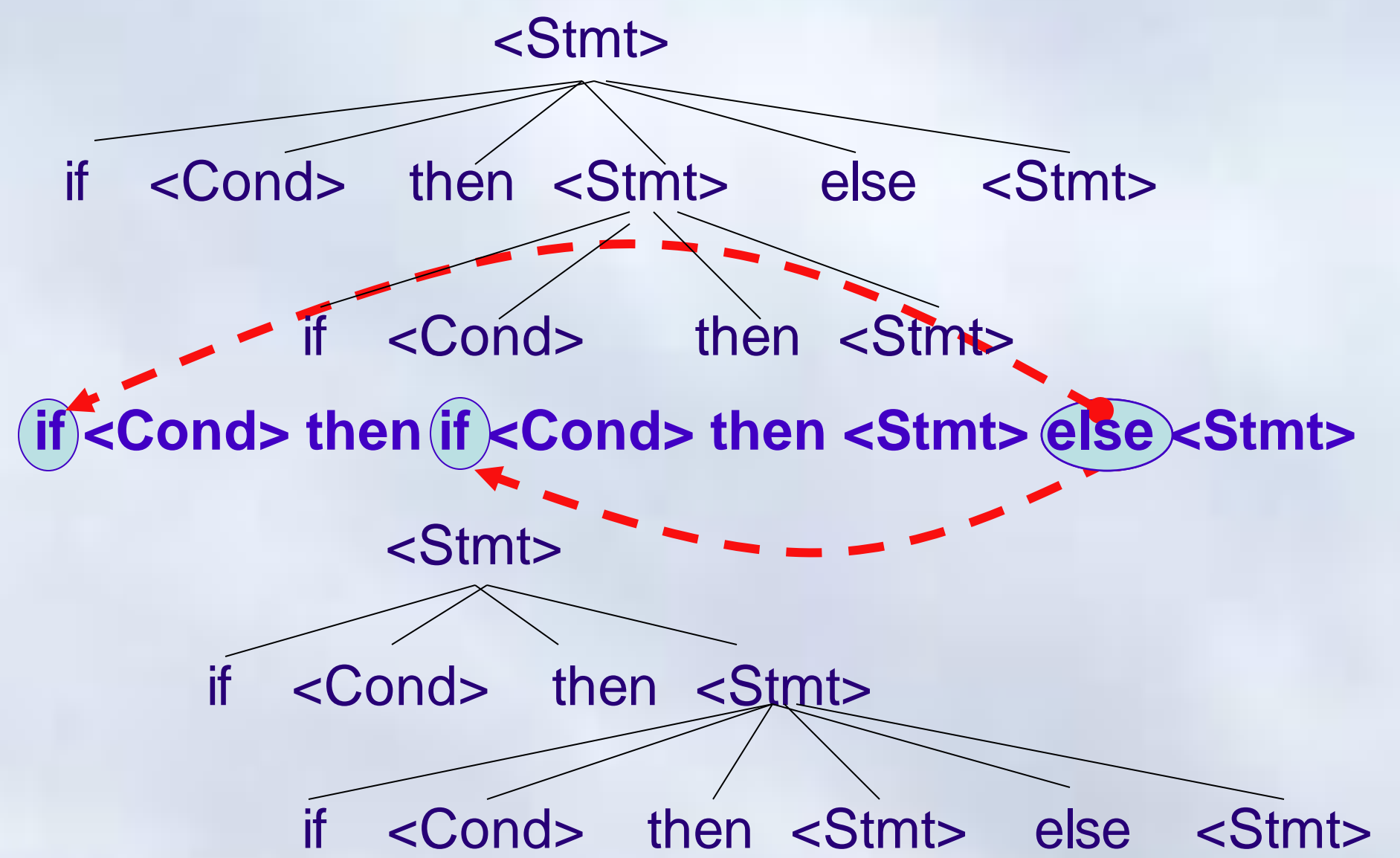
$$P = \{ \langle \text{Stmt} \rangle \rightarrow \text{if } \langle \text{Cond} \rangle \text{ then } \langle \text{Stmt} \rangle$$

$$\langle \text{Stmt} \rangle \rightarrow \text{if } \langle \text{Cond} \rangle \text{ then } \langle \text{Stmt} \rangle \text{ else } \langle \text{Stmt} \rangle \}$$

Xét dạng câu

if $\langle \text{Cond} \rangle$ then if $\langle \text{Cond} \rangle$ then $\langle \text{Stmt} \rangle$ else $\langle \text{Stmt} \rangle$

Văn phạm đơn nghĩa → Ví dụ 2



Văn phạm đơn nghĩa → Khử nhập nhằng

Đưa thêm các ký hiệu không kết thúc và các sản xuất để được các văn phạm tương đương

$$E \rightarrow E + E \mid E * E \mid (E) \mid a$$

$$\begin{aligned} E &\rightarrow E + T \mid E * T \mid T \\ T &\rightarrow (E) \mid a \end{aligned}$$

Luôn thực hiện từ trái qua phải trừ khi gặp biểu thức trong ngoặc

Phép nhân có độ ưu tiên cao hơn phép cộng khi không gặp biểu thức trong ngoặc

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

Sản xuất đệ quy

- Sản xuất là đệ quy có dạng:

$$A \rightarrow \alpha A \beta, \alpha, \beta \in V^*$$

- Dùng để biểu diễn các quá trình lặp hay cấu trúc lồng nhau

- Đệ quy trái: $A \rightarrow b|Aa$

$$A \Rightarrow Aa \Rightarrow Aaa \Rightarrow Aaaa \Rightarrow baaaa \dots$$

- Đệ quy phải: $A \rightarrow b|aA$

$$A \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaaab \dots$$

- Đệ quy giữa $A \rightarrow b|aAb$

$$\text{VD: } A \rightarrow b|\{A\}$$

$$A \Rightarrow \{A\} \Rightarrow \{\{A\}\} \Rightarrow \{\{\{A\}\}\} \Rightarrow \{\{\{b\}\}\}$$

Sản xuất đệ quy \rightarrow Khử đệ quy trái

Khử đệ quy trái bằng cách thêm ký hiệu không kết thúc và sản xuất mới để được văn phạm tương đương

$$\begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \varepsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \varepsilon \\ F &\rightarrow (E) \mid a \end{aligned}$$

Ký hiệu vô ích

Cho văn phạm PNC $G = (V_T, V_N, P, S)$

Ký hiệu $X \in V$ được gọi là **có ích** nếu tồn tại

$$S \Rightarrow^* \alpha X \beta \Rightarrow^* \omega \text{ với } \alpha, \beta \in V^* \text{ và } \omega \in V_T^*$$

Ngược lại, ký hiệu X là **vô ích**.

Như vậy để X là ký hiệu có ích cần phải

- X là đến được (*từ ký hiệu khởi đầu*)
- X là hữu sinh

Ký hiệu vô ích \rightarrow Ví dụ

$$G = (V_T, V_N, P, S)$$

$$V_T = \{a, b, c, d\}$$

$$V_N = \{S, A, B, C, D\}$$

$$P = \{ \begin{array}{l} S \rightarrow aAb \mid \varepsilon \\ A \rightarrow aS \mid bA \mid aB \\ B \rightarrow cSB \mid aBSC \\ C \rightarrow a \mid aS \\ D \rightarrow aS \mid bA \mid c \end{array} \}$$

$$V_T : \{a, b\}$$

$$V_N : \{S, A\}$$

$$P : \{ \begin{array}{l} S \rightarrow aAb \mid \varepsilon \\ A \rightarrow aS \mid bA \end{array} \}$$

Ký hiệu không đến được: **D, c**

Ký hiệu vô sinh: **B, C**

Vấn đề phân tích cú pháp (1/4)

- Cho văn phạm $G = (V_T, V_N, P, S)$

$$L(G) = \{\omega \mid \omega \in V_T^* \text{ và } S \Rightarrow^* \omega\}$$

- Nhiệm vụ của phân tích cú pháp là xem xét một xâu có thuộc ngôn ngữ được sản sinh bởi văn phạm không

$$\text{Cho } x \in V_T^*; \text{ } S \Rightarrow^* x ?$$

- Nếu xâu x được đoán nhận, cần chỉ ra các sản xuất đã sử dụng để sinh ra
 - Cấu trúc lên cây suy dẫn
- Tồn tại 2 phương pháp **trên xuống/dưới lên**

Vấn đề phân tích cú pháp (2/4)

- Trên xuống (Top-down)
 - Xuất phát từ ký hiệu khởi đầu S đi dần tới nút lá (xâu cần phân tích), bằng cách áp dụng liên tục các sản xuất
- Dưới lên (Bottom-up)
 - Xâu cần phân tích được xem xét từ trái qua phải và tìm cách thu gọn về ký hiệu khởi đầu S bằng cách áp dụng các sản xuất

Vấn đề phân tích cú pháp (3/4)

Xét văn phạm: $E \rightarrow E+T \mid T$

$T \rightarrow T^*F \mid F$

$F \rightarrow (E) \mid a$

Xâu cần phân tích ω : $a+a^*a$

Top-Down

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T^*F \Rightarrow$$

$$a+F^*F \Rightarrow a+a^*F \Rightarrow a+a^*a$$

Bottom-Up

$$a+a^*a \Leftarrow F+a^*a \Leftarrow T+a^*a \Leftarrow E+a^*a \Leftarrow E+F^*a \Leftarrow$$

$$E+T^*a \Leftarrow E+T^*F \Leftarrow E+T \Leftarrow E$$

Vấn đề phân tích cú pháp (4/4)

- Phân tích quay lui

- Tại mỗi bước có nhiều sản xuất để lựa chọn
- Nếu lựa chọn nhầm thực hiện quay lui

Phân tích quay lui tăng thời gian thực hiện

- Phân tích tất định

- Tại mỗi bước xác định duy nhất một sản xuất
- Yêu cầu văn phạm cần có tính chất nhất định
- Các kỹ thuật

- Trên xuống: Phân tích LL(k), Độ quy trên xuống,..
- Dưới lên: Thao tác trước, Phân tích LR(k),..

Chương 1: Những khái niệm cơ bản

1. Ngôn ngữ lập trình cấp cao và trình dịch
2. Đặc trưng của ngôn ngữ lập trình cấp cao
3. Các giai đoạn chính của chương trình dịch
4. Khái niệm ngôn ngữ
5. Văn phạm phi ngữ cảnh
6. Giới thiệu ngôn ngữ PL/0 mở rộng

Dạng chuẩn BNF

- Siêu ngữ (metalanguage)
 - Ngôn ngữ sử dụng các lệnh để mô tả ngôn ngữ khác
- *Dạng chuẩn BNF*
 - *Backus-Normal-Form/Backus-Naur-Form*
 - Là dạng siêu cú pháp để mô tả các ngôn ngữ lập trình
 - BNF được sử dụng rộng rãi để mô tả văn phạm của các ngôn ngữ lập trình, tập lệnh và các giao thức truyền thông.

Ký pháp BNF

Là một tập các luật thỏa mãn:

- Vế trái của mỗi luật là một cấu trúc cú pháp.
- Tên cấu trúc cú pháp là ký hiệu không kết thúc.
- Ký hiệu không kết thúc được bao trong cặp $\langle \rangle$
- Ký hiệu kết thúc được phân cách bằng cặp nháy đơn hoặc nháy kép
- Mỗi ký hiệu không kết thúc được định nghĩa bằng một hay nhiều luật.
- Các luật có dạng $N ::= s$
 - N là ký hiệu không kết thúc,
 - s là một xâu gồm 0 hay nhiều ký hiệu kết thúc và không kết thúc.
- Các luật có chung vế trái được phân cách bằng $/$

Ký pháp BNF → Ví dụ

- Mô tả cách viết một số nguyên, thực dấu phẩy tĩnh

$$\langle \text{Số} \rangle ::= '-' \langle \text{Số thập phân} \rangle | \langle \text{số thập phân} \rangle$$

$$\langle \text{Số thập phân} \rangle ::= \langle \text{Dãy chữ số} \rangle | \langle \text{Dãy chữ số} \rangle '.' \langle \text{Dãy chữ số} \rangle$$

$$\langle \text{Dãy chữ số} \rangle ::= \langle \text{Chữ số} \rangle | \langle \text{Chữ số} \rangle \langle \text{Dãy chữ số} \rangle$$

$$\langle \text{Chữ số} \rangle ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'$$

Ký pháp EBNF (Extended BNF)

Được phát triển từ BNF, có ký pháp tương tự nhưng được đơn giản hoá bằng cách dùng một số ký hiệu đặc biệt :

- Không cần dùng ‘ ’ cho ký hiệu kết thúc
- Dùng **[]** để chỉ phần bên trong là tùy chọn (có hoặc không)
- Dùng **{ }** phần bên trong có thể lặp lại một số lần tùy ý hoặc không xuất hiện lần nào
 - Nếu lặp lại **m** hay **n** lần , dùng **m** hay **n** là chỉ số trên hoặc dưới
- Dùng **(|)** cho biết có thể lựa chọn 1 trong các ký hiệu nằm bên trong

Ký pháp EBNF \rightarrow Ví dụ

- $S \rightarrow a \{b\} \quad \Leftrightarrow \quad S \rightarrow a |ab|abb|....$
- $S \rightarrow [a]\alpha \quad \Leftrightarrow \quad S \rightarrow \alpha |a \alpha$
- $S \rightarrow (a|b|c)\alpha \quad \Leftrightarrow \quad S \rightarrow a\alpha | b\alpha | c\alpha$

Trong BNF

BNF $>$ EBNF

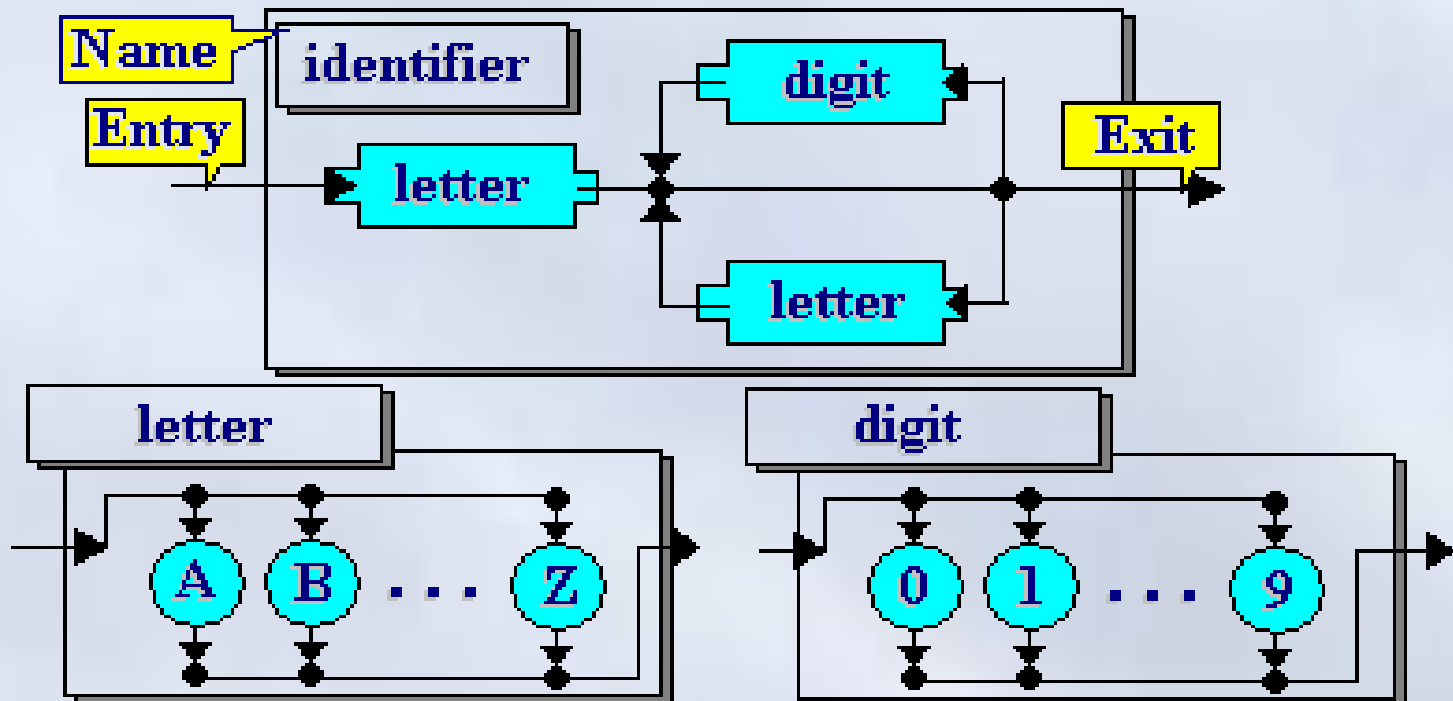
$\langle \text{Lệnh if} \rangle ::= \text{'IF' } \langle \text{Biểu thức} \rangle \text{'THEN' } \langle \text{Lệnh} \rangle | \text{'IF' } \langle \text{Biểu thức} \rangle \text{'THEN' } \langle \text{Lệnh} \rangle \text{'ELSE' } \langle \text{Lệnh} \rangle$

Trong EBNF

$\langle \text{Lệnh if} \rangle ::= \text{IF } \langle \text{Biểu thức} \rangle \text{'THEN' } \langle \text{Lệnh} \rangle [\text{'ELSE' } \langle \text{Lệnh} \rangle]$

Sơ đồ cú pháp

- Là công cụ để mô tả cú pháp của ngôn ngữ lập trình dưới dạng đồ thị
- Mỗi sơ đồ cú pháp là một đồ thị định hướng với lối vào và lối ra xác định.
- Mỗi sơ đồ cú pháp có một tên duy nhất



Ngôn ngữ PL/0

- Được giới thiệu bởi *Niklaus Wirth* năm 1975
 - Trong quyển *Algorithms + Data Structures = Programs*
- Là ngôn ngữ lập trình phục vụ giáo dục
 - Tựa Pascal, chứa đặc trưng của một NNLT cấp cao

Đặc trưng cơ bản

- Từ vựng:
 - **Chữ cái:** a..z, A..Z ; **Chữ số:** 0..9
 - Dấu đơn: + - * / % () [] > < = , ; .
 - Dấu kép: := >= <= <>
 - Từ khóa: 15 từ khóa : Begin, Call, Const, Do, End, Else, For, If, Odd, Procedure, Program, Then, To, Var, While
 - Tên: Bắt đầu bởi chữ cái, tiếp theo là tổ hợp chữ cái /chữ số. Độ dài tối đa 10. Nếu vượt quá sẽ bị bỏ qua

Ngôn ngữ PL/0

Đặc trưng cơ bản (tiếp)

- Hỗ trợ kiểu dữ liệu cơ bản *kiểu nguyên*
 - Nếu hằng số nguyên có hơn 6 chữ \Rightarrow Báo lỗi **số quá lớn**
 - PL/0 mở rộng có cấu trúc *mảng một chiều các số nguyên*
- Biểu thức số học, biểu thức so sánh
- Cho phép định nghĩa hằng (**const**), biến (**var**)
- Câu lệnh gán (**:=**) , gộp (**begin end**), gọi thủ tục (**call**), rẽ nhánh (**if..else**), lặp (**while, for**)
- Câu lệnh vào/ra (*read/readln/write/writeln*)
- Có cấu trúc khối – chương trình con dạng thủ tục
 - Các thủ tục có thể lồng nhau
 - PL/0 mở rộng cho phép truyền tham số theo trị, theo biến

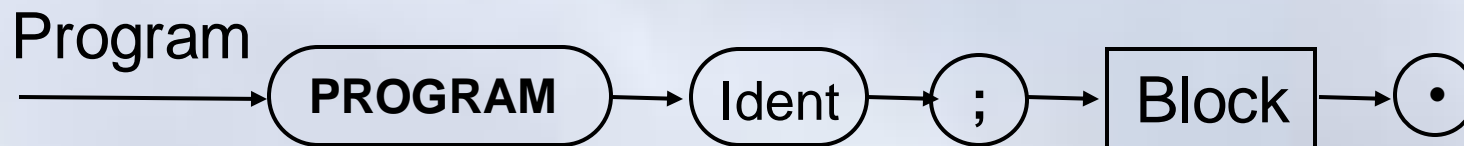
Sơ đồ cú pháp cho ngôn ngữ PL/0 mở rộng

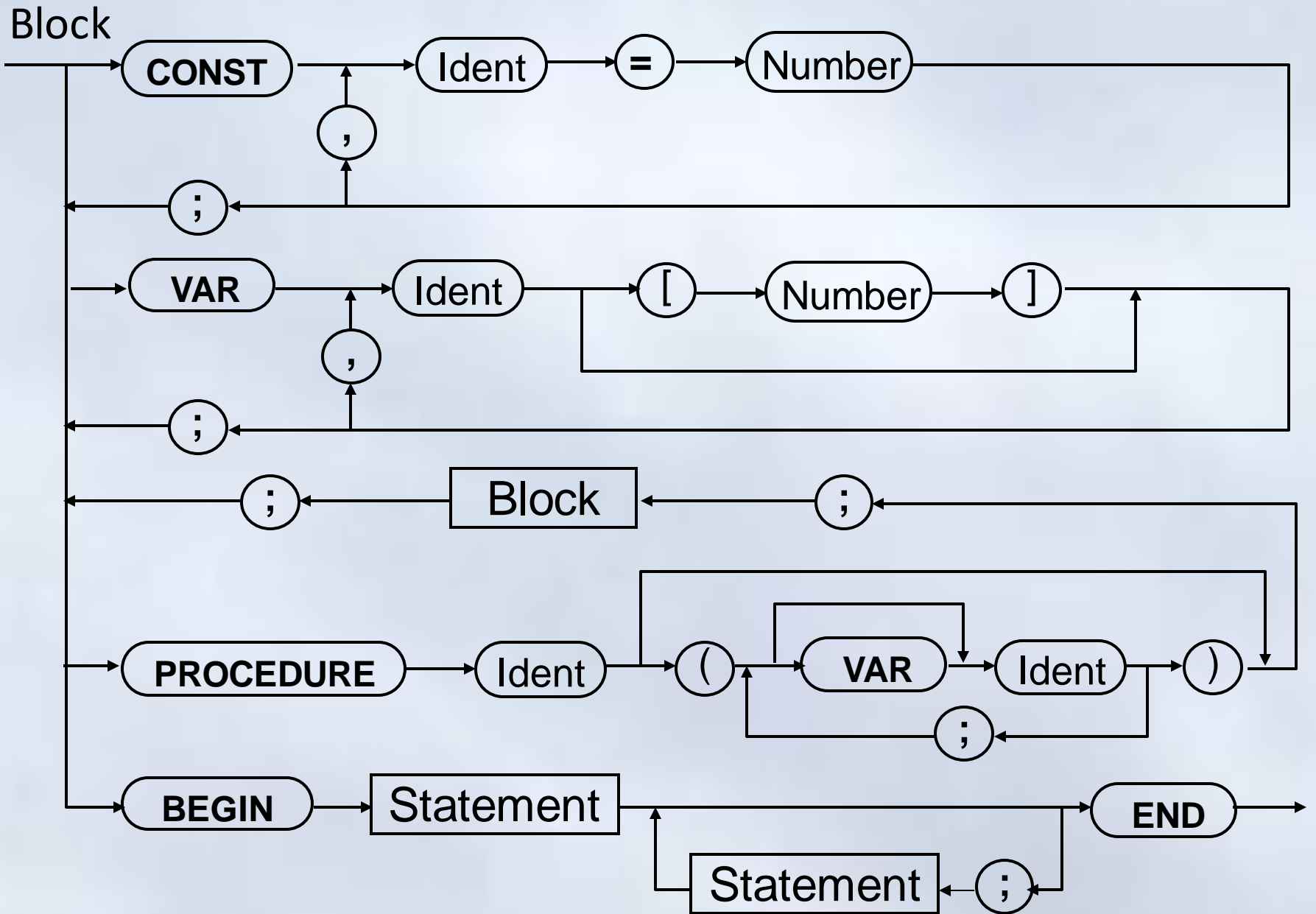


Dạng EBNF

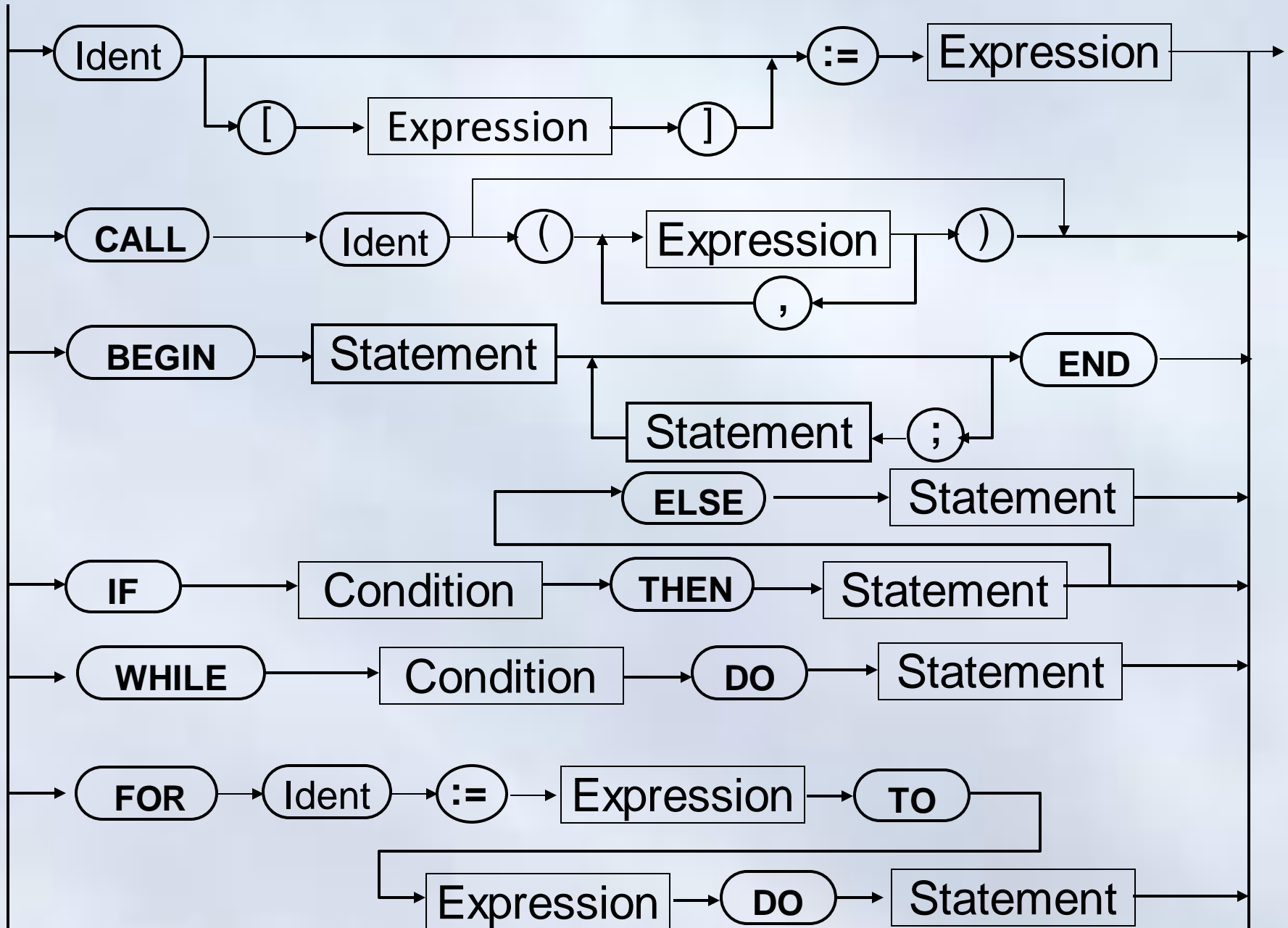
$\langle \text{program} \rangle ::= \text{PROGRAM } \langle \text{Identifier} \rangle ; \langle \text{Block} \rangle .$

Dạng sơ đồ

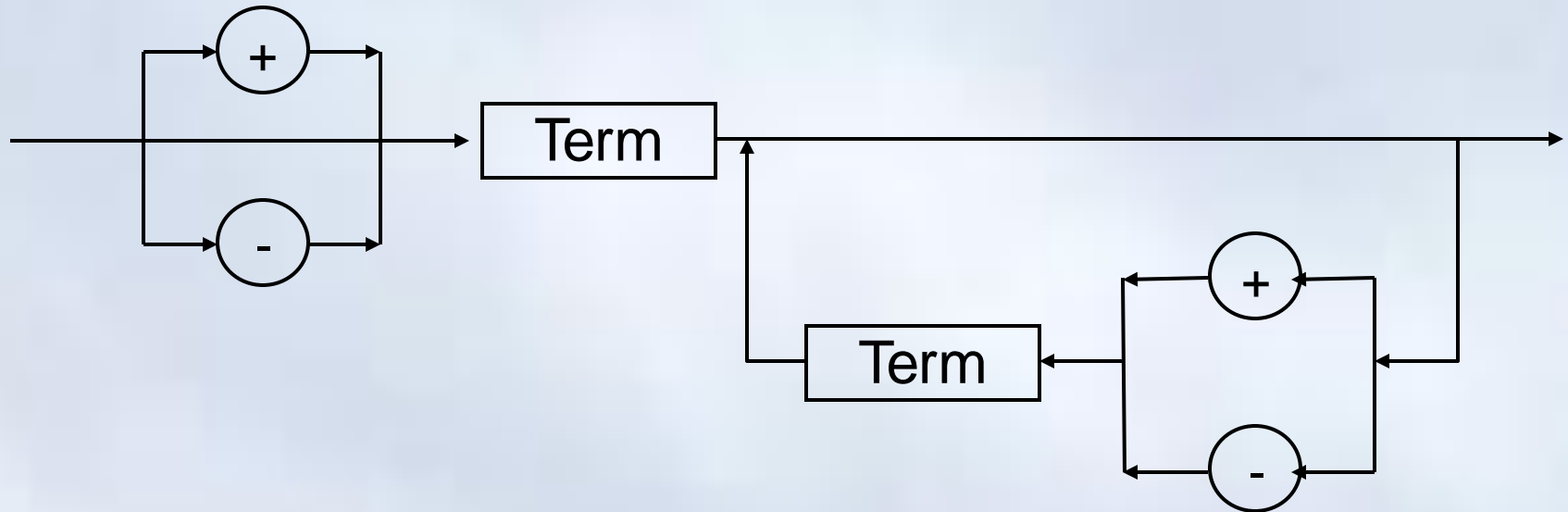




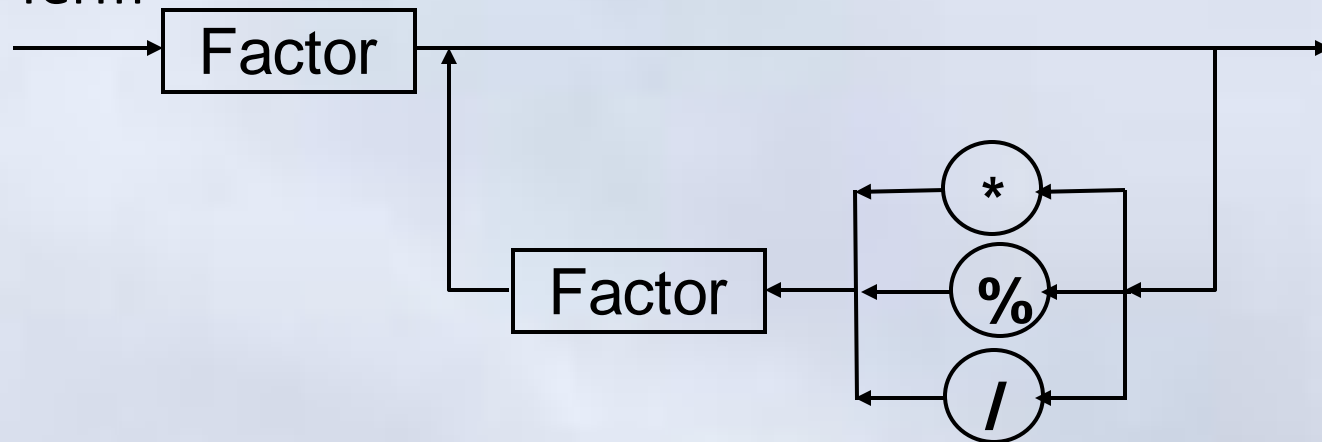
Statement



Expression



Term



Ví dụ viết trên ngôn ngữ PL/0 mở rộng

Program Example2;

Var a,b,c;

Procedure gcd(a; b; var d);

Begin

While a <> b do

if a > b then a := a – b

else b := b – a;

d := b;

End;

Begin

Call Readln(a); Call Readln(b);

Call gcd(a,b,c);

Call Writeln(c);

End

Phân tích một chương trình PL/0 mở rộng

```
Program Vidu;  
Begin  
    X := 10  
End.
```

$\omega = \text{Program Vidu ; Begin } X := 10 \text{ End .}$

$L(\omega) = 9$

$\langle \text{program} \rangle \Rightarrow^* \omega?$

Phân tích một chương trình PL/0 mở rộng

<program>

⇒ **Program** Ident ; <block> .

⇒ **Program Vidu** ; <block>.

⇒ **Program Vidu**; **Begin** <Statement> **End**.

⇒ **Program Vidu**; **Begin** Ident := <Expression> **End**.

⇒ **Program Vidu**; **Begin** X := <Expression> **End**.

⇒ **Program Vidu**; **Begin** X := <Term> **End**.

⇒ **Program Vidu**; **Begin** X := <Factor> **End**.

⇒ **Program Vidu**; **Begin** X:= Number **End**.

⇒ **Program Vidu**; **Begin** X:= 10 **End**

Ident(Vidu), Ident (X),
Number(10) được xác định
bởi bộ phân tích từ vựng