Bài 5.
PHÂN TÍCH NGỮ NGHĨA

Hoàng Anh Việt
Viện CNTT&TT - ĐHBKHN

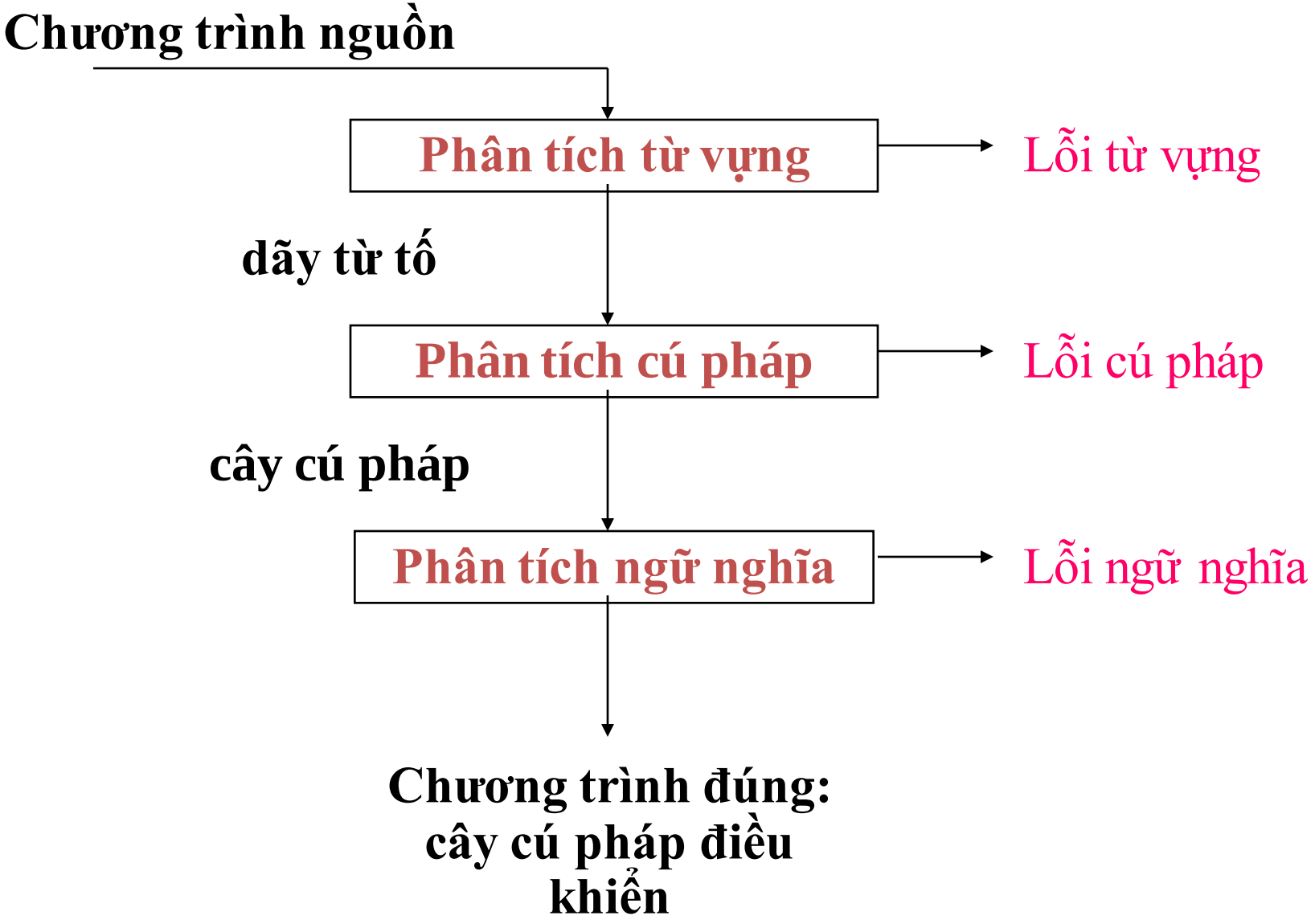
Tổng kết bài 4

- Trước đây: tự viết bộ phân tích cú pháp
- Hiện nay: sử dụng các trình sinh bộ phân tích cú pháp. VD: yacc, cup, bison
- Ưu điểm:
  - Sử dụng phương pháp phân tích LALR(1)
  - Cho phép khai báo thứ tự ưu tiên, kết hợp của các phép toán
  - Tự động sinh code phân tích cú pháp (kể cả bảng phân tích LALR(1))

Nội dung

1. Những vấn đề của ngữ nghĩa
2. Kiểm tra kiểu
   - Hệ thống kiểu trong ngôn ngữ lập trình
3. Bảng ký hiệu

Phân tích ngữ nghĩa



1. Những vấn đề của ngữ nghĩa

- Tìm ra các lỗi sau giai đoạn phân tích cú pháp
  - Kiểm tra sự tương ứng về kiểu
  - Kiểm tra sự tương ứng giữa việc sử dụng hàm, biến với khai báo của chúng.
  - Xác định phạm vi ảnh hưởng của các biến trong chương trình
- Phân tích ngữ nghĩa thường sử dụng cây cú pháp

2. Kiểm tra kiểu

- Kiểm tra xem chương trình có tuân theo các luật về kiểu của ngôn ngữ không
- Trình biên dịch quản lý thông tin về kiểu
- Việc kiểm tra kiểu được thực hiện bởi bộ kiểm tra kiểu (type checker), một bộ phận của trình biên dịch

Ví dụ kiểm tra kiểu

- Toán tử % của C chỉ thực hiện khi các toán hạng là số nguyên
- Chỉ có mảng mới có chỉ số và kiểu của chỉ số phải là số nguyên
- Một hàm phải có một số lượng tham số nhất định và các tham số phải đúng kiểu
- Các phép toán đòi hỏi các toán hạng phải phù hợp kiểu
- Các hàm đòi hỏi tham số phù hợp kiểu
- Lệnh return phải trả về đúng kiểu trả về của hàm
- Lệnh gán đòi hỏi kiểu của vế phải phù hợp với kiểu của vế trái
- Lệnh khai báo kiểu: typedef, class

2. Kiểm tra kiểu

- Có hai phương pháp tĩnh và động
- Phương pháp áp dụng trong thời gian dịch là tĩnh
- Trong các ngôn ngữ như C hay Pascal, kiểm tra kiểu là tĩnh và được dùng để kiểm tra tính đúng đắn của chương trình trước khi nó được thực hiện
- Kiểm tra kiểu tĩnh cũng được sử dụng khi xác định dung lượng bộ nhớ cần thiết cho các biến
- Bộ kiểm tra kiểu được xây dựng dựa trên:
  - Các biểu thức kiểu của ngôn ngữ
  - Bộ luật để định kiểu cho các cấu trúc

2.1 Biểu thức kiểu

- Biểu diễn kiểu của một cấu trúc ngôn ngữ
- Một biểu thức kiểu là một kiểu dữ liệu chuẩn hoặc được xây dựng từ các kiểu dữ liệu khác bởi cấu trúc kiểu (Type Constructor).
  - Kiểu dữ liệu chuẩn (int, real, boolean, char) là biểu thức kiểu
  - Biểu thức kiểu có thể liên hệ với một tên. Tên kiểu là biểu thức kiểu
  - Cấu trúc kiểu được ứng dụng vào các biểu thức kiểu tạo ra biểu thức kiểu

2.1 Biểu thức kiểu

- Mỗi ngôn ngữ lập trình có hệ thống kiểu riêng
- Mỗi kiểu là một giới hạn dữ liệu
- VD: int = $[-2^{31}, 2^{31}]$, char = [-128, 127]
- Các kiểu dữ liệu phức hợp được tạo từ các kiểu đơn giản bởi các biểu thức kiểu (type expressions, type contructors)
- VD: int, string, Array[int], Object

Ví dụ: C++

- Kiểu cơ bản: int, char, …
- Kiểu phức hợp:

  int[100], struct {int a, char b}
- Biểu thức kiểu:

  T là kiểu

  T[ ] là kiểu với mọi T

Định nghĩa kiểu

- Một số ngôn ngữ cho phép người lập trình tự định nghĩa kiểu
- VD: C++

```
typedef int int_array[ ];
class cView { … };
```

- int_array là một kiểu giống với int[ ]
- Có thể có nhiều định nghĩa kiểu của cùng một kiểu

Biểu thức kiểu: Mảng

- Mỗi ngôn ngữ có một cách định nghĩa mảng
- Mảng không giới hạn:
  - C/C++: T[ ]
- Mảng có giới hạn
  - C/C++/Java: T[L] – L phần tử kiểu T
- Mảng có giới hạn trên, dưới
  - Pascal: T[L, U] – đánh chỉ số từ L đến U
- Mảng nhiều chiều
  - C/C++/Java/Pascal

Biểu thức kiểu: Cấu trúc

- Là biểu thức kiểu khá phức tạp
- Biểu thức kiểu có dạng {id1: T1, id2: T2, …} với id và T là tên và kiểu của các trường
- Ví dụ
  - C/C++: struct { int a; float b; } tương ứng với biểu thức kiểu {a: **int**, b: **float**}
- Các kiểu lớp (Class) là mở rộng của kiểu struct (cho phép thành viên là hàm)

Biểu thức kiểu: Hàm

- Hàm cho phép nhận nhiều tham số và trả về giá trị
- Tham số thứ i có kiểu T_i, kiểu trả là T
- Biểu thức kiểu: T_1 x T_2 x … T_n → T
- Ví dụ: int f(int, char) tương ứng với biểu thức kiểu int x char → int
- Trong C++/Java, cần mở rộng biểu thức kiểu của hàm để có thể trả lại ngoại lệ

2.1 Biểu thức kiểu

(a) Mảng (Array).Nếu T là biểu thưc kiểu thì array(I,T) là biểu thức kiểu biểu diễn một mảng với các phần tử kiểu T và chỉ số trong miền I.

- Ví dụ : array [10] of integer có kiểu array(1..10,int);

(b) Tích Descarter. Nếu T1và T2 là các biểu thức kiểu thì tích Descarter T1 T2 là biểu thức kiểu.

(c) Bản ghi (Record) Tương tự như tích Descarter nhưng chứa các tên khác nhau cho các kiểu khác nhau

2.1 Biểu thức kiểu

- Ví dụ:

```
struct
{
double r;
int i;
}
```

Có kiểu ((r x double) x (i x char))

(e) Hàm Nếu D là miền xác định và R là miền giá trị của hàm thì kiểu của nó được biểu diễn là: D : R.

Ví dụ hàm của C

```
int  f(char a, b)
```

Có kiểu: char char : int.

2.2 Hệ thống kiểu

- Hệ thống kiểu là một bộ sưu tập các quy tắc để gán các biểu thức kiểu vào các phần của một chương trình.
- Được định nghĩa thông qua dịch trực tiếp cú pháp
- Bộ kiểm tra kiểu cài đặt một hệ thống kiểu.

2.3 Đặc tả bộ kiểm tra kiểu

- Văn phạm sau sinh ra một chương trình, biểu diễn bởi một ký hiệu chưa kết thúc P chứa một chuỗi các khai báo D và một biểu thức đơn giản E.

$$P \rightarrow D \; ; \; E$$

$$D \rightarrow D \; ; \; D \mid \textbf{id} \; : \; T$$

$$T \rightarrow \textbf{char} \mid \textbf{integer} \mid \textbf{array[num] of} \; T_1 \mid \uparrow T_1$$

$$E \rightarrow \textbf{literal} \mid \textbf{num} \mid \textbf{id} \mid E_1 \; \textbf{mod} \; E_2 \mid E_1 \; \textbf{[}E_2\textbf{]} \mid E_1 \uparrow$$

- Các kiểu cơ sở: char, integer và type-error
- Mảng bắt đầu từ 1. Chẳng hạn array[256] of char là biểu thức kiểu (1...256, char)
- Kiểu con trỏ ↑T là một biểu thức kiểu pointer(T).

Kiểm tra kiểu của định danh

$P \rightarrow D$ **;** E	
$D \rightarrow D$ **;** D	
$D \rightarrow$ **id :** T	*{addtype(id.entry, T.type) }*
$T \rightarrow$ **char**	*{T.type := char }*
$T \rightarrow$ **integer**	*{T.type := integer }*
$T \rightarrow \uparrow T_1$	*{T.type := pointer(T_1.type) }*
$T \rightarrow$ **array[num] of** T_1	*{T.type := array(1...num.val, T_1.type) }*

Kiểm tra kiểu của biểu thức

SẢN XUẤT	QUY TẮC NGỮ NGHĨA
$E \rightarrow$ literal	$E.type := char$
$E \rightarrow$ num	$E.type := int$
$E \rightarrow$ id	$E.type := lookup(\text{id}.entry)$
$E \rightarrow E_1$ mod E_2	$E.type :=$ if $E_1.type = int$ and $E_2.type = int$ then int else $type\_error$
$E \rightarrow E_1[E_2]$	$E.type :=$ if $E_2.type = int$ and $E_1.type = array(s,t)$ then t else $type\_error$
$E \rightarrow E_1\uparrow$	$E.type :=$ if $E_1.type = pointer(t)$ then t else $type\_error$

Kiểm tra kiểu của lệnh

SẢN XUẤT	QUY TẮC NGỮ NGHĨA
$S \rightarrow \text{id} := E$	$S.type :=$ if id.$type = E.type$ then $void$ else $type\_error$
$S \rightarrow \text{if } E \text{ then } S_1$	$S.type :=$ if $E.type = boolean$ then $S_1.type$ else $type\_error$
$S \rightarrow \text{while } E \text{ do } S_1$	$S.type :=$ if $E.type = boolean$ then $S_1.type$ else $type\_error$
$S \rightarrow S_1; S_2$	$S.type :=$ if $S_1.type = void$ and $S_2.type = void$ then $void$ else $type\_error$

Kiểm tra kiểu của hàm

SẢN XUẤT	QUY TẮC NGỮ NGHĨA
$D \rightarrow \text{id} : T$	*addtype*(id.*entry*,*T.type*); $D.type := T.type$
$D \rightarrow D_1 ; D_2$	$D.type := D_1.type \times D_2.type$
$Fun \rightarrow \text{fun id}(D) : T ; B$	*addtype*(id.*entry*,*D.type*:*T.type*)
$B \rightarrow \{S\}$	
$S \rightarrow \text{id}(EList)$	$E.type :=$ if *lookup*(id.*entry*) $= t_1 : t_2$ and $EList.type = t_1$ then t_2 else *type_error*
$EList \rightarrow E$	$EList.type := E.type$
$EList \rightarrow EList, E$	$EList.type := EList_1.type \times E.type$

Thảo luận