CHƯƠNG VI
KIỂM TRA KIỂU

Nội dung chính:
Hai cách kiểm tra kiểu là *kiểm tra tĩnh* được thực hiện trong thời gian biên dịch chương trình nguồn và *kiểm tra động* được thực hiện trong thời gian thực thi chương trình đích. Trong chương này ta tập trung vào phần xử lý ngữ nghĩa bằng cách kiểm tra tĩnh mà cụ thể là *kiểm tra kiểu*. Phần đầu của chương trình bày các khái niệm về *hệ thống kiểu*, *các biểu thức kiểu*. Phần còn lại mô tả cách tạo ra một *bộ kiểm tra kiểu* đơn giản.

Mục tiêu cần đạt:
Sau khi học xong chương này, sinh viên phải nắm được:

- Hệ thống kiểu với các biểu thức kiểu (kiểu cơ sở và kiểu có cấu trúc) thường gặp ở bất cứ một ngôn ngữ lập trình nào.
- Dịch trực tiếp cú pháp cài đặt bộ kiểm tra kiểu đơn giản từ đó có thể mở rộng để cài đặt cho những ngôn ngữ phức tạp hơn.

Kiến thức cơ bản:
Sinh viên phải biết một số ngôn ngữ lập trình cấp cao như Pascal, C++, Java, v.v hoặc đã được học môn ngôn ngữ lập trình (phần đề cập đến các kiểu cơ sở và kiểu có cấu trúc).

Tài liệu tham khảo:

[1] **Compilers : Principles, Technique and Tools** - Alfred V.Aho, Jeffrey D.Ullman - Addison - Wesley Publishing Company, 1986.

[2] **Modern Compiler Implementation in C -** Andrew W. Appel - Cambridge University Press, 1997.

[3] **Compiler Design** – Reinhard Wilhelm, Dieter Maurer - Addison - Wesley Publishing Company, 1996.

I. HỆ THỐNG KIỂU

Trong các ngôn ngữ nói chung đều có kiểu cơ sở và kiểu có cấu trúc. Chẳng hạn trang Pascal, kiểu cơ sở là: boolean, char, integer, real, kiểu miền con và kiểu liệt kê. Các kiểu có cấu trúc như mảng, mẩu tin, tập hợp, ...

1. Biểu thức kiểu

Biểu thức kiểu bao gồm:

1. Kiểu cơ sở là một biểu thức kiểu: boolean, char, integer, real. Ngoài ra còn có các kiểu cơ sở đặc biệt như: kiểu type_error: chỉ ra một lỗi trong quá trình kiểm tra kiểu; kiểu void, "không có giá trị", cho phép kiểm tra kiểu đối với lệnh.

2. Vì biểu thức kiểu có thể được đặt tên, tên kiểu là một biểu thức kiểu.

3. Cấu trúc kiểu là một biểu thức kiểu, các cấu trúc bao gồm:

a. **Mảng** (array): Nếu T là một biểu thức kiểu thì array(I, T) là một biểu thức kiểu. Một mảng có tập chỉ số I và các phần tử có kiểu T.

b. **Tích** (products): Nếu T1, T2 là biểu thức kiểu thì tích Decas T1* T2 là biểu thức kiểu.

c. **Mẩu tin** (records): Là cấu trúc bao gồm một bộ các tên trường, kiểu trường.

d. **Con trỏ** (pointers): Nếu T là một biểu thức kiểu thì pointer(T) là một biểu thức kiểu T.

e. **Hàm** (functions): Một cách toán học, hàm ánh xạ các phần tử của tập xác định (domain) lên tập giá trị (range). Một hàm là một biểu thức kiểu D → R

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. Bộ kiểm tra kiểu cài đặt một hệ thống kiểu.

3. Kiểm tra kiểu tĩnh và động

Kiểm tra được thực hiện bởi chương trình dịch được gọi là kiểm kiểu tĩnh. Kiểm tra được thực hiện trong khi chạy chương trình đích gọi là kiểm tra kiểu động.

II. ĐẶC TẢ MỘT BỘ KIỂM TRA KIỂU ĐƠN GIẢN

Trong phần này chúng ta mô tả một bộ kiểm tra kiểu cho một ngôn ngữ đơn giản trong đó kiểu của mỗi một danh biểu phải được khai báo trước khi sử dụng. Bộ kiểm tra kiểu là một lược đồ dịch mà nó tổng hợp kiểu của mỗi biểu thức từ kiểu của các biểu thức con của nó.

1. Một ngôn ngữ đơn giản

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 → D **;** E

D → D **;** D | **id :** T

T → **char** | **integer** | **array[num] of** T_1 | ↑T_1

E → **literal** | **num** | **id** | E_1 **mod** E_2 | E_1 **[**E_2**]** | E_1↑

Hình 6.1 - *Văn phạm của một ngôn ngữ đơn giản*

- 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).

Ta có lược đồ dịch để lưu trữ kiểu của một danh biểu

P → D **;** E

D → D **;** D

D → **id :** T {*addtype(id.entry, T.type)* }

T → **char** {*T.type := char* }

T → **integer** {*T.type := integer* }

T → ↑T_1 {*T.type := pointer(T_1.type)* }

T → **array[num] of** T_1 {*T.type := array(1...num.val, T_1.type)* }

Hình 6.2- *Lược đồ dịch lưu trữ kiểu của một danh biểu*

2. Kiểm tra kiểu của biểu thức

Lược đồ dịch cho kiểm tra kiểu của biểu thức như sau:

E → **literal** {*E.type := char* }

E → **num** {*E.type := integer* }

E → **id** {*E.type := lookup(id.entry)* }

E → E_1 **mod** E_2 {*E.type := if E_1.type = integer and E_2.type = integer then integer else type_error* }

E → E_1**[**E_2**]** {*E.type := if E_2.type = integer and E_1.type = array(s,t) then t else type_error* }

E → E_1↑ { *E.type := if E_1.type = pointer(t) then t else type_error* }

Hình 6.3- *Lược đồ dịch kiểm tra kiểu của biểu thức*

Ở đây ta dùng hàm lookup(e) để tìm kiểu được lưu trữ trong ô của bảng ký hiệu mà ô đó được trỏ bởi e.

3. Kiểm tra kiểu của các lệnh

Ta có lược đồ dịch cho kiểm tra kiểu của lệnh

S → **id :=** E

{ *S.type := if id.type = E.type then void else type_error* }

S → **if** E **then** S_1

{*S.type := if E.type = boolean then S_1.type else type_error* }

S → **while** E **do** S_1

{*S.type := if E.type = boolean then S_1.type else type_error* }

S → S_1 **;** S_2 {*S.type := if S_1.type = void and S_2.type = void then void else type_error* }

Hình 6.4- *Lược đồ dịch kiểm tra kiểu của các lệnh*

4. Kiểm tra kiểu của các hàm

Áp dụng hàm vào một đối số có thể được cho bởi luật sinh E → E (E). Lược đồ dịch cho kiểm tra kiểu cho một áp dụng hàm là:

E → E_1 **(**E_2**)** {*E.type := if* E_2*.type = s and* E_1*.type = s -> t then t*
else type_error }

Hình 6.5- *Lược đồ dịch kiểm tra kiểu của hàm*

Luật sinh trên biểu diễn rằng một biểu thức được hình thành áp dụng E_1 lên E_2, kiểu của E_1 phải là một hàm s -> t từ kiểu s của E_2 tới một kiểu giới hạn t nào đó; kiểu của E_1 (E_2) là t.

III. SỰ TƯƠNG ĐƯƠNG CỦA CÁC BIỂU THỨC KIỂU

Thông thường kiểm tra kiểu có dạng: "nếu hai biểu thức kiểu bằng nhau thì trả về một kiểu ngược lại trả về type_error". Điều quan trọng là cần xác định khi nào hai biểu thức kiểu tương đương.

1. Tương đương cấu trúc

Hai biểu thức kiểu được gọi là tương đương cấu trúc nếu cấu trúc của chúng giống hệt nhau.

Ví dụ 6.1:

- Biểu thức kiểu integer tương đương với integer vì chúng là một kiểu cơ sở.
- pointer(integer) tương đương với pointer(integer) vì cả hai được hình thành bằng cách áp dụng cùng một cấu trúc con trỏ pointer lên các kiểu tương đương.

Giả sử, s và t là hai biểu thức kiểu, hàm sau kiểm tra xem chúng có tương đương hay không?

Function *sequiv(s, t)* **: boolean;**

Begin

if *s và t cùng là một kiểu cơ sở* ***then***

return true

else if $s = array(s_1, s_2)$ **and** $t = array(t_1, t_2)$ **then**

return $sequiv(s_1, t_1)$ **and** $sequiv(s_2, t_2)$

else if $s = pointer(s_1)$ **and** $t = pointer(t_1)$ **then**

return $sequiv(s_1, t_1)$

else if $s = s_1 \text{ -> } s_2$ **and** $t = t_1 \text{ -> } t_2$ **then**

return $sequiv(s_1, t_1)$ **and** $sequiv(s_2, t_2)$

else return false;

end;

Hình 6.6- *Đoạn ngôn ngữ giả kiểm tra sự tương đương cấu trúc của hai biểu thức kiểu s và t*

2. Tương đương tên

Trong một số ngôn ngữ, kiểu được cho bởi tên. Ví dụ trong Pascal

```
type   link = ↑ cell;
var    next : link;
       last  : link;
       p     : ↑cell;
       q, r  : ↑cell;
```

Danh biểu link được khai báo là tên của kiểu ↑cell. Vấn đề đặt ra là next, last, p, q, r có kiểu giống nhau hay không? Câu trả lời phụ thuộc vào sự cài đặt. Hai biểu thức kiểu là tương đương tên nếu tên của chúng giống nhau. Theo quan niệm tương đương tên thì last và next có cùng kiểu; p, q và r có cùng một kiểu nhưng next và p có kiểu khác nhau.

IV. CHUYỂN ĐỔI KIỂU

Xét biểu thức x + i trong đó x có kiểu real và i có kiểu integer. Vì biểu diễn các số nguyên, số thực khác nhau trong máy tính do đó các chỉ thị máy khác nhau được dùng cho số thực và số nguyên. Trình biên dịch có thể thực hiện việc chuyển đổi kiểu để hai toán hạng có cùng kiểu khi phép toán cộng xảy ra.

Bộ kiểm tra kiểu trong trình biên dịch có thể được dùng để thêm các phép toán biến đổi kiểu vào trong biểu diễn trung gian của chương trình nguồn. Chẳng hạn ký hiệu hậu tố của x + i có thể là: **x i inttoreal real+**

Trong đó: **inttoreal** đổi số nguyên i thành số thực, real+ thực hiện phép cộng các số thực.

Sự ép buộc chuyển đổi kiểu

Sự chuyển đổi từ kiểu này sang kiểu khác được gọi là ẩn (implicit) nếu nó được làm một cách tự động bởi chương trình dịch. Chuyển đổi kiểu ẩn còn gọi là ép buộc chuyển đổi kiểu (coercions).

Ví dụ 6.2: Định nghĩa trực tiếp cú pháp cho kiểm tra kiểu và ép buộc chuyển đổi kiểu biến đổi kiểu từ integer thành real:

Luật sinh	**Luật ngữ nghĩa**
E → **num**	*E.type := integer*
E → **num.num**	*E.type := real*
E → **id**	*E.type := lookup(id.entry)*
E → E_1 **op** E_2	*E.type := if E_1.type = integer and E_2.type = integer then integer else if E_1.type = integer and E_2.type = real*

then real

else if E_1*.type = real and* E_2*.type = integer*

then real

else if E_1*.type = real and* E_2*.type = real*

then real

else type_error

Hình 6.7- *Định nghĩa trực tiếp cú pháp cho kiểm tra kiểu và ép buộc chuyển đổi kiểu*

Chý ý rằng việc ép buộc chuyển đổi kiểu có thể dẫn đến sự lãng phí thời gian thực hiện chương trình.

Ví dụ 6.3: Với khai báo x là một mảng các số thực thì lệnh for i:=1 to n do x[i]:=1 thực hiện trong 48,4 micro giây còn lệnh for i:=1 to n do x[i]:=1.0 thực hiện trong 5,4 micro giây. Sở dĩ như vậy vì mã phát sinh cho lệnh thứ nhất chứa một lời gọi thủ tục đổi số nguyên thành số thực tại thời gian thực hiện.

BÀI TẬP CHƯƠNG VI

6.1. Viết các biểu thức kiểu cho các kiểu dữ liệu sau đây:

a) Một mảng của các con trỏ có kích thước từ 1 đến 100, trỏ đến đối tượng các số thực.

b) Mảng 2 chiều của các số nguyên, hàng có kích thước từ 0 đến 9, cột có chỉ số từ -10 đến 10.

c) Các hàm mà miền định nghĩa là các hàm với các đối số nguyên, trị là con trỏ trỏ đến các số nguyên và miền xác định của nó là các mẫu tin chứa số nguyên và ký tự.

6.2. Giả sử có một khai báo C như sau:

```
typedef   struct  {
                        int a, b ;
                  }  CELL, * PCELL ;
CELL      foo [ 100 ] ;
PCELL     bar (x, y)      int x ;   CELL y  { ..... }
```

Viết các biểu thức kiểu cho các kiểu dữ liệu foo và bar.

6.3. Cho văn phạm sau đây định nghĩa chuỗi của các chuỗi ký tự:

P → D **;** E
D → D **;** D | id **:** T
T → **list of** T | **char** | **integer**
E → **(** L **)** | **literal** | **num** | **id**
L → E **,** L | E

Hãy viết các quy tắc biên dịch để xác định các biểu thức kiểu (E) và list (L).

6.4. Giả sử tên kiểu là link và cell được định nghĩa như ở phần tên cho biểu thức kiểu. Hãy xác định những biểu thức kiểu nào dưới đây là tương đương cấu trúc, những biểu thức kiểu nào tương đương tên.

a) **link**

b) *pointer* (**cell**)

c) *pointer* (**link**)

d) *pointer* (*record* ((info x *integer*) x (next x *pointer* (**cell**)))

6.5. Giả sử rằng kiểu của mỗi định danh là một miền con của số nguyên. Cho biểu thức với các phép toán +, - , * , div và mod như trong Pascal, hãy viết quy tắc kiểm tra kiểu để gán mỗi biểu thức con vào vùng miền con giá trị mã nó sẽ nằm trong đó.