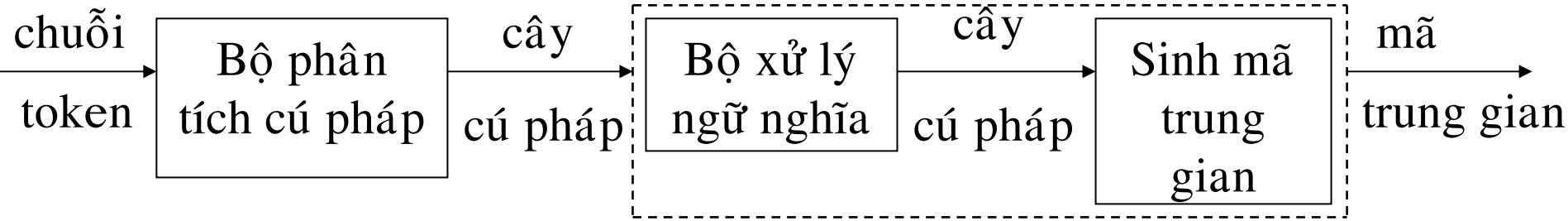
CHƯƠNG 6
XỬ LÝ NGỮ NGHĨA

Xử lý ngữ nghĩa có hai cách: *kiểm tra tĩnh* (static check) và *kiểm tra động* (dynamic check).

Trong chương này chúng ta chỉ bàn đến kiểm tra ngữ nghĩa tĩnh.

Xử lý ngữ nghĩa tĩnh bao gồm:

1. Truyền thuộc tính
2. Kiểm tra kiểu
3. Kiểm tra trình tự điều khiển
4. Kiểm tra tính duy nhất
5. Kiểm tra mối liên hệ của tên
6. Xử lý các phát biểu goto tham khảo trước.



Hình 6.1. Vị trí của bộ xử lý ngữ nghĩa.

6.1. Truyền thuộc tính

1. Mã trung gian

Mã trung gian có nhiều loại: mã cambridge, mã Balan ngược, mã bộ tam (triple code), mã bộ tứ (quadruple code).

Bộ tứ cho biểu thức số học

Dạng tổng quát: <toán tử> (<tác tố 1>, <tác tố 2>, <kết quả>)

Một cách biểu thị biến tạm ở bảng danh biểu:

Tên:rỗng

Loại: 4

Kiểu dữ liệu: tùy theo kiểu của các toán hạng tham gia phép toán.

Địa chỉ : địa chỉ tương đối. Địa chỉ này được gán khi sinh mã.

Một số mã bộ tứ cho các phép toán học

JMP (i, 0, 0)	nhảy đến bộ tứ có chỉ số i
JPG (i, p1, p2)	nhảy đến bộ tứ i nếu toán hạng thứ nhất lớn hơn toán hạng hai
as1 (p1, p2, 0)	gán trị p1 cho p2. p2 là biến đơn
FLT (p1, p2, 0)	Đổi trị của p1 thành số thực, gán sang p2
FIX (p1, p2, 0)	Đổi trị của p1 thành số nguyên, gán sang p2

6.2. Xử lý ngữ nghĩa với phân tích cú pháp từ dưới lên

1. Vấn đề truyền thuộc tính

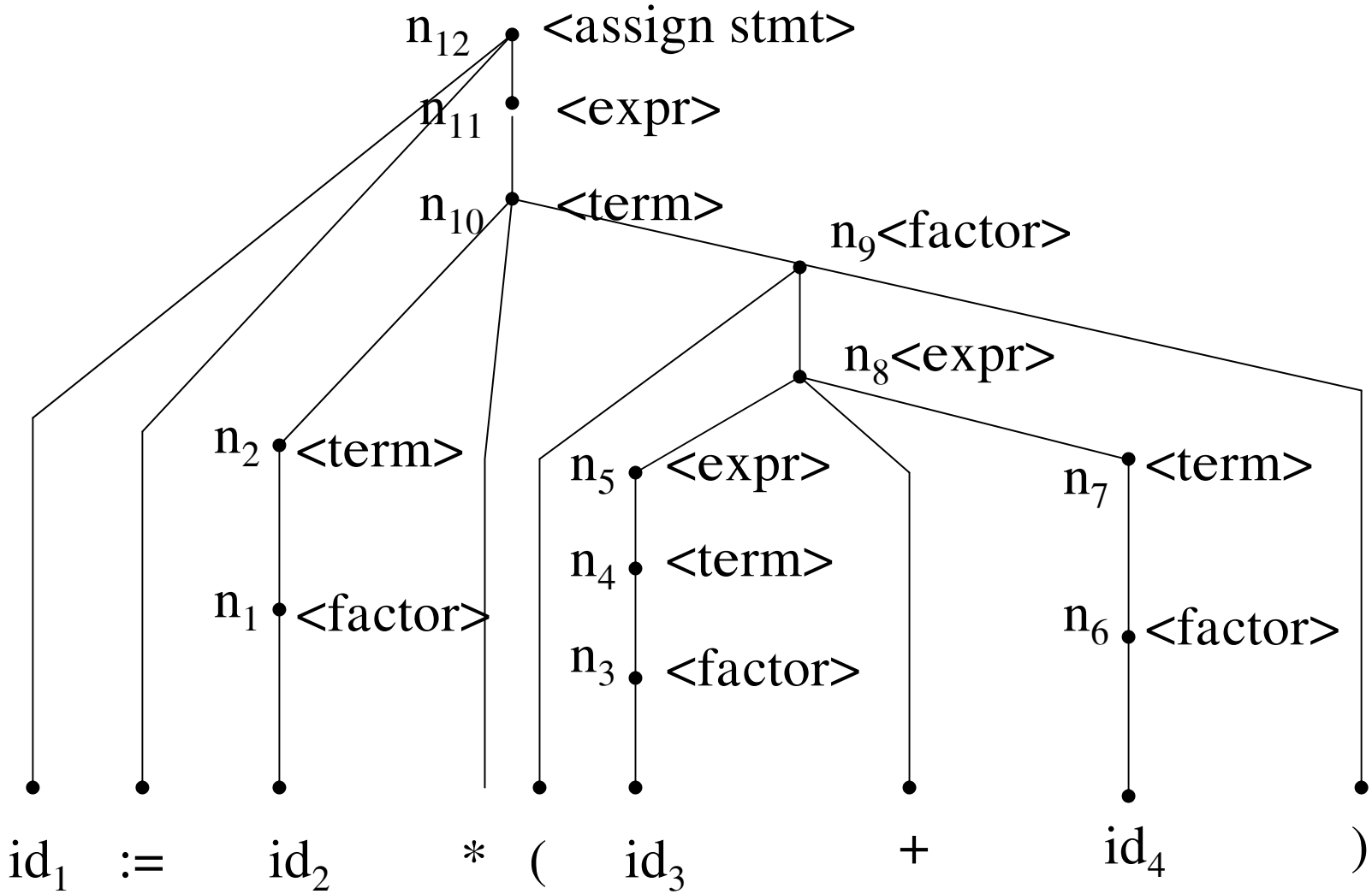
Thí dụ 6.1. Chúng ta có văn phạm G.

<assign stmt> → id := <expr>

<expr> → <expr> + <term> | <term>

<term> → <term> * <factor> | < factor>

< factor > → id | (< expr>)



*Hình 6.2. Cây cú pháp A := X * (R + Q).*

Bảng danh biểu

	Token	Trị từ vựng	Kiểu dữ liệu
1	id	A	thực
2	id	X	thực
3	id	R	thực
4	id	Q	thực

- Truyền thuộc tính
- Sinh ra biến tạm khi thu giảm

2. Phương pháp thực hiện sự truyền thuộc tính

Để thực hiện xử lý ngữ nghĩa trong quá trình phân tích cú pháp, chúng ta sẽ dùng một stack đặc biệt gồm các phần:

A: ký hiệu văn phạm (tượng trưng cho một danh hiệu)

B: có trị 0 hoặc 1 (ký hiệu 1 là biến tạm)

C: con trỏ chỉ đến bảng danh biểu (thực chất là vị trí của danh biểu ở trong bảng danh biểu

Thí dụ 6.2. Cho văn phạm G như ở thí dụ 6.1.

<asign stmt> → id := <expr>
<expr> → <expr> + <term> | <term>
<term> → <expr> * <factor> | <factor>
<factor> → id | (<expr>)

3. Nguyên tắc xử lý ngữ nghĩa

Khi có một chuỗi con $x = x_1x_2...x_n$ sắp được thu giảm về KHKKT U (với luật sinh U → $x_1, x_2 ...x_n$) thì hành vi xử lý ngữ nghĩa tại nút V là hàm của:

1) Luật sinh U → $x_1x_2...x_n$
2) Các xử lý ngữ nghĩa của các nút x_i, tức là các nút con của V có thể là:
    i) Tra cứu bảng danh biểu
    ii) Tạo ra biến tạm
    iii) Sinh mã trung gian
    iv) Truyền thuộc tính từ x về V

6.3. Kiểm tra kiểu dữ liệu

1. Hệ thống kiểu

Định nghĩa biểu thức kiểu

1. Kiểu dữ liệu cơ bản
2. Khi biểu thức kiểu được đặt tên
3. Bộ kiến thiết kiểu bao gồm:
    1) Dãy (array): array (I, T).
    2) Tích số (product): tích số *cartesian* T1 x T2.
    3) Bản ghi (record): kiểu của bản ghi là tích số của biểu thức kiểu các thành phần của nó.

Thí dụ:

```
type row = record
        address : integer;
        lexeme : array [1..15] of char;
        end;
var  table : array [1..10] of row;
```

Kiểu row được biểu diễn bằng biểu thức kiểu:

record ((address x integer) x (lexeme x array (1...15, char))).

4) Con trỏ (pointer): pointer (T).

5) Hàm (function): D → R.

Thí dụ: trong Pascal có khai báo:

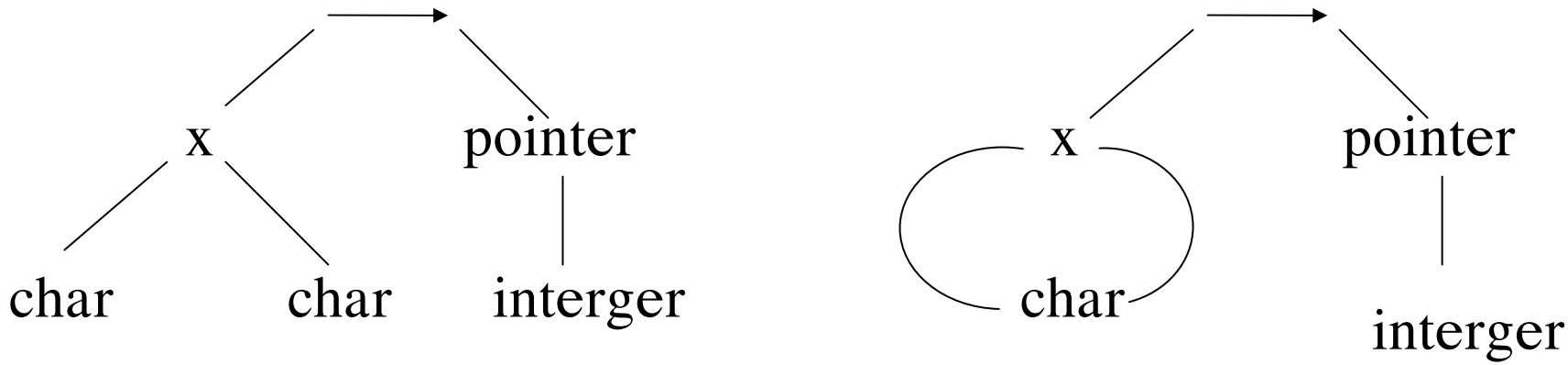
function f (a, b : char) : ↑ integer;

Biểu thức kiểu của f là:

char x char → pointer (integer)

4. Biểu thức kiểu chứa các biến mà trị của chúng là biểu thức kiểu. Để biểu diễn biểu thức kiểu ta dùng đồ thị. (H.6.4) là cây và dag, biểu thị cho biểu thức kiểu char x char → pointer (integer).

Hệ thống kiểu



Hình 6.4. Cây và dag, biểu thị cho biểu thức char x char → point (interger)

Kiểm tra kiểu tĩnh và kiểm tra kiểu động

Phát hiện lỗi

2. Đặc tả bộ kiểm tra kiểu đơn giản

Ngôn ngữ đơn giản

Chúng ta có ngôn ngữ đơn giản được sinh ra từ văn phạm G

$P \rightarrow D ; E$

$D \rightarrow D ; D \mid id : T$

$T \rightarrow char \mid integer \mid array\ [num]\ of\ T \mid \uparrow T$

$E \rightarrow literal \mid num \mid id \mid E\ mod\ E \mid [E] \mid E \uparrow$

Mô phỏng 6.1. *Sơ đồ biên dịch dùng để lưu giữ kiểu của các danh biểu*

$P \rightarrow D ; E$

$D \rightarrow D ; D$

$D \rightarrow id : T$ {addtype (id. entry, T. type)}

$T \rightarrow char$ { T. type := char}

$T \rightarrow integer$ {T. type := integer}

$T \rightarrow \uparrow T1$ {T. type := pointer (T1. type)}

$T \rightarrow array\ [num]\ of\ T1$ {T. type = array (num. val, T1 . type)}

Kiểm tra cho biểu thức

1. Kiểu token là literal và num thí có kiểu là char và integer.

E → literal {E. type := char}

E → num {E. type := integer}

2. E → id {E. type := lookup (id. Entry)}

3. E → E1 mod E2 {E. type := **if** (E1 . type = integer) and
(E2. type := integer) **then** integer
else type - error}

4. E → E1 {E2} {E. type := if (E1 . type = integer) and
(E1. type E2 = array (s, t)) **then** t **else** type – error}

5. E → E1 ↑ {E. type := if E1. type = pointer (t) **then** t
else type – error}

Kiểm tra kiểu dữ liệu cho các phát biểu

Một chương trình bao gồm các khai báo, sau đó là các phát biểu, điều này được biểu thị bằng luật sinh P → D; S.

Mô phỏng 6.2. *Sơ đồ biên dịch cho kiểu dữ liệu của các phát biểu*

$P \rightarrow D; S$

$S \rightarrow$ id := E {S.type := **if** id. type = E. type **then** void
else type - error}

$S \rightarrow$ if E then S1 {S.type := **if** E. type = boolean **then**
S1. type **else** type - error}

$S \rightarrow$ while E do S1 {S.type := **if** E. type = boolean **then**
S1. type **else** type - error}

$S \rightarrow S1; S2$ {S.type := **if** (S1. type = void) and
(S2. Type = void) **then** void
else type - error}

Kiểm tra kiểu của hàm

E → E (E)

Để diễn tả kiểu cho biểu thức kiểu ta dùng ký hiệu T và thêm luật sinh

T → T1' → ' T2 {T. type := T1. type → T2. type}

Quy tắc kiểm tra kiểu của hàm là

E → E1 (E2) {E. type := **if** (E2. type = s) and
(E1. type = s → t) **then** t **else**
{type - error}

T1 x T2 x .. x Tn

Thí dụ: root → (real → real) x real → real

Chúng ta sẽ hiểu là có khai báo:

function root (**function**f (real) : real; x : real) : real

3. Sự tương đương của biểu thức kiểu

Sự tương đương cấu trúc của biểu thức kiểu

Giải thuật kiểm tra tương đương cấu trúc của các biểu thức kiểu

Mô phỏng 6.3. *Kiểm tra tương đương cấu trúc của hai biểu thức kiểu s và t.*

```
function sequiv (s, t): boolean;
    begin
        if s và t cùng một kiểu cơ bản then true
            else if s = array (s1, s2) and t = array (t1, t2) then
                return sequiv (s1, s2) and sequiv (s2, t2)
            else if  s = s1 x s2 and t = t1 x t2 then
                return sequiv (s1, t1) and sequiv (s2, t2)
            else if  s = pointer (s1) and t = pointer (t1) then
                return sequiv (s1, t1)
            else if  s = s1→ s2 and t = t1→ t2  then
                return sequiv (s1, t1) and sequiv (s2, t2)
               else return false;
    end;
```

Thí dụ 6.3. Chúng ta sẽ giới thiệu cách mã hoá các biểu thức kiểu của trình biên dịch C do D.M. Ritchie viết.

Mô phỏng 6.4. *Các thí dụ về biểu thức kiểu.*

char

freturns (char)

pointer (freturns (char))

array (pointer (freturns (char))

Bộ kiến thức kiểu	Mã hóa
pointer	01
array	10
freturns	11

Các kiểu cơ bản của ngôn ngữ C được John (1979) mã hóa bằng 4 bit

Kiểu cơ bản	Mã hóa
boolean	0000
char	0001
integer	0010
real	0011

Mô phỏng 6.5. *Mã hóa biểu thức kiểu ở mô phỏng 6.4.*

Biểu thức kiểu	Mã hóa
char	000000 0001
freturns (char)	000011 0001
pointer (freturns (char))	0111 0001
array (pointer (freturns (char))	100111 0001

- Tên cho biểu thức kiểu

Sau đây là một đoạn khai báo kiểu trong Pascal:

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

Thí dụ 6.4.

Biến	Biểu thức kiểu
next	link
last	link
p	pointer (cell)
q	pointer (cell)
r	pointer (cell)

Hình 6.11. Biến và các biểu thức kiểu tương ứng

6.4. Chuyển đổi kiểu

Thí dụ ký hiệu hậu tố của biểu thức a + i sau khi thực hiện hành vi chuyển đổi kiểu:

a i **intereal real** +

- Áp đặt toán tử (Coercion)

Thí dụ 6.5.

Mô phỏng 6.6. *Quy tắc kiểm tra kiểu cho việc áp đặt toán tử để đổi trị toán hạng từ số nguyên sang số thực.*

Luật sinh	Luật ngữ nghĩa
E → num	E. type := integer
E → num. num	E. type := real
E → id	E. type := lookup (id. entry)
E → E1 op E2	E. type := **if** (E1. type = integer) and (E2. type = integer) **then** integer **else if** (E1. type = integer) and (E2. type = real) **then** real **else if** (E1. type = real) and (E2. type = integer) **then** real **else if** (E1. type = real) and (E2. type = real) **then** real **else** type - error

Lưu ý:

for I := 1 to N do x [i] := 1 (1)

for I := 1 to N do x [i] := 1.0 (2)

6.5. Xử lý ngữ nghĩa cho phát biểu goto tham khảo trước

Thí dụ 6.6. Giả sử chúng ta có đoạn chương trình

```
...
goto L          (10)        JMP (0,0,0)
...
goto L          (50)        JMP (10,0,0)
...
goto L          (90)        JMP (50,0,0)
...
L: x := x + 1   (120)
```

Bảng 6.2. *Bảng lưu giữ tên phát biểu và chỉ số đầu danh sách liên kết*

Tên p/b	Địa chỉ	Định nghĩa
L	90	0

Bảng 6.3. *Điền chỉ số của tên L vào các lệnh nhảy*

Tên p/b	Địa chỉ	Định nghĩa
L	120	1

```
(10)    JMP    (120,0,0)
(50)    JMP    (120,0,0)
(90)    JMP    (120,0,0)
```