

CHƯƠNG TRÌNH DỊCH

Chương 5. Biên dịch dựa trên cú pháp

TS. Phạm Văn Cảnh
Khoa Công nghệ thông tin

Email: canh.phamvan@phenikaa-uni.edu.vn

-
- 1. Định nghĩa dịch trực tiếp cú pháp**
 - 2. Xây dựng cây cú pháp**
 - 3. Đánh giá từ dưới lên**
 - 4. Định nghĩa L_thuộc tính**
 - 5. Dịch từ trên xuống**

1. Định nghĩa trực tiếp cú pháp

- Trong một định nghĩa trực tiếp cú pháp, mỗi luật sinh $A \rightarrow \alpha$ kết hợp một tập luật ngữ nghĩa có dạng **$b := f(c_1, c_2, \dots, c_k)$** trong đó f là một hàm và
 - b là một **thuộc tính tổng hợp** của A và c_1, c_2, \dots, c_k là các thuộc tính của các ký hiệu văn phạm của luật sinh. Hoặc
 - b là một **thuộc tính kế thừa** của một trong các ký hiệu văn phạm trong vế phải của luật sinh và c_1, c_2, \dots, c_k là các thuộc tính của các ký hiệu văn phạm của luật sinh.
- Ta nói b phụ thuộc c_1, c_2, \dots, c_k .

1. Định nghĩa trực tiếp cú pháp

□ Thuộc tính tổng hợp:

- Là thuộc tính mà giá trị của nó tại mỗi nút trên cây phân tích cú pháp được tính từ giá trị thuộc tính tại các nút con của nó.
- Định nghĩa trực tiếp cú pháp chỉ sử dụng các thuộc tính tổng hợp gọi là định nghĩa **S _ thuộc tính**.
- Cây phân tích cú pháp của định nghĩa **S _ thuộc tính** có thể được chú thích từ dưới lên trên.

1. Định nghĩa trực tiếp cú pháp

- Ví dụ: Định nghĩa trực tiếp cú pháp cho một máy tính tay đơn giản.

Luật sinh	Luật ngữ nghĩa
$L \rightarrow \mathbf{En}$	$print(E.val)$
$E \rightarrow E_1 + T$	$E.val := E_1.val + T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 * F$	$T.val := T_1.val * F.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow (E)$	$F.val := E.val$
$F \rightarrow \mathbf{digit}$	$F.val := digit.lexval$

1. Định nghĩa trực tiếp cú pháp

□ Thuộc tính kế thừa:

- Là một thuộc tính mà giá trị của nó được xác định từ giá trị các thuộc tính của các nút cha hoặc anh em của nó.
- Nói chung ta có thể viết một định nghĩa trực tiếp cú pháp thành một định nghĩa $S_{\text{thuộc tính}}$.
- Trong một số trường hợp, việc sử dụng thuộc tính kế thừa lại thuận tiện vì tính tự nhiên của nó.

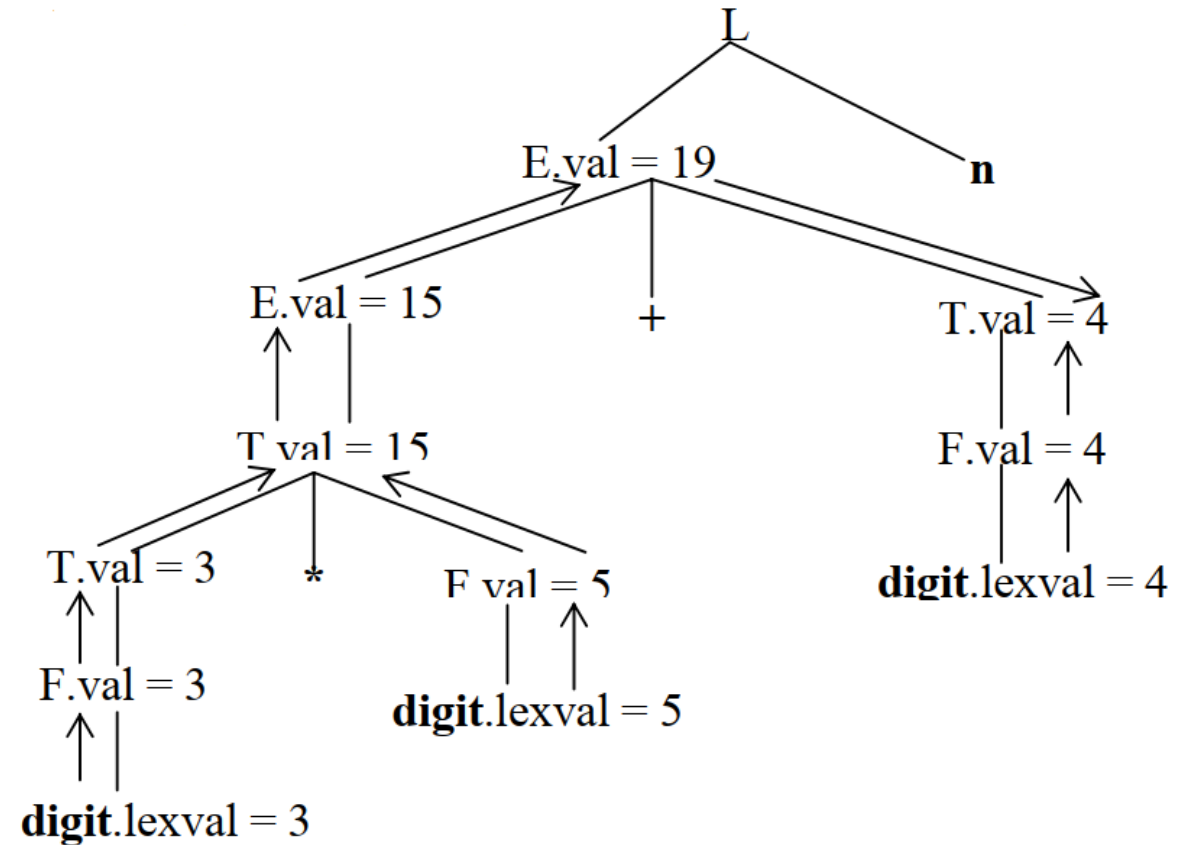
1. Định nghĩa trực tiếp cú pháp

□ Ví dụ: Định nghĩa trực tiếp cú pháp với luật kế thừa L.in

Luật sinh	Luật ngữ nghĩa
$D \rightarrow TL$	$L.in := T.type$
$T \rightarrow \text{int}$	$T.type := integer$
$T \rightarrow \text{real}$	$T.type := real$
$L \rightarrow L_1, \text{id}$	$L_1.in := L.in; addtype(id.entry, L.in)$
$L \rightarrow \text{id}$	$addtype(id.entry, L.in)$

1. Định nghĩa trực tiếp cú pháp

□ Ví dụ: Cây phân tích cú pháp cho biểu thức $3 * 5 + 4n$



1. Định nghĩa trực tiếp cú pháp

□ Đồ thị phụ thuộc:

- Đồ thị phụ thuộc là một đồ thị có hướng mô tả sự phụ thuộc giữa các thuộc tính tại mỗi nút của cây phân tích cú pháp.
- Cho một cây phân tích cú pháp thì đồ thị phụ thuộc tương ứng được xây dựng theo giải thuật sau:

- FOR mỗi một nút n trong **cây phân tích cú pháp** DO
 - FOR với mỗi một thuộc tính a của ký hiệu văn phạm tại n DO
 - Xây dựng một nút trong đồ thị phụ thuộc cho a
 - FOR với mỗi một nút n trên cây phân tích cú pháp DO
 - FOR với mỗi một luật ngữ nghĩa dạng $\mathbf{b} = \mathbf{f}(\mathbf{c1}, \mathbf{c2}, \dots, \mathbf{ck})$ kết hợp với luật sinh được dùng tại nút n DO
 - FOR $i:=1$ TO k DO
 - Xây dựng một cạnh từ nút cho c_i đến nút cho b

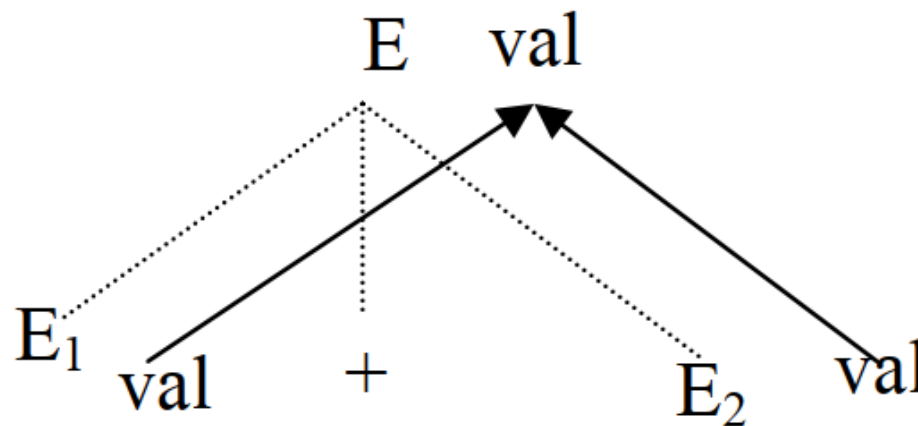
1. Định nghĩa trực tiếp cú pháp

□ Ví dụ: với định nghĩa S_thuộc tính:

$$E \rightarrow E_1 + E_2$$

$$E.val := E_1.val + E_2.val$$

Ta có đồ thị phụ thuộc:

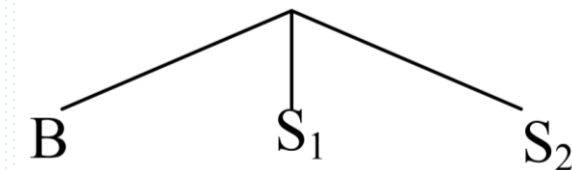


2. Xây dựng cây cú pháp

- ❑ Cây cú pháp (syntax - tree) là dạng rút gọn của cây phân tích cú pháp dùng để biểu diễn cấu trúc ngôn ngữ.
- ❑ Trong cây cú pháp các toán tử và từ khóa không phải là nút lá mà là các nút trong.
- ❑ Một kiểu rút gọn khác của cây cú pháp là chuỗi các luật sinh đơn được rút gọn lại.

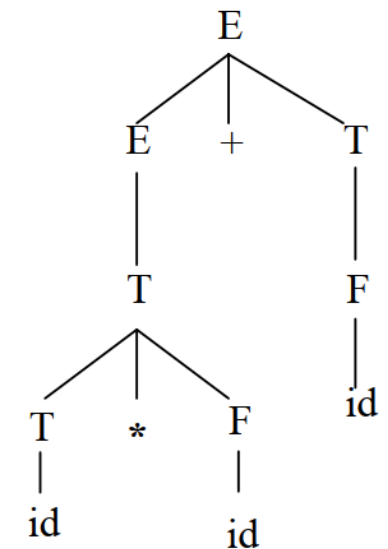
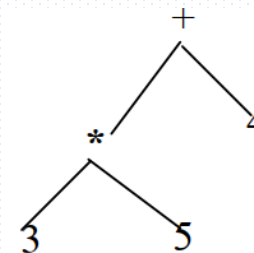
- ❑ Ví dụ với luật sinh S (if B then S1 else S2 được biểu diễn bởi cây cú pháp:

if - then - else



- ❑ Cây cú pháp rút gọn:

được rút gọn từ



2. Xây dựng cây cú pháp

- ❑ Xây dựng cây cú pháp cho biểu thức:
 - Tương tự như việc dịch một biểu thức thành dạng hậu tố.
 - Xây dựng cây con cho biểu thức con bằng cách tạo ra một nút cho toán hạng và toán tử.
 - Con của nút toán tử là gốc của cây con biểu diễn cho biểu thức con toán hạng của toán tử đó.
 - Mỗi một nút có thể cài đặt bằng một mẫu tin có nhiều trường.
 - Trong nút toán tử, có một trường chỉ toán tử như là nhãn của nút, các trường còn lại chứa con trỏ, trỏ tới các nút toán hạng.

2. Xây dựng cây cú pháp

- ❑ Để xây dựng cây cú pháp cho biểu thức chúng ta sử dụng các hàm sau đây:
 - 1) **mknode(op, left, right)**: Tạo một nút toán tử có nhãn là op và hai trường chứa con trỏ, trỏ tới con trái left và con phải right.
 - 2) **mkleaf(id, entry)**: Tạo một nút lá với nhãn là id và một trường chứa con trỏ entry, trỏ tới ô trong bảng ký hiệu.
 - 3) **mkleaf(num, val)**: Tạo một nút lá với nhãn là num và trường val, giá trị của số.

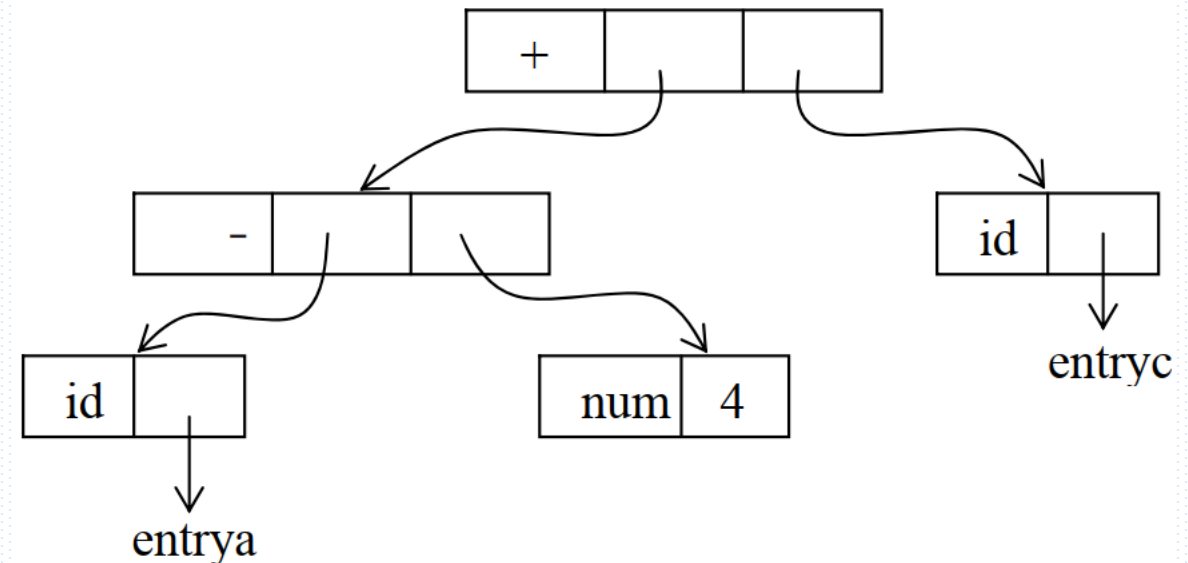
2. Xây dựng cây cú pháp

❑ Ví dụ: Để xây dựng cây cú pháp cho biểu thức: **a - 4 + c** ta dùng một dãy các lời gọi các hàm nói trên.

- (1): $p1 := \text{mkleaf}(\text{id}, \text{entrya})$
- (2): $p2 := \text{mkleaf}(\text{num}, 4)$
- (3): $p3 := \text{mknode}('-', p1, p2)$
- (4): $p4 := \text{mkleaf}(\text{id}, \text{entryc})$
- (5): $p5 := \text{mknode}('+', p3, p4)$

❑ Cây được xây dựng từ dưới lên:

- **entrya** là con trỏ, trỏ tới ô của a trong bảng ký hiệu
- **entryc** là con trỏ, trỏ tới ô của c trong bảng ký hiệu



2. Xây dựng cây cú pháp

- ❑ Xây dựng cây cú pháp từ định nghĩa trực tiếp cú pháp: Căn cứ vào các luật sinh văn phạm và luật ngữ nghĩa kết hợp mà ta phân bổ việc.
- ❑ **Ví dụ:** Định nghĩa trực tiếp cú pháp giúp việc xây dựng cây cú pháp cho biểu thức là:

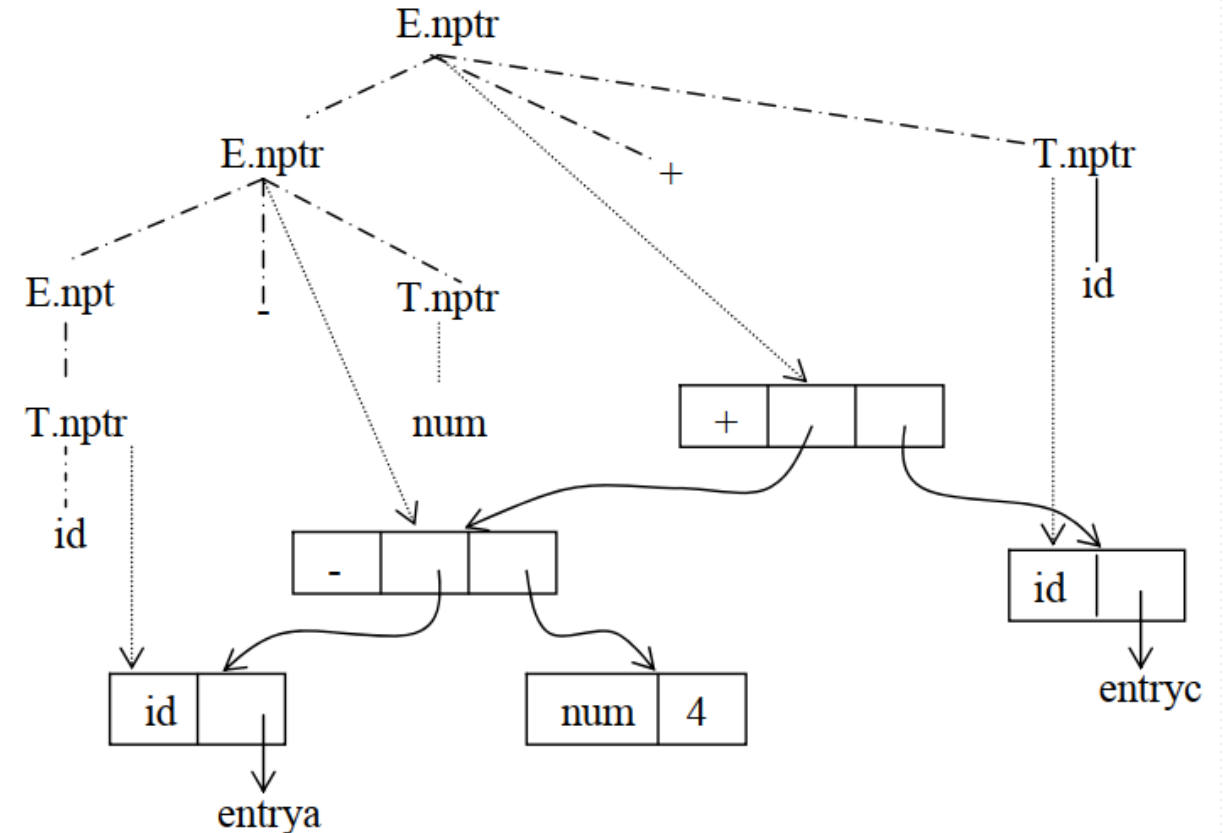
Luật sinh	Luật ngữ nghĩa
$E \rightarrow E_1 + T$	$E.nptr := mknnode('+', E_1.nptr, T.nptr)$
$E \rightarrow E_1 - T$	$E.nptr := mknnode('-', E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr := T.nptr$
$T \rightarrow (E)$	$T.nptr := E.nptr$
$T \rightarrow id$	$T.nptr := mkleaf(id, id.entry)$
$T \rightarrow num$	$T.nptr := mkleaf(num, num.val)$

2. Xây dựng cây cú pháp

❑ **Ví dụ:** Định nghĩa trực tiếp cú pháp giúp việc xây dựng cây cú pháp cho biểu thức là:

Xây dựng cây cú pháp cho $a - 4 + c$

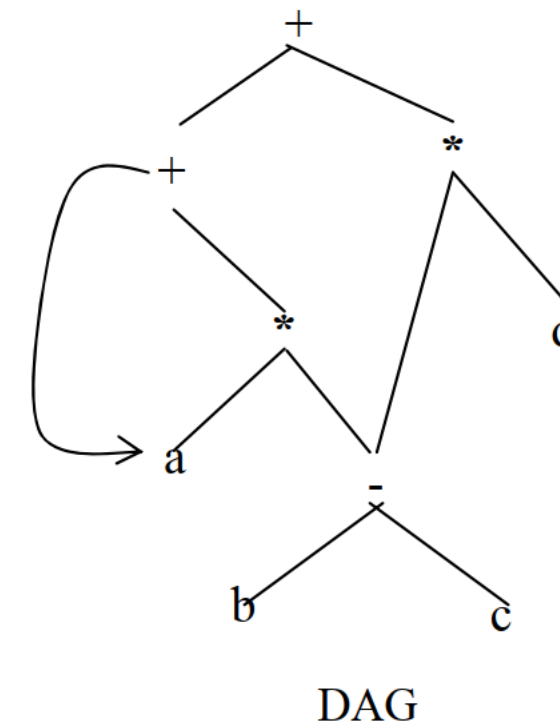
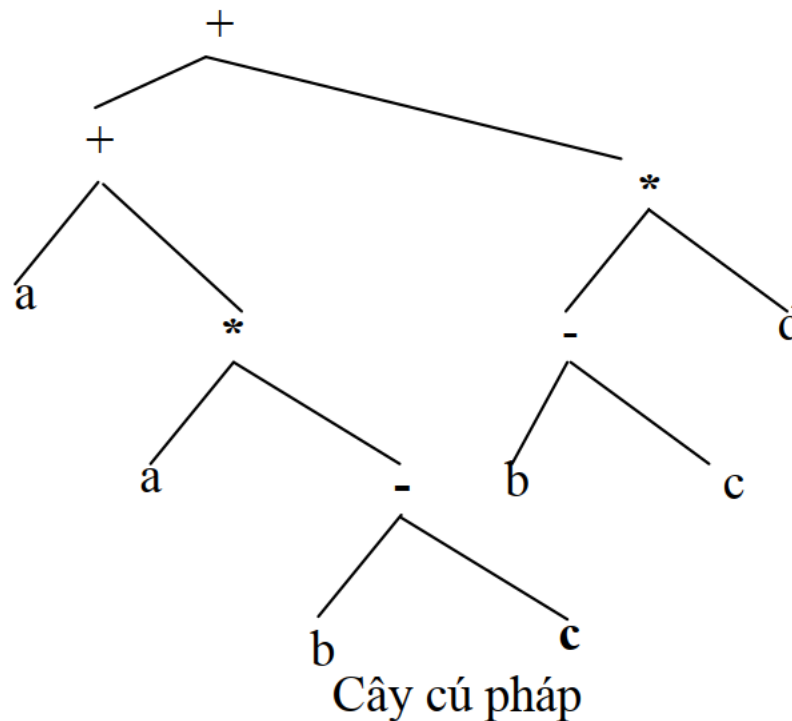
Luật sinh	Luật ngữ nghĩa
$E \rightarrow E_1 + T$	$E.nptr := mknnode('+', E_1.nptr, T.nptr)$
$E \rightarrow E_1 - T$	$E.nptr := mknnode('-', E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr := T.nptr$
$T \rightarrow (E)$	$T.nptr := E.nptr$
$T \rightarrow id$	$T.nptr := mkleaf(id, id.entry)$
$T \rightarrow num$	$T.nptr := mkleaf(num, num.val)$



2. Xây dựng cây cú pháp

Đồ thị có hướng không tuần hoàn cho biểu thức (Directed Acyclic Graph - DAG)

- ❑ DAG cũng giống như cây cú pháp, tuy nhiên trong cây cú pháp các biểu thức con giống nhau được biểu diễn lặp lại còn trong DAG thì không. Trong DAG, một nút con có thể có nhiều "cha"
- ❑ Ví dụ: Cho biểu thức $a + a * (b - c) + (b - c) * d$. Ta có cây cú pháp và DAG:



2. Xây dựng cây cú pháp

Đồ thị có hướng không tuần hoàn cho biểu thức (Directed Acyclic Graph - DAG)

- ❑ Để xây dựng một DAG, trước khi tạo một nút phải kiểm tra xem nút đó đã tồn tại chưa:
 - nếu đã tồn tại thì hàm tạo nút (mknode, mkleaf) trả về con trỏ của nút đã tồn tại,
 - nếu chưa thì tạo nút mới.
 - **Cài đặt DAG:** Người ta thường sử dụng một mảng mẫu **tin** , mỗi mẫu tin là một nút. Ta có thể tham khảo tới nút bằng chỉ số của mảng.

2. Xây dựng cây cú pháp

Đồ thị có hướng không tuần hoàn cho biểu thức (Directed Acyclic Graph - DAG)

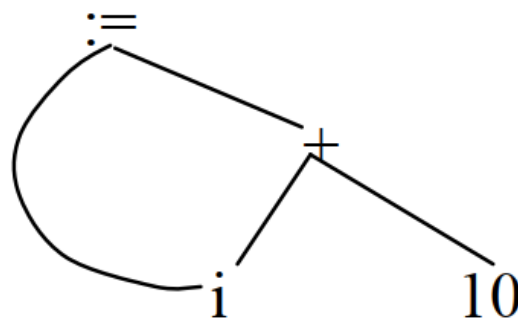
❑ Ví dụ: minh họa DAG

- Nút 1: có nhãn là id, con trỏ trỏ tới entry i.
- Nút 2: có nhãn là num, giá trị là 10.
- Nút 3: có nhãn là +, con trái là nút 1, con phải là nút 2.
- Nút 4: có nhãn là :=, con trái là nút 1, con phải là nút 3.

Lệnh gán

$i := i + 10$

DAG



Biểu diễn

1	id	entry i	
2	num	10	
3	+	1	2
4	:=	1	3

2. Xây dựng cây cú pháp

Đồ thị có hướng không tuần hoàn cho biểu thức (Directed Acyclic Graph - DAG)

- ❑ Giải thuật: Phương pháp số giá trị (value – number) để xây dựng một nút trong DAG.
 - Giả sử rằng các nút được lưu trong một mảng và mỗi nút được tham khảo bởi số giá trị của nó. Mỗi một nút toán tử là một bộ ba $\langle \text{op}, l, r \rangle$
- ❑ **Input:** Nhận op, nút l và nút r.
- ❑ **Output:** Một nút với $\langle \text{op}, l, r \rangle$
- ❑ Phương pháp: Tìm trong mảng một nút m có nhãn là op con trái là l, con phải là r.
 - Nếu tìm thấy thì trả về m, ngược lại tạo ra một nút mới n, có nhãn là op, con trái là l, con phải là r và trả về n.

3. Đánh giá dưới lên đối với các định nghĩa S_thuộc tính



□ Sử dụng Stack

- Như đã biết, định nghĩa S_ thuộc tính chỉ chứa các thuộc tính tổng hợp do đó phương pháp phân tích dưới lên là phù hợp với định nghĩa trực tiếp cú pháp này.
- Phương pháp phân tích dưới lên sử dụng một STACK để lưu trữ thông tin về cây con đã được phân tích.
- Chúng ta có thể mở rộng STACK này để lưu trữ giá trị thuộc tính tổng hợp.
- STACK được cài đặt bởi một cặp mảng state và val.

3. Đánh giá dưới lên đối với các định nghĩa

S_thuộc tính

□ Sử dụng Stack:

- Giả sử luật ngữ nghĩa $A.a := f (X.x, Y.y, Z.z)$ kết hợp với luật sinh $A \rightarrow XYZ$.
- Trước khi XYZ được rút gọn thành A thì $val[top] = Z.z$, $val[top - 1] = Y.y$, $val[top - 2] = X.x$. Sau khi rút gọn, top bị giảm 2 đơn vị, A nằm trong $state[top]$ và thuộc tính tổng hợp nằm trong $val[top]$.

3. Đánh giá dưới lên đối với các định nghĩa S_thuộc tính



□ Ví dụ: Stack phân tích cú pháp vào một trường lưu giữ thuộc tính tổng hợp

- Mỗi ô trong stack là một con trỏ trỏ tới bảng phân tích LR(1).
- Nếu phần tử thứ i của stack là ký hiệu A thì $val[i]$ là giá trị thuộc tính kết hợp với A .

State	val
...	...
X	X.x
Y	Y.y
Z	Z.z

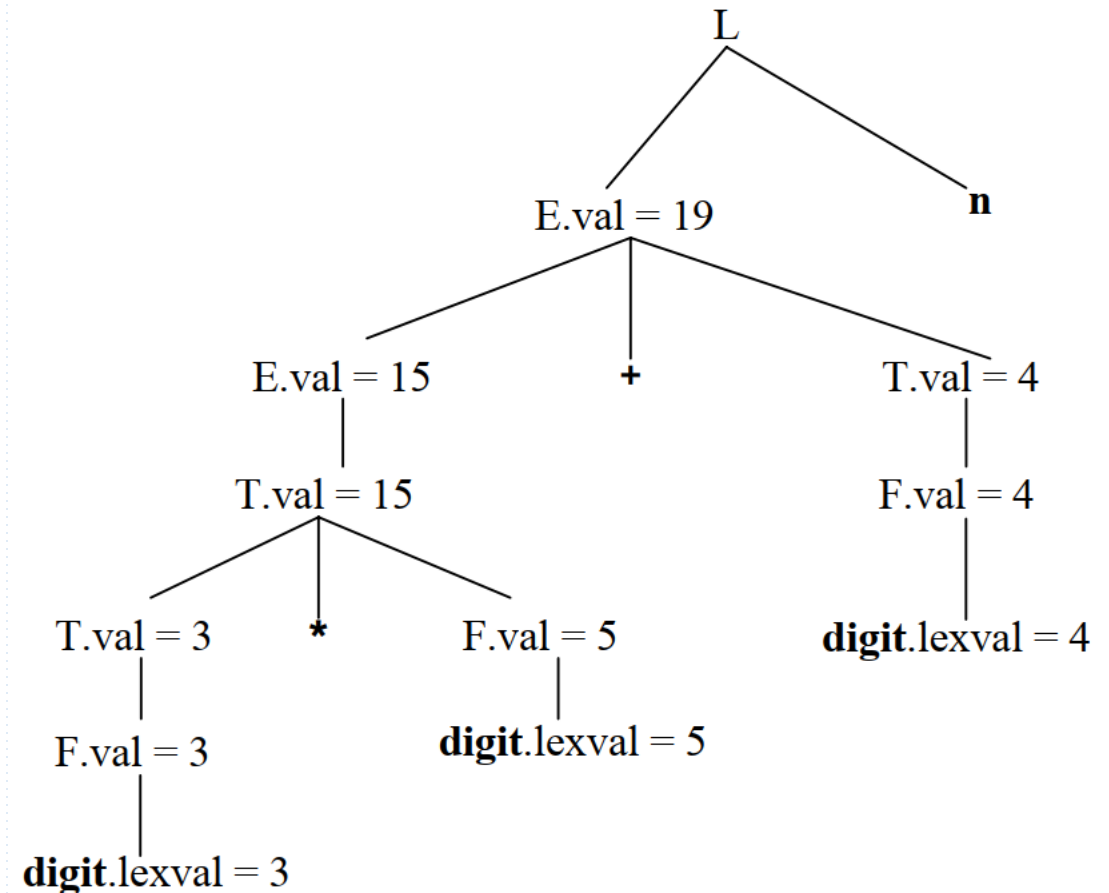
← top

3. Đánh giá dưới lên đối với các định nghĩa S_thuộc tính

□ Ví dụ: Xét định nghĩa trực tiếp cú pháp:

- Với biểu thức $3 * 5 + 4$ n ta có cây chú thích:

Luật sinh	Luật ngữ nghĩa
$L \rightarrow En$	$print(E.val)$
$E \rightarrow E_1 + T$	$E.val := E_1.val + T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 * F$	$T.val := T_1.val * F.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow (E)$	$F.val := E.val$
$F \rightarrow \text{digit}$	$F.val := \text{digit.lexval}$



3. Đánh giá dưới lên đối với các định nghĩa



□ Ví dụ: Xét định nghĩa trực tiếp cú pháp:

- Cây chú thích này có thể được đánh giá bằng một bộ phân tích cú pháp LR từ dưới lên trên.
- Chú ý rằng bộ phân tích đã nhận biết giá trị thuộc tính `digit.lexval`. Khi `digit` được đưa vào `stack` thì token `digit` được đưa vào `state[top]` và giá trị thuộc tính của nó được đưa vào `val[top]`.
- Để đánh giá các thuộc tính chúng ta thay đổi bộ phân tích cú pháp để thực hiện đoạn mã sau:

Luật sinh	Luật ngữ nghĩa
$L \rightarrow E_n$	<code>print(val[top])</code>
$E \rightarrow E_1 + T$	<code>val[ntop] := val[top - 2] + val[top]</code>
$E \rightarrow T$	
$T \rightarrow T_1 * F$	<code>val[ntop] := val[top - 2] * val[top]</code>
$T \rightarrow F$	
$F \rightarrow (E)$	<code>val[ntop] := val[top - 1]</code>
$F \rightarrow \text{digit}$	

3. Đánh giá dưới lên đối với các định nghĩa

S_thuộc tính

Bảng sau trình bày quá trình thực hiện của bộ phân tích cú pháp

□ Ví dụ: Xét định nghĩa trực tiếp cú pháp:

- Khi một luật sinh với r ký hiệu bên vế phải được rút gọn thì $ntop = top - r + 1$.
- Sau khi một đoạn mã thực hiện thì đặt $top = ntop$
- Bảng sau trình bày quá trình thực hiện của bộ phân tích cú pháp

Input	State	Val	Luật sinh được dùng
$3 * 5 + 4 n$	—	—	
$* 5 + 4 n$	3	3	
$* 5 + 4 n$	F	3	$F \rightarrow digit$
$* 5 + 4 n$	T	3	$T \rightarrow F$
$5 + 4 n$	T^*	3 -	
$+ 4 n$	$T * 5$	3 - 5	
$+ 4 n$	$T * F$	3 - 5	$F \rightarrow digit$
$+ 4 n$	T	15	$T \rightarrow T * F$
$+ 4 n$	E	15	$E \rightarrow T$
$4 n$	$E +$	15 -	
n	$E + 4$	15 - 4	
n	$E + F$	15 - 4	$F \rightarrow digit$
n	$E + T$	15 - 4	$T \rightarrow F$
n	E	19	$E \rightarrow E + T$
	$E n$	19 -	
	L	19	$L \rightarrow E n$

4. Định nghĩa L_thuộc tính

- Mỗi định nghĩa trực tiếp cú pháp là một định nghĩa L thuộc tính nếu mỗi một thuộc tính kế thừa của X_i ($1 \leq i \leq n$) trong vế phải của luật sinh $A \rightarrow X_1X_2...X_n$ chỉ phụ thuộc vào:
- Các thuộc tính của $X_1, X_2, ..., X_{i-1}$
 - Các thuộc tính kế thừa của A
 - Ví dụ: Định nghĩa trực tiếp cú pháp không phải là định nghĩa L_thuộc tính.
 - Lý do: Thuộc tính kế thừa Q_i phụ thuộc vào thuộc tính $R.s$ của ký hiệu bên phải nó trong luật sinh

Luật sinh	Luật ngữ nghĩa
$A \rightarrow L M$	$L.i := l(A.i)$ $M.i := m(L.s) \quad A.s := f(M.s)$ $R.i := r(A.i)$
$A \rightarrow Q R$	$Q.i := q(R.s) \quad A.s := f(Q.r)$

4. Định nghĩa L_thuộc tính

□ Lược đồ dịch: Lược đồ dịch là một văn phạm phi ngữ cảnh trong đó các thuộc tính được kết hợp với các ký hiệu văn phạm và các hành vi ngữ nghĩa nằm trong cặp dấu { } được xen vào bên phải của luật sinh.

- Các thuộc tính của X_1, X_2, \dots, X_{i-1}
- Các thuộc tính kế thừa của A
- Ví dụ: Lược đồ dịch một biểu thức trung tố với phép cộng và trừ thành dạng hậu tố:

$$E \rightarrow T R$$

$$R \rightarrow \mathbf{addop} \ T \ \{ \textit{print} \ (\textit{addop.lexeme}) \} \ R_1 \mid \varepsilon$$

$$T \rightarrow \mathbf{num} \quad \{ \textit{print} \ (\textit{num.val}) \}$$

4. Định nghĩa L_thuộc tính

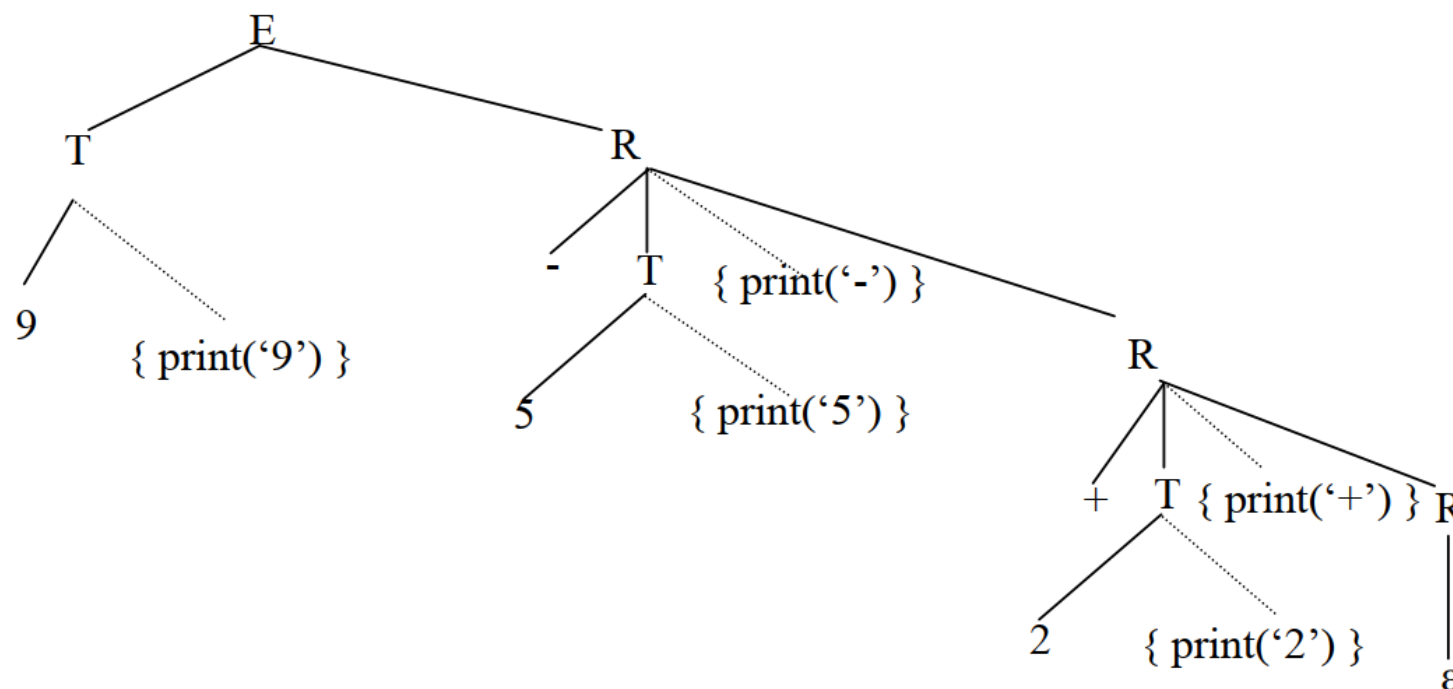
- Ví dụ: Lược đồ dịch một biểu thức trung tố với phép cộng và trừ thành dạng hậu tố:

$$E \rightarrow T R$$

$$R \rightarrow \text{addop } T \{ \text{print}(\text{addop.lexeme}) \} R_1 \mid \varepsilon$$

$$T \rightarrow \text{num} \quad \{ \text{print}(\text{num.val}) \}$$

- Với biểu thức $9 - 5 + 2$ ta có cây phân tích cú pháp.



4. Định nghĩa L_thuộc tính

□ Để xây dựng một lược đồ dịch, chúng ta xét hai trường hợp sau đây:

- Trường hợp 1: Chỉ chứa thuộc tính tổng hợp
 - Với mỗi một luật ngữ nghĩa, ta tạo một hành vi ngữ nghĩa và đặt hành vi này vào cuối vế phải luật sinh.

○ Ví dụ:

Luật sinh

$T \rightarrow T_1 * F$

Luật ngữ nghĩa

$T.val := T_1.val * F.val$

Ta có lược đồ dịch:

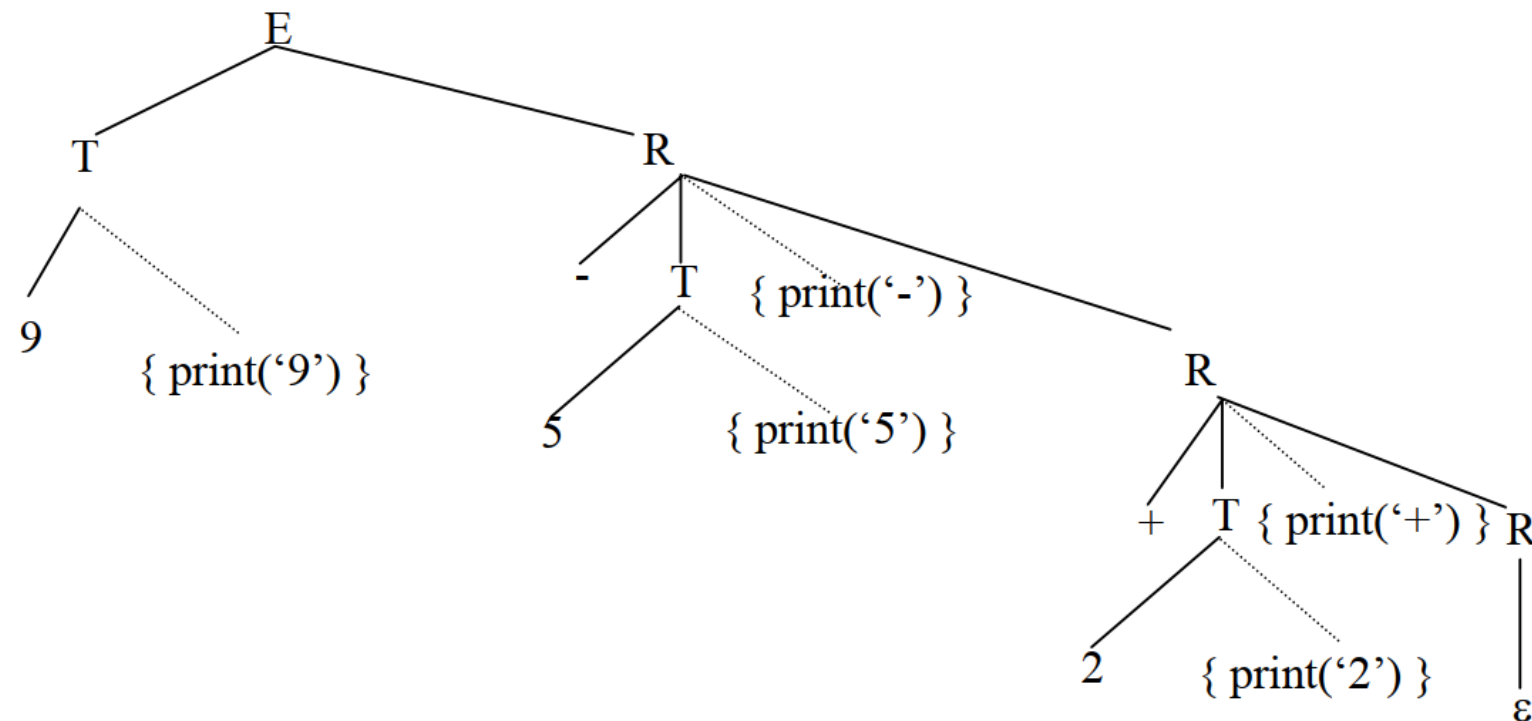
$T \rightarrow T_1 * F \{ T.val := T_1.val * F.val \}$

4. Định nghĩa L_thuộc tính

- Để xây dựng một lược đồ dịch, chúng ta xét hai trường hợp sau đây:
- Trường hợp 2: Có cả thuộc tính tổng hợp và kế thừa phải thỏa mãn 3 yêu cầu sau đây:
 - Thuộc tính kế thừa của một ký hiệu trong vế phải của luật sinh phải được xác định trong hành vi nằm trước ký hiệu đó;
 - Một hành vi không được tham khảo tới thuộc tính tổng hợp của một ký hiệu nằm bên phải hành vi đó;
 - Thuộc tính tổng hợp của ký hiệu chưa kết thúc trong vế trái chỉ có thể được xác định sau khi tất cả các thuộc tính mà nó tham khảo đã được xác định. Hành vi xác định các thuộc tính này luôn đặt cuối vế phải của luật sinh.

4. Định nghĩa L_thuộc tính

- Để xây dựng một lược đồ dịch, chúng ta xét hai trường hợp sau đây:
- Với một định nghĩa trực tiếp cú pháp L_thuộc tính ta có thể xây dựng lược đồ dịch thỏa mãn 3 yêu cầu nói trên.



4. Định nghĩa L_thuộc tính

- Ví dụ: Bộ xử lý các công thức toán học – EQN - có thể xây dựng các biểu thức toán học từ các toán tử sub (subscripts) và sup (superscripts). Chẳng hạn:
- Để xác định chiều rộng và chiều cao của các hộp ta có định nghĩa L_thuộc tính như sau:

input	output
BOX sub box	BOX _{box}
a sub {i sup 2 }	a_{i^2}

Luật sinh	Luật ngữ nghĩa
$S \rightarrow B$	$B.ps := 10$ $S.ht := B.ht$
$B \rightarrow B_1 B_2$	$B_1.ps := B.ps$ $B_2.ps := B.ps$ $B.ht := \max(B_1.ht, B_2.ht)$
$B \rightarrow B_1 \text{ sub } B_2$	$B_1.ps := B.ps$ $B_2.ps := \text{shrink}(B.ps)$ $B.ht := \text{disp}(B_1.ht, B_2.ht)$
$B \rightarrow \text{text}$	$B.ht := \text{text.h} * B.ps$

4. Định nghĩa L_thuộc tính

Trong đó:

- Ký hiệu chưa kết thúc B biểu diễn một công thức.
- Luật sinh $B \rightarrow BB$ biểu diễn sự kề nhau của hai hộp.
- Luật sinh $B \rightarrow B \text{ sub } B$ biểu diễn sự sắp đặt, trong đó hộp chỉ số thứ 2 có kích thước nhỏ hơn, nằm thấp hơn hộp thứ nhất.
- Thuộc tính kế thừa ps (point size - kích thước điểm) phản ánh độ lớn của công thức.
- Luật sinh $B \rightarrow \text{text}$ ứng với luật ngữ nghĩa $B.ht := \text{text.h} * B.ps$ lấy chiều cao thực của text (h) nhân với kích thước điểm của B để có được chiều cao của hộp.
- Luật sinh $B \rightarrow B_1 B_2$ được áp dụng thì B_1, B_2 kế thừa kích thước điểm của B bằng luật copy. Độ cao của B bằng giá trị lớn nhất trong độ cao của B_1, B_2 .
- Khi luật sinh $B \rightarrow B_1 \text{ sub } B_2$ được áp dụng thì hàm shrink sẽ giảm kích thước điểm của B_2 còn 30% và hàm disp đẩy hộp B_2 xuống.

input

BOX **sub** box

a **sub** {i **sup** 2 }

$S \rightarrow$

$\{B.ps := 10\}$

B

$\{S.ht := B.ht\}$

$B \rightarrow$

$\{B_1.ps := B.ps\}$

B_1

$\{B_2.ps := B.ps\}$

B_2

$\{B.ht := \max(B_1.ht, B_2.ht)\}$

$B \rightarrow$

$\{B_1.ps := B.ps\}$

B_1

sub

$\{B_2.ps := \text{shrink}(B.ps)\}$

B_2

$\{B.ht := \text{disp}(B_1.ht, B_2.ht)\}$

$B \rightarrow \text{text}$

$\{B.ht := \text{text.h} * B.ps\}$

output

BOX_{box}

a_{i^2}

5. Dịch từ trên xuống: Loại bỏ đệ quy trái

□ Ở đây chúng ta giải quyết vấn đề chuyển một lược đồ dịch của văn phạm đệ quy trái thành một lược đồ dịch mới không còn đệ quy.

○ Giả sử, ta có lược đồ dịch dạng

$$A \rightarrow A_1 Y \quad \{A.a := g(A_1.a, Y.y) \}$$

$$A \rightarrow X \quad \{A.a := f(X.x) \}$$

Đây là một văn phạm đệ quy trái, áp dụng giải thuật khử đệ quy trái ta được văn phạm không đệ quy trái

$$A \rightarrow X R$$

$$R \rightarrow Y R \mid \varepsilon$$

Bổ sung hành vi ngữ nghĩa cho văn phạm ta được lược đồ dịch:

$$A \rightarrow X \quad \{ R.i := f(X.x) \}$$

$$R \quad \{ A.a := R.s \}$$

$$R \rightarrow Y \quad \{ R_1.i := g(R.i, Y.y) \}$$

$$R_1 \quad \{ R.s := R_1.s \}$$

$$R \rightarrow \varepsilon \quad \{ R.s := R.i \}$$

5. Dịch từ trên xuống: Loại bỏ đệ quy trái

□ Ví dụ: Xét lược đồ dịch của văn phạm đệ quy trái cho biểu thức.

$E \rightarrow E_1 + T$

$\{E.val := E_1.val + T.val\}$

$E \rightarrow E_1 - T$

$\{E.val := E_1.val - T.val\}$

$E \rightarrow T$

$\{E.val := T.val\}$

$T \rightarrow (E)$

$\{T.val := E.val\}$

$T \rightarrow \text{num}$

$\{T.val := \text{num.val}\}$

Vận dụng ý kiến trên ta khử đệ quy trái để được lược đồ dịch không đệ quy trái

$E \rightarrow T \quad \{R.i := T.val\}$

$R \quad \{E.val := R.s\}$

$R \rightarrow +$

$T \quad \{R_1.i := R.i + T.val\}$

$R_1 \quad \{R.s := R_1.s\}$

$R \rightarrow -$

$T \quad \{R_1.i := R.i - T.val\}$

$R_1 \quad \{R.s := R_1.s\}$

$R \rightarrow \varepsilon \quad \{R.s := R.i\}$

$T \rightarrow ($

E

$) \quad \{T.val := E.val\}$

$T \rightarrow \text{num} \quad \{T.val := \text{num.val}\}$

5. Dịch từ trên xuống: Loại bỏ đệ quy trái

□ Ví dụ: Xét lược đồ dịch xây dựng cây cú pháp cho biểu thức

$E \rightarrow E_1 + T$	$\{E.nptr := mknode('+', E_1.nptr, T.nptr) \}$
$E \rightarrow E_1 - T$	$\{E.nptr := mknode('-', E_1.nptr, T.nptr) \}$
$E \rightarrow T$	$\{E.nptr := T.nptr \}$
$T \rightarrow (E)$	$\{T.nptr := E.nptr \}$
$T \rightarrow id$	$\{T.nptr := mkleaf(id, id.entry) \}$
$T \rightarrow num$	$\{T.nptr := mkleaf(num, num.val) \}$

Áp dụng quy tắc khử đệ quy trái trên với $E \approx A$, $+T$, $-T \approx Y$ và $T \approx X$ ta có lược đồ dịch

$E \rightarrow T$	$\{R.i := T.nptr \}$
$R \rightarrow$	$\{E.nptr := R.s \}$
$R \rightarrow +$	
$T \rightarrow$	$\{R_1.i := mknode('+', R.nptr, T.nptr) \}$
$R_1 \rightarrow$	$\{R.s := R_1.s \}$
$R \rightarrow -$	
$T \rightarrow$	$\{R_1.i := mknode('-', R.nptr, T.nptr) \}$
$R_1 \rightarrow$	$\{R.s := R_1.s \}$
$R \rightarrow \epsilon$	$\{R.s := R.i \}$
$T \rightarrow ($	
E	
$)$	$\{T.nptr := E.nptr \}$
$T \rightarrow id$	$\{T.nptr := mkleaf(id, id.entry) \}$
$T \rightarrow num$	$\{T.nptr := mkleaf(num, num.val) \}$

5. Dịch từ trên xuống: Thiết kế bộ dịch

- Giải thuật: Xây dựng bộ dịch trực tiếp cú pháp dự đoán (Predictive - Syntax -Directed Translation) phạm đệ quy trái thành một lược đồ dịch mới không còn đệ quy.
 - Input: Một lược đồ dịch cú pháp trực tiếp với văn phạm có thể phân tích dự đoán.
 - Output: Mã cho bộ dịch trực tiếp cú pháp.

5. Dịch từ trên xuống: Thiết kế bộ dịch

□ Phương pháp:

- 1. Với mỗi ký hiệu chưa kết thúc A, xây dựng một hàm có các tham số hình thức tương ứng với các thuộc tính kế thừa của A và trả về giá trị của thuộc tính tổng hợp của A.
- 2. Mã cho ký hiệu chưa kết thúc A quyết định luật sinh nào được dùng cho ký hiệu nhập hiện hành.
- 3. Mã kết hợp với mỗi luật sinh như sau: chúng ta xem xét token, ký hiệu chưa kết thúc và hành vi bên phải của luật sinh từ trái sang phải
 - i) Đối với token X với thuộc tính tổng hợp x, lưu giá trị của x vào trong biến được khai báo cho X.x. Sau đó phát sinh lời gọi để hợp thức (match) token X và dịch chuyển đầu đọc.
 - ii) Đối với ký hiệu chưa kết thúc B, phát sinh lệnh gán $C := B(b_1, b_2, \dots, b_k)$ với lời gọi hàm trong vế phải của lệnh gán, trong đó b_1, b_2, \dots, b_k là các biến cho các thuộc tính kế thừa của B và C là biến cho thuộc tính tổng hợp của B.
 - iii) Đối với một hành vi, chép mã vào trong bộ phân tích cú pháp, thay thế mỗi tham chiếu tới một thuộc tính bởi biến cho thuộc tính đó.

5. Dịch từ trên xuống: Thiết kế bộ dịch

□ Ví dụ: Xét lược đồ dịch cho việc xây dựng cây cú pháp cho biểu thức. Ta thấy đó là một văn phạm LL(1) nên phù hợp cho phân tích trên xuống.

○ Với 3 ký hiệu chưa kết thúc E, R, T ta xây dựng 3 hàm tương ứng:

```
function E: ↑ syntax - tree - node;      /* E không có thuộc tính kế thừa */
```

```
function R (i : ↑ syntax - tree - node) : ↑ syntax - tree - node
```

```
function T : ↑ syntax - tree - node;
```


5. Dịch từ trên xuống: Thiết kế bộ dịch

Dùng token `addop` biểu diễn cho `+` và `-` ta có thể kết hợp hai luật sinh thành một luật sinh mới

$R \rightarrow \text{addop}$
 $T \quad \{R_1.i := \text{mknode}(\text{addop.lexeme}, R.i, T.nptr) \}$
 $R_1 \quad \{R.s := R_1.s \}$
 $R \rightarrow \varepsilon \quad \{R.s := R.i \}$

Ta có hàm `R` như sau:

```

function R(i : ↑syntax_tree_node) : ↑syntax_tree_node;
var nptr, i1, s1, s : ↑syntax_tree_node;
      addoplexeme : char;
begin
  if lookahead = addop then
    begin /* luật sinh  $R \rightarrow \text{addop TR}$  */
      addoplexeme := lexval;
      match(addop);
      nptr := T;
      i1 := mknode(addoplexeme, i, nptr);
      s1 := R(i1);
      s := s1;
    end
  else s := i; /* Luật sinh  $R \rightarrow \varepsilon$  */
  return s;

```

1. Xây dựng một cây phân tích cú pháp chú thích cho biểu thức số học sau:
 $(4 * 7 + 1) * 2$
2. Xây dựng một cây phân tích cú pháp và cây cú pháp cho biểu thức $((a) + (b))$ theo:
 - a) Định nghĩa trực tiếp cú pháp cho biểu thức số học.
 - b) Lược đồ dịch cho biểu thức số học.
3. Xây dựng một DAG cho các biểu thức sau đây:
 - a) $a + a + (a + a + a + (a + a + a + a))$
 - b) $x * (3 * x + x * x)$

4. Văn phạm sau đây sinh ra các biểu thức có được khi áp dụng một toán tử số học + cho các hằng số nguyên và số thực. Khi 2 số nguyên được công lại, kiểu kết quả là kiểu nguyên, ngược lại nó là kiểu số thực.

$E \rightarrow E + T \mid T$

$T \rightarrow \text{num} \bullet \text{num} \mid \text{num}$

a) Đưa ra một định nghĩa trực tiếp cú pháp xác định kiểu của mỗi biểu thức con.

b) Mở rộng định nghĩa trực tiếp cú pháp trên để dịch các biểu thức thành ký pháp hậu tố và xác định các kiểu. Dùng toán tử một ngôn ngữ inttoreal để chuyển một giá trị nguyên thành giá trị thực tương đương mà nhờ đó cả hai toán hạng của + ở dạng hậu tố có cùng kiểu