

第五章 语法制导翻译

语法制导翻译概述

哈尔滨工业大学 陈鄞



什么是语法制导翻译

➤ 编译的阶段

➤ 词法分析

➤ 语法分析

➤ 语义分析

➤ 中间代码生成

➤ 代码优化

➤ 目标代码生成

语义翻译

语法制导翻译
(Syntax-Directed Translation)

语法制导翻译使用 *CFG* 来引导对语言的翻译，
是一种面向文法的翻译技术

语法制导翻译的基本思想

➤ 如何表示语义信息？

- 为 CFG 中的文法符号设置语义属性，用来表示语法成分对应的语义信息

➤ 如何计算语义属性？

- 文法符号的语义属性值是用与文法符号所在产生式（语法规则）相关联的语义规则来计算的
- 对于给定的输入串 x ，构建 x 的语法分析树，并利用与产生式（语法规则）相关联的语义规则来计算分析树中各结点对应的语义属性值

两个概念

- 将语义规则同语法规则（产生式）联系起来要涉及两个概念
- 语法制导定义 (*Syntax-Directed Definitions, SDD*)
- 语法制导翻译方案 (*Syntax-Directed Translation Scheme, SDT*)

语法制导定义(*SDD*)

- *SDD* 是对 *CFG* 的推广
 - 将每个 **文法符号** 和一个 **语义属性** 集合相关联
 - 将每个 **产生式** 和一组 **语义规则** 相关联，这些规则用于计算该产生式中各文法符号的属性值
- 如果 X 是一个文法符号， a 是 X 的一个属性，则用 $X.a$ 表示属性 a 在某个标号为 X 的分析树结点上的值

语法制导定义(SDD)

➤ SDD 是对 CFG 的推广

➤ 将每个文法符号和一个语义属性集合相关联

➤ 将每个产生式和一组语义规则相关联，这些规则用于计算该产生式中各文法符号的属性值

➤ 例

产生式	语义规则
$D \rightarrow T L$	$L.inh = T.type$
$T \rightarrow \text{int}$	$T.type = \text{int}$
$T \rightarrow \text{real}$	$T.type = \text{real}$
$L \rightarrow L_1, \text{id}$	$L_1.inh = L.inh$
...	...

语法制导翻译方案(*SDT*)

- *SDT*是在产生式右部嵌入了程序片段的*CFG*，这些程序片段称为语义动作。按照惯例，语义动作放在花括号内

➤ 例

$$\begin{aligned} D &\rightarrow T \{ L.inh = T.type \} L \\ T &\rightarrow \text{int} \{ T.type = \text{int} \} \\ T &\rightarrow \text{real} \{ T.type = \text{real} \} \\ L &\rightarrow \{ L_1.inh = L.inh \} L_1, \text{id} \\ &\dots \end{aligned}$$

一个语义动作在产生式中的位置决定了这个动作的执行时间


SDD与SDT

➤ ***SDD***

- 是关于语言翻译的高层次规格说明
- 隐蔽了许多具体实现细节，使用户不必显式地说明翻译发生的顺序

➤ ***SDT***

- 可以看作是对*SDD*的一种补充，是*SDD*的具体实施方案
- 显式地指明了语义规则的计算顺序，以便说明某些实现细节




第五章 语法制导翻译

语法制导翻译概述

哈尔滨工业大学 陈鄞





第五章 语法制导翻译

语法制导定义SDD

哈尔滨工业大学 陈鄞



语法制导定义 *SDD*

- 语法制导定义 *SDD* 是对 *CFG* 的推广
 - 将每个 **文法符号** 和一个 **语义属性** 集合相关联
 - 将每个 **产生式** 和一组 **语义规则** 相关联，用来计算该产生式中各文法符号的属性值
- 文法符号的属性
 - 综合属性 (*synthesized attribute*)
 - 继承属性 (*inherited attribute*)

综合属性 (*synthesized attribute*)

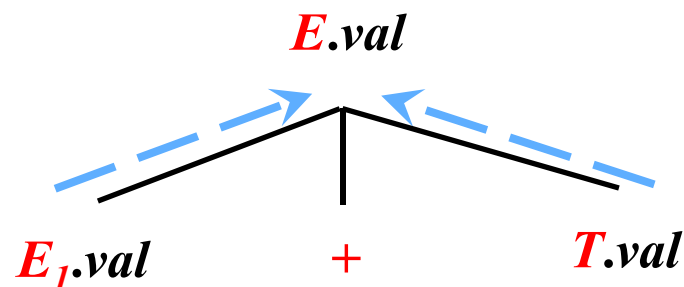
- 在分析树结点 N 上的非终结符 A 的 **综合属性** 只能通过 N 的子结点或 N 本身的属性值来定义

例

产生式

语义规则

$E \rightarrow E_1 + T$ $E.val = E_1.val + T.val$



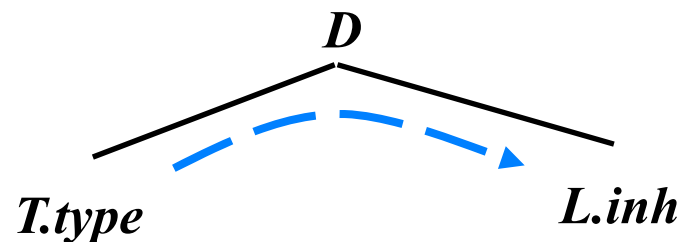
- 终结符可以具有 **综合属性**。终结符的综合属性值是由词法分析器提供的 **词法值**，因此在 *SDD* 中没有计算终结符属性值的语义规则

继承属性 (*inherited attribute*)

➤ 在分析树结点 N 上的非终结符 A 的继承属性只能通过 N 的父结点、 N 的兄弟结点或 N 本身的属性值来定义

➤ 例

产生式	语义规则
$D \rightarrow T L$	$L.inh = T.type$



➤ 终结符没有继承属性。终结符从词法分析器处获得的属性值被归为综合属性值

例：带有综合属性的SDD

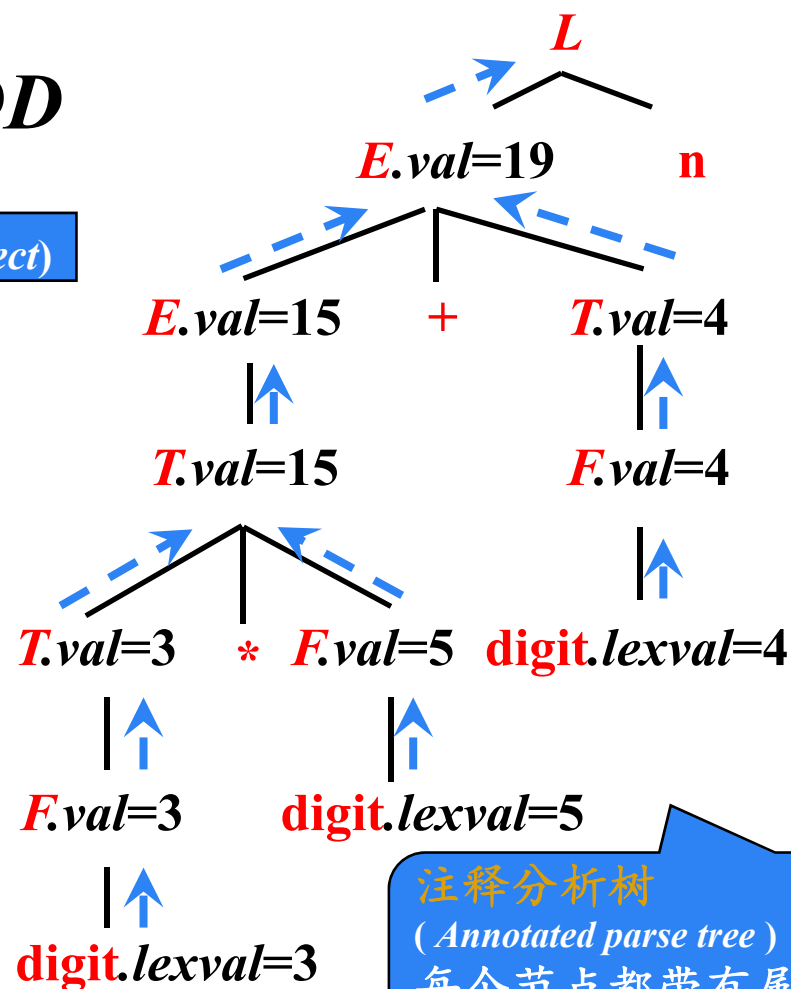
SDD:

副作用 (Side effect)

产生式	语义规则
(1) $L \rightarrow E n$	$\text{print}(E.val)$
(2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
(3) $E \rightarrow T$	$E.val = T.val$
(4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
(5) $T \rightarrow F$	$T.val = F.val$
(6) $F \rightarrow (E)$	$F.val = E.val$
(7) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

输入:

$3*5+4n$



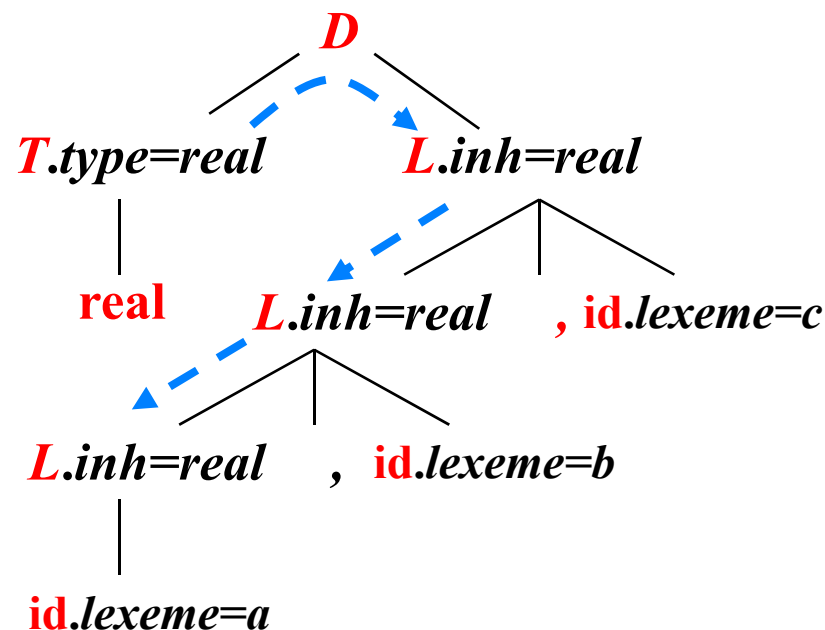
例：带有继承属性 $L.in$ 的SDD

SDD:

	产生式	语义规则
(1)	$D \rightarrow T L$	$L.inh = T.type$
(2)	$T \rightarrow \text{int}$	$T.type = \text{int}$
(3)	$T \rightarrow \text{real}$	$T.type = \text{real}$
(4)	$L \rightarrow L_1, \text{id}$	$L_1.inh = L.inh$ $\text{addtype}(\text{id.lexeme}, L.inh)$
(5)	$L \rightarrow \text{id}$	$\text{addtype}(\text{id.lexeme}, L.inh)$

输入:

$\text{real } a, b, c$




属性文法 (*Attribute Grammar*)

- 一个没有副作用的*SDD*有时也称为属性文法
- 属性文法的规则仅仅通过其它属性值和常量来定义一个属性值

➤ 例

产生式	语义规则
(1) $L \rightarrow E \text{ n}$	$L.val = E.val$
(2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
(3) $E \rightarrow T$	$E.val = T.val$
(4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
(5) $T \rightarrow F$	$T.val = F.val$
(6) $F \rightarrow (E)$	$F.val = E.val$
(7) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$




第五章 语法制导翻译

语法制导定义SDD

哈尔滨工业大学 陈鄞





第五章 语法制导翻译

SDD的求值顺序

哈尔滨工业大学 陈鄞



***SDD*的求值顺序**

- *SDD*为*CFG*中的文法符号设置语义属性。对于给定的输入串 x ，应用语义规则计算分析树中各结点对应的属性值
- 按照什么顺序计算属性值？
 - 语义规则建立了属性之间的依赖关系，在对语法分析树节点的一个属性求值之前，必须首先求出这个属性值所依赖的所有属性值

依赖图 (Dependency Graph)

- 依赖图是一个描述了分析树中结点属性间依赖关系的有向图
- 分析树中每个标号为 X 的结点的每个属性 a 都对应着依赖图中的一个结点
- 如果属性 $X.a$ 的值依赖于属性 $Y.b$ 的值，则依赖图中有一条从 $Y.b$ 的结点指向 $X.a$ 的结点的有向边

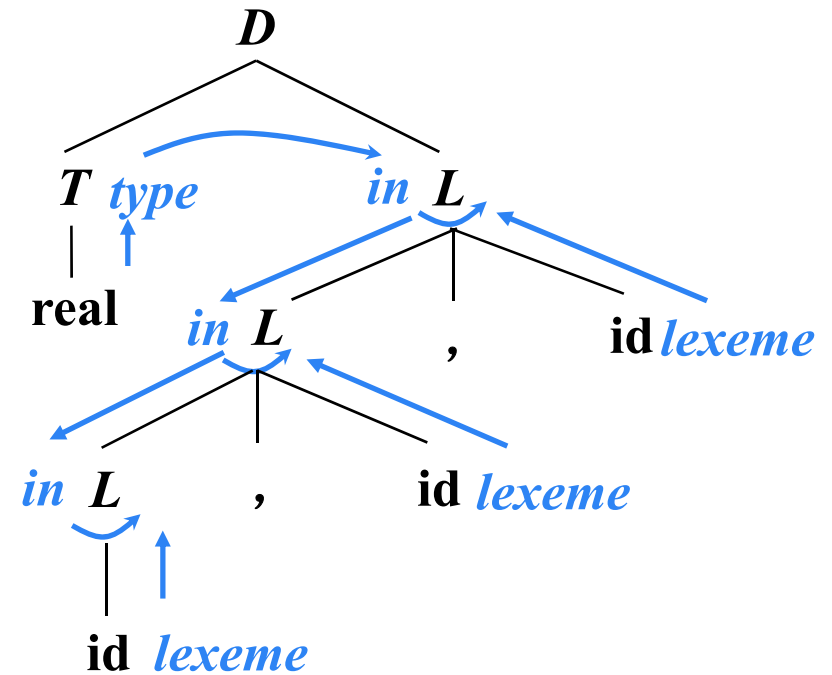
例

SDD:

	产生式	语义规则
(1)	$D \rightarrow T L$	$L.in = T.type$
(2)	$T \rightarrow \text{int}$	$T.type = \text{int}$
(3)	$T \rightarrow \text{real}$	$T.type = \text{real}$
(4)	$L \rightarrow L_1, \text{id}$	$L_1.in = L.in$ $\text{addtype}(\text{id.lexeme}, L.in)$
(5)	$L \rightarrow \text{id}$	$\text{addtype}(\text{id.lexeme}, L.in)$

输入:

$\text{real } a, b, c$



属性值的计算顺序

- 可行的求值顺序是满足下列条件的结点序列 N_1, N_2, \dots, N_k : 如果依赖图中有一条从结点 N_i 到 N_j 的边 ($N_i \rightarrow N_j$), 那么 $i < j$ (即: 在节点序列中, N_i 排在 N_j 前面)
- 这样的排序将一个有向图变成了一个线性排序, 这个排序称为这个图的 **拓扑排序** (*topological sort*)

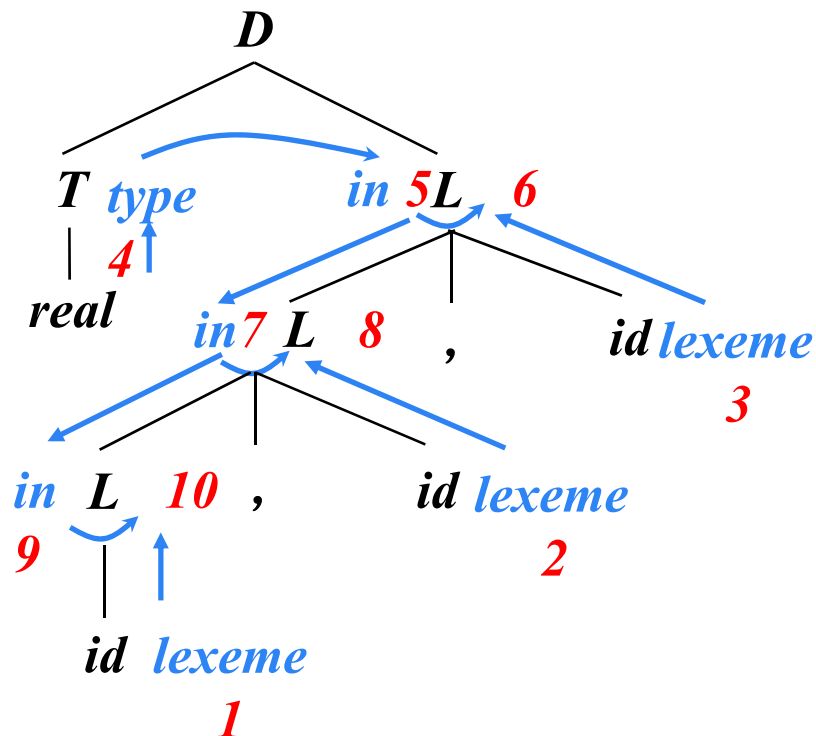
例

SDD:

	产生式	语义规则
(1)	$D \rightarrow T L$	$L.in = T.type$
(2)	$T \rightarrow \text{int}$	$T.type = \text{int}$
(3)	$T \rightarrow \text{real}$	$T.type = \text{real}$
(4)	$L \rightarrow L_1, \text{id}$	$L_1.in = L.in$ $\text{addtype}(\text{id.lexeme}, L.in)$
(5)	$L \rightarrow \text{id}$	$\text{addtype}(\text{id.lexeme}, L.in)$

输入:

real a, b, c



拓扑排序:

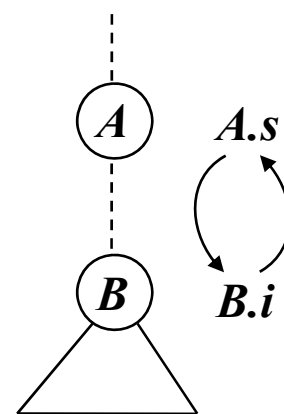
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

4, 3, 2, 1, 5, 7, 6, 9, 8, 10


- 对于只具有综合属性的 *SDD*，可以按照任何自底向上的顺序计算它们的值
- 对于同时具有继承属性和综合属性的 *SDD*，不能保证存在一个顺序来对各个节点上的属性进行求值


➤ 例

产生式	语义规则
$A \rightarrow B$	$A.s = B.i$ $B.i = A.s + 1$



如果图中没有环，那么至少存在一个拓扑排序

- 
- 从计算的角度看，给定一个 SDD ，很难确定是否存在某棵语法分析树，使得 SDD 的属性之间存在循环依赖关系
 - 幸运的是，存在一个 SDD 的有用子类，它们能够保证对每棵语法分析树都 **存在一个求值顺序**，因为它们不允许产生带有环的依赖图
 - 不仅如此，接下来介绍的两类 SDD 可以和 **自顶向下** 及 **自底向上** 的语法分析过程一起高效地实现
 - S -属性定义 (S -Attributed Definitions, **S - SDD**)
 - L -属性定义 (L -Attributed Definitions, **L - SDD**)



第五章 语法制导翻译

SDD的求值顺序

哈尔滨工业大学 陈鄞





第五章 语法制导翻译

S-属性定义与L-属性定义

哈尔滨工业大学 陈鄞



S-属性定义

- 仅仅使用综合属性的SDD称为S属性的SDD，或S-属性定义、S-SDD

➤ 例

产生式	语义规则
(1) $L \rightarrow E n$	$L.val = E.val$
(2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
(3) $E \rightarrow T$	$E.val = T.val$
(4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
(5) $T \rightarrow F$	$T.val = F.val$
(6) $F \rightarrow (E)$	$F.val = E.val$
(7) $F \rightarrow \text{digit}$	$F.val = \text{digit.lexval}$

- 如果一个SDD是S属性的，可以按照语法分析树节点的任何自底向上顺序来计算它的各个属性值
- S-属性定义可以在自底向上的语法分析过程中实现

L -属性定义


- L -属性定义(也称为 L 属性的 SDD 或 L - SDD)的
直观含义: 在一个产生式所关联的各属性之间,
依赖图的边可以从左到右, 但不能从右到左
(因此称为 L 属性的, L 是 $Left$ 的首字母)

***L-SDD*的正式定义**

- 一个*SDD*是***L-属性定义***，当且仅当它的每个属性要么是一个**综合属性**，要么是满足如下条件的继承属性：假设存在一个产生式 $A \rightarrow X_1X_2...X_n$ ，其右部符号 $X_i (1 \leq i \leq n)$ 的**继承属性**仅依赖于下列属性：
 - A 的**继承属性**
 - 产生式中 X_i **左边**的符号 X_1, X_2, \dots, X_{i-1} 的属性
 - X_i 本身的属性，但 X_i 的全部属性不能在依赖图中形成环路

每个*S-属性定义*都是*L-属性定义*

L -SDD的正式定义

- 一个SDD是 **L -属性定义**，当且仅当它的每个属性要么是一个**综合属性**，要么是满足如下条件的继承属性：假设存在一个产生式 $A \rightarrow X_1X_2 \dots X_n$ ，其右部符号 $X_i (1 \leq i \leq n)$ 的**继承属性**仅依赖于下列属性：
 - A 的**继承属性**...  为什么不能是综合属性？
 - 产生式中 X_i **左边**的符号 X_1, X_2, \dots, X_{i-1} 的属性
 - X_i 本身的属性，但 X_i 的全部属性不能在依赖图中形成环路

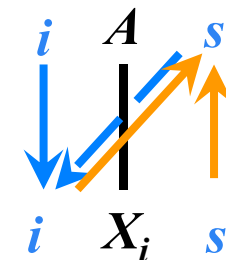
L -SDD的正式定义

- 一个SDD是 **L -属性定义**，当且仅当它的每个属性要么是一个**综合属性**，要么是满足如下条件的继承属性：假设存在一个产生式 $A \rightarrow X_1X_2 \dots X_n$ ，其右部符号 $X_i (1 \leq i \leq n)$ 的**继承属性**仅依赖于下列属性：

- A 的**继承属性**

- 产生式中 X_i **左边**的符号 X_1, X_2, \dots, X_{i-1} 的属性

- X_i 本身的属性，但 X_i 的全部属性不能在依赖图中形成环路



例：L-SDD

	产生式	语义规则
(1)	$T \rightarrow F T'$	$\underline{T'.inh} = F.val$ $\underline{T.val} = \underline{T'.syn}$
(2)	$T' \rightarrow * F T_1'$	$\underline{T_1'.inh} = \underline{T'.inh} \times F.val$ $\underline{T'.syn} = \underline{T_1'.syn}$
(3)	$T' \rightarrow \varepsilon$	$\underline{T'.syn} = \underline{T'.inh}$
(4)	$F \rightarrow \text{digit}$	$\underline{F.val} = \underline{\text{digit.lexval}}$

继承属性

综合属性

非L属性的SDD

➤ 例

产生式	语义规则
(1) $A \rightarrow LM$	$L.i = l(A.i)$ $M.i = m(L.s)$ $A.s = f(M.s)$
(2) $A \rightarrow QR$	$R.i = r(A.i)$ $Q.i = q(R.s) \times$ $A.s = f(Q.s)$

继承属性

综合属性



第五章 语法制导翻译

S-属性定义与L-属性定义

哈尔滨工业大学 陈鄞

