Compiler Design

Fatemeh Deldar

Isfahan University of Technology

1403-1404

Semantic Analysis & Syntax-Directed Translation

Syntax-Directed Translation

- همان‌طور که در بخش قبل گفته شد، کنترل‌های معنایی را اغلب نمی‌توان با گرامرهای مستقل از متن اعمال کرد
- برای پیاده‌سازی تحلیل‌گر معنایی از گرامری به نام گرامر خصیصه (attribute grammar) استفاده می‌شود
- هدف از این فصل این است که نشان دهیم چگونه کامپایلر دنباله‌ای از توکن‌ها را ترجمه می‌کند
  - ترجمه صرفاً به معنای تولید کد میانی یا نهایی نیست، بلکه شامل اضافه‌کردن اطلاعاتی راجع به شناسه‌ها در جدول نمادها، گزارش خطاهای معنایی و غیره نیز است
- در این فصل به بررسی تکنیک ترجمه‌ای خواهیم پرداخت که عمل ترجمه یک ساختار را با استفاده از گرامر آن ساختار انجام می‌دهد و به همین جهت به آن ترجمه مبتنی بر نحو (syntax-directed translation) گفته می‌شود

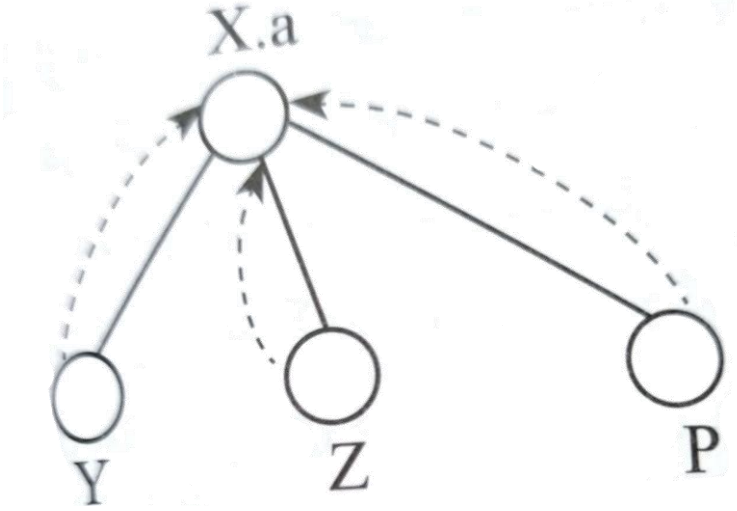
Attribute Grammar

- با توجه به اینکه برای توصیف ساختار زبان برنامه‌نویسی از گرامر مستقل از متن استفاده می‌شود، می‌توان برای انجام ترجمه مبتنی بر نحو با اضافه‌کردن مجموعه‌ای از خصیصه‌ها (attribute) به نمادهای گرامر (پایانه‌ها و غیرپایانه‌ها) و مجموعه‌ای از قوانین معنایی (semantic rules) به قواعد گرامر، گرامر زبان را توسعه داد
  - قوانین معنایی برای ارزیابی و مقداردهی خصیصه‌ها به کار می‌روند
- بسته به کاربرد و هدف ترجمه، برای هر نماد گرامر خصیصه‌های مختلفی می‌توان تعریف کرد
- به‌عنوان مثال برای تعیین نوع داده‌ای عبارت، نگهداری کد میانی معادل یک ساختار و ... می‌توان خصیصه‌های مختلفی تعریف کرد
- خصیصه‌ها به دو شکل هستند:
  - خصیصه ساختگی (synthesized attribute)
  - خصیصه موروثی (inherited attribute)

Synthesized Attribute

- یک خصیصه مربوط به یک گره از درخت تجزیه **synthesized** است در صورتی که مقدارش از روی مقادیر خصیصه‌های فرزندان گره و خصیصه‌های همان گره محاسبه شود

- **مثال**
  - در درخت شکل زیر **a** یک خصیصه **synthesized** برای گره **X** است زیرا مقدار **X.a** از روی مقادیر خصیصه‌های فرزندان **X** و خصیصه‌های خود گره **X** به دست می‌آید.
  - در این مثال یال‌های خط‌چین جهت‌دار وابستگی را نشان می‌دهد. به‌عنوان مثال یال جهت‌دار از **P** به **X** به این معناست که برای ارزیابی خصیصه **a** گره **X** به خصیصه‌های گره **P** نیاز است.



Synthesized Attribute

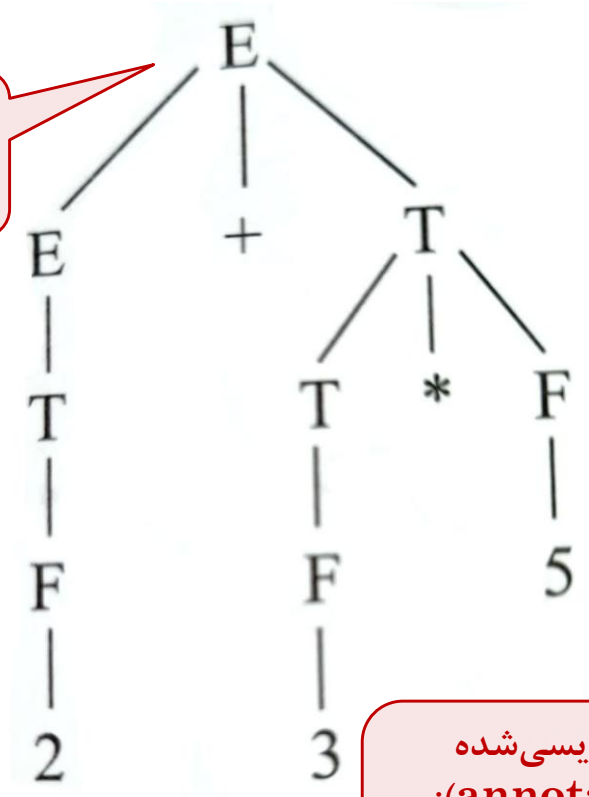
- **مثال**
  - گرامر زیر برای ساخت عبارت‌های میانوندی را در نظر بگیرید:
    - $E \rightarrow E + T \mid T$
    - $T \rightarrow T * F \mid F$
    - $F \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$
  - با تعریف خصیصه‌ای به نام t برای غیر پایانه‌های E و T و F و تعریف قوانین معنایی برای ارزیابی و محاسبه مقدار این خصیصه، می‌خواهیم عبارت میانوندی را به یک عبارت پسوندی معادل تبدیل کنیم.
  - عملگر || بیانگر الحاق است

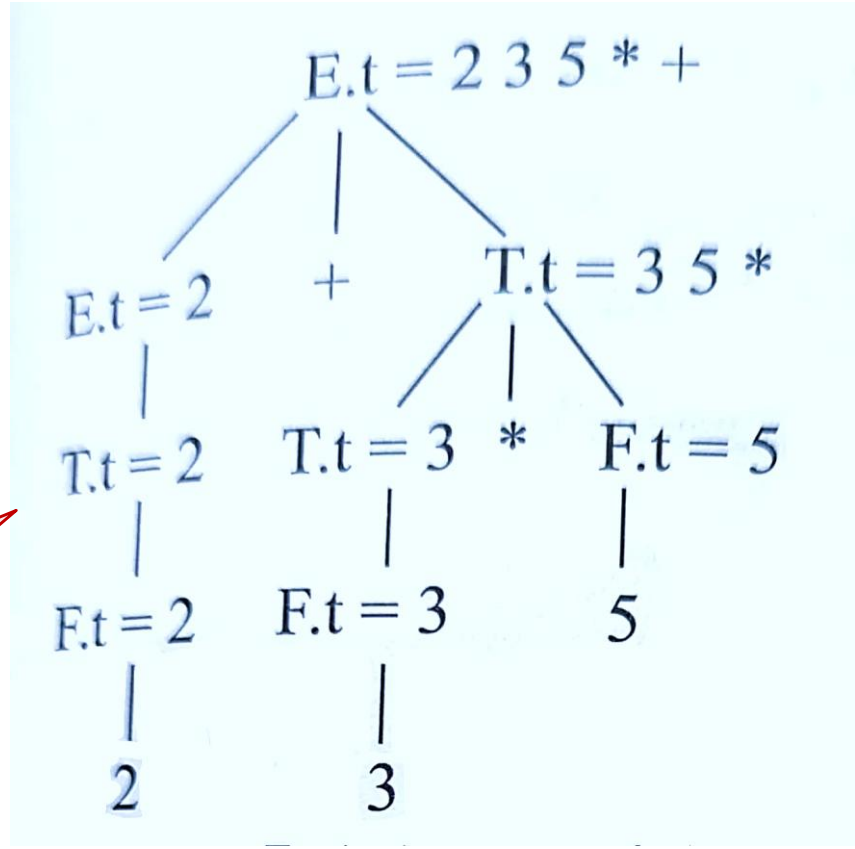
قواعد گرامر	قوانین معنایی
$E \rightarrow E_1 + T$	$E.t := E_1.t \mid\mid T.t \mid\mid '+'$
$E \rightarrow T$	$E.t := T.t$
$T \rightarrow T_1 * F$	$T.t := T_1.t \mid\mid F.t \mid\mid '*'$
$T \rightarrow F$	$T.t := F.t$
$F \rightarrow 0$	$F.t := '0'$
$F \rightarrow 1$	$F.t := '1'$
...	...
$F \rightarrow 9$	$F.t := '9'$

Synthesized Attribute

- مثال (ادامه)

درخت تجزیه
عبارت 2+3*5



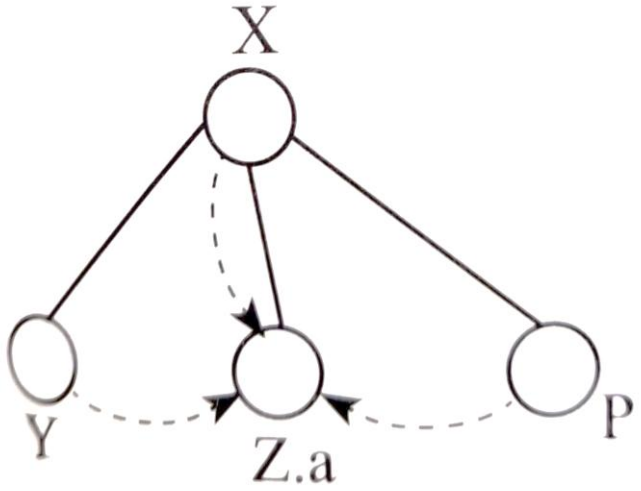


درخت تجزیه حاشیه‌نویسی‌شده (annotated parse tree):
درخت تجزیه‌ای که مقادیر خصیصه‌ها در هر گره را نشان می‌دهد.

Inherited Attribute

- یک خصیصه مربوط به یک گره از درخت تجزیه **inherited** است در صورتی که مقدارش از روی مقادیر خصیصه‌های پدر و **sibling**های گره محاسبه شود

- **مثال**
  - در درخت شکل زیر **a** یک خصیصه **inherited** برای گره **Z** است زیرا مقدار **Z.a** از روی مقادیر خصیصه‌های **X**، **Y** و **P** به دست می‌آید.



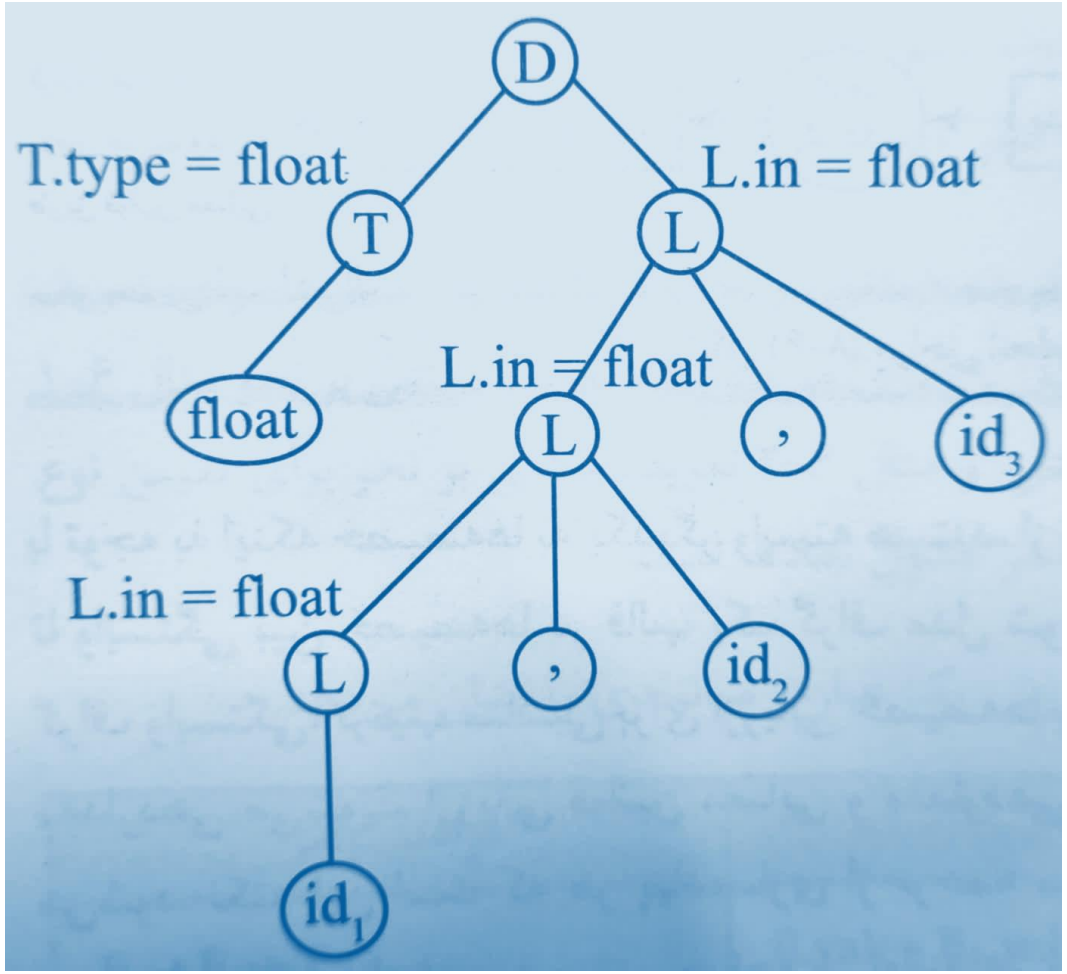
Inherited Attribute

- **مثال**
- شکل زیر تعریف یک گرامر به همراه قوانین معنایی جهت ارزیابی خصیصه‌ها را نشان می‌دهد. این تعریف اطلاعات مربوط به لیستی از متغیرها را به جدول نماد اضافه می‌کند.

قواعد گرامر	قوانین معنایی
$D \rightarrow T\ L$	L.in := T. type
$T \rightarrow char$	T.type := char
$T \rightarrow float$	T.type := float
$L \rightarrow L_1, id$	L_1.in := L.in addtype (id.entry, L.in)
$L \rightarrow id$	addtype (id.entry, L.in)

- در این مثال خصیصه **type** یک خصیصه **synthesized** و خصیصه **in** یک خصیصه **inherited** است
- خصیصه **entry** برای توکن **id** اشاره‌گر به مکانی از جدول نماد است که اطلاعات **id** در آن قرار دارد. مقدار خصیصه **entry** توسط تحلیل‌گر لغوی مشخص شده است.
- روال **addType** نوع متغیر که توسط **L.in** مشخص می‌شود را در جدول نماد قرار می‌دهد.

Inherited Attribute



- **مثال (ادامه)**
  - درخت تجزیه حاشیه‌نویسی‌شده برای رشته **float id1, id2, id3**
  - در هر گره که مربوط به یک **id** است روال **addtype** فراخوانی می‌شود تا نوع آن شناسه را وارد جدول نماد کند.
- **نکته**
  - پایانه‌ها فقط دارای خصیصه‌های **synthesized** هستند و معمولاً مقادیر خصیصه‌های پایانه‌ها توسط تحلیل‌گر لغوی مشخص می‌شود.

Syntax-Directed Translation

- مهم‌ترین کاربردهای ترجمه مبتنی بر نحو
  - ایجاد و مدیریت جدول نمادها و قراردادن اطلاعات جمع‌آوری‌شده راجع به idها داخل جدول
  - ترجمه درخت تجزیه برای تحلیل معنایی و تولید کد میانی
- ترجمه مبتنی بر نحو به دو شکل قابل انجام است:
  1. تعریف مبتنی بر نحو (Syntax-Directed Definition)
  2. طرح ترجمه مبتنی بر نحو (Syntax-Directed Translation Scheme)

Syntax-Directed Definitions

- A syntax-directed definition (SDD) is a context-free grammar together with attributes and rules
- **Attributes** are associated with ***grammar symbols*** and **rules** are associated with ***productions***
- ***The two previous examples were two instances of syntax-directed definitions.***

Syntax-Directed Definitions

- **Example**
  - The following SDD evaluates expressions terminated by an endmarker ***n***
  - In the SDD, each of the non-terminals has a single synthesized attribute, called *val*
  - The terminal digit has a synthesized attribute *lexval*

	PRODUCTION	SEMANTIC RULES
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)	$F \rightarrow$ **digit**	$F.val =$ **digit**.lexval

Evaluating an SDD at the Nodes of a Parse Tree

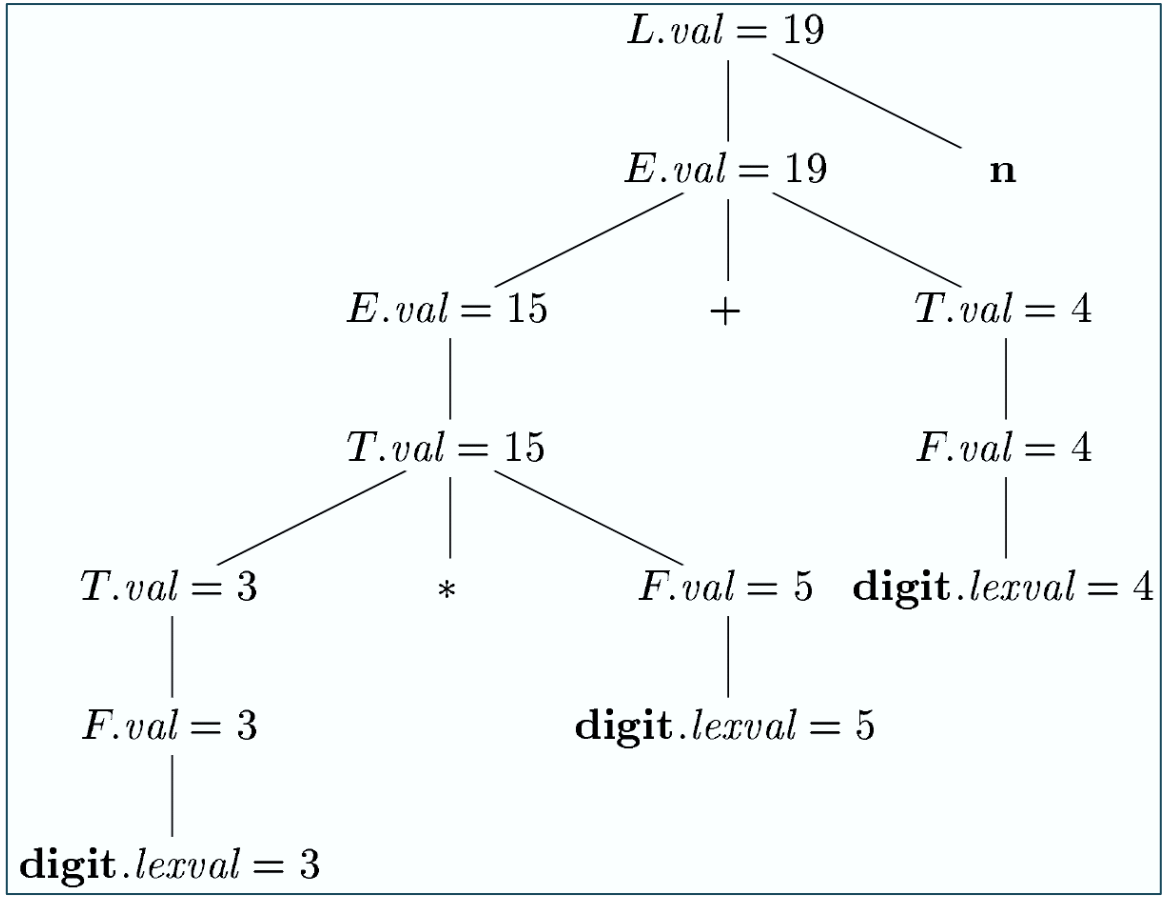
- As we said before, a parse tree, showing the value(s) of its attribute(s) is called an ***annotated parse tree***
- With synthesized attributes, we can evaluate attributes in any bottom-up order, such as that of a postorder traversal of the parse tree
- For SDD's with both inherited and synthesized attributes, there is no guarantee that there is even one order in which to evaluate attributes at nodes
  - **Example**
    - The following rules are circular; it is impossible to evaluate either $A.s$ at a node N or $B.i$ at the child of N without first evaluating the other

PRODUCTION	SEMANTIC RULES
$A \rightarrow B$	$A.s = B.i;$ $B.i = A.s + 1$

Evaluating an SDD at the Nodes of a Parse Tree

- **Example**
  - Annotated parse tree for $3 * 5 + 4$ ***n***

	PRODUCTION	SEMANTIC RULES
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)	$F \rightarrow$ **digit**	$F.val =$ **digit**.lexval



Evaluating an SDD at the Nodes of a Parse Tree

- **Example**
  - Annotated parse tree for $(3 + 4) * (5 + 6)$ ***n***

	PRODUCTION	SEMANTIC RULES
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)	$F \rightarrow$ **digit**	$F.val =$ **digit**.lexval