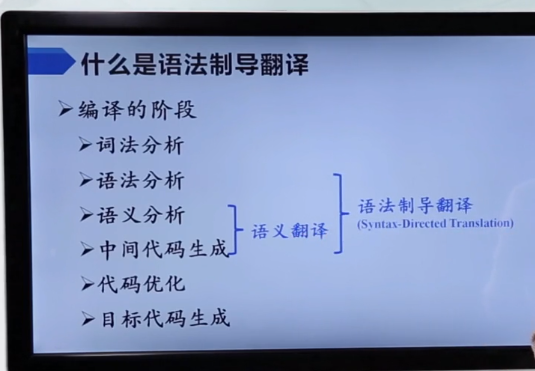
第三阶段：

语义分析的结果是中间代码。

所以语义分析经常与中间代码一起实现，称作语义翻译

. 

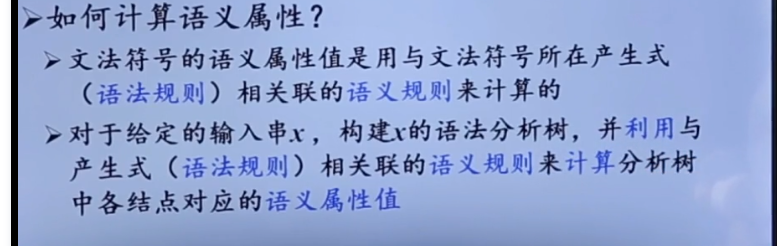
而通过语法制导翻译，可以将语法分析与语义翻译一起实现，也就是二三阶段合并

语法分析阶段可以顺便把语义翻译的地基也就是AST tree打了

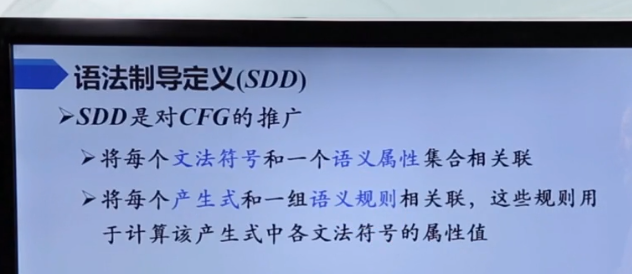
语法致导翻译基本思想：  
为文法符号设置语义属性，表示对应的信息 //文法符号， terminal and nonterminal

例如一个变量，他的值是多少，位置等等

如何计算语义属性

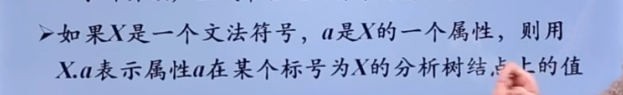


构建它的语法分析树，并利用与产生式相关联的语义规则计算各个节点的语义属性

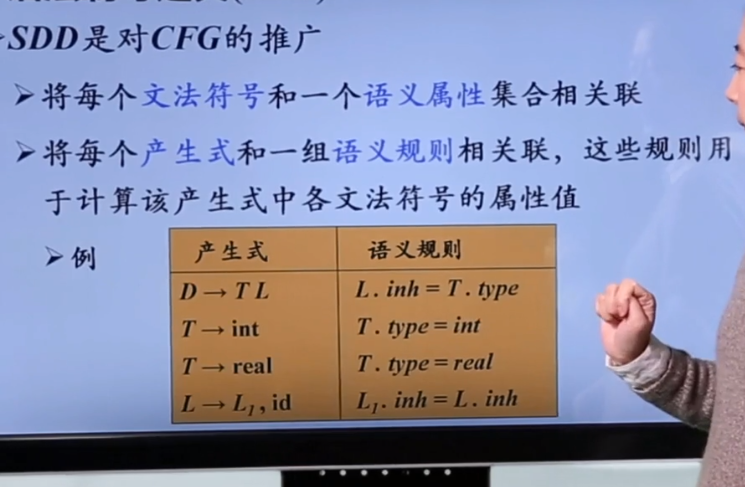


语法致导定义，每一个文法符号与一个语义属性集合 相关联

将每个grammar rule和一组语义规则相关联， 这些规则 能产生各个文法符号terminal nonterminal的属性值



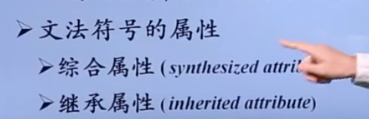
X.a 等于某个节点X的值



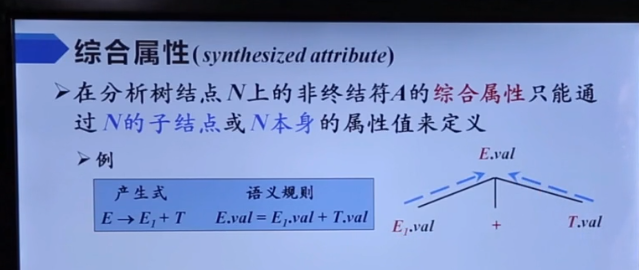
产生式并没有给我们的代码实际意思，我们要给产生式加上对应的语义规则，

如果有T->INT 那么T.TYPE=int

L1与L是一个 non-terminal，只不过 这是L出现在不同的位置，因此加上 下标区分

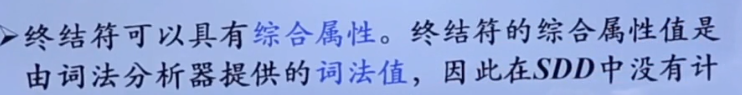


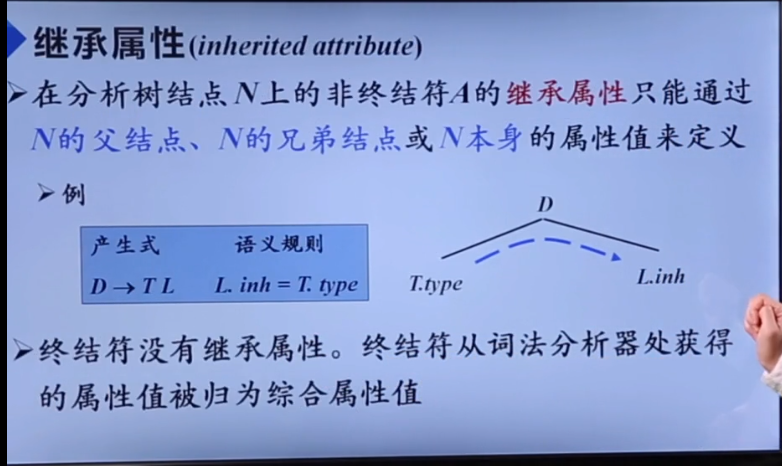
文法符号的属性分为综合属性和继承属性



树节点的非终结符的synthesized attribute只能通过他的子节点或者他本身的属性来定义

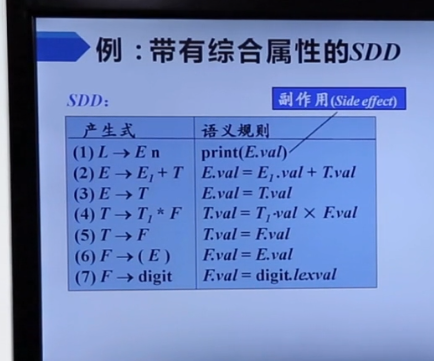
E1就是E，为了防止混淆写成E1





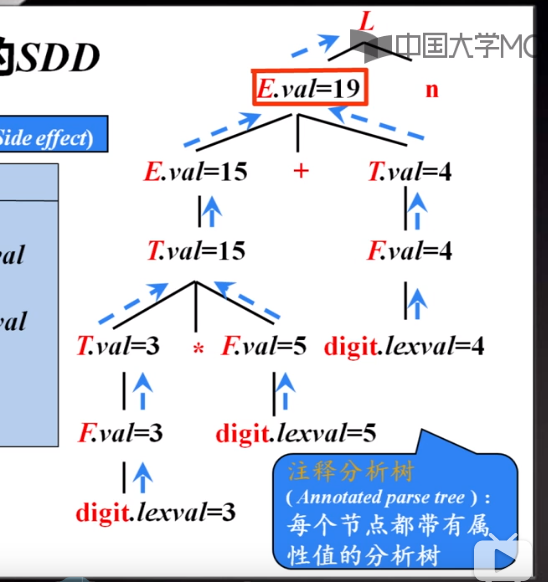
.继承属性：从兄弟节点与父节点获得的属性

终结符的属性是由词法分析器给的，算作综合属性

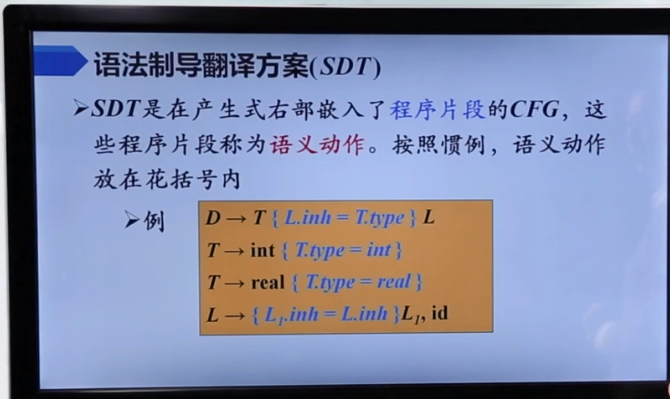


可以看见综合属性都是左边属性等于子节点属性

第一个是副作用，他目的只是单纯的print，但也是子节点调用的



从下往上看，digit.lexval是词法分析器提供的词法值



括号内就是语义动作

意思就是当遇到T以后，我们就可以用T的type值计算L的inh属性

遇到T->int后，我们就能让T的type为int

最下面的，在转换L1之前，我们就能根据L的Inh属性计算L1的InhInh属性