

Syntax (VII)

Abstract Syntax Tree

- It's also worth noting that the [output of syntactic analyzer is an abstract syntax tree](#).
- So, what is an abstract syntax tree? We are going to use Clite to illustrate it.
- Before moving forward to the details of abstract syntax tree, let's take a look at Clite syntax first.
- Clite is a subset of C language. Its syntax is given below.
 - {} and [] in bold are EBNF meta-symbols.

Concrete Syntax of Clite (from Tucker and Noonan, 2007).

$Program \rightarrow \text{int main } () \{ \textit{Declarations} \textit{Statements} \}$
 $\textit{Declarations} \rightarrow \{ \textit{Declaration} \}$
 $\textit{Declaration} \rightarrow \textit{Type} \textit{Identifier} [[\textit{Integer}]] \{ , \textit{Identifier} [[\textit{Integer}]] \}$
 $\textit{Type} \rightarrow \text{int} \mid \text{bool} \mid \text{float} \mid \text{char}$
 $\textit{Statements} \rightarrow \{ \textit{Statement} \}$
 $\textit{Statement} \rightarrow ; \mid \textit{Block} \mid \textit{Assignment} \mid \textit{IfStatement} \mid \textit{WhileStatement}$
 $\textit{Block} \rightarrow \{ \textit{Statements} \}$
 $\textit{Assignment} \rightarrow \textit{Identifier} [[\textit{Expression}]] = \textit{Expression};$
 $\textit{IfStatement} \rightarrow \text{if } (\textit{Expression}) \textit{Statement} [\text{else } \textit{Statement}]$
 $\textit{WhileStatement} \rightarrow \text{while } (\textit{Expression}) \textit{Statement}$

$\textit{Expression} \rightarrow \textit{Conjunction} \{ \mid \mid \textit{Conjunction} \}$
 $\textit{Conjunction} \rightarrow \textit{Equality} \{ \&\& \textit{Equality} \}$
 $\textit{Equality} \rightarrow \textit{Relation} [\textit{EquOp} \textit{Relation}]$
 $\textit{EquOp} \rightarrow == \mid !=$
 $\textit{Relation} \rightarrow \textit{Addition} [\textit{RelOp} \textit{Addition}]$
 $\textit{RelOp} \rightarrow < \mid < = \mid > \mid > =$
 $\textit{Addition} \rightarrow \textit{Term} \{ \textit{AddOp} \textit{Term} \}$
 $\textit{AddOp} \rightarrow + \mid -$
 $\textit{Term} \rightarrow \textit{Factor} \{ \textit{MulOp} \textit{Factor} \}$
 $\textit{MulOp} \rightarrow * \mid / \mid \%$
 $\textit{Factor} \rightarrow [\textit{UnaryOp}] \textit{Primary}$
 $\textit{UnaryOp} \rightarrow - \mid !$
 $\textit{Primary} \rightarrow \textit{Identifier} [[\textit{Expression}]] \mid \textit{Literal} \mid$
 $(\textit{Expression}) \mid \textit{Type} (\textit{Expression})$

$\textit{Identifier} \rightarrow \textit{Letter} \{ \textit{Letter} \mid \textit{Digit} \}$
 $\textit{Letter} \rightarrow \text{a} \mid \text{b} \mid \dots \mid \text{z} \mid \text{A} \mid \text{B} \mid \dots \mid \text{Z}$
 $\textit{Digit} \rightarrow 0 \mid \dots \mid 9$
 $\textit{Literal} \rightarrow \textit{Integer} \mid \textit{Boolean} \mid \textit{Float} \mid \textit{Char}$
 $\textit{Integer} \rightarrow \textit{Digit} \{ \textit{Digit} \}$
 $\textit{Boolean} \rightarrow \text{true} \mid \text{false}$
 $\textit{Float} \rightarrow \textit{Integer} . \textit{Integer}$
 $\textit{Char} \rightarrow ' \textit{ASCIIChar} '$

- Read Clite syntax, and answer the following questions.
 - What are the T, N, P, S in Clite?
 - T: $A \sim Z$, $a \sim z$, $0 \sim 9$, ASCIIChar, true, false, int, bool, float, char, main, if, while, operators
 - N: the symbols on the left-hand-side of the productions
 - P: all the rules
 - S: Program
 - What is the precedence of operators?
 - UnaryOp > MulOp > AddOp > RelOp > EquOp > && > ||
- Now, let draw a parse tree for $a = b + 3 \times c$; using Clite syntax.

Concrete grammar for *Assignment* and *Expression* (Tucker and Noonan, 2007).

$Assignment \rightarrow Identifier \ [\ [Expression] \] = Expression ;$

$Expression \rightarrow Conjunction \ \{ \ || Conjunction \}$

$Conjunction \rightarrow Equality \ \{ \ \&\& Equality \}$

$Equality \rightarrow Relation \ [\ EquOp Relation \]$

$EquOp \rightarrow == \ | \ !=$

$Relation \rightarrow Addition \ [\ RelOp Addition \]$

$RelOp \rightarrow < \ | \ <= \ | \ > \ | \ >=$

$Addition \rightarrow Term \ \{ \ AddOp Term \}$

$AddOp \rightarrow + \ | \ -$

$Term \rightarrow Factor \ \{ \ MulOp Factor \}$

$MulOp \rightarrow * \ | \ / \ | \ \%$

$Factor \rightarrow [\ UnaryOp \] \ Primary$

$UnaryOp \rightarrow - \ | \ !$

$Primary \rightarrow Identifier \ [\ [Expression] \] \ | \ Literal \ |$

$(Expression) \ | \ Type (Expression)$

