Abstract Syntax Tree

What is it?
Why use it instead of CST?
CST $\rightarrow$ AST

What is AST?

pruned CST

From Wikipedia:
"A finite, labeled, directed tree, where the internal
nodes are labeled by operators, and the leaf
nodes represent the operands of the operators."

A CST absent of Token and Production nodes that
conveyed structure during the parsing phase.
This information is now available in the tree.

CST vs. AST

$S \rightarrow (E)^* E$
$E \rightarrow E + E \mid E * E \mid id$

CST vs. AST

$S \rightarrow (E)^* E$
$E \rightarrow E + E \mid E * E \mid id$
Why use it?

- clean
- closer to BNF grammar
- more convenient than CST

I want more curly braces in my sableCC grammar file.

CST → AST

CST

Productions

cst_full_name = [last].name comma [first].name;

AST

Productions

cst_full_name (-> full_name) = [last].name comma [first].name;

Abstract Syntax Tree

full_name = [first].name [last].name;

CST

AST

FN → LN . FN
CST → AST

\[ E \rightarrow E + E | \text{id} \]

**Productions**
- \( \text{cst} \_\text{exp} \rightarrow \exp \) = (plus_rule) \( \text{cst} \_\text{exp} \) plus \( \text{cst} \_\text{term} \)
- \( \text{cst} \_\text{term} \) -> New exp.plus(\( \text{cst} \_\text{exp} \).exp, \( \text{cst} \_\text{term} \).exp)
- \( \text{term} \_\text{rule} \)
- \( \text{cst} \_\text{term} \rightarrow \exp \) = id
- \( \text{cst} \_\text{term} \) -> New exp.id(id);

**Abstract Syntax Tree**
- \( \text{exp} \rightarrow \text{plus} \) [left]:exp [right]:exp
- \( \text{id} \) \( \rightarrow \) id;

CST

\[ \text{OL} \rightarrow \text{O}^* \]
\[ \text{O} \rightarrow \text{name} : \text{attribute} \]

**Productions**
- \( \text{cst} \_\text{objects} \rightarrow \text{objects} \) = cst_objects
- \( \text{cst} \_\text{object} \rightarrow \text{object} \) = New objects(\( \text{cst} \_\text{object} \).object)
- \( \text{cst} \_\text{object} \rightarrow \text{object} \) = [name]:alpha_str colon [attribute]:alpha_str
- \( \text{cst} \_\text{object} \rightarrow \text{object} \) = New object(name, attribute);

**Abstract Syntax Tree**
- \( \text{objects} \rightarrow \text{object}^* \)
- \( \text{object} \rightarrow \text{first}:alpha \_str \ [last]:alpha \_str \)
**Productions (without transformations, your turn)**

- cst_names = cst_name_list;
  
  - cst_name_list = cst_name_list_item* cst_name_last_item;
  
  - cst_name_list_item = cst_full_name colon;
  
  - cst_name_last_item = cst_full_name;
    
  - cst_full_name = [last]:name comma [first]:name;

**Abstract Syntax Tree**

- names = full_name*;
  
  - full_name = [first]:name [last]:name;
\[ \text{NL} \rightarrow ((\text{FN})^* \text{FN})? \]
\[ \text{FN} \rightarrow \text{LN}, \text{FN} \]

Productions
- \( \text{cst_name} \rightarrow \text{name} \)
  \( \rightarrow \) New names([cst_name_list.full_name])
- \( \text{cst_name_list} \rightarrow \text{full_name} \)
  \( \rightarrow \) cst_name_list_item
  \( \rightarrow \) [cst_name_list_item.full_name]
- \( \text{cst_name_list_item} \rightarrow \text{full_name} \)
  \( \rightarrow \) cst_full_name colon
  \( \rightarrow \) cst_full_name full_name
- \( \text{cst_full_name} \rightarrow \text{full_name} \)
  \( \rightarrow \) New full_name(first, last)

Abstract Syntax Tree
- \( \text{names} = \text{full_name}*; \)
  \( \rightarrow \) [first]:name [last]:name

\[ S \rightarrow E (, E)^* \]
\[ E \rightarrow E + E | E * E | \text{id} \]

Productions (without transformations, your turn)
- cst_stm = cst_exp_list
  \( \rightarrow \) [plus_rule] cst_exp plus cst_term
  \| (term_rule) cst_term;
- cst_term = [multi_rule] cst_term mult cst_factor
  \| [fact_rule] cst_factor;
- cst_factor = (id_rule) id;
- cst_exp_list = cst_exp cst_exp_rest*
  \( \rightarrow \) comma cst_exp;
- cst_exp_rest = comma cst_exp;

Abstract Syntax Tree
- \( \text{stm} = \text{exp}; \)
  \( \rightarrow \) [plus] [left]:exp [right]:exp
  \| [multi] [left]:exp [right]:exp
  \| [id] id
Wednesday

Going over some of the suggested exercises from the textbook.

Email me with exercises that you want covered.