

Abstract Syntax Tree

What is it?
 Why use it instead of CST?
 CST → AST

What is AST?

pruned CST

From Wikipedia:

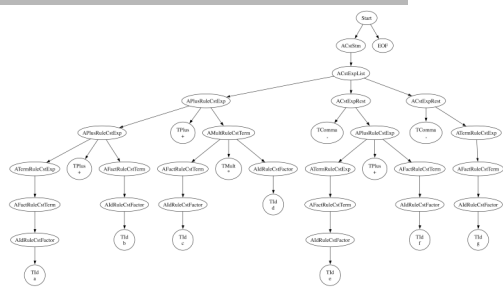
"A finite, labeled, directed tree, where the internal nodes are labeled by operators, and the leaf nodes represent the operands of the operators."

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

$$S \rightarrow (E,)* E$$

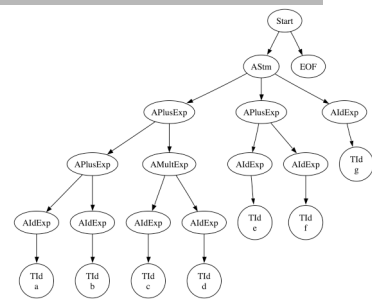
$$E \rightarrow E + E \mid E * E \mid id$$



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.

CS453

5

CST \rightarrow AST

FN \rightarrow LN, FN

CST

```
Productions
cst_full_name = [last]:name comma [first]:name;
```

AST

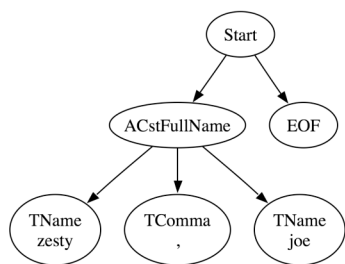
```
Productions
cst_full_name {-> full_name} = [last]:name comma [first]:name
                                   {-> New full_name(first, last)};
```

```
Abstract Syntax Tree
full_name = [first]:name [last]:name;
```

CS453

6

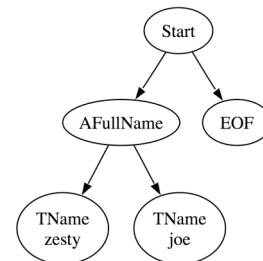
CST



CS453

7

AST



CS453

8

CST → AST

$E \rightarrow E + E \mid id$

```

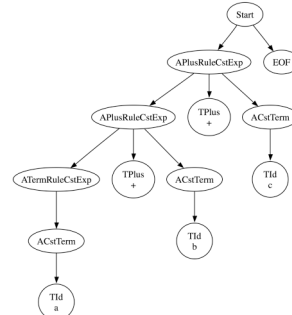
Productions
cst_exp {-> exp} = {plus_rule} cst_exp plus cst_term
              {-> New exp.plus(cst_exp.exp, cst_term.exp)}
              | {term_rule} cst_term
              {-> cst_term.exp};
cst_term {-> exp} = id
              {-> New exp.id(id)};

Abstract Syntax Tree
exp = {plus} [left]:exp [right]:exp
      | {id} id;
    
```

CS453

9

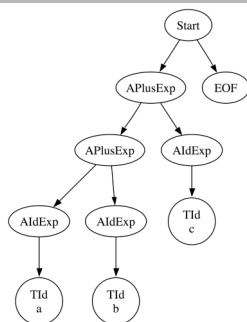
CST



CS453

10

AST



CS453

11

CST → AST

$OL \rightarrow O^*$
 $O \rightarrow name : attribute$

```

Productions
cst_objects {-> objects} = cst_object*
              {-> New objects([cst_object.object])};
cst_object {-> object} = [name]:alpha_str colon [attribute]:alpha_str
              {-> New object(name, attribute)};
    
```

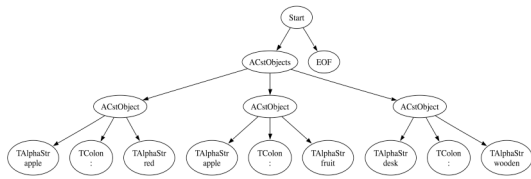
```

Abstract Syntax Tree
objects = object*;
object = [first]:alpha_str [last]:alpha_str;
    
```

CS453

12

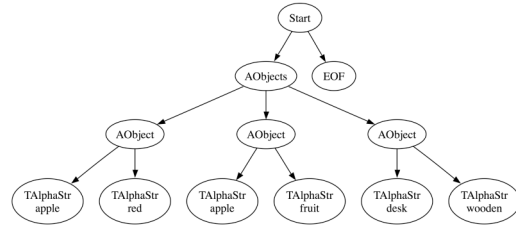
CST



CS453

13

AST



CS453

14

CST → AST

NL → ((FN:)* FN)?
 FN → LN , FN

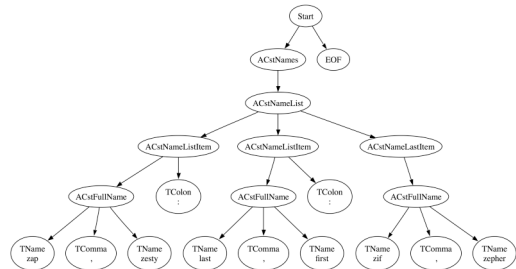
```
Productions (without transformations, your turn)
cst_names = cst_name_list?;
cst_name_list = cst_name_list_item* cst_name_last_item;
cst_name_list_item = cst_full_name colon;
cst_name_last_item = cst_full_name;
cst_full_name = [last]:name comma [first]:name;
```

```
Abstract Syntax Tree
names = full_name*;
full_name = [first]:name [last]:name;
```

CS453

15

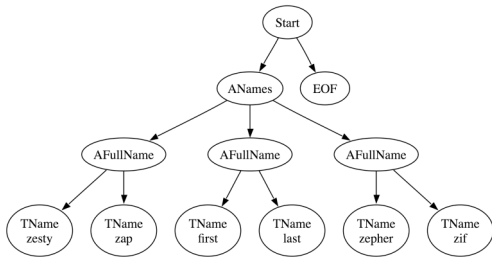
CST



CS453

16

AST



CS453

17

CST → AST

NL → ((FN:)* FN)?
FN → LN, FN

```

Productions
cst_names {-> names} = cst_name_list?
  {-> New names([cst_name_list.full_name]);}
cst_name_list {-> full_name*} = cst_name_list_item*
cst_name_list_item
  {-> [cst_name_list_item.full_name,
cst_name_list_item.full_name];}
cst_name_list_item {-> full_name} = cst_full_name colon
  {-> cst_full_name.full_name};
cst_name_list_item {-> full_name} = cst_full_name
  {-> cst_full_name.full_name};
cst_full_name {-> full_name} = [last]:name comma [first]:name
  {-> New full_name(first, last)};
  
```

```

Abstract Syntax Tree
names = full_name*;
full_name = [first]:name [last]:name;
  
```

CS453

18

CST → AST

S → E (, E)*
E → E + E | E * E | id

```

Productions (without transformations, your turn)
cst_stm = cst_exp_list;
cst_exp = (plus_rule) cst_exp plus cst_term
  | (term_rule) cst_term;
cst_term = (mult_rule) cst_term mult cst_factor
  | (fact_rule) cst_factor;
cst_factor = (id_rule) id;
cst_exp_list = cst_exp cst_exp_rest*;
cst_exp_rest = comma cst_exp;
  
```

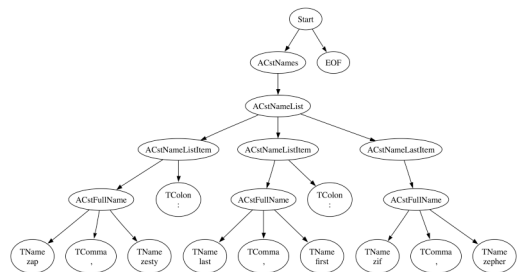
```

Abstract Syntax Tree
stm = exp+;
exp = {plus} [left]:exp [right]:exp
  | {mult} [left]:exp [right]:exp
  | {id} id;
  
```

CS453

19

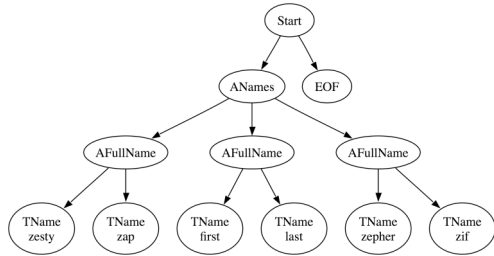
CST



CS453

20

AST



CS453

21

CST \rightarrow AST

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

```
Productions
cst_stm (-> stm) = cst_exp_list
  (-> New stm([cst_exp_list.exp]));
cst_exp (-> exp) = (plus_rule) cst_exp plus cst_term
  (-> New exp.plus(cst_exp.exp, cst_term.exp))
  | (term_rule) cst_term
  (-> cst_term.exp);
cst_term (-> exp) = (mult_rule) cst_term mult cst_factor
  (-> New exp.mult(cst_term.exp, cst_factor.exp))
  | (fact_rule) cst_factor
  (-> cst_factor.exp);
cst_factor (-> exp) = (id_rule) id
  (-> New exp.id(id));
cst_exp_list (-> exp+) = cst_exp cst_exp_rest*
cst_exp_rest (-> exp) = comma cst_exp
  (-> cst_exp.exp);
```

```
Abstract Syntax Tree
stm = exp+;
exp = (plus) [left]:exp [right]:exp
  | (mult) [left]:exp [right]:exp
  | (id) id;
```

CS453

22

Wednesday

Going over some of the suggested exercises from the textbook.

Email me with exercises that you want covered.

CS453

23