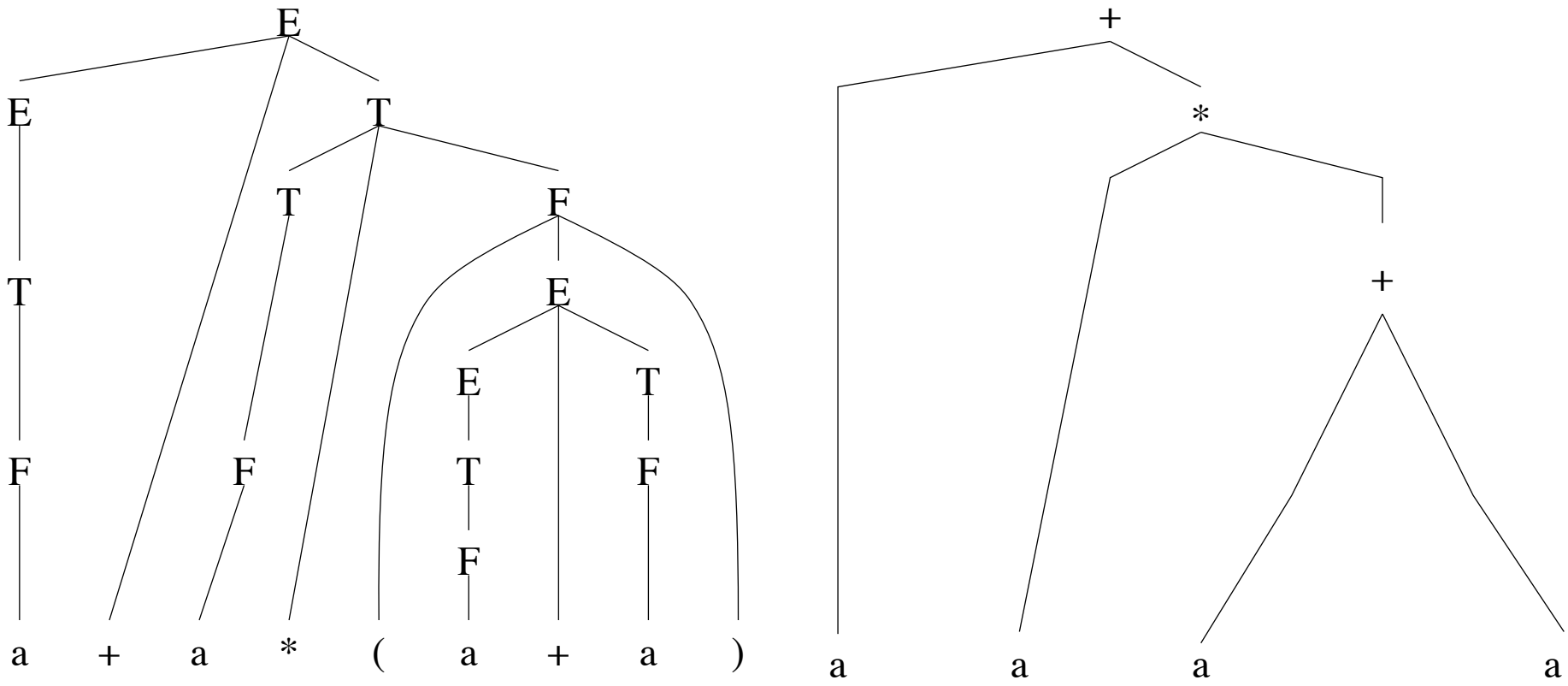
Abstract syntax trees (ASTs)



The AST eliminates the scaffolding introduced to render the grammar unambiguous. Items such as temporary variables can be introduced into the AST to simplify subsequent activity (optimization, code generation).

Creating an AST

We can easily add actions to the grammar to create AST nodes and properly link these nodes to form the AST.

```
S → E $
E → E + T
      {$$ = MakeBinTree(PLUS,$1,$3);}
    | T
      {$$ = $1;}
T → T * F
      {$$ = MakeBinTree(TIMES,$1,$3);}
    | F
      {$$ = $1;}
F → ( E )
      {$$ = $2;}
    | const
      {$$ = MakeConst($1);}
    | id
      {$$ = MakeSymb($1);}
```

Free of clutter, the resulting tree can then be traversed to instantiate symbol tables, perform type checking, optimize the program, and generate code.

AST routines

```
typedef struct _TreeNode {
   struct {
     int linenumber;
     int colnumber;
   } sourceinfo;
   NodeInfo info;
   struct _TreeNode *child;
   struct _TreeNode *sibling;
   struct _TreeNode *head;
   struct _TreeNode *parent;
   struct _TreeNode *leftsib;
} TreeNode;
```

NodeInfo is a *union* of tree node information: symbol table pointers, integer values, operator types, etc.

$MakeFamily(parent, sibs)$:
adopts $sibs$ into the $parent$'s family, returning the parent.

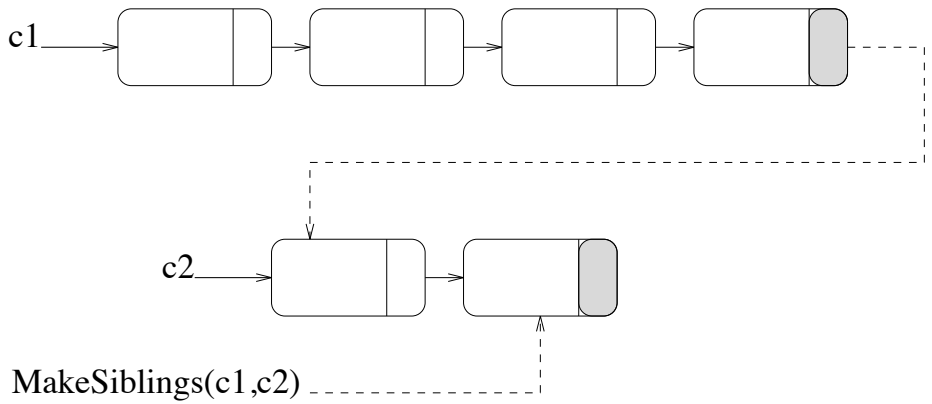
$MakeSiblings(c1, c2)$: **units siblings $c1$ and $c2$, returning the end of the resulting list (shown below).**

$MakeOperatorNode(opnum)$: **creates an operator node, where $opnum$ is the "name" of a "token".**

$MakeIntegerNode(intval)$: **creates an integer node with value $intval$.**

$MakeStringNode(str)$: **creates a string node with value str.**

$MakeSymbolNode(sym)$: **creates a symbol reference node to sym.**



Using the AST routines

Num $\rightarrow$ **D $**

```
{$$ = $1 → head;}
```

D $\rightarrow$ **D d**

```
{$$ = MakeSiblings($1,
       MakeIntegerNode($2));}
```

| **B**

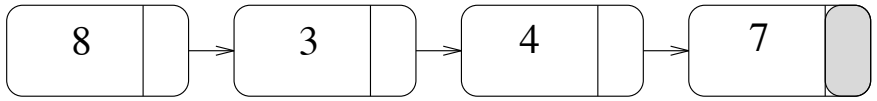
```
{$$ = MakeIntegerNode($1);}
```

B $\rightarrow$ **x d**

```
{$$ = $2;}
```

| λ

```
{$$ = 10;}
```



The above list is created by the actions shown to the left. The first number in the list is the base, and the subsequent numbers are the digits as parsed from left to right.

Example AST

```
int a1;
extern int a2;

int factorial(X)
int X;
{
  int Y;

  Y = X;
      if (Y > 0) [ Y*factorial(X-1); ];
               else [1;];
}

void main() {
   int i;
   a1 = factorial(i=5);
   a2 = factorial(3);
}
```

The AST is shown to the right, with indentation reflecting tree depth.
Note the regular structure:

- functions and inline procedures are represented similarly.
- an if-then structure is represented as an if-then-else with trivial "else" code.

```
Operator PROGRAM
  Operator FORMALS
  Operator SDCLS
    Ref Symbol001(02)  int   *0    [0] auto         : a1
    Ref Symbol002(02)  int   *0    [0] extern       : a2
  Operator FDCLS
    Ref Symbol003(02)  int   *0 () [0] auto         : factorial
      Operator FORMALS
        Ref Symbol004(03)  int   *0    [0] auto         : X
      Operator SDCLS
        Ref Symbol005(04)  int   *0    [0] auto         : Y
      Operator FDCLS
      Operator EXPRBLOCK
        Operator OTHEREXPRS
          Operator ASSIGN
            Ref Symbol005(04)  int   *0    [0] auto         : Y
            Ref Symbol004(03)  int   *0    [0] auto         : X
        Operator LASTEXPR
          Operator IF
            Operator GT_OP
              Ref Symbol005(04)  int   *0    [0] auto         : Y
              Integer 0
            Operator INLINEPROC
              Operator FORMALS
              Operator SDCLS
              Operator FDCLS
              Operator EXPRBLOCK
                Operator OTHEREXPRS
                Operator LASTEXPR
                  Operator TIMES
                    Ref Symbol005(04)  int   *0    [0] auto         : Y
                    Operator INVOKE
                      Ref Symbol003(02)  int   *0 () [0] auto         : factorial
                      Operator ARGS
                        Operator MINUS
                          Ref Symbol004(03)  int   *0    [0] auto         : X
                          Integer 1
            Operator INLINEPROC
              Operator FORMALS
              Operator SDCLS
              Operator FDCLS
              Operator EXPRBLOCK
                Operator OTHEREXPRS
                Operator LASTEXPR
                  Integer 1
  Ref Symbol006(02)  void  *0 () [0] auto         : main
    Operator FORMALS
    Operator SDCLS
      Ref Symbol007(04)  int   *0    [0] auto         : i
    Operator FDCLS
    Operator EXPRBLOCK
      Operator OTHEREXPRS
        Operator ASSIGN
          Ref Symbol001(02)  int   *0    [0] auto         : a1
          Operator INVOKE
            Ref Symbol003(02)  int   *0 () [0] auto         : factorial
            Operator ARGS
              Operator ASSIGN
                Ref Symbol007(04)  int   *0    [0] auto         : i
                Integer 5
      Operator LASTEXPR
        Operator ASSIGN
          Ref Symbol002(02)  int   *0    [0] extern       : a2
          Operator INVOKE
            Ref Symbol003(02)  int   *0 () [0] auto         : factorial
            Operator ARGS
              Integer 3
  Operator EXPRBLOCK
    Operator OTHEREXPRS
    Operator LASTEXPR
```