In a more complicated language, we might make multiple passes over the tree:

- **Extended Calculator Grammar**
  - See the text for more info.

Semantic checks and (later) generate middle routines to build the AST.

No comparable notion of parsing: structure of tree is self-consistent with the order in which the parser builds the tree.

Language for a tree grammar is the set of possible appropriate action routines embedded in a CFG.

Here's a syntax tree for a tiny program:

- Each "production" of tree grammar has parent on LHS and children on RHS.
- We could even do it while building an explicit parse tree.
- We could do some of this checking while building the AST.
- Consider an extended version with types and declarations:

  Decorate a Syntax Tree

  It's also straightforward to turn that grammar into a table of productions.