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

So, for example, in

More common to make symbol table and error lists global variables

each production w/out actually embedding them in the RHS.

making each action routine a semantic rule and then listing the rules for

consistent with the order in which the parser builds the tree

The book goes into a bit of AG theory, talking about

language

semantic checks and (later) generate mid

Our tree grammar helps guide us as we write (by hand) the action

Language for a tree grammar is the set of possible

appropriate action routines embedded in a CFG.

Structure is given by the tree grammar. Construction would be via execution of

(given production.

This will help motivate the code to build an AST.

We could do some of this checking while building the AST.

As noted in Chap. 1, job of semantic analyzer is to

compliance at compile time. This makes programs run faster.

A smart compiler may avoid run

Semantic

- require type match on arithmetic ops

Now we can

Note that "program → item" does

to see how we

require type match on arithmetic ops

At the other extreme, perform all static dynamic checks and generate

something else that has semantics (meaning)

As noted in Chap. 1, job of semantic analyzer is to

- require type match on arithmetic ops

This is

We could even do it while building an explicit parse tree.

The calculator language we've been using for examples doesn't have

So how do we build a syntax tree instead?

5:  F  →  ( E )

3:  T  →  F  FT

This will help motivate the code to build an AST.

to see how we

require type match on arithmetic ops

At the other extreme, perform all static dynamic checks and generate

something else that has semantics (meaning)

As noted in Chap. 1, job of semantic analyzer is to

- require type match on arithmetic ops

This is

We could even do it while building an explicit parse tree.

The calculator language we've been using for examples doesn't have

So how do we build a syntax tree instead?

5:  F  →  ( E )

3:  T  →  F  FT

This will help motivate the code to build an AST.

to see how we

require type match on arithmetic ops

At the other extreme, perform all static dynamic checks and generate

something else that has semantics (meaning)

As noted in Chap. 1, job of semantic analyzer is to

- require type match on arithmetic ops

This is

We could even do it while building an explicit parse tree.

The calculator language we've been using for examples doesn't have

So how do we build a syntax tree instead?

5:  F  →  ( E )

3:  T  →  F  FT

This will help motivate the code to build an AST.

to see how we

require type match on arithmetic ops

At the other extreme, perform all static dynamic checks and generate

something else that has semantics (meaning)

As noted in Chap. 1, job of semantic analyzer is to

- require type match on arithmetic ops

This is

We could even do it while building an explicit parse tree.

The calculator language we've been using for examples doesn't have

So how do we build a syntax tree instead?

5:  F  →  ( E )

3:  T  →  F  FT

This will help motivate the code to build an AST.

To see how we