Parse Trees

Beefed-up calculator language

run $ ightarrow$ stmt run \mid stmt
$stmt ightarrow ares ext{STOP}$
ares $ ightarrow$ VAR ASN bres \mid bres
bres $ ightarrow$ bres <code>BOP</code> res \mid res
$\mathit{res} ightarrow \mathit{res}$ <code>COMP</code> $\mathit{exp} \mid \mathit{exp}$
exp $ ightarrow$ exp <code>OPA</code> term \mid term
term $ ightarrow$ term <code>OPM</code> factor \mid factor
$\mathit{factor} ightarrow \texttt{NUM} \mid \texttt{VAR} \mid \texttt{LP} \textit{ bres } \texttt{RP}$

Download today's tarball and run make to get a parse tree for some string in this language.

We notice that the parse tree is large and unwieldy with many unnecessary nodes. $% \left({{{\boldsymbol{x}}_{i}}} \right) = {{\left({{{\boldsymbol{x}}_{i}}} \right)}} \right)$

SI 413 (USNA) Unit 5 Fall 2021 1/6

Abstract Syntax Tree

Consider the program x := (5 + 3) * 2; x - 7;. What should the AST for this look like?

SI 413 (USNA)

SI 413 (USNA)

Unit 5

Fall 2021 2 / 6

Fall 2021 3 / 6

AST Properties
Remember, <i>ASTs are not about the syntax!</i> They <i>remove</i> syntactic details from the program, leaving only the semantics.
Typically, we show ordering (e.g. of <i>ares</i> 's in the previous example) by nesting: the last child of a statement is the next statement, or null.
Are ASTs language independent?

Unit 5

Static type checking

SI 413 (USNA)

Consider the string (7 > 2) + 3; This is an error. But where should this error be identified?

Each node in the AST has a type, possibly "void".

Unit 5

Static type checking with variables

What about the string x = 6 > 3; x * 12;?

We have to know the type of the variable x. Otherwise, there is no way to detect this error at compile-time.

Only *statically-typed languages* allow this sort of checking. Remember, in this class *errors are a good thing!*

SI 413 (USNA)

Unit 5

Fall 2021 5 / 6

Fall 2021 4 / 6

Unit outcomes

You should know:

- What an AST is, and why we need them.
- The relationship between language, parse tree, and AST.
- How static type-checking works, at a basic level.

You should be able to:

- Draw a parse tree for a given string, given the grammar.
- Determine the AST from the parse tree. Note that there is some flexibility here!