Static semantics

- compile-time analysis
- syntax-directed
Nonnegative integers

\[ \text{<number>} \ ::= \text{<digit>} \mid \text{<number>} \ \text{<digit>} \]
\[ \text{<digit>} \ ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]
Expressions

<S> ::= <E>
<E> ::= <E> + <T>
<E> ::= <E> - <T>
<E> ::= <T>
<T> ::= <number>

What does 5 - 3 - 1 “mean”?
\( E - T \)

\( E - T \)

\( T \) \( number \) \( digit \)

\( number \) \( digit \) 1

\( digit \) 3

5
JK;LNMPOQO
Attribute grammars

Attributes: quantities (values) associated with a construct.

\[ X.a \] — \( X \) is a terminal or a nonterminal

\[ a \] an attribute of \( X \)

Attributes for terminal symbols come with the symbol

Attributes for nonterminal symbols are defined by semantic rules attached to productions in a grammar
A simple example

\[
<\text{number}> ::= <\text{digit}>
\]

\[
<\text{number}>.val = <\text{digit}>.val
\]

\[
<\text{number}> ::= <\text{number2}> <\text{digit}>
\]

\[
<\text{number}>.val = <\text{digit}>.val + 10 \times (<\text{number2}>.val)
\]

\[
<\text{digit}> ::= 0
\]

\[
<\text{digit}>.val = 0
\]
Information flows bottom-up in the parse tree

Synthesized attributes
Type declarations

\[ \texttt{<vardec>} \quad ::= \texttt{int <intvlist>} \]
\[ \texttt{<intvlist>} \quad ::= \texttt{<name>} \mid \texttt{<name>, <intvlist>} \]

\[
\begin{aligned}
\texttt{<vardec>} & \\
\texttt{int} & \quad \texttt{<intvlist>} \\
\texttt{<name>} & \quad , \quad \texttt{<intvlist>} \\
\texttt{<name>} & \quad , \quad \texttt{<intvlist>} \\
\texttt{x} & \quad \texttt{y} & \quad \texttt{z}
\end{aligned}
\]
\textbf{Inherited} attributes
Semantic rules

<vardec> ::= int <intvlist>

<intvlist>.type = int.type = int

<intvlist> ::= <name>

<name>.type = <intvlist>.type

<intvlist> ::= <name>, <intvlist2>

<name>.type = <intvlist>.type

<intvlist2>.type = <intvlist>.type
Attribute grammars

Every nonterminal \( A \) has two sets of attributes:

synthesized attributes \( S(A) \)

inherited attributes \( I(A) \)

\[
\text{Att}(A) = S(A) \cup I(A)
\]

Every production in the grammar has associated semantic rules.

A semantic rule is of the form

an attribute = an expression in terms of attributes
L-attributed grammars

- Arbitrary semantic rules may cause circular dependencies

- introduced by Lewis, Rosenkrantz & Stearns (1973)

- attributes can be evaluated in depth-first order
  - suitable for syntax-directed translation
L-attributed grammars

\[ A ::= A_1 \ldots A_j \ldots A_m \]

The semantic rules are of the form

\[ lhs = expression \]

where

(a) \( lhs \) is in \( S(A) \) or one of \( I(A_j) \)

(b) if the \( lhs \) is in \( S(A) \), then the expression is in terms of

- inherited attributes of \( A \), namely \( I(A) \),
  and
- all attributes of the symbols on the right-hand side of the rule

if the \( lhs \) is in \( I(A_j) \), then the expression is in terms of

- inherited attributes of \( A \), namely \( I(A) \),
  and
- all attributes of \( A_k \), for all \( k < j \)
\[ \text{invoke } A_1 (\ldots ) \]
\[ \vdots \]
\[ \text{invoke } A_i (\ldots ) \]
\[ \vdots \]
\[ \text{invoke } A_m (\ldots ) \]

.. returns synthesized attributes ..
$I_1$ depends only on $I$

$I_i$ depends on all attributes to the left of node $A_i$
Normalized L-attributed grammars

\[ A ::= A_1 \ldots A_j \ldots A_m \]

The semantic rules are of the form

\[ lhs = expression \]

where

(a) \( lhs \) is in \( S(A) \) or one of \( I(A_j) \)

(b) if the \( lhs \) is in \( S(A) \), then the expression is in terms of
   • inherited attributes of \( A \), namely \( I(A) \), and
   • \textit{synthesized} attributes of the symbols on the right-hand side of the rule

if the \( lhs \) is in \( I(A_j) \), then the expression is in terms of
   • inherited attributes of \( A \), namely \( I(A) \), and
   • \textit{synthesized} attributes of \( A_k \), for all \( k < j \)
Decorated parse tree

A parse tree with all attributes evaluated

- Attribute values of different *instances* of the same nonterminal may be different
Example: Conditional expressions

\[ \langle E \rangle ::= \text{if } \langle C \rangle \text{ then } \langle E \rangle \text{ else } \langle E \rangle \]

\[ \langle E \rangle.val = \text{if } \langle C \rangle.val \text{ then } \langle E1 \rangle.val \]
\[ \text{else } \langle E2 \rangle.val \]
Conditional Attribute Grammars

There is a boolean attribute to represent validity

Can generate non-context-free languages

\[
\begin{align*}
\langle S \rangle & ::= \langle A \rangle \langle B \rangle \langle C \rangle \\
\langle S \rangle.\text{wf} &= (\langle A \rangle.\text{val} == \langle B \rangle.\text{val} == \langle C \rangle.\text{val}) \\
\langle A \rangle & ::= \text{a}\langle A \rangle \\
\langle A \rangle.\text{val} &= \langle A1 \rangle.\text{val} + 1 \\
\langle A \rangle & ::= \text{empty} \\
\langle A \rangle.\text{val} &= 0
\end{align*}
\]

Similarly for \langle B \rangle and \langle C \rangle
<S> ::= A>B>C

<S>.wf = (<A>.val == <B>.val == <C>.val)

<A> ::= a<A>

<A>.val = <A1>.val + 1

<A> ::= <empty>

<A>.val = 0

<B> ::= b<B>

<B>.val = <B1>.val + 1

<B> ::= <empty>

<B>.val = 0

<C> ::= c<C>

<C>.val = <C1>.val + 1

<C> ::= <empty>

<C>.val = 0
Sample problem 4

\[
\begin{align*}
\langle S \rangle & \::= \langle A \rangle \langle B \rangle \\
\langle S \rangle . val & = \langle B \rangle . s \\
\langle B \rangle . i & = \langle A \rangle . val \\
\langle A \rangle & \::= \mathtt{a} \langle A \rangle \\
\langle A \rangle . val & = \langle A2 \rangle . val + 1 \\
\langle A \rangle & \::= \mathtt{a} \\
\langle A \rangle . val & = 1 \\
\langle B \rangle & \::= \mathtt{b} \langle B \rangle \\
\langle B \rangle . s & = \langle B2 \rangle . s + \langle B \rangle . i \\
\langle B2 \rangle . i & = \langle B \rangle . i \\
\langle B \rangle & \::= \mathtt{b} \\
\langle B \rangle . s & = \langle B \rangle . i
\end{align*}
\]