Static semantics

- compile-time analysis
- syntax-directed
Nonnegative integers

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

\(<S> ::= <E>\)
\(<E> ::= <E> + <T>\)
\(<E> ::= <E> - <T>\)
\(<E> ::= <T>\)
\(<T> ::= <number>\)

What does 5 - 3 - 1 “mean”?
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

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

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

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

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

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

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

Synthesized attributes
Type declarations

<vardec> ::= int <intvlist>
<intvlist> ::= <name> | <name>, <intvlist>

\[
\begin{align*}
\text{<vardec>} & \\
\text{int} & \quad \text{<intvlist>}
\end{align*}
\]

\[
\begin{align*}
\text{<name>} & , \quad \text{<intvlist>}
\end{align*}
\]

\[
\begin{align*}
\text{<name>} & , \quad \text{<intvlist>}
\end{align*}
\]

\[
\begin{align*}
\text{<name>} & \\
x & \\
y & \\
z
\end{align*}
\]
Inherited attributes
Semantic rules

\[ \texttt{<vardec> ::= int <intvlist>} \]

\[ \texttt{<intvlist>.type = int.type = int} \]

\[ \texttt{<intvlist> ::= <name>} \]

\[ \texttt{<name>.type = <intvlist>.type} \]

\[ \texttt{<intvlist> ::= <name>, <intvlist2>} \]

\[ \texttt{<name>.type = <intvlist>.type} \]

\[ \texttt{<intvlist2>.type = <intvlist>.type} \]
Attribute grammars

Every nonterminal $A$ has two sets of attributes:

synthesized attributes $S(A)$

inherited attributes $I(A)$

$$Att(A) = S(A) \cup I(A)$$

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 \( k < j \)
invoke \( A_1(\ldots) \)

\[ \cdot \]

invoke \( A_i(\ldots) \)

\[ \cdot \]

invoke \( A_m(\ldots) \)

\[ \cdot \]

.. 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

\[ <E> ::= \text{if } <C> \text{ then } <E> \text{ else } <E> \]

\[ <E>.val = \left\{ \begin{array}{ll}
\text{if } <C>.val \text{ then } <E1>.val \\
\text{else } <E2>.val
\end{array} \right. \]
Conditional Attribute Grammars

There is a boolean attribute to represent validity

Can generate non-context-free languages

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

\[ <S> ::= <A><B> \]

\[ <S>.val = <B>.s \]

\[ <B>.i = <A>.val \]

\[ <A> ::= a<A> \]

\[ <A>.val = <A2>.val + 1 \]

\[ <A> ::= a \]

\[ <A>.val = 1 \]

\[ <B> ::= b<B> \]

\[ <B>.s = <B2>.s + <B>.i \]

\[ <B2>.i = <B>.i \]

\[ <B> ::= b \]

\[ <B>.s = <B>.i \]