Syntax

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine
Syntax

- Symbols for building words,
- Structure of words,
- Structure of well-formed phrases,
- Structure of sentences.

*Only syntactically correct programs also have a semantics.*
Examples

Arithmetic

• Symbols: 0–9, +, −, *, /, (, )
• Words: numerals.
• Phrases: arithmetic expressions.
• Sentences: phrases.

Pascal-like programming language

• Symbols: letters, digits, operators, ...
• Words: keywords, idents, numerals, ...
• Phrases: expressions, statements, ...
• Sentences: programs.

*Languages have internal structure.*
Backus-Naur Form (BNF)

Specification of formal languages.

- Set of equations.
- Left-hand-side: *non-terminal*
  
  Name of a structural type.
- Right-hand-side: list of forms
  
  (Terminal) symbols and non-terminals.

\[
\langle \text{non-terminal} \rangle ::= \\
form_1 \mid \form_2 \mid \ldots \mid \form_n
\]
Example

\[ \langle \text{digit} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

\[ \langle \text{operator} \rangle ::= + \mid - \mid * \mid / \]

\[ \langle \text{numeral} \rangle ::= \langle \text{digit} \rangle \mid \langle \text{digit} \rangle \langle \text{numeral} \rangle \]

\[ \langle \text{expression} \rangle ::= \]
\[ \langle \text{numeral} \rangle \mid (\langle \text{expression} \rangle) \mid \]
\[ \langle \text{expression} \rangle \langle \text{operator} \rangle \langle \text{expression} \rangle \]

*Structure of an expression is illustrated by its derivation tree.*
Ambiguous Syntax Definitions

Expression $4 \times 2 + 1$ has two derivation trees!

Unambiguous definition

\[
\begin{align*}
\langle \text{expression} \rangle & ::= \\
& \langle \text{expression} \rangle \langle \text{lowop} \rangle \langle \text{term} \rangle \mid \\
& \langle \text{term} \rangle \\
\langle \text{term} \rangle & ::= \langle \text{term} \rangle \langle \text{highop} \rangle \langle \text{factor} \rangle \mid \langle \text{factor} \rangle \\
\langle \text{factor} \rangle & ::= \langle \text{numeral} \rangle \mid (\langle \text{expression} \rangle ) \\
\langle \text{lowop} \rangle & ::= + \mid - \\
\langle \text{highop} \rangle & ::= * \mid / 
\end{align*}
\]

Extra level of structure makes derivation unique but syntax complicated.
Semantics

We do not need to use artificially complex BNF definitions!

Why?

*Derivation trees are the real sentences of the language!*

(Strings of symbols are just abbreviations of trees; these abbreviations may be ambiguous).
Two BNF Definitions

- **Concrete syntax**
  
  Determine derivation tree from string abbreviation (parsing).

- **Abstract syntax**
  
  Analyze structure of tree and determine its semantics.

*Tree generated by concrete definition identifies a derivation tree for the string in the abstract definition.*
Abstract Syntax Definitions

- Descriptions of structure.
- Terminal symbols disappear.
- Building blocks are words.

Abstract syntax is studied at the word level.

\[
\begin{align*}
\langle \text{expression} \rangle & ::= \\
& \langle \text{numeral} \rangle \mid \\
& \langle \text{expression} \rangle \langle \text{operator} \rangle \langle \text{expression} \rangle \mid \\
& \text{left-paren} \ \langle \text{expression} \rangle \ \text{right-paren}
\end{align*}
\]

\[
\begin{align*}
\langle \text{operator} \rangle & ::= \text{plus} \mid \text{minus} \mid \text{mult} \mid \text{div} \\
\langle \text{numeral} \rangle & ::= \text{zero} \mid \text{one} \mid \ldots \mid \text{ninety} \mid \ldots
\end{align*}
\]

Structure remains, text vanishes.
Set Theory

More abstract view of abstract syntax.

- Non-terminal names set of phrases specified by corresponding BNF rule.
  
  Expression, Op, Numeral

- Rules replaced by syntax builder operations, one for each form of the rule.

  numeral-exp: Numeral → Expression
  compound-exp: Expression × Op × Expression → Expression
  bracket-exp: Expression → Expression

- Terminal words replaced by constants

  plus: Op
  zero: Numeral

  ...

World of words and derivation trees replaced by world of sets and operations.
More Readable Version

• Syntax domains.
• BNF rules.

Abstract Syntax:

\[ E \in \text{Expression} \]
\[ O \in \text{Operator} \]
\[ N \in \text{Numeral} \]
\[ E ::= N \mid EO E \mid (E) \]
\[ O ::= + \mid - \mid * \mid / \]

\textit{N is just set of values.}
Mathematical Induction

Strategy for proving $P$ on natural numbers.

- Induction basis: Show that $P(0)$ holds.
- Induction hypothesis: assume $P(i)$.
- Induction step: prove $P(i + 1)$

**Proposition** There exist exactly $n!$ permutations of $n$ objects.

**Proof** We use mathematical induction.

- **Basis:** There exists exactly $1 = 0!$ permutation of 0 objects (the “empty” permutation).
- **Hypothesis:** $n!$ permutations of $n$ objects exist.
- **Step:** Add a new object $j$ to $n$ objects. For each permutation $\langle k_{i_1}, k_{i_2}, \ldots, k_{i_n} \rangle$ of the $n$ objects, $n + 1$ permutations result: $\langle j, k_{i_1}, k_{i_2}, \ldots, k_{i_n} \rangle, \langle k_{i_1}, j, k_{i_2}, \ldots, k_{i_n} \rangle, \ldots, \langle k_{i_1}, k_{i_2}, \ldots, k_{i_n}, j \rangle$. Since there are $n!$ permutations of $n$ objects, there are $(n + 1) \times n! = (n + 1)!$ permutations of $n + 1$ objects.
**Structural Induction**

Mathematical induction relies on structure of natural numbers:

$$N ::= 0 \mid N + 1$$

- Show that all trees of zero depth has $P$.
- Assume trees of depth $m$ or less have $P$.
- Prove that tree of depth $m + 1$ has $P$.

**Arbitrary syntax domains:**

- $D ::= \text{Option}_1 \mid \text{Option}_2 \mid \ldots \mid \text{Option}_n$
- To prove that all members of $D$ have $P$
  1. Assume occurrences of $D$ in $\text{Option}_i$ have $P$,
  2. Prove that $\text{Option}_i$ has $P$.

(for each $\text{Option}_i$).
Example

\[ E : \text{Expression} \]
\[ E ::= \text{zero} \mid E_1 \ast E_2 \mid (E) \]

Proposition All members of Expression have the same number of left parentheses as the number of right parentheses.

Proof

1. zero: \( \text{left}(E) = 0 = \text{right}(E) \).
2. \( E_1 \ast E_2 \): \( \text{left}(E) = \text{left}(E_1) + \text{left}(E_2) = \text{right}(E_1) + \text{right}(E_2) = \text{right}(E) \).
3. \( (E') \): \( \text{left}(E) = 1 + \text{left}(E') = 1 + \text{right}(E') = \text{right}(E) \).
Simultaneous Induction

\[
S ::= *E*
\]
\[
E ::= +S \mid **
\]

Mutually recursive definition of syntax domains.

**Proposition** All $S$-values have an even number of * occurences.

**Proof** We prove by simultaneous induction on $S$ and $E$ “all $S$-values and all $E$-values have an even number of * occurences”.

1. $*E*$: $\text{number}(S) = 2 + \text{number}(E)$ which is even, since $\text{number}(E)$ is even.
2. $+S$: $\text{number}(E) = \text{number}(S)$ which is even.
3. $**$: $\text{number}(**) = 2$ which is even.