3. Syntax

3. Syntax

What is a Language?

Concrete Syntax

What is a Language?

Language: A system of communication in a structured way

Natural language

- *used for arbitrary communication*
- *complex, nuanced, and imprecise*

English, Chinese, Hindi, Spanish, …

Programming language

- *used to describe computation*
- *programs have **structure** and **meaning***

Elm, Java, C, Python, SQL, XML, …

Object vs. Metalanguage

Important to distinguish two ***kinds of languages***:

- ***Object language***: the language we're defining
- ***Metalanguage***: the language we're using to define the structure and meaning of the object language!

A single language can fill both roles at different times!
Examples: English, Elm

Syntax vs. Semantics & Metalanguages

Two main ***aspects of a language***:

- ***Syntax***: the structure of its programs
- ***Semantics***: the meaning of its programs

Scope of Metalanguages

	Syntax	*Denotational Semantics*	*Operational Semantics*	*Type Systems*
Regular Expressions	●			
Grammars	●			
Elm	●	●		●
Inference Rules	●	●	●	●

3. Syntax

What is a Language?

Concrete Syntax

Grammar Metalanguage

Grammar Concepts

- ***Grammar:*** *Set of* ***productions*** *(or* ***rules****)*
- ***Production:*** *L ::= R where*
  - *L: nonterminal symbol (in a context-free grammar)*
  - *R: sequence of terminal & nonterminal symbols*
- ***Derivation:*** *Sequence of substitutions ("L by R")*
- ***Sentence:*** *Sequence of terminals derivable from nonterminal*
- ***Language:*** *Set of derivable sentences*

Context-Free Grammars

Formal Grammar Definition

*A **grammar** is a four-tuple* (N, Σ, P, S) *where:*

- N *is a set of **nonterminal symbols***
- Σ *is a set of **terminal symbols** with* $N \cap \Sigma = \varnothing$
- $P \subseteq N \times (N \cup \Sigma)^*$ *is a set of **productions***
- $S \in N$ *is the **start symbol**.*

Grammar for Binary Numbers

$(\{dig, bin\}, \{\texttt{0}, \texttt{1}\}, \{(dig, \texttt{0}), (dig, \texttt{1}), (bin, dig), (bin, dig\, bin)\}, bin)$

Backus-Naur Form (BNF)

Grammar for Binary Numbers

$$(\{dig, bin\}, \{\texttt{0}, \texttt{1}\}, \{(dig, \texttt{0}), (dig, \texttt{1}), (bin, dig), (bin, dig\, bin)\}, bin)$$

BNF $\approx$ Only Productions

$$\begin{array}{lcl} dig & ::= & \texttt{0} \\ dig & ::= & \texttt{1} \\ bin & ::= & dig \\ bin & ::= & dig\, bin \end{array}$$

Grouping RHSs

$$\begin{array}{lcl} dig & ::= & \texttt{0} \mid \texttt{1} \\ bin & ::= & dig \mid dig\, bin \end{array}$$

Example Derivations

Binary numbers

$$
\begin{array}{llll}
\mathit{dig} & ::= & \mathtt{0} & (\mathrm{P1}) \\
\mathit{dig} & ::= & \mathtt{1} & (\mathrm{P2}) \\
\mathit{bin} & ::= & \mathit{dig} & (\mathrm{P3}) \\
\mathit{bin} & ::= & \mathit{dig}\ \mathit{bin} & (\mathrm{P4})
\end{array}
$$

$$
\begin{array}{lll}
& \mathit{bin} & \\
\Rightarrow & \mathit{dig}\ \mathit{bin} & (\mathrm{P4}) \\
\Rightarrow & \mathit{dig}\ \mathit{dig}\ \mathit{bin} & (\mathrm{P4}) \\
\Rightarrow & \mathit{dig}\ \mathit{dig}\ \mathit{dig} & (\mathrm{P3}) \\
\Rightarrow & \mathtt{1}\ \mathit{dig}\ \mathit{dig} & (\mathrm{P2}) \\
\Rightarrow & \mathtt{1}\ \mathtt{0}\ \mathit{dig} & (\mathrm{P1}) \\
\Rightarrow & \mathtt{1}\ \mathtt{0}\ \mathtt{1} & (\mathrm{P2})
\end{array}
\qquad
\begin{array}{lll}
& \mathit{bin} & \\
\Rightarrow & \mathit{dig}\ \mathit{bin} & (\mathrm{P4}) \\
\Rightarrow & \mathit{dig}\ \mathit{dig}\ \mathit{bin} & (\mathrm{P4}) \\
\Rightarrow & \mathit{dig}\ \mathit{dig}\ \mathit{dig} & (\mathrm{P3}) \\
\Rightarrow & \mathit{dig}\ \mathit{dig}\ \mathtt{1} & (\mathrm{P2}) \\
\Rightarrow & \mathit{dig}\ \mathtt{0}\ \mathtt{1} & (\mathrm{P1}) \\
\Rightarrow & \mathtt{1}\ \mathtt{0}\ \mathtt{1} & (\mathrm{P2})
\end{array}
$$

Note: One sentence may have different derivations!

Derivation $\approx$ Trace
(only substitutions, no simplifications)

Grammars Define Languages

Language

Grammar $G = (N, \Sigma, P, S)$ *defines the* ***language*** $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$

Example

$L(\{dig, bin\}, \{\mathtt{0}, \mathtt{1}\}, \{(dig, \mathtt{0}), (dig, \mathtt{0}), \ldots\}, bin) = \{b^k \mid b \in \{\mathtt{0}, \mathtt{1}\} \wedge k > 0\}$

Exercise ...

Question 1

Consider the language L defined by the following grammar.

$$
\begin{aligned}
S &::= A\,B \\
A &::= \mathtt{0}A \mid \mathtt{0} \\
B &::= \mathtt{1}B \mid \mathtt{1}
\end{aligned}
$$

Which of the following statements about are true for words/sentences of L? Each sentence contains ...

(a) ... one or more `1`s
(b) ... one or more `0`s
(c) ... at least as many `1`s as `0`s
(d) ... at least as many `0`s as `1`s
(e) ... exactly as many `1`s as `0`s
(f) ... at least two digits
(g) All `0`s precede all `1`s

a, b, f, g

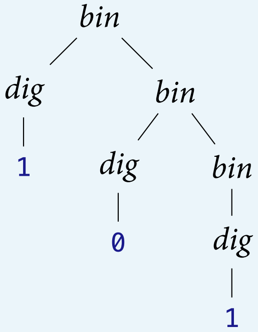
Sentence Structure = Parse Tree

Binary numbers

$$\begin{array}{lcl} \mathit{dig} & ::= & \texttt{0} \mid \texttt{1} \\ \mathit{bin} & ::= & \mathit{dig} \mid \mathit{dig}\ \mathit{bin} \end{array}$$

Rules can be interpreted as instructions to build trees:

“Add R as children to L”



Internal nodes: ***Nonterminals***

Leaves: ***Terminals***

Parse tree ignores the order of rule applications

$\Rightarrow$ *Appropriate representation of sentence structure*

Exercise ...

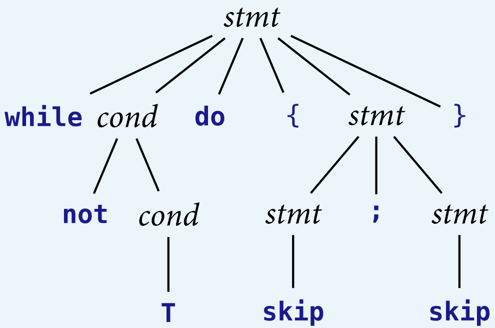
Concrete Syntax

cond ::= **T**
| **not** *cond*
| **(** *cond* **)**

stmt ::= **skip**
| **while** *cond* **do** **{** *stmt* **}**
| *stmt* **;** *stmt*

Question 2

Create a parse tree for the following sentence: **while not T do {skip ; skip}**



Ambiguity

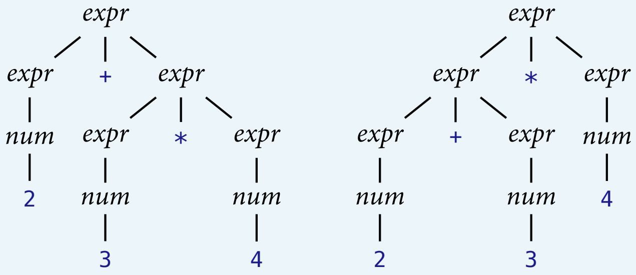
Ambiguous Grammar

Some sentences have more than one parse tree.

Ambigous Grammar

$$
\begin{aligned}
num &::= \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \ldots \\
expr &::= num \\
&\mid\ \ expr + expr \\
&\mid\ \ expr * expr
\end{aligned}
$$

Parse tree for `2 + 3 * 4`



In a Nutshell ...

What is a Grammar?

*A formalism to define linear representations for **typed** tree data structures*

Each nonterminal (≈ type) denotes a set of trees with a specific structure.

Parser: *Algorithm to recover tree structures from strings*

Pretty Printer: *Algorithm to turn tree structures into strings*