Computación y Estructuras Discretas III

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co
Ángela Villota
apvillota@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

2024-1

Languages

To define a languages we should define its lexical, syntactic and semantic components.

- The lexical, what?
- The syntactic, how?
- The semantic, what does it mean?

Agenda del día

1. **The process to design and implement a programming language**
   - EBNF
   - EBNF Rules and Descriptions
   - Language and Syntax
   - EBNF Rules and Descriptions
   - Proving Symbols Match EBNF Rules
   - Syntax versus Semantics
   - Exercises

Syntax and parsing

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form (BNF) or its extended version (EBNF).

Syntax and parsing

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form (BNF) or its extended version (EBNF).

Syntax and parsing

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form (BNF) or its extended version (EBNF).

Languages described in BNF offer significant advantages for the design of syntactic analyzers in compilers.

Agenda del día

1. **The process to design and implement a programming language**
   - EBNF
   - EBNF Rules and Descriptions
   - Language and Syntax
   - EBNF Rules and Descriptions
   - Proving Symbols Match EBNF Rules
   - Syntax versus Semantics
   - Exercises

EBNF Rules and Descriptions

What is an EBNF description?

EBNF Rules and Descriptions

What is an EBNF description?

- Is an unordered list of EBNF rules.
- Each EBNF rule has three parts:
  - A left-hand side (LHS),
  - A right-hand side (RHS),
  - The $\Leftarrow$ character separating these two sides.
- The LHS is one italicized word (possibly with underscores) written in lower-case.
  - It names the EBNF rule.
- The RHS supplies a description of this name.
  - It can include names, characters (standing for themselves), and combinations of the four control forms (sequence, choice, option repetition).

What is EBNF?

What is EBNF?

- **MetaLanguage**
- Notation for formally describing syntax (how to write the linguistic features in a language).
- Also similar is the ability to name descriptions and reuse these names to build more complex structures.

1. **The process to design and implement a programming language**
    - EBNF
    - EBNF Rules and Descriptions
    - Language and Syntax
    - EBNF Rules and Descriptions
    - Proving Symbols Match EBNF Rules
    - Syntax versus Semantics
    - Exercises

Language and Syntax

What were the first high-level programming languages?

Language and Syntax

What were the first high-level programming languages?

- FORTRAN (FORmula TRANslator), developed by the IBM corporation in the United States.
- ALGOL (ALGOrithmic Language), sponsored by a consortium of North American and European countries.

Language and Syntax

What were the first high-level programming languages?

- FORTRAN (FORmula TRANslator), developed by the IBM corporation in the United States.
- ALGOL (ALGOrithmic Language), sponsored by a consortium of North American and European countries.

And who was involved in both initiatives?

What were the first high-level programming languages?

- FORTRAN (FORmula TRANslator), developed by the IBM corporation in the United States.
- ALGOL (ALGOrithmic Language), sponsored by a consortium of North American and European countries.

And who was involved in both initiatives?

- John Backus
- He led the effort to develop FORTRAN.
- Later became a member of the ALGOL design committee, where he studied the problem of describing the syntax of these programming languages simply and precisely.

Language and Syntax

What did Backus do?

Language and Syntax

What did Backus do?

- He invented a notation (based on the work of logician Emil Post) that was simple, precise, and powerful enough to describe the syntax of any programming language.
- Using this notation, a programmer or compiler can determine whether a program is syntactically correct: whether it adheres to the grammar and punctuation rules of the programming language.

Language and Syntax

What did Backus do?

- He invented a notation (based on the work of logician Emil Post) that was simple, precise, and powerful enough to describe the syntax of any programming language.
- Using this notation, a programmer or compiler can determine whether a program is syntactically correct: whether it adheres to the grammar and punctuation rules of the programming language.

And who popularized this notation?

Language and Syntax

What did Backus do?

- He invented a notation (based on the work of logician Emil Post) that was simple, precise, and powerful enough to describe the syntax of any programming language.
- Using this notation, a programmer or compiler can determine whether a program is syntactically correct: whether it adheres to the grammar and punctuation rules of the programming language.

And who popularized this notation?

- Peter Naur
- As an editor of the ALGOL report, he popularized this notation by using it to describe the complete syntax of ALGOL.
- In Backus and Naur's honor this notation is called Backus-Naur Form (BNF).

Parallel to this work, what was Noam Chomsky doing?

Language and Syntax

Parallel to this work, what was Noam Chomsky doing?

- The linguist Noam Chomsky began working on a harder problem: describing the syntactic structure of natural languages.
- He developed four different notations that describe languages of increasing complexity.
    - Type 0 (unrestricted),
    - Type 1 (context-sensitive),
    - Type 2 (context-free),
    - Type 3 (regular).
- The power of Chomsky's type 2 notation is equivalent to EBNF.

EBNF Rules and Descriptions

What is an EBNF description?

EBNF Rules and Descriptions

What is an EBNF description?

- Is an unordered list of EBNF rules.
- Each EBNF rule has three parts:
  - A left-hand side (LHS),
  - A right-hand side (RHS),
  - The $\Leftarrow$ character separating these two sides.
- The LHS is one italicized word (possibly with underscores) written in lower-case.
  - It names the EBNF rule.
- The RHS supplies a description of this name.
  - It can include names, characters (standing for themselves), and combinations of the four control forms (sequence, choice, option repetition).

EBNF Rules and Descriptions

Which are the four control forms of RHS?

EBNF Rules and Descriptions

Which are the four control forms of RHS?

- Sequence
  - Items appear left-to-right.
  - Their order is important.
- Choice
  - Alternative items are separated by a | (stroke).
  - One item is chosen from this list of alternatives.
  - Their order is unimportant.
- Option
  - The optional item is enclosed between [and] (square-brackets).
  - The item can be either included or discarded.
- Repetition
  - The repeatable item is enclosed between { and } (curly-braces).
  - The item can be repeated zero or more times.

EBNF Rules and Descriptions

Example

EBNF Description: *integer*

digit $\Leftarrow$ *0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9*
integer $\Leftarrow$ *[+ | -] digit{digit}*

Agenda del día

1. **The process to design and implement a programming language**
   - EBNF
   - EBNF Rules and Descriptions
   - Language and Syntax
   - EBNF Rules and Descriptions
   - Proving Symbols Match EBNF Rules
   - Syntax versus Semantics
   - Exercises

Proving Symbols Match EBNF Rules

How to prove that a symbol is legal according to some EBNF rule?

Proving Symbols Match EBNF Rules

How to prove that a symbol is legal according to some EBNF rule?

- We must match all its characters with all the items in the EBNF rule, according to that rule's description.
- If there is an exact match we classify the symbol as legal according to that EBNF description and say it matches.
- Otherwise we classify the symbol as illegal and say it doesn't match.

Proving Symbols Match EBNF Rules

Example

EBNF Description: *integer*

digit	$\Leftarrow$ *0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9*
integer	$\Leftarrow$ *[+ | -] digit{digit}*

- *We can prove that the symbol +142 matches the integer EBNF rule.*
- *We can easily prove that 1,024 is an illegal integer.*

Proving Symbols Match EBNF Rules

What is a tabular proof?

Proving Symbols Match EBNF Rules

What is a tabular proof?

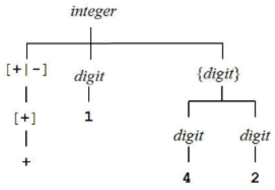
- A tabular proof is a more formal demonstration that a symbol matches an EBNF description.
- The first line in a tabular proof is always the name of the EBNF rule that specifies the syntactic structure we are trying to match the symbol against.
- The last line is the symbol we are matching.
- Each line is derived from the previous according to one of the following rules:
  1. Replace a name (LHS) by its definition (RHS)
  2. Choose an alternative
  3. Determine whether to include or discard an option
  4. Determine the number of times to repeat
- Combining rules 1 and 2 (1&2) simplifies our proofs by allowing us, in a single step, to replace a left-hand side by one of the alternatives in its right-hand side.

Proving Symbols Match EBNF Rules

Example

A tabular proof and its derivation tree that shows the symbol +142 matches the integer EBNF rule.

Status	Reason (rule #)
integer	Given
[+|−]*digit*{*digit*}	Replace *integer* by it RHS (1)
[+]*digit*{*digit*}	Choose + alternative (2)
+*digit*{*digit*}	Include option (3)
+1{*digit*}	Replace the first *digit* by 1 alternative (1&2)
+1*digit digit*	Use two repetitions (rule 4)
+14*digit*	Replace the first *digit* by 4 alternative (1&2)
+142	Replace the first *digit* by 2 alternative (1&2)



Agenda del día

1. **The process to design and implement a programming language**
    - EBNF
    - EBNF Rules and Descriptions
    - Language and Syntax
    - EBNF Rules and Descriptions
    - Proving Symbols Match EBNF Rules
    - Syntax versus Semantics
    - Exercises

Syntax versus Semantics

What to take into account?

Syntax versus Semantics

What to take into account?

- EBNF descriptions specify only syntax: the form in which something is written.
- They do not specify semantics: the meaning of what is written.
- Syntax = Form
- Semantics = Meaning
- Two semantic issues are important in programming languages:
  - Can different symbols have the same meaning?
  - Can one symbol have different meanings?
- Different symbols can have the same meaning and one symbol can have different meanings depending on its context

Agenda del día

1. **The process to design and implement a programming language**
   - EBNF
   - EBNF Rules and Descriptions
   - Language and Syntax
   - EBNF Rules and Descriptions
   - Proving Symbols Match EBNF Rules
   - Syntax versus Semantics
   - Exercises