Syntax of Mini-Pascal (Welsh & McKeag, 1980)

<program> **: : =** **program** *<identifier>* **;** *<block>* **.**

<block> **: : =** *<variable declaration part>*
<statement part>

<variable declaration part> **: : =** *<empty>* **|**
var *<variable declaration>* **;**
{ *<variable declaration>* **;** }

<variable declaration> **: : =** *<identifier >* { **,** *<identifier>* } **:** *<type>*

<type> **: : =** *<simple type>* **|** *<array type>*

<array type> **: : =** **array [** *<index range>* **] of** *<simple type>*

<index range> **: : =** *<integer constant>* **..** *<integer constant>*

<simple type> **: : =** **char | integer | boolean**

<type identifier> **: : =** *<identifier>*

<statement part> **: : =** *<compound statement>*

<compound statement> **: : =** **begin** *<statement>*{ **;** *<statement>* } **end**

<statement> **: : =** *<simple statement>* **|** *<structured statement>*

<simple statement> **: : =** *<assignment statement>* **|** *<read statement>* **|** *<write statement>*

<assignment statement> **: : =** *<variable>* **:=** *<expression>*

<read statement> **: : =** **read (** *<variable>* { **,** *<variable>* } **)**

<*write statement*> **::=** **write (** <*variable*> { **,** <*variable*> } **)**

<*structured statement*> **::=** <*compound statement*> | <*if statement*> |
<*while statement*>

<*if statement*> **::=** **if** <*expression*> **then** <*statement*> |
if <*expression*> **then** <*statement*> **else** <*statement*>

<*while statement*> **::=** **while** <*expression*> **do** <*statement*>

<*expression*> **::=** <*simple expression*> |
<*simple expression*> <*relational operator*> <*simple expression*>

<*simple expression*> **::=** <*sign*> <*term*> { <*adding operator*> <*term*> }

<*term*> **::=** <*factor*> { <*multiplying operator*> <*factor*> }

<*factor*> **::=** <*variable*> | <*constant*> | **(** <*expression*> **)** | **not** <*factor*>

<*relational operator*> **::=** **=** | **<>** | **<** | **<=** | **>=** | **>** | **or** | **and**

<*sign*> **::=** + | – | <*empty*>

<*adding operator*> **::=** + | –

<*multiplying operator*> **::=** * | **div**

<*variable*> **::=** <*entire variable*> | <*indexed variable*>

<*indexed variable*> **::=** <*array variable*> **[** <*expression*> **]**

<*array variable*> **::=** <*entire variable*>

<*entire variable*> **::=** <*variable identifier*>

<variable identifier> **::=** *<identifier>*

Lexical grammar

<constant> **::=** *<integer constant>* | *<character constant>* | *<constant identifier>*

<constant identifier> **::=** *<identifier>*

<identifier> **::=** *<letter>* { *<letter or digit>* }

<letter or digit> **::=** *<letter>* | *<digit>*

<integer constant> **::=** *<digit>* { *<digit>* }

<character constant> **::=** '< letter or digit >' | ''< letter or digit > {< letter or digit >}''

<letter> **::=** a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
p | q | r | s | t | u | v | w | x | y | z | A | B | C |
D | E | F | G | H | I | J | K | L | M | N | O | P
| Q | R | S | T | W | V | W | X | Y | Z

<digit> **::=** 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<special symbol> **::=** + | - | * | = | <> | < | > | <= | >= |
(|) | [|] | **:=** | . | , | ; | : | .. | **div** | **or** | **and** | **not** | **if** | **then** | **else** | **of** |
while | **do** | **begin** | **end** | **read** | **write** | **var** | **array** | **function** |
procedure | **program** | **true** | **false** | **char** | **integer** | **boolean**
