**CS445/CS545 Compiler Construction**

**The Dragon Project**

**Context-Free Grammar**

Here is a context-free grammar for a programming language that is given in Appendix A in Dragon 1e.

program:

**program ident ‘**(‘ identifier\_list ‘)’ ‘;’

declarations

subprogram\_declarations

compound\_statement

‘.’

identifier\_list: **ident**

| identifier\_list ‘,’ **ident**

declarations: declarations **var** identifier\_list ‘:’ type ‘;’

|

type: standard\_type

| **array** ‘[‘ **number** ‘..’  **number** ‘]’ **of** standard\_type

standard\_type: **integer**

| **real**

subprogram\_declarations: subprogram\_declarations subprogram\_declaration ‘;’

|

subprogram\_declaration:

subprogram\_head

declarations

compound\_statement

subprogram\_head: **function ident** arguments ‘:’ standard\_type ‘;’

| **procedure ident** arguments ‘;’

arguments: ‘(‘ parameter\_list ‘)’

|

parameter\_list: identifier\_list ‘:’ type

| parameter\_list ‘;’ identifier\_list ‘:’ type

compound\_statement: **begin** optional\_statements **end**

optional\_statements: statement\_list

|

statement\_list: statement

| statement\_list ‘;’ statement

statement: variable **assignop** expression

| procedure\_statement

| compound\_statement

| **if** expression **then** statement **else** statement

| **while** expression **do** statement

variable: **ident**

| **ident** ‘[‘ expression ‘]’

procedure\_statement: **ident**

| **ident** ‘(‘ expression\_list ‘)’

expression\_list: expression

| expression\_list ‘,’ expression

expression: simple\_expression

| simple\_expression **relop** simple\_expression

simple\_expression: term

| sign term

| simple\_expression **addop** term

term: factor

| term **mulop** factor

factor: **ident**

| **ident** ‘(‘ expression\_list ‘)’

| **number**

| ‘(‘ expression ‘)’

| **not** factor

sign: ‘+’ | ‘-’

**Lexical Constructs**

Here are the definitions of all the tokens valid in the language.

1. Comments are surrounded by either a pair of ‘{‘ and ‘}’ or by a pair of “(\*” and “\*)”

Comments may span several lines.

2. Whitespaces are defined to be spaces, tabs and newlines. Whitespaces between tokens are optional. But keywords must be surrounded by whitespaces.

3. Token **ident** for user-defined idenfitiers are defined as a letter followed by a sequence of letters or digits. There is no limit on the length of an identifier.

**ident** ::= **letter** ( **letter** | **digit** )\*

**letter** ::= [a-zA-Z]

**digit** ::= [0-9]

4. Keywords are reserved (may not be used for anything else). These are:

**program** *starts the main program*

**begin** *starts a new block*

**end** *ends a block*

**var** *starts a list of identifier names*

**array** *signals an array type*

**of** *used in array type declaration*

**integer** *basic integer type*

**real** *basic real (float) type*

**function** *starts a function (returns values,have no side-effects) declaration*

**procedure** *starts a procedure (returns no values, may have side-effects) declaration*

**if** *starts an IF statement*

**then** *part of an IF statement*

**else** *starts the ELSE part of an IF-THEN-ELSE statement*

**while** *starts a WHILE statement*

**do** *the DO part of a WHILE-DO statement*

**not** *logical NOT (negation)*

5. The relational operator **relop** include:

= *equal*

<> *not-equal*

< *less-than*

<= *less-or-equal*

> *greater-than*

>= g*reater-or-equal*

6. The additive operator **addop** include:

+ *addition for both integer and real arguments*

- addition for both integer and real arguments

or *logical OR*

7. The multiplicative operator **mulop** include:

\* *multiplication for both integer and real arguments*

/ *division for both integer and real arguments*

div *quotient in integer division*

mod *remainder in integer division*

and *logical AND*

8. The *lexeme* for the assignment operator **assignop** is

:=

9. The token **number** matches unsigned integers.

**number** ::= **digits** | **digits** ‘.’ **digits**

**digits** ::= **digit digit**\*

**Add-Ons**

1. Allow nesting of subprograms within each other

2. Adjust the “sign” problem

3. Allow array access to appear on right-hand side of an assignment statement

4. Allow IF-THEN statement without the ELSE option

5. (optional) Add a FOR-DO statement

**for** index := 1 **to** size **do**

result := result + index