ECE 683 Compiler Theory
Final Project

TJ Ellis

February 12, 2011

1 Parser

1.1 Left Factored Grammar with Left Recursion Eliminated

The original grammar provided had a few ambiguous production choices that needed to be left-factored and a few left-recursive rules that needed to be eliminated. Specifically, ⟨*function_call*⟩ and ⟨*name*⟩ both started with ⟨*identifier*⟩ and needed to be left factored, and ⟨*expression*⟩, ⟨*arithOp*⟩, and ⟨*relation*⟩ are left recursive in the original grammar. Below is the updated grammar, left factored and with left recursion eliminated. (λ is the empty string.)

⟨*function_declaration*⟩ **::=** ⟨*function_header*⟩ ⟨*function_body*⟩

⟨*function_header*⟩ **:: = function**
⟨*identifier*⟩ **(** [⟨*parameter_list*⟩] **)**

⟨*parameter_list*⟩ **::=** ⟨*parameter*⟩ **,** ⟨*parameter_list*⟩
| ⟨*parameter*⟩

⟨*parameter*⟩ **::=** ⟨*type_mark*⟩ ⟨*variable_declaration*⟩

⟨*function_body*⟩ **::=** **(** ⟨*declaration*⟩ **;** **)***
begin
(⟨*statement*⟩ **;** **)***
end function

⟨*declaration*⟩ **::=** ⟨*type_mark*⟩ ⟨*declaration2*⟩

⟨*declaration2*⟩ **::=** ⟨*function_declaration*⟩
| ⟨*variable_declaration*⟩

⟨*variable_declaration*⟩ **::=** ⟨*identifier*⟩ [[⟨*array_size*⟩]]

⟨*type_mark*⟩ **::=** **integer**
| **boolean**
| **string**

⟨*array_size*⟩ **::=** ⟨*number*⟩

⟨*statement*⟩ **::=** ⟨*assignment_statement*⟩
| ⟨*if_statement*⟩
| ⟨*while_statement*⟩
| ⟨*case_statement*⟩

⟨*assignment_statement*⟩ **::=** ⟨*destination*⟩ **:=** ⟨*expression*⟩

⟨*destination*⟩ **::=** ⟨*identifier*⟩ [[⟨*expression*⟩]]

⟨*if_statement*⟩ **::= if** ⟨*expression*⟩ **then (** ⟨*statement*⟩ **;)+**
[else (⟨*statement*⟩ **;)+]**
end if

⟨*while_statement*⟩ **::= while** ⟨*expression*⟩ **(** ⟨*statement*⟩ **;)***
end while

⟨*case_statement*⟩ **::= case** ⟨*expression*⟩ **is**
(when ⟨*number*⟩ **then (** ⟨*statement*⟩ **;)+)+**
[default then (⟨*statement*⟩ **;)+]**
end case

⟨*identifier*⟩ **::= [a-zA-Z][a-zA-Z0-9_]***

⟨*expression*⟩ **::= [not]** ⟨*arithOp*⟩ ⟨*expression2*⟩

⟨*expression2*⟩ **::= and** ⟨*arithOp*⟩ ⟨*expression2*⟩
| **or** ⟨*arithOp*⟩ ⟨*expression2*⟩
| λ

⟨*arithOp*⟩ **::=** ⟨*relation*⟩ ⟨*arithOp2*⟩

⟨*arithOp2*⟩ **::= +** ⟨*relation*⟩ ⟨*arithOp2*⟩
| **-** ⟨*relation*⟩ ⟨*arithOp2*⟩
| **&** ⟨*relation*⟩ ⟨*arithOp2*⟩
| **|** ⟨*relation*⟩ ⟨*arithOp2*⟩
| λ

⟨*relation*⟩ **::=** ⟨*term*⟩ ⟨*relation2*⟩

⟨*relation2*⟩ **::= <** ⟨*term*⟩ ⟨*relation2*⟩
| **>** ⟨*term*⟩ ⟨*relation2*⟩
| **==** ⟨*term*⟩ ⟨*relation2*⟩
| **!=** ⟨*term*⟩ ⟨*relation2*⟩
| λ

⟨*term*⟩ **::=** ⟨*factor*⟩ ***** ⟨*term*⟩
| ⟨*factor*⟩ **/** ⟨*term*⟩
| ⟨*factor*⟩

⟨*factor*⟩ **::= (** ⟨*expression*⟩ **)**
| ⟨*function_call*⟩
| ⟨*name*⟩

| ⟨*number*⟩
| ⟨*string*⟩

⟨*starts_with_ID*⟩ **::=** ⟨*identifier*⟩ ⟨*name_or_function_call*⟩

⟨*name_or_function_call*⟩ **::=** [[⟨*expression*⟩]]
| **([argument_list])**

⟨*argument_list*⟩ **::=** ⟨*expression*⟩ **,** ⟨*argument_list*⟩
| ⟨*expression*⟩

⟨*number*⟩ **::=** **[0-9]+**

⟨*string*⟩ **:: =** **[a-zA-Z0-9_]***