March 7, 2014

Jani Viherväs
jani.vihervas@cs.helsinki.fi

Mini-PL Interpreter

58144 Compilers Project

Grammar:

$$
\begin{aligned}
<prog> &\rightarrow <stmts> \\
<stmts> &\rightarrow <stmt> \ ; \ <stmts'> \\
<stmts'> &\rightarrow \epsilon \mid <stmts> \\
<stmt> &\rightarrow \textbf{var} \ <ident'> \ : \ <type> <stmt'> \\
&\mid <ident> \ := \ <expr> \\
&\mid \textbf{for} \ <ident> \ \textbf{in} \ <expr> \ .. \ <expr> \ \textbf{do} \ <stmts> \ \textbf{end for} \\
&\mid \textbf{read} \ <ident> \\
&\mid \textbf{print} \ <expr> \\
&\mid \textbf{assert} \ (<expr>) \\
<stmt'> &\rightarrow \epsilon \mid := \ <expr> \\
<expr> &\rightarrow <opnd> <expr'> \\
<expr'> &\rightarrow \epsilon \mid <op> <opnd> \\
<opnd> &\rightarrow <int> \\
&\mid <string> \\
&\mid <opnd'> <bool> \\
&\mid <ident> \\
&\mid (\ <expr> \ ) \\
<opnd'> &\rightarrow \epsilon \mid <unary> \\
<type> &\rightarrow \textbf{int} \mid \textbf{string} \mid \textbf{bool} \\
<reserved \ \ keyword> &\rightarrow \textbf{var} \mid \textbf{for} \mid \textbf{end} \mid \textbf{in} \mid \textbf{do} \mid \textbf{read} \mid \textbf{print} \mid \textbf{assert} \\
&\mid \textbf{int} \mid \textbf{string} \mid \textbf{bool} \mid \texttt{true} \mid \texttt{false} \\
<unary> &\rightarrow \textbf{!} \\
<op> &\rightarrow + \mid \textbf{-} \mid \textbf{*} \mid / \mid < \mid > \mid <= \mid >= \mid \textbf{!}= \mid = \mid \&
\end{aligned}
$$

$<ident'>$ adds identifier to symbol table, where as $<ident>$ looks the identifier from the symbol table. Operators $>$, $<=$, $>=$ and $!=$ are added, because they are very easy to implement.

Predict sets:

Production	Predict set
$< prog >$	**var**, $< ident >$, **for**, **read**, **print**, **assert**
$< stmts >$	**var**, $< ident >$, **for**, **read**, **print**, **assert**
$< stmts' >$	$$ **var**, $< ident >$, **for**, **read**, **print**, **assert**
$< stmt >$	**var** $< ident >$ **for** **read** **print** **assert**
$< stmt' >$	; :=
$< expr >$	`i`, `s`, **!**, `b`, $< ident >$, (
$< expr' >$	;, .., **do** `i`, `s`, **!**, `b`, $< ident >$, (

$< opnd >$	`i`
	`s`
	!, `b`
	$< ident >$
	(

$< opnd' >$	`b` **!**
$< type >$	**int** **string** **bool**