Project Part 2

Monday, March 28, 2022 7:07 PM

Cameron S. Williamson
MyPL to Golang

lexer → AST parser → Static checker → go visitor → printing
run on VM ← Create bytecode

- Package main
- Imports will be required
  - fmt.Print()
- Weird Behavior:
  - Getting characters
    - Get(string, int) == string[int]
  - Reading strings
    - Reader := bufio.NewReader(os.Stdin)
    - Fmt.Print("Enter text: ")
    - Text, _ := reader.ReadString('\n')
    - Fmt.Println(text)

is this a translation?

Program	(<tdecl> | <fdecl>)*	(<tdecl> | <fdecl>)*
Tdecl	TYPE ID LBRACE <vdecls> RBRACE	TYPE ID STRUCT LBRACE <vdecls> RBRACE
Vdecls	(<vdecl_stmt>)*	(<vdecl_stmt>)*
Fdecl	FUN (<dtype>|VOID) ID LPAREN <params> RPAREN LBRACE <stmts> RBRACE	FUN ID LPAREN <params> RPAREN LBRACE <stmts>RBRACE
Params	<dtype> ID (COMMA <dtype> ID)*ϵ	ID <dtype> (COMMA <dtype> ID)*ϵ
Dtype	INT_TYPE|DOUBLE_TYPE|BOOL_TYPE|CHAR_TYPE|STRING_TYPE|ID	INT_TYPE|DOUBLE_TYPE|BOOL_TYPE|CHAR_TYPE|STRING_TYPE|ID
Stmts	(<stmt>)*	(<stmt>)*
Stmt	<vdecl_stmt>|<assign_stmt>|<cond_stmt>|<while_stmt>|<for_stmt>|<call_expr>|<ret_stmt>|<delete_stmt>	<vdecl_stmt>|<assign_stmt>|<cond_stmt>|<while_stmt>|<for_stmt>|<call_expr>|<ret_stmt>|<delete_stmt>
Vdecl_stmt	VAR (<dtype>|ϵ) ID ASSIGN <expr>	(VAR ID dtype ASSIGN <expr>) | (ID COLON ASSIGN <expr>)
Assign_stmt	<lvalue> ASSIGN <expr>	<lvalue> ASSIGN <expr>
Lvalue	ID (DOT ID)*	ID (DOT ID)*
Cond_stmt	IF <expr> LBRACE <stmts> RBRACE <condt>	IF <expr> LBRACE <stmts> RBRACE <condt>
Condt	ELIF <expr> LBRACE <stmts> RBRACE <condt> | ELSE LBRACE <stmts> RBRACE |ϵ	ELIF <expr> LBRACE <stmts> RBRACE <condt> | ELSE LBRACE <stmts> RBRACE |ϵ
While_stmt	WHILE <expr> LBRACE <stmts> RBRACE	FOR <expr> LBRACE <stmts> RBRACE
For_stmt	FOR ID FROM <expr> (UPTO | DOWNTO) <expr> LBRACE <stmts> RBRACE	FOR <vdecl_stmt> SEMICOLON <expr> <expr> R
Call_expr	ID LPAREN <args> RPAREN	ID LPAREN <args> RPAREN
Args	<expr> (COMMA <expr>)*|ϵ	<expr> (COMMA <expr>)* | ϵ
Ret_stmt	RETURN (<expr> | ϵ)	RETURN (<expr> |ϵ
Delete_stmt	DELETE ID	DNE
Expr	(<rvalue> | NOT <expr> | LPAREN <expr> RPAREN) (<operator> <expr> | ϵ)	
Operator	PLUS | MINUS | DIVIDE | MULTIPLY | MODULO | AND | OR | EQUAL | LESS_THAN | GREATER_THAN | LESS_THAN_EQUAL | GREATER_THAN_EQUAL | NOT_EQUAL	
Rvalue	<pval> | NIL | NEW ID | <idrval> | <call_expr> | NEG <expr>	
Pval	INT_VAL | DOUBLE_VAL | BOOL_VAL | CHAR_VAL | STRING_VAL	
Idrval	ID (DOT ID)*	

- Things I need to watch out for
  - Converting the loops
    - A while loop in MyPL is a for loop in golang.
    - A for loop in MyPL is a different type of for loop in golang.
  - Flags for imports.

I'd like to see a performance comparison of the translation vs

"normal" Hyper-
since the translation will incur
some overhead