Trevor Newell

Jaron Foster

Compiler Construction – Assignment 4

FIRST[Command] := FIRST[Single-Command]

:= {Identifier, begin, let, if, while}

FOLLOW[Command] := {;,end,$}

FIRST[SingleCommand] := FIRST[Vname] ∪ FOLLOW[Single-Command] ∪

{Identifier, begin, let, if, while}

:= {;, end, $, Identifier, begin, let, if, while, in}

FOLLOW[SingleCommand] := {else} ∪ FOLLOW[Command] ∪ FIRST[SingleCommand] ∪ FOLLOW[SingleDecl]

:= {else, ;, end, Identifier, begin, let, if, while, in,$}

FIRST[Expression] := FIRST[SecondaryExpr] ∪ {let, if}

:= {Identifier, Character, Number, Operator,(,{,[, let, if}

FOLLOW[Expression] := {then, else, do} ∪ {,} ∪ {)} ∪ {]} ∪ FOLLOW[SingleCommand] ∪

FOLLOW[RecordAgg] ∪ FOLLOW[ArrayAgg] ∪ FOLLOW[SingleDecl] ∪

FOLLOW[ActualParam]

:= {then, else, do, COMMA,),],SEMICOLON, end, Identifier, begin, let, if, while, in, character, Number, Operator, (, {, [, $}

FIRST[SecondaryExpr] := FIRST[PrimaryExpr]

:= {Identifier, Character, Number, Operator ,( , {, [}

FOLLOW[SecondaryExpr] := {Operator} ∪ FOLLOW[Expression]

:= {then, else, do, COMMA,),],SEMICOLON, end, Identifier, begin, let, if, while, in, character, Number, Operator, (, {, [, $}

FIRST[PrimaryExpr] := {Identifier, Character, Number, Operator , (, {, [}

FOLLOW[PrimaryExpr] := FOLLOW[SecondaryExpr]

:= {then, else, do, COMMA,),],SEMICOLON, end, Identifier, begin, let, if, while, in, character, Number, Operator, (, {, [, $}

FIRST[RecordAgg] := {Identifier}

FOLLOW[RecordAgg] := {}}

FIRST[ArrayAgg] := FIRST[Expression]

:= { Identifier, Character, Number, Operator,(,{,[, let, if}

FOLLOW[ArrayAgg] := {]}

FIRST[Vname] := {Identifier}

FOLLOW[Vname] := {.} ∪ {[} ∪ {:=} ∪ FOLLOW[ActualParam] ∪ FOLLOW[PrimaryExpr]

:= {DOT, [, :=, then, else, do, COMMA, ), ], SEMICOLON, end, Identifier, begin, let, if, while, in, character, Number, Operator, (, {, $}

FIRST[Declaration] := FIRST[SingleDecl]

:= {const, var, proc, func, type}

FOLLOW[Declaration] := {;,in}

FIRST[SingleDecl] := {const, var, proc, func, type}

FOLLOW[SingleDecl] := FOLLOW[Declaration]

:= {;,in}

FIRST[FormalParamSeq] := FIRST[ProperFormalParamSeq] ∪ FOLLOW[FormalParamSeq]

:= {Identifier, var, proc, func, )}

FOLLOW[FormalParamSeq] := {)}

FIRST[ProperFormalParamSeq] := FIRST[FormalParam]

:= {Identifier, var, proc, func}

FOLLOW[ProperFormalParamSeq] := FOLLOW[FormalParamSeq]

:= {)}

FIRST[FormalParam] := {Identifier, var, proc, func}

FOLLOW[FormalParam] := {,} ∪ FOLLOW[ProperFormalParamSeq]

:= {COMMA, )}

FIRST[ActualParamSeq] := FIRST[ProperActualParamSeq] ∪ Follow[ActualParamSeq]

:= {Identifier, Character, Number, Operator,(,{,[, let, if,

var, proc, func, ) }

FOLLOW[ActualParamSeq] := {)}

FIRST[ProperActualParamSeq] := FIRST[ActualParam]

:= {Identifier, Character, Number, Operator,(,{,[, let, if,

var, proc, func}

FOLLOW[ProperActualParamSeq] := FOLLOW[ActualParamSeq]

:= {)}

FIRST[ActualParam] := FIRST[Expression] ∪ {var, proc, func}

:= {Identifier, Character, Number, Operator,(,{,[, let, if,

var, proc, func}

FOLLOW[ActualParam] := {COMMA} ∪ FOLLOW[ProperActualParamSeq]

:= {COMMA, )}

FIRST[TypeDenoter] := {Identifier, array, record}

FOLLOW[TypeDenoter] := {,} ∪ {~} ∪ FOLLOW[FormalParam] ∪ FOLLOW[SingleDecl] ∪

FOLLOW[RecTypeDenoter]

:= {COMMA, ), ~, end, SEMICOLON, in}

FIRST[RecTypeDenoter] := {Identifier}

FOLLOW[RecTypeDenoter]:= {end}