COMPILER CONSTRUCTION ASSIGNMENT
PAPER WORK - GRAMMAR (LL(1)).

GROUP NUMBER 18.

MADHAV GUPTA. — 2019A7PS0063P

SANKALA DAS. 2019A7PS0029A

MEENAL GUPTA. 2019A7PS0243A

PRATHAM GUPTA 2019A7PS0051P

YASH GUPTA 2019A7PS1138P

1. The <definetypestmt> was unreachable in the original grammar. Since the <definetypestmt> is used in the type definition of constructed datatypes (record, union), it is a statement which would be present in the typedefinition section. Hence, appropriate production rule: <typeDefinition> → <definetypestmt> introduced.
2. The <primitiveDatatype> in rule 17 is replaced with <dataType> in order to allow nested records [according to language specifications].
3. Left Recursion is sufficiently handled.
4. Ambiguity removed by introduction of left associativity & precedence rules.
5. Left factoring done.
6. Related rules are grouped together in blocks.

The modified grammar rules are written in red. The remaining are mentioned in blue

1. <program> → <otherFunctions> <mainFunction>
2. <mainFunction> → TK_MAIN <stmts> TK_END.
3. <otherFunctions> → <function> <otherFunctions> | Epsilon (ε)
4. <function> → TK_FUNID <input_par> <output_par> TK_SEM <stmts> TK_END
5. <input_par> → TK_INPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR.
6. <output_par> → TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR | ε
7. <parameter_list> → <dataType> TK_ID <remaining_list>
8. <dataType> → <primitiveDatatype> | <constructedDatatype>
9. <primitiveDatatype> → TK_INT | TK_REAL
10. <constructedDatatype> → TK_RECORD TK_RUID | TK_UNION TK_RUID
11. <remaining_list> → TK_COMMA <parameter_list> | ε
12. <stmts> → <typeDefinitions> <declarations> <otherStmts> <returnStmt>.
13. <typeDefinitions> → <typeDefinition> <typeDefinitions> | ε
14. <typeDefinition> → TK_RECORD TK_RUID <fieldDefinitions> TK_ENDRECORD |
    TK_UNION TK_UNION <fieldDefinitions> TK_ENDUNION |
    <definetypestmt>
15. <fieldDefinitions> → <fieldDefinition> <fieldDefinition> ~~<fieldDefinitions>~~ <moreFields>
16. <moreFields> → <fieldDefinition> <moreFields> | ε
17. <fieldDefinition> → TK_TYPE <dataType> TK_COLON TK_FIELDID TK_SEM
18. <declarations> → <declaration> <declarations> | ε
19. <declaration> → TK_TYPE <dataType> TK_COLON TK_ID TK_COLON <global_or_not> TK_SEM.
20. <global_or_not> → TK_GLOBAL | ε
21. <otherStmts> → <stmt> <otherStmts> | ε
22. <stmt> → <assignmentStmt> | <iterativeStmt> | <conditionalStmt> | <ioStmt> |
    <funCallStmt>
23. <assignStmt> → <singleOrRecId> TK_ASSIGNOP <arithmeticExpression> TK_SEM.
24. <singleOrRecId> → TK_ID <singleOrRecIdSuffix> // ~~removing~~ left factoring
25. <singleOrRecIdSuffix> → TK_DOT TK_FIELDID <singleOrRecIdSuffix> | ε
    // recursive for nested records.
26. <funCallStmt> → <outputParameters> TK_CALL TK_FUNID TK_WITH TK_PARAMETERS
    <inputParameters>
27. <outputParameters> → TK_SQL <idList> TK_SQR TK_ASSIGNOP | ε
28. <inputParameters> → TK_SQL <idList> TK_SQR.

29. <iterativeStmt> → TK-WHILE TK-OP <booleanExpression> TK-CL <stmt><otherStmts> TK-ENDWHILE

30. <conditionalStmt> → TK-IF <booleanExpression> TK-THEN <stmt><otherStmts>
<conditionalStmtSuffix> // left factoring.

31. <conditionalStmtSuffix> → TK-ELSE <otherStmts> ENDIF | ENDIF

32. <ioStmt> → TK-READ TK-OP <var> TK-CL TK-SEM |
TK-WRITE TK-OP <var> TK-CL TK-SEM.

~~33.~~ // left associativity and precedence information embedded.
In order to remove ambiguity & left recursion removed.

33. <arithmeticExpression> → <term> <arithmeticExpressionPrime>

34. <arithmeticExpressionPrime> → <addOrSub> <arithmeticExpression> <arithmeticExpressionPrime> | ε.

35. <addOrSub> → TK-PLUS | TK-MINUS.

36. <term> → <factor> <termPrime>

37. <termPrime> → <mulOrDiv> <factor> <termPrime> | ε.

38. <mulOrDiv> → TK-MUL | TK-DIV.

39. <factor> → TK-OP <arithmeticExpression> TK-CL | <var>.

40. <booleanExpression> → TK-OP <booleanExpression> TK-CL <logicalOp> TK-OP
<booleanExpression> TK-CL |
<var> <relationalOp> <var> | TK-NOT <booleanExpression>

41. <var> → TK-ID | TK-NUM | TK-RNUM.

42. <logicalOp> → TK-AND | TK-OR.

43. <relationalOp> → TK-LT | TK-LE | TK-EQ | TK-GT | TK-GE | TK-NE

44. <returnStmt> → TK-RETURN <optionalReturn> TK-SEM.

45. <optionalReturn> → TK-SQL <idList> TK-SQR | ε.

46. <idList> → TK-ID <more_ids>

47. <more_ids> → TK-COMMA <idList> | ε.

48. <definetypestmt> → TK-DEFINETYPE <A> TK-RUID TK-AS TK-RUID.

49. <A> → TK-RECORD | TK-UNION.

CS F363 - GROUP 18

MADHAV GUPTA 2019A7PS0063P
MEENAL GUPTA 2019A7PS0243P
PRATHAM GUPTA 2019A7PS0051P
SANKHA DAS 2019A7PS0029P
YASH GUPTA 2019A7PS1138P

FIRST SETS FOR THE MODIFIED GRAMMAR

FIRST(<program>) = { TK_MAIN, TK_FUNID }

FIRST(<mainFunction>) = { TK_MAIN }

FIRST(<otherFunctions>) = { TK_FUNID, ε }

FIRST(<function>) = { TK_FUNID }

FIRST(<input_par>) = { TK_INPUT }

FIRST(<output_par>) = { TK_OUTPUT, ε }

FIRST(<parameter_list>) = { TK_INT, TK_REAL, TK_RECORD, TK_UNION } = FIRST(<dataType>)

FIRST(<primitiveDatatype>) = { TK_INT, TK_REAL }

FIRST(<constructedDatatype>) = { TK_RECORD, TK_UNION }

FIRST(<remaining_list>) = { TK_COMMA }

FIRST(<stmts>) = { TK_RETURN, TK_UNION, TK_RECORD, TK_TYPE, TK_DEFINETYPE, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL }

FIRST(<typeDefinitions>) = { TK_RECORD, TK_UNION, TK_DEFINETYPE, ε }

FIRST(<typeDefinition>) = { TK_RECORD, TK_UNION, TK_DEFINETYPE }

FIRST(<fieldDefinitions>) = FIRST(<fieldDefinition>) = FIRST(<declaration>) = { TK_TYPE }

FIRST(<moreFields>) = FIRST(<declarations>) = { TK_TYPE, ε }

FIRST(<global_or_not>) = { TK_GLOBAL, ε }

FIRST(<otherStmts>) = { ε, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL }

FIRST(<stmt>) = { TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL }

FIRST(<assignmentStmt>) = FIRST(<singleOrRecId>) = { TK_ID }

FIRST(<singleOrRecIdSuffix>) = { ε, TK_DOT }

FIRST(<funCallStmt>) = { TK_CALL, TK_SQL }

FIRST(<outputParameters>) = { ε, TK_SQL }

FIRST(<inputParameters>) = { TK_SQL }

FIRST(<iterativeStmt>) = { TK_WHILE }

FIRST(<conditionalStmt>) = { TK_IF }

FIRST(<conditionalStmtSuffix>) = { TK_ELSE, TK_ENDIF }

FIRST(<ioStmt>) = { TK_READ, TK_WRITE }

FIRST(<arithmeticExpression>) = { TK_OP, TK_ID, TK_NUM, TK_RNUM }

FIRST(<arithmeticExpressionPrime>) = { TK_PLUS, TK_MINUS, ε }

FIRST(<addOrSub>) = { TK_PLUS, TK_MINUS }

FIRST(<term>) = { TK_OP, TK_ID, TK_NUM, TK_RNUM } = FIRST(<factor>)

FIRST(<termPrime>) = { ε, TK_MUL, TK_DIV }

FIRST(<mulOrDiv>) = { TK_MUL, TK_DIV }

FIRST(<booleanExpression>) = { TK_OP, TK_NOT, TK_ID, TK_NUM, TK_RNUM }

FIRST(<var>) = { TK_ID, TK_NUM, TK_RNUM }

FIRST(<logicalOp>) = { TK_AND, TK_OR }

FIRST(<relationalOp>) = { TK_LT, TK_LE, TK_EQ, TK_NE, TK_GE, TK_GT }

FIRST(<returnStmt>) = { TK_RETURN }

FIRST(<optionalReturn>) = { ε, TK_SQL }

FIRST(<idList>) = { TK_ID }

FIRST(<more_ids>) = { ε, TK_COMMA }

FIRST(<definetypestmt>) = { TK_DEFINETYPE }

FIRST(<A>) = { TK_RECORD, TK_UNION }

CS F363 – GROUP 18

FOLLOW SETS FOR THE MODIFIED GRAMMAR

FOLLOW(<program>) = FOLLOW(<mainFunction>) = { $ }

FOLLOW(<otherFunctions>) = { TK_MAIN }

FOLLOW(<function>) = { TK_MAIN, TK_FUNID }

FOLLOW(<input_par>) = { TK_SEM, TK_OUTPUT }

FOLLOW(<output_par>) = { TK_SEM }

FOLLOW(<parameter_list>) = { TK_SQR }

FOLLOW(<dataType>) = FOLLOW(<primitiveDatatype>)
= FOLLOW(<constructedDatatype>) = { TK_ID, TK_COLON }

FOLLOW(<remaining_list>) = { TK_SQR }

FOLLOW(<stmts>) = { TK_END }

FOLLOW(<typeDefinitions>) = { TK_RETURN, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL, TK_TYPE }

FOLLOW(<typeDefinition>) = { TK_RETURN, TK_WHILE, TK_IF, TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL, TK_TYPE, TK_RECORD, TK_UNION, TK_DEFINETYPE }

FOLLOW(<fieldDefinitions>) = FOLLOW(<moreFields>)
= { TK_ENDRECORD, TK_ENDUNION }

FOLLOW(<fieldDefinition>) = { TK_ENDRECORD, TK_ENDUNION, TK_TYPE }

FOLLOW(<declarations>) = { TK_RETURN, TK_WHILE, TK_IF, TK_SQL, TK_READ, TK_WRITE, TK_CALL, TK_ID }

FOLLOW(<declaration>) = FOLLOW(<declarations>) ∪ { TK_TYPE }

FOLLOW(<global_or_not>) = { TK_SEM }

FOLLOW(<otherStmts>) = { TK_RETURN, TK_ENDWHILE, TK_ELSE, TK_ENDIF }

FOLLOW(<singleOrRecId>) = FOLLOW(<singleOrRecIdStuff>)
= { TK_ASSIGNOP }

FOLLOW(<Stmt>) = FOLLOW(<assignmentStmt>) = FOLLOW(<iterStmt>)
= FOLLOW(<iterativeStmt>) = FOLLOW(<conditionalStmt>)
= FOLLOW(<conditionalStmtSuffix>) = FOLLOW(<funCallStmt>)
= FOLLOW(<inputParameters>)
= { TK_RETURN, TK_WHILE, TK_IF, TK_READ, TK_WRITE,
TK_CALL, TK_ID, TK_SQL, TK_ENDWHILE, TK_ELSE, TK_ENDIF }

FOLLOW(<outputParameters>) = { TK_CALL }

FOLLOW(<arithmeticExpression>) = FOLLOW(<arithmeticExpressionRemain>)
= { TK_SEM, TK_CL }

FOLLOW(<addOrSub>) = FOLLOW(<mulOrDiv>)
= { TK_OP, TK_ID, TK_NUM, TK_RNUM }

FOLLOW(<term>) = FOLLOW(<termRemain>)
= { TK_SEM, TK_PLUS, TK_MINUS, TK_CL }

FOLLOW(<factor>) = { TK_SEM, TK_PLUS, TK_MUL, TK_DIV, TK_MINUS, TK_CL }

FOLLOW(<booleanExpression>) = { TK_CL, TK_THEN }

FOLLOW(<var>) = { TK_CL, TK_SEM, TK_PLUS, TK_MINUS, TK_MUL,
TK_DIV, TK_THEN, TK_LT, TK_LE, TK_EQ, TK_GT,
TK_GE, TK_NE }

FOLLOW(<logicalOp>) = { TK_OP }

FOLLOW(<relationalOp>) = { TK_ID, TK_NUM, TK_RNUM }

FOLLOW(<returnStmt>) = { TK_END }

FOLLOW(<optionalReturn>) = { TK_SEM }

FOLLOW(<idList>) = FOLLOW(<more_ids>) = { TK_SQR }

FOLLOW(<declarestype Stmt>) = { TK_RETURN, TK_WHILE, TK_IF,
TK_READ, TK_WRITE, TK_CALL, TK_ID, TK_SQL,
TK_TYPE, TK_RECORD, TK_UNION, TK_DEFINETYPE }

FOLLOW(<A>) = { TK_RUID }