CFG LL(1) Analysis for Revised Grammar

Table of First Sets, Follow Sets, and LL(1) Condition Verification

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<PS>	{import}	{$}	Only one production, condition satisfied	Not nullable, condition satisfied
<import_st>	{import}	{public, private, protected, static, final, abstract, class, $}	Only one production, condition satisfied	Not nullable, condition satisfied
<import_tail>	{;, .}	{public, private, protected, static, final, abstract, class, $}	First(;) = {;}, First(. *) = {.} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<qualified_name>	{ID}	{;, .}	Only one production, condition satisfied	Not nullable, condition satisfied
<qualified_name_tail>	{., ε}	{;, .}	First(. ID <qualified_name_tail>) = {.}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<qualified_name_tail>) ∩ Follow(<qualified_name_tail>) = {ε} ∩ {;, .} = ∅, condition satisfied
<classname>	{ID}	{(, {, ;, .}	Only one production, condition satisfied	Not nullable, condition satisfied
<main class>	{public}	{$}	Only one production, condition satisfied	Not nullable, condition satisfied
<classes>	{public, private, protected, static, final, abstract, class, ε}	{public}	First(<class><classes>) = {public, private, protected, static, final, abstract, class}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<classes>) ∩ Follow(<classes>) = {ε} ∩ {public} = ∅, condition satisfied
<class>	{public, private, protected, static, final, abstract, class}	{public, private, protected, static, final,	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
		abstract, class, $}		
<class header>	{public, private, protected, static, final, abstract, class}	{extends, {}	Only one production, condition satisfied	Not nullable, condition satisfied
<Inheritance>	{extends, ε}	{{}	First(extends ID) = {extends}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<Inheritance>) ∩ Follow(<Inheritance>) = {ε} ∩ {{} = ∅, condition satisfied
<class body>	{{, ε}	{public, private, protected, static, final, abstract, class, $}	First({) = {{}, First(<Attributes> <class body>) = First(<constructors> <class body>) = First(<Methods> <class body>) = {public, private, protected, static, final, abstract}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<class body>) ∩ Follow(<class body>) = {ε} ∩ {public, private, protected, static, final, abstract, class, $} = ∅, condition satisfied
<Attributes>	{public, private, protected, static, final, abstract, DT}	{public, private, protected, static, final, abstract, class, $, }}	Only one production, condition satisfied	Not nullable, condition satisfied
<Modifiers'>	{public, private, protected, static, final, abstract}	{DT, ID}	First(<Access_Modifier>) = {public, private, protected}, First(static) = {static}, First(final) = {final}, First(abstract) = {abstract} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<Access_Modifier>	{public, private, protected}	{DT, ID}	First(public) = {public}, First(private) = {private},	Not nullable, condition satisfied

Non-Terminal	**First Set**	**Follow Set**	**LL(1) Condition 1 (First Sets Disjoint)** First(protected) = {protected} ⟹ Disjoint, condition satisfied	**LL(1) Condition 2 (Nullable First ∩ Follow Empty)**
<constructor>	{public, private, protected, static, final, abstract, ID}	{public, private, protected, static, final, abstract, class, $, }}	Only one production, condition satisfied	Not nullable, condition satisfied
<constructor header>	{public, private, protected, static, final, abstract, ID}	{{}	Only one production, condition satisfied	Not nullable, condition satisfied
<Methods>	{public, private, protected, static, final, abstract, DT, ε}	{public, private, protected, static, final, abstract, class, $, }}	First(<Method> <Methods>) = {public, private, protected, static, final, abstract, DT}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<Methods>) ∩ Follow(<Methods>) = {ε} ∩ {public, private, protected, static, final, abstract, class, $, }} = ∅, condition satisfied
<Method>	{public, private, protected, static, final, abstract, DT}	{public, private, protected, static, final, abstract, DT, }}	Only one production, condition satisfied	Not nullable, condition satisfied
<Method header>	{public, private, protected, static, final, abstract, DT}	{{}	Only one production, condition satisfied	Not nullable, condition satisfied
<Method body>	{{}	{public, private, protected, static, final, abstract, DT, }}	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<Parameters>	{DT, ε}	{)}	First(<Parameter> <Parameter'>) = {DT}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<Parameters>) ∩ Follow(<Parameters>) = {ε} ∩ {)} = ∅, condition satisfied
<Parameter'>	{ε, DT}	{)}	First(ε) = {ε}, First(<Parameters>) = {DT, ε} ⟹ NOT DISJOINT because of ε, condition violated	Nullable: Yes, First(<Parameter'>) ∩ Follow(<Parameter'>) = {ε} ∩ {)} = ∅, condition satisfied
<Parameter>	{DT}	{DT, ε,)}	Only one production, condition satisfied	Not nullable, condition satisfied
<MST>	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, ε}	{}, catch}	First(<SST> <MST>) = {ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<MST>) ∩ Follow(<MST>) = {ε} ∩ {}, catch} = ∅, condition satisfied
<SST>	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, }, catch, else}	Production first sets are unique, condition satisfied	Not nullable, condition satisfied
<Unary Opr>	{inc, dec, NOT}	{ID, const, (, -}	First(inc dec) = {inc, dec}, First(NOT) = {NOT} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<Binary Opr>	{PM, MDM, RO1, RO2, AND, OR}	{ID, const, (, -, NOT}	Each operator has a distinct first token, condition satisfied	Not nullable, condition satisfied
<assign_st>	{ID}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<Assign Opr>	{=, +=, -=, *=, /=, %=}	{ID, inc, dec, NOT,	First sets for all productions are distinct,	Not nullable, condition satisfied

Non-Terminal	**First Set**	**Follow Set** PM, (, const}	**LL(1) Condition 1 (First Sets Disjoint)** condition satisfied	**LL(1) Condition 2 (Nullable First ∩ Follow Empty)**
<Method Call>	{ID}	{;, ., ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<constructor call>	{new}	{;, ., ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<Args>	{ID, inc, dec, NOT, PM, const, (, new, ε}	{)}	First(<Exp><Args'>) = {ID, inc, dec, NOT, PM, const, (, new}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<Args>) ∩ Follow(<Args>) = {ε} ∩ {)} = ∅, condition satisfied
<Args'>	{ε, ,}	{)}	First(ε) = {ε}, First(,<Args>) = {,} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<Args'>) ∩ Follow(<Args'>) = {ε} ∩ {)} = ∅, condition satisfied
<TS>	{this, super, ID}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<This or Super or ID>	{this, super, ID}	{(}	First(this) = {this}, First(super) = {super}, First(ID) = {ID} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<Return St>	{return}	{;}	First(return <Exp>) = {return}, First(return this.) = {return} ⟹ NOT DISJOINT, condition violated	Not nullable, condition satisfied
<main method>	{public}	{$}	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<m.m body>	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, ε}	{}}	Only one production, condition satisfied	Not nullable, condition satisfied
<m.m header>	{public}	{{}	Only one production, condition satisfied	Not nullable, condition satisfied
<object decl>	{Type}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<obj header>	{Type}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<new expr>	{new}	{;, ., ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<arg list opt>	{ID, const, new, ε}	{)}	First(<arg list>) = {ID, const, new}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<arg list opt>) ∩ Follow(<arg list opt>) = {ε} ∩ {)} = ∅, condition satisfied
<arg list>	{ID, const, new}	{)}	Only one production, condition satisfied	Not nullable, condition satisfied
<arg list tail>	{,, ε}	{)}	First(, <arg list>) = {,}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<arg list tail>) ∩ Follow(<arg list tail>) = {ε} ∩ {)} = ∅, condition satisfied
Expr	{ID, const, new}	{,,)}	First(ID <expr tail>) = {ID}, First(<Const>) = {const}, First(<new expr>) = {new} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<expr tail>	{(, ε}	{,,)}	First((<arg list opt>)) = {(}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<expr tail>) ∩ Follow(<expr tail>) = {ε} ∩ {,,)} = ∅, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<Type>	{ID}	{ID}	Only one production, condition satisfied	Not nullable, condition satisfied
<Const>	{int_const, string_const, boolean_const}	{;, ., ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(int_const) = {int_const}, First(string_const) = {string_const}, First(boolean_const) = {boolean_const} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<object call>	{ID, this, super, new, Method Call}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<primary expr>	{ID, this, super, new, Method Call}	{.}	First(ID) = {ID}, First(this) = {this}, First(super) = {super}, First(<new expr>) = {new}, First(<method call>) = {ID} ⟹ NOT DISJOINT (ID and method call), condition violated	Not nullable, condition satisfied
<access chain>	{., ε}	{;}	First(<access> <access chain>) = {.}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<access chain>) ∩ Follow(<access chain>) = {ε} ∩ {;} = ∅, condition satisfied
<access>	{.}	{., ;}	Only one production, condition satisfied	Not nullable, condition satisfied
<access tail>	{(, ε}	{., ;}	First((<arg list opt>)) = {(}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<access tail>) ∩ Follow(<access tail>) = {ε} ∩ {., ;} = ∅, condition satisfied
<Exp>	{ID, const, Method Call, constructor call, -, NOT, (}	{;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<OE>	{ID, const, Method Call, constructor call, -, NOT, ()}	{;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<OE'>	{OR, ε}	{;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(OR <AE> <OE'>) = {OR}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<OE'>) ∩ Follow(<OE'>) = {ε} ∩ {;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<AE>	{ID, const, Method Call, constructor call, -, NOT, ()}	{OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<AE'>	{AND, ε}	{OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(AND <RE2> <AE'>) = {AND}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<AE'>) ∩ Follow(<AE'>) = {ε} ∩ {OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<RE2>	{ID, const, Method Call, constructor call, -, NOT, ()}	{AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<RE2'>	{RO2, ε}	{AND, OR, ;, ,,), ID, inc, dec, NOT, PM,	First(RO2 <RE1> <RE2'>) = {RO2}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<RE2'>) ∩ Follow(<RE2'>) = {ε} ∩ {AND, OR, ;, ,,), ID, inc, dec, NOT, PM,

Non-Terminal	**First Set**	**Follow Set** MDM, RO1, RO2, AND, OR}	**LL(1) Condition 1 (First Sets Disjoint)**	**LL(1) Condition 2 (Nullable First ∩ Follow Empty)** MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<RE1>	{ID, const, Method Call, constructor call, -, NOT, (}	{RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<RE1'>	{RO1, ε}	{RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(RO1 <E> <RE1'>) = {RO1}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<RE1'>) ∩ Follow(<RE1'>) = {ε} ∩ {RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<E>	{ID, const, Method Call, constructor call, -, NOT, (}	{RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	Only one production, condition satisfied	Not nullable, condition satisfied
<E'>	{PM, ε}	{RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(PM <T> <E'>) = {PM}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<E'>) ∩ Follow(<E'>) = {ε} ∩ {RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<T>	{ID, const, Method Call, constructor call, -, NOT, (}	{PM, RO1, RO2, AND, OR, ;, ,,), ID, inc,	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
		dec, NOT, PM, MDM, RO1, RO2, AND, OR}		
<T'>	{MDM, ε}	{PM, RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(MDM <F> <T'>) = {MDM}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<T'>) ∩ Follow(<T'>) = {ε} ∩ {PM, RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR} = ∅, condition satisfied
<F>	{ID, const, Method Call, constructor call, -, NOT, (}	{MDM, PM, RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(<primary>) = {ID, const, Method Call, constructor call}, First(-<F>) = {-}, First(NOT <F>) = {NOT}, First((<OE>)) = {(} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<primary>	{ID, const, Method Call, constructor call, assign st}	{MDM, PM, RO1, RO2, AND, OR, ;, ,,), ID, inc, dec, NOT, PM, MDM, RO1, RO2, AND, OR}	First(ID) = {ID}, First(const) = {const}, First(<method call>) = {ID}, First(<constructor call>) = {new}, First(<assign st>) = {ID} ⟹ NOT DISJOINT (ID, method call, assign st), condition violated	Not nullable, condition satisfied
<try>	{try}	{ID, inc, dec, NOT, PM, this, super, return, if,	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	**First Set**	**Follow Set** while, for, DT, }, catch, else}	**LL(1) Condition 1 (First Sets Disjoint)**	**LL(1) Condition 2 (Nullable First ∩ Follow Empty)**
<catch_list>	{catch}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, }, catch, else}	Only one production, condition satisfied	Not nullable, condition satisfied
<catch_list_tail>	{catch, ε}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, }, catch, else}	First(catch (ID) { <MST> } <catch_list_tail>) = {catch}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<catch_list_tail>) ∩ Follow(<catch_list_tail>) = {ε} ∩ {ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, }, catch, else} = ∅, condition satisfied
<throw>	{throw}	{;}	Only one production, condition satisfied	Not nullable, condition satisfied
<throw_options>	{ID, Const, new}	{;}	First(ID) = {ID}, First(Const) = {Const}, First(new ID (<param_list>)) = {new} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<While St>	{while}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, }, catch, else}	Only one production, condition satisfied	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<cond>	{ID, Const, inc, dec, NOT, PM, const, Method Call, constructor call, -, ()}	{)}	First(<Const_or_ID>) = {ID, Const}, First(<Const_or_ID> <ROP> <Const_or_ID>) = {ID, Const}, First(<exp>) = {ID, inc, dec, NOT, PM, const, Method Call, constructor call, -, ()} ⟹ NOT DISJOINT, condition violated	Not nullable, condition satisfied
<ROP>	{RO1, RO2}	{ID, Const}	First(RO1) = {RO1}, First(RO2) = {RO2} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<loop_body>	{;, ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, {}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, else, }, catch}	First(;) = {;}, First(<SST>) = {ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT}, First({<MST>}) = {{} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<for_loop>	{for}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, else, }, catch}	Only one production, condition satisfied	Not nullable, condition satisfied
<F1>	{DT, ID, =, ;}	{ID, Const, ε, ;}	First(<dt_dec>) = {DT}, First(<assign_st>) = {ID, =}, First(;) = {;} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<F2>	{ID, Const, inc, dec, NOT, PM,	{;}	First(<cond>) = {ID, Const, inc, dec, NOT, PM, const,	Nullable: Yes, First(<F2>) ∩ Follow(<F2>) = {ε} ∩ {;} = ∅,

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint)	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
	const, Method Call, constructor call, -, (, ε}		Method Call, constructor call, -, (}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	condition satisfied
<F3>	{inc, dec, ID, =, null}	{)}	First(<inc_dec>) = {inc, dec}, First(<assign_st>) = {ID, =}, First(null) = {null} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<if>	{if}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, else, }, catch}	Only one production, condition satisfied	Not nullable, condition satisfied
<else>	{else, null}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, else, }, catch}	First(else <loop_body>) = {else}, First(null) = {null} ⟹ Disjoint, condition satisfied	Not nullable, condition satisfied
<array_dec>	{DT, ID}	{ID, inc, dec, NOT, PM, this, super, return, if, while, for, DT, else, }, catch}	Only one production, condition satisfied	Not nullable, condition satisfied
<arr_type>	{DT, ID}	{ID}	First(DT) = {DT}, First(ID) = {ID} ⟹ Disjoint, condition	Not nullable, condition satisfied

Non-Terminal	First Set	Follow Set	LL(1) Condition 1 (First Sets Disjoint) satisfied	LL(1) Condition 2 (Nullable First ∩ Follow Empty)
<arr_const_or_id>	{ε, ID, Const}	{}}	First(ε) = {ε}, First(<Const_or_ID>) = {ID, Const}, First(ID ,) = {ID}, First(Const ,) = {Const} ⟹ NOT DISJOINT (ID and Const appear in multiple productions), condition violated	Nullable: Yes, First(<arr_const_or_id>) ∩ Follow(<arr_const_or_id>) = {ε} ∩ {}} = ∅, condition satisfied
<dt_dec>	{=, ε}	{;}	First(<var_init> <var_init_tail> ;) = {=, ε} ⟹ Only one production, condition satisfied	Not nullable, condition satisfied
<var_init>	{=, ε}	{,, ;}	First(= <Const_or_ID>) = {=}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<var_init>) ∩ Follow(<var_init>) = {ε} ∩ {,, ;} = ∅, condition satisfied
<var_init_tail>	{,, ε}	{;}	First(, ID <var_init> <var_init_tail>) = {,}, First(ε) = {ε} ⟹ Disjoint, condition satisfied	Nullable: Yes, First(<var_init_tail>) ∩ Follow(<var_init_tail>) = {ε} ∩ {;} = ∅, condition satisfied

Summary of LL(1) Violations

1. **<Parameter'>**: First(ε) and First(<Parameters>) both contain ε, making them not disjoint.
2. **<Return St>**: Both productions start with "return", making First sets not disjoint.
3. **<primary expr>**: ID and <method call> both start with ID, making First sets not disjoint.
4. **<primary>**: ID, <method call>, and <assign st> all start with ID, making First sets not disjoint.
5. **<cond>**: Multiple productions have overlapping First sets.
6. **<arr_const_or_id>**: Multiple productions have overlapping First sets for ID and Const.

To transform this grammar into a proper LL(1) grammar, these conflicts need to be resolved through left-factoring or other grammar transformations.