Antlr4

Version: d85ea0649ae9aa3a2d50ef0172d80d4a22e88369

Parents:

163ec98afc995f3935cb0971e512de3781dd8919

e3fc04bda1ce8bafaa55a2a584f26f2238c910c8

Merge base:

5bd415b19526aea8719eeaf6e1cb8e0174ec9b9f

antlr4/tool/src/org/antlr/v4/tool/ErrorType.java

Chunk 01: (version 2/Enum)

```
         */
<<<<<<< HEAD
        CANNOT_CREATE_TARGET_GENERATOR(31, "ANTLR cannot generate '<arg>' code as of version
"+ Tool.VERSION, ErrorSeverity.ERROR_ONE_OFF),
=======
        CANNOT_CREATE_TARGET_GENERATOR(31, "ANTLR cannot generate <arg> code as of version
"+ Tool.VERSION, ErrorSeverity.ERROR),
>>>>>>> e3fc04bda1ce8bafaa55a2a584f26f2238c910c8
        /**
```

```
         */
        CANNOT_CREATE_TARGET_GENERATOR(31, "ANTLR cannot generate <arg> code as of version
"+ Tool.VERSION, ErrorSeverity.ERROR),
        /**
```

Version: 2f902da3d2b932717a92c93a30c53b91de754adf

Parents:

4055c2417132ff81ff4e2fe98b004bad325a99c1

df0bbf42e1f17f239dfa88b1501874f9b6388720

Merge base:

6d1d0e0488670c5f394f2b1b025f57e5082d9856

antlr4/runtime/Java/src/org/antlr/v4/runtime/Parser.java

Chunk 02: (concatenation/Annotation, method signature, return statement, variable)

```
	}

<<<<<<< HEAD
	public void setContext(ParserRuleContext ctx) {
		_ctx = ctx;
=======
	@Override
	public boolean precpred(RuleContext localctx, int precedence) {
		return precedence >= _precedenceStack.peek();
>>>>>>> df0bbf42e1f17f239dfa88b1501874f9b6388720
	}
```

```
	}

	public void setContext(ParserRuleContext ctx) {
		_ctx = ctx;
	}

	@Override
	public boolean precpred(RuleContext localctx, int precedence) {
		return precedence >= _precedenceStack.peek();
	}
```

antlr4/tool/test/org/antlr/v4/test/BaseTest.java

Chunk 03: (concatenation/Import)

```
import org.antlr.v4.runtime.misc.Nullable;
<<<<<<< HEAD
=======
import org.antlr.v4.runtime.misc.Pair;
>>>>>>> df0bbf42e1f17f239dfa88b1501874f9b6388720
import org.antlr.v4.runtime.tree.ParseTree;
```

```
import org.antlr.v4.runtime.misc.Nullable;
import org.antlr.v4.runtime.misc.Pair;
import org.antlr.v4.runtime.tree.ParseTree;
```

Version: b14ca56441196d63b8974455c0050bfaee4cb3a4

Parents:

05b0f645ef1359d787df472036b7068314da535a

b80ad5052d1b693be6e5c0a2be8bf87e15b86f18

Merge base:

f7d0cacb09e9051af3f39f8f4c7be5f566665486

antlr4/runtime/Java/src/org/antlr/v4/runtime/Parser.java

Chunk 04: (concatenation/ commentary, member initialization, variable)

```
	protected TokenStream _input;

<<<<<<< HEAD
	protected final IntegerStack _precedenceStack;
	{
		_precedenceStack = new IntegerStack();
		_precedenceStack.push(0);
	}

	/** The RuleContext object for the currently executing rule. This
	 *  must be non-null during parsing, but is initially null.
	 *  When somebody calls the start rule, this gets set to the
	 *  root context.
=======
	/**
	 * The {@link ParserRuleContext} object for the currently executing rule.
	 * This is always non-null during the parsing process.
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
	 */
```

```
	protected TokenStream _input;

	protected final IntegerStack _precedenceStack;
	{
		_precedenceStack = new IntegerStack();
		_precedenceStack.push(0);
	}

	/**
	 * The {@link ParserRuleContext} object for the currently executing rule.
	 * This is always non-null during the parsing process.
	 */
```

Chunk 05: (combination/method invocation, variable)

```
		_syntaxErrors = 0;
<<<<<<< HEAD
		_tracer = null;
		_precedenceStack.clear();
		_precedenceStack.push(0);
=======
		setTrace(false);
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
		ATNSimulator interpreter = getInterpreter();
```

```
                _syntaxErrors = 0;
                setTrace(false);
                _precedenceStack.clear();
                _precedenceStack.push(0);
                ATNSimulator interpreter = getInterpreter();
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ATNSimulator.java

Chunk 06: (version 2/static block, variable)

```
        static {
<<<<<<< HEAD
                SERIALIZED_VERSION = 3;
=======
                /* This value should never change. Updates following this version are
                 * reflected as change in the unique ID SERIALIZED_UUID.
                 */
                SERIALIZED_VERSION = 3;
        }

        public static final UUID SERIALIZED_UUID;
        static {
                /* WARNING: DO NOT MERGE THIS LINE. If UUIDs differ during a merge,
                 * resolve the conflict by generating a new ID!
                 */
                SERIALIZED_UUID = UUID.fromString("33761B2D-78BB-4A43-8B0B-4F5BEE8AACF3");
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
        }
```

```
        static {
                /* This value should never change. Updates following this version are
                 * reflected as change in the unique ID SERIALIZED_UUID.
                 */
                SERIALIZED_VERSION = 3;
        }

        public static final UUID SERIALIZED_UUID;
        static {
                /* WARNING: DO NOT MERGE THIS LINE. If UUIDs differ during a merge,
                 * resolve the conflict by generating a new ID!
                 */
                SERIALIZED_UUID = UUID.fromString("33761B2D-78BB-4A43-8B0B-4F5BEE8AACF3");
        }
```

Chunk 07: (combination/ switch case, variable)

```
                                return pt;
<<<<<<< HEAD
                        case Transition.PRECEDENCE:
                                return new PrecedencePredicateTransition(target, arg1);
                        case Transition.ATOM : return new AtomTransition(target, arg1);
=======
                        case Transition.ATOM :
                                if (arg3 != 0) {
                                        return new AtomTransition(target, Token.EOF);
                                }
                                else {
                                        return new AtomTransition(target, arg1);
                                }
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
                        case Transition.ACTION :
                                return pt;
                        case Transition.PRECEDENCE:
                                return new PrecedencePredicateTransition(target, arg1);
                        case Transition.ATOM :
                                if (arg3 != 0) {
                                        return new AtomTransition(target, Token.EOF);
                                }
                                else {
                                        return new AtomTransition(target, arg1);
                                }
                        case Transition.ACTION :
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 08: (new code/ annotation, method declaration, method signature)

```
        @Nullable
<<<<<<< HEAD
        public ATNConfig precedenceTransition(@NotNull ATNConfig config,
                                                                    @NotNull
PrecedencePredicateTransition pt,
                                                                    boolean
collectPredicates,
                                                                    boolean inContext,
                                                                    boolean fullCtx)
        {
                if ( debug ) {
                        System.out.println("PRED (collectPredicates="+collectPredicates+") "+
                     pt.precedence+">=_p"+
                                 ", ctx dependent=true");
                        if ( parser != null ) {
                 System.out.println("context surrounding pred is "+
                                   parser.getRuleInvocationStack());
             }
                }

        ATNConfig c = null;
        if (collectPredicates && inContext) {
                        if ( fullCtx ) {
                                // In full context mode, we can evaluate predicates on-the-fly
                                // during closure, which dramatically reduces the size of
                                //  the  config  sets.  It  also  obviates  the  need  to  test
predicates
                                // later during conflict resolution.
                                int currentPosition = _input.index();
                                _input.seek(_startIndex);
                                boolean    predSucceeds    =    pt.getPredicate().eval(parser,
_outerContext);
                                _input.seek(currentPosition);
                                if ( predSucceeds ) {
                                        c  =  new  ATNConfig(config,  pt.target);  //  no  pred
context
                                }
                        }
                        else {
                                SemanticContext newSemCtx =
                                        SemanticContext.and(config.semanticContext,
pt.getPredicate());
                                c = new ATNConfig(config, pt.target, newSemCtx);
                        }
        }
               else {
                      c = new ATNConfig(config, pt.target);
               }

               if ( debug ) System.out.println("config from pred transition="+c);
        return c;
        }

        @Nullable
        public ATNConfig predTransition(@NotNull ATNConfig config,
=======
        protected ATNConfig predTransition(@NotNull ATNConfig config,
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
                                                                  @NotNull
PredicateTransition pt,
```

```
        @Nullable
        protected ATNConfig predTransition(@NotNull ATNConfig config,
                                                                  @NotNull
PredicateTransition pt,
                                                                  boolean
collectPredicates,
                                                                  boolean inContext,
                                                                  boolean fullCtx)
        {
               if ( debug ) {
                      System.out.println("PRED (collectPredicates="+collectPredicates+") "+
                    pt.ruleIndex+":"+pt.predIndex+
                                    ", ctx dependent="+pt.isCtxDependent);
                      if ( parser != null ) {
                System.out.println("context surrounding pred is "+
                                  parser.getRuleInvocationStack());
            }
               }

               ATNConfig c = null;
               if ( collectPredicates &&
                        (!pt.isCtxDependent || (pt.isCtxDependent&&inContext)) )
               {
                      if ( fullCtx ) {
                             // In full context mode, we can evaluate predicates on-the-fly
                             // during closure, which dramatically reduces the size of
                             //  the  config  sets.  It  also  obviates  the  need  to  test
predicates
                             // later during conflict resolution.
                             int currentPosition = _input.index();
                             _input.seek(_startIndex);
                             boolean    predSucceeds    =    pt.getPredicate().eval(parser,
_outerContext);
                             _input.seek(currentPosition);
                             if ( predSucceeds ) {
                                    c  =  new  ATNConfig(config,  pt.target);  //  no  pred
context
                             }
                      }
                      else {
                             SemanticContext newSemCtx =
                                    SemanticContext.and(config.semanticContext,
pt.getPredicate());
                             c = new ATNConfig(config, pt.target, newSemCtx);
                        }
                }
                else {
                        c = new ATNConfig(config, pt.target);
                }

                if ( debug ) System.out.println("config from pred transition="+c);
        return c;
        }
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/RuleStartState.java

Chunk 09: (concatenation/annotation, method signature, variable)

```
        public RuleStopState stopState;
<<<<<<< HEAD
        public boolean isPrecedenceRule;
=======

        @Override
        public int getStateType() {
                return RULE_START;
        }
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
}
```

```
        public RuleStopState stopState;
        public boolean isPrecedenceRule;

        @Override
        public int getStateType() {
                return RULE_START;
        }
}
```

antlr4/tool/src/org/antlr/v4/analysis/LeftRecursiveRuleAnalyzer.java

Chunk 10: (new code/commentary, for statement, if statement, method invocation, variable)

```
                if ( t==null ) return null;
<<<<<<< HEAD
                for (GrammarAST rref : t.getNodesWithType(RULE_REF)) {
                        if ( rref.getText().equals(ruleName) ) {

        rref.setText(ruleName+"<"+LeftRecursiveRuleTransformer.PRECEDENCE_OPTION_NAME+"="+pr
ec+">");
=======
                // get all top-level rule refs from ALT
                List<GrammarAST>                      outerAltRuleRefs                      =
t.getNodesWithTypePreorderDFS(IntervalSet.of(RULE_REF));
                for (GrammarAST rref : outerAltRuleRefs) {
                        boolean recursive = rref.getText().equals(ruleName);
                        boolean          rightmost           =           rref           ==
outerAltRuleRefs.get(outerAltRuleRefs.size()-1);
                        if ( recursive && rightmost ) {
                                rref.setText(ruleName+"["+prec+"]");
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
                        }
                if ( t==null ) return null;
                // get all top-level rule refs from ALT
                List<GrammarAST>                      outerAltRuleRefs                    =
t.getNodesWithTypePreorderDFS(IntervalSet.of(RULE_REF));
                for (GrammarAST rref : outerAltRuleRefs) {
                        boolean recursive = rref.getText().equals(ruleName);
                        boolean            rightmost           =           rref            ==
outerAltRuleRefs.get(outerAltRuleRefs.size()-1);
                        if ( recursive && rightmost ) {

        rref.setText(ruleName+"<"+LeftRecursiveRuleTransformer.PRECEDENCE_OPTION_NAME+"="+pr
ec+">");
                        }
```

Chunk 11: (version 2/ method declaration)

```
        }

<<<<<<< HEAD
        public AltAST addPrecedenceArgToLastRule(AltAST t, int prec) {
                if ( t==null ) return null;
                GrammarAST last = null;
                for (GrammarAST rref : t.getNodesWithType(RULE_REF)) { last = rref; }
                if ( last !=null && last.getText().equals(ruleName) ) {

        last.setText(ruleName+"<"+LeftRecursiveRuleTransformer.PRECEDENCE_OPTION_NAME+"="+pr
ec+">");
                }
                return t;
        }

=======
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
        public void stripAssocOptions(GrammarAST t) {
```

```
        }

        public void stripAssocOptions(GrammarAST t) {
```

antlr4/tool/src/org/antlr/v4/automata/ParserATNFactory.java

Chunk 12: (concatenation/import declaration)

```
import org.antlr.v4.runtime.atn.ATNState;
<<<<<<< HEAD
import org.antlr.v4.runtime.atn.AbstractPredicateTransition;
=======
import org.antlr.v4.runtime.atn.ATNType;
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
import org.antlr.v4.runtime.atn.ActionTransition;
```

```
import org.antlr.v4.runtime.atn.ATNState;
import org.antlr.v4.runtime.atn.ATNType;
import org.antlr.v4.runtime.atn.AbstractPredicateTransition;
import org.antlr.v4.runtime.atn.ActionTransition;
```

Chunk 13: (concatenation/import declaration)

```
import org.antlr.v4.tool.LeftRecursiveRule;
<<<<<<< HEAD
=======
import org.antlr.v4.tool.LexerGrammar;
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
import org.antlr.v4.tool.Rule;
```

```
import org.antlr.v4.tool.LeftRecursiveRule;
import org.antlr.v4.tool.LexerGrammar;
import org.antlr.v4.tool.Rule;
```

antlr4/tool/src/org/antlr/v4/semantics/SymbolChecks.java

Chunk 14: (concatenation/import declaration)

```
package org.antlr.v4.semantics;

<<<<<<< HEAD
import org.antlr.v4.parse.ANTLRParser;
=======
import org.antlr.v4.runtime.misc.NotNull;
import org.antlr.v4.runtime.misc.Nullable;
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18
import org.antlr.v4.tool.Alternative;
```

```
package org.antlr.v4.semantics;

import org.antlr.v4.parse.ANTLRParser;
import org.antlr.v4.runtime.misc.NotNull;
import org.antlr.v4.runtime.misc.Nullable;
import org.antlr.v4.tool.Alternative;
```

antlr4/tool/src/org/antlr/v4/tool/Grammar.java

Chunk 15: (version 2/ method invocation, static block, variable)

```
	/** Legal options for terminal refs like ID<assoc=right> */
<<<<<<< HEAD
	public static final Set<String> tokenOptions = new HashSet<String>() {{
		add("assoc");
	}};

	public static final Set<String> actionOptions = new HashSet<String>() {{
	}};

	public static final Set<String> semPredOptions = new HashSet<String>() {{
		add(LeftRecursiveRuleTransformer.PRECEDENCE_OPTION_NAME);
		add("fail");
	}};

	public static final Set doNotCopyOptionsToLexer =
	 new HashSet() {{
				add("superClass");
		 add("TokenLabelType");
				add("tokenVocab");
	 }};

    public static Map<String, AttributeDict> grammarAndLabelRefTypeToScope =
        new HashMap<String, AttributeDict>() {{
            put("parser:RULE_LABEL", Rule.predefinedRulePropertiesDict);
            put("parser:TOKEN_LABEL", AttributeDict.predefinedTokenDict);
            put("combined:RULE_LABEL", Rule.predefinedRulePropertiesDict);
            put("combined:TOKEN_LABEL", AttributeDict.predefinedTokenDict);
		}};
=======
	public static final Set<String> tokenOptions = new HashSet<String>();
	static {
		tokenOptions.add("assoc");
	}

	public static final Set<String> actionOptions = new HashSet<String>();

	public static final Set<String> semPredOptions = new HashSet<String>();
	static {
		semPredOptions.add("fail");
	}

	public static final Set<String> doNotCopyOptionsToLexer = new HashSet<String>();
	static {
		doNotCopyOptionsToLexer.add("superClass");
		doNotCopyOptionsToLexer.add("TokenLabelType");
		doNotCopyOptionsToLexer.add("tokenVocab");
	}

	public static final Map<String, AttributeDict> grammarAndLabelRefTypeToScope =
		new HashMap<String, AttributeDict>();
	static {
		grammarAndLabelRefTypeToScope.put("parser:RULE_LABEL",
Rule.predefinedRulePropertiesDict);
		grammarAndLabelRefTypeToScope.put("parser:TOKEN_LABEL",
AttributeDict.predefinedTokenDict);
		grammarAndLabelRefTypeToScope.put("combined:RULE_LABEL",
Rule.predefinedRulePropertiesDict);
		grammarAndLabelRefTypeToScope.put("combined:TOKEN_LABEL",
AttributeDict.predefinedTokenDict);
	}
>>>>>>> b80ad5052d1b693be6e5c0a2be8bf87e15b86f18

	public String name;
```

```
	/** Legal options for terminal refs like ID<assoc=right> */
	public static final Set<String> tokenOptions = new HashSet<String>();
	static {
		tokenOptions.add("assoc");
	}

	public static final Set<String> actionOptions = new HashSet<String>();

	public static final Set<String> semPredOptions = new HashSet<String>();
	static {
		semPredOptions.add(LeftRecursiveRuleTransformer.PRECEDENCE_OPTION_NAME);
		semPredOptions.add("fail");
	}

	public static final Set<String> doNotCopyOptionsToLexer = new HashSet<String>();
	static {
		doNotCopyOptionsToLexer.add("superClass");
		doNotCopyOptionsToLexer.add("TokenLabelType");
		doNotCopyOptionsToLexer.add("tokenVocab");
	}

	public static final Map<String, AttributeDict> grammarAndLabelRefTypeToScope =
		new HashMap<String, AttributeDict>();
	static {
```

```
               grammarAndLabelRefTypeToScope.put("parser:RULE_LABEL",
Rule.predefinedRulePropertiesDict);
               grammarAndLabelRefTypeToScope.put("parser:TOKEN_LABEL",
AttributeDict.predefinedTokenDict);
               grammarAndLabelRefTypeToScope.put("combined:RULE_LABEL",
Rule.predefinedRulePropertiesDict);
               grammarAndLabelRefTypeToScope.put("combined:TOKEN_LABEL",
AttributeDict.predefinedTokenDict);
        }

        public String name;
```

Version: c3af4e9b7b231a00fd37a253e97d66443539b508

Parents:

ca213689619bd108f2fd3863676ea4400f2c220e

84324f1dad2594eeb658c07307dd2b1c8231e97c

Merge base:

eeda06b698af194c2684004ea810b82595474ac1

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/LL1Analyzer.java

Chunk 16: (new code/method invocation)

```
                                             calledRuleStack.clear(returnState.ruleIndex);
<<<<<<< HEAD
                                             _LOOK(returnState,  stopState,  p.parent,  look,
lookBusy, calledRuleStack, seeThruPreds, addEOF);
=======
                                             _LOOK(returnState,   ctx.getParent(i),   look,
lookBusy, calledRuleStack, seeThruPreds, addEOF);
>>>>>>> 84324f1dad2594eeb658c07307dd2b1c8231e97c
                                      }
```

```
                                             calledRuleStack.clear(returnState.ruleIndex);
                                             _LOOK(returnState, stopState, ctx.getParent(i),
look, lookBusy, calledRuleStack, seeThruPreds, addEOF);
                                      }
```

Version: e5e4402ea9cf2901d34539991d079e1ea7baef45

Parents:

b822070790d7978faaefd79ec226f5d1385ebd35

2673e08bfc1cbdfb79f600506d1c1e9e1a145799

Merge base:

9539572ee7155ca403d0cb6bf9ee0d74fee4d0c1

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 17: (version 1/ if statement, method invocation, variable)

```
                        }

<<<<<<< HEAD
                        if ( D.isAcceptState && D.configs.hasSemanticContext ) {
                                predicateDFAState(D, decState);

                                if ( D.predicates!=null ) {
=======
                        if ( D.isAcceptState && D.configset.hasSemanticContext ) {
                                int nalts = decState.getNumberOfTransitions();
                                DFAState.PredPrediction[] predPredictions =
                                        predicateDFAState(D,     D.configset,     outerContext,
nalts);
                                if ( predPredictions!=null ) {
>>>>>>> 2673e08bfc1cbdfb79f600506d1c1e9e1a145799
                                        int stopIndex = input.index();
```

```
                        }

                        if ( D.isAcceptState && D.configs.hasSemanticContext ) {
                                predicateDFAState(D, decState);

                                if ( D.predicates!=null ) {
                                        int stopIndex = input.index();
```

Chunk 18: (version 1/commentary, If statement, Method invocation, return statement, switch case, variable)

```
                        reportContextSensitivity(dfa, reach, startIndex, input.index());
<<<<<<< HEAD
                        if ( predictedAlt == SLL_min_alt ) {
                                retry_with_context_predicts_same_alt++;
=======
                        return reach;
                }

                if ( reach.hasSemanticContext ) {
                        SemanticContext[]                      altToPred                          =
getPredsForAmbigAlts(reach.conflictingAlts, reach, nalts);
                        // altToPred[uniqueAlt] is now our validating predicate (if any)
                        DFAState.PredPrediction[] predPredictions;
                        if ( altToPred!=null ) {
                                // we have a validating predicate; test it
                                predPredictions                                           =
getPredicatePredictions(reach.conflictingAlts, altToPred);
                                input.seek(startIndex);
```

```
                              IntervalSet   alts   =   evalSemanticContext(predPredictions,
outerContext, reportAmbiguities);
                              reach.uniqueAlt = ATN.INVALID_ALT_NUMBER;
                              switch (alts.size()) {
                              case 0:
                                     throw   noViableAlt(input,   outerContext,   reach,
startIndex);

                              case 1:
                                     reach.uniqueAlt = alts.getMinElement();
                                     return reach;

                              default:
                                     // reach.conflictingAlts holds the post-evaluation set
of ambig alts
                                     reach.conflictingAlts = alts;
                                     break;
                              }
>>>>>>> 2673e08bfc1cbdfb79f600506d1c1e9e1a145799
                       }
```

```
                       reportContextSensitivity(dfa, reach, startIndex, input.index());
                       if ( predictedAlt == SLL_min_alt ) {
                              retry_with_context_predicts_same_alt++;
                       }
```

Chunk 19: (version 1/ commentary, method declaration, return statement)

```
               }

<<<<<<< HEAD
               return null;
=======
        /** collect and set D's semantic context */
        public DFAState.PredPrediction[] predicateDFAState(DFAState D,

           ATNConfigSet configs,

           RuleContext outerContext,

           int nalts)
        {
               IntervalSet conflictingAlts = getConflictingAltsFromConfigSet(configs);
               if ( debug ) System.out.println("predicateDFAState "+D);
               SemanticContext[] altToPred = getPredsForAmbigAlts(conflictingAlts, configs,
nalts);
               // altToPred[uniqueAlt] is now our validating predicate (if any)
               DFAState.PredPrediction[] predPredictions = null;
               if ( altToPred!=null ) {
                       // we have a validating predicate; test it
                       // Update DFA so reach becomes accept state with predicate
                       predPredictions      =      getPredicatePredictions(conflictingAlts,
altToPred);
                       D.predicates = predPredictions;
                       D.prediction = ATN.INVALID_ALT_NUMBER; // make sure we use preds
               }
               return predPredictions;
>>>>>>> 2673e08bfc1cbdfb79f600506d1c1e9e1a145799
        }
```

```
               }
```

```
                return null;
        }
```

Chunk 20: (version 1/ method signature)

```
        }

<<<<<<< HEAD
        public List<DFAState.PredPrediction> getPredicatePredictions(BitSet ambigAlts,

                                    SemanticContext[] altToPred)
        {
=======
        public   DFAState.PredPrediction[]   getPredicatePredictions(IntervalSet   ambigAlts,
SemanticContext[] altToPred) {
>>>>>>> 2673e08bfc1cbdfb79f600506d1c1e9e1a145799
                List<DFAState.PredPrediction>             pairs              =             new
ArrayList<DFAState.PredPrediction>();
```

```
        }

        public DFAState.PredPrediction[] getPredicatePredictions(BitSet ambigAlts,

                            SemanticContext[] altToPred)
        {
                List<DFAState.PredPrediction>             pairs              =             new
ArrayList<DFAState.PredPrediction>();
```

Case 21: (new code/ method signature)

```
         */
<<<<<<< HEAD
        public BitSet evalSemanticContext(List<DFAState.PredPrediction> predPredictions,
                                                                      ParserRuleContext
outerContext,
                                                                      boolean complete)
=======
        public     IntervalSet     evalSemanticContext(@NotNull     DFAState.PredPrediction[]
predPredictions,

ParserRuleContext<?> outerContext,
                                                                              boolean
complete)
>>>>>>> 2673e08bfc1cbdfb79f600506d1c1e9e1a145799
        {
```

```
         */
        public      BitSet      evalSemanticContext(@NotNull      DFAState.PredPrediction[]
predPredictions,
                                                                      ParserRuleContext
outerContext,
                                                                      boolean complete)
        {
```

Version: 18f5354d1b956733dabc3225c8dda719ce41291e

Parents:

dd0944b9c469608bb13d16620ff79a78728d1c1f

fdf3a86969bf684c4decff0efb4da37aa94d3b35

Merge base:

ea7037dd2dff6c36b358b0a641c197d499367c0c

antlr4/runtime/Java/src/org/antlr/v4/runtime/BufferedTokenStream.java

Chunk 22: (version 2/Import declaration)

```
package org.antlr.v4.runtime;

<<<<<<< HEAD
import org.antlr.v4.runtime.misc.NotNull;

import java.util.*;
=======
import org.antlr.v4.runtime.misc.Interval;
import org.antlr.v4.runtime.misc.NotNull;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35

/** Buffer all input tokens but do on-demand fetching of new tokens from
```

```
package org.antlr.v4.runtime;

import org.antlr.v4.runtime.misc.Interval;
import org.antlr.v4.runtime.misc.NotNull;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

/** Buffer all input tokens but do on-demand fetching of new tokens from
```

Chunk 23: (version 1/class signature)

```
*/
<<<<<<< HEAD
public class BufferedTokenStream<T extends Token> implements TokenStream {
	@NotNull
=======
public class BufferedTokenStream implements TokenStream {
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
    protected TokenSource tokenSource;
```

```
*/
public class BufferedTokenStream implements TokenStream {
	@NotNull
    protected TokenSource tokenSource;
```

Chunk 24: (combination/commentary, for statement, if statement, method invocation, method signature, variable)

```
    }

<<<<<<< HEAD
    /** Add {@code n} elements to buffer.
         *
         * @return The actual number of elements added to the buffer.
         */
    protected int fetch(int n) {
                if (fetchedEOF) {
                        return 0;
                }

        for (int i = 0; i < n; i++) {
            T t = (T)tokenSource.nextToken();
=======
    /** add n elements to buffer */
    protected void fetch(int n) {
        for (int i=1; i<=n; i++) {
            Token t = tokenSource.nextToken();
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
            if ( t instanceof WritableToken ) {
```

```
    }

    /** Add {@code n} elements to buffer.
         *
         * @return The actual number of elements added to the buffer.
         */
    protected int fetch(int n) {
                if (fetchedEOF) {
                        return 0;
                }

        for (int i = 0; i < n; i++) {
            Token t = tokenSource.nextToken();
            if ( t instanceof WritableToken ) {
```

Chunk 25: Manual (combination/method invocation, variable)

```
                if ( start<0 || stop<0 ) return null;
<<<<<<< HEAD
                lazyInit();
                List<T> subset = new ArrayList<T>();
=======
                if ( p == -1 ) setup();
                List<Token> subset = new ArrayList<Token>();
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
                if ( stop>=tokens.size() ) stop = tokens.size()-1;
```

```
                if ( start<0 || stop<0 ) return null;
                lazyInit();
                List<Token> subset = new ArrayList<Token>();
                if ( stop>=tokens.size() ) stop = tokens.size()
```

Chunk 26: (Combination/ Method declaration)

```
    @Override
<<<<<<< HEAD
```

```
    public T LT(int k) {
        lazyInit();
=======
    public Token LT(int k) {
        if ( p == -1 ) setup();
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
        if ( k==0 ) return null;
```

```
    @Override
    public Token LT(int k) {
        lazyInit();
        if ( k==0 ) return null;
```

Chunk 27: (Combination/ if statement, method invocation, method signature, throw statement)

```
     */
<<<<<<< HEAD
    public List<T> getTokens(int start, int stop, Set<Integer> types) {
        lazyInit();
        if ( stop>=tokens.size() ) stop=tokens.size()-1;
        if ( start<0 ) start=0;
=======
    public List<Token> getTokens(int start, int stop, Set<Integer> types) {
        if ( p == -1 ) setup();
               if ( start<0 || stop>=tokens.size() ||
                        stop<0  || start>=tokens.size() )
               {
                       throw new IndexOutOfBoundsException("start "+start+" or stop "+stop+
                                                                                            "
not in 0.."+(tokens.size()-1));
               }
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
        if ( start>stop ) return null;
```

```
     */
    public List<Token> getTokens(int start, int stop, Set<Integer> types) {
        lazyInit();
               if ( start<0 || stop>=tokens.size() ||
                        stop<0  || start>=tokens.size() )
               {
                       throw new IndexOutOfBoundsException("start "+start+" or stop "+stop+
                                                                                            "
not in 0.."+(tokens.size()-1));
               }
        if ( start>stop ) return null;
```

Chunk 28: (combination/annotation, commentary, method declaration)

```
    public String getSourceName() {   return tokenSource.getSourceName();   }

<<<<<<< HEAD
    /** Grab *all* tokens from stream and return string */
    @Override
    public String toString() {
        lazyInit();
        fill();
        return toString(0, tokens.size()-1);
    }
=======
       /** Get the text of all tokens in this buffer. */
```

```
        @NotNull
        @Override
        public String getText() {
                if ( p == -1 ) setup();
                fill();
                return getText(Interval.of(0,size()-1));
        }
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35

        @NotNull
```

```
    public String getSourceName() {   return tokenSource.getSourceName();   }

        /** Get the text of all tokens in this buffer. */
        @NotNull
        @Override
        public String getText() {
         lazyInit();
                fill();
                return getText(Interval.of(0,size()-1));
        }

        @NotNull
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/CommonTokenStream.java

Chunk 29: (new code / method declararion)

```
        @Override
<<<<<<< HEAD
        protected int adjustSeekIndex(int i) {
                return skipOffTokenChannels(i);
=======
        public void reset() {
                super.reset();
                p = nextTokenOnChannel(p, channel);
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
        }
```

```
        @Override
        protected int adjustSeekIndex(int i) {
                return nextTokenOnChannel(i, channel);
        }
```

Chunk 30: (new code, commentary, if statement, variable)

```
         while ( n<k ) {
<<<<<<< HEAD
            // skip off-channel tokens, but make sure to not look past EOF
                      if (sync(i + 1)) {
                              i = skipOffTokenChannels(i+1);
                      }
=======
            // skip off-channel tokens
            i = nextTokenOnChannel(i + 1, channel);
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
            n++;
```

```
         while ( n<=k ) {
            // skip off-channel tokens
            i = previousTokenOnChannel(i - 1, channel);
            n++;
        }
```

Chunk 31: (new code/Annotation, commentary, method declaration, method invocation, variable)

```
    }

<<<<<<< HEAD
    /** Given a starting index, return the index of the first on-channel
     *  token.
     */
    protected int skipOffTokenChannels(int i) {
        sync(i);
=======
    @Override
    protected void setup() {
        p = 0;
        sync(0);
        int i = 0;
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
        Token token = tokens.get(i);
        while ( token.getType()!=Token.EOF && token.getChannel()!=channel ) {
            i++;
            sync(i);
            token = tokens.get(i);
        }
<<<<<<< HEAD
        return i;
    }

    protected int skipOffTokenChannelsReverse(int i) {
        while ( i>=0 && tokens.get(i).getChannel()!=channel ) {
            i--;
        }
        return i;
=======
        p = i;
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
    }

        /** Count EOF just once. */
        public int getNumberOfOnChannelTokens() {
                int n = 0;
                fill();
                for (int i = 0; i < tokens.size(); i++) {
                        Token t = tokens.get(i);
                        if ( t.getChannel()==channel ) n++;
                        if ( t.getType()==Token.EOF ) break;
                }
                return n;
        }
}
```

```
    }

    @Override
    public Token LT(int k) {
        //System.out.println("enter LT("+k+")");
        lazyInit();
        if ( k == 0 ) return null;
        if ( k < 0 ) return LB(-k);
        int i = p;
        int n = 1; // we know tokens[p] is a good one
        // find k good tokens
        while ( n<k ) {
            // skip off-channel tokens, but make sure to not look past EOF
                        if (sync(i + 1)) {
                                i = nextTokenOnChannel(i + 1, channel);
                        }
            n++;
        }
//              if ( i>range ) range = i;
        return tokens.get(i);
    }

        /** Count EOF just once. */
        public int getNumberOfOnChannelTokens() {
                int n = 0;
                fill();
                for (int i = 0; i < tokens.size(); i++) {
                        Token t = tokens.get(i);
                        if ( t.getChannel()==channel ) n++;
                        if ( t.getType()==Token.EOF ) break;
                }
                return n;
        }
}
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/TokenStreamRewriter.java

Chunk 32: (version 2/ method signature, variable)

```
        /** Map String (program name) -> Integer index */
<<<<<<< HEAD:runtime/Java/src/org/antlr/v4/runtime/TokenRewriteStream.java
        protected Map<String, Integer> lastRewriteTokenIndexes = null;

        protected void init() {
=======
        protected final Map<String, Integer> lastRewriteTokenIndexes;

        public TokenStreamRewriter(TokenStream tokens) {
                this.tokens = tokens;
>>>>>>>
fdf3a86969bf684c4decff0efb4da37aa94d3b35:runtime/Java/src/org/antlr/v4/runtime/TokenStreamRe
writer.java
                programs = new HashMap<String, List<RewriteOperation>>();
```

```
        /** Map String (program name) -> Integer index */
        protected final Map<String, Integer> lastRewriteTokenIndexes;

        public TokenStreamRewriter(TokenStream tokens) {
                this.tokens = tokens;
                programs = new HashMap<String, List<RewriteOperation>>();
```

antlr4/tool/test/org/antlr/v4/test/TestCommonTokenStream.java

Chunk 33: (version 2/ annotation, method declaration)

```
    }
```

```
<<<<<<< HEAD
	@Test
	public void testSingleEOF() throws Exception {
		TokenSource lexer = new TokenSource() {

			@Override
			public Token nextToken() {
				return new CommonToken(Token.EOF);
			}

			@Override
			public int getLine() {
				return 0;
			}

			@Override
			public int getCharPositionInLine() {
				return 0;
			}

			@Override
			public CharStream getInputStream() {
				return null;
			}

			@Override
			public String getSourceName() {
				return null;
			}

			@Override
			public void setTokenFactory(TokenFactory<?> factory) {
				throw new UnsupportedOperationException("Not supported yet.");
			}
		};

		CommonTokenStream tokens = new CommonTokenStream(lexer);
		tokens.fill();

		assertEquals(Token.EOF, tokens.LA(1));
		assertEquals(0, tokens.index());
		assertEquals(1, tokens.size());
		tokens.consume();

		assertEquals(Token.EOF, tokens.LA(1));
		assertEquals(0, tokens.index());
		assertEquals(1, tokens.size());
		tokens.consume();

		assertEquals(Token.EOF, tokens.LA(1));
		assertEquals(0, tokens.index());
		assertEquals(1, tokens.size());
		tokens.consume();
	}

=======
	@Test public void testFetchOffChannel() throws Exception {
		TokenSource lexer = // simulate input " x =34  ; \n"
		                    // token indexes   01234 56789
			new TokenSource() {
				int i = 0;
				WritableToken[] tokens = {
```

```
                        new CommonToken(1," ") {{channel = Lexer.HIDDEN;}}, // 0
                        new CommonToken(1,"x"),
                // 1
                        new CommonToken(1," ") {{channel = Lexer.HIDDEN;}},  // 2
                        new CommonToken(1,"="),
                // 3
                        new CommonToken(1,"34"),
        // 4
                        new CommonToken(1," ") {{channel = Lexer.HIDDEN;}},  // 5
                        new CommonToken(1," ") {{channel = Lexer.HIDDEN;}}, // 6
                        new CommonToken(1,";"),
                // 7
                        new CommonToken(1," ")  {{channel = Lexer.HIDDEN;}},// 8
                        new CommonToken(1,"\n") {{channel = Lexer.HIDDEN;}},// 9
                        new CommonToken(Token.EOF,"")
// 10
                        };
                        @Override
                        public Token nextToken() {
                                return tokens[i++];
                        }
                        @Override
                        public String getSourceName() { return "test"; }
                        @Override
                        public int getCharPositionInLine() {
                                return 0;
                        }
                        @Override
                        public int getLine() {
                                return 0;
                        }
                        @Override
                        public CharStream getInputStream() {
                                return null;
                        }

                        @Override
                        public void setTokenFactory(TokenFactory<?> factory) {
                        }

                        @Override
                        public TokenFactory<?> getTokenFactory() {
                                return null;
                        }
                };

        CommonTokenStream tokens = new CommonTokenStream(lexer);
        tokens.fill();
        assertEquals(null, tokens.getHiddenTokensToLeft(0));
        assertEquals(null, tokens.getHiddenTokensToRight(0));

        assertEquals("[[@0,0:0=' ',<1>,channel=1,0:-1]]",
                                tokens.getHiddenTokensToLeft(1).toString());
        assertEquals("[[@2,0:0=' ',<1>,channel=1,0:-1]]",
                                tokens.getHiddenTokensToRight(1).toString());

        assertEquals(null, tokens.getHiddenTokensToLeft(2));
        assertEquals(null, tokens.getHiddenTokensToRight(2));

        assertEquals("[[@2,0:0=' ',<1>,channel=1,0:-1]]",
                                tokens.getHiddenTokensToLeft(3).toString());
        assertEquals(null, tokens.getHiddenTokensToRight(3));
```

```
                assertEquals(null, tokens.getHiddenTokensToLeft(4));
                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1], [@6,0:0=' ',<1>,channel=1,0:-
1]]",
                                        tokens.getHiddenTokensToRight(4).toString());

                assertEquals(null, tokens.getHiddenTokensToLeft(5));
                assertEquals("[[@6,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(5).toString());

                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToLeft(6).toString());
                assertEquals(null, tokens.getHiddenTokensToRight(6));

                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1], [@6,0:0=' ',<1>,channel=1,0:-
1]]",
                                        tokens.getHiddenTokensToLeft(7).toString());
                assertEquals("[[@8,0:0='                              ',<1>,channel=1,0:-1],
[@9,0:0='\\n',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(7).toString());

                assertEquals(null, tokens.getHiddenTokensToLeft(8));
                assertEquals("[[@9,0:0='\\n',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(8).toString());

                assertEquals("[[@8,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToLeft(9).toString());
                assertEquals(null, tokens.getHiddenTokensToRight(9));
        }
>>>>>>> fdf3a86969bf684c4decff0efb4da37aa94d3b35
}
```

```
        @Test public void testFetchOffChannel() throws Exception {
                TokenSource lexer = // simulate input " x =34  ; \n"
                                    // token indexes   01234 56789
                        new TokenSource() {
                                int i = 0;
                                WritableToken[] tokens = {
                                new CommonToken(1," ") {{channel = Lexer.HIDDEN;}}, // 0
                                new CommonToken(1,"x"),
                        // 1
                                new CommonToken(1," ") {{channel = Lexer.HIDDEN;}},  // 2
                                new CommonToken(1,"="),
                        // 3
                                new CommonToken(1,"34"),
                // 4
                                new CommonToken(1," ") {{channel = Lexer.HIDDEN;}},  // 5
                                new CommonToken(1," ") {{channel = Lexer.HIDDEN;}}, // 6
                                new CommonToken(1,";"),
                        // 7
                                new CommonToken(1," ")  {{channel = Lexer.HIDDEN;}},// 8
                                new CommonToken(1,"\n") {{channel = Lexer.HIDDEN;}},// 9
                                new CommonToken(Token.EOF,"")
        // 10
                                };
                                @Override
                                public Token nextToken() {
                                        return tokens[i++];
                                }
                                @Override
                                public String getSourceName() { return "test"; }
                                @Override
                                public int getCharPositionInLine() {
                                        return 0;
                                }
                                @Override
                                public int getLine() {
                                        return 0;
                                }
                                @Override
                                public CharStream getInputStream() {
                                        return null;
                                }

                                @Override
                                public void setTokenFactory(TokenFactory<?> factory) {
                                }

                                @Override
                                public TokenFactory<?> getTokenFactory() {
                                        return null;
                                }
                        };

                CommonTokenStream tokens = new CommonTokenStream(lexer);
                tokens.fill();
                assertEquals(null, tokens.getHiddenTokensToLeft(0));
                assertEquals(null, tokens.getHiddenTokensToRight(0));

                assertEquals("[[@0,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToLeft(1).toString());
                assertEquals("[[@2,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(1).toString());

                assertEquals(null, tokens.getHiddenTokensToLeft(2));
                assertEquals(null, tokens.getHiddenTokensToRight(2));

                assertEquals("[[@2,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToLeft(3).toString());
                assertEquals(null, tokens.getHiddenTokensToRight(3));

                assertEquals(null, tokens.getHiddenTokensToLeft(4));
                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1], [@6,0:0=' ',<1>,channel=1,0:-
1]]",
                                        tokens.getHiddenTokensToRight(4).toString());

                assertEquals(null, tokens.getHiddenTokensToLeft(5));
                assertEquals("[[@6,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(5).toString());

                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToLeft(6).toString());
                assertEquals(null, tokens.getHiddenTokensToRight(6));

                assertEquals("[[@5,0:0=' ',<1>,channel=1,0:-1], [@6,0:0=' ',<1>,channel=1,0:-
1]]",
                                        tokens.getHiddenTokensToLeft(7).toString());
                assertEquals("[[@8,0:0='                               ',<1>,channel=1,0:-1],
[@9,0:0='\\n',<1>,channel=1,0:-1]]",
                                        tokens.getHiddenTokensToRight(7).toString());

                assertEquals(null, tokens.getHiddenTokensToLeft(8));
                assertEquals("[[@9,0:0='\\n',<1>,channel=1,0:-1]]",
                                tokens.getHiddenTokensToRight(8).toString());

        assertEquals("[[@8,0:0=' ',<1>,channel=1,0:-1]]",
                                tokens.getHiddenTokensToLeft(9).toString());
        assertEquals(null, tokens.getHiddenTokensToRight(9));
}

@Test
public void testSingleEOF() throws Exception {
        TokenSource lexer = new TokenSource() {

                @Override
                public Token nextToken() {
                        return new CommonToken(Token.EOF);
                }

                @Override
                public int getLine() {
                        return 0;
                }

                @Override
                public int getCharPositionInLine() {
                        return 0;
                }

                @Override
                public CharStream getInputStream() {
                        return null;
                }

                @Override
                public String getSourceName() {
                        return null;
                }

                @Override
                public TokenFactory<?> getTokenFactory() {
                        throw new UnsupportedOperationException("Not supported yet.");
                }

                @Override
                public void setTokenFactory(TokenFactory<?> factory) {
                        throw new UnsupportedOperationException("Not supported yet.");
                }
        };

        CommonTokenStream tokens = new CommonTokenStream(lexer);
        tokens.fill();

        assertEquals(Token.EOF, tokens.LA(1));
        assertEquals(0, tokens.index());
        assertEquals(1, tokens.size());
        tokens.consume();

        assertEquals(Token.EOF, tokens.LA(1));
        assertEquals(0, tokens.index());
        assertEquals(1, tokens.size());
        tokens.consume();

        assertEquals(Token.EOF, tokens.LA(1));
        assertEquals(0, tokens.index());
```

```
                assertEquals(1, tokens.size());
                tokens.consume();
        }
}
```

Version: 92ae0f0fa66bf2fb09c094d5a223c42455da8c65

Parents:

fdf3a86969bf684c4decff0efb4da37aa94d3b35

e8765ef2413dc6fff1ac71bebe3efa9e5be39b80

Merge base:

542e70064493b90689e38af0c9009eb10c75b284

antlr4/runtime/Java/src/org/antlr/v4/runtime/DefaultErrorStrategy.java

Chunk 34: (combination/method invocation, variable)

```
		if ( lastErrorIndex==recognizer.getInputStream().index() &&
<<<<<<< HEAD
			lastErrorStates != null &&
			lastErrorStates.contains(recognizer._ctx.s) ) {
=======
		lastErrorStates.contains(recognizer.getState()) ) {
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
			// uh oh, another error at same token index and previously-visited
```

```
		if ( lastErrorIndex==recognizer.getInputStream().index() &&
			lastErrorStates != null &&
			lastErrorStates.contains(recognizer.getState()) ) {
			// uh oh, another error at same token index and previously-visited
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/FailedPredicateException.java

Chunk 35: (version 1/method declaration, method invocation, method signature, variable)

```
	}

<<<<<<< HEAD
	public   FailedPredicateException(@NotNull   Parser   recognizer,   @Nullable   String
predicate) {
		this(recognizer, predicate, null);
	}

	public FailedPredicateException(@NotNull Parser recognizer,
							@Nullable         String
predicate,
							@Nullable         String
message)
	{
		super(formatMessage(predicate,             message),             recognizer,
recognizer.getInputStream(), recognizer._ctx);
		ATNState s = recognizer.getInterpreter().atn.states.get(recognizer._ctx.s);
=======
	public FailedPredicateException(Parser recognizer, @Nullable String predicate) {
		super(recognizer, recognizer.getInputStream(), recognizer._ctx);
		ATNState                                s                                =
recognizer.getInterpreter().atn.states.get(recognizer.getState());
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
		PredicateTransition trans = (PredicateTransition)s.transition(0);
```

```
	}
```

```
        public   FailedPredicateException(@NotNull   Parser   recognizer,   @Nullable   String
predicate) {
                this(recognizer, predicate, null);
        }

        public FailedPredicateException(@NotNull Parser recognizer,
                                                                    @Nullable          String
predicate,
                                                                    @Nullable          String
message)
        {
                super(formatMessage(predicate,             message),            recognizer,
recognizer.getInputStream(), recognizer._ctx);
                ATNState                                  s                                  =
recognizer.getInterpreter().atn.states.get(recognizer.getState());
                PredicateTransition trans = (PredicateTransition)s.transition(0);
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/Parser.java

Chunk 36: (combination/method invocation, method signature)

```
         */
<<<<<<< HEAD
        public void enterRule(ParserRuleContext localctx, int ruleIndex) {
=======
        public void enterRule(ParserRuleContext<Token> localctx, int state, int ruleIndex) {
                setState(state);
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
                _ctx = localctx;
```

```
         */
        public void enterRule(ParserRuleContext localctx, int state, int ruleIndex) {
                setState(state);
                _ctx = localctx;
```

Chunk 37: (new code/ method invocation/ variable)

```
         if ( _parseListeners != null) triggerExitRuleEvent();
<<<<<<< HEAD
                _ctx = (ParserRuleContext)_ctx.parent;
=======
                setState(_ctx.invokingState);
                _ctx = (ParserRuleContext<Token>)_ctx.parent;
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
    }
```

```
         if ( _parseListeners != null) triggerExitRuleEvent();
                setState(_ctx.invokingState);
                _ctx = (ParserRuleContext)_ctx.parent;
    }
```

Chunk 38: (version 1/commentary, method invocation, method signature)

```
        }

<<<<<<< HEAD
        public void enterRecursionRule(ParserRuleContext localctx, int ruleIndex) {
=======
        /* like enterRule but for recursive rules; no enter events for recursive rules. */
        public  void  pushNewRecursionContext(ParserRuleContext<Token>  localctx,  int  state,
int ruleIndex) {
                setState(state);
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
                _ctx = localctx;
```

```
        }

        public void enterRecursionRule(ParserRuleContext localctx, int ruleIndex) {
                _ctx = localctx;
```

Chunk 39: (version 1/commentary, method invocation, method signature, variable)

```
        }

<<<<<<< HEAD
        public void unrollRecursionContexts(ParserRuleContext _parentctx) {
                _ctx.stop = _input.LT(-1);
                ParserRuleContext retctx = _ctx; // save current ctx (return value)
=======
        public    void    unrollRecursionContexts(ParserRuleContext<Token>    _parentctx,    int
_parentState) {
                ParserRuleContext<Token> retctx = _ctx; // save current ctx (return value)
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80

                // unroll so _ctx is as it was before call to recursive method
```

```
        }

        public void unrollRecursionContexts(ParserRuleContext _parentctx) {
                _ctx.stop = _input.LT(-1);
                ParserRuleContext retctx = _ctx; // save current ctx (return value)

                // unroll so _ctx is as it was before call to recursive method
```

Chunk 40: (version 1/method invocation, variable)

```
                                triggerExitRuleEvent();
<<<<<<< HEAD
                                _ctx = (ParserRuleContext)_ctx.parent;
=======
                                setState(_ctx.invokingState);
                                _ctx = (ParserRuleContext<Token>)_ctx.parent;
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
                        }
```

```
                                triggerExitRuleEvent();
                                _ctx = (ParserRuleContext)_ctx.parent;
                        }
```

Chunk 41: (combination/method invocation, variable)

```
        ATN atn = getInterpreter().atn;
<<<<<<< HEAD
                ParserRuleContext ctx = _ctx;
        ATNState s = atn.states.get(ctx.s);
=======
                ParserRuleContext<?> ctx = _ctx;
        ATNState s = atn.states.get(getState());
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
```

```
        IntervalSet following = atn.nextTokens(s);
```

```
        ATN atn = getInterpreter().atn;
                ParserRuleContext ctx = _ctx;
        ATNState s = atn.states.get(getState());
        IntervalSet following = atn.nextTokens(s);
```

Chunk 42: (combination/ method invocation, variable)

```
        ATN atn = getInterpreter().atn;
<<<<<<< HEAD
                ParserRuleContext ctx = _ctx;
        ATNState s = atn.states.get(ctx.s);
=======
                ParserRuleContext<?> ctx = _ctx;
        ATNState s = atn.states.get(getState());
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
        IntervalSet following = atn.nextTokens(s);
```

```
        ATN atn = getInterpreter().atn;
                ParserRuleContext ctx = _ctx;
        ATNState s = atn.states.get(getState());
        IntervalSet following = atn.nextTokens(s);
```

Chunk 43: (new code/commentary, method declaration)

```
        }

<<<<<<< HEAD
        /** Indicate that the recognizer has changed internal state that is
         *  consistent with the ATN state passed in.  This way we always know
         *  where we are in the ATN as the parser goes along. The rule
         *  context objects form a stack that lets us see the stack of
         *  invoking rules. Combine this and we have complete ATN
         *  configuration information.
         */
        public void setState(int atnState) {
//              System.err.println("setState "+atnState);
                _ctx.s = atnState;
//              if ( traceATNStates ) _ctx.trace(atnState);
        }

        /** During a parse is sometimes useful to listen in on the rule entry and exit
=======
        /** During a parse is extremely useful to listen in on the rule entry and exit
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
         *  events as well as token matches. This is for quick and dirty debugging.
```

```
        }

        /** During a parse is sometimes useful to listen in on the rule entry and exit
         *  events as well as token matches. This is for quick and dirty debugging.
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/ParserRuleContext.java

Case 44: (new code/commentary, variable)

```
//      public List<Integer> states;
```

```
<<<<<<< HEAD
	/** Current ATN state number we are executing.
	 *
	 *  Not used during ATN simulation/prediction; only used during parse that updates
	 *  current location in ATN.
	 */
	public int s = -1;

	public Token start, stop;
=======
	public Symbol start, stop;

	/** Set during parsing to identify which rule parser is in. */
	public int ruleIndex;
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80

	/** Set during parsing to identify which alt of rule parser is in. */
```

```
//	public List<Integer> states;

	public Token start, stop;

	/** Set during parsing to identify which alt of rule parser is in. */
```

Chuhnk 45: (new code/method declaration)

```
	}

<<<<<<< HEAD
	public      ParserRuleContext(@Nullable      ParserRuleContext      parent,      int
invokingStateNumber, int stateNumber) {
		super(parent, invokingStateNumber);
		this.s = stateNumber;
	}

	public ParserRuleContext(@Nullable ParserRuleContext parent, int stateNumber) {
		this(parent, parent!=null ? parent.s : -1 /* invoking state */, stateNumber);
=======
	public     ParserRuleContext(@Nullable     ParserRuleContext<Symbol>     parent,     int
invokingStateNumber) {
		super(parent, invokingStateNumber);
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80
	}
```

```
	}

	public      ParserRuleContext(@Nullable      ParserRuleContext      parent,      int
invokingStateNumber) {
		super(parent, invokingStateNumber);
	}
```

Chunk 46: (version 1/ annotation, commentary, method declaration)

```
	}

<<<<<<< HEAD
	public Token getStart() { return start; }
	public Token getStop() { return stop; }
=======
	/** Return the text matched by this context and below in the parse
```

```
         *  tree. It includes tokens from this.start .. this.stop inclusive.
         *  It includes hidden channel tokens between start, stop.  The
         *  edge tokens are always on-channel tokens.
         */
        public String getText(TokenStream tokens) {
               Interval range = getSourceInterval();
               return range==Interval.EMPTY ? null : tokens.toString(range.a, range.b);
        }

        public Symbol getStart() { return start; }
        public Symbol getStop() { return stop; }

        @Override
        public String toString(@NotNull Recognizer<?,?> recog, RuleContext stop) {
               if ( recog==null ) return super.toString(recog, stop);
               StringBuilder buf = new StringBuilder();
               ParserRuleContext<?> p = this;
               buf.append("[");
               int state = recog.getState();
               while ( p != null && p != stop ) {
                      ATN atn = recog.getATN();
                      ATNState s = atn.states.get(state);
                      String ruleName = recog.getRuleNames()[s.ruleIndex];
                      buf.append(ruleName);
                      if ( p.parent != null ) buf.append(" ");
                      state = p.invokingState;
                      p = (ParserRuleContext<?>)p.parent;
               }
               buf.append("]");
               return buf.toString();
        }
>>>>>>> e8765ef2413dc6fff1ac71bebe3efa9e5be39b80

    /** Used for rule context info debugging during parse-time, not so much for ATN
debugging */
```

```
        }

        public Token getStart() { return start; }
        public Token getStop() { return stop; }

    /** Used for rule context info debugging during parse-time, not so much for ATN
debugging */
```

Version: a115490d5e2f2fbdadbd9f7043f85779190a109c

Parents:

ee647907391fbdbc0fa6e64624aee89ced3a3197

4304ba251fb93c044d57a5a16d5ff494e50ac468

Merge base:

bf9c7c3a28c04e0f8ad2e304416d828d308894ec

antlr4/tool/src/org/antlr/v4/Tool.java

Chunk 47: (combination/variable)

```
	public boolean gen_visitor = false;
<<<<<<< HEAD
	public boolean gen_dependencies = false;
	public String genPackage = null;
	public Map<String, String> grammarOptions = null;
=======
	public boolean abstract_recognizer = false;
	public boolean warnings_are_errors = false;
>>>>>>> 4304ba251fb93c044d57a5a16d5ff494e50ac468

    public static Option[] optionDefs = {
```

```
	public boolean gen_visitor = false;
	public boolean gen_dependencies = false;
	public String genPackage = null;
	public Map<String, String> grammarOptions = null;
	public boolean warnings_are_errors = false;

    public static Option[] optionDefs = {
```

Chunk 48: (combination, method invocation)

```
		new Option("gen_visitor",             "-no-visitor",  "don't  generate  parse
tree visitor (default)"),
<<<<<<< HEAD
		new Option("genPackage",              "-package",       OptionArgType.STRING,
"specify a package/namespace for the generated code"),
		new Option("gen_dependencies",        "-depend",        "generate        file
dependencies"),
		new Option("",                                "-D<option>=value",
"set/override a grammar-level option"),
=======
		new Option("abstract_recognizer", "-abstract", "generate abstract recognizer
classes"),
		new Option("warnings_are_errors", "-Werror", "treat warnings as errors"),
>>>>>>> 4304ba251fb93c044d57a5a16d5ff494e50ac468

		new Option("saveLexer",                       "-Xsave-lexer", "save temp lexer
file created for combined grammars"),
```

```
		new Option("gen_visitor",             "-no-visitor",  "don't  generate  parse
tree visitor (default)"),
		new Option("genPackage",              "-package",       OptionArgType.STRING,
"specify a package/namespace for the generated code"),
                new Option("gen_dependencies",       "-depend",        "generate        file
dependencies"),
                new Option("",                              "-D<option>=value",
"set/override a grammar-level option"),
                new Option("warnings_are_errors", "-Werror", "treat warnings as errors"),
                new Option("saveLexer",                     "-Xsave-lexer", "save temp lexer
file created for combined grammars"),
```

antlr4/tool/src/org/antlr/v4/tool/ErrorType.java

Chunk 49: (new code/enum)

```
        FILE_AND_GRAMMAR_NAME_DIFFER(8,  "grammar  name  <arg>  and  file  name  <arg2>  differ",
ErrorSeverity.ERROR),
<<<<<<< HEAD
        BAD_OPTION_SET_SYNTAX(9, "invalid -Dname=value syntax: <arg>", ErrorSeverity.ERROR),
=======
        WARNING_TREATED_AS_ERROR(9, "warning treated as error", ErrorSeverity.ERROR),
//      FILENAME_EXTENSION_ERROR("", ErrorSeverity.ERROR),
>>>>>>> 4304ba251fb93c044d57a5a16d5ff494e50ac468

        INTERNAL_ERROR(20,      "internal      error:      <arg>      <arg2><if(exception)>:
<exception><endif>\n" +
```

```
        FILE_AND_GRAMMAR_NAME_DIFFER(8,  "grammar  name  <arg>  and  file  name  <arg2>  differ",
ErrorSeverity.ERROR),
        BAD_OPTION_SET_SYNTAX(9, "invalid -Dname=value syntax: <arg>", ErrorSeverity.ERROR),
        WARNING_TREATED_AS_ERROR(10, "warning treated as error", ErrorSeverity.ERROR),

        INTERNAL_ERROR(20,      "internal      error:      <arg>      <arg2><if(exception)>:
<exception><endif>\n" +
```

Version: 201db8b6d0c1b4072fe1df0e71e72783f0c3b757

Parents:

a0563656f74a67a8ee9d4258ba78e5084be2c56a

dd12508f5d289c08c3259943024b208b675d42c6

Merge base:

1155c40fc8f4acf93b7b2e400b540ee0a8d5e437

antlr4/runtime/Java/src/org/antlr/v4/runtime/ParserRuleContext.java

Chunk 50: (version 2/annotation, method declaration)

```
        public Symbol getStop() { return stop; }

<<<<<<< HEAD
        @Override
        public String toString(@NotNull Recognizer<?,?> recog, RuleContext stop) {
                if ( recog==null ) return super.toString(recog, stop);
                StringBuilder buf = new StringBuilder();
                RuleContext p = this;
                buf.append("[");
                String[] ruleNames = recog.getRuleNames();
                while ( p != null && p != stop ) {
                        int ruleIndex = p.getRuleIndex();
                        String ruleName = ruleIndex >= 0 && ruleIndex < ruleNames.length ?
ruleNames[ruleIndex] : Integer.toString(ruleIndex);
                        buf.append(ruleName);
                        if ( p.parent != null ) buf.append(" ");
                        p = p.parent;
                }
                buf.append("]");
                return buf.toString();
        }

=======
>>>>>>> dd12508f5d289c08c3259943024b208b675d42c6
    /** Used for rule context info debugging during parse-time, not so much for ATN
debugging */
```

```
        public Symbol getStop() { return stop; }

    /** Used for rule context info debugging during parse-time, not so much for ATN
debugging */
```

Version: 2d62b73a14f360a14ad51162e82f923453082d8f

Parents:

64c050f2338afabb3a39e90c2bb4fe9ad9cba30a

199e9892dc6d9e263148481797296498ecedbf66

Merge base:

2947fe6a2ad1c01f98920a8d42d668664719d23b

antlr4/runtime/Java/src/org/antlr/v4/runtime/RuleContext.java

Chunk 51: (version 1/ class signature, method invocation, variable)

```
*/
<<<<<<< HEAD
public class RuleContext implements ParseTree.RuleNode {
	public static final ParserRuleContext<Token> EMPTY = new ParserRuleContext<Token>();

=======
public class RuleContext implements RuleNode {
>>>>>>> 199e9892dc6d9e263148481797296498ecedbf66
	/** What context invoked this rule? */
```

```
*/
public class RuleContext implements RuleNode {
	public static final ParserRuleContext<Token> EMPTY = new ParserRuleContext<Token>();

	/** What context invoked this rule? */
```

Version: 492980de71f30014fd5d4c23712cc18abfbc8555

Parents:

0d92c25056d1c8e0ef0422a1bbef34c6851bd308

2947fe6a2ad1c01f98920a8d42d668664719d23b

Merge base:

1e88980db5309fc000d54e299a6ac3130e8ec572

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 52: (version 1/commentary, if statement, method invocation, variable, while statement)

```
                        else if ( config.state.getClass()==LoopEndState.class ) {
<<<<<<< HEAD
                                if ( debug ) System.out.print("Loop end; pop, stack=" +
config.context);
                                LoopEndState end = (LoopEndState)config.state;
                                // pop all the way back until we don't see the loopback state
anymore
                                config.context                                          =
config.context.popAll(end.loopBackStateNumber,

          configs.fullCtx,

          mergeCache);
                                if ( debug ) System.out.println(" becomes "+config.context);
=======
                                if ( debug ) System.out.println("Loop end; pop, stack=" +
config.context);
                                RuleContext p = config.context;
                                LoopEndState end = (LoopEndState) config.state;
                                int loopBackStateNumber = end.loopBackState.stateNumber;
                                while ( !p.isEmpty() && p.invokingState == loopBackStateNumber
) {
                                        p = config.context = config.context.parent; // "pop"
                                }
>>>>>>> 2947fe6a2ad1c01f98920a8d42d668664719d23b
                        }
```

```
                        else if ( config.state.getClass()==LoopEndState.class ) {
                                if ( debug ) System.out.print("Loop end; pop, stack=" +
config.context);
                                LoopEndState end = (LoopEndState)config.state;
                                // pop all the way back until we don't see the loopback state
anymore
                                int loopBackStateNumber = end.loopBackState.stateNumber;
                                config.context = config.context.popAll(loopBackStateNumber,

          configs.fullCtx,

          mergeCache);
                                if ( debug ) System.out.println(" becomes "+config.context);
                        }
```

Version: aed26c690eb967f58bff1a4929901525929bfc57

Parents:

c2b49bd94eb0d186584765294ed87f91cf9e8daf

1e88980db5309fc000d54e299a6ac3130e8ec572

Merge base:

c7d1ea7e2313a1bc10d959361373ef1f7a2ad2d8

antlr4/tool/src/org/antlr/v4/semantics/SemanticPipeline.java

Chunk 53: (concatenation/import declaration)

```
import org.antlr.v4.runtime.Token;
<<<<<<< HEAD
import org.antlr.v4.runtime.misc.Pair;
=======
import org.antlr.v4.runtime.misc.NotNull;
>>>>>>> 1e88980db5309fc000d54e299a6ac3130e8ec572
import org.antlr.v4.tool.ErrorType;
```

```
import org.antlr.v4.runtime.Token;
import org.antlr.v4.runtime.misc.NotNull;
import org.antlr.v4.runtime.misc.Pair;
import org.antlr.v4.tool.ErrorType;
```

Version: c2b49bd94eb0d186584765294ed87f91cf9e8daf

Parents:

b7b2a45c8b0ec2cab083c708ea3bceef694c1a7b

c7d1ea7e2313a1bc10d959361373ef1f7a2ad2d8

Merge base:

170a8347bb597535f565e18d8275f9afc6b830fc

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/LexerATNSimulator.java

Chunk 54: (new code/ If statement, Switch statement, Variable)

```
               ATNState p = config.state;
<<<<<<< HEAD
               LexerATNConfig c = null;
               if ( t.getClass() == RuleTransition.class ) {
                       PredictionContext newContext =
                               new SingletonPredictionContext(config.context, p.stateNumber);
                       c = new LexerATNConfig(config, t.target, newContext);
               }
               else if ( t.getClass() == PredicateTransition.class ) {
=======
               ATNConfig c;

               switch (t.getSerializationType()) {
               case Transition.RULE:
                       RuleContext newContext =
                               new RuleContext(config.context, p.stateNumber);
```

```
                        c = new ATNConfig(config, t.target, newContext);
                        break;

                case Transition.PREDICATE:
>>>>>>> c7d1ea7e2313a1bc10d959361373ef1f7a2ad2d8
                        if (recog == null) {
```

```
                ATNState p = config.state;

                LexerATNConfig c = null;
                switch (t.getSerializationType()) {
                        case Transition.RULE:
                                PredictionContext newContext =
                                        new          SingletonPredictionContext(config.context,
p.stateNumber);
                                c = new LexerATNConfig(config, t.target, newContext);
                                break;
                        case Transition.PREDICATE:
//                              if (recog == null) {
```

Chunk 55: (new code/ commentary, if stament, switch statement)

```
                        }
<<<<<<< HEAD
                }
                // ignore actions; just exec one per rule upon accept
                else if ( t.getClass() == ActionTransition.class ) {
                        c         =         new         LexerATNConfig(config,         t.target,
((ActionTransition)t).actionIndex);
                }
                else if ( t.isEpsilon() ) {
                        c = new LexerATNConfig(config, t.target);
=======
                        else {
                                c = null;
                        }

                        break;

                case Transition.ACTION:
                        // ignore actions; just exec one per rule upon accept
                        c = new ATNConfig(config, t.target);
                        c.lexerActionIndex = ((ActionTransition)t).actionIndex;
                        break;

                case Transition.EPSILON:
                        c = new ATNConfig(config, t.target);
                        break;

                default:
                        c = null;
                        break;
>>>>>>> c7d1ea7e2313a1bc10d959361373ef1f7a2ad2d8
                }
```

```
                                }
                                break;
                        // ignore actions; just exec one per rule upon accept
                        case Transition.ACTION:
                                c       =       new       LexerATNConfig(config,       t.target,
((ActionTransition)t).actionIndex);
```

```
                                break;
                        case Transition.EPSILON:
                                c = new LexerATNConfig(config, t.target);
                                break;
                }
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 56: (new code/ if statement, method signature, switch statement, return statement)

```
        @Nullable
<<<<<<< HEAD
        public ATNConfig getEpsilonTarget(@NotNull ATNConfig config,
                                                                          @NotNull Transition t,
                                                                          boolean
collectPredicates,
                                                                          boolean inContext,
                                                                          boolean fullCtx)
        {
                if ( t instanceof RuleTransition ) {
                        return ruleTransition(config, t);
                }
                else if ( t instanceof PredicateTransition ) {
                        return predTransition(config, (PredicateTransition)t,
                                                              collectPredicates,
                                                              inContext,
                                                              fullCtx);
                }
                else if ( t instanceof ActionTransition ) {
=======
        public ATNConfig getEpsilonTarget(@NotNull ATNConfig config, @NotNull Transition t,
boolean collectPredicates, boolean inContext) {
                switch (t.getSerializationType()) {
                case Transition.RULE:
                        return ruleTransition(config, t);

                case Transition.PREDICATE:
                        return         predTransition(config,         (PredicateTransition)t,
collectPredicates, inContext);

                case Transition.ACTION:
>>>>>>> c7d1ea7e2313a1bc10d959361373ef1f7a2ad2d8
                        return actionTransition(config, (ActionTransition)t);

                case Transition.EPSILON:
                        return new ATNConfig(config, t.target);

                default:
                        return null;
```

```
        @Nullable
        public ATNConfig predTransition(@NotNull ATNConfig config,
                                                                        @NotNull
PredicateTransition pt,
                                                                        boolean
collectPredicates,
                                                                        boolean inContext,
                                                                        boolean fullCtx)
        {
                if ( debug ) {
                    System.out.println("PRED (collectPredicates="+collectPredicates+") "+
                  pt.ruleIndex+":"+pt.predIndex+
                                ", ctx dependent="+pt.isCtxDependent);
                    if ( parser != null ) {
              System.out.println("context surrounding pred is "+
                              parser.getRuleInvocationStack());
            }
               }

               ATNConfig c = null;
               if ( collectPredicates &&
                        (!pt.isCtxDependent || (pt.isCtxDependent&&inContext)) )
               {
                       if ( fullCtx ) {
                               // In full context mode, we can evaluate predicates on-the-fly
                               // during closure, which dramatically reduces the size of
                               // the config sets. It also obviates the need to test
predicates
                               // later during conflict resolution.
                               int currentPosition = _input.index();
                               _input.seek(_startIndex);
                               boolean predSucceeds = pt.getPredicate().eval(parser,
_outerContext);
                               _input.seek(currentPosition);
                               if ( predSucceeds ) {
                                       c = new ATNConfig(config, pt.target); // no pred
context
                               }
                       }
                       else {
                               SemanticContext newSemCtx =
                                       SemanticContext.and(config.semanticContext,
pt.getPredicate());
                               c = new ATNConfig(config, pt.target, newSemCtx);
                       }
               }
               else {
                       c = new ATNConfig(config, pt.target);
               }

               if ( debug ) System.out.println("config from pred transition="+c);
        return c;
        }

        @NotNull
```

Version: c893f2af08180ee55f298ccfe3d57a29b9171be8

Parents:

c0ece0bd091bdd2a23464043969416a8f2593487

7d4f71d829b24a2aaf6a52d2add6123212d3cb37

Merge base:

9539572ee7155ca403d0cb6bf9ee0d74fee4d0c1

antlr4/tool/src/org/antlr/v4/automata/ATNSerializer.java

Chunk 57: (combination/ import declaration)

```
import org.antlr.v4.parse.ANTLRParser;
<<<<<<< HEAD
import org.antlr.v4.runtime.atn.ATN;
import org.antlr.v4.runtime.atn.ATNSimulator;
import org.antlr.v4.runtime.atn.ATNState;
import org.antlr.v4.runtime.atn.ActionTransition;
import org.antlr.v4.runtime.atn.AtomTransition;
import org.antlr.v4.runtime.atn.DecisionState;
import org.antlr.v4.runtime.atn.LoopEndState;
import org.antlr.v4.runtime.atn.PredicateTransition;
import org.antlr.v4.runtime.atn.RangeTransition;
import org.antlr.v4.runtime.atn.RuleTransition;
import org.antlr.v4.runtime.atn.SetTransition;
import org.antlr.v4.runtime.atn.Transition;
=======
import org.antlr.v4.runtime.atn.*;
import org.antlr.v4.runtime.misc.IntegerList;
>>>>>>> 7d4f71d829b24a2aaf6a52d2add6123212d3cb37
import org.antlr.v4.runtime.misc.Interval;
```

```
import org.antlr.v4.parse.ANTLRParser;
import org.antlr.v4.runtime.atn.ATN;
import org.antlr.v4.runtime.atn.ATNSimulator;
import org.antlr.v4.runtime.atn.ATNState;
import org.antlr.v4.runtime.atn.ActionTransition;
import org.antlr.v4.runtime.atn.AtomTransition;
import org.antlr.v4.runtime.atn.DecisionState;
import org.antlr.v4.runtime.atn.LoopEndState;
import org.antlr.v4.runtime.atn.PredicateTransition;
import org.antlr.v4.runtime.atn.RangeTransition;
import org.antlr.v4.runtime.atn.RuleTransition;
import org.antlr.v4.runtime.atn.SetTransition;
import org.antlr.v4.runtime.atn.Transition;
import org.antlr.v4.runtime.misc.IntegerList;
import org.antlr.v4.runtime.misc.Interval;
```

antlr4/tool/src/org/antlr/v4/tool/DOTGenerator.java

Chunk 58: (version 2/method invocation, variable)

```
                              altList.addAll(alts);
<<<<<<< HEAD
                              Collections.sort(altList);
                              Set<ATNConfig> configurations = s.configs;
=======
                              altList.sort();
                              Set<ATNConfig> configurations = s.configset;
>>>>>>> 7d4f71d829b24a2aaf6a52d2add6123212d3cb37
                              for (int altIndex = 0; altIndex < altList.size(); altIndex++)
{
```

```
                              altList.addAll(alts);
                              altList.sort();
                              Set<ATNConfig> configurations = s.configs;
                              for (int altIndex = 0; altIndex < altList.size(); altIndex++)
{
```

antlr4/tool/test/org/antlr/v4/test/TestATNInterpreter.java

Chunk 59: (combination/ method invocation, variable)

```
               ATN lexatn = createATN(lg);
<<<<<<< HEAD
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn,null,null);
               List<Integer> types = getTokenTypesViaATN(inputString, lexInterp);
=======
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn);
               IntegerList types = getTokenTypesViaATN(inputString, lexInterp);
>>>>>>> 7d4f71d829b24a2aaf6a52d2add6123212d3cb37
               System.out.println(types);
```

```
               ATN lexatn = createATN(lg);
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn,null,null);
               IntegerList types = getTokenTypesViaATN(inputString, lexInterp);
               System.out.println(types);
```

antlr4/tool/test/org/antlr/v4/test/TestATNParserPrediction.java

Chunk 60: (combination/ method invocation, variable)

```
               ATN lexatn = createATN(lg);
<<<<<<< HEAD
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn,null,null);
               List<Integer> types = getTokenTypesViaATN(inputString, lexInterp);
=======
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn);
               IntegerList types = getTokenTypesViaATN(inputString, lexInterp);
>>>>>>> 7d4f71d829b24a2aaf6a52d2add6123212d3cb37
               System.out.println(types);
```

```
               ATN lexatn = createATN(lg);
               LexerATNSimulator lexInterp = new LexerATNSimulator(lexatn,null,null);
               IntegerList types = getTokenTypesViaATN(inputString, lexInterp);
               System.out.println(types);
```

Version: 0141bc058a57f68dec77b359cc7fecbb99dda62a

Parents:

3ece2c8640821cd1103c6e0a15aceceda008abb4

885f6530ada3a97b050dfdfaa7570d8b0121bd5e

Merge base:

abc0e2ef878ee86eac1574de737297d2a55eeaa7

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ATNConfigSet.java

Chunk 61: (combination/if statement, method invocation)

```
		StringBuilder buf = new StringBuilder();
<<<<<<< HEAD
		buf.append(elements().toString());
		if                  (                  hasSemanticContext                  )
buf.append(",hasSemanticContext="+hasSemanticContext);
		if ( uniqueAlt!=ATN.INVALID_ALT_NUMBER ) buf.append(",uniqueAlt="+uniqueAlt);
		if ( conflictingAlts!=null ) buf.append(",conflictingAlts="+conflictingAlts);
=======
		buf.append(super.toString());
		if                  (                  hasSemanticContext                  )
buf.append(",hasSemanticContext=").append(hasSemanticContext);
		if             (             uniqueAlt!=ATN.INVALID_ALT_NUMBER             )
buf.append(",uniqueAlt=").append(uniqueAlt);
		if                 (                 conflictingAlts!=null                 )
buf.append(",conflictingAlts=").append(conflictingAlts);
>>>>>>> 885f6530ada3a97b050dfdfaa7570d8b0121bd5e
		if ( dipsIntoOuterContext ) buf.append(",dipsIntoOuterContext");
```

```
		StringBuilder buf = new StringBuilder();
//		buf.append(elements().toString());
		buf.append(super.toString());
		if                  (                  hasSemanticContext                  )
buf.append(",hasSemanticContext=").append(hasSemanticContext);
		if             (             uniqueAlt!=ATN.INVALID_ALT_NUMBER             )
buf.append(",uniqueAlt=").append(uniqueAlt);
		if                 (                 conflictingAlts!=null                 )
buf.append(",conflictingAlts=").append(conflictingAlts);
		if ( dipsIntoOuterContext ) buf.append(",dipsIntoOuterContext");
```

Version: 3ece2c8640821cd1103c6e0a15aceceda008abb4

Parents:

d46c7db52c8008fe02447572bf7cf1e0ced1459a

abc0e2ef878ee86eac1574de737297d2a55eeaa7

Merge base:

c590ba8fd8a593d63659bc36c52aab42bcd3cb0f

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/LL1Analyzer.java

Chunk 62: (version 1/ method invocation, variable)

```
                       if ( t.getClass() == RuleTransition.class ) {
<<<<<<< HEAD
                               PredictionContext newContext =
                                       new SingletonPredictionContext(ctx, s.stateNumber);
=======
                               RuleContext newContext = new RuleContext(ctx, s.stateNumber);
>>>>>>> abc0e2ef878ee86eac1574de737297d2a55eeaa7
                               _LOOK(t.target, newContext, look, lookBusy, seeThruPreds);
```

```
                       if ( t.getClass() == RuleTransition.class ) {
                               PredictionContext newContext =
                                       new SingletonPredictionContext(ctx, s.stateNumber);
                               _LOOK(t.target, newContext, look, lookBusy, seeThruPreds);
```

antlr4/tool/playground/TestT.java

Chunk 63: (version 2/ commentary, method invocation, variable)

```
               CommonTokenStream tokens = new CommonTokenStream(lex);
<<<<<<< HEAD
//             tokens.fill();
//             System.out.println(tokens);
               TParser parser = new TParser(tokens);
               parser.setBuildParseTree(true);
               parser.s();
=======
               tokens.fill();
               System.out.println(tokens.getTokens());
>>>>>>> abc0e2ef878ee86eac1574de737297d2a55eeaa7
        }
```

```
               CommonTokenStream tokens = new CommonTokenStream(lex);
               tokens.fill();
               System.out.println(tokens.getTokens());
        }
```

Version: 9d9244612541132a3c5c766be9df160ff5356ebc

Parents:

ea434982fb9f4aa7d7cd68554b172079375055d8

585aa0a14b2e89c20f6b2ac25724b71337b54ed2

Merge base:

ea7037dd2dff6c36b358b0a641c197d499367c0c

antlr4/tool/test/org/antlr/v4/test/BaseTest.java

Chunk 64: (version 1/ if statement, method invocation, variable)

```
			antlr.processGrammarsOnCommandLine();
<<<<<<< HEAD
=======
			allIsWell = equeue.errors.isEmpty();
			if ( !defaultListener && !equeue.errors.isEmpty() ) {
				System.err.println("antlr reports errors from "+options);
				for (int i = 0; i < equeue.errors.size(); i++) {
					ANTLRMessage msg = equeue.errors.get(i);
					System.err.println(msg);
				}
				System.out.println("!!!\ngrammar:");
				System.out.println(grammarStr);
				System.out.println("###");
			}
			if ( !defaultListener && !equeue.warnings.isEmpty() ) {
				System.err.println("antlr reports warnings from "+options);
				for (int i = 0; i < equeue.warnings.size(); i++) {
					ANTLRMessage msg = equeue.warnings.get(i);
					System.err.println(msg);
				}
			}
>>>>>>> 585aa0a14b2e89c20f6b2ac25724b71337b54ed2
		}
```

```
			antlr.processGrammarsOnCommandLine();
		}
```

Version: 9ef61279829dcee16823f8c8b2ced9159c0dd026

Parents:

    c9aef6fdbeefcda788aa65006b3866eae7263a46
    adad53ee18f14f2e1c8a695c604b701bd0926a1e

Merge base:

    9fbe9b6e21e306820c340ba29a177644c28d9775

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/LexerATNSimulator.java

Chunk 65: (version 1/import declaration)

```
import org.antlr.v4.runtime.LexerNoViableAltException;
<<<<<<< HEAD
=======
import org.antlr.v4.runtime.RuleContext;
>>>>>>> adad53ee18f14f2e1c8a695c604b701bd0926a1e
import org.antlr.v4.runtime.Token;
```

```
import org.antlr.v4.runtime.LexerNoViableAltException;
import org.antlr.v4.runtime.Token;
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 66: (version 1/commentary, if statement, while statement, variable)

```
                        else if ( config.state.getClass()==LoopEndState.class ) {
<<<<<<< HEAD
                               if    (    debug    )    System.out.print("Loop    end;    pop,
stack="+config.context);
                               LoopEndState end = (LoopEndState)config.state;
                               // pop all the way back until we don't see the loopback state
anymore
                               config.context                                            =
config.context.popAll(end.loopBackStateNumber, configs.fullCtx);
                               if ( debug ) System.out.println(" becomes "+config.context);
=======
                               if  (  debug  )  System.out.println("Loop  end;  pop,  stack="  +
config.context);
                               RuleContext p = config.context;
                               LoopEndState end = (LoopEndState) config.state;
                               while     (     !p.isEmpty()     &&     p.invokingState     ==
end.loopBackStateNumber ) {
                                       p = config.context = config.context.parent; // "pop"
                               }
>>>>>>> adad53ee18f14f2e1c8a695c604b701bd0926a1e
                        }
```

```
                        else if ( config.state.getClass()==LoopEndState.class ) {
                               if    (    debug    )    System.out.print("Loop    end;    pop,
stack="+config.context);
                               LoopEndState end = (LoopEndState)config.state;
                               // pop all the way back until we don't see the loopback state
anymore
                               config.context                                            =
config.context.popAll(end.loopBackStateNumber, configs.fullCtx);
                               if ( debug ) System.out.println(" becomes "+config.context);
```

```
                                        }
```

Version: 6791bf60cf90524ab0b480b1c26c49c5af19389a

Parents:

b1bcde76b746f1fef82d78e9f478ca51042b7bc4

3f1f76df7d44332c637e5a92f27933e9c9f3e5ac

Merge base:

768bfc0cf2e705cd0eeaa0ab11bcd18f453442a0

antlr4/runtime/Java/src/org/antlr/v4/runtime/ANTLRErrorStrategy.java

Chunk 67: (commentary, method internface)

```
        throws RecognitionException;
<<<<<<< HEAD

        /**  Called  when  the  parser  detects  a  true  ambiguity:  an  input  sequence  can  be
matched
         * literally by two or more pass through the grammar. ANTLR resolves the ambiguity
in
         * favor of the alternative appearing first in the grammar. The start and stop index
are
     * zero-based absolute indices into the token stream. ambigAlts is a set of alternative
numbers
     * that can match the input sequence. This method is only called when we are parsing
with
     * full context.
     */
    void reportAmbiguity(@NotNull Parser recognizer,
                                             DFA   dfa,   int   startIndex,   int   stopIndex,
@NotNull IntervalSet ambigAlts,
                                             @NotNull ATNConfigSet configs);

        void reportAttemptingFullContext(@NotNull Parser recognizer,
                                                                      @NotNull DFA dfa,
                                                                      int    startIndex,    int
stopIndex,
                                                                      @NotNull    ATNConfigSet
configs);

        /** Called by the parser when it find a conflict that is resolved by retrying the
parse
     *  with full context. This is not a warning; it simply notifies you that your grammar
     *  is more complicated than Strong LL can handle. The parser moved up to full context
     *  parsing for that input sequence.
     */
    void reportContextSensitivity(@NotNull Parser recognizer,
                                  @NotNull DFA dfa,
                                  int startIndex, int stopIndex,
                                  @NotNull ATNConfigSet configs);
=======
>>>>>>> 3f1f76df7d44332c637e5a92f27933e9c9f3e5ac
}
```

```
                                          @Nullable RecognitionException e)
        throws RecognitionException;
}
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/DefaultErrorStrategy.java

Chunk 68: (version 2/ annotation, method declaration)

```
    }
<<<<<<< HEAD

    @Override
    public void reportAmbiguity(@NotNull Parser recognizer,
                                                                DFA  dfa,  int  startIndex,  int
stopIndex, @NotNull IntervalSet ambigAlts,
                                                                @NotNull ATNConfigSet configs)
    {
    }

        @Override
        public void reportAttemptingFullContext(@NotNull Parser recognizer,
                                                                                    @NotNull
DFA dfa,
                                                                                    int
startIndex, int stopIndex,
                                                                                    @NotNull
ATNConfigSet configs)
        {
        }

        @Override
    public void reportContextSensitivity(@NotNull Parser recognizer, @NotNull DFA dfa,
                                         int    startIndex,    int    stopIndex,    @NotNull
ATNConfigSet configs)
    {
    }
=======
>>>>>>> 3f1f76df7d44332c637e5a92f27933e9c9f3e5ac
}
```

```
    }
}
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/DiagnosticErrorListener.java

Chunk 69: (new code/ class signature, import declaration)

```
import org.antlr.v4.runtime.misc.NotNull;

<<<<<<< HEAD:runtime/Java/src/org/antlr/v4/runtime/DiagnosticErrorStrategy.java
public class DiagnosticErrorStrategy extends DefaultErrorStrategy {
=======
import java.util.Arrays;

public class DiagnosticErrorListener extends BaseErrorListener<Token> {
>>>>>>>
3f1f76df7d44332c637e5a92f27933e9c9f3e5ac:runtime/Java/src/org/antlr/v4/runtime/DiagnosticErr
orListener.java
    @Override
```

```
import org.antlr.v4.runtime.misc.NotNull;

public class DiagnosticErrorListener extends BaseErrorListener<Token> {
    @Override
```

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/ParserATNSimulator.java

Chunk 70: (version 1/method declaration)

```
    }
<<<<<<< HEAD
=======

    public  void  reportInsufficientPredicates(@NotNull  DFA  dfa,  int  startIndex,  int
stopIndex,
                                                                                  @NotNull
IntervalSet ambigAlts,

DecisionState decState,
                                                                                  @NotNull
SemanticContext[] altToPred,
                                                                                  @NotNull
ATNConfigSet configs,
                                                                                  boolean
fullContextParse)
    {
        if ( debug || retry_debug ) {
            System.out.println("reportInsufficientPredicates "+
                               ambigAlts+",                          decState="+decState+":
"+Arrays.toString(altToPred)+
                               parser.getInputString(startIndex, stopIndex));
        }
        if ( parser!=null ) {
            parser.getErrorListenerDispatch().reportInsufficientPredicates(parser,      dfa,
startIndex, stopIndex, ambigAlts,

                                decState, altToPred, configs, fullContextParse);
        }
    }

>>>>>>> 3f1f76df7d44332c637e5a92f27933e9c9f3e5ac
}
```

```
    }
}
```

Version: 48d663667ba2fd0fdd9c3dfdc305d0e987862bc2

Parents:

85b40c7d2efd53de3d4cfa90445aefe06fb2d02c

2232ea5101daddfba2e53bb4e07104f298f65a85

Merge base:

ea7037dd2dff6c36b358b0a641c197d499367c0c

antlr4/runtime/Java/src/org/antlr/v4/runtime/atn/LL1Analyzer.java

Chunk 71: (new code/ if statement, method invocation, variable)

```
        for (int i=0; i<n; i++) {
<<<<<<< HEAD
            Transition t = s.transition(i);
            if ( t.getClass() == RuleTransition.class ) {
                              PredictionContext newContext =
                    new SingletonPredictionContext(ctx, s.stateNumber);
                _LOOK(t.target, newContext, look, lookBusy, seeThruPreds);
            }
            else if ( t.isEpsilon() && seeThruPreds ) {
                _LOOK(t.target, ctx, look, lookBusy, seeThruPreds);
            }
            else if ( t.getClass() == WildcardTransition.class ) {
                look.addAll( IntervalSet.of(Token.MIN_USER_TOKEN_TYPE, atn.maxTokenType) );
            }
            else {
                              System.out.println("adding "+ t);
                IntervalSet set = t.label();
                if (set != null) {
                    if (t instanceof NotSetTransition) {
                        set    =    set.complement(IntervalSet.of(Token.MIN_USER_TOKEN_TYPE,
atn.maxTokenType));
                    }
                    look.addAll(set);
                }
            }
        }
    }
=======
                      Transition t = s.transition(i);
                      if ( t.getClass() == RuleTransition.class ) {
                              RuleContext newContext =
                              new RuleContext(ctx, s.stateNumber);
                              _LOOK(t.target, newContext, look, lookBusy, seeThruPreds);
                      }
                      else if ( t instanceof PredicateTransition ) {
                              if ( seeThruPreds ) {
                                      _LOOK(t.target, ctx, look, lookBusy, seeThruPreds);
                              }
                      }
                      else if ( t.isEpsilon() ) {
                              _LOOK(t.target, ctx, look, lookBusy, seeThruPreds);
                      }
                      else if ( t.getClass() == WildcardTransition.class ) {
                              look.addAll(        IntervalSet.of(Token.MIN_USER_TOKEN_TYPE,
atn.maxTokenType) );
```

```
                        }
                        else {
//                              System.out.println("adding "+ t);
                                IntervalSet set = t.label();
                                if (set != null) {
                                        if (t instanceof NotSetTransition) {
                                                set                                =
set.complement(IntervalSet.of(Token.MIN_USER_TOKEN_TYPE, atn.maxTokenType));
                                        }
                                        look.addAll(set);
                                }
                        }
                }
        }
>>>>>>> 2232ea5101daddfba2e53bb4e07104f298f65a85

}
```

```
        for (int i=0; i<n; i++) {
                        Transition t = s.transition(i);
                        if ( t.getClass() == RuleTransition.class ) {
                                PredictionContext newContext =
                                        new SingletonPredictionContext(ctx, s.stateNumber);
                                _LOOK(t.target, newContext, look, lookBusy, seeThruPreds);
                        }
                        else if ( t instanceof PredicateTransition ) {
                                if ( seeThruPreds ) {
                                        _LOOK(t.target, ctx, look, lookBusy, seeThruPreds);
                                }
                        }
                        else if ( t.isEpsilon() ) {
                                _LOOK(t.target, ctx, look, lookBusy, seeThruPreds);
                        }
                        else if ( t.getClass() == WildcardTransition.class ) {
                                look.addAll(         IntervalSet.of(Token.MIN_USER_TOKEN_TYPE,
atn.maxTokenType) );
                        }
                        else {
                                System.out.println("adding "+ t);
                                IntervalSet set = t.label();
                                if (set != null) {
                                        if (t instanceof NotSetTransition) {
                                                set                                =
set.complement(IntervalSet.of(Token.MIN_USER_TOKEN_TYPE, atn.maxTokenType));
                                        }
                                        look.addAll(set);
                                }
                        }
                }
        }
}
```

Version: 1a2094b2ddf798b43b4f5a00db023965d8a120ab

Parents:

7287f5a2d3719f992f34bfea5071c8d7d9c16ab5

27806dc4906135a4a8adca06947009320808b7bc

Merge base:

f426e8781ba84f340714b7db2d848fbe3bb8a528

antlr4/tool/test/org/antlr/v4/test/TestPerformance.java

Chunk 72: (version 2/ import declaration)

```
import org.antlr.v4.runtime.misc.Nullable;
<<<<<<< HEAD
import org.antlr.v4.runtime.tree.*;
import org.junit.*;
=======
import org.antlr.v4.runtime.tree.ParseTree;
import org.antlr.v4.runtime.tree.ParseTreeListener;
import org.antlr.v4.runtime.tree.ParseTreeWalker;
import org.junit.Assert;
import org.junit.Ignore;
import org.junit.Test;
>>>>>>> 27806dc4906135a4a8adca06947009320808b7bc

import java.io.*;
```

```
import org.antlr.v4.runtime.misc.Nullable;
import org.antlr.v4.runtime.tree.ParseTree;
import org.antlr.v4.runtime.tree.ParseTreeListener;
import org.antlr.v4.runtime.tree.ParseTreeWalker;
import org.junit.Assert;
import org.junit.Ignore;
import org.junit.Test;

import java.io.*;
```

Version: 27806dc4906135a4a8adca06947009320808b7bc

Parents:

9e192fe71ac3d505e4b8c1bb16f87ee445146829

f426e8781ba84f340714b7db2d848fbe3bb8a528

Merge base:

46094f57ba083f31b327a00e7938ced0829f97e9

antlr4/tool/test/org/antlr/v4/test/TestActionTranslation.java

Chunk 73: (version 2/ variable)

```
    @Test public void testRuleRefs() throws Exception {
<<<<<<< HEAD
        String action = "$lab.start;";
               String expected = "(_localctx.lab!=null?(_localctx.lab.start):null);";
=======
        String action = "$lab.start; $c.text;";
               String                               expected                                =
"(((aContext)_localctx).lab!=null?(((aContext)_localctx).lab.start):null);
(((aContext)_localctx).c!=null?_input.toString(((aContext)_localctx).c.start,((aContext)_loc
alctx).c.stop):null);";
>>>>>>> f426e8781ba84f340714b7db2d848fbe3bb8a528
               testActions(attributeTemplate, "inline", action, expected);
```

```
    @Test public void testRuleRefs() throws Exception {
        String action = "$lab.start; $c.text;";
               String                               expected                                =
"(((aContext)_localctx).lab!=null?(((aContext)_localctx).lab.start):null);
(((aContext)_localctx).c!=null?_input.toString(((aContext)_localctx).c.start,((aContext)_loc
alctx).c.stop):null);";
               testActions(attributeTemplate, "inline", action, expected);
```

antlr4/tool/test/org/antlr/v4/test/TestPerformance.java

Chunk 74: (version 2/ annotation, commentary)

```
    @Test
<<<<<<< HEAD
    @Ignore
=======
    //@Ignore
>>>>>>> f426e8781ba84f340714b7db2d848fbe3bb8a528
    public void compileJdk() throws IOException {
```

```
   @Test
    //@Ignore
    public void compileJdk() throws IOException {
```