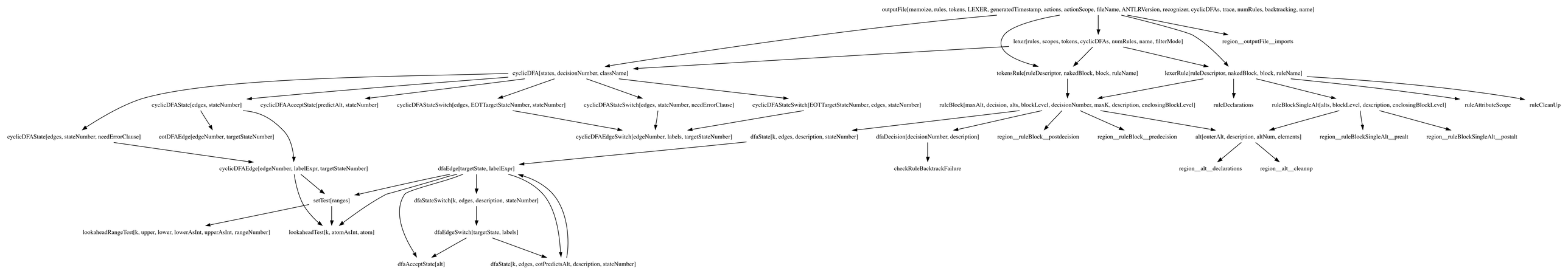
outputFile[memoize, rules, tokens, LEXER, generatedTimestamp, actions, actionScope, fileName, ANTLRVersion, recognizer, cyclicDFAs, trace, numRules, backtracking, name]
lexer[rules, scopes, tokens, cyclicDFAs, numRules, name, filterMode]
region__outputFile__imports
cyclicDFA[states, decisionNumber, className]
tokensRule[ruleDescriptor, nakedBlock, block, ruleName]
lexerRule[ruleDescriptor, nakedBlock, block, ruleName]
cyclicDFAState[edges, stateNumber]
cyclicDFAAcceptState[predictAlt, stateNumber]
cyclicDFAStateSwitch[edges, EOTTargetStateNumber, stateNumber]
cyclicDFAStateSwitch[edges, stateNumber, needErrorClause]
cyclicDFAStateSwitch[EOTTargetStateNumber, edges, stateNumber]
ruleBlock[maxAlt, decision, alts, blockLevel, decisionNumber, maxK, description, enclosingBlockLevel]
ruleDeclarations
ruleBlockSingleAlt[alts, blockLevel, description, enclosingBlockLevel]
ruleAttributeScope
ruleCleanUp
cyclicDFAState[edges, stateNumber, needErrorClause]
eotDFAEdge[edgeNumber, targetStateNumber]
cyclicDFAEdgeSwitch[edgeNumber, labels, targetStateNumber]
dfaState[k, edges, description, stateNumber]
dfaDecision[decisionNumber, description]
region__ruleBlock__postdecision
region__ruleBlock__predecision
alt[outerAlt, description, altNum, elements]
region__ruleBlockSingleAlt__prealt
region__ruleBlockSingleAlt__postalt
cyclicDFAEdge[edgeNumber, labelExpr, targetStateNumber]
dfaEdge[targetState, labelExpr]
checkRuleBacktrackFailure
region__alt__declarations
region__alt__cleanup
setTest[ranges]
dfaStateSwitch[k, edges, description, stateNumber]
lookaheadRangeTest[k, upper, lower, lowerAsInt, upperAsInt, rangeNumber]
lookaheadTest[k, atomAsInt, atom]
dfaEdgeSwitch[targetState, labels]
dfaAcceptState[alt]
dfaState[k, edges, eotPredictsAlt, description, stateNumber]