EBNF of Mini Java cfg

Program ⇒ (ClassDecl)* eot

ClassDecl ⇒ class id { (Vis Access (Type id (FieldDecl | MethodDecl)
| void id MethodDecl))* }

FieldDecl ⇒ ;

MethodDecl ⇒ (ParamList ?) { Stmnt* }

Vis ⇒ (public | private) ?

Access ⇒ static ?

Type ⇒ boolean | (int | id) (ε | [ ])

ParamList ⇒ Type id (, Type id)*

ArgList ⇒ Expr (, Expr)*

Ref ⇒ (id | this) (. id)*

Stmnt ⇒ { Stmnt* } | Type id = Expr ;
| Ref (= Expr ; | [Expr] = Expr ; | (ArgList ?) ;)
| return Expr ? ;
| if (Expr) Stmnt (else Stmnt) ?
| while (Expr) Stmnt

Expr ⇒ (
Ref ([Expr] | (ArgList ?) | ε)
| unop Expr
| (Expr) | num | true | false
| new (id (( ) | [Expr]) | int [Expr])
) (binop Expr)*

Transformations made to the grammar

ClassDecl ⟹ class id { (FieldDecl | MethodDecl)* }

FieldDecl ⟹ Vis Access Type id ;

left factor

MethodDecl ⟹ Vis Access (Type | void) id (ParamList ?) { Stmnt* }

left factor

Type ⟹ int | boolean | id | (int | id) [ ]

Stmnt ⟹ { Stmnt* } | Type id = Expr ;

left factor

| Ref = Expr ;
| Ref [Expr] = Expr ;
| Ref (ArgList ?) ;
| return Expr ? ;
| if (Expr) Stmnt (else Stmnt) ?
| while (Expr) Stmnt

Expr ⟹ Ref left factor

| Ref [Expr]
| Ref (ArgList ?)
| unop Expr
| Expr binop Expr left recursion
| (Expr)

Ref ⟹ id | this | Ref . id left recursion