JavaScript Analysis Framework (JSAF)

Programming Language Research Group
(PL알지!) @ KAIST
원 순 철

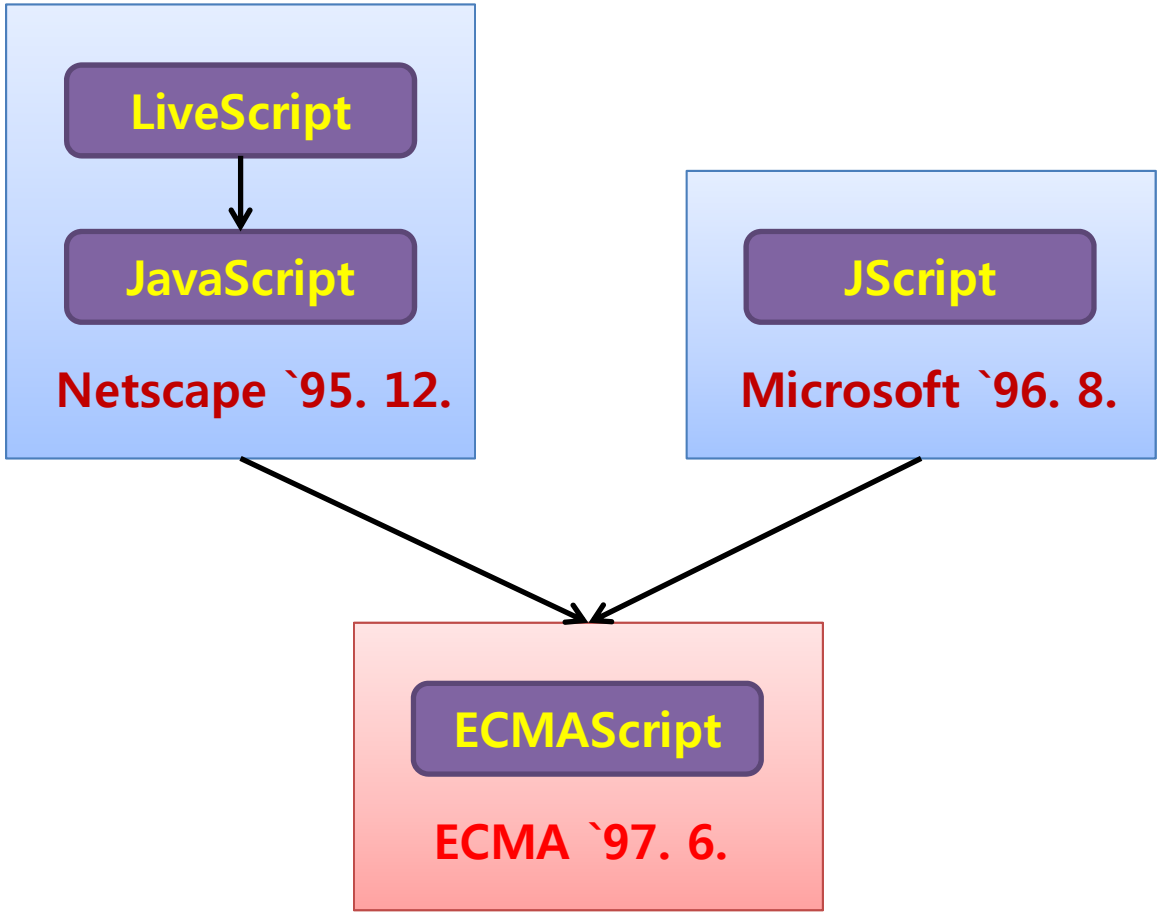
목 차

1. JavaScript 소개

JavaScript 소개

- 웹 브라우저에서 스크립트 언어로 사용
  - 정적인 웹 페이지로부터 탈피
- 쉽게 소스 코드를 볼 수 있음
- 기존의 언어들과 다른 점이 많음
- 언어의 명세를 ECMA에서 관리

JavaScript 소개



European **C**omputer **M**anufacturers **A**ssociation

2. JSAF 개요

1) JSAF 소개
2) 개발 이유
3) JSAF 특징
4) JSAF 활용

JSAF 소개

- JSAF란?
  - **JavaScript Analysis Framework**

1) 수행 분석

- Control Flow Analysis & Data Flow Analysis
- Code Coverage

2) 코드 최적화

- Unused Variable Elimination
- Dead Code Elimination

JSAF 소개

- JSAF란?
  - **JavaScript Analysis Framework**

3) 오류 확인

- 기본적인 JavaScript 문법 오류 확인
- Strict Mode 체크 (ECMAScript5)

4) 기타

- Clone Detector
- Interpreter

개발 이유

- 기존 툴들이 대부분 ECMAScript3를 활용
- ECMAScript의 부분 언어를 활용
- 부족한 도구 및 불필요한 작업의 반복
- 향후 JavaScript 연구에 사용될 도구로써 개발 시작

JSAF 특징

- ECMAScript5 지원
- 간결한 정형명세 문서를 제공
  - ast2ir, ir, ir2cfg, cfg
- AST를 생성하기 위해 자동화 도구를 사용
  - Rats!, ASTGen
- Java와 Scala를 이용하여 구현

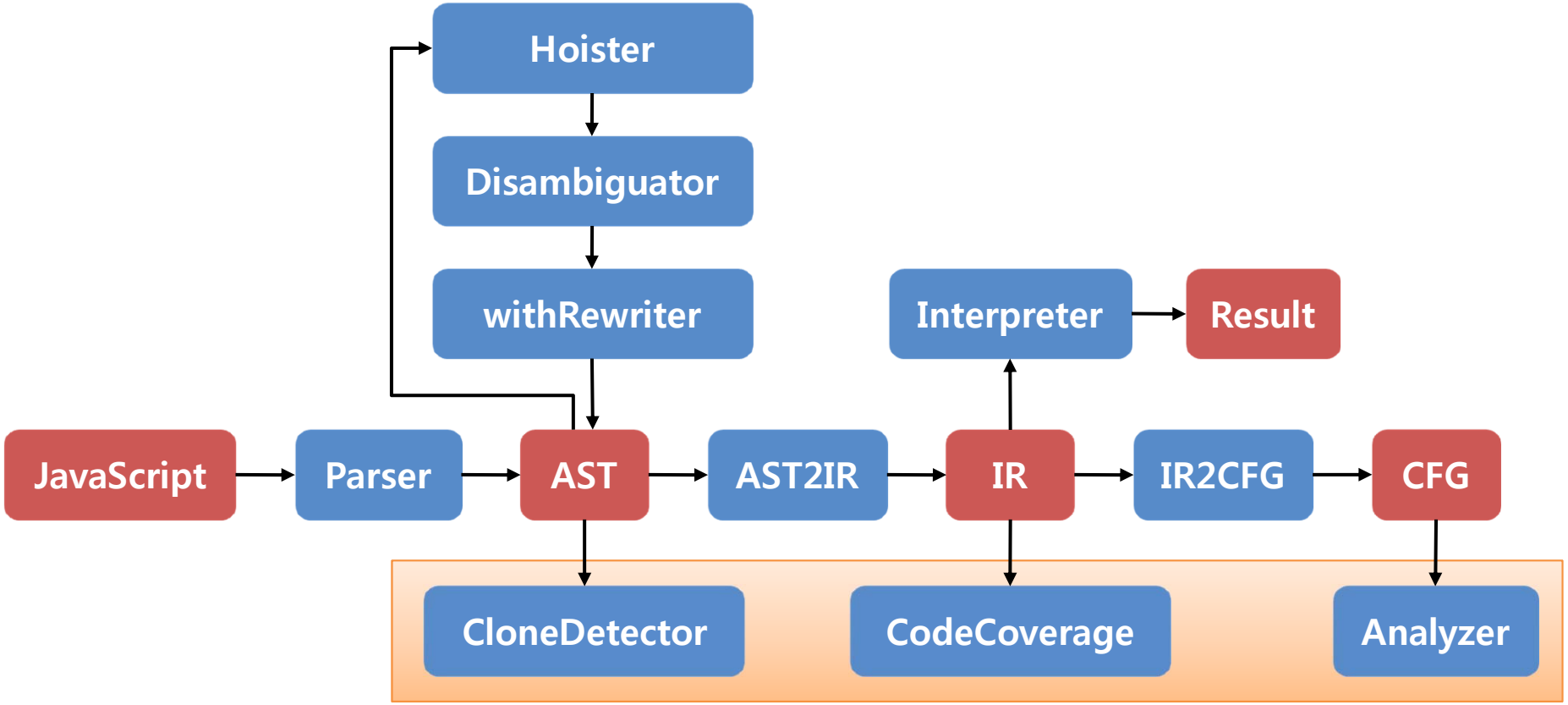
JSAF 활용

- JavaScript의 문법부터 의미구조까지 자유롭게 확장/변경하는 연구에 활용 가능
- 다양한 JavaScript 분석 기법을 컴파일러의 여러 단계에서 동시에 연구에 활용 가능
- JavaScript 분석 프레임워크를 open source로 다른 연구팀과 공유

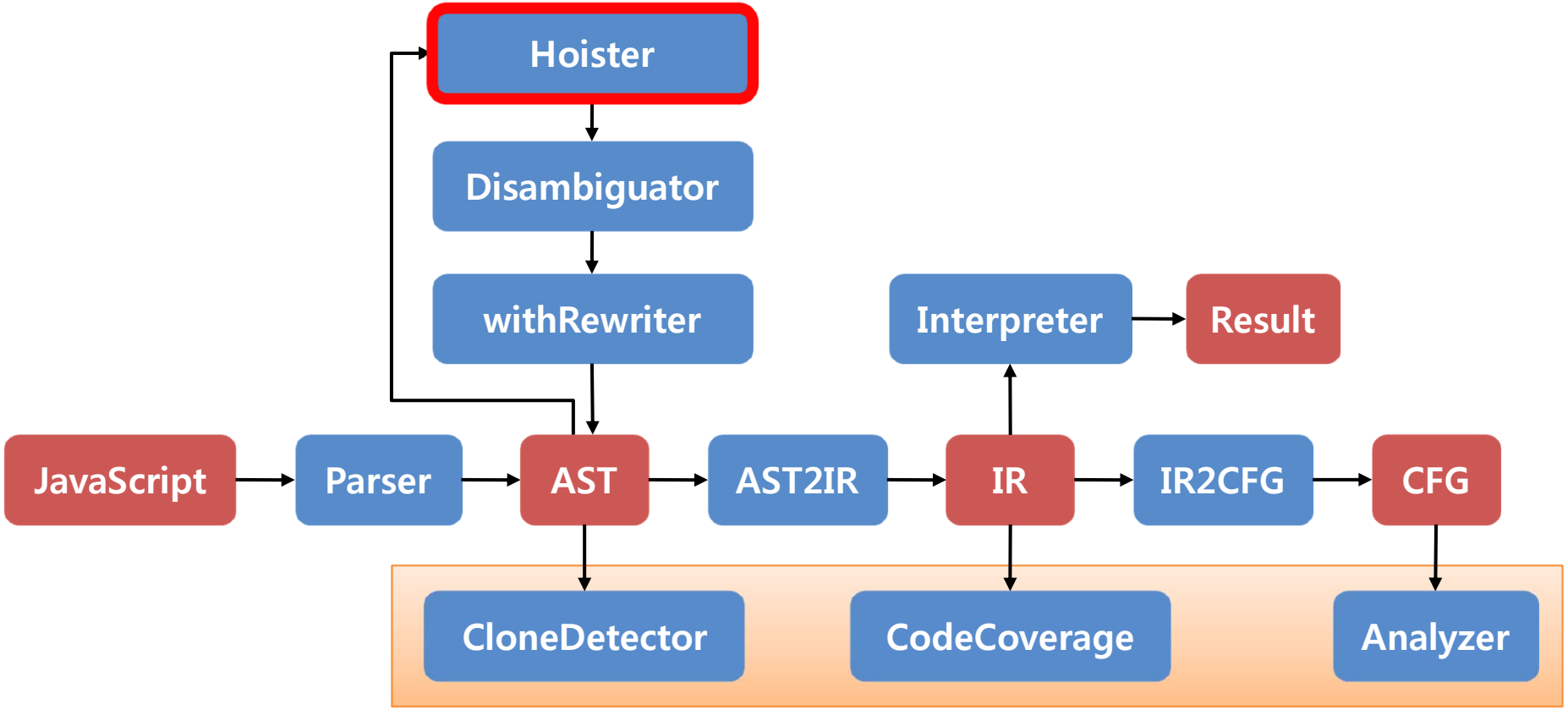
3. JSAF 구현

1) JSAF Flow Graph
2) 전반부 주요 구현
   (1) Hoister
   (2) Disambiguator
   (3) with Rewriter
   (4) AST to IR
3) 후반부 주요 구현
   (1) IR
   (2) Interpreter
   (3) IR to CFG
   (4) Analyzer

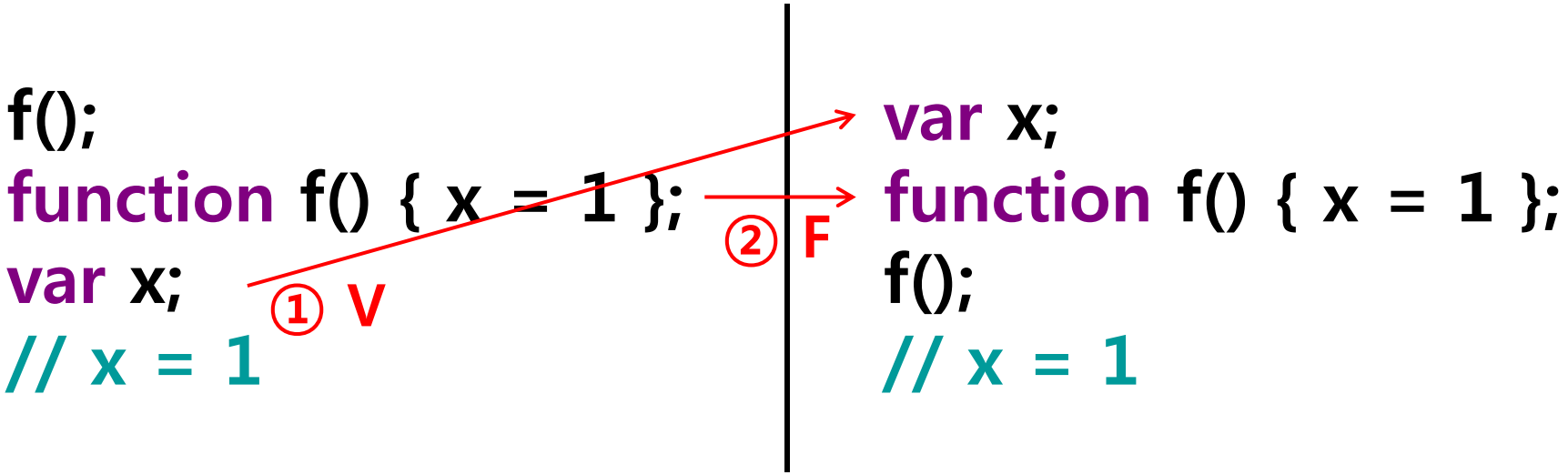
JSAF Flow Graph



Hoister


Hoister
Disambiguator
withRewriter
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
CloneDetector
CodeCoverage
Analyzer

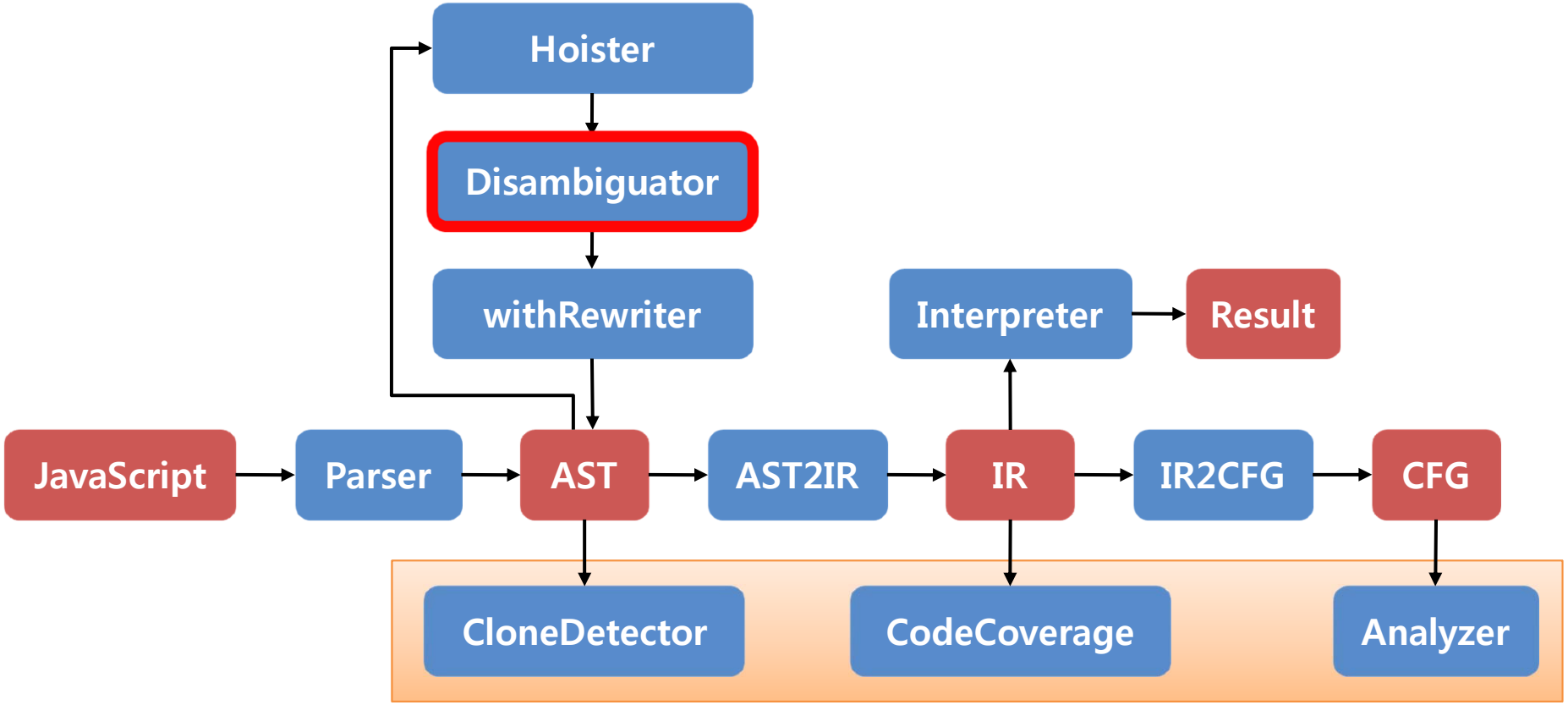
Hoister



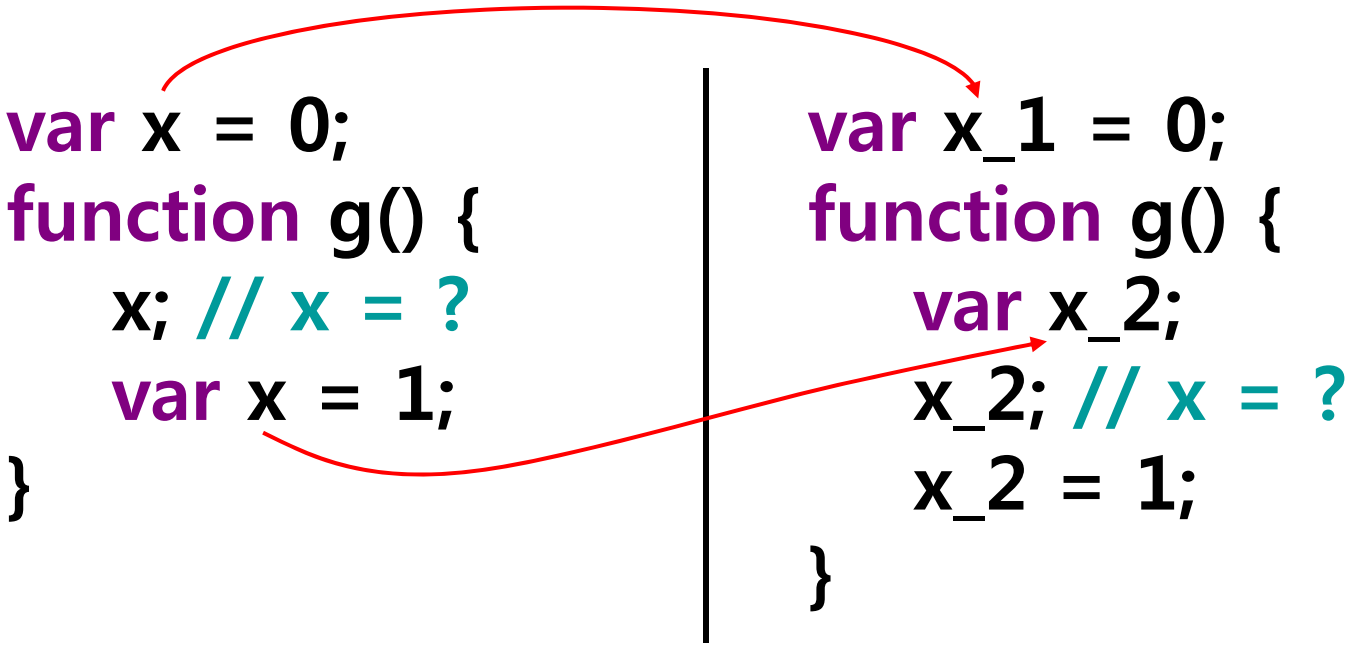
ECMAScript 문서에 기술되어 있음

다른 언어들과 동일하게, 선언된 뒤에 사용하게 됨

Disambiguator



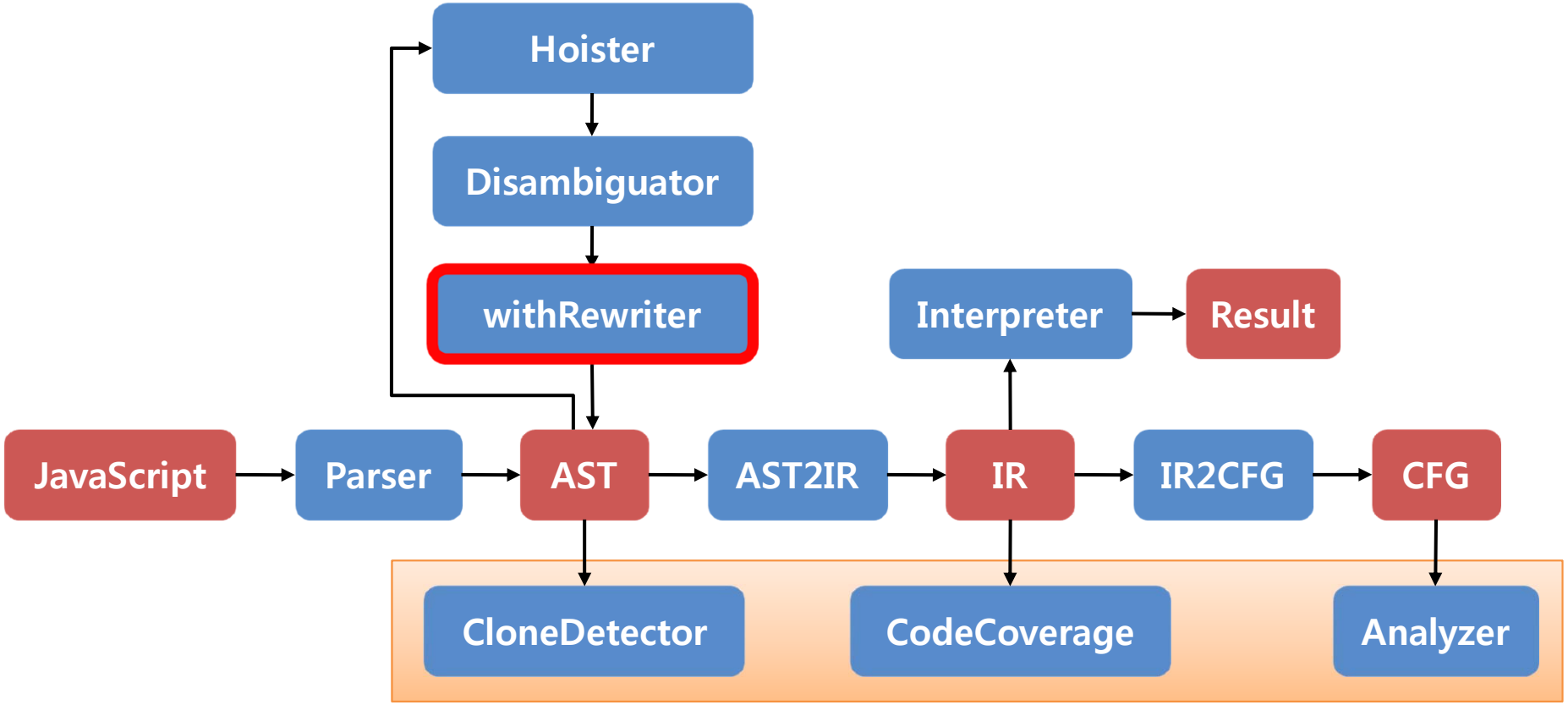
Disambiguator



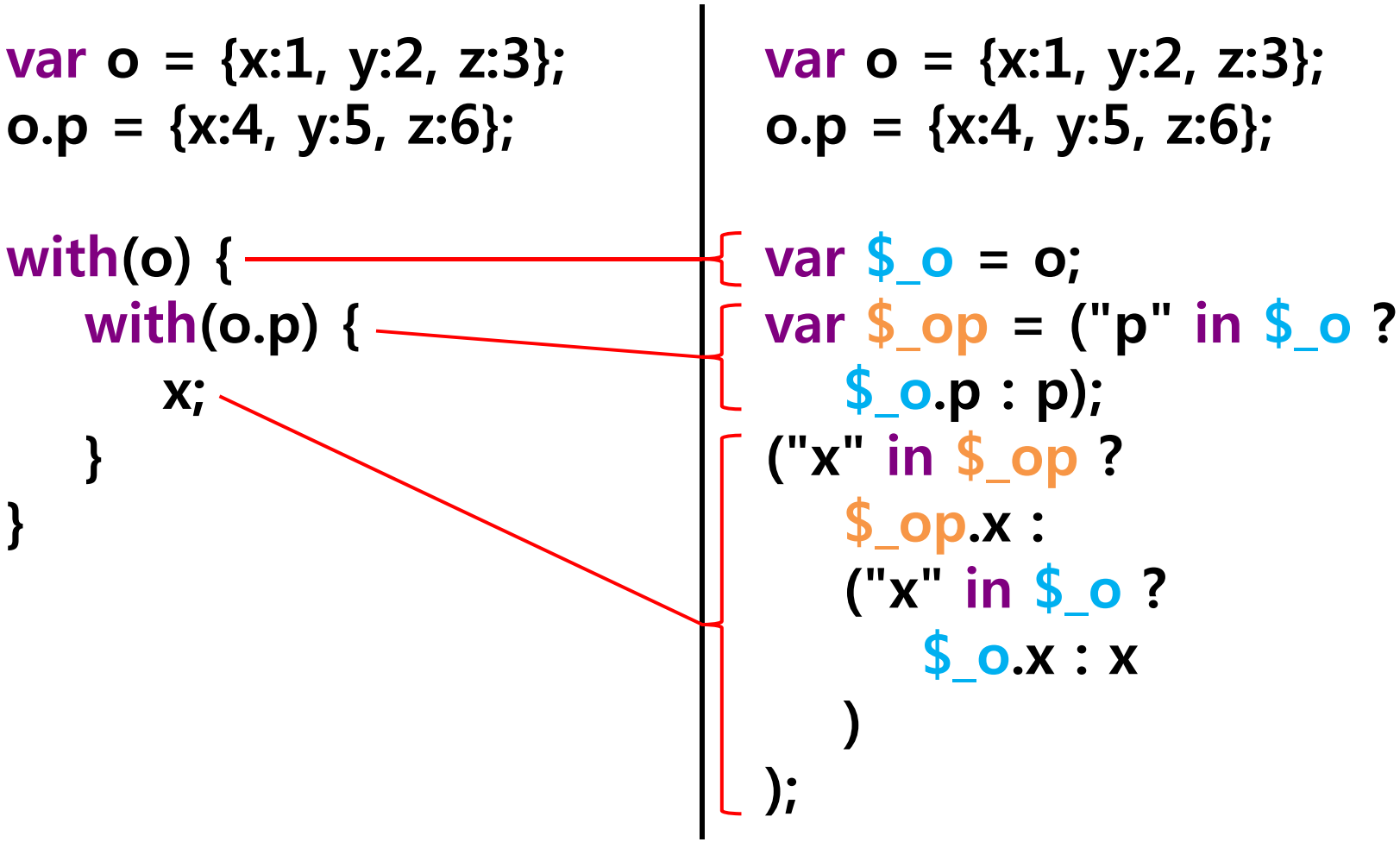
"undefined"

변수명이 유일하기 때문에 바로 구별 가능

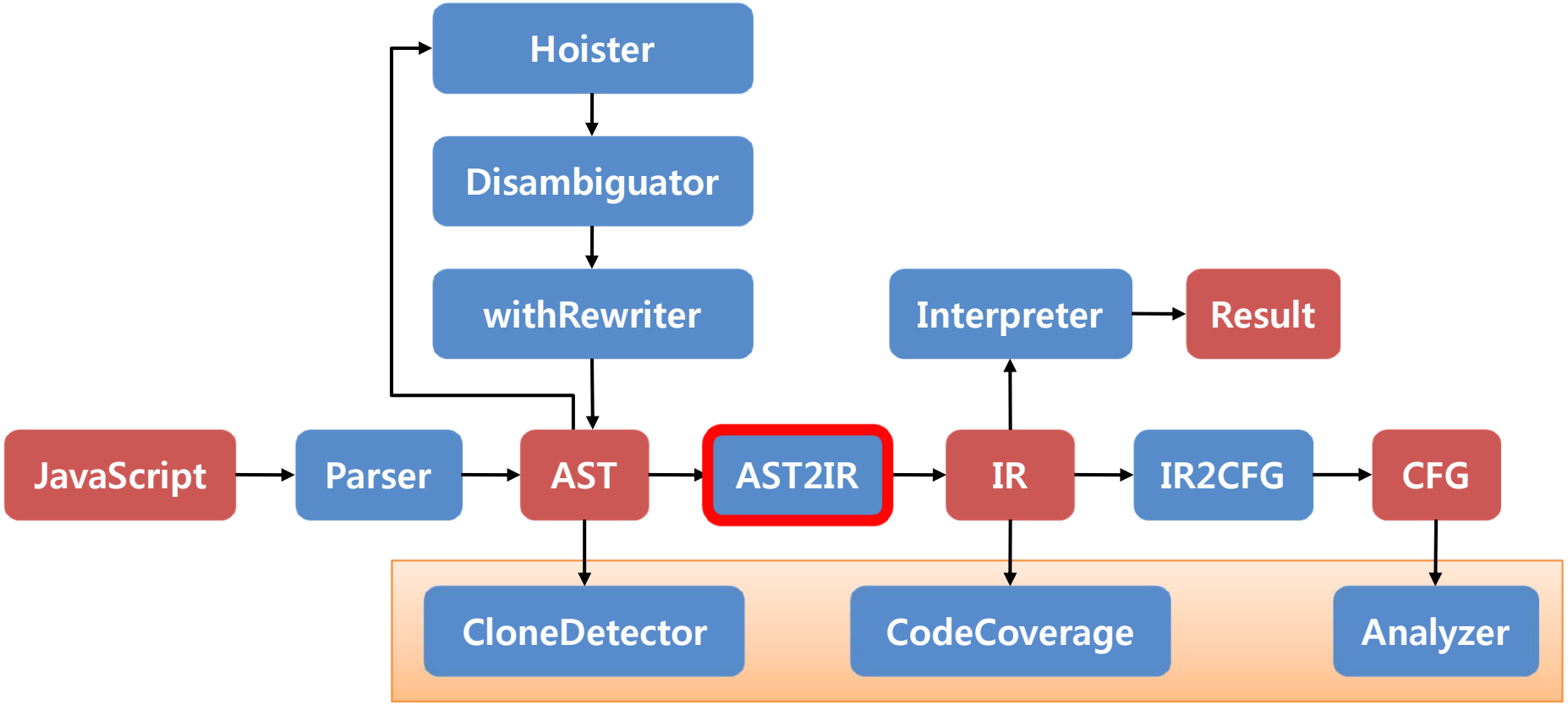
with Rewriter



with Rewriter



AST to IR


Hoister
Disambiguator
withRewriter
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
CloneDetector
CodeCoverage
Analyzer

AST to IR

- 더 작은 언어로 변환 (케이스의 간소화)
  - switch-case ➔ if-then-else
  - 여섯 종류의 iteration ➔ while
  - break & continue ➔ break
  - obj.prop & obj[“prop”] ➔ obj[“prop”]
  - x++ ➔ x = x + 1
  - ……
- 변환된 IR도 동일한 의미구조를 가짐

AST to IR

- AST

1. AST

p ::=vd^* fd^* s^*	Program(TopLevel body) TopLevel(List<VarDecl> vds, List<FunDecl> fds, List<SourceElement> program)
vd ::=var x	VarDecl(Id name, Option<Expr> expr)
fd ::=function f $((x,)^*)$ {vd^* fd^* s^*}	FunDecl(Id name, Functional ftn) Functional(List<VarDecl> vds, List<FunDecl> fds, List<SourceElement> program, List<Id> params)
s ::={s^*}	Block(List<Stmt> stmts)
| var $vd(,vd)^*$;	VarStmt(List<VarDecl> vds)
| ;	EmptyStmt()
| e;	ExprStmt(Expr expr)
| if (e) s (else s)$^?$	If(Expr cond, Stmt trueBranch, Option<Stmt> falseBranch)
| switch (e) {cc^* (default:s^*)$^?$ cc^*}	Switch(Expr cond, List<Case> frontCases, Option<List<Stmt>> def, List<Case> backCases)
| do s while (e);	DoWhile(Stmt body, Expr cond)
| while (e) s	While(Expr cond, Stmt body)
| for ($e^?$;$e^?$;$e^?$) s	For(Option<Expr> init, Option<Expr> cond, Option<Expr> action, Stmt body)
| for (lhs in e) s	ForIn(LHS lhs, Expr expr, Stmt body)
| for (var $vd(,vd)^*$;$e^?$;$e^?$) s	ForVar(List<VarDecl> vars, Option<Expr> cond, Option<Expr> action, Stmt body)
| for (var vd in e) s	ForVarIn(VarDecl var, Expr expr, Stmt body)
| continue $id^?$;	Continue(Option<Label> target)
| break $id^?$;	Break(Option<Label> target)
| return $e^?$;	Return(Option<Expr> expr)
| with (e) s	With(Expr expr, Stmt stmt)
| l : s	LabelStmt(Label label, Stmt stmt)
| throw e;	Throw(Expr expr)
| try{s^*}(catch(id){s^*})$^?$(finally{s^*})$^?$	Try(Block body, Option<Catch> catchBlock, Option<Block> fin)
| debugger ;	Catch(Id id, Block body) Debugger()
cc ::=case e : s^*	Case(Expr cond, Block body)
e ::=e, e	ExprList(List<Expr> exprs)
| e ? e : e	Cond(Expr cond, Expr trueBranch, Expr falseBranch)
| e ⊗ e	InfixOpApp(Expr left, Op op, Expr right)
| ⊖ e	PrefixOpApp(Op op, Expr right)
| lhs ⊘	UnaryAssignOpApp(LHS lhs, Op op)
| lhs	LHS()
| lhs ⊙ e	AssignOpApp(LHS lhs, Op op, Expr right)
lhs::=lit	Literal()
| id	VarRef(Id id)
| [$(e,)^*$]	ArrayExpr(List<Expr> elements)
| {$(m,)^*$}	ObjectExpr(List<Member> members)
| (e)	Parenthesized(Expr expr)
| function $id^?$ $((id,)^*)$ {vd^* fd^* s^*}	FunExpr(Option<Id> name, Functional ftn)
| lhs[e]	Bracket(LHS obj, Expr index)
| lhs.id	Dot(LHS obj, Id member)
| new lhs	New(LHS lhs)
| lhs($(e,)^*$)	FunApp(LHS fun, List<Expr> args)
lit ::=this	This()
| null	Null()
| true	Bool(boolean bool)
| false	Bool(boolean bool)
| num	DoubleLiteral(ignoreForEquals String text, Double num), IntLiteral(BigInteger intVal, int radix)
| str	StringLiteral(String str, String quote)
| reg	RegularExpression(String reg)
m ::=id : e	Field(Id prop, Expr expr)
| get id () {vd^* fd^* s^*}	GetProp(Id name, Functional ftn)
| set id (id) {vd^* fd^* s^*}	SetProp(Id name, Functional ftn)

- IR

2. Syntax

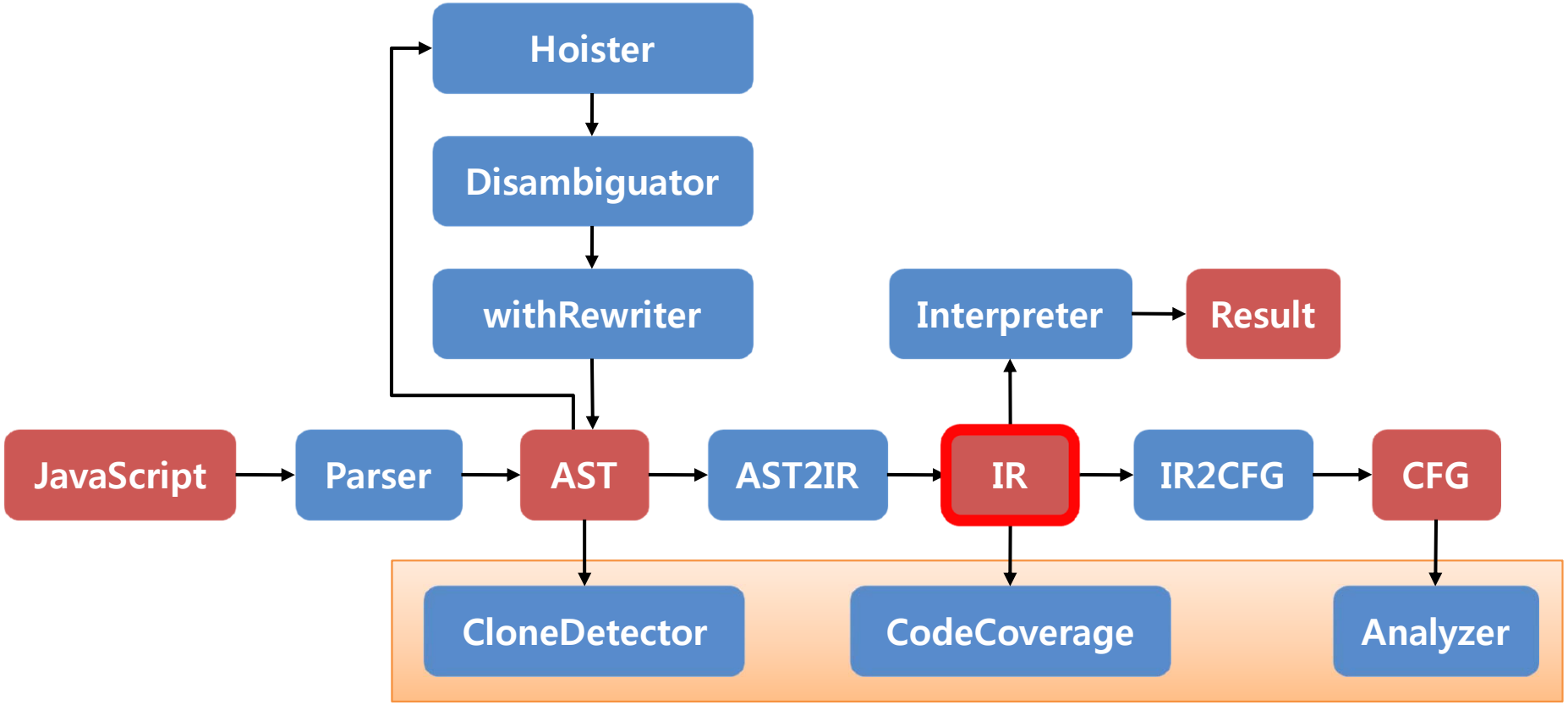
p ::= s^* s^* s^*	IRRoot(List<IRVarStmt> vds, List<IRFunDecl> fds, List<IRStmt> irs)
s ∈ *Stmt*	
s ::= x = e	IRExprStmt(IRId lhs, IRExpr right)
| x = x instanceof x	IRBin(IRId lhs, IRId first, IROp op, IRId second)
| x = x in x	
| x = x ⊗ x	
| x = typeof x	IRUn(IRId lhs, IROp op, IRId id)
| x = void x	
| x = ⊖ x	
| x = delete x	IRDelete(IRId lhs, IRId id)
| x = delete x[x]	IRDeleteProp(IRId lhs, IRId obj, IRId index)
| x = x[e]	IRLoad(IRId lhs, IRId obj, IRExpr index)
| x[x] = x	IRStore(IRId obj, IRId index, IRId rhs)
| x = {$(m,)^*$}	IRObject(IRId lhs, List<IRMember> members)
| x = [$(x,)^*$]	IRArray(IRId lhs, List<IRId> elements)
| x = x($(x,)^*$)	IRCall(IRId lhs, IRId fun, List<IRId> args)
| x = new x($(x,)^*$)	IRNew(IRId lhs, IRId fun, List<IRId> args)
| x = function f($this$,$arguments$) {s}	IRFunExpr(IRId lhs, IRFunctional ftn) IRFunctional(IRId name, List<IRId> thisArguments, List<IRStmt> params, List<IRVarStmt> vds, List<IRFunDecl> fds, List<IRStmt> stmts)
| function f($this$,$arguments$) {s}	IRFunDecl(IRFunctional ftn)
| x = eval(e)	IREval(IRId lhs, IRExpr arg)
| break x	IRBreak(IRLabel label)
| return $x^?$	IRReturn(Option<IRId> id)
| with (x) s	IRWith(IRId id, IRStmt stmt)
| x : { s }	IRLabelStmt(IRLabel label, IRStmt stmt)
| var x	IRVarStmt(IRId lhs)
| throw x	IRThrow(IRId id)
| s^*	IRSeq(List<IRStmt> stmts)
| if (x) then s (else s)$^?$	IRIf(IRId id, IRStmt trueB, Option<IRStmt> falseB)
| while (x) s	IRWhile(IRId cond, IRStmt body)
| try {s} (catch (x){s})$^?$ (finally {s})$^?$	IRTry(IRStmt body, Option<IRId> name, Option<IRStmt> catchB, Option<IRStmt> finallyB)
e ::= this	IRThis()
| x	IRVarRef(IRId id)
| loc	IRLoc(String text)
| undefined	IRUndef()
| null	IRNull()
| true	IRBool(boolean bool)
| false	IRBool(boolean bool)
| n	IRDouble(ignoreForEquals String text, Double num),IRInt(BigInteger intVal)
| s	IRString(String str)
m ::= x : x	IRField(IRId prop, IRId id)
| get f($this$,$arguments$) {s}	IRGetProp(IRFunctional ftn)
| set f($this$,$arguments$) {s}	IRSetProp(IRFunctional ftn)

AST to IR

- AST to IR 변환 규칙

$$
\begin{array}{l}
ast2ir_s[\![\texttt{for }(e_1^?;e_2;e_3^?)\ s]\!](\Sigma,\Gamma) = \langle \diamond\text{break} : \{\ \text{IRSeq}(\langle \\
\quad (ast2ir_e[\![e_1]\!](\Sigma,\Gamma)(\diamond\_);)^? \\
\quad ast2ir_e[\![e_2]\!](\Sigma,\Gamma)(\diamond\text{new}_2); \\
\quad \diamond\text{new}_3 = \diamond\text{toBoolean}(\diamond\text{new}_2); \\
\quad \text{while } (\diamond\text{new}_3)\ \text{IRSeq}(\langle \\
\quad\quad \diamond\text{continue} : \{\ \text{IRSeq}(\langle ast2ir_s[\![s]\!](\Sigma;\diamond\text{break};\diamond\text{continue},\Gamma)\rangle)\ \}; \\
\quad\quad (ast2ir_e[\![e_3]\!](\Sigma,\Gamma)(\diamond\text{new}_3);)^? \\
\quad\quad ast2ir_e[\![e_2]\!](\Sigma,\Gamma)(\diamond\text{new}_2); \\
\quad\quad \diamond\text{new}_3 = \diamond\text{toBoolean}(\diamond\text{new}_2); \\
\quad \rangle) \\
\rangle)\ \}\rangle \\
ast2ir_s[\![\texttt{continue ;}]\!](\Sigma,\Gamma) = \langle \text{break } \Sigma(\diamond\text{continue})\rangle \\
ast2ir_s[\![\texttt{continue } l;]\!](\Sigma,\Gamma) = \langle \text{break } l\rangle \\
ast2ir_s[\![\texttt{try } \{s_1^*\}\ (\texttt{catch}(x)\{s_2^*\})^?(\texttt{finally } \{s_3^*\})^?]\!](\Sigma,\Gamma) = \langle \text{try } \{\ \text{IRSeq}(\ (ast2ir_s[\![s_1]\!](\Sigma,\Gamma))^*\ )\ \} \\
\quad (\texttt{catch}(x)\{\ \text{IRSeq}(\ (ast2ir_s[\![s_2]\!](\Sigma,\Gamma))^*\ )\ \})^? \\
\quad (\texttt{finally } \{\ \text{IRSeq}(\ (ast2ir_s[\![s_3]\!](\Sigma,\Gamma))^*\ )\ \})^?\rangle
\end{array}
$$

IR


Hoister
Disambiguator
withRewriter
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
CloneDetector
CodeCoverage
Analyzer

IR

- JavaScript

```
var sum = 0;
for(var i = 1; i <= 10; i++)
  sum+= i;
_<>_print(sum);
```

- AST

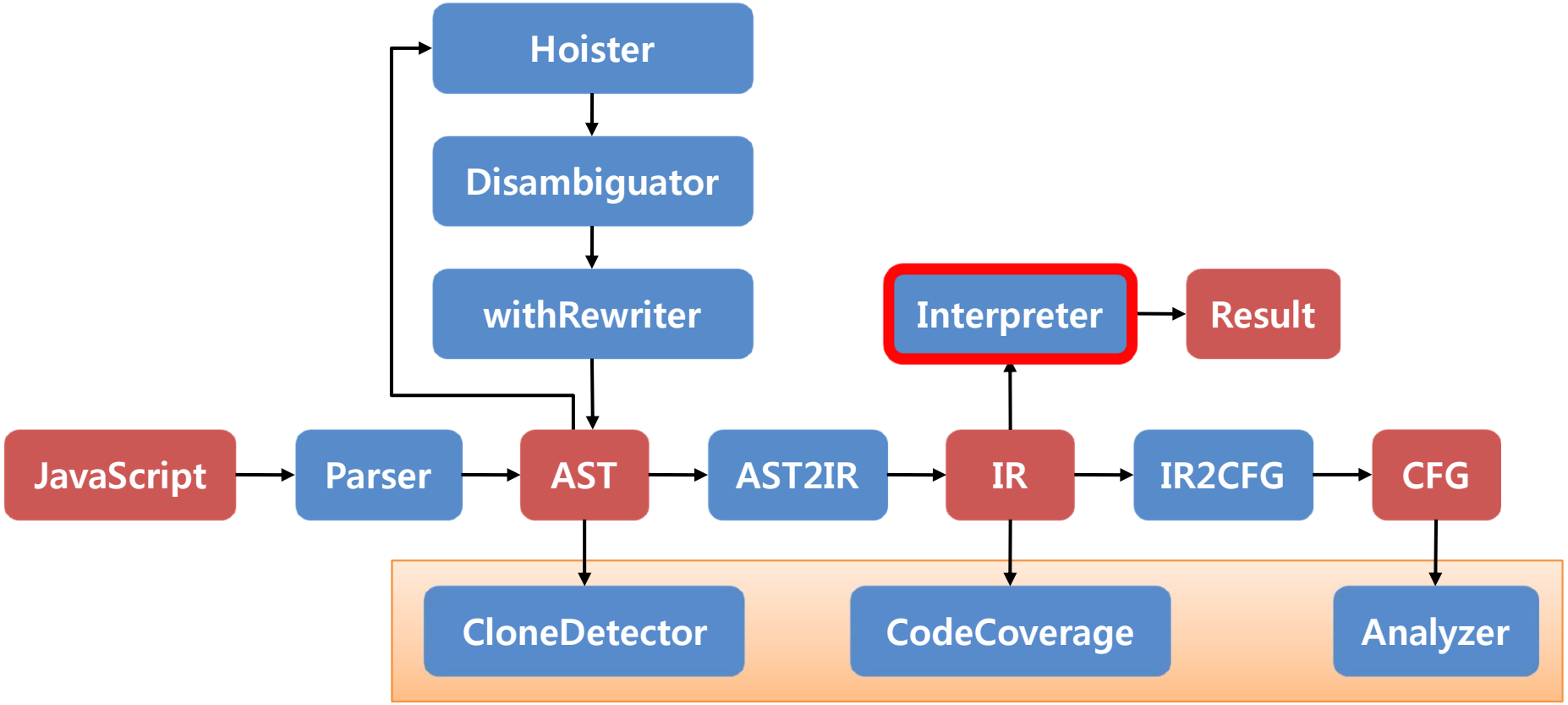
```
(Program @"test.js":1:1~4:16
 _body=(TopLevel
  _stmts=[
   (VarStmt
    _vds=[
     (VarDecl
      _expr=(Some _value=(IntLiteral _intVal=0 _radix=10))
      _name=(Id _text="sum"))])
   (ForVar
    _action=(Some _value=(UnaryAssignOpApp
      _lhs=(VarRef _id=(Id _text="i"))
      _op=(Op
       _text="++")))
    _body=(ExprStmt
     _expr=(AssignOpApp
      _lhs=(VarRef _id=(Id _text="sum"))
      _op=(Op
       _text="+=")
      _right=(VarRef _id=(Id _text="i")))
     _cond=(Some _value=(InfixOpApp
      _left=(VarRef _id=(Id _text="i"))
      _op=(Op
       _text="<=")
      _right=(IntLiteral _intVal=10 _radix=10)))
     _vars=[
      (VarDecl
       _expr=(Some _value=(IntLiteral _intVal=1 _radix=10))
       _name=(Id _text="i"))])
   (ExprStmt
    _expr=(FunApp
     _fun=(VarRef _id=(Id _text="_<>_print"))
     _args=[
      (VarRef _id=(Id _text="sum"))])
```

IR

- IR

```
(IRRoot @"test.js":1:1~3:16
 _irs=[
  (IRExprStmt @"<>global":0:0
   _lhs=(IRId _temp=true _text="<>_one_202957491637263")
   _right=(IRNumber _num=1.0 _text="1"))
  (IRStmtUnit @"test.js":1:5~10
   _stmts=[
    (IRSeq
     _stmts=[
      (IRExprStmt
       _lhs=(IRId _temp=true _text="<>_old_202957496278583")
       _right=(IRId _text="sum"))
      (IRExprStmt @1:11
       _lhs=(IRId _temp=true _text="<>_temp_202957496268018")
       _right=(IRNumber _num=0.0 _text="0"))
      (IRExprStmt @1:5~10
       _lhs=(IRId _text="sum")
       _right=(IRId _temp=true _text="<>_temp_202957496268018"))
      (IRExprStmt
       _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
       _right=(IRId _text="sum"))])])
  (IRStmtUnit @2:9~12
   _stmts=[
    (IRStmtUnit
     _stmts=[
      (IRSeq
       _stmts=[
        (IRExprStmt
         _lhs=(IRId _temp=true _text="<>_old_202957497658965")
         _right=(IRId _text="i"))
        (IRExprStmt @2:13
         _lhs=(IRId _temp=true _text="<>_temp_202957497650814")
         _right=(IRNumber _num=0.0 _text="0"))
        (IRExprStmt @2:9~12
         _lhs=(IRId _text="i")
         _right=(IRId _temp=true _text="<>_temp_202957497650814"))
        (IRExprStmt
         _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
         _right=(IRId _text="i"))])])
    (IRStmtUnit @2:1~34
     _stmts=[
      (IRLabelStmt
       _label=(IRId _temp=true _text="<>_break_202957497742276")
       _stmt=(IRSeq
        _stmts=[
         (IRSeq
         (IRSeq @2:16~20
          _stmts=[
           (IRExprStmt @2:16
            _lhs=(IRId _temp=true _text="<>_temp_202957500141117")
            _ref=true
            _right=(IRId _text="i"))
           (IRExprStmt @2:20
            _lhs=(IRId _temp=true _text="<>_temp_202957500151983")
            _right=(IRNumber _num=10.0 _text="10"))
           (IRBin @2:16~20
            _first=(IRId _temp=true _text="<>_temp_202957500141117")
            _lhs=(IRId _temp=true _text="<>_temp_202957500116063")
            _op=?kr.ac.kaist.jsaf.nodes.IROp
            _second=(IRId _temp=true _text="<>_temp_202957500151983"))])
          (IRCall @2:1~34
           _fun=(IRId _temp=true _text="<>toBoolean")
           _lhs=(IRId _temp=true _text="<>_temp_202957500125118")
           _args=[
            (IRId _temp=true _text="<>_temp_202957500116063")])
          (IRWhile
           _body=(IRSeq
            _stmts=[
             (IRLabelStmt
              _label=(IRId _temp=true _text="<>_temp_202957497751332")
              _stmt=(IRStmtUnit @2:29~34
               _stmts=[
                (IRSeq
                 _stmts=[
                  (IRExprStmt
                   _lhs=(IRId _temp=true _text="<>_old_202957499548274")
                   _right=(IRId _text="sum"))
                  (IRExprStmt @2:35
                   _lhs=(IRId _temp=true _text="<>_temp_202957499555820")
                   _ref=true
                   _right=(IRId _text="i"))
                  (IRBin @2:29~34
                   _first=(IRId _temp=true _text="<>_old_202957499548274")
                   _lhs=(IRId _temp=true _text="<>_new_202957499563065")
                   _op=?kr.ac.kaist.jsaf.nodes.IROp
                   _second=(IRId _temp=true _text="<>_temp_202957499555820"))
                  (IRExprStmt
                   _lhs=(IRId _text="sum")
                   _right=(IRId _temp=true _text="<>_new_202957499563065"))
                  (IRExprStmt
                   _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
                   _right=(IRId _text="sum"))])])
             (IRSeq @2:23~25
              _stmts=[
               (IRSeq @2:23
                _stmts=[
                 (IRExprStmt
                  _lhs=(IRId _temp=true _text="<>_old_202957497785140")
                  _right=(IRId _text="i"))
                 (IRCall @2:23~25
                  _fun=(IRId _temp=true _text="<>toNumber")
                  _lhs=(IRId _temp=true _text="<>_new_202957497794196")
                  _args=[
                   (IRId _temp=true _text="<>_old_202957497785140")])
                 (IRBin
                  _first=(IRId _temp=true _text="<>_new_202957497794196")
                  _lhs=(IRId _temp=true _text="<>_new2_202957497802647")
                  _op=?kr.ac.kaist.jsaf.nodes.IROp
                  _second=(IRId _temp=true _text="<>_one_202957491637263"))
                 (IRExprStmt @2:23
                  _lhs=(IRId _text="i")
                  _right=(IRId _temp=true _text="<>_new2_202957497802647"))
                 (IRExprStmt
                  _lhs=(IRId _temp=true _text="<>_temp_202957497757973")
                  _right=(IRId _text="i"))])
                (IRExprStmt @2:23~25
                 _lhs=(IRId _temp=true _text="<>_temp_202957497757973")
                 _right=(IRId _temp=true _text="<>_new_202957497794196"))])
               (IRSeq @2:16~20
                _stmts=[
                 (IRExprStmt @2:16
                  _lhs=(IRId _temp=true _text="<>_temp_202957500141117")
                  _ref=true
                  _right=(IRId _text="i"))
                 (IRExprStmt @2:20
                  _lhs=(IRId _temp=true _text="<>_temp_202957500151983")
                  _right=(IRNumber _num=10.0 _text="10"))
                 (IRBin @2:16~20
                  _first=(IRId _temp=true _text="<>_temp_202957500141117")
                  _lhs=(IRId _temp=true _text="<>_temp_202957500116063")
                  _op=?kr.ac.kaist.jsaf.nodes.IROp
                  _second=(IRId _temp=true _text="<>_temp_202957500151983"))])
                (IRCall @2:1~34
                 _fun=(IRId _temp=true _text="<>toBoolean")
                 _lhs=(IRId _temp=true _text="<>_temp_202957500125118")
                 _args=[
                  (IRId _temp=true _text="<>_temp_202957500116063")])])
               _cond=(IRId _temp=true _text="<>_temp_202957500125118")
              (IRStmtUnit @3:1~13
               _stmts=[
                (IRSeq @3:1~12
                 _stmts=[
                  (IRExprStmt @3:11~12
                   _lhs=(IRId _temp=true _text="<>_temp_202957500634046")
                   _ref=true
                   _right=(IRId _text="sum"))
                  (IRCall @3:1~12
                   _fun=(IRId _temp=true _text="<>print")
                   _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
                   _args=[
                    (IRId _temp=true _text="<>_temp_202957500634046")])])])]
          _vds=[
           (IRVarStmt @1:1~3:16
            _lhs=(IRId _text="sum"))
           (IRVarStmt
            _lhs=(IRId _text="i"))])
```

Interpreter



Interpreter

- IR 코드를 실제로 실행시켜 결과를 출력
- IR 명세에 기술된 Semantics에 따라 실행

Ex) The **`while`** Statement

12.6.2 The `while` Statement

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} err}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(err)}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad ToBoolean(v) = \texttt{false}}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H, A), \texttt{Normal}(\texttt{empty})}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad ToBoolean(v) = \texttt{true} \qquad (H, A, tb), \underline{s} \rightarrow_{\underline{s}} (H', A'), ac}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H', A'), ac}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad ToBoolean(v) = \texttt{true} \qquad (H, A, tb), \underline{s} \rightarrow_{\underline{s}} (H', A'), nc \qquad (H', A', tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), ac}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), ac}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad ToBoolean(v) = \texttt{true} \qquad (H, A, tb), \underline{s} \rightarrow_{\underline{s}} (H', A'), nc \qquad (H', A', tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(\texttt{empty})}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), nc}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad ToBoolean(v) = \texttt{true} \qquad (H, A, tb), \underline{s} \rightarrow_{\underline{s}} (H', A'), nc \qquad (H', A', tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(v)}{(H, A, tb), \text{while } (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(v)}$$

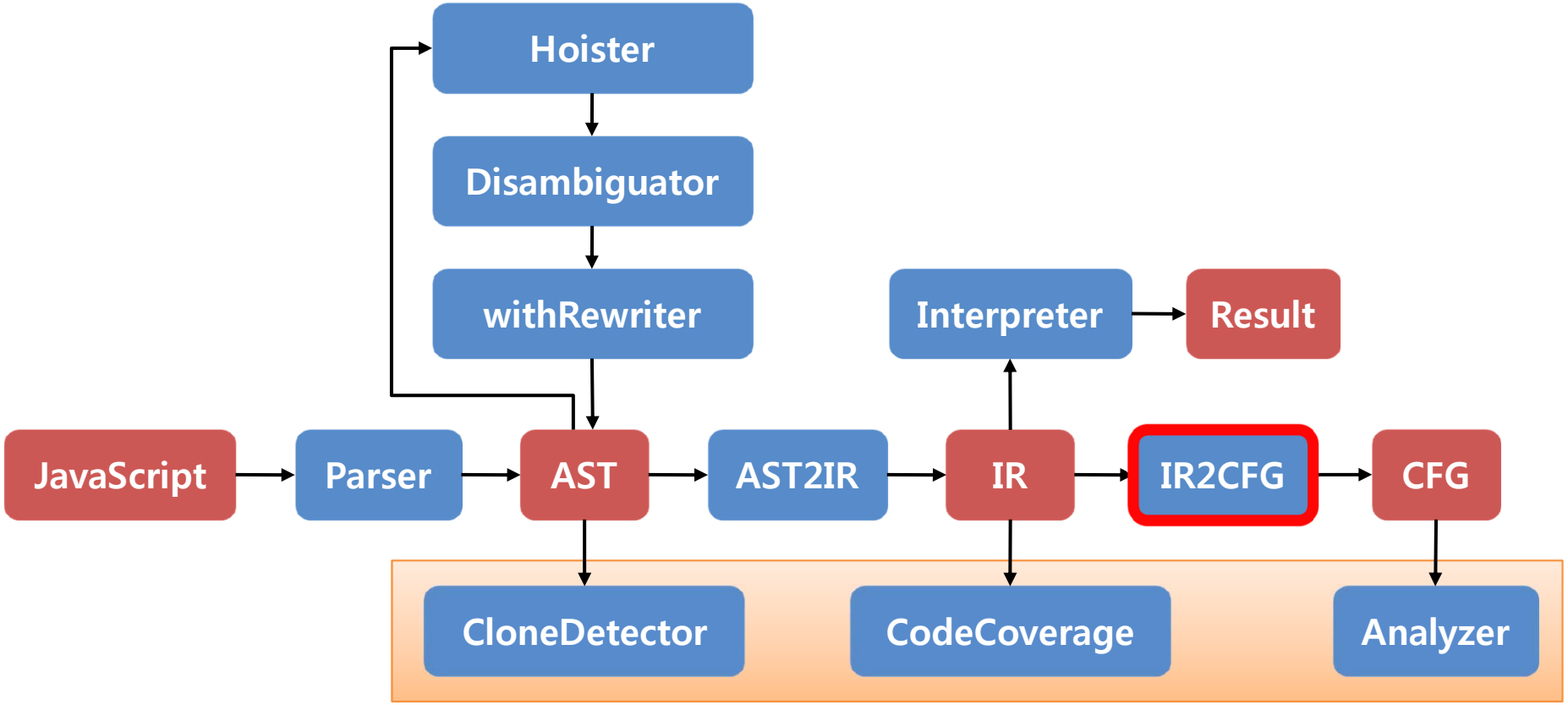
Interpreter

- IR

```
(IRRoot @"test.js":1:1~3:16
 _irs=[
  (IRExprStmt @"<>global":0:0
   _lhs=(IRId _temp=true _text="<>_one_202957491637263")
   _right=(IRNumber _num=1.0 _text="1"))
  (IRStmtUnit @"test.js":1:5~10
   _stmts=[
    (IRSeq
     _stmts=[
      (IRExprStmt
       _lhs=(IRId _temp=true _text="<>_old_202957496278583")
       _right=(IRId _text="sum"))
      (IRExprStmt @1:11
       _lhs=(IRId _temp=true _text="<>_temp_202957496268018")
       _right=(IRNumber _num=0.0 _text="0"))
      (IRExprStmt @1:5~10
       _lhs=(IRId _text="sum")
       _right=(IRId _temp=true _text="<>_temp_202957496268018"))
      (IRExprStmt
       _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
       _right=(IRId _text="sum"))]))]
  (IRStmtUnit @2:9~12
   _stmts=[
    (IRStmtUnit
     _stmts=[
      (IRSeq
       _stmts=[
        (IRExprStmt
         _lhs=(IRId _temp=true _text="<>_old_202957497658965")
         _right=(IRId _text="i"))
        (IRExprStmt @2:13
         _lhs=(IRId _temp=true _text="<>_temp_202957497650814")
         _right=(IRNumber _num=0.0 _text="0"))
        (IRExprStmt @2:9~12
         _lhs=(IRId _text="i")
         _right=(IRId _temp=true _text="<>_temp_202957497650814"))
        (IRExprStmt
         _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
         _right=(IRId _text="i"))])])
    (IRStmtUnit @2:1~34
     _stmts=[
      (IRLabelStmt
       _label=(IRId _temp=true _text="<>_break_202957497742276")
       _stmt=(IRSeq
        _stmts=[
         (IRSeq
         (IRSeq @2:16~20
          _stmts=[
           (IRExprStmt @2:16
            _lhs=(IRId _temp=true _text="<>_temp_202957500141117")
            _ref=true
            _right=(IRId _text="i"))
           (IRExprStmt @2:20
            _lhs=(IRId _temp=true _text="<>_temp_202957500151983")
            _right=(IRNumber _num=10.0 _text="10"))
           (IRBin @2:16~20
            _first=(IRId _temp=true _text="<>_temp_202957500141117")
            _lhs=(IRId _temp=true _text="<>_temp_202957500116063")
            _op=?kr.ac.kaist.jsaf.nodes.IROp
            _second=(IRId _temp=true _text="<>_temp_202957500151983"))])
          (IRCall @2:1~34
           _fun=(IRId _temp=true _text="<>toBoolean")
           _lhs=(IRId _temp=true _text="<>_temp_202957500125118")
           _args=[
            (IRId _temp=true _text="<>_temp_202957500116063")])
          (IRWhile
           _body=(IRSeq
            _stmts=[
             (IRLabelStmt
              _label=(IRId _temp=true _text="<>_temp_202957497751332")
              _stmt=(IRStmtUnit @2:29~34
               _stmts=[
                (IRSeq
                 _stmts=[
                  (IRExprStmt
                   _lhs=(IRId _temp=true _text="<>_old_202957499548274")
                   _right=(IRId _text="sum"))
                  (IRExprStmt @2:35
                   _lhs=(IRId _temp=true _text="<>_temp_202957499555820")
                   _ref=true
                   _right=(IRId _text="i"))
                  (IRBin @2:29~34
                   _first=(IRId _temp=true _text="<>_old_202957499548274")
                   _lhs=(IRId _temp=true _text="<>_new_202957499563065")
                   _op=?kr.ac.kaist.jsaf.nodes.IROp
                   _second=(IRId _temp=true _text="<>_temp_202957499555820"))
                  (IRExprStmt
                   _lhs=(IRId _text="sum")
                   _right=(IRId _temp=true _text="<>_new_202957499563065"))
                  (IRExprStmt
                   _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
                   _right=(IRId _text="sum"))])]))
             (IRSeq @2:23~25
              _stmts=[
               (IRSeq @2:23
                _stmts=[
                 (IRExprStmt
                  _lhs=(IRId _temp=true _text="<>_old_202957497785140")
                  _right=(IRId _text="i"))
                 (IRCall @2:23~25
                  _fun=(IRId _temp=true _text="<>toNumber")
                  _lhs=(IRId _temp=true _text="<>_new_202957497794196")
                  _args=[
                   (IRId _temp=true _text="<>_old_202957497785140")])
                 (IRBin
                  _first=(IRId _temp=true _text="<>_new_202957497794196")
                  _lhs=(IRId _temp=true _text="<>_new2_202957497802647")
                  _op=?kr.ac.kaist.jsaf.nodes.IROp
                  _second=(IRId _temp=true _text="<>_one_202957491637263"))
                 (IRExprStmt @2:23
                  _lhs=(IRId _text="i")
                  _right=(IRId _temp=true _text="<>_new2_202957497802647"))
                 (IRExprStmt
                  _lhs=(IRId _temp=true _text="<>_temp_202957497757973")
                  _right=(IRId _text="i"))])
                (IRExprStmt @2:23~25
                 _lhs=(IRId _temp=true _text="<>_temp_202957497757973")
                 _right=(IRId _temp=true _text="<>_new_202957497794196"))])
               (IRSeq @2:16~20
                _stmts=[
                 (IRExprStmt @2:16
                  _lhs=(IRId _temp=true _text="<>_temp_202957500141117")
                  _ref=true
                  _right=(IRId _text="i"))
                 (IRExprStmt @2:20
                  _lhs=(IRId _temp=true _text="<>_temp_202957500151983")
                  _right=(IRNumber _num=10.0 _text="10"))
                 (IRBin @2:16~20
                  _first=(IRId _temp=true _text="<>_temp_202957500141117")
                  _lhs=(IRId _temp=true _text="<>_temp_202957500116063")
                  _op=?kr.ac.kaist.jsaf.nodes.IROp
                  _second=(IRId _temp=true _text="<>_temp_202957500151983"))])
                (IRCall @2:1~34
                 _fun=(IRId _temp=true _text="<>toBoolean")
                 _lhs=(IRId _temp=true _text="<>_temp_202957500125118")
                 _args=[
                  (IRId _temp=true _text="<>_temp_202957500116063")])])
               _cond=(IRId _temp=true _text="<>_temp_202957500125118")
              (IRStmtUnit @3:1~13
               _stmts=[
                (IRSeq @3:1~12
                 _stmts=[
                  (IRExprStmt @3:11~12
                   _lhs=(IRId _temp=true _text="<>_temp_202957500634046")
                   _ref=true
                   _right=(IRId _text="sum"))
                  (IRCall @3:1~12
                   _fun=(IRId _temp=true _text="<>print")
                   _lhs=(IRId _temp=true _text="<>_ignore_202957491738384")
                   _args=[
                    (IRId _temp=true _text="<>_temp_202957500634046")])])])]
  _vds=[
   (IRVarStmt @1:1~3:16
    _lhs=(IRId _text="sum"))
   (IRVarStmt
    _lhs=(IRId _text="i"))])
```

- Interpret 결과 55

IR to CFG


Hoister
Disambiguator
withRewriter
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
CloneDetector
CodeCoverage
Analyzer

IR to CFG

- JavaScript

```
var sum = 0;
for(var i = 1; i <= 10; i++)
  sum+= i;
_<>_print(sum);
```

- IR

```
var sum
var i

<>_one_1 = 1
{
  <>_old_2 = sum
  <>_temp_3 = 0
  sum = <>_temp_3
  <>_ignore_4 = sum
}
{
  {
    <>_old_5 = i
    <>_temp_6 = 1
    i = <>_temp_6
    <>_ignore_4 = i
  }
  <>_break_7 : {
    {
      <>_temp_8 = i
      <>_temp_9 = 10
      <>_temp_10 = <>_temp_8 <= <>_temp_9
    }
    <>_temp_11 = <>toBoolean(<>_temp_10)
    while(<>_temp_11)
     {
       <>_temp_12 : {
         <>_old_13 = sum
         <>_temp_14 = i
         <>_new_15 = <>_old_13 + <>_temp_14
         sum = <>_new_15
         <>_ignore_4 = sum
       }
       {
         {
           <>_old_16 = i
           <>_new_17 = <>toNumber(<>_old_16)
           <>_new2_18 = <>_new_17 + <>_one_1
           i = <>_new2_18
           <>_temp_19 = i
         }
         <>_temp_19 = <>_new_17
       }
       {
         <>_temp_8 = i
         <>_temp_9 = 10
         <>_temp_10 = <>_temp_8 <= <>_temp_9
       }
       <>_temp_11 = <>toBoolean(<>_temp_10)
     }
  }
}
{
  <>_temp_20 = sum
  <>_ignore_4 = <>print(<>_temp_20)
}
```

- IR to CFG 변환 규칙

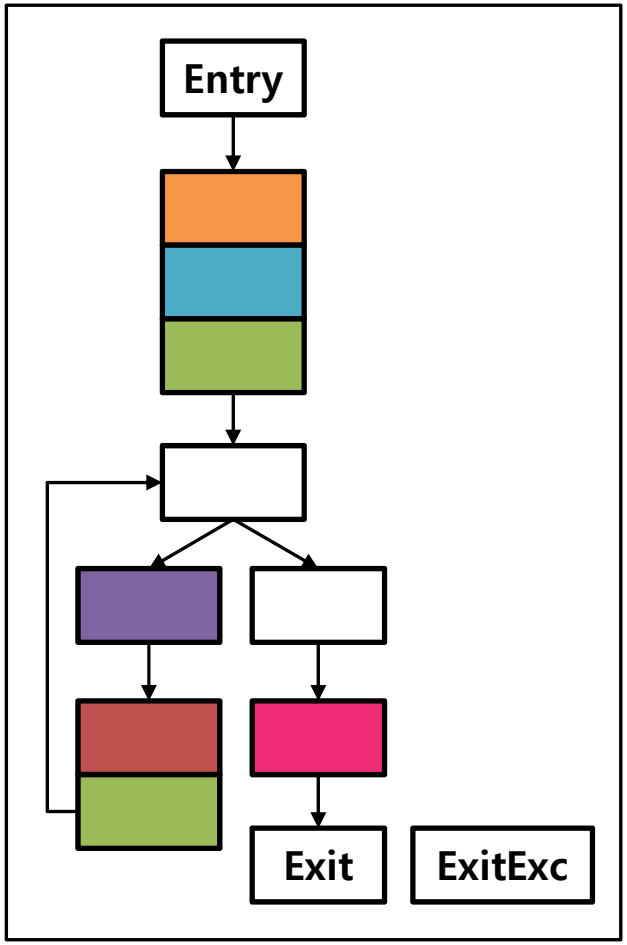
$$
\begin{aligned}
[\![\text{while}(x)\ s]\!]_{stmt}(G,N,L)(fid) = n_1 &\overset{let}{=} GetTail(G,N)(fid)\\
n_{head} &\overset{let}{=} G.NewBlock(fid)\\
n_2 &\overset{let}{=} G.NewBlock(fid)\\
n_3 &\overset{let}{=} G.NewBlock(fid)\\
&G.AddEdge(n_1, n_{head})\\
&G.AddEdge(n_{head}, n_2)\\
&G.AddEdge(n_{head}, n_3)\\
(N_1, L_1) &\overset{let}{=} [\![s]\!]_{stmt}(G,[n_2],L)(fid)\\
&G.AddEdge(N_1, n_{head})\\
&([n_3], L_1)
\end{aligned}
$$

$$
\begin{aligned}
[\![\text{throw } x]\!]_{stmt}(G,N,L)(fid) &= n \overset{let}{=} GetTail(G,N)(fid)\\
&G.AddInst(n, \texttt{throw}(x))\\
&([\,], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}
$$

$$
\begin{aligned}
[\![\text{return } x]\!]_{stmt}(G,N,L)(fid) &= n \overset{let}{=} GetTail(G,N)(fid)\\
&G.AddInst(n, \texttt{return}(x))\\
&([\,], L[\#return \mapsto n :: L(\#return)])
\end{aligned}
$$

IR to CFG

- CFG



- IR

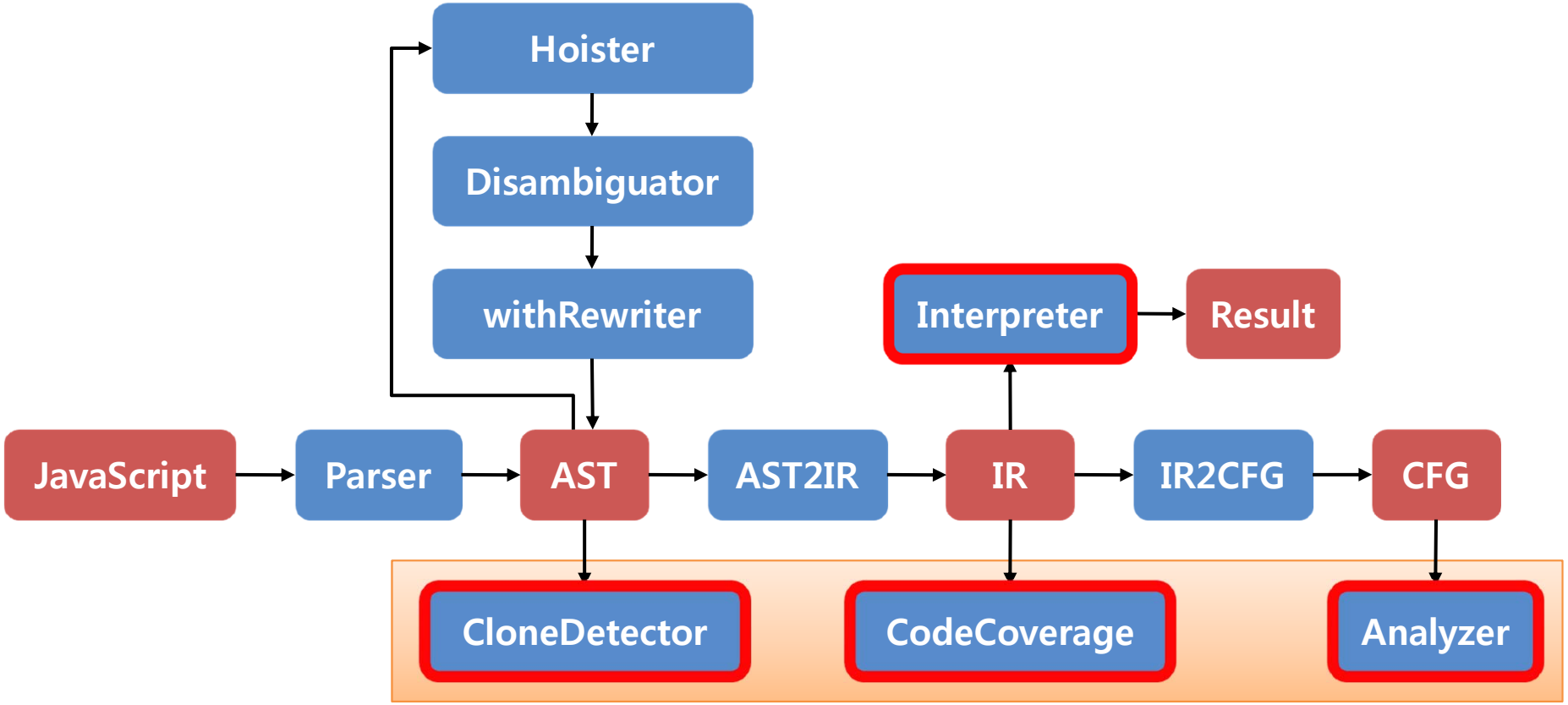
```
var sum
var i

<>_one_1 = 1
{
  <>_old_2 = sum
  <>_temp_3 = 0
  sum = <>_temp_3
  <>_ignore_4 = sum
}
{
  {
    <>_old_5 = i
    <>_temp_6 = 1
    i = <>_temp_6
    <>_ignore_4 = i
  }
  <>_break_7 : {
    {
      <>_temp_8 = i
      <>_temp_9 = 10
      <>_temp_10 = <>_temp_8 <= <>_temp_9
    }
    <>_temp_11 = <>toBoolean(<>_temp_10)
    while(<>_temp_11)
     {
       <>_temp_12 : {
         <>_old_13 = sum
         <>_temp_14 = i
         <>_new_15 = <>_old_13 + <>_temp_14
         sum = <>_new_15
         <>_ignore_4 = sum
       }
       {
        {
          <>_old_16 = i
          <>_new_17 = <>toNumber(<>_old_16)
          <>_new2_18 = <>_new_17 + <>_one_1
          i = <>_new2_18
          <>_temp_19 = i
        }
        <>_temp_19 = <>_new_17
       }
       {
         <>_temp_8 = i
         <>_temp_9 = 10
         <>_temp_10 = <>_temp_8 <= <>_temp_9
       }
       <>_temp_11 = <>toBoolean(<>_temp_10)
     }
  }
}
{
  <>_temp_20 = sum
  <>_ignore_4 = <>print(<>_temp_20)
}
```

4. 기타

1) 진행 상황
2) 정리

진행 상황


Hoister
Disambiguator
withRewriter
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
CloneDetector
CodeCoverage
Analyzer

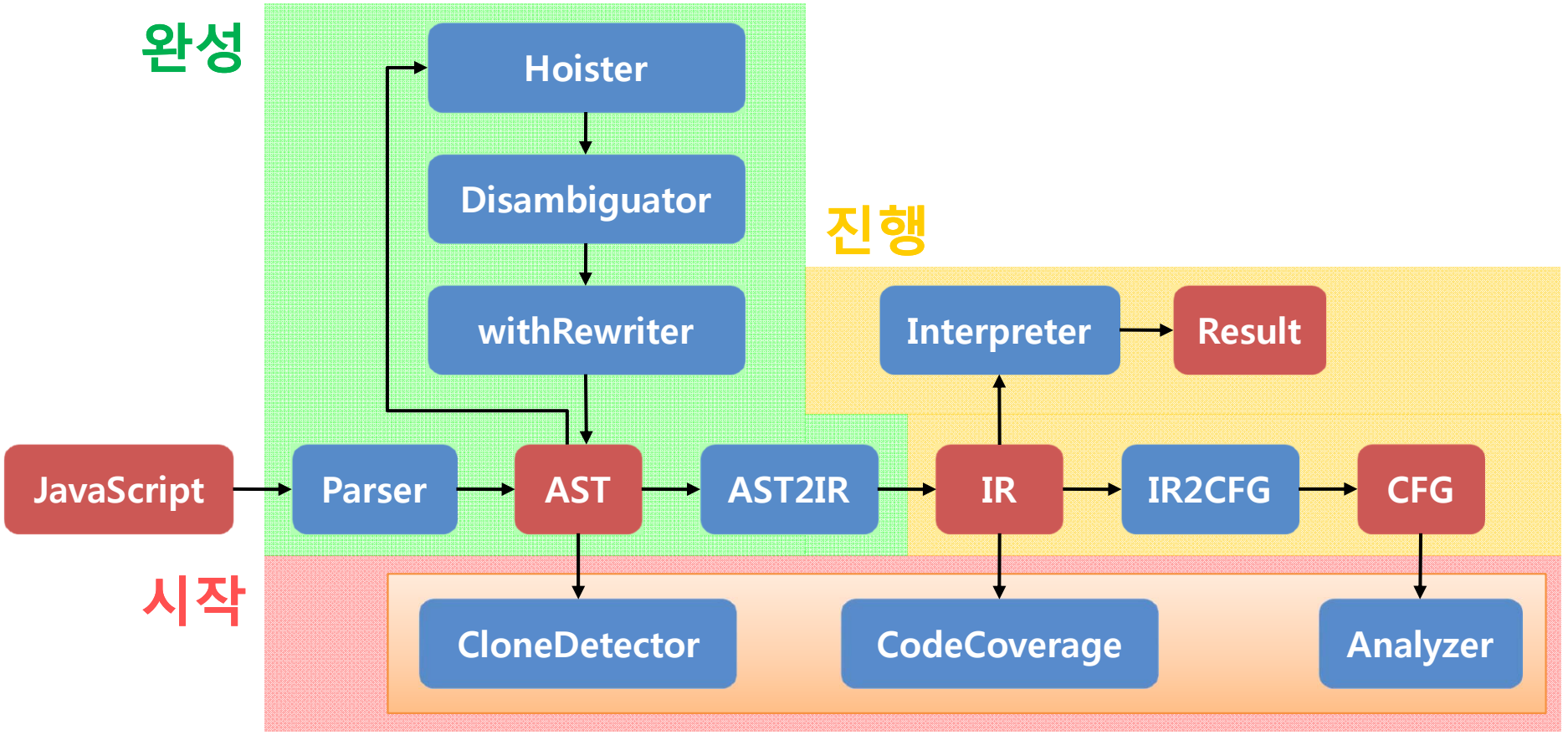
진행 상황

- Interpreter
  - ECMAScript5의 Built-in Object 구현 중
  - 테스트 및 버그 수정
- Clone Detector
  - 간단한 Prototype 구현 및 관련 연구 조사 중
  - 성능 및 정확도 향상 계획

진행 상황

- Code Coverage
  - 기존의 연구들을 조사 중
  - Interpreter 완성 후 구현할 계획
- Analyzer
  - 간단한 Type 분석 기능을 구현 및 테스트 중
  - 분석기 성능 및 정확도 향상 계획
    - Sparse 분석
    - Flow Sensitive 분석
    - Recency 타입 분석

정리


완성
Hoister
Disambiguator
withRewriter
진행
Interpreter
Result
JavaScript
Parser
AST
AST2IR
IR
IR2CFG
CFG
시작
CloneDetector
CodeCoverage
Analyzer

감사합니다!