Discovered *i-rules*

February 2020

1 *i-rule*

This document presents *i-rules* to avoid useless mutants. To understand the concepts behind an *i-rule*, please refer to the article of Fernandes et al.([1]).

Next we present the meta-variables needed to understand the elements of a rule, then we show the rules that avoid equivalent mutants (e-rule), followed by those that avoid duplicate mutants (d-rule).

1.1 Meta-variables

All meta-variables referred by the *i-rules* are depict at Table 1.

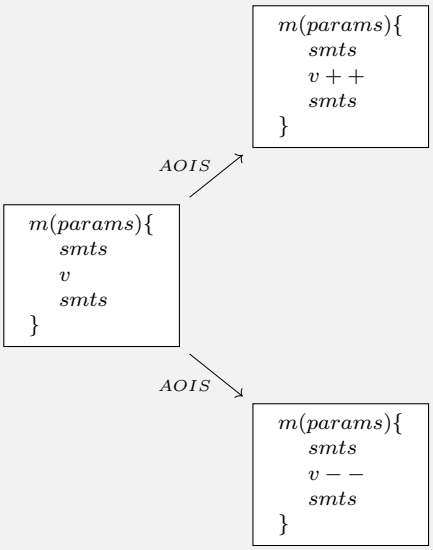
Table 1: Meta-variables referred by the *i-rules*.

Meta-variables	**Description**
aop	any arithmetic operator
dv	the default value of any specified type (i.e., int=0, boolean=false, char='u0000', double=0.0d, float=0.0f, long=0L, Object=null, etc.)
exp	any expression
f	any identifier of a field
m	method declaration or references
n	any number literal
op	any binary or unary operator
params	method parameters
s	any statement
smts	any block of statements
sv	any identifier of a String or Array variable
v	any identifier of a local variable for a primitive integral type (byte, short, int, long)
A, B	class references
;	an empty statement

1.2 E-RULEs

1.2.1 AOIS-01 (MuJava)

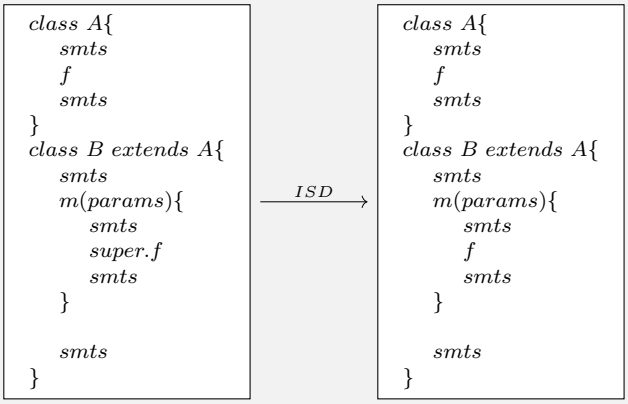
E-RULE:AOIS-01 (MuJava)



constraints
$(\neg)$ (v is declared in m) $\wedge$ (v is not used in $smts$) $\wedge$ (v is not inside a loop)

1.2.2 ISD-01 (MuJava)

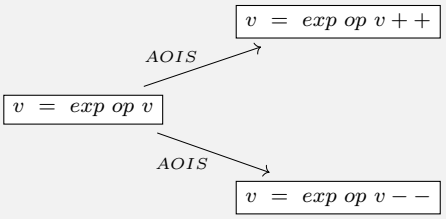
ISD-01 (MuJava)



constraints
(¬) (f is declared in A) ∧ (there is no declaration of f in B)

1.2.3 AOIS-02 (MuJava)

AOIS-02 (MuJava)



constraints
(¬) (v appears in the LHS and RHS of an assignment statement) ∧ (v on the RHS is the last use of v in the statement)

1.2.4 ROR-01 (Major)

ROR-01 (Major)

$if(sv.length\ op_1\ 0)\{\text{smts}\} \xrightarrow{ROR} if(sv.length\ op_2\ 0)\{\text{smts}\}$

constraints

$(\neg)\ (\ op_1 \in \{<\} \wedge op_2 \in \{!=\}\ )\ \vee$
$(\ op_1 \in \{!=\} \wedge op_2 \in \{<\}\ )\ \vee$
$(\ op_1 \in \{>\} \wedge op_2 \in \{!=\}\ )\ \vee$
$(\ op_1 \in \{!=\} \wedge op_2 \in \{>\}\ )\ \vee$
$(\ op_1 \in \{==\} \wedge op_2 \in \{<=\}\ )$

Obs 1. This E-RULE also works for Array type variables.
Obs 2. This E-RULE also applies to MuJava (see 1.2.5).

1.2.5 ROR-01 (MuJava)

ROR-01 (MuJava)

$if(sv.length\ op_1\ 0)\{\text{smts}\} \xrightarrow{ROR} if(sv.length\ op_2\ 0)\{\text{smts}\}$

constraints

$(\neg)\ (\ op_1 \in \{<\} \wedge op_2 \in \{!=\}\ )\ \vee$
$(\ op_1 \in \{!=\} \wedge op_2 \in \{<\}\ )\ \vee$
$(\ op_1 \in \{>\} \wedge op_2 \in \{!=\}\ )\ \vee$
$(\ op_1 \in \{!=\} \wedge op_2 \in \{>\}\ )\ \vee$
$(\ op_1 \in \{==\} \wedge op_2 \in \{<=\}\ )$

Obs 1. This E-RULE also works for Array type variables.
Obs 2. This E-RULE also applies to MuJava (see 1.2.4).

1.2.6 AOR-01 (Major)

AOR-01 (Major)

$new\ StringBuilder(exp_1\ aop_1\ exp_2) \xrightarrow{AOR} new\ StringBuilder(exp_1\ aop_2\ exp_2)$

constraints

$(\neg)$ (op_2 does not turn the resulted value to negative)

Obs 1. This E-RULE also applies to MuJava (see 1.2.27).

1.2.7 JID-01 (MuJava)

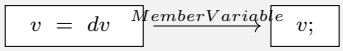
JID-01 (MuJava)

constraints
$(\neg)\ \varnothing$

Obs 1. This E-RULE also applies to Pit (see 1.2.8)

1.2.8 MemberVariable-01 (Pit)

MemberVariable-01 (Pit)

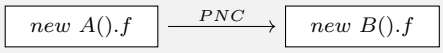


constraints
$(\neg)\ \varnothing$

Obs 1. This E-RULE also applies to MuJava (see 1.2.7)

1.2.9 PNC-01 (MuJava)

PNC-01 (MuJava)



constraints
$(\neg)$ (A extends B) $\wedge$
(f exists only in A) $\wedge$
(B constructor does not change v) $\wedge$
(B constructor calls the same C constructor)

1.2.10 JSI-01 (MuJava)

JSI-01 (MuJava)

$$\boxed{\textit{private type } f \ = \ \textit{value};} \xrightarrow{JSI} \boxed{\textit{private static type } f \ = \ \textit{value};}$$

constraints
$(\neg)$ (f is read-only)

1.2.11 ROR-02 (Major)

ROR-02 (Major)

$$\boxed{\begin{array}{l}\textit{if } (v_1\ op_1\ v_2)\{\\ \quad \textit{return } v_1;\\ \}\ \textit{else}\ \{\\ \quad \textit{return } v_2;\\ \}\end{array}} \xrightarrow{ROR} \boxed{\begin{array}{l}\textit{if } (v_1\ op_2\ v_2)\{\\ \quad \textit{return } v_1;\\ \}\ \textit{else}\ \{\\ \quad \textit{return } v_2;\\ \}\end{array}}$$

constraints
$(\neg)\ (op_1 \in \{>\} \wedge op_2 \in \{>=\}) \vee (op_1 \in \{<\} \wedge op_2 \in \{<=\})$

Obs 1. This E-RULE also applies to Pit (see 1.2.12)

1.2.12 ConditionalBoundary-01 (Pit)

ConditionalBoundary-01 (Pit)

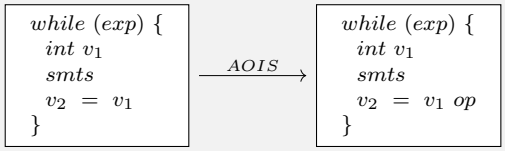
$$\boxed{(exp_1\ op_1\ exp_2)\ ?\ exp_1\ :\ exp_2} \xrightarrow{ConditionalBoundary} \boxed{(exp_1\ op_2\ exp_2)\ ?\ exp_1\ :\ exp_2}$$

constraints
$(\neg)\ (op_1 \in \{>\} \wedge op_2 \in \{>=\}) \vee (op_1 \in \{<\} \wedge op_2 \in \{<=\})$

Obs 1. This E-RULE also applies to Major (see 1.2.11)

1.2.13 AOIS-03 (MuJava)

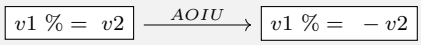
AOIS-03 (MuJava)



constraints
(v_1 is a local variable) $\wedge$
(v_1 on the RHS is the last use of v_1 in the loop) $\wedge$
($op \in \{++, --\}$)

1.2.14 AOIU-01 (MuJava)

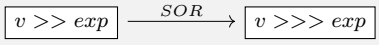
AOIU-01 (MuJava)



constraints
($\neg$) $\varnothing$

1.2.15 SOR-01 (Major)

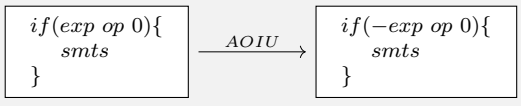
SOR-01 (Major)



constraints
($\neg$) ($v > 0$)

1.2.16 AOIU-02 (MuJava)

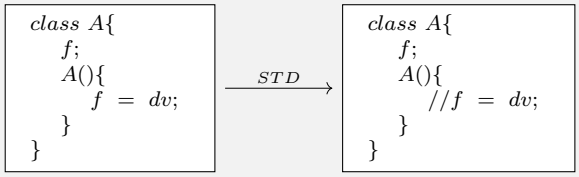
AOIU-02 (MuJava)



constraints
($\neg$) ($op \in \{==, !=\}$)

1.2.17 STD-01 (Major)

STD-01 (Major)

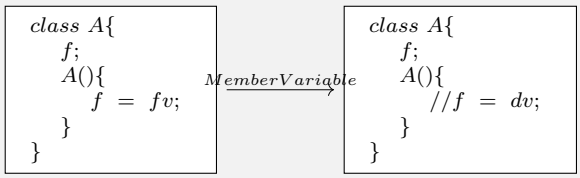


constraints
(¬) (f does not change before execute the constructor)

Obs 1. This E-RULE also applies to Pit (see 1.2.18).

1.2.18 MemberVariable-02 (Pit)

MemberVariable-02 (Pit)

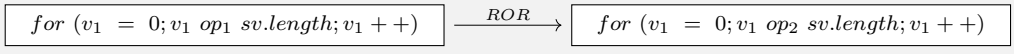


constraints
(¬) (f does not change before execute the constructor)

Obs 1. This E-RULE also applies to Major (see 1.2.17).

1.2.19 ROR-03 (Major)

ROR-03 (Major)



constraints
(there is no definition of v_1 inside loop body) ∧
$((op_1 \in \{<\} \wedge op_2 \in \{!=\}) \vee (op_1 \in \{!=\} \wedge op_2 \in \{<\}))$

1.2.20 ROR-02 (MuJava)

ROR-02 (MuJava)

$$\boxed{\begin{array}{l} if(sv.length\ op_1\ 0)\{ \\ \quad smts \\ \} \end{array}} \xrightarrow{ROR} \boxed{\begin{array}{l} if(sv.length\ op_2\ 0)\{ \\ \quad smts \\ \} \end{array}}$$

constraints

$(\neg)\ (op_1 \in \{>\} \wedge op_2 \in \{!=\}) \vee (op_1 \in \{==\} \wedge op_2 \in \{<=\})$

1.2.21 LVR-01 (Major)

LVR-01 (Major)

$$\boxed{\begin{array}{l} v = value; \\ smts \\ v = exp; \end{array}} \xrightarrow{LVR} \boxed{\begin{array}{l} v = value + 1; \\ smts \\ v = exp; \end{array}}$$

constraints

$(\neg)$(v_1 value isn't read between its first definition and last attribution.)

Obs 1. This E-RULE also applies to Pit (see 1.2.22).

1.2.22 InlineConstant-01 (Pit)

InlineConstant-01 (Pit)

$$\boxed{\begin{array}{l} v = value; \\ smts \\ v = exp; \end{array}} \xrightarrow{InlineConstant} \boxed{\begin{array}{l} v = value + 1; \\ smts \\ v = exp; \end{array}}$$

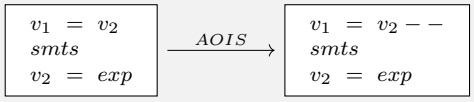
constraints

$(\neg)$(v_1 value isn't read between its first definition and last attribution.)

Obs 1. This E-RULE also applies to Major (see 1.2.21).

1.2.23 AOIS-04 (MuJava)

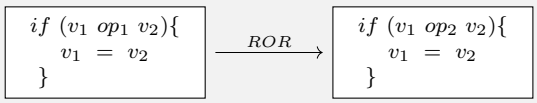
AOIS-04 (MuJava)



constraints
$(\neg)(v_2$ isn't read between the first and last attribution$)\wedge$
$(op \in \{++, --\})$

1.2.24 ROR-03 (MuJava)

ROR-03 (MuJava)

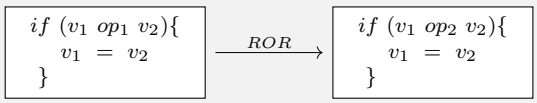


constraints
$(\neg)(op_1 \in \{<\} \wedge op_2 \in \{<=\}) \vee$
$(op_1 \in \{>\} \wedge op_2 \in \{>=\})$

Obs 1. This E-RULE also applies to Major and Pit (see 1.2.25 and **??**).

1.2.25 ROR-04 (Major)

ROR-04 (Major)



constraints
$(\neg)(op_1 \in \{<\} \wedge op_2 \in \{<=\}) \vee$
$(op_1 \in \{>\} \wedge op_2 \in \{>=\})$

Obs 1. This E-RULE also applies to MuJava and Pit (see 1.2.24 and 1.2.25).

1.2.26 ConditionalBoundary-02 (Pit)

ConditionalBoundary-02 (Pit)

$$\boxed{\begin{array}{l} if\ (v_1\ op_1\ v_2)\{ \\ \quad v_1\ =\ v_2 \\ \} \end{array}} \xrightarrow{ConditionalBoundary} \boxed{\begin{array}{l} if\ (v_1\ op_2\ v_2)\{ \\ \quad v_1\ =\ v_2 \\ \} \end{array}}$$

constraints
$(\neg)(op_1\ \in \{<\}\ \wedge\ op_2\ \in\ \{<=\}) \vee$
$(op_1\ \in \{>\}\ \wedge\ op_2\ \in\ \{>=\})$

1.2.27 AORB-01 (MuJava)

AORB-01 (MuJava)

$$\boxed{new\ StringBuilder(exp_1\ aop_1\ exp_2)} \xrightarrow{AORB} \boxed{new\ StringBuilder(exp_1\ aop_2\ exp_2)}$$

constraints
$(\neg)$ (op_2 does not turn the resulted value to negative)

Obs 1. This E-RULE also applies to Major (see 1.2.6).

1.2.28 JSD-01 (MuJava)

JSD-01 (MuJava)

$$\boxed{private\ static\ final\ type\ =\ value;} \xrightarrow{JSD} \boxed{private\ final\ type\ =\ value;}$$

constraints
$(\neg)\ \varnothing$

1.2.29 ROR-05 (MuJava)

ROR-05 (MuJava)

$$\boxed{\begin{array}{l} if(v\ op_1\ value)\{ \\ \quad smts \\ \} \end{array}} \xrightarrow{ROR} \boxed{\begin{array}{l} if(v\ op_2\ value)\{ \\ \quad smts \\ \} \end{array}}$$

constraints

$(\neg)\ (value == Integer.MAX\_VALUE\ \wedge\ op_1\ \in\ \{<\}\ \wedge\ op_2\ \in\ \{<=\}\ ) \vee$
$(value == Integer.MIN\_VALUE\ \wedge\ op_1\ \in\ \{>\}\ \wedge\ op_2\ \in\ \{>=\})$

1.2.30 LVR-02 (Major)

LVR-02 (Major)

$$\boxed{v\ =\ new\ CollectionType(value_1);} \xrightarrow{LVR} \boxed{v\ =\ new\ CollectionType(value_1 + 1);}$$

constraints

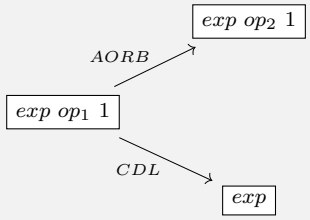
$(\neg)\ \varnothing$

1.3 D-RULEs

...

1.3.1 AORB|CDL-01 (MuJava)

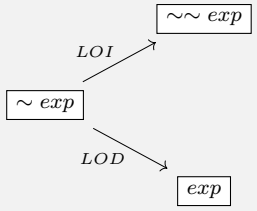
D-RULE:AORB|CDL-01 (MuJava)



constraints
$(\neg)\ (aop_1 \in \{+, -\}) \wedge (aop_2 \in \{*, /\})$

1.3.2 LOI|LOD-01 (MuJava)

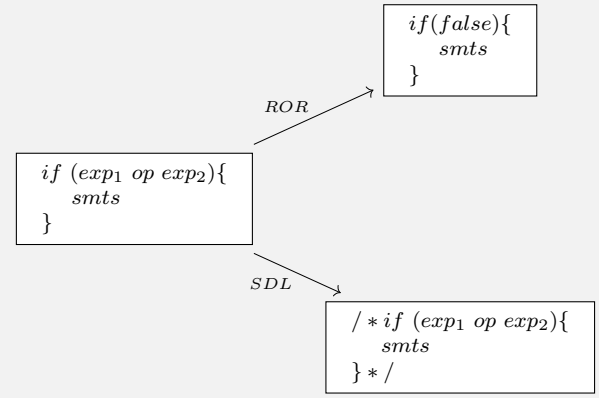
LOI|LOD-01 (MuJava)



constraints
$(\neg)\ \varnothing$

1.3.3 ROR|SDL-01 (MuJava)

ROR|SDL-01 (MuJava)

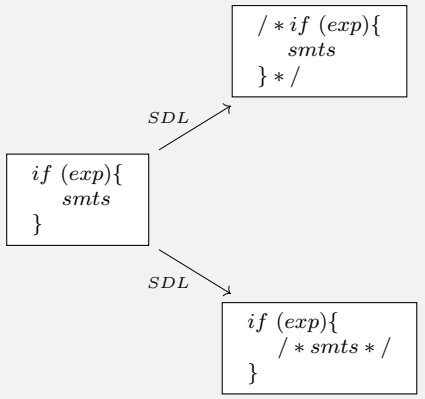


constraints
(¬) ∅

Obs 1. This D-RULE also applies to Major (see **??**).

1.3.4 SDL|SDL-01 (MuJava)

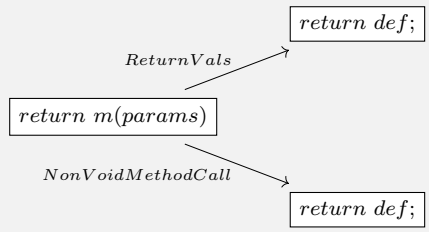
SDL|SDL-01 (MuJava)



constraints
(¬) (*exp* does not contain side effect statements)

1.3.5 ReturnVals|NonVoidMethodCall-01 (Pit)

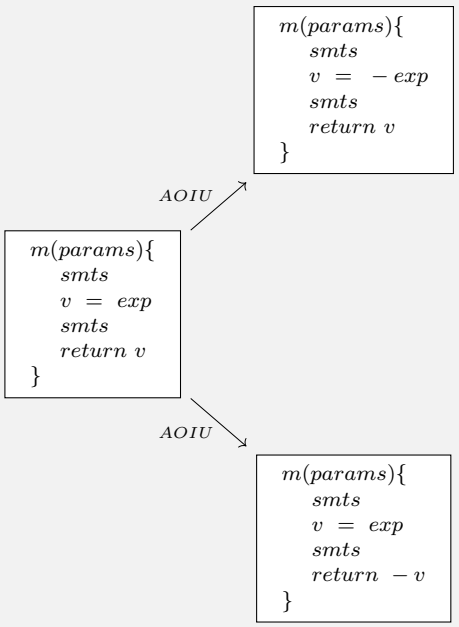
ReturnVals|NonVoidMethodCall-01 (Pit)



constraints
(¬) (the type of m is `boolean`, `int`, `byte`, `short`, or `object`)

1.3.6 AOIU|AOIU-01 (MuJava)

AOIU|AOIU-01 (MuJava)

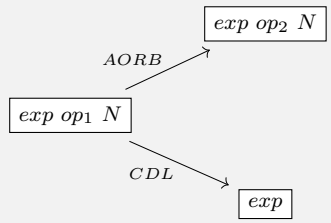


constraints
(¬) (there is neither a definition nor a use of v in *smts*)

Obs 1. This D-RULE works similar to LOI|LOI-01 (MuJava).

1.3.7 AORB|CDL-01 (MuJava)

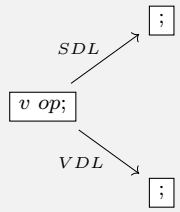
D-RULE:AORB|CDL-01 (MuJava)



constraints
$(\neg)\ (aop_1 \in \{+, -\}) \wedge (aop_2 \in \{*, /\})$

1.3.8 SDL|VDL-01 (MuJava)

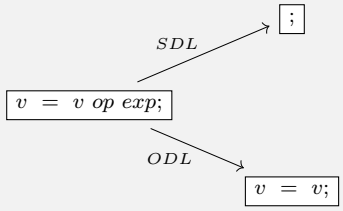
D-RULE:SDL|VDL-01 (MuJava)



constraints
$(\neg)\ (op\ \in\ \{++,\ --\})$

1.3.9 SDL|ODL-01 (MuJava)

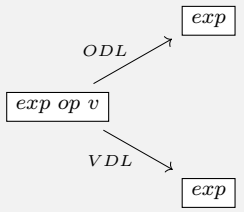
D-RULE:SDL|ODL-01 (MuJava)



constraints
(¬)(*op* is any binary (infix) operator)

1.3.10 ODL|VDL-01 (MuJava)

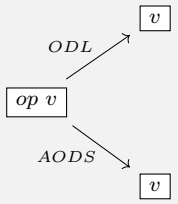
D-RULE:ODL|VDL-01 (MuJava)



constraints
(¬) (∅)

1.3.11 ODL|AODS-01 (MuJava)

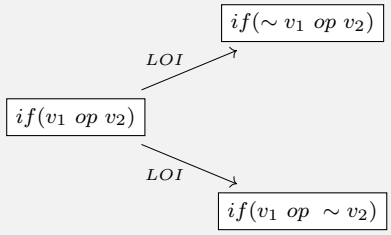
D-RULE:ODL|AODS-01 (MuJava)



constraints
(¬) ($op \in \{++, --\}$)

1.3.12 LOI|LOI-01 (MuJava)

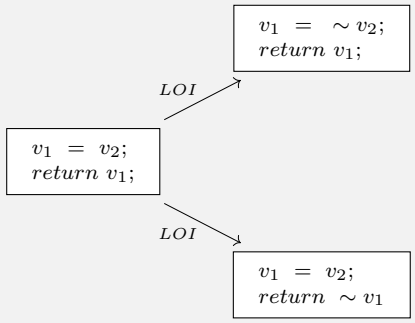
D-RULE:LOI|LOI-01 (MuJava)



constraints
$(\neg)(op\ \in\ \{==,\ !=\})$

1.3.13 LOI|LOI-02 (MuJava)

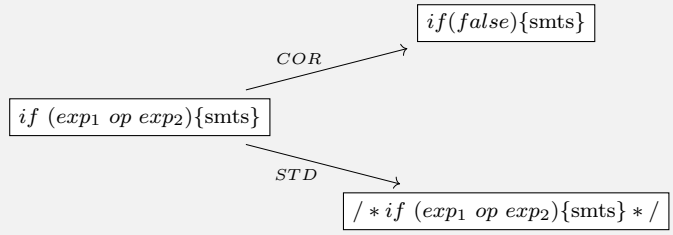
D-RULE:LOI|LOI-02 (MuJava)



constraints
$(\neg)\ (op\ \in\ \{==,\ !=\})$

1.3.14 COR|STD-01 (Major)

D-RULE:COR|STD-01 (Major)

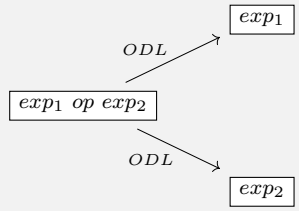


constraints
$(\neg)\ \varnothing$

Obs 1. This D-RULE also applies to MuJava (see **??**).

1.3.15 ODL|ODL-01 (MuJava)

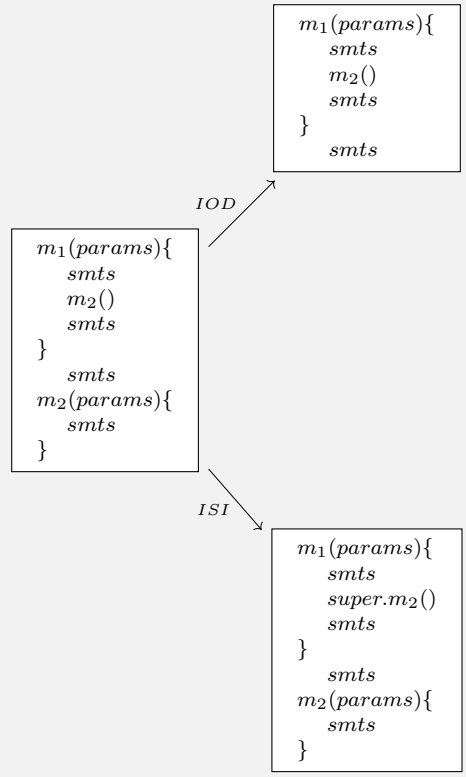
D-RULE:ODL|ODL-01 (MuJava)



constraints
$(\neg)(exp_1$ is equal to $exp_2)$

1.3.16 IOD|ISI-01 (MuJava)

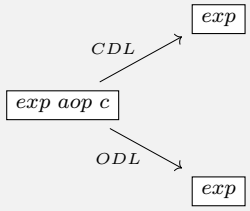
D-RULE:IOD|ISI-01 (MuJava)



constraints
$(\neg)(m_2$ must override superclass $m_2)$

1.3.17 CDL|ODL-01 (MuJava)

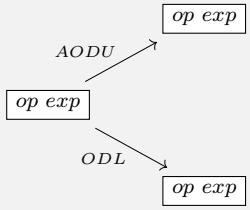
D-RULE:CDL|ODL-01 (MuJava)



constraints
$(\neg)$ $(\varnothing)$

1.3.18 ODL|AODU-01 (MuJava)

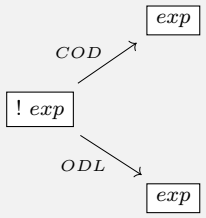
D-RULE:ODL|AODU-01 (MuJava)



constraints
$(\neg)$ $(op \in \{+, -\})$

1.3.19 COD|ODL-01 (MuJava)

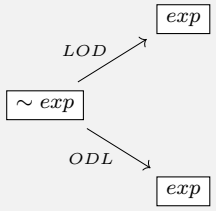
D-RULE:COD|ODL-01 (MuJava)



constraints
$(\neg)$ $(\varnothing)$

1.3.20 LOD|ODL-01 (MuJava)

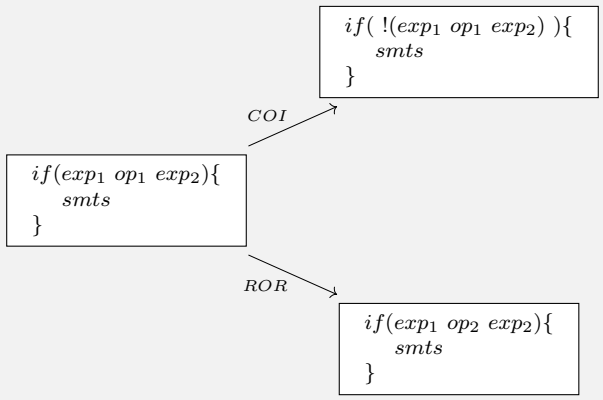
D-RULE:LOD|ODL-01 (MuJava)



constraints
(¬) (∅)

1.3.21 COI|ROR-01 (MuJava)

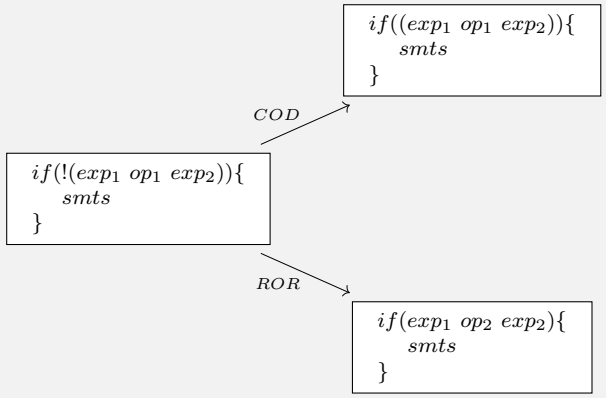
D-RULE:COI|ROR-01 (MuJava)



constraints
$(\neg)\ (op_1 \in \{>\} \wedge op_2 \in \{<=\}) \vee$
$(op_1 \in \{<\} \wedge op_2 \in \{>=\}) \vee$
$(op_1 \in \{>=\} \wedge op_2 \in \{<\}) \vee$
$(op_1 \in \{<=\} \wedge op_2 \in \{>\}) \vee$
$(op_1 \in \{==\} \wedge op_2 \in \{!=\}) \vee$
$(op_1 \in \{!=\} \wedge op_2 \in \{==\})$

1.3.22 COD|ROR-01 (MuJava)

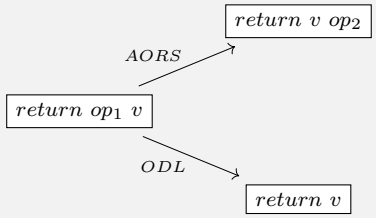
D-RULE:COD|ROR-01 (MuJava)



constraints
$(\neg)$ $(op_1 \in \{>\} \wedge op_2 \in \{<=\}) \vee$
$(op_1 \in \{<\} \wedge op_2 \in \{>=\}) \vee$
$(op_1 \in \{>=\} \wedge op_2 \in \{<\}) \vee$
$(op_1 \in \{<=\} \wedge op_2 \in \{>\}) \vee$
$(op_1 \in \{==\} \wedge op_2 \in \{!=\}) \vee$
$(op_1 \in \{!=\} \wedge op_2 \in \{==\})$

1.3.23 AORS|ODL-01 (MuJava)

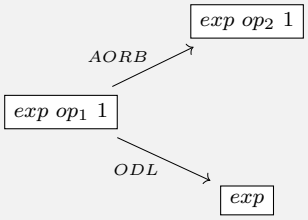
D-RULE:AORS|ODL-01 (MuJava)



constraints
$(\neg)$ (v has a local scope) $\wedge$
(the use of v is the last one in its scope) $\wedge$
(op_1 is a pre-increment or pre-decrement operator) $\wedge$
(op_2 is a post-increment or post-decrement operator)

1.3.24 AORB|ODL-01 (MuJava)

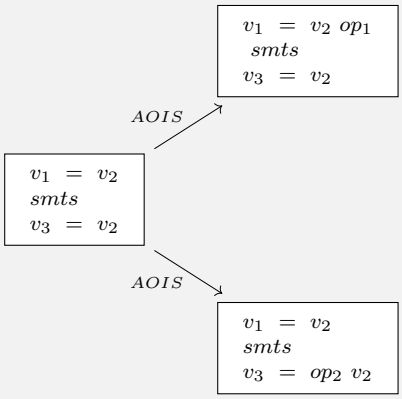
D-RULE:AORB|ODL-01 (MuJava)



constraints
$(\neg)\ (aop_1 \in \{+, -\}) \wedge (aop_2 \in \{*, /\})$

1.3.25 AOIS|AOIS-01 (MuJava)

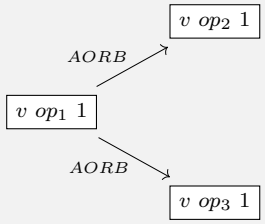
D-RULE:AOIS|AOIS-01 (MuJava)



constraints
$(\neg)$ (there is no definition of v_2 in b)
(op_1 is a post-increment $\wedge$ op_2 is pre-increment operator) $\vee$
(op_1 is a post-decrement $\wedge$ op_2 is pre-decrement operator)

1.3.26 AORB|AORB-01 (MuJava)

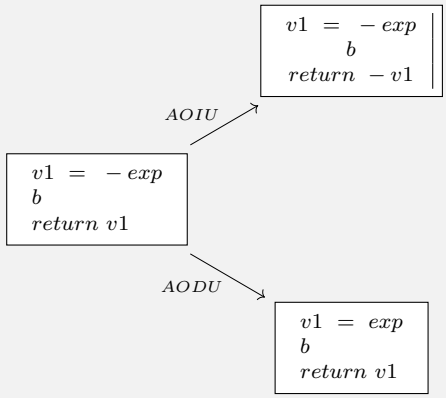
D-RULE:AORB|AORB-01 (MuJava)



constraints
$(\neg)\ (op_1 \in \{+, -\}) \wedge$
$(op_2 \in \{*, /\} \wedge op_3 \in \{*, /\})$

1.3.27 AOIU|AODU-01 (MuJava)

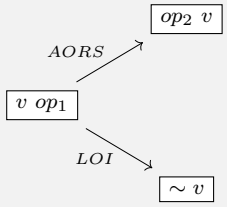
D-RULE:AOIU|AODU-01 (MuJava)



constraints
$(\neg)$ (there is no definition of v_1 in b)
(v_1 needs to have a local scope)

1.3.28 AORS|LOI-01 (MuJava)

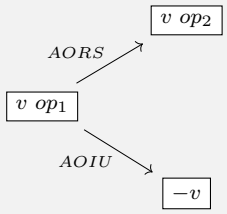
D-RULE:AORS|LOI-01 (MuJava)



constraints
$(\neg)\ (op_1\ \in\ \{++\})$
$(op_2\ \in\ \{--\})$
(v must must start with zero)
(the only definition of v in the program is in this line of the term)

1.3.29 AORS|AOIU-01 (MuJava)

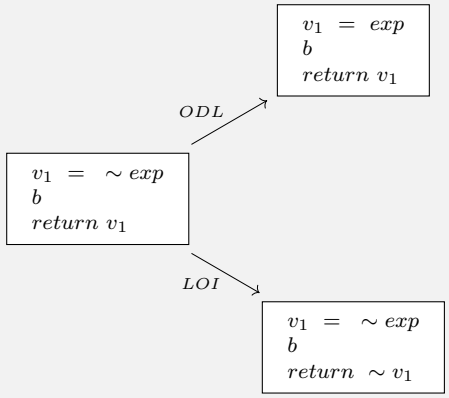
D-RULE:AORS|AOIU-01 (MuJava)



constraints
$(\neg)\ (op_1\ \in\ \{++\})$
$(op_2\ \in\ \{--\})$
(v must must start with zero)
(the only definition of v in the program is in this line of the term)

1.3.30 ODL|LOI-01 (MuJava)

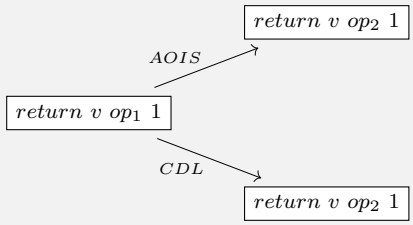
D-RULE:ODL|LOI-01 (MuJava)



constraints
(¬) (there is no definition of v_1 in b)

1.3.31 AOIS|CDL-01 (MuJava)

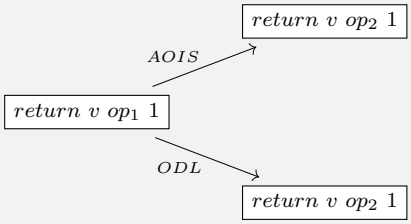
D-RULE:AOIS|CDL-01 (MuJava)



constraints
(¬) (op_1 is a plus operator and op_2 is pre-decrement operator or)
(op_1 is a minus operator and op_2 is pre-increment operator)
(v has a local scope)
(the use of v is the last one in its scope)

1.3.32 AOIS|ODL-01 (MuJava)

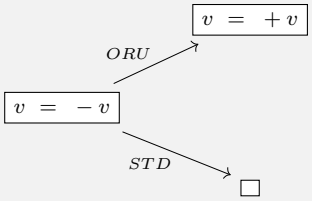
D-RULE:AOIS|ODL-01 (MuJava)



constraints ($op_1 \in \{+\} \wedge op_2 \in \{--\}) \vee (op_1 \in \{-\} \wedge op_2 \in \{++\}) \wedge$ (v has a local scope) $\wedge$ (the use of v is the last one in its scope)
$(\neg)$

1.3.33 ORU|STD-01 (Major)

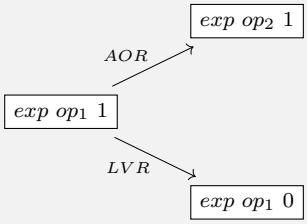
D-RULE:ORU|STD-01 (Major)



constraints
$(\neg)(\varnothing)$

1.3.34 AOR|LVR-01 (Major)

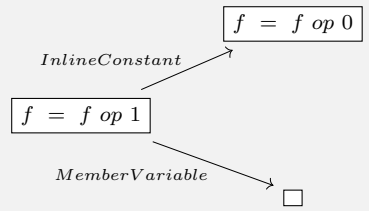
D-RULE:AOR|LVR-01 (Major)



constraints
$(\neg)\ (op_1 \in \{+, -\}) \wedge (op_2 \in \{*, /\})$

1.3.35 InlineConstant|MemberVariable-01 (Pit)

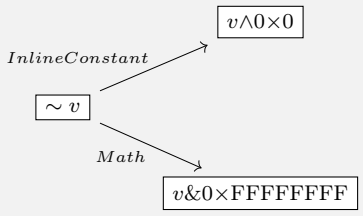
D-RULE:InlineConstant|MemberVariable-01 (Pit)



constraints
$(\neg)(op \in \{+, -\})$

1.3.36 InlineConstant|Math-01 (Pit)

D-RULE:InlineConstant|Math-01 (Pit)

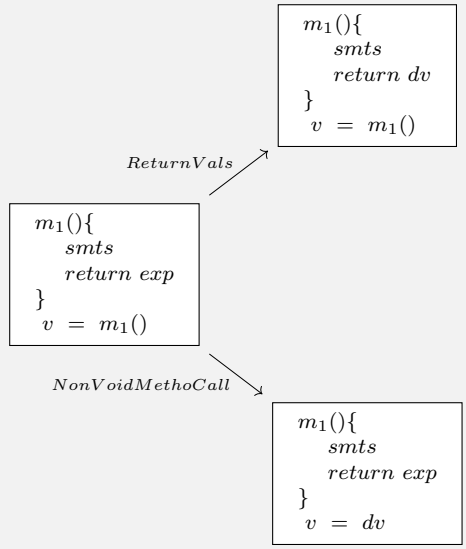


constraints
$(\neg)(\varnothing)$

Obs 1. In byte code: $\sim v$ transform to $v \wedge 0{\times}\text{FFFFFFFF}$

1.3.37 ReturnVals|NonVoidMethoCall-02 (Pit)

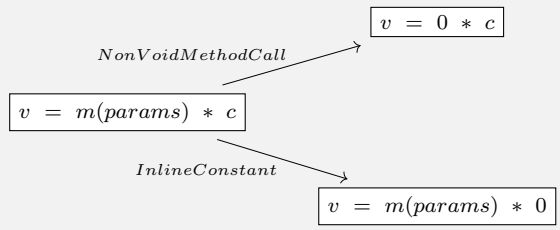
D-RULE:ReturnVals|NonVoidMethoCall-02 (Pit)



constraints
$(\neg)$ (m_1 has no side effects) $\wedge$ (m_1 has only one return) $\wedge$
(m_1 isn't called elsewhere)

1.3.38 NonVoidMethodCall|InlineConstant-01 (Pit)

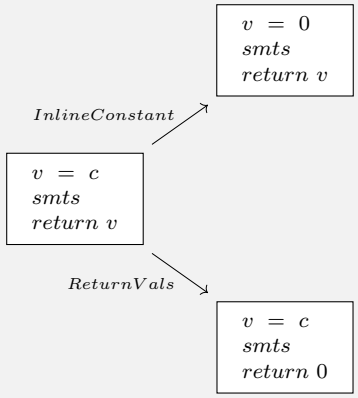
D-RULE:NonVoidMethodCall|InlineConstant-01 (Pit)



constraints
$(\neg)$

1.3.39 InlineConstant|ReturnVals-01 (Pit)

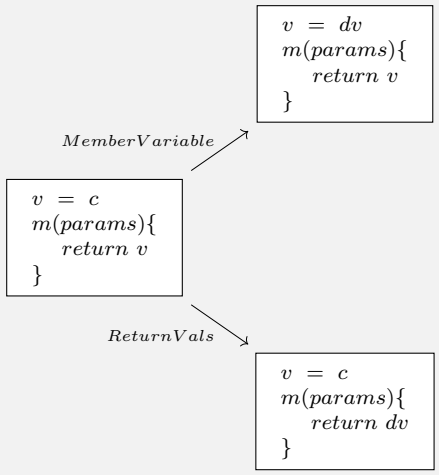
D-RULE:InlineConstant|ReturnVals-01 (Pit)



constraints
$(\neg())$ $(op \in \{*\})$

1.3.40 MemberVariable|ReturnVals-01 (Pit)

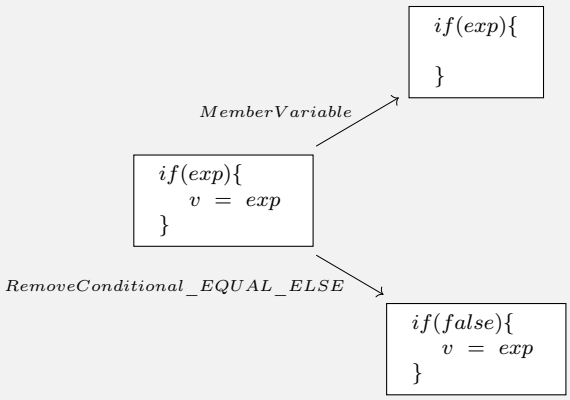
D-RULE:MemberVariable|ReturnVals-01 (Pit)



constraints
$(\neg)$ (v is used one time) $\wedge$ (v is not redefined)

1.3.41 MemberVariable|RemoveConditional_EQUAL_ELSE-01 (Pit)

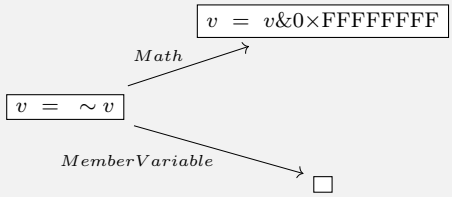
D-RULE:MemberVariable|RemoveConditional_EQUAL_ELSE-01 (Pit)



constraints
(¬) ($if - expression$ has no side effects) ∧
(there is no *else* statement) ∧
(v is a field)

1.3.42 Math|MemberVariable-01 (Pit)

D-RULE:Math|MemberVariable-01 (Pit)

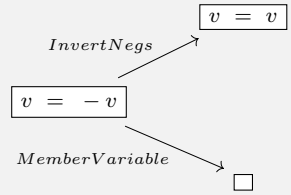


constraints
(¬) (∅)

Obs 1. In byte code: $\sim v$ transform to $v \wedge 0{\times}\text{FFFFFFFF}$

1.3.43 InvertNegs|MemberVariable-01 (Pit)

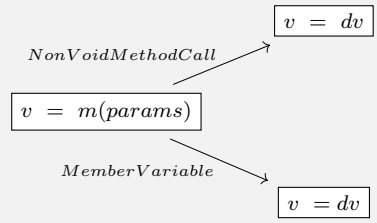
D-RULE:InvertNegs|MemberVariable-01 (Pit)



constraints
(¬) (∅)

1.3.44 NonVoidMethodCall|MemberVariable-01 (Pit)

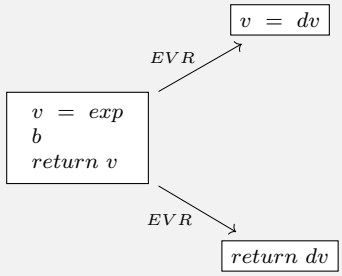
D-RULE:NonVoidMethodCall|MemberVariable-01 (Pit)



constraints
(¬) (v is a field)

1.3.45 EVR|EVR-01 (Major)

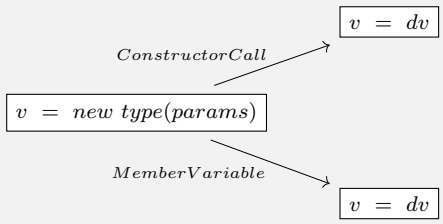
D-RULE:EVR|EVR-01 (Major)



constraints
(¬) (v is not used between it's definition and it's use in *return expression*)

1.3.46 ConstructorCall|MemberVariable-01 (Pit)

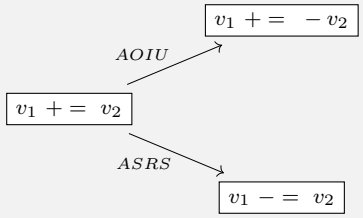
D-RULE:ConstructorCall|MemberVariable-01 (Pit)



constraints
(¬) (v is a field)

1.3.47 AOIU|ASRS-01 (MuJava)

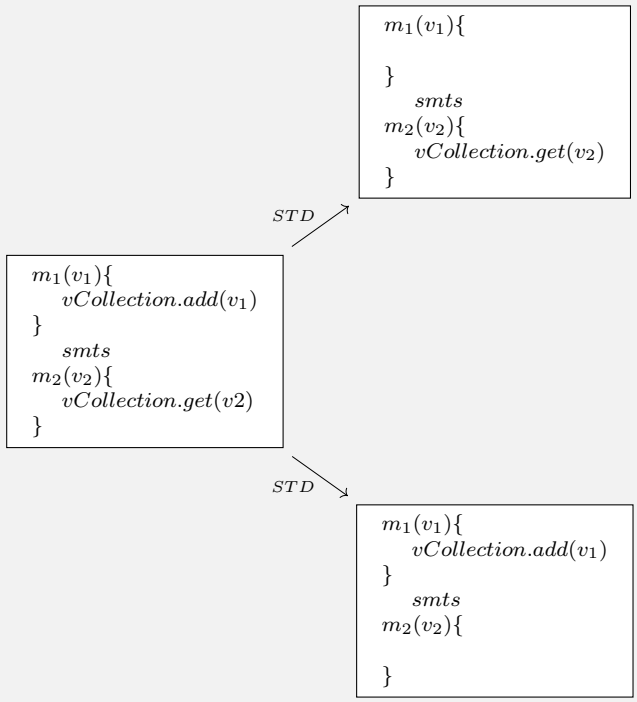
D-RULE:AOIU|ASRS-01 (MuJava)



constraints
(¬) (∅)

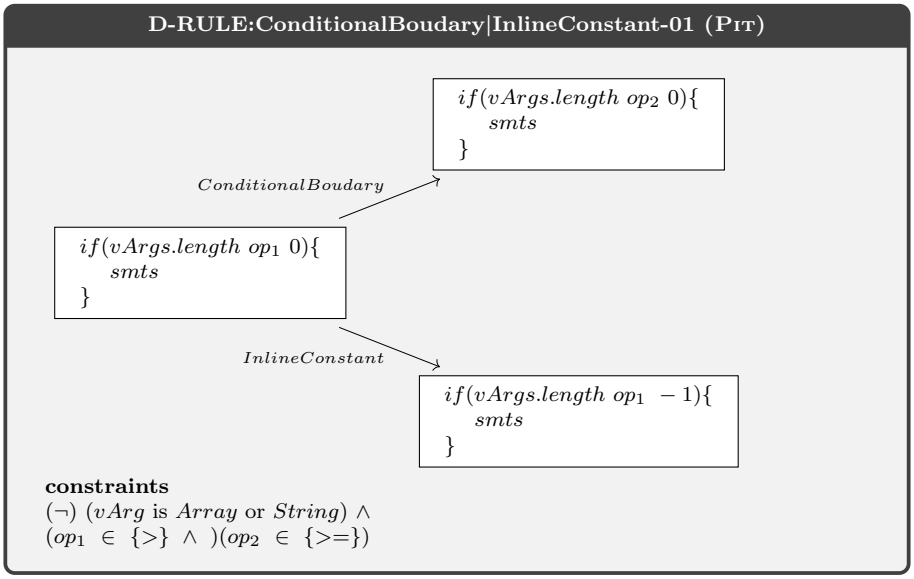
1.3.48 STD|STD-01 (Major)

D-RULE:STD|STD-01 (Major)



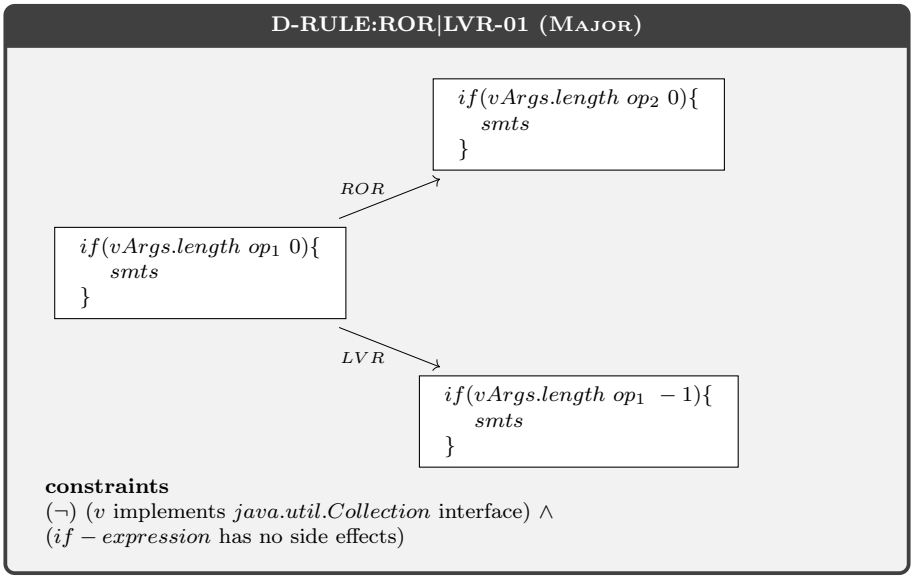
constraints
(¬) (*vCollection* implements *java.util.Collection* interface) ∧
(There is only one site that changes the content of *vCollection*)

1.3.49 ConditionalBoudary|InlineConstant-01 (Pit)



Obs 1. This D-RULE also applies to Major (see 1.3.50).

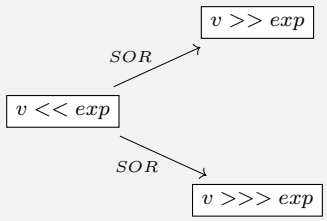
1.3.50 ROR|LVR-01 (Major)



Obs 1. This D-RULE also applies to PIT (see 1.3.49).

1.3.51 SOR|SOR-01 (MAJOR)

D-RULE:SOR|SOR-01 (MAJOR)

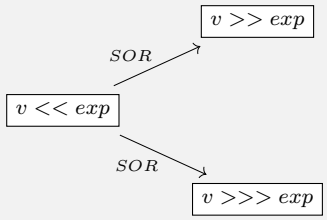


constraints
$(\neg)\ (v\ >\ 0)$

Obs 1. This D-RULE also applies to MUJAVA (see 1.3.52).

1.3.52 SOR|SOR-01 (MUJAVA)

D-RULE:SOR|SOR-01 (MUJAVA)

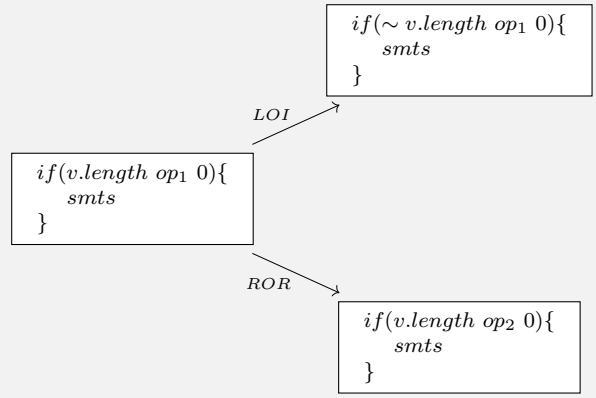


constraints
$(\neg)\ (v\ >\ 0)$

Obs 1. This D-RULE also applies to MAJOR (see 1.3.51).

1.3.53 LOI|ROR-01 (MuJava)

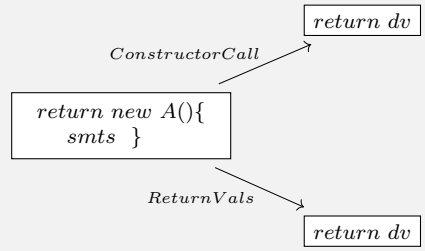
D-RULE:LOI|ROR-01 (MuJava)



constraints
$(\neg)$ (v is $String$ or $Array$) $\wedge$
$(op_1 \in \{<, <=, ==\} \wedge op_2 \in \{false\}) \vee$
$(op_1 \in \{>, >=, !=\} \wedge op_2 \in \{true\}) \vee$

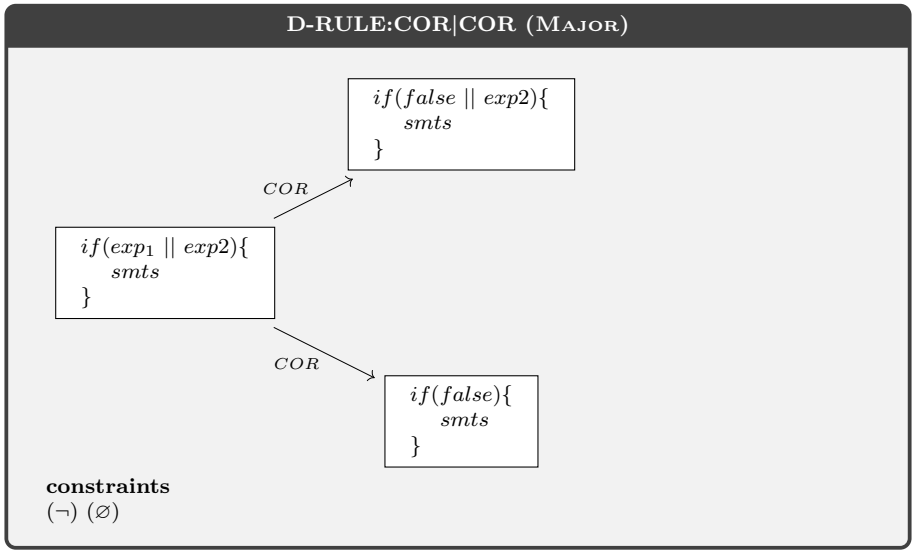
1.3.54 ConstructorCall|ReturnVals-01 (Pit)

D-RULE:ConstructorCall|ReturnVals (Pit)



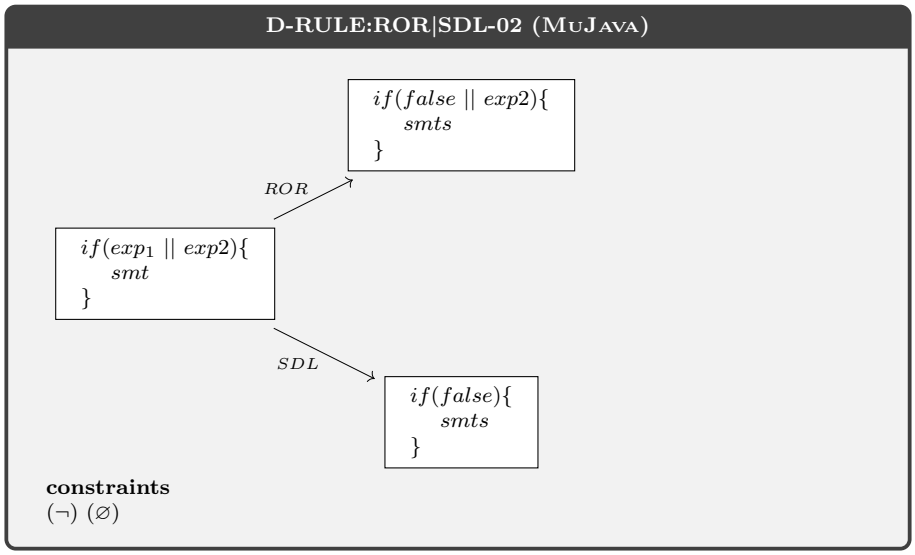
constraints
$(\neg)$ $(\varnothing)$

1.3.55 COR|COR-01 (Major)



Obs 1. This D-RULE also applies to MuJava (see 1.3.56).

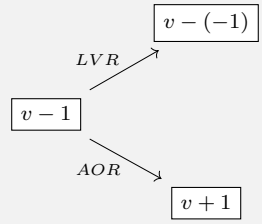
1.3.56 ROR|SDL-02 (MuJava)



Obs 1. This D-RULE also applies to MAJOR (see 1.3.55).

1.3.57 LVR|AOR-01 (MAJOR)

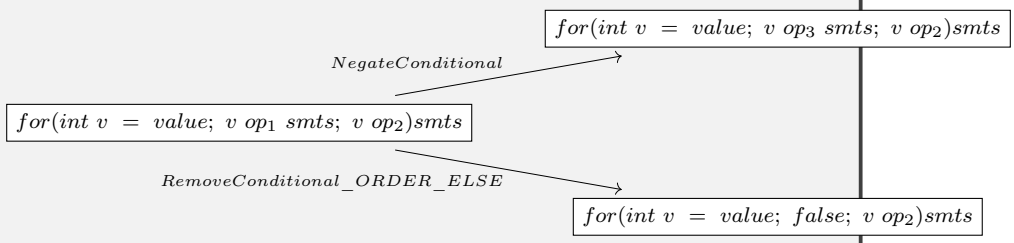
D-RULE:LVR|AOR-01 (MAJOR)



constraints
$(\neg)$ $(\varnothing)$

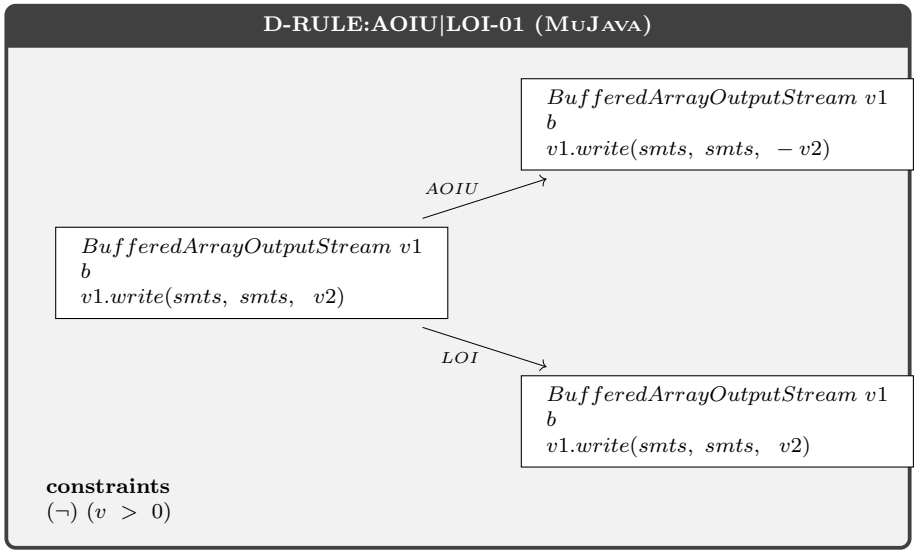
1.3.58 NegateConditional|RemoveConditional_ORDER_ELSE-01 (PIT)

D-RULE:NegateConditional|RemoveConditional_ORDER_ELSE-01 (PIT)

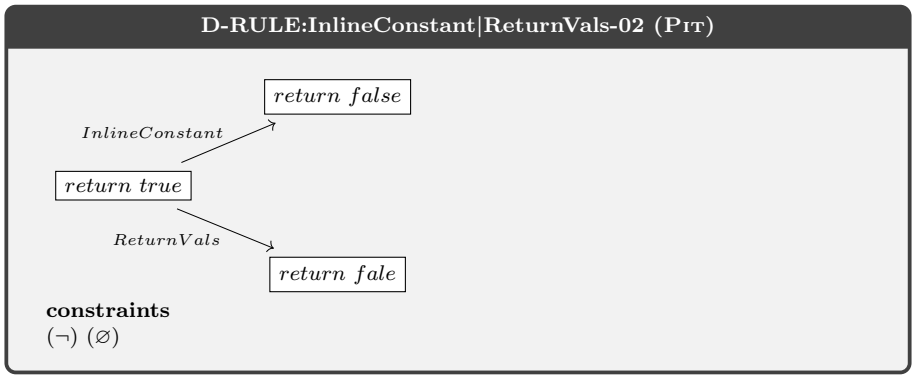


constraints
$(\neg)\ ((v_1\ ==\ 0\ \wedge\ v_2\ >\ 0)\ \wedge\ (op_1\ \in\ \{<\})\ \wedge\ (op_2\ \in\ \{++\})\ \wedge\ (op_3\ in\ \{>=\}))$
$\vee((v_1\ >\ 0\ \wedge\ v_2\ ==\ 0)\ \wedge\ (op_1\ \in\ \{>\})\ \wedge\ (op_2\ \in\ \{--\})\ \wedge\ (op_3\ in\ \{<=\}))$

1.3.59 AOIU|LOI-01 (MuJava)

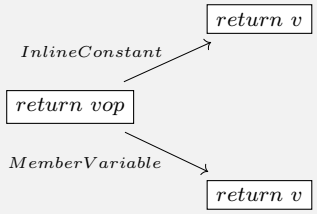


1.3.60 InlineConstant|ReturnVals-02 (Pit)



1.3.61 InlineConstant|MemberVariable-02 (Pit)

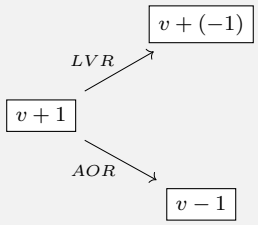
D-RULE:InlineConstant|MemberVariable-02 (Pit)



constraints
$(\neg)$ (v is a field) $\wedge$ op $\in \{++, --\}$

1.3.62 LVR|AOR-02 (Major)

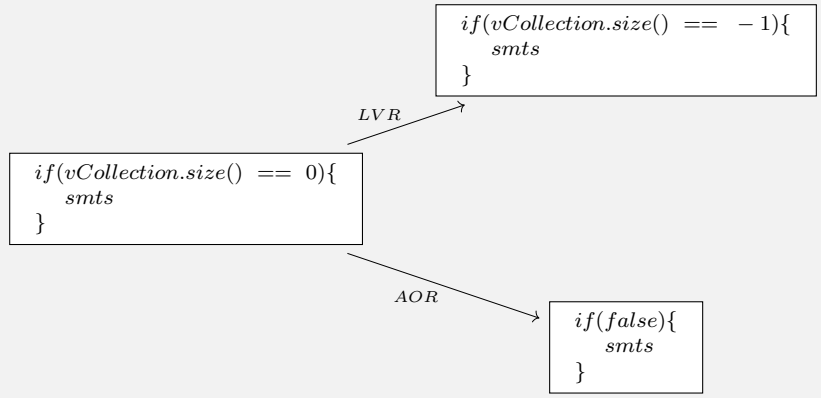
D-RULE:LVR|AOR-02 (Major)



constraints
$(\neg)$ $(\varnothing)$

1.3.63 ROR|LVR-02 (Major)

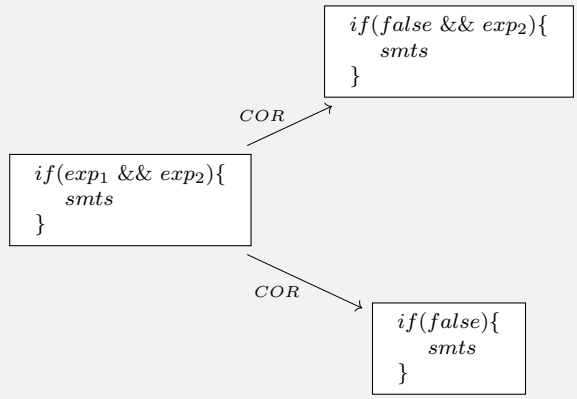
D-RULE:ROR|LVR-02 (Major)



constraints
(¬) (v implements $java.util.Collection$) ∧
($if - expression$ has no side effects)

1.3.64 COR|COR-02 (Major)

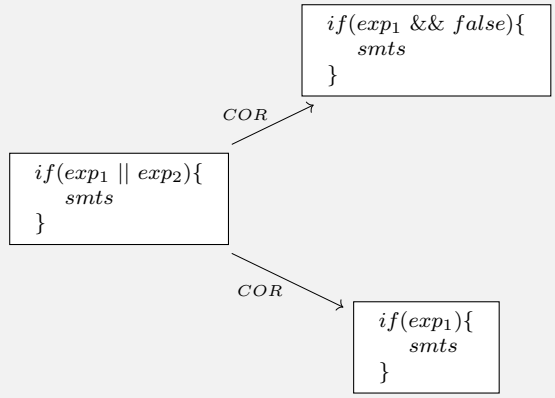
D-RULE:COR|COR-02 (Major)



constraints
(¬) (∅)

1.3.65 COR|COR-03 (Major)

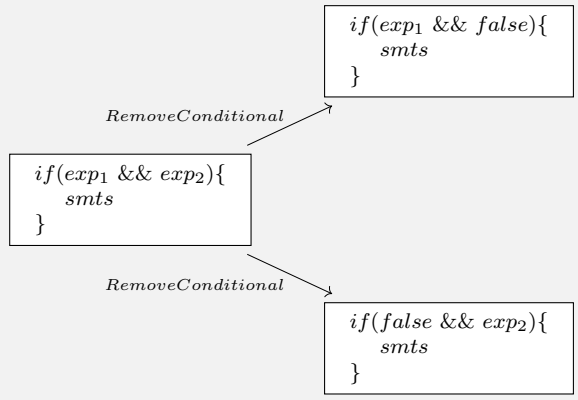
D-RULE:COR|COR-03 (Major)



constraints
(¬) (exp_1 has no side effects)

1.3.66 RemoveConditional|RemoveConditional-01 (Pit)

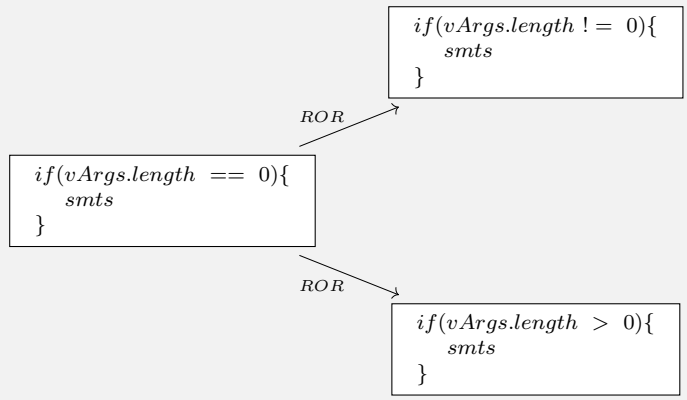
D-RULE:RemoveConditional|RemoveConditional-01 (Pit)



constraints
(¬) (exp_1 and exp_2 have no side effects)

1.3.67 ROR|ROR-01 (MuJava)

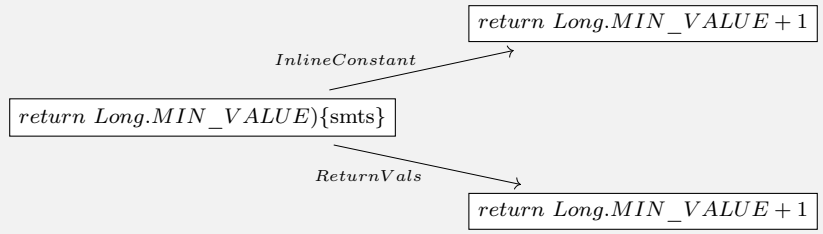
D-RULE:ROR|ROR-01 (MuJava



constraints
(¬) (v is *String* or *Array*)

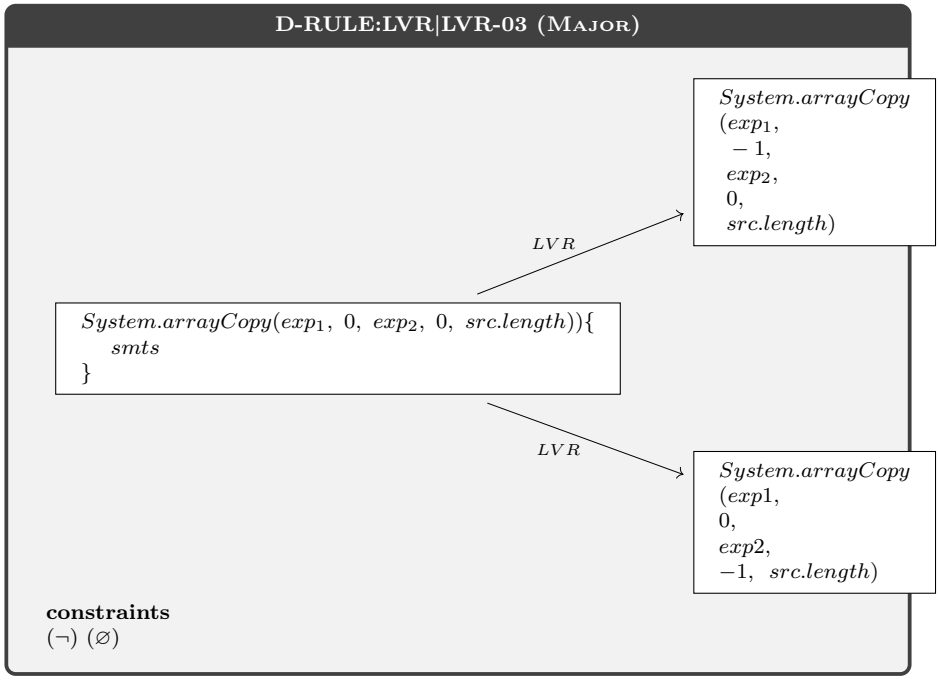
1.3.68 InlineConstant|ReturnVals-03 (Pit)

D-RULE:InlineConstant|ReturnVals-03 (Pit)



constraints
(¬) (∅)

1.3.69 LVR|LVR-03 (Major)



Obs. In both cases they rise `IndexOutOfBoundsException`.

References

[1] L. Fernandes, M. Ribeiro, P. Pinheiro, F. Ferrari, R. Gheyi, and A. Santos. Improving transformation rules to avoid useless mutants. 2020. To Appear.