Syntax Syntax
SYNTAX #Id ::= object null SYNTAX Variable ::= #Int SYNTAX VariableName ::= v #Int
SYNTAX VariableName ::= V #Int SYNTAX MethodName ::= <init></init>
SYNTAX TypeName ::= #Id #Id / TypeName SYNTAX TypeReference ::= < #Id , TypeName > SYNTAX FieldReference ::= < #Id , TypeName , #Id , TypeReference > SYNTAX MethodReference ::= < #Id , TypeName , Selector >
none SYNTAX NewInstructionBase ::= Variable = new TypeReference @ #Int SYNTAX NewInstruction ::= NewInstructionBase NewInstructionBase (Params) SYNTAX GetInstruction ::= Variable = getfield FieldReference Variable
Variable = getstatic FieldReference SYNTAX PutInstruction ::= putfield Variable = Variable FieldReference putstatic Variable FieldReference SYNTAX PhiInstruction ::= Variable = phi(Params) SYNTAX PhiPhiInstruction ::= Variable = phiphi(Params) SYNTAX Invoke Special Instruction ::= i pyckespecial Method Reference Params 0 #Intervation :: Variable
SYNTAX InvokeSpecialInstruction ::= invokespecial MethodReference Params @ #Int exception: Variable SYNTAX Instruction ::= NewInstruction GetInstruction PutInstruction PhiInstruction PhiInstruction
PhiPhithstruction return InvokeSpecialInstruction noinstruction noinstruction SYNTAX BBEdge ::= #Id -> #Id ; SYNTAX BlockBody ::= List{Instruction,";" }
SYNTAX Block::= #Id: { BlockBody } SYNTAX TaskUnit::= BBEdge Block SYNTAX Task::= TaskUnit Task Task
SYNTAX Program ::= MethodDefinition start analysis done Program +++ Program
SYNTAX MethodDefinition ::= MethodReference { Task } END MODULE MODULE KWALA IMPORTS KWALA-SYNTAX
Semantics
Configuration
CONFIGURATION:
method * basicBlocks basicBlock basic
invocations invocation * body
cenvs* variables null noinstruction 0
objects constraints task
pointsTo copy fieldRead fieldWrite 0 ik mname \$\fields \\ \fields
Processing Basic Blocks
RULE T_1 $T_2 \Rightarrow T_1 \curvearrowright T_2$ $ \text{RULE } I_1 \; ; \; BBl_2 \Rightarrow I_1 \curvearrowright BBl_2 $
RULE $Pg_1 + ++ Pg_2 \Rightarrow Pg_1 \curvearrowright Pg_2$ SYNTAX $ListItem ::= [\#Id, \#Id]$
RULE $BB_1 \rightarrow BB_2$; MR_1 MR_1 BB_2]
RIUE BB: { BI } MB.
RULE $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}MR_1\end{array}\right\}$ $\left\{\begin{array}{c}MR_1\end{array}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{Bl\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{BB:\left\{Bl\right\}\right\}\right\}$ $\left\{\begin{array}{c}BB:\left\{BB:\left\{BB:\left\{BB:\left\{BB:\left\{BB:\left\{BB:\left\{BB:\left$
RULE $BB:\{\}$ MR_I MR_I BB BB BB BB
RULE $MR_1 \{ T_2 \}$ MR_1 MR_1 MR_1 MR_1 MR_1
RULE start
method method
RULE ik methodName "<" Ap ₁ "," TN ₁ "," "main" "(" FP ₁ ")" "V" ">" BBls invocation variables BBls
Gathering Constraints
SYNTAX $K ::= wrappedList(Bag)$ Phi functions
RULE $ \begin{array}{c c} & & & & \\ \hline & k & & & \\ \hline & V_1 = phi(P) & & & \\ \hline & V_1 = phiphi(P) & & & \\ \hline & V_1 = phiphi(P) & & & \\ \hline \end{array} $
$V_1 = \text{phiphi}(P)$ v $V_1 \mapsto NP$ $NP +_{Int} 1$
$ \text{RULE} \left\{ \begin{array}{c} V_1 = \text{phiphi}(\ V_2\ ,\ P\) \\ \hline V_1 = \text{phiphi}(\ P\) \end{array} \right\} \left\{ \begin{array}{c} \text{v}\ V_1 \mapsto P_1 \ -\ \text{v}\ V_2 \mapsto P_2 \\ \hline \end{array} \right\} \left\{ \begin{array}{c} P_2 \mapsto \text{wrappedList}(\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $
RULE $V_1 = \text{phiphi}(V_2, P)$ $V_1 \mapsto P_1 - v V_2 \mapsto P_2$ when $\neg_{Bool} P_2$ in keys $Rest$
$ \begin{array}{ c c c c c c c c c c c c c c c c c c c$
RULE $V_1 = phi()$
Get field
SYNTAX FieldToPointer ::= (FieldReference > #Int) invocation
RULE $ \begin{array}{c c} V_1 = \text{getfield } F & V_2 \\ \hline \bullet & & & V_2 \mapsto P_2 \\ \hline \bullet & & & & \\ \hline \bullet & & & $
invocation k variables NB
$ \frac{\left\{ \begin{array}{c c} V_{I} = getfield \; F \; \; V_{2} \\ \hline \bullet \end{array} \right\} \left\{ \begin{array}{c c} \bullet & v \; V_{2} \mapsto P_{2} \\ \hline v \; V_{I} \mapsto NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} P_{2} \mapsto wrappedList(\left(\begin{array}{c} \bullet \\ \hline F \; \triangleright \; NP \; \right) \end{array} \right\} \left\{ \begin{array}{c c} \bullet & RestSet \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; 1 \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \hline NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ \end{smallmatrix} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c} \bullet & NP \\ NP +_{Int} \; NP \end{array} \right\} \left\{ \begin{array}{c c}$
Put field pointsTo
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
$ egtharpoonup_{Bool} P_1$ in keys $Rest$
RULE
New instruction
RULE $V_1 = \text{new } TR @ - \\ \hline V_1 = \text{new } TR @ - \\ \hline V_1 \rightarrow NP \\ \hline V_1 \rightarrow NP \\ \hline V_2 \rightarrow NP \\ \hline V_3 \rightarrow NP \\ \hline V_4 \rightarrow NP \\ \hline V_4 \rightarrow NP \\ \hline V_5 \rightarrow NP \\ \hline V_7 \rightarrow NP \\ \hline V_8 \rightarrow NP \\ \hline V_9 \rightarrow NP \\ \hline $
$NP \mapsto wrappedList(NO)$
Resolving Constraints
First type of graph: if copy encountered, propagate points-to
$RULE & B \mapsto wrappedList(A &) \\ \hline B \mapsto wrappedList(A &) & B \mapsto wrappedList(O &) - A \mapsto wrappedList(O & O \\ \hline O & O & O \\ \hline O &$
Copy points To cenvs cenvs
$B \mapsto wrappedList(\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$
Second type of graph: if field-read and points-to encountered, propagate copy
$ \begin{array}{c} \text{Copy} \\ \hline \\ P_1 \mapsto \text{wrappedList}(\begin{array}{c} \bullet \\ \hline A \end{array} \begin{array}{c} OldSet \end{array}) \end{array} \end{array} \right) \begin{array}{c} \text{pointsTo} \\ \hline \\ B \mapsto \text{wrappedList}(\begin{array}{c} \bullet \\ \hline O \end{array} \begin{array}{c} OldSet \end{array}) \end{array} \end{array} \\ \begin{array}{c} B \mapsto \text{wrappedList}(\begin{array}{c} \bullet \\ \hline O \end{array} \begin{array}{c} \bullet \\ \\ \hline O \end{array} \begin{array}{c} \bullet \\ \hline O \end{array} \begin{array}{c} \bullet \\ \hline O \end{array} \begin{array}{c} \bullet \\ \end{array} \begin{array}{c} \bullet \\ \\ \hline O \end{array}$
pointsTo
$ \begin{array}{c c} & & & \\ \hline & & \\ $
Third type of graph: if field-write and points-to encountered, propagate copy
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
$\overbrace{P_1}$
$A \mapsto wrappedList(\begin{tabular}{ c c c c c c c c c c c c c c c c c c c$
$B\mapsto wrappedList(egin{array}{c} P_1 \end{array})$ END MODULE