The Bytecode Verifier (BV)

What and how

- The BV is an implementation of the type system.
  The BV tries to compute a method type for each method.
- The BV is an iterative data flow analyzer.
  The method type is computed by a fixed point iteration.

Example

`Load` *0*	$\lfloor([], [OK\ (Class\ B), OK\ Integer])\rfloor$
`Store` *1*	
`Load` *0*	
`Getfield` *F A*	
`Goto` -3	

Assumptions: $P \vdash B \preceq^* A$

$P \vdash A\ sees\ F{:}Class\ A\ in\ A$

Example (2)

What if we remove `Store` *1*?

`Load` *0*	$\lfloor([], [OK\ (Class\ B), OK\ Integer])\rfloor$
`Load` *0*	
`Getfield` *F A*	
`Goto` -3	

Questions

- Why does fixed point iteration always terminate?
- Why does successful termination imply well-typedness?

Forward Data Flow Analysis (FDFA)

Semilattice

Def Let $\sqsubseteq :: 'a \Rightarrow 'a \Rightarrow bool$ be a partial ordering.

- a is an *upper bound* of a_1 and a_2 iff $a_1 \sqsubseteq a$ and $a_2 \sqsubseteq a$.
- a is the *least upper bound* (*supremum*) of a_1 and a_2 iff
  - a is an upper bound and
  - for every upper bound a' we have $a \sqsubseteq a'$.
- $\sqsubseteq$ is a *semilattice* if any two elements have a supremum (written $a_1 \sqcup a_2$).

Formalization of FDFA

Nodes of CFG $N = \{0,\ldots,n-1\}$ (later: $n = |is|$)

Abstract program states S

Transfer function $Step :: N \Rightarrow S \Rightarrow S$ (later: app + eff)

Successors $Succs :: N \Rightarrow N\ set$

Ordering $\sqsubseteq :: S \Rightarrow S \Rightarrow bool$

Assumption $\sqsubseteq$ is a semilattice (with supremum $\sqcup$).

Def $\tau s :: S\ list$ is *stable at* $p \in N$ iff
$\forall q \in Succs\ p.\ Step\ p\ \tau s_{[p]} \sqsubseteq \tau s_{[q]}$.

τs is *stable* iff it is stable at all $p \in N$.

Fixpoint iteration (Kildall's algorithm)

Starting with some initial τs (such that $|\tau s| = n$)
update τs until it is stable:

`while` there is a $p \in N$ such that τs is not stable at p do
  `for all` $q \in Succs\ p$ do $\tau s_{[q]} := \tau s_{[q]} \sqcup Step\ p\ \tau s_{[p]}$

Functional version: $dfa :: S\ list \Rightarrow S\ list$

Example

Theorems

Thm If every $\sqsubset$-chain ($s_0 \sqsubset s_1 \sqsubset \ldots$) is finite then Kildall terminates.

Thm Let *Step* be monotone:

$$\forall\, p \in N.\ \forall\, \tau\ \tau'.\ \tau \sqsubseteq \tau' \longrightarrow \mathit{Step}\ p\ \tau \sqsubseteq \mathit{Step}\ p\ \tau'.$$

Let τs_0 be the initial and $\tau s'$ be the final value of a terminating run of Kildall.
Then $\tau s'$ is the least stable list $\sqsupseteq \tau s_0$.

Refining the framework

Assumption $S = T\ err$ for some T.

Lemma Let *Step* be monotone. Then
$Err \notin set\,(dfa\ \tau s_0) = (\exists \tau s \sqsupseteq \tau s_0.\ stable\ \tau s \wedge Err \notin set\ \tau s)$

Assumption There are two functions $App :: pc \Rightarrow T \Rightarrow bool$ and $Eff :: pc \Rightarrow T \Rightarrow T$ and *Step* is defined as follows:

$$Step\ p\ Err = Err$$
$$Step\ p\ (OK\ \tau) = if\ App\ p\ \tau\ then\ OK\,(Eff\ p\ \tau)\ else\ Err$$

Refining the framework

Def *App* is *monotone* iff
$\forall p \in N.\ \forall \tau\ \tau'.\ \tau \sqsubseteq \tau' \longrightarrow App\ p\ \tau' \longrightarrow App\ p\ \tau.$
Eff is *monotone* iff
$\forall p \in N.\ \forall \tau\ \tau'.\ \tau \sqsubseteq \tau' \longrightarrow Eff\ p\ \tau \sqsubseteq Eff\ p\ \tau.$

Lemma *Step* is monotone if *App* and *Eff* are monotone.

Corollary Let *App* and *Eff* be monotone. Then
$Err \notin set(dfa\ \tau s_0) =$
$(\exists \tau s \sqsupseteq \tau s_0.\ \forall p{<}n.\ \exists \tau.\ \tau s_{[p]} = OK\ \tau \wedge App\ p\ \tau \wedge$
$(\forall q \in Succs\ p.\ Eff\ p\ \tau \sqsubseteq \tau s_{[q]}))$

Close to well-typedness of methods.

The BV as an Instance of FDFA

Instantiating the framework

- $n = |is|$
- $Succs\ p \equiv succs\ is_{[p]}\ p$
- $App\ p\ \tau \equiv app\ is_{[p]}\ \tau$
- $Eff\ p\ \tau \equiv eff\ is_{[p]}\ \tau$

Instantiating T

T is (a subset of) ty_i'

Needed: $S = T\ err$ is a semilattice

Note: $\sqsubseteq$ on ty_i' is not a semilattice.

Example: $\lfloor([Integer], [])\rfloor \sqcup \lfloor([Boolean], [])\rfloor = ?$

Now: prove that $ty_i'\ err$ is a semilattice by induction over the structure of $ty_i' = (ty\ list \times ty\ err\ list)option$.

Constructing semilattices

Lemma $\sqsubseteq :: 'a\ err \Rightarrow 'a\ err \Rightarrow bool$ is a semilattice iff the restriction $\sqsubseteq :: 'a \Rightarrow 'a \Rightarrow bool$ has the following property:

> if $a_1, a_2 :: 'a$ have an upper bound,
> they have a least upper bound.

Remember: $\sqsubseteq$ on *ty* is the subtype relation (w.r.t. *P*).

Lemma If the class hierarchy is acyclic, $\sqsubseteq$ on *ty err* is a semilattice.

Constructing semilattices

Lemma If $\sqsubseteq$ on $'a\ err$ is a semilattice,
then $\sqsubseteq$ on $'a\ option\ err$ is a semilattice, too.

Lemma If $\sqsubseteq$ on $'a\ err$ and on $'b\ err$ are semilattices,
then $\sqsubseteq$ on $('a \times 'b)\ err$ is a semilattice, too.

Lemma If $\sqsubseteq$ on $'a\ err$ is a semilattice,
then $\sqsubseteq$ on $'a\ list\ err$ is a semilattice, too.

Lemma If $\sqsubseteq$ on $'a$ is a semilattice,
then $\sqsubseteq$ on $'a\ err$ is a semilattice, too.

Instantiating T

Thm If the class hierarchy is acyclic, then $\sqsubseteq$ on
$ty_i'\ err = (ty\ list \times ty\ err\ list)option\ err$
is a semilattice.

Def $T = (\{ST :: ty\ list \mid |ST| \leq mxs\} \times \{LT :: ty\ err\ list \mid |LT| = mxl\})option$

Corollary If the class hierarchy is acyclic, then $\sqsubseteq$ on
$S = T\ err$ is a semilattice.

Proving further assumptions

Lemma *Eff* is of type $pc \Rightarrow T \Rightarrow T$.

Thm Both *App* and *Eff* are monotone.

Termination

Thm If the class hierarchy is acyclic, all $\sqsubset$-chains in T are finite. If m is the maximal length of any subclass chain ($\prec^1$), the maximal length of any $\sqsubset$-chain in T is $O(m*(mxs+mxl))$.

Remember

The method is well-typed w.r.t. τs iff

- $is \neq [] \wedge |\tau s| = |is|$
- $\forall\, ST\ LT.\ \lfloor (ST, LT) \rfloor \in set\ \tau s \longrightarrow |ST| < mxs \wedge |LT| = mxl$
- $\tau_0 \sqsubseteq \tau s_{[0]}$ where
  $\tau_0 = \lfloor ([], OK\ (Class\ C_M) \cdot map\ OK\ Ts\ @\ replicate\ mxl_0\ Err) \rfloor$
- $\forall\, p < |is|.\ \forall\, q \in succs\ is_{[p]}\ p.\ q < |is|$
- $\forall\, p < |is|.\ app\ is_{[p]}\ \tau s_{[p]}$
- $\forall\, p < |is|.\ \forall\, q \in succs\ is_{[p]}\ p.\ eff\ is_{[p]}\ \tau s_{[p]} \sqsubseteq \tau s_{[q]}$

Well-typedness

Def The method is well-typed iff there is a τs such that the method is well-typed w.r.t. τs.

Thm The method is well-typed iff

- $is \neq []$
- $\forall\, p < |is|.\ \forall\, q \in succs\ is_{[p]}\ p.\ q < |is|$
- $Err \notin set\ (dfa\ (OK\ \tau_0 \cdot replicate\ (n - 1)\ (OK\ None)))$.

Example

`Load` *0*	*OK* $\lfloor([], [OK\ (Class\ B)])\rfloor$
`Goto` -1	*OK None*