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

<table>
<thead>
<tr>
<th>Load 0</th>
<th>( [(] , [\text{OK (Class B), OK Integer}] ) ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store 1</td>
<td></td>
</tr>
<tr>
<td>Load 0</td>
<td></td>
</tr>
<tr>
<td>Getfield ( FA )</td>
<td></td>
</tr>
<tr>
<td>Goto (-3)</td>
<td></td>
</tr>
</tbody>
</table>

**Assumptions:**

\( P \vdash B \preceq^* A \)

\( P \vdash A \text{ sees } F: \text{Class A in A} \)
What if we remove Store 1?

<table>
<thead>
<tr>
<th>Load 0</th>
<th>([([]), [\text{OK (Class B), OK Integer}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load 0</td>
<td></td>
</tr>
<tr>
<td>Getfield F A</td>
<td></td>
</tr>
<tr>
<td>Goto −3</td>
<td></td>
</tr>
</tbody>
</table>
Questions

- Why does fixed point iteration always terminate?
- Why does successful termination imply well-typedness?
Forward Data Flow Analysis (FDFA)
**Semilattice**

**Def** Let $\sqsubseteq : \mathit{\text{'a}} \to \mathit{\text{'a}} 	o \text{bool}$ 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 \( \text{Step} :: N \Rightarrow S \Rightarrow S \) (later: \( \text{app} + \text{eff} \))

Successors \( \text{Succs} :: N \Rightarrow N \text{ set} \)

Ordering \( \sqsubseteq :: S \Rightarrow S \Rightarrow \text{bool} \)

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

Def \( \tau S :: S \text{ list} \) is stable at \( p \in N \) iff
\[
\forall q \in \text{Succs } p. \text{Step } p \tau s[p] \sqsubseteq \tau s[q].
\]
\( \tau S \) is stable iff it is stable at all \( p \in N \).
Fixpoint iteration (Kildall’s algorithm)

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

\[
\text{while there is a } p \in N \text{ such that } \tau s \text{ is not stable at } p \text{ do }
\]
\[
\text{for all } q \in \text{Succs } p \text{ do } \tau s[q] := \tau s[q] \cup \text{Step } p \tau s[p]
\]

Functional version: $\text{dfa} :: S \text{ list } \Rightarrow S \text{ list}$
Example
Theorems

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

Thm Let $\text{Step}$ be monotone:

$$\forall p \in \mathbb{N}. \forall \tau \tau'. \tau \sqsubseteq \tau' \longrightarrow \text{Step } p \tau \sqsubseteq \text{Step } p \tau'.$$

Let $\tau s_0$ 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
$$Err \notin set(dfa \tau s_0) = (\exists \tau s \supseteq \tau s_0. \text{stable } \tau s \land Err \notin set \tau s)$$

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

$$Step\ p\ Err = Err$$

$$Step\ p\ (OK\ \tau) = \text{if } App\ p\ \tau \text{ then } OK(Eff\ p\ \tau) \text{ else } Err$$
Refining the framework

**Def** App is *monotone* iff
\[ \forall p \in \mathbb{N}. \forall \tau \tau'. \tau \subseteq \tau' \rightarrow \text{App} p \tau' \rightarrow \text{App} p \tau. \]

Eff is *monotone* iff
\[ \forall p \in \mathbb{N}. \forall \tau \tau'. \tau \subseteq \tau' \rightarrow \text{Eff} p \tau \subseteq \text{Eff} p \tau. \]

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

**Corollary** Let App and Eff be monotone. Then
\[
\text{Err} \notin \text{set}(\text{dfa } \tau s_0) = \\
(\exists \tau s \supseteq \tau s_0. \forall p < n. \exists \tau. \tau s[p] = \text{OK} \ \tau \land \text{App} p \tau \land \\
(\forall q \in \text{Succs } p. \text{Eff} p \tau \subseteq \tau s[q]))
\]

Close to well-typedness of methods.
The BV as an Instance of FDFA
Instantiating the framework

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

$T$ is (a subset of) $ty_{i'}$

Needed: $S = T \text{err}$ is a semilattice

Note: $\subseteq$ on $ty_{i'}$ is not a semilattice.

Example: $\llbracket\llbracket \text{Integer}, [] \rrbracket \rrbracket \sqcup \llbracket \llbracket \text{Boolean}, [] \rrbracket \rrbracket = ?$

Now: prove that $ty_{i'} \text{err}$ is a semilattice by induction over the structure of $ty_{i'} = (ty\ \text{list} \times ty\ \text{err}\ \text{list})\ \text{option}$. 
Lemma $\sqsubseteq :: \text{'a err } \Rightarrow \text{'a err } \Rightarrow \text{bool}$ is a semilattice iff the restriction $\sqsubseteq :: \text{'a } \Rightarrow \text{'a } \Rightarrow \text{bool}$ has the following property:

if $a_1, a_2 :: \text{'a}$ 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 $ty \text{ err}$ is a semilattice.
Lemma If \( \subseteq \) on \( 'a \text{ err} \) is a semilattice, then \( \subseteq \) on \( 'a \text{ option err} \) is a semilattice, too.

Lemma If \( \subseteq \) on \( 'a \text{ err} \) and on \( 'b \text{ err} \) are semilattices, then \( \subseteq \) on \( ('a \times 'b) \text{ err} \) is a semilattice, too.

Lemma If \( \subseteq \) on \( 'a \text{ err} \) is a semilattice, then \( \subseteq \) on \( 'a \text{ list err} \) is a semilattice, too.

Lemma If \( \subseteq \) on \( 'a \) is a semilattice, then \( \subseteq \) on \( 'a \text{ err} \) is a semilattice, too.
**Instantiating $T$**

**Thm** If the class hierarchy is acyclic, then $\subseteq$ 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 $\subseteq$ 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 \( \sqsupseteq \)-chains in \( T \) are finite. If \( m \) is the maximal length of any subclass chain \((\prec_1)\), the maximal length of any \( \sqsupseteq \)-chain in \( T \) is \( O(m* (mxs + mxl)) \).
The method is well-typed w.r.t. $\tau$ iff

1. $is \neq [] \land |\tau s| = |is|$
2. $\forall ST LT. \left[ (ST, LT) \right] \in set \tau s \rightarrow |ST| < mxs \land |LT| = mxl$
3. $\tau_0 \sqsubseteq \tau s_{[0]}$ where
   $\tau_0 = \left[ ([], OK \ (Class \ C_M) \cdot map \ OK \ Ts \ @ \ replicate \ mxl_0 \ Err) \right]$
4. $\forall p < |is|. \forall q \in succs is_{[p]} p. q < |is|$
5. $\forall p < |is|. \ app is_{[p]} \tau s_{[p]}$
6. $\forall p < |is|. \forall q \in succs is_{[p]} p. \ eff is_{[p]} \tau s_{[p]} \sqsubseteq \tau s_{[q]}$
Def The method is well-typed iff there is a $\tau s$ such that the method is well-typed w.r.t. $\tau s$.

Thm The method is well-typed iff

- $is \neq []$
- $\forall p < |is|. \forall q \in \text{succs } is[p] \ p. q < |is|$
- $\text{Err } \not\in \text{set } (\text{dfa } (OK \ \tau_0 \cdot \text{replicate } (n - 1) \ (OK \text{ None})))$. 
<table>
<thead>
<tr>
<th>Load 0</th>
<th>OK  $(\emptyset, [OK (Class B)])$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goto $-1$</td>
<td>OK None</td>
</tr>
</tbody>
</table>