Constraint Semantics for Abstract Read Permissions

John Tang Boyland (UW-Milwaukee/ETH Zurich)
Peter Müller, Malte Schwerhoff, Alexander J. Summers (ETH Zurich)

28th July 2014, FTfJP, Uppsala
Unbounded Counting with Fractional Permissions over $\mathbb{Q}$

General Framework for Proving Soundness of Permission Assumptions
foo(x)  bar(x)
Modular Static Verification + Shared State

foo(x)  bar(x)
Permissions

$\text{foo}(x)$  $\text{bar}(x)$
Permission Transfer

foo(x)  bar(x)
foo(x)  bar(x)
Fractional Permissions

foo(x)  bar(x)
Splitting Fractional Permissions

foo(x) bar(x)

Q: \( \frac{1}{2} + \frac{1}{2} \) or \( \frac{1}{4} + \frac{3}{4} \) or ...
Merging Fractional Permissions

foo(x)  bar(x)
method visit(it: Iterator, d: Data) {
    var n: Int := 0

    while (it.hasNext()) {
        fork worker(it.next(), d)
        n := n + 1
    }

    ...
}

method worker(e: Element, d: Data) {
    // reads d.f
}
method visit(it: Iterator, d: Data) {
    var n: Int := 0
    while (it.hasNext()) {
        fork worker(it.next(), d)
        n := n + 1
    }
    ...
}

method worker(e: Element, d: Data) {
    // reads d.f
}
method visit(it: Iterator, d: Data) {
    var n: Int := 0
    while (it.hasNext()) {
        fork worker(it.next(), d)
        n := n + 1
    }
    ...
}

method worker(e: Element, d: Data) {
// reads d.f
}
method visit(it: Iterator, d: Data) {
    var n: Int := 0
    while (it.hasNext()) {
        fork worker(it.next(), d)
        n := n + 1
    }
    ...
}

method worker(e: Element, d: Data) {
    // reads d.f
}
Iterator Example

```java
method visit(it: Iterator, d: Data) {
    var n: Int := 0
    while (it.hasNext()) {
        fork worker(it.next(), d)
        n := n + 1
    }
}
...
}

method worker(e: Element, d: Data) {
    // reads d.f
}
```

Can \((n+1)\textsuperscript{th}\) \(v_2\) still be given away?

... on the other hand, for any \(n\), a suitable \(v_2\) can be chosen
Abstract Read Permissions

(Heule et al., VMCAI’13)

1. Use symbolic values $v_i$ instead of concrete fractions

2. Constrain $v_i$ as verification proceeds to make it sufficiently small
method visit(it: Iterator, d: Data, v1: Perm) {
    var n: Int := 0
    var v2: Perm := fresh()

    while (it.hasNext()) {
        fork worker(it.next(), d, v2)
        n := n + 1
    }

    ...
}

method worker(e: Element, d: Data, v: Perm) {
    // reads d.f
}
method visit(it: Iterator, d: Data, v1: Perm) {
    var n: Int := 0
    var v2: Perm := fresh()

    while (it.hasNext()) {
        assume v2 < v1 - n * v2
        fork worker(it.next(), d, v2)
        n := n + 1
    }

    ...
}

method worker(e: Element, d: Data, v: Perm) {
    // reads d.f
}
Other Permission Model Properties (Short Digression)

```java
method visit(it: Iterator, d: Data, v1: Perm) {
    var n: Int := 0
    var v2: Perm := fresh()

    while (it.hasNext()) {
        assume v2 < v1 - n * v2
        fork worker(it.next(), d, v2)
        n := n + 1
    }

    ...
}
```

Recursively fork visit

→ unbounded splitting

```java
method worker(e: Element, d: Data, v: Perm) {
    // reads d.f
}
```
Other Permission Model Properties (Short Digression)

```java
method visit(it: Iterator, d: Data, v1: Perm) {
    var n: Int := 0
    var v2: Perm := fresh()

    while (it.hasNext()) {
        assume v2 < v1 - n * v2
        fork worker(it.next(), d, v2)
        n := n + 1
    }
}

...  
}

method worker(e: Element, d: Data, v: Perm) {
    // reads d.f
}
```

Abstract predicates

→ permission multiplication
Permission Assumptions Sound?

⇔

Constraint System Satisfiable?
Abstract Read Permissions (Heule et al., VMCAI’13)

- Must avoid unsound assumptions (in a modular way)
  - $v < v$
  - $v_1 < v_2 \&\& v_2 < v_1$

- Factoring key property for constraint satisfiability out of general soundness proof is not straight-forward

- Heule et al. therefore allow only limited application of abstract read permissions
  - Ensures satisfiable constraints
  - Can’t support unbounded counting
Layered Constraints
Layered Constraints

\[ \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \ldots \quad q \in \mathbb{Q}^+ \]

\begin{align*}
&v_1 & v_3 \\
&v_4 & v_7 \\
&v_6 \\
&v_5 & v_2 & v_8
\end{align*}

0 ------------------------ 0
Layered Constraints

Partially ordered set of variables \((V, \prec)\)
\(v_2 \prec v_1 \equiv \text{“}v_2\text{ layered below } v_1\text{”}\)

Extend \(\prec\) to \(v \prec E\)
\[E ::= q \mid v \mid E + E \mid E \times E \mid E - E \mid E / E\]

Define *layered* for sets \(C\) of permission inequalities \(v < E\)
Extending $< \text{to } E$

$q \in \mathbb{Q}^+$  
\[ v < q \]

\[ v < E_1 \quad v < E_2 \]
\[ v < E_1 \odot E_2 \]
with $\odot \in \{+, \ast, /\}$

\[ v < v' \quad v' < E \]
\[ v < E - v' \]
Extending $<$ to $E$

$q \in \mathbb{Q}^+$

<table>
<thead>
<tr>
<th>$v &lt; E_1$</th>
<th>$v &lt; E_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v &lt; E_1 \otimes E_2$</td>
<td></td>
</tr>
</tbody>
</table>

with $\otimes \in \{+, \ast, /\}$

<table>
<thead>
<tr>
<th>$v &lt; v'$</th>
<th>$v' &lt; E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v &lt; E - v'$</td>
<td></td>
</tr>
</tbody>
</table>

Ensures that $v' < E$ could be added to $C$
Extending $\prec$ to $E$

$q \in \mathbb{Q}^+$

$\begin{array}{c}
q \in \mathbb{Q}^+ \\
\hline \\
v \prec q
\end{array}$

$v < E_1$  $v < E_2$

$\begin{array}{c}
v \prec E_1 \odot E_2 \\
\hline \\
\text{with } \odot \in \{+, *, /\}
\end{array}$

$v < v'$  $v' < E$

$\begin{array}{c}
v \prec E - v' \\
\hline \\
\end{array}$

Ensures that $v$ is layered below $v'$
Layered and Layerable Constraints

- A set $C$ of permission inequalities $v < E$
  is layered w.r.t. to $<$ if

  \[ \forall (v < E) \in C \cdot v < E \]
Layered and Layerable Constraints

- A set $C$ of permission inequalities $v < E$ is layered w.r.t. to $<$ if

\[ \forall (v < E) \in C \cdot \exists (v' < E') \cdot (v' < E' \Rightarrow v < E) \land v' < E' \]

- A set $C$ is layerable if there exists a $<$ s.t. $C$ is layered w.r.t. to $<$

**Theorem:** If a constraint system is layerable, then it is satisfiable

Makes simple derivation system more expressive
Iterator Example Revisited

$q \in \mathbb{Q}^+$

\[
\begin{align*}
q \in \mathbb{Q}^+ & \quad \frac{v \prec E_1 \quad v \prec E_2}{v \prec E_1 \odot E_2} \\
& \quad \text{with } \odot \in \{+, *, /\}
\end{align*}
\]

Constraint from iterator example

\[
v_2 \prec v_1 - n \cdot v_2
\]

is layered if

\[
v_2 \prec v_1 - n \cdot v_2
\]

is derivable
Iterator Example Revisited

\[ q \in \mathbb{Q}^+ \]

\[ v < E_1 \quad v < E_2 \]

\[ v < E_1 \odot E_2 \]

with \( \odot \in \{+, \ast, /\} \)

\[ v < v' \quad v' < E \]

\[ v < E - v' \]

Constraint from iterator example

\[ v_2 < v_1 - n \ast v_2 \]

is layered if

\[ v_2 < v_1 - n \ast v_2 \Leftrightarrow v_2 < v_1 / (n + 1) \]

is derivable
**Iterator Example Revisited**

<table>
<thead>
<tr>
<th>q ∈ ℚ⁺</th>
<th>v &lt; E₁</th>
<th>v &lt; E₂</th>
<th>v &lt; v'</th>
<th>v' &lt; E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>v &lt; E₁ ⊗ E₂</td>
<td></td>
<td>v &lt; E₁ ⊗ E₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with ⊗ ∈ {+, *, /}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\frac{v_2 < v_1 \quad (n + 1) ∈ ℚ⁺}{v_2 < v_1 \quad v_2 < (n + 1) \quad v_2 < v_1 \quad / (n + 1)\}.
\]
Using our Work

1. Define a suitable $\prec$-relation, e.g., introduced after

   ```
   method visit(v1: Perm) {
     ...
     var v2: Perm := fresh()
     ...
   }
   ```

2. Define a methodology for generating layered constraints w.r.t. to chosen $\prec$ (soundness proof)

3. Turn constraints into free assumptions
Related Work

- Counting Permissions (Bornat et al., POPL’05)
  - Neither divisibility nor multiplication

- Compound Models (Dockins et al., APLAS’09, Leino et al., ESOP’09)
  - No multiplication
  - Potentially slow due to disjunctions

- $\mathbb{Z}[\varepsilon]^+$ (Boyland, LNCS Volume 7850, 2013)
  - Satisfies all three properties
  - Complex and subtle model
  - No existing implementation (as far as we know)
Conclusion

- Identified a property that guarantees satisfiability of constraints over fractional permissions over $\mathbb{Q}$

- Formalised a derivation system enforcing the property

- Enabled fractional permissions over $\mathbb{Q}$ to support unbounded counting

- Factor soundness proof for permission book-keeping out of general soundness proof

- Future work: define methodology for fully exploiting layerable constraints in a front-end tool
Questions?

www.pm.inf.ethz.ch
malte.schwerhoff@inf.ethz.ch