Proving the Correctness of Fractional Permissions for a Java-like Kernel Language

John Boyland, Chao Sun
University of Wisconsin - Milwaukee

FOOL 2011

Summary

- Soundness proof for permission type system
  1. Based on a Java-like kernel language;
  2. Machine-checked using Twelf.
- Soundness proof for a non-null type system
  1. Reducing to fractional permissions;
  2. Reusing the existing proof for the first.

Fractional Permissions

- A system for managing access to mutable states;
- Each field is associated with a permission, represents its accessibility to the object;
- Nesting is used to model object invariants and ownership.

Basic Permission

- Basic permission : $o.f \to o'$
  - Represents full access (read and write) to the field f of object o ;
  - Gives additional information that points to another object o';
  - **Linear**: cannot be duplicated.

Formula

- Represents facts that remain true;
- Non-linear;
  - can be duplicated and discarded.
- Most noticeable one: $\Pi \prec o.f$

Other Permissions

$$\begin{aligned}
&\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \to i) \prec r.\mathrm{All} \wedge \\
&\quad (\exists n \cdot r.\mathrm{next} \to n + \\
&\quad n = 0 \,?\, \emptyset : n.\mathrm{All} \to 0 \; + \mathrm{Node}(n)) \prec r.\mathrm{All}
\end{aligned}$$

Other Permissions

class predicate

$$
\begin{aligned}
\boxed{\mathrm{Node}(r)} = {} & (\exists i \cdot r.\mathrm{data} \to i) \prec r.\mathrm{All} \wedge \\
& (\exists n \cdot r.\mathrm{next} \to n + \\
& n = 0\,?\,\emptyset : n.\mathrm{All} \to 0 \; + \mathrm{Node}(n)) \prec r.\mathrm{All}
\end{aligned}
$$

Other Permissions

existential

$$
\begin{aligned}
\mathrm{Node}(r) = \boxed{(\exists i \cdot r.\mathrm{data} \rightarrow i)} \prec r.\mathrm{All} \wedge \\
(\exists n \cdot r.\mathrm{next} \rightarrow n + \\
n = 0\,?\,\emptyset : n.\mathrm{All} \rightarrow 0\ + \mathrm{Node}(n)) \prec r.\mathrm{All}
\end{aligned}
$$

Other Permissions

$$
\begin{aligned}
&\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \rightarrow i) \prec r.\mathrm{All} \wedge \\
&\quad (\exists n \cdot r.\mathrm{next} \rightarrow n + \\
&\quad \boxed{(n = 0\,?\,\emptyset : n.\mathrm{All} \rightarrow 0 \;+ \mathrm{Node}(n))}) \prec r.\mathrm{All}
\end{aligned}
$$

conditional

Other Permissions

$$
\begin{aligned}
&\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \rightarrow i) \prec r.\mathrm{All} \, \wedge \\
&\quad (\exists n \cdot r.\mathrm{next} \rightarrow n \, + \\
&\quad n = 0 \; ? \; \emptyset : n.\mathrm{All} \rightarrow 0 \; + \mathrm{Node}(n)) \prec r.\mathrm{All}
\end{aligned}
$$

empty permission

Other Permissions

$$\begin{aligned}
&\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \to i) \prec r.\mathrm{All} \wedge \\
&\quad (\exists n \cdot r.\mathrm{next} \to n + \\
&\quad n = 0 \;?\; \emptyset : n.\mathrm{All} \to 0 \; + \mathrm{Node}(n)) \prec r.\mathrm{All}
\end{aligned}$$

nesting

Other Permissions

$$\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \rightarrow i) \prec r.\mathrm{All} \land$$
$$(\exists n \cdot r.\mathrm{next} \rightarrow n +$$
$$n = 0\,?\,\emptyset : n.\mathrm{All} \rightarrow 0 \;+ \mathrm{Node}(n)) \prec r.\mathrm{All}$$

Other Permissions

$$\begin{aligned}\mathrm{Node}(r) = {} & (\exists i \cdot r.\mathrm{data} \to i) \prec r.\mathrm{All} \wedge \\ & (\exists n \cdot r.\mathrm{next} \to n + \\ & n = 0\,?\,\emptyset : n.\mathrm{All} \to 0 + \mathrm{Node}(n)) \prec r.\mathrm{All}\end{aligned}$$

Other Permissions

$$\mathrm{Node}(r) = (\exists i \cdot r.\mathrm{data} \rightarrow i) \prec r.\mathrm{All} \wedge$$
$$(\exists n \cdot r.\mathrm{next} \rightarrow n + \boxed{n = 0\,?\,\emptyset : n.\mathrm{All} \rightarrow 0\ + \mathrm{Node}(n))} \prec r.\mathrm{All}$$

Transformation

$$(\exists r \cdot (o.f \rightarrow r + (r = 0 \,?\, \emptyset : r.\mathrm{All}))) \prec o.\mathrm{All}$$

$$+\ o.\mathrm{All} \rightarrow 0$$

Transformation

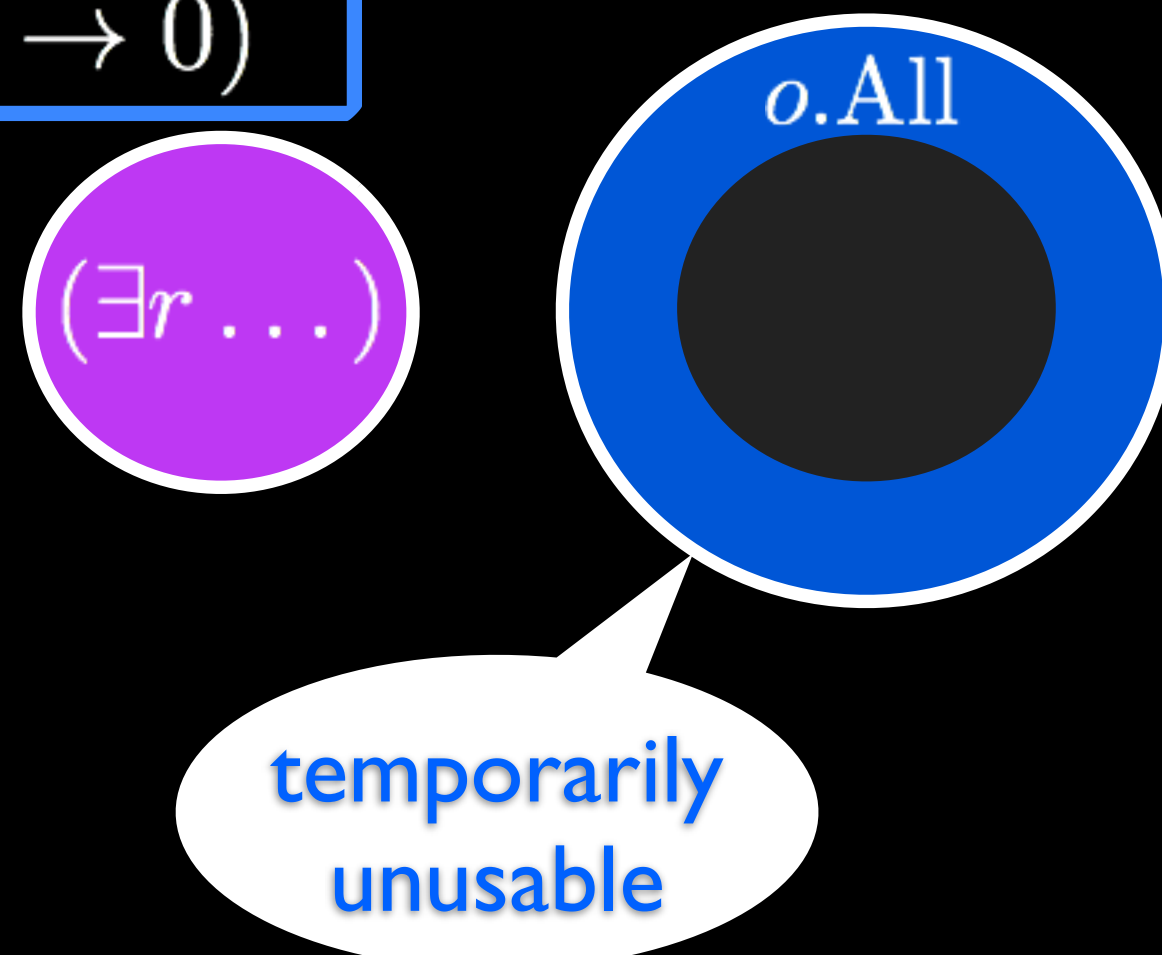
$$\boxed{\begin{aligned}(\exists r \cdot (o.f \rightarrow r + (r = 0\,?\,\emptyset : r.\mathrm{All}))) \prec o.\mathrm{All} \\ + \; o.\mathrm{All} \rightarrow 0\end{aligned}}$$

Transformation

$$(\exists r \cdot (o.f \to r + (r = 0\,?\,\emptyset : r.\mathrm{All}))) \prec o.\mathrm{All}$$
$$+\; o.\mathrm{All} \to 0$$

carving out

$$(\exists r \ldots) + ((\exists r \ldots) \mathrel{-\!\!\!+} o.\mathrm{All} \to 0)$$

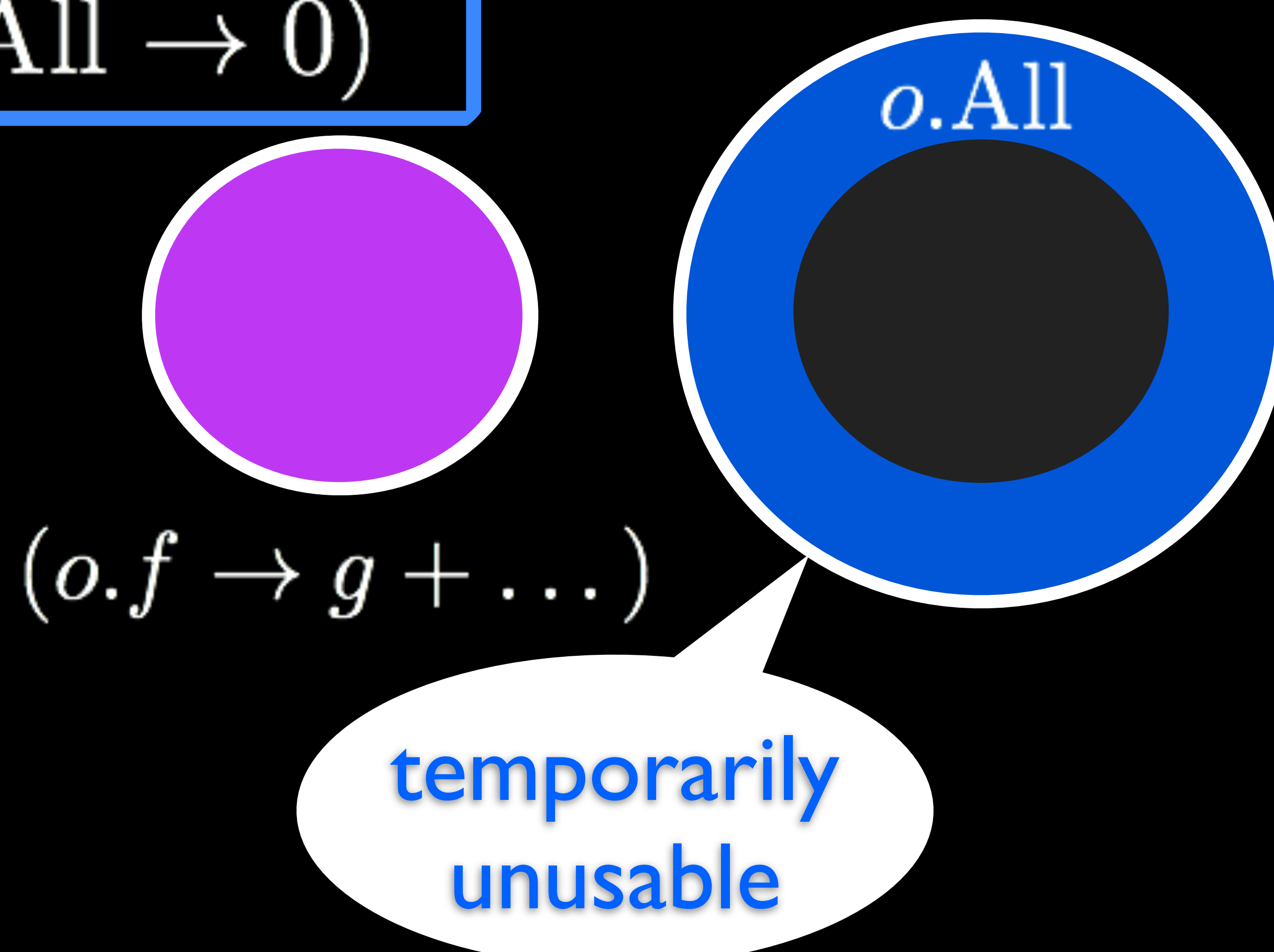


Transformation

$$(\exists r \cdot (o.f \rightarrow r + (r = 0\,?\,\emptyset : r.\mathrm{All}))) \prec o.\mathrm{All}$$
$$+\ o.\mathrm{All} \rightarrow 0$$

carving out

$$(\exists r \ldots) + ((\exists r \ldots) \mathbin{-\!\!+} o.\mathrm{All} \rightarrow 0)$$

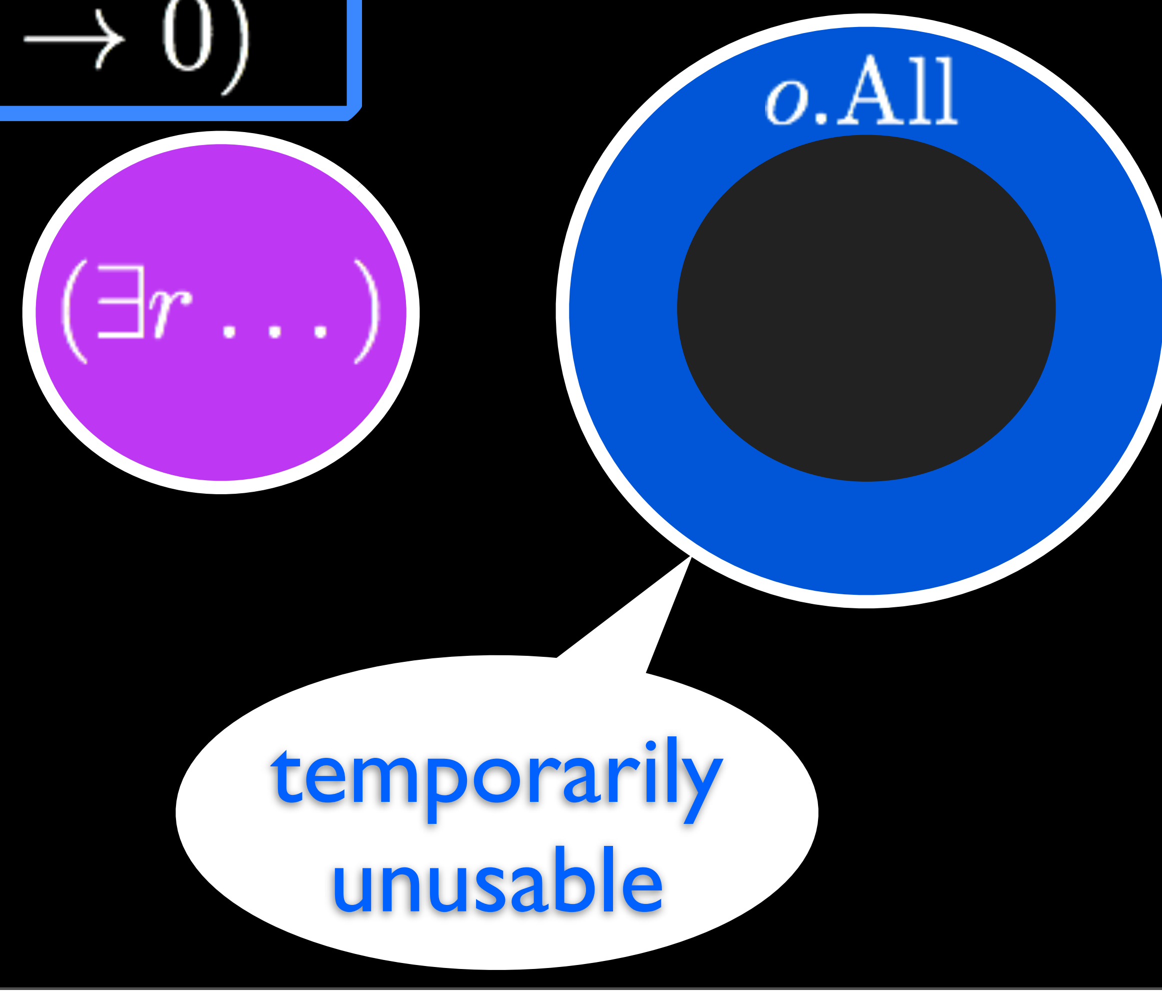


Transformation

$$\boxed{\begin{array}{l}(\exists r \cdot (o.f \to r + (r = 0 \,?\, \emptyset : r.\mathrm{All}))) \prec o.\mathrm{All} \\ \quad + \; o.\mathrm{All} \to 0\end{array}}$$

carving out

$$\boxed{(\exists r \ldots) + ((\exists r \ldots) \mathbin{-\!\!+} o.\mathrm{All} \to 0)}$$



Transformation

$$(\exists r \cdot (o.f \to r + (r = 0\ ?\ \emptyset : r.\mathrm{All}))) \prec o.\mathrm{All}$$
$$+\ o.\mathrm{All} \to 0$$

carving out

$$(\exists r \ldots) + ((\exists r \ldots) \mathrel{-\!\!+} o.\mathrm{All} \to 0)$$

linear *modus ponens*

$$o.\mathrm{All} \to 0$$

Kernel Language

$$
\begin{array}{lll}
e & ::= & o \mid x \mid \texttt{new}\ C(\overline{f}) \mid e\texttt{.}f \mid e\texttt{.}f\texttt{:=}e \\
 & & \mid \texttt{let}\ x\texttt{=}e\ \texttt{in}\ e \mid \texttt{while}\ c\ \texttt{do}\ e \\
 & & \mid \texttt{if}\ c\ \texttt{then}\ e\ \texttt{else}\ e \mid m(\overline{e}) \\
c & ::= & \texttt{true} \mid \texttt{not}\ c \mid c\ \texttt{and}\ c \mid e\ \texttt{==}\ e \\
d & ::= & m(\overline{x})\ \texttt{=}\ e \\
g & ::= & d\texttt{;}\ldots\texttt{;}d
\end{array}
$$

$$(e; \mu) \rightarrow_g (e'; \mu')$$

concurrency is omitted here

Permission Type

Procedure type

$$\alpha ::= \forall_{\Delta}\ \Pi \rightarrow \exists_{\Delta'}\ \Pi'$$

Program type

$$\omega ::= \{\overline{m \mapsto \alpha}\}$$

Type Judgment

$$\Delta; \Pi \vdash_{\omega} e \Downarrow \rho \dashv \Delta'; \Pi'$$

P-Write

writable

$$\textsc{P-Write}\quad \frac{\begin{array}{c}\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \\ \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \boxed{\rho_1.f \to \rho'} + \Pi'\end{array}}{\Delta_1;\Pi_1 \vdash_\omega e_1.f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \boxed{\rho_1.f \to \rho_2} + \Pi'}$$

value updated

Consistency

- Between Fractional Permissions and Memory

1. Evaluation depends on memory;
2. Type checking depends on permission;
3. Fractional heaps connect the two.

$$h : (O \times F) \rightarrow (\mathbb{Q}^{+} \times O)$$

Consistency

$$\frac{\begin{array}{c}\forall_{(o,f)\mapsto(q,o')\in h}\ \mu\, o\, f = o' \\ \Pi' = \sum\{\Pi_{o,f} \mid (o,f)\mapsto \Pi_{o,f}\in N, \mu\, o\, f\ \text{undefined}\} \\ h \models_N \Pi + \Pi'\end{array}}{\Pi\ \text{consistent with}\ \mu\ \text{using}\ N}$$

How to Prove?

- Standard approach:
  1. Define syntax and semantics;
  2. Define type system;
  3. Define consistency;
  4. Prove progress/preservation.
- All checked in Twelf

Piggy-backing Proof

- Prove soundness of one type system by <u>reducing</u> to another (more powerful) one;
- Reuse machine-checked proof.
- No dynamic semantics;
- No progress/preservation needed.

Non-null Type

- Same kernel language;
- Reference type is augmented to be either not-null or possibly-null;
- No restriction on reference access
  - i.e., every reference is owned by “world”
- Constructor is restricted to avoid leakage of `this` reference.

Not-Null

$$\textsc{NotNull}\quad \frac{E(x) = c^{+} \qquad C; M; E, x : c^{-} \vdash e_1 : c' \qquad C; M; E \vdash e_2 : c'}{C; M; E \vdash \texttt{if not } x\texttt{==}0 \texttt{ then } e_1 \texttt{ else } e_2 : c'}$$

Not-Null

$$\textsc{NotNull}\quad \frac{E(x) = c^{+} \qquad C; M; E, x : c^{-} \vdash e_1 : c' \qquad C; M; E \vdash e_2 : c'}{C; M; E \vdash \texttt{if not } x\texttt{==}0 \texttt{ then } e_1 \texttt{ else } e_2 : c'}$$

class map, method map and context

Not-Null

NotNull

$$\frac{E(x) = c^{+} \qquad C; M; E, x : c^{-} \vdash e_1 : c' \qquad C; M; E \vdash e_2 : c'}{C; M; E \vdash \texttt{if not } x\texttt{==}0 \texttt{ then } e_1 \texttt{ else } e_2 : c'}$$

class map, method map and context

$$\begin{aligned}
C &::= \epsilon \mid C, c : F \\
M &::= \epsilon \mid M, m : (c_1, \ldots, c_n) \to c \\
F &::= \epsilon \mid F, f : c \\
E &::= \epsilon \mid E, x : c
\end{aligned}$$

Not-Null

possibly null

$$\textsc{NotNull}\quad \frac{E(x) = c^{+} \qquad C; M; E, x : c^{-} \vdash e_1 : c' \qquad C; M; E \vdash e_2 : c'}{C; M; E \vdash \texttt{if not } x\texttt{==}0 \texttt{ then } e_1 \texttt{ else } e_2 : c'}$$

class map, method map and context

$$\begin{aligned} C &::= \epsilon \mid C, c : F \\ M &::= \epsilon \mid M, m : (c_1, \ldots, c_n) \to c \\ F &::= \epsilon \mid F, f : c \\ E &::= \epsilon \mid E, x : c \end{aligned}$$

Not-Null

not null

possibly null

NOTNULL

$$\frac{E(x) = c^{+} \qquad C;M;E,x:c^{-} \vdash e_1 : c' \qquad C;M;E \vdash e_2 : c'}{C;M;E \vdash \texttt{if not } x\texttt{==}0 \texttt{ then } e_1 \texttt{ else } e_2 : c'}$$

class map, method map and context

$$
\begin{aligned}
C &::= \epsilon \mid C, c : F \\
M &::= \epsilon \mid M, m : (c_1, \ldots, c_n) \rightarrow c \\
F &::= \epsilon \mid F, f : c \\
E &::= \epsilon \mid E, x : c
\end{aligned}
$$

Converting Class

For a class c with with fields $f_1, \ldots, f_n$:

Owned by "world"

$$p(o) \stackrel{\mathrm{def}}{=} (\exists r_1 \cdot (o.f_1 \rightarrow r_1 + \Pi_1) + \\ \ldots + \exists r_n \cdot (o.f_n \rightarrow r_n + \Pi_n)) \prec 0.\mathrm{Owned}$$

class predicate

where

$$\Pi_i = \begin{cases} \neg(r_i = 0) + p_i(r_i) & r_i \text{ not null} \\ \neg(r_i = 0) \mathbin{?} p_i(r_i) : \emptyset & r_i \text{ possibly null} \end{cases}$$

Converting Method

$$m : (c_1, \ldots, c_n) \to c_t$$

converted to

$$m : \forall r_1, \ldots, r_n \; (\Pi_1 + \ldots + \Pi_n + 0.\text{Owned} \to 0) \\ \longrightarrow \exists r_t \; (\Pi_t + 0.\text{Owned} \to 0)$$

permission to write world

Lemma

For every e in the kernel language, if it has type t in non-null system under C, M, and E, then after converting these to ω and Π, e is also well-typed under permission system. In addition, the output permission is $\Pi + \Pi_t$, where Π_t is the permission converted from t.

Example

NN-Write

$$\frac{C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C; M; E \vdash e_1\,.\,f\texttt{=}e_2 : c_2}$$

P-Write

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f\texttt{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho_2 + \Pi'}$$

Example

NN-Write

$$\dfrac{C;M;E \vdash e_1 : c_1^- \qquad C;M;E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C;M;E \vdash e_1 . f\text{=}e_2 : c_2}$$

P-Write

$$\dfrac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1 . f\text{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho_2 + \Pi'}$$

Example

NN-Write

$$\frac{C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C; M; E \vdash e_1 . f\text{=}e_2 : c_2}$$

P-Write

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1 . f\text{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

Example

$$\textsc{NN-Write}\quad \frac{C;M;E \vdash e_1 : c_1^- \qquad C;M;E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C;M;E \vdash e_1\,.\,f\texttt{=}e_2 : c_2}$$

$$\boxed{\Delta_1;\Pi_1} + \Pi''$$

$$\textsc{P-Write}\quad \frac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f\texttt{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho_2 + \Pi'}$$

P-Write

$$\frac{\begin{array}{c}\textsc{Write}\\ C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \\ C(c_1)(f) = c_2\end{array}}{C; M; E \vdash e_1\,.\,f\texttt{=}e_2 : c_2}$$

assuming e_2 is
not null and f
is annotated as not null

P-Write

$$\frac{\begin{array}{c}\textsc{Write} \\ C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \\ C(c_1)(f) = c_2\end{array}}{C; M; E \vdash e_1\,.\,f\texttt{=}e_2 : c_2}$$

$$\boxed{\neg(\rho_1 = 0) + p_1(\rho_1) + \Pi_1}$$

assuming e_2 is
not null and f
is annotated as not null

P-Write

WRITE

$$\frac{C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \qquad \mathcal{C}(c_1)(f) = c_2}{C; M; E \vdash e_1 \,.\, f\texttt{=}e_2 : c_2}$$

$$\boxed{\neg(\rho_1 = 0) + p_1(\rho_1) + \Pi_1}$$

assuming e_2 is
not null and f
is annotated as not null

$$\boxed{\neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1}$$

P-Write

$$\neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1$$

$$\Pi_1 = \Pi_1' + 0.\mathrm{Owned} \to 0$$

$$\begin{aligned}
&\exists\rho' \cdot (\rho_1.f \to \rho' + \Pi_f) \; + \\
&\exists\rho' \cdot (\rho_1.f \to \rho' + \Pi_f) -\!\!+ 0.\mathrm{Owned} \; + \\
&\neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'
\end{aligned}$$

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

$$\boxed{\begin{array}{l} \exists \rho' \cdot (\rho_1.f \to \rho' + \Pi_f) + \\ \exists \rho' \cdot (\rho_1.f \to \rho' + \Pi_f) \multimap 0.\mathrm{Owned} + \\ \neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1' \end{array}}$$

P-Write

P-Write

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \boxed{\rho_1.f \rightarrow \rho'} + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho_2 + \Pi'}$$

$$\rho_1.f \rightarrow \rho_r + \Pi_r +$$
$$\exists \rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) \multimap 0.\mathrm{Owned} +$$
$$\neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'$$

P-Write

P-Write

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \boxed{\rho_1.f \to \rho_2} + \Pi'}$$

$$\rho_1.f \to \rho_2 + \Pi_\rho +$$
$$\exists \rho' \cdot (\rho_1.f \to \rho' + \Pi_f) \multimap 0.\mathrm{Owned} +$$
$$\neg(\rho_1 = 0) + p_1(\rho_1) + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi'_1$$

P-Write

P-Write

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1 \,.\, f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

$$\begin{aligned} &\rho_1.f \to \rho_2 + \Pi_\rho + \\ &\exists \rho' \cdot (\rho_1.f \to \rho' + \Pi_f) \multimap 0.\mathrm{Owned} + \\ &\neg(\rho_1 = 0) + p_1(\rho_1) + \boxed{\neg(\rho_2 = 0) + p_2(\rho_2)} + \Pi_1' \end{aligned}$$

duplicate

P-Write

$$\textsc{P-Write}\quad \frac{\begin{array}{c}\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \\ \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho' + \Pi'\end{array}}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho_2 + \Pi'}$$

$$\begin{array}{l}\rho_1.f \rightarrow \rho_2 + \Pi_\rho + \boxed{\neg(\rho_2 = 0) + p_2(\rho_2) +} \\ \exists\rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) \mathbin{-\!\!+} 0.\mathrm{Owned} + \\ \neg(\rho_1 = 0) + p_1(\rho_1) + \boxed{\neg(\rho_2 = 0) + p_2(\rho_2)} + \Pi_1'\end{array}$$

duplicate

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

$$\begin{aligned}
&\rho_1.f \to \rho_2 + \Pi_\rho + \neg(\rho_2 = 0) + p_2(\rho_2) + \\
&\exists \rho' \cdot (\rho_1.f \to \rho' + \Pi_f) -\!\!\!+ 0.\mathrm{Owned} + \\
&\boxed{\neg(\rho_1 = 0) + p_1(\rho_1)} + \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'
\end{aligned}$$

discarded

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho_2 + \Pi'}$$

$$\boxed{\rho_1.f \to \rho_2 + \Pi_\rho + \neg(\rho_2 = 0) + p_2(\rho_2) +}$$
$$\exists\rho' \cdot (\rho_1.f \to \rho' + \Pi_f) \multimap 0.\mathrm{Owned} +$$
$$\neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'$$

pack

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho_2 + \Pi'}$$

$$\begin{aligned}
&\exists\rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) + \\
&\exists\rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) \multimap 0.\mathrm{Owned} + \\
&\qquad\qquad \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'
\end{aligned}$$

P-Write

P-WRITE

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho_2 + \Pi'}$$

$$\boxed{\begin{array}{l} \exists\rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) + \\ \exists\rho' \cdot (\rho_1.f \rightarrow \rho' + \Pi_f) \multimap 0.\mathrm{Owned} \end{array}} + \\ \neg(\rho_2 = 0) + p_2(\rho_2) + \Pi_1'$$

linear *modus ponens*

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f\text{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \rightarrow \rho_2 + \Pi'}$$

$$0.\mathrm{Owned} \rightarrow 0 +$$
$$\neg(\rho_2 = 0) + p_2(\rho_2) + \Pi'_1$$

P-Write

P-WRITE

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

$$\boxed{0.\mathrm{Owned} \to 0} +$$
$$\neg(\rho_2 = 0) + p_2(\rho_2) + \boxed{\Pi_1'}$$

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho_2 + \Pi'}$$

$$\boxed{= \Pi_1}$$

$$\boxed{0.\mathrm{Owned} \to 0} + \neg(\rho_2 = 0) + p_2(\rho_2) + \boxed{\Pi_1'}$$

P-Write

$$\textsc{P-Write}\quad \frac{\Delta_1;\Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2;\Pi_2 \qquad \Delta_2;\Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho' + \Pi'}{\Delta_1;\Pi_1 \vdash_\omega e_1\,.\,f\text{=}e_2 \Downarrow \rho_2 \dashv \Delta';\rho_1.f \to \rho_2 + \Pi'}$$

$$\neg(\rho_2 = 0) + p_2(\rho_2) + \boxed{\Pi_1}$$

P-Write

$$\textsc{P-Write}\quad \frac{\begin{array}{c}\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \\ \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'\end{array}}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

$$\textsc{NN-Write}\quad \frac{\begin{array}{c}C; M; E \vdash e_1 : c_1^- \\ C; M; E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2\end{array}}{C; M; E \vdash e_1\,.\,f{=}e_2 : c_2}$$

$$\neg(\rho_2 = 0) + p_2(\rho_2) + \boxed{\Pi_1}$$

P-Write

P-WRITE

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho' + \Pi'}{\Delta_1; \Pi_1 \vdash_\omega e_1\,.\,f{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \to \rho_2 + \Pi'}$$

NN-WRITE

$$\frac{C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C; M; E \vdash e_1\,.\,f{=}e_2 : \boxed{c_2}}$$

$$\boxed{\neg(\rho_2 = 0) + p_2(\rho_2)} + \boxed{\Pi_1}$$

P-Write

P-WRITE

$$\frac{\Delta_1; \Pi_1 \vdash_\omega e_1 \Downarrow \rho_1 \dashv \Delta_2; \Pi_2 \qquad \Delta_2; \Pi_2 \vdash_\omega e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho' + \Pi'}{\Delta_1; \boxed{\Pi_1} \vdash_\omega e_1\,.\,f\text{=}e_2 \Downarrow \rho_2 \dashv \Delta'; \rho_1.f \rightarrow \rho_2 + \Pi'}$$

NN-WRITE

$$\frac{C; M; E \vdash e_1 : c_1^- \qquad C; M; E \vdash e_2 : c_2 \qquad C(c_1)(f) = c_2}{C; M; E \vdash e_1\,.\,f\text{=}e_2 : \boxed{c_2}}$$

$$\boxed{\neg(\rho_2 = 0) + p_2(\rho_2)} + \boxed{\Pi_1}$$

Soundness

For every program g in kernel language, if g can be type checked under the non-null system, with consistent environments C and M, then with converted program type ω , g can also be type checked under the permission system.

How to Prove?

- *Piggy-back* approach:
  1. Define type system;
  2. Convert classes and methods;
  3. Prove the lemma and theorem.
- All checked in Twelf.

Conclusion

1. Fractional permissions with nesting is sound for a Java-like kernel language;
2. Fractional permissions can serve as a basis to prove soundness of other type systems;
3. Machine-checked proof has its benefits.

Thanks!