Permissions Accounting
CPL, May 30th 2011

John Boyland: **Checking Interference with Fractional Permissions**. SAS 2003.

Richard Bornat, Cristiano Calcagno, Peter W. O'Hearn, Matthew J. Parkinson: **Permission accounting in separation logic**. POPL 2005.

Permission and Ownership

- Separation logic uses heap predicates
  - E.g., {**emp**}, {E --> E'}, etc.
- View heap predicates as describing **ownership**
- View ownership as **permission:**
  - To **read**
  - To **write**
  - To **dispose** (i.e., to release memory)

Ownership Transfer

O'Hearn: Separation logic ("Resources, concurrency and local reasoning", CONCUR'04])

- Transfer ownership of heap cells in and out of shared resources
- Slogan: "permission rather than prohibition"
- permission = **exclusive ownership**, no room for **passivity**

Passivity

- Means:
  - Permission to **read**,
  - but no permission to **write** (or **dispose**)
- Giving out **read** permissions is “easy”
- Challenge: gathering them back up
- (How do we know if we have them all?)
- **Write**/**dispose** require exclusive permission

Fractional Permissions

- Boyland's "**Checking interference with fractional permissions**" (SAS 2003)
- Rational number z models permission
- Total control, $z = 1$ (**Read, write**, **dispose)**
- Shared access, $z < 1$ (**Read**-only)
- Permission is linear
  - **New** creates ($z=1$), **dispose** destroys ($z=1$)

Fractional Permissions

$$x \underset{z}{\mapsto} E \implies 0 < z \leq 1$$

$$x \underset{z}{\mapsto} E \star x \underset{z'}{\mapsto} E \iff x \underset{z+z'}{\longmapsto} E \wedge z > 0 \wedge z' > 0$$

Review: Separation Logic

$$\frac{\{Q\}C\{R\}}{\{P \star Q\}C\{P \star R\}} \; (\mathit{modifies}\ C\ \cap \mathit{vars}\ P = \emptyset)$$

$$\{\mathbf{emp}\}\ x := \mathrm{new}()\ \{x \mapsto \_\}$$
$$\{E \mapsto \_\}\ \ \mathrm{dispose}\, E\ \ \{\mathbf{emp}\}$$

$$\{R^x_E\}\quad x{:=}E\quad \{R\}$$
$$\{E' \mapsto \_\}\ [E']{:=}E\quad \{E' \mapsto E\}$$
$$\{E' \mapsto E\}\quad x{:=}[E']\ \{E' \mapsto E \wedge x = E\}$$

Review: Concurrent Sep Logic

$$\frac{\{Q_1\}\, C_1\, \{R_1\} \cdots \{Q_n\}\, C_n\, \{R_n\}}{\{Q_1 \star \cdots \star Q_n\}(C_1 \parallel \cdots \parallel C_n)\{R_1 \star \cdots \star R_n\}}$$

Fractional Permissions

$$\{\mathbf{emp}\}\ x := \mathrm{new}()\ \{x \overset{1}{\mapsto} \_\}$$

$$\{E \overset{1}{\mapsto} \_\}\ \ \mathrm{dispose}\, E\ \ \{\mathbf{emp}\}$$

$$\{R^x_E\}\quad x{:=}E\quad \{R\}$$

$$\{x \overset{1}{\mapsto} \_\}\ [x]{:=}E\quad \{x \overset{1}{\mapsto} E\}$$

$$\{E' \overset{z}{\mapsto} E\}\quad x{:=}[E']\ \{E' \overset{z}{\mapsto} E \wedge x = E'\}$$

Fractional Permissions Example

$$
\begin{array}{l}
\{\mathbf{emp}\} \\
x := \mathrm{new}(); \\
\{x \overset{1}{\mapsto} \_\} \\
[x] := 7; \\
\{x \overset{1}{\mapsto} 7\} \therefore \{x \overset{0.5}{\longmapsto} 7 \star x \overset{0.5}{\longmapsto} 7\} \\
\left(\begin{array}{l||l}
\{x \overset{0.5}{\longmapsto} 7\} & \{x \overset{0.5}{\longmapsto} 7\} \\
y := [x] - 1 & z := [x] + 1 \\
\{x \overset{0.5}{\longmapsto} 7 \wedge y = 6\} & \{x \overset{0.5}{\longmapsto} 7 \wedge z = 8\}
\end{array}\right); \\
\{x \overset{0.5}{\longmapsto} 7 \star x \overset{0.5}{\longmapsto} 7 \wedge y = 6 \wedge z = 8\} \therefore \\
\{x \overset{1}{\mapsto} 7 \wedge y = 6 \wedge z = 8\} \\
\mathrm{dispose}\, x; \\
\{\mathbf{emp} \wedge y = 6 \wedge z = 8\}
\end{array}
$$

Boyland's Result

- Not yet in context of separation logic
- Rather: **type system** for fractional permissions
- Simple imperative parallel language:
  - Parallel statements
  - Aliasing of memory
- **Soundness/Determinacy:** Programs that type-check do not exhibit interference: execution leads to deterministic results.

Utility of Fractional Permissions

- Good for
  - Symmetrical splitting of resources
  - Indefinite subdivision
  - "Predictable" split/combine behavior

- Examples:
  - lambda-term substitution (section 9.1 of PASL)
  - (no large examples given by Boyland)

Limitations of Fractional Permissions

- Permissions not always divided/recombined
- Sometimes, they are **counted**

Counting Permission Problem:

- Permissions given out at one point
  - By one thread, by one subroutine, etc
- Permissions gathered back at another
- "give-out" and "gather-back" orders may differ

Readers and Writers Example

READERS

```
P(m);
   count := count + 1;
   if count = 1 then P(write);
V(m);

   ... reading happens here ...;

P(m);
   count := count − 1;
   if count = 0 then V(write);
V(m)
```

WRITER

```
P(write);

   ... writing happens here ...

V(write)
```

The Concurrent Components

- the four uses of the binary mutex m;
- the reader prologue $count := count + 1$...;
- the reader action section;
- the reader epilogue $count := count - 1$...;
- the two uses of the binary mutex $write$;
- the writer action section.

Readers and Writers Example

- Questions
  - How are these concurrent components controlled?
  - How is the `count` variable restricted to reader prologue and epilogue?
- (Partial) answers to come:
  - Uses **permission accounting** with CCRs (conditional critical regions)

Counting Permissions

- Bornat, Calcagno, O'Hearn, Parkinson: "**Permission Accounting in Separation Logic**"
- A natural *n* >= 0 counts "split-off" parts
- Source permission, no readers: n = 0
- Source permission, k split-off readers: n=k
- Reader permission: n = -1

Counting Permissions

$$E \overset{n}{\mapsto} E' \rightarrow n \geq 0$$

$$E \overset{n}{\mapsto} E' \wedge n \geq 0 \iff E \overset{n+1}{\longmapsto} E' \star E \rightarrowtail E'$$

$$\begin{array}{lll}
\{\mathbf{emp}\} & x{:=}\mathrm{new}(E) & \{x \overset{0}{\mapsto} E\} \\
\{E' \overset{0}{\mapsto} _\} & \mathrm{dispose}\, E' & \{\mathbf{emp}\} \\
\{R^x_E\} & x{:=}E & \{R\} \\
\{E' \overset{0}{\mapsto} _\} & [E']{:=}E & \{E' \overset{0}{\mapsto} E\} \\
\{E' \rightarrowtail E\} & x{:=}[E'] & \{E' \rightarrowtail E \wedge x = E\}
\end{array}$$

Review:
CCR = Conditional Critical Region

with b when G do C od

1. acquire bundle b;
2. evaluate the boolean guard G;
3. if G is true, execute the command C and release b;
4. if G is false, release b and try again.

Review: CCR Rule

$$\frac{\{(Q \star I_b) \wedge G\} C \{R \star I_b\}}{\{Q\} \text{with } b \text{ when } G \text{ do } C \text{ od} \{R\}}$$

Non-interference Side Condition:
Processes cannot refer to variables of a bundle outside of a conditional critical region

Review:
Mutex as a Resource Bundle *m*

Following [15], a mutex semaphore m is a bundle whose CCRs are either

P: with m when $m \neq 0$ do $m := 0$ od, or

V: with m when true do $m := 1$ od.

- P **waits** for resource
- V **signals** that resource is free

Readers & Writers (Original Version)

READERS

```
P(m);
   count := count + 1;
   if count = 1 then P(write);
V(m);

   ... reading happens here ...;

P(m);
   count := count − 1;
   if count = 0 then V(write);
V(m)
```

WRITER

```
P(write);

   ... writing happens here ...

V(write)
```

Readers & Writers (CCR Version)

READERS

with *read* when true do
  if *count* = 0 then P(*write*) else skip fi;
  count +:= 1
od;

  ... reading happens here ...;

with *read* when *count* > 0 do
  count −:= 1;
  if *count* = 0 then V(*write*) else skip fi
od

WRITER

P(*write*);

  ... writing happens here ...

V(*write*)

$$write\text{: if } write = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{0}{\longmapsto} \_ \text{ fi}$$
$$read\text{: if } count = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{count}{\longmapsto} \_ \text{ fi}$$

Counting Permissions: Example

$$\{\mathbf{emp}\}$$

$$\mathrm{P}(write):\ \left(\begin{array}{l}\{(\mathbf{emp} \star \text{if } write = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{0}{\mapsto} \_ \text{ fi}) \wedge write = 1\}\ \therefore \\ \{(\mathbf{emp} \star y \overset{0}{\mapsto} \_) \wedge write = 1\} \\ \quad write := 0 \\ \{y \overset{0}{\mapsto} \_ \star (\mathbf{emp} \wedge write = 0)\}\ \therefore \\ \{y \overset{0}{\mapsto} \_ \star (\text{if } write = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{0}{\mapsto} \_ \text{ fi} \wedge write = 0)\}\end{array}\right)$$

$$\{y \overset{0}{\mapsto} \_\}$$

P(write) waits until it is **okay to write**.

Counting Permissions: Example

$$\{y \overset{0}{\mapsto} \_\}$$

$$\mathrm{V}(write): \begin{pmatrix} \{y \overset{0}{\mapsto} \_ \star \text{if } write = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{0}{\mapsto} \_ \text{ fi}\} \ \therefore \\ \{y \overset{0}{\mapsto} \_ \star (\mathbf{emp} \wedge write = 0)\} \\ \quad write := 1 \\ \{\mathbf{emp} \star (y \overset{0}{\mapsto} \_ \wedge write = 1)\} \ \therefore \\ \{\mathbf{emp} \star (\text{if } write = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{0}{\mapsto} \_ \text{ fi} \wedge write = 1)\} \end{pmatrix}$$

$$\{\mathbf{emp}\}$$

V(write) signals that it is **okay to read**.

Reader Prologue

$$
\begin{array}{l}
\{\mathbf{emp}\} \\
\quad \text{with } \mathit{read} \text{ when true do} \\
\qquad \{\text{if } \mathit{count} = 0 \text{ then } \mathbf{emp} \text{ else } y \overset{\mathit{count}}{\longmapsto} \_ \text{ fi} \star \mathbf{emp}\} \\
\qquad\quad \text{if } \mathit{count} = 0 \text{ then } \quad \{\mathbf{emp}\}\ \mathrm{P}(\mathit{write})\ \{y \overset{0}{\longmapsto} \_\} \\
\qquad\qquad\qquad\qquad \text{else } \{y \overset{\mathit{count}}{\longmapsto} \_\} \quad \text{skip} \quad \{y \overset{\mathit{count}}{\longmapsto} \_\} \\
\qquad\quad \text{fi} \\
\qquad \{y \overset{\mathit{count}}{\longmapsto} \_\} \\
\qquad\quad \mathit{count} +:= 1 \\
\qquad \{y \overset{\mathit{count}-1}{\longmapsto} \_\} \therefore \{y \overset{\mathit{count}}{\longmapsto} \_ \star z \rightarrowtail \_\} \\
\quad \text{od} \\
\{z \rightarrowtail N\}
\end{array}
$$

Reader Epilogue

$\{z \rightarrowtail N\}$

with *read* when $count > 0$ do

$\{\text{if } count = 0 \text{ then } \mathbf{emp} \text{ else } y \xmapsto{count} \_ \text{ fi} \star z \rightarrowtail N \wedge count > 0\}$

$count -:= 1$

$\{\text{if } count + 1 = 0 \text{ then } \mathbf{emp} \text{ else } y \xmapsto{count+1} \_ \text{ fi} \star z \rightarrowtail N \wedge count + 1 > 0\} \ \therefore$

$\{y \xmapsto{count+1} \_ \star z \rightarrowtail N \wedge count \geq 0\} \therefore \{y \xmapsto{count} \_ \wedge count \geq 0\}$

if $count = 0$ then $\{y \xmapsto{0} \_\}$ V(*write*) $\{\mathbf{emp}\}$

else $\{y \xmapsto{count} \_\}$ skip $\{y \xmapsto{count} \_\}$

fi

$\{\text{if } count = 0 \text{ then } \mathbf{emp} \text{ else } y \xmapsto{count} \_ \text{ fi} \star \mathbf{emp}\}$

od

$\{\mathbf{emp}\}$

Combining Fractional and Counting

- Fractional and counting are **both useful**
- So, combine and use together.

Combined Model – Rational q's

$$q \star_3 q' = \begin{cases} \text{undefined} & \text{if } q \geq 0 \text{ and } q' \geq 0 \\ \text{undefined} & \text{if } (q \geq 0 \text{ or } q' \geq 0) \text{ and } q + q' < 0 \\ q + q' & \text{otherwise} \end{cases}$$

Counting

$$E \overset{q}{\longmapsto} E' \iff E \overset{q+q'}{\longmapsto} E' \star E \overset{-q'}{\longmapsto} E' \quad \text{when } q \geq 0 \text{ and } q' > 0$$

Fractional

$$E \overset{-(q+q')}{\longmapsto} E' \iff E \overset{-q}{\longmapsto} E' \star E \overset{-q'}{\longmapsto} E' \quad \text{when } q, q' > 0$$

Combined Model: Axioms

$$\{R^x_E\} \quad x{:=}E \quad \{R\}$$

$$\{E' \overset{m_W}{\longmapsto} \_\} \; [E']{:=}E \; \{E' \overset{m_W}{\longmapsto} E\}$$

$$\{E' \overset{m}{\longmapsto} E\} \quad x{:=}[E'] \quad \{E' \overset{m}{\longmapsto} E \wedge x = E\}$$

$$\{\mathbf{emp}\} \quad x{:=}\mathrm{new}(E) \quad \{x \overset{m_W}{\longmapsto} E\}$$

$$\{E' \overset{m_W}{\longmapsto} \_\} \quad \mathrm{dispose}\, E' \quad \{\mathbf{emp}\}$$

m_W is the (unique) write permission (total perm.)

PASL: Future Work

- Inductive definitions
  - Sometimes **DAGs allowed** when we **want trees**
  - Does "Fresh look at separation algebras and share accounting" (Robert Dockins et al) address this?
- Variables as resources
  - "Hoare logic's variable assignment rule finesses the distinction between program variables and logic variables and assumes an absence of program-variable aliasing"
  - In Separation Logic: the **heap is localized** via frame rule, but the **stack is global**!

Conclusions

- Ownership versus Permission
- Problem: SL lacks support for passivity
- **Fractional Permissions**
  - Good for indefinite divisibility of permission
  - Bad when divide/recombine pattern not obvious
- **Counting Permissions**
  - Good for Readers & Writers example
  - No way to subdivide existing read permission
- **Fractional + Counting can be combined**