ER04: Separation logics and applications

Frame Rule, Local Reasoning and Information Hiding

Hongseok Yang
Queen Mary University of London

Two programming disciplines

- Locality: One function accesses only a few data structures.
- Information hiding: The internal resource of a module is hidden from its client programs.
- Exploited by the frame rule and the hypothetical frame rule in separation logic.

Part 1: Frame rule and local reasoning

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$$

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$$

Prove a triple under an assumption on functions.

Proving about procedure calls

$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$

Assumed spec for f().

Prove a triple under an assumption on functions.

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y{=}f()\{Q[y/ret]\}$$

Assumed spec for f().

Instantiation of the spec for the caller.

Prove a triple under an assumption on functions.

Proving about procedure calls

$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();

x=create_list();

y=create_list();

z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

{P}f(){Q} ⊦ {P}y=f(){Q[y/ret]}

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();


y=create_list();


z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y{=}f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

Preconditions do not match.

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();


y=create_list();


z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();


z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y{=}f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

Preconditions do not match again!

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();


z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y=f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();
```

{ls (w,0) * ls (x,0) * ls(y,0)}

```
z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y{=}f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

Yes, the same problem. Third times!

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();
```

{ls (w,0) * ls (x,0) * ls(y,0)}

```
z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Proving about procedure calls

$$\{P\}f()\{Q\} \vdash \{P\}y{=}f()\{Q[y/ret]\}$$

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

Yes, the same problem Third times!

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();
```

{ls (w,0) ls (x,0) ls(y,0) ls(z,0)}

1) Need 4 Hoare triples for `create_list`.
2) However, one triple should be enough.
3) Additional overhead in formal verification.

Expected outcome of part I

- Understand the frame rule and local reasoning.
- Should be able to use them to remove the overhead seen in the previous slide.

Language with simple procedures

$$\begin{array}{lcl} \text{ret} & \in & \text{Vars} \\ C & ::= & \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C \end{array}$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

Language with simple procedures

$$\text{ret} \in \text{Vars}$$

$$C ::= \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
 $\texttt{local}\ t_1, t_2.$
 $t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
 $t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
 $\text{ret}{=}t_2$
$\texttt{in}$
 $x{=}\texttt{alloc2}();$
 $y{=}\texttt{alloc2}()$

Language with simple procedures

$$\text{ret} \quad \in \quad \text{Vars}$$

$$C \quad ::= \quad \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
  $\texttt{local}\ t_1, t_2.$
  $t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
  $t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
  $\text{ret}{=}t_2$
$\texttt{in}$
  $x{=}\texttt{alloc2}();$
  $y{=}\texttt{alloc2}()$

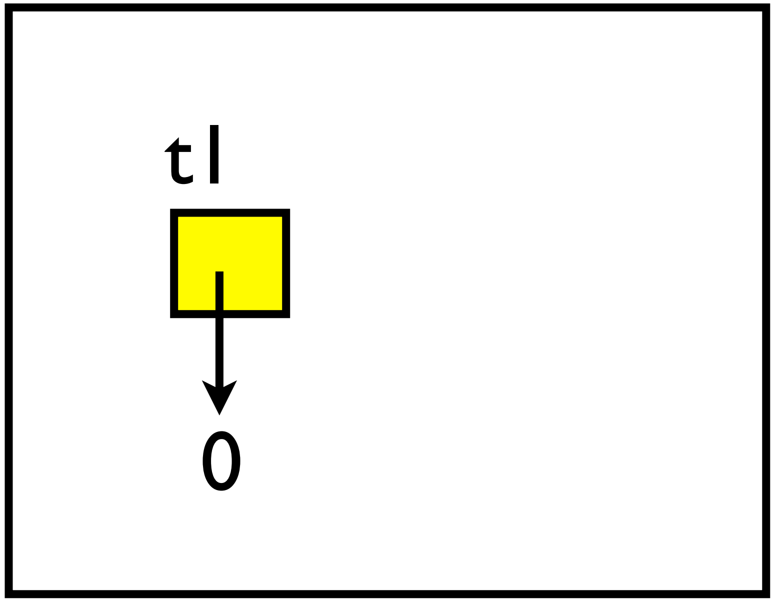
Language with simple procedures

$$\text{ret} \in \text{Vars}$$

$$C ::= \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
  $\texttt{local}\ t_1, t_2.$
  $t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
  $t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
  $\text{ret}{=}t_2$
$\texttt{in}$
  $x{=}\texttt{alloc2}();$
  $y{=}\texttt{alloc2}()$



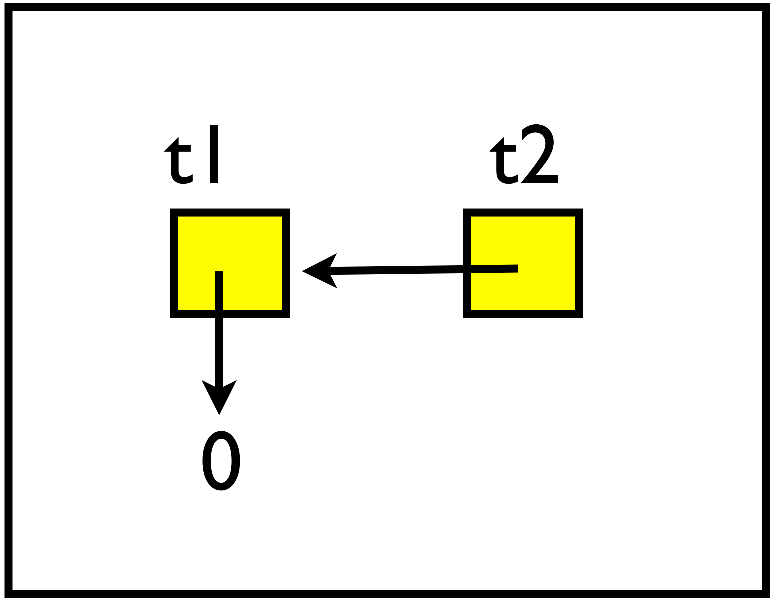
Language with simple procedures

$$\text{ret} \quad \in \quad \text{Vars}$$

$$C \quad ::= \quad \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
$\quad \texttt{local}\ t_1, t_2.$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad \text{ret}{=}t_2$
$\texttt{in}$
$\quad x{=}\texttt{alloc2}();$
$\quad y{=}\texttt{alloc2}()$



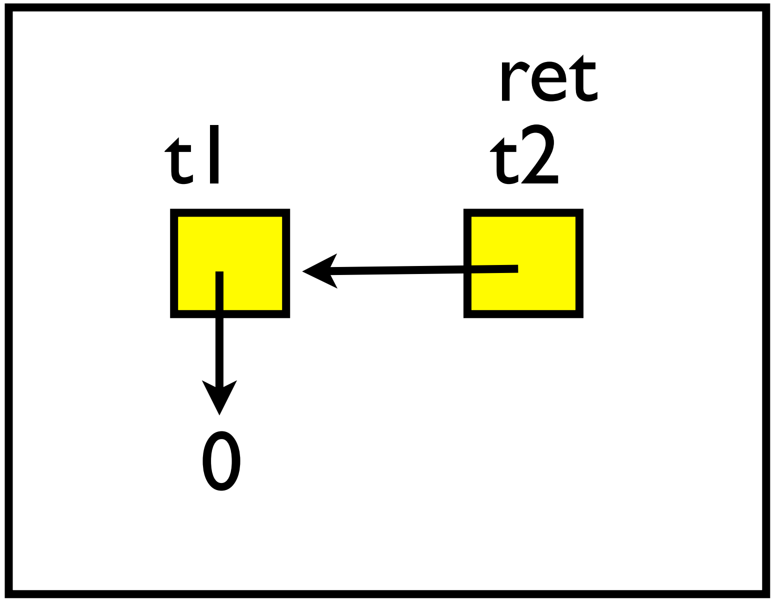
Language with simple procedures

$$\mathsf{ret} \in \mathsf{Vars}$$

$$C ::= \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
$\quad \texttt{local}\ t_1, t_2.$
$\quad t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
$\quad t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
$\quad \mathsf{ret}{=}t_2$
$\texttt{in}$
$\quad x{=}\texttt{alloc2}();$
$\quad y{=}\texttt{alloc2}()$



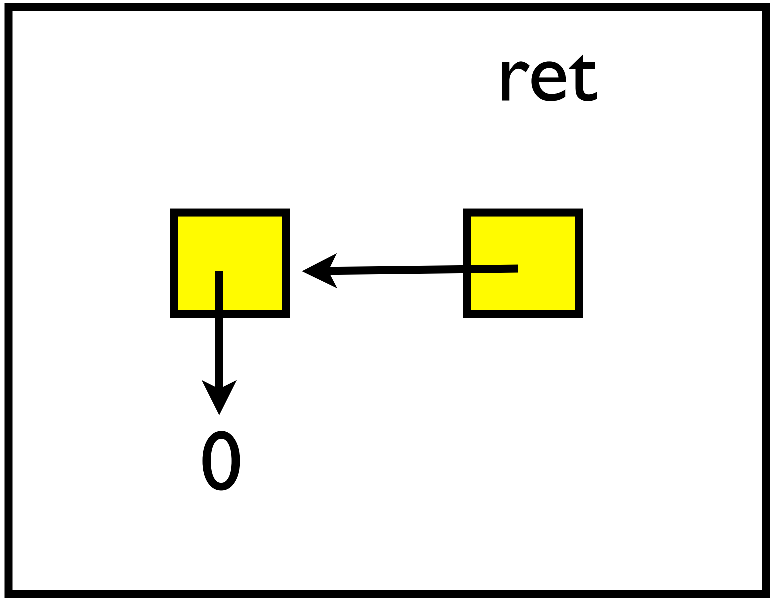
Language with simple procedures

$$\text{ret} \in \text{Vars}$$

$$C ::= \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
  $\texttt{local}\ t_1, t_2.$
  $t_1{=}\texttt{new}(1);\ *t_1{=}0;$
  $t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
  $\text{ret}{=}t_2$
$\texttt{in}$
  $x{=}\texttt{alloc2}();$
  $y{=}\texttt{alloc2}()$



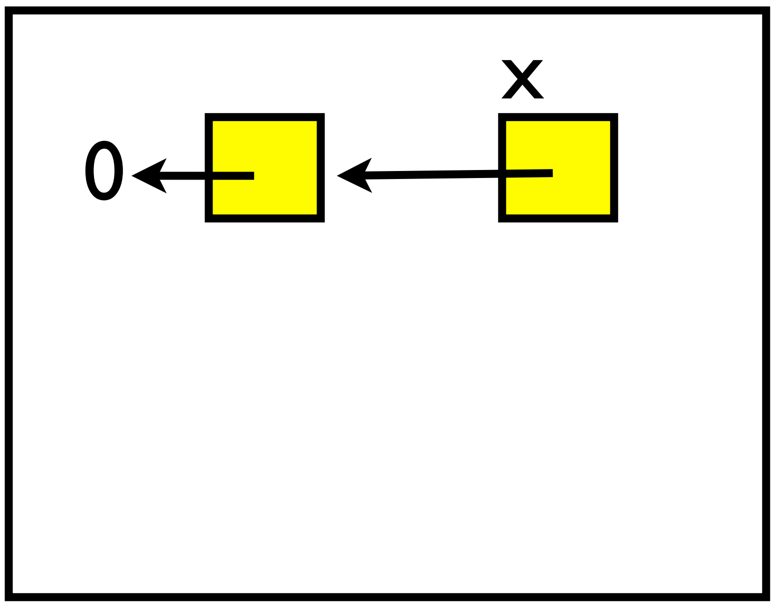
Language with simple procedures

$$\text{ret} \quad \in \quad \text{Vars}$$

$$C \quad ::= \quad \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
  $\texttt{local}\ t_1, t_2.$
  $t_1{=}\texttt{new}(1);\ *t_1{=}0;$
  $t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
  $\text{ret}{=}t_2$
$\texttt{in}$
  $x{=}\texttt{alloc2}();$
  $y{=}\texttt{alloc2}()$



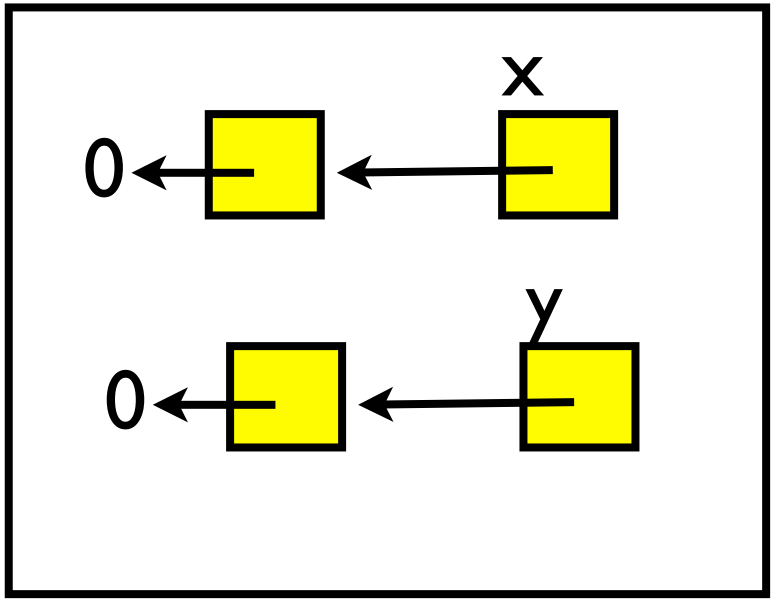
Language with simple procedures

$$\text{ret} \quad \in \quad \text{Vars}$$

$$C \quad ::= \quad \ldots \mid \texttt{local}\ x.C \mid x{=}f(\vec{E}) \mid \texttt{let}\ f(\vec{x}){=}C\ \texttt{in}\ C$$

- Call-by-value procedures as in Java.
- Returning a value is done by the assignment to ret.
- No global variables are modified by functions.

$\texttt{let alloc2}() =$
  $\texttt{local}\ t_1, t_2.$
  $t_1{=}\texttt{new}(1);\ *t_1{=}0;$
  $t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
  $\text{ret}{=}t_2$
$\texttt{in}$
  $x{=}\texttt{alloc2}();$
  $y{=}\texttt{alloc2}()$



Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\; f(x){=}C\; \texttt{in}\; D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\},\ldots,\{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$

```
let alloc2() =
  local t1, t2.
  t1=new(1); *t1=0;
  t2=new(1); *t2=t1;
  ret=t2
in
  x=alloc2();
  y=alloc2()
```

$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\; f(x){=}C\; \texttt{in}\; D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
let alloc2() =
  $\{\mathsf{emp}\}$
  local t_1, t_2.
  t_1=new(1); $*t_1$=0;
  t_2=new(1); $*t_2$=t_1;
  ret=t_2
  $\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
in
  x=alloc2();
  y=alloc2()
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
  $\{\mathsf{emp}\}$
  $\texttt{local}\ t_1, t_2.$
  $\{\mathsf{emp}\}$
  $t_1{=}\texttt{new}(1);\ *t_1{=}0;$

  $t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$

  $\mathsf{ret}{=}t_2$

  $\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$

  $x{=}\texttt{alloc2}();$

  $y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
  $\{\mathsf{emp}\}$
  $\texttt{local}\ t_1, t_2.$
  $\{\mathsf{emp}\}$
  $t_1{=}\texttt{new}(1);\ *t_1{=}0;$
  $\{t_1{\mapsto}0\}$
  $t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$

  $\mathsf{ret}{=}t_2$

  $\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$

  $x{=}\texttt{alloc2}();$

  $y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\,x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\ t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad\{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\mathsf{ret}{=}t_2$
$\quad\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$
$\quad x{=}\texttt{alloc2}();$
$\quad y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\ t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad\{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad \mathsf{ret}{=}t_2$
$\quad\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$

$\quad\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$

$\quad x{=}\texttt{alloc2}();$

$\quad y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad \{\mathsf{emp}\}$
$\quad \texttt{local}\ t_1, t_2.$
$\quad \{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad \{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad \{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad \mathsf{ret}{=}t_2$
$\quad \{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad \{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad \{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$

$\quad x{=}\texttt{alloc2}();$

$\quad y{=}\texttt{alloc2}()$
$\{\exists x' y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P, Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
`let alloc2() =`
 $\{\mathsf{emp}\}$
 `local` t_1, t_2.
 $\{\mathsf{emp}\}$
 $t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
 $\{t_1{\mapsto}0\}$
 $t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
 $\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
 $\mathsf{ret}{=}t_2$
 $\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
 $\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
 $\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
`in`

 $x{=}\texttt{alloc2}();$

 $y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\ t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
$\quad\{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
$\quad\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\mathsf{ret}{=}t_2$
$\quad\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\exists t_1.\,\mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$

$\quad x{=}\texttt{alloc2}();$

$\quad y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
  $\{\mathsf{emp}\}$
  $\texttt{local}\ t_1, t_2.$
  $\{\mathsf{emp}\}$
  $t_1{=}\texttt{new}(1);\ {*t_1}{=}0;$
  $\{t_1{\mapsto}0\}$
  $t_2{=}\texttt{new}(1);\ {*t_2}{=}t_1;$
  $\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
  $\mathsf{ret}{=}t_2$
  $\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
  $\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
  $\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$
$\{\mathsf{emp}\}$
  $x{=}\texttt{alloc2}();$

  $y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\ x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\ t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ {*}t_1{=}0;$
$\quad\{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ {*}t_2{=}t_1;$
$\quad\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad \mathsf{ret}{=}t_2$
$\quad\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$
$\{\mathsf{emp}\}$
$\quad x{=}\texttt{alloc2}();$
$\{\exists x'.\, x{\mapsto}x' * x'{\mapsto}0\}$
$\quad y{=}\texttt{alloc2}()$
$\{\exists x'y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\ x.C\{Q\}}\ x \notin \mathsf{FV}(P,Q)$$

$$\frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\ f(x){=}C\ \texttt{in}\ D\{Q\}}\ x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\ t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad\{t_1{\mapsto}0\}$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\mathsf{ret}{=}t_2$
$\quad\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\quad\{\exists t_1.\,\mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$
$\{\mathsf{emp}\}$
$\quad x{=}\texttt{alloc2}();$
$\{\exists x'.\,x{\mapsto}x' * x'{\mapsto}0\}$
$\quad y{=}\texttt{alloc2}()$
$\{\exists x'y'.\,x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Cannot apply the rule!

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\; f(x){=}C\; \texttt{in}\; D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\quad\{\mathsf{emp}\}$
$\quad\texttt{local}\; t_1, t_2.$
$\quad\{\mathsf{emp}\}$
$\quad t_1{=}\texttt{new}(1);\ *t_1{=}0;$
$\quad\{t_1 {\mapsto} 0\}$
$\quad t_2{=}\texttt{new}(1);\ *t_2{=}t_1;$
$\quad\{t_2 {\mapsto} t_1 * t_1 {\mapsto} 0\}$
$\quad\mathsf{ret}{=}t_2$
$\quad\{\mathsf{ret}{=}t_2 \wedge t_2 {\mapsto} t_1 * t_1 {\mapsto} 0\}$
$\quad\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret} {\mapsto} t_1 * t_1 {\mapsto} 0\}$
$\quad\{\exists t_1.\, \mathsf{ret} {\mapsto} t_1 * t_1 {\mapsto} 0\}$
$\texttt{in}$
$\{\mathsf{emp}\}$
$\quad x{=}\texttt{alloc2}();$
$\{\exists x'.\, x {\mapsto} x' * x' {\mapsto} 0\}$
$\quad y{=}\texttt{alloc2}()$
$\{\exists x' y'.\, x {\mapsto} x' * x' {\mapsto} 0 * y {\mapsto} y' * y' {\mapsto} 0\}$

$\{\exists x'.\, x {\mapsto} x' * x' {\mapsto} 0\}$

$\{\exists x' t_1.\, x {\mapsto} x' * x' {\mapsto} 0 * \mathsf{ret} {\mapsto} t_1 * t_1 {\mapsto} 0\}$

Proof rules for local vars. and proc.

$$\frac{\Gamma \vdash \{P\}C\{Q\}}{\Gamma \vdash \{P\}\texttt{local}\, x.C\{Q\}}\; x \notin \mathsf{FV}(P,Q) \qquad \frac{}{\Gamma, \{P\}f(x)\{Q\} \vdash \{P[E/x]\}y{=}f(E)\{Q[y/\mathsf{ret}, E/x]\}}$$

$$\frac{\Gamma \vdash \{P_i\}C\{Q_i\} \quad \Gamma, \{P_0\}f(x)\{Q_0\}, \ldots, \{P_n\}f(x)\{Q_n\} \vdash \{P\}D\{Q\}}{\Gamma \vdash \{P\}\texttt{let}\; f(x){=}C\; \texttt{in}\; D\{Q\}}\; x \notin \mathsf{FV}(Q_i)$$

- E.g.

$\{\mathsf{emp}\}$
$\texttt{let alloc2}() =$
$\{\mathsf{emp}\}$ $\qquad\qquad \{\exists x'.\, x{\mapsto}x' * x'{\mapsto}0\}$
$\texttt{local}\; t_1, t_2.$
$\{\mathsf{emp}\}$
$t_1{=}\texttt{new}(1);\; {*t_1}{=}0;$
$\{t_1{\mapsto}0\}$
$t_2{=}\texttt{new}(1);\; {*t_2}{=}t_1;$
$\{t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\mathsf{ret}{=}t_2$
$\{\mathsf{ret}{=}t_2 \wedge t_2{\mapsto}t_1 * t_1{\mapsto}0\}$
$\{\mathsf{ret}{=}t_2 \wedge \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\{\exists t_1.\, \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$ $\qquad\qquad \{\exists x' t_1.\, x{\mapsto}x' * x'{\mapsto}0 * \mathsf{ret}{\mapsto}t_1 * t_1{\mapsto}0\}$
$\texttt{in}$
$\{\mathsf{emp}\}$
$x{=}\texttt{alloc2}();$
$\{\exists x'.\, x{\mapsto}x' * x'{\mapsto}0\}$
$y{=}\texttt{alloc2}()$
$\{\exists x' y'.\, x{\mapsto}x' * x'{\mapsto}0 * y{\mapsto}y' * y'{\mapsto}0\}$

Frame rule

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}}\ \mathsf{Modifies}(C) \cap \mathsf{FV}(R) = \emptyset$$

- Means that R can be added as an invariant.
- * and $\top$-avoiding triple take care of the heap access of C, and the side cond. takes care of the stack access.

Frame rule

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \quad \text{Modifies}(C) \cap \text{FV}(R) = \emptyset$$

- Means that R can be added as an invariant
- * and ⊤-a ... ess of C, and th ... ss.

$\{\mathsf{emp}\}$
let alloc2() =
  $\{\mathsf{emp}\}$
  local t_1, t_2.
  $\{\mathsf{emp}\}$
  t_1=new(1); $*t_1$=0;
  $\{t_1 \mapsto 0\}$
  t_2=new(1); $*t_2$=t_1;
  $\{t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  ret=t_2
  $\{\mathsf{ret}=t_2 \wedge t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\mathsf{ret}=t_2 \wedge \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\exists t_1 .\, \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
in
$\{\mathsf{emp}\}$
  x=alloc2();
$\{\exists x' .\, x \mapsto x' * x' \mapsto 0\}$
  y=alloc2()
$\{\exists x' y' .\, x \mapsto x' * x' \mapsto 0 * y \mapsto y' * y' \mapsto 0\}$

Frame rule

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \quad \text{Modifies}(C) \cap \text{FV}(R) = \emptyset$$

- Means that R can be added as an invariant
- * and ⊤-a… …ess of
  C, and th… …ss.

$\{\mathsf{emp}\}$
let alloc2() =
  $\{\mathsf{emp}\}$
  local t_1, t_2.
  $\{\mathsf{emp}\}$
  t_1=new(1); $*t_1$=0;
  $\{t_1 \mapsto 0\}$
  t_2=new(1); $*t_2$=t_1;
  $\{t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  ret=t_2
  $\{\mathsf{ret}=t_2 \wedge t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\mathsf{ret}=t_2 \wedge \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\exists t_1 .\, \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
in
$\{\mathsf{emp}\}$
  x=alloc2();
$\{\exists x' .\, x \mapsto x' * x' \mapsto 0\}$
  y=alloc2()
$\{\exists x' y' .\, x \mapsto x' * x' \mapsto 0 * y \mapsto y' * y' \mapsto 0\}$

$\{\mathsf{emp}\}$
  y=alloc2()
$\{\exists y' .\, y \mapsto y' * y' \mapsto 0\}$

Frame rule

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \quad \text{Modifies}(C) \cap \text{FV}(R) = \emptyset$$

- Means that R can be added as an invariant
- * and ⊤-a ... ess of C, and th ... ss.

$\{\text{emp}\}$
`let alloc2() =`
  $\{\text{emp}\}$
  `local` t_1, t_2.
  $\{\text{emp}\}$
  t_1=`new`(1); $*t_1$=0;
  $\{t_1 \mapsto 0\}$
  t_2=`new`(1); $*t_2$=t_1;
  $\{t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  ret=t_2
  $\{\text{ret}=t_2 \wedge t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\text{ret}=t_2 \wedge \text{ret} \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\exists t_1.\, \text{ret} \mapsto t_1 * t_1 \mapsto 0\}$
`in`
$\{\text{emp}\}$
  x=`alloc2()`;
$\{\exists x'.\, x \mapsto x' * x' \mapsto 0\}$
  y=`alloc2()`
$\{\exists x' y'.\, x \mapsto x' * x' \mapsto 0 * y \mapsto y' * y' \mapsto 0\}$

$\{\text{emp} * (\exists x'.\, x \mapsto x' * x' \mapsto 0)\}$
$\{\text{emp}\}$
  y=`alloc2()`
$\{\exists y'.\, y \mapsto y' * y' \mapsto 0\}$
$\{(\exists y'.\, y \mapsto y' * y' \mapsto 0) * (\exists x'.\, x \mapsto x' * x' \mapsto 0)\}$

Frame rule

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \quad \mathsf{Modifies}(C) \cap \mathsf{FV}(R) = \emptyset$$

- Means that R can be added as an invariant
- * and ⊤-a ... ess of C, and th ... ss.

$\{\mathsf{emp}\}$
let alloc2() $=$
  $\{\mathsf{emp}\}$
  local t_1, t_2.
  $\{\mathsf{emp}\}$
  t_1=new(1); $*t_1=0$;
  $\{t_1 \mapsto 0\}$
  t_2=new(1); $*t_2=t_1$;
  $\{t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  ret$=t_2$
  $\{\mathsf{ret}=t_2 \wedge t_2 \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\mathsf{ret}=t_2 \wedge \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
  $\{\exists t_1 .\, \mathsf{ret} \mapsto t_1 * t_1 \mapsto 0\}$
in
$\{\mathsf{emp}\}$
  x=alloc2();
$\{\exists x' .\, x \mapsto x' * x' \mapsto 0\}$
  y=alloc2()
$\{\exists x' y' .\, x \mapsto x' * x' \mapsto 0 * y \mapsto y' * y' \mapsto 0\}$

$\{\exists x' .\, x \mapsto x' * x' \mapsto 0\}$
$\{\mathsf{emp} * (\exists x' .\, x \mapsto x' * x' \mapsto 0)\}$
$\{\mathsf{emp}\}$
  y=alloc2()
$\{\exists y' .\, y \mapsto y' * y' \mapsto 0\}$
$\{(\exists y' .\, y \mapsto y' * y' \mapsto 0) * (\exists x' .\, x \mapsto x' * x' \mapsto 0)\}$
$\{\exists x' y' .\, x \mapsto x' * x' \mapsto 0 * y \mapsto y' * y' \mapsto 0\}$

Frame rule and local reasoning

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \ \ \mathsf{Modifies}(C) \cap \mathsf{FV}(R) = \emptyset$$

- Local reasoning means that the verification of a prog. fragment should focus on what's accessed by the frag.
- The frame rule supports local reasoning.
- This support of local reasoning is the reason that sep. logic has been successful.

Example again

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();


y=create_list();


z=create_list()
```

Example again

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();


z=create_list()
```

Example again

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();
```

{ls (w,0) * ls (x,0) * ls(y,0)}

```
z=create_list()
```

Example again

- Spec for create_list:

{emp}

```
create_list()
```

{ls(ret,0)}

- Verification of the caller program:

{emp}

```
w=create_list();
```

{ls(w,0)}

```
x=create_list();
```

{ls (w,0) * ls (x,0)}

```
y=create_list();
```

{ls (w,0) * ls (x,0) * ls(y,0)}

```
z=create_list()
```

{ls (w,0) * ls (x,0) * ls(y,0) * ls(z,0)}

Exercise 1

Use the frame rule and the following rule

$$\frac{}{\Gamma, \{P\} f(a) \{Q\} \ \vdash \ \{P[E/a]\} f(E) \{\exists \mathsf{ret}.\, Q[E/a]\}}$$

$$\frac{}{\Gamma, \{P\} f(a) \{Q\} \ \vdash \ \{P[E/a]\} y{=}f(E) \{Q[y/\mathsf{ret}, E/a]\}}$$

and prove the judgement below:

{ls(a,0)} rev(a) {ls(ret,0)}, {ls(a,0)} freeL(a) {emp}
⊢
{ls(x,0) * ls(y,0)} z=rev(x); freeL(y); freeL(z) {emp}

Exercise 2

The following rules are called small axioms.

$$\frac{}{\Gamma \vdash \{E \mapsto E_0\} {*}E{=}F \{E \mapsto F\}} \qquad \frac{}{\Gamma \vdash \{E \mapsto E_0\} \mathtt{free}(E) \{\mathsf{emp}\}}$$

$$\frac{}{\Gamma \vdash \{\mathsf{emp}\} x{=}\mathtt{new}(1) \{\exists x'.\, x \mapsto x'\}} \qquad \frac{}{\Gamma \vdash \{\mathsf{emp}\} x{=}E \{x{=}E \wedge \mathsf{emp}\}}\ x \notin \mathsf{FV}(E)$$

From these rules, derive the rules below:

$$\frac{}{\Gamma \vdash \{P * E \mapsto E_0\} {*}E{=}F \{P * E \mapsto F\}} \qquad \frac{}{\Gamma \vdash \{P * E \mapsto E_0\} \mathtt{free}(E) \{P * \mathsf{emp}\}}$$

$$\frac{}{\Gamma \vdash \{P\} x{=}\mathtt{new}(1) \{\exists x'.\, P * x \mapsto x'\}}\ x, x' \notin \mathsf{FV}(P) \qquad \frac{}{\Gamma \vdash \{P\} x{=}E \{x{=}E \wedge P\}}\ x \notin \mathsf{FV}(P, E)$$

Recall: Locality properties

$$\mathsf{LocalAction} \stackrel{def}{=} \mathsf{States} \rightarrow_{\mathsf{local}} \mathcal{P}(\mathsf{States})^{\top}$$

- Safety monotonicity: when $h_0 \# h_1$,

  $$\mathrm{safe}(c, (s, h_0)) \Longrightarrow \mathrm{safe}(c, (s, h_0 * h_1)).$$

- Frame property: when $\mathrm{safe}(c, (s, h_0))$ and $h_0 \# h_1$,

  $$(s', m) \in c(s, h_0 * h_1) \Longrightarrow (\exists m_0.\ (s', m_0) \in c(s, h_0) \wedge m = m_0 * h_1)$$

[Exercise 3] Prove the soundness of the frame rule using these two properties.

Part 2: Hypothetical frame rule and information hiding

Hypothetical frame rule

- Frame rule:

$$\frac{\Gamma \;\vdash\; \{A\}C\{B\}}{\Gamma \;\vdash\; \{A * R\}C\{B * R\}}\ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

- Hypothetical frame rule:

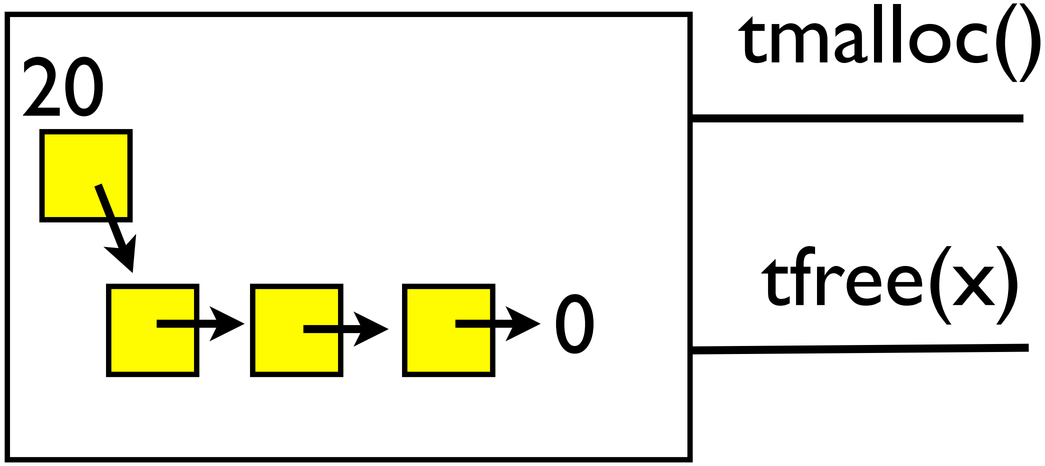
$$\frac{\Gamma, \{P\}f()\{Q\} \;\vdash\; \{A\}C\{B\}}{\Gamma, \{P * R\}f()\{Q * R\} \;\vdash\; \{A * R\}C\{B * R\}}\ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

- The hypo. frame rule allows us to exploit information hiding.

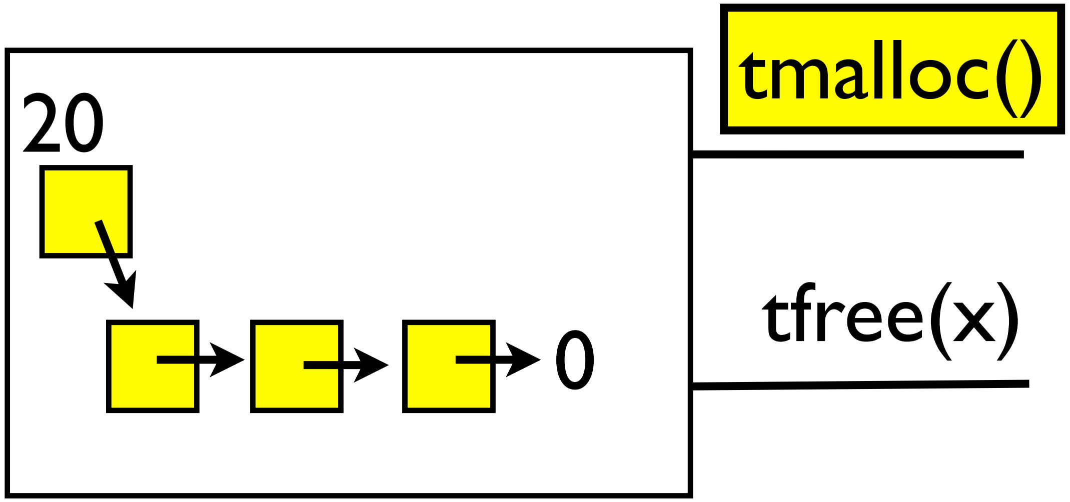
Expected outcome of part 2

- Understand the hypothetical frame rule and the modular procedure-call rule.
- Can use those rules to exploit information hiding in formal verification.

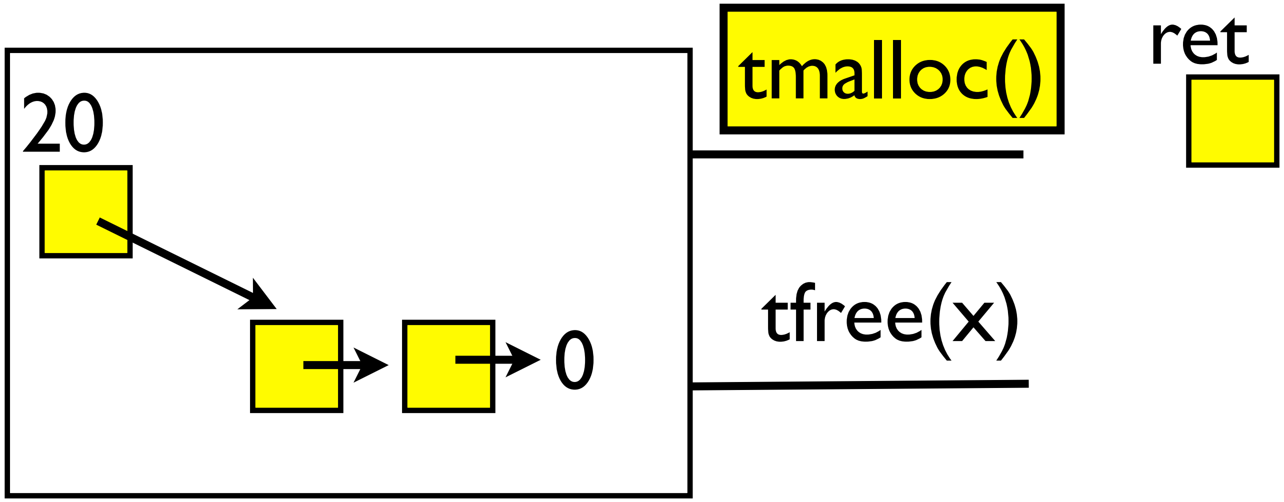
Toy memory manager



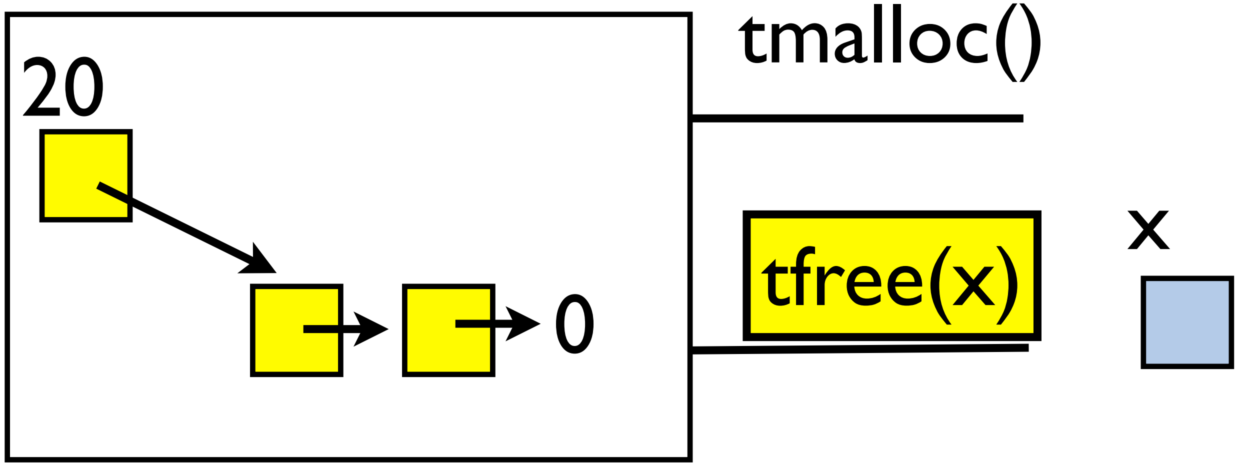
Toy memory manager



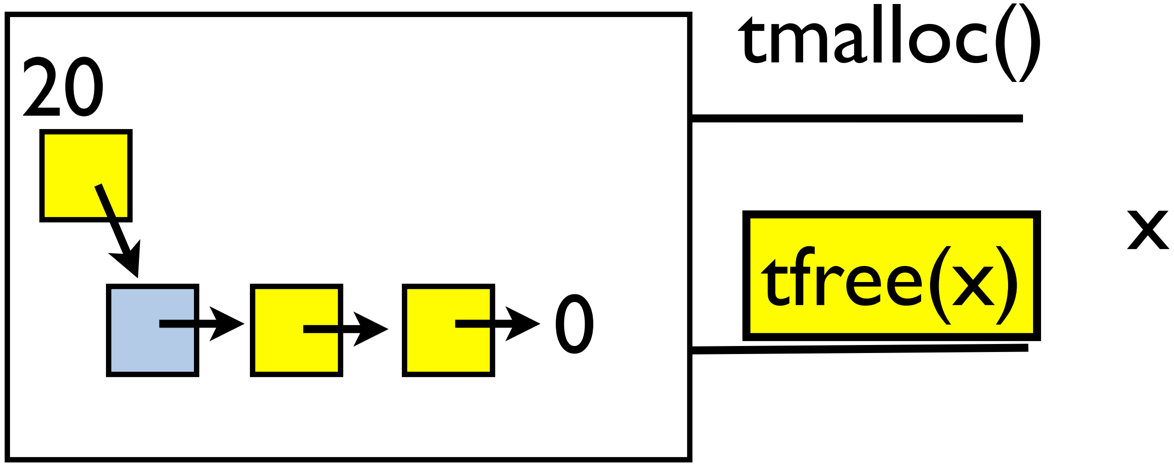
Toy memory manager



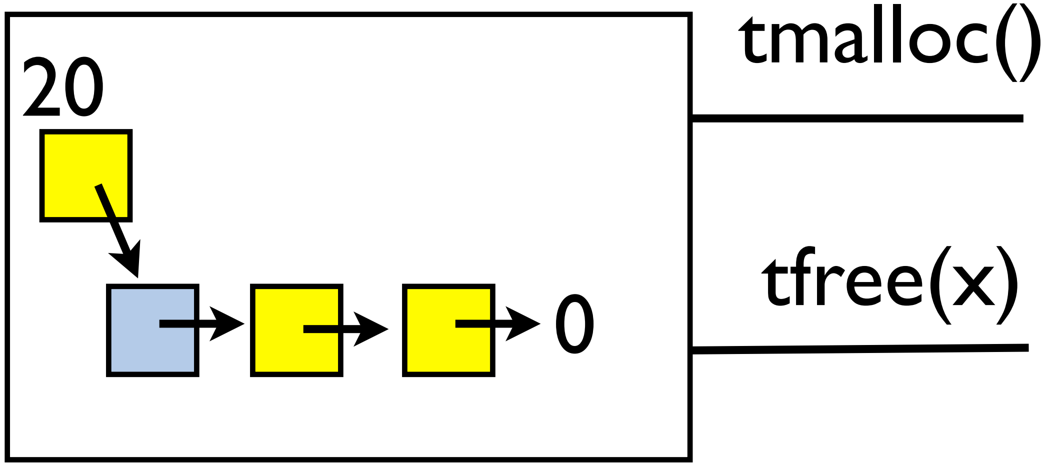
Toy memory manager



Toy memory manager



Toy memory manager



- Implementation

$$\texttt{tmalloc}() \;=\; \texttt{local}\, t.\; t{=}{*}20;\, \texttt{if}\; (t{=}0)\; \{\; \mathsf{ret}{=}\texttt{new}(1)\; \}\; \texttt{else}\; \{\; \mathsf{ret}{=}t;\, t{=}{*}\mathsf{ret};\, {*}20{=}t\; \}$$

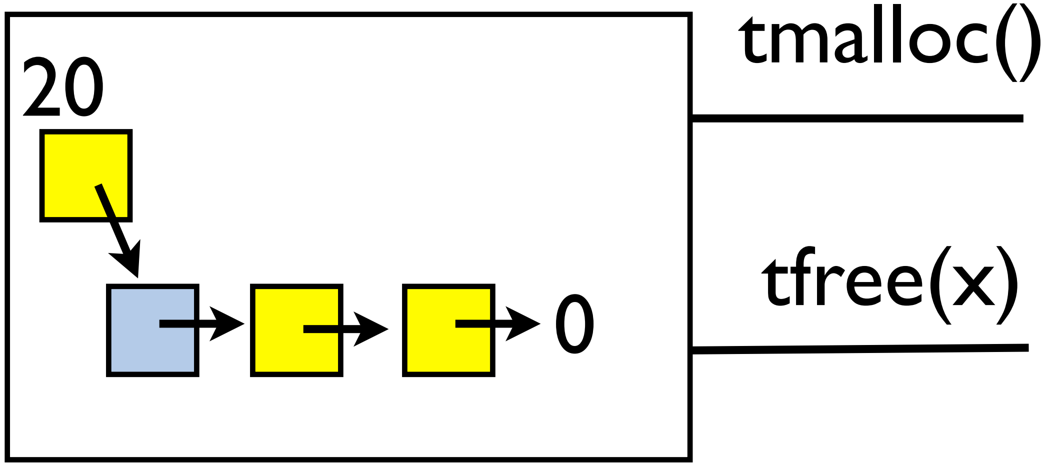
$$\texttt{tfree}(x) \;=\; \texttt{local}\, t.\; t{=}{*}20;\, {*}x{=}t;\, {*}20{=}x$$

- Specification

$$\{\mathsf{emp} * \exists a.\, 20{\mapsto}a * \mathsf{ls}(a,0)\}\texttt{tmalloc}()\{\mathsf{ret}{\mapsto}\_ * \exists a.\, 20{\mapsto}a * \mathsf{ls}(a,0)\}$$

$$\{x{\mapsto}\_ * \exists a.\, 20{\mapsto}a * \mathsf{ls}(a,0)\}\texttt{tfree}(x)\{\mathsf{emp} * \exists a.\, 20{\mapsto}a * \mathsf{ls}(a,0)\}$$

Toy memory manager



- Implementation

$$\texttt{tmalloc}() \;=\; \texttt{local}\, t.\ t{=}{*}20; \texttt{if}\ (t{=}0)\ \{\ \mathsf{ret}{=}\texttt{new}(1)\ \}\ \texttt{else}\ \{\ \mathsf{ret}{=}t; t{=}{*}\mathsf{ret}; {*}20{=}t\ \}$$

$$\texttt{tfree}(x) \;=\; \texttt{local}\, t.\ t{=}{*}20; {*}x{=}t; {*}20{=}x$$

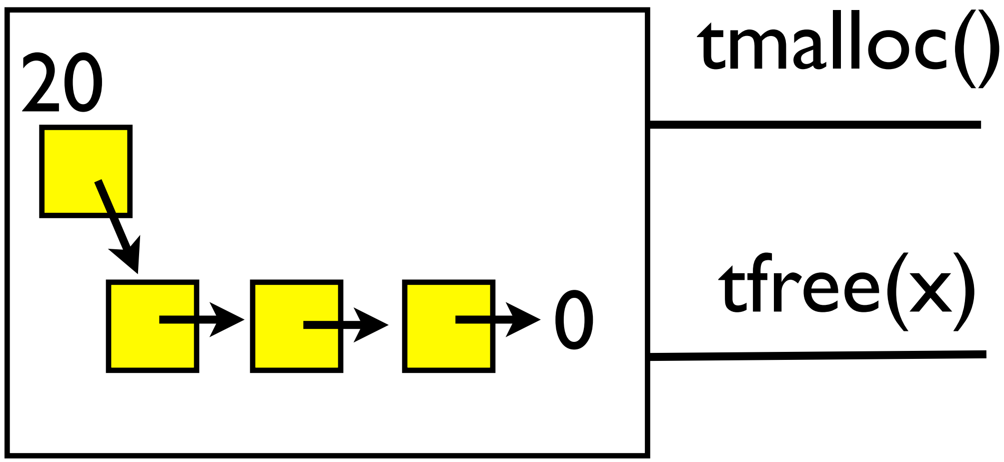
- Specification

- Resource inv: ∃a. 20↦a * ls(a,0).

$$\{\mathsf{emp}\}\ \texttt{tmalloc}()\ \{\mathsf{ret}\mapsto\_\}$$

$$\{x\mapsto\_\}\ \texttt{tfree}(x)\ \{\mathsf{emp}\}$$

Client side reasoning



$\{\mathsf{emp} * \exists a.\, 20 \mapsto a * \mathsf{ls}(a, 0)\}$

$x = \mathtt{tmalloc}();$

$\{x \mapsto \_ * \exists a.\, 20 \mapsto a * \mathsf{ls}(a, 0)\}$

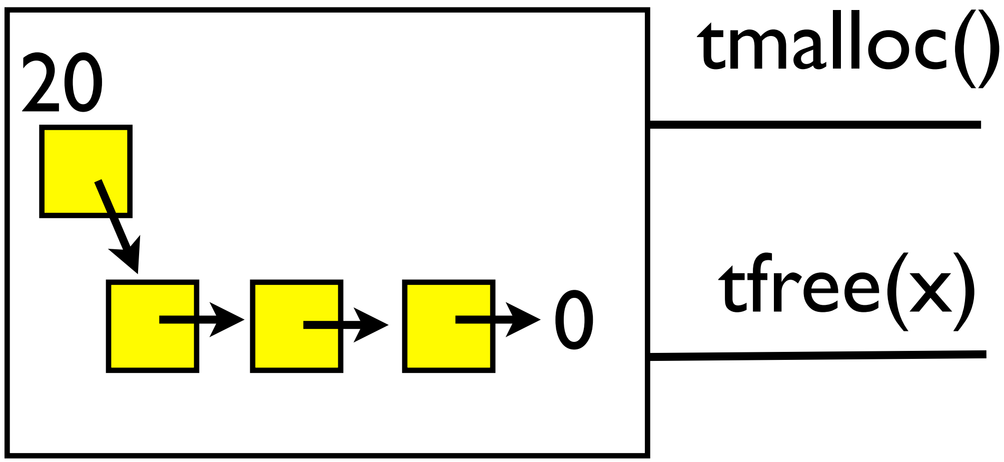
$*x = 0;$

$\{x \mapsto 0 * \exists a.\, 20 \mapsto a * \mathsf{ls}(a, 0)\}$

$\mathtt{tfree}(x);$

$\{\mathsf{emp} * \exists a.\, 20 \mapsto a * \mathsf{ls}(a, 0)\}$

Client side reasoning



$\{\textsf{emp}\}$

$x = \texttt{tmalloc}();$

$\{x \mapsto \_\}$

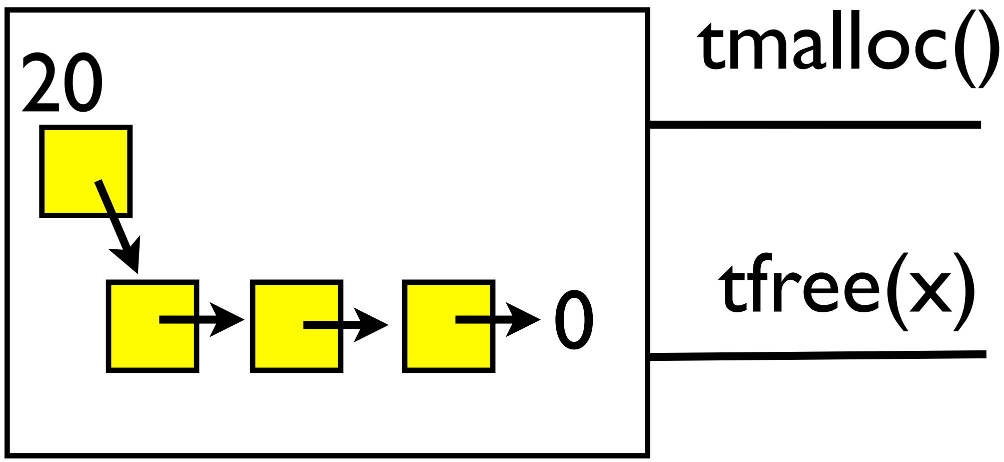
$*x = 0;$

$\{x \mapsto 0\}$

$\texttt{tfree}(x);$

$\{\textsf{emp}\}$

Client side reasoning



$\{\mathsf{emp}\}$

$x = \mathtt{tmalloc}();$

$\{x \mapsto \_\}$

$*x = 0;$

$\{x \mapsto 0\}$

$\mathtt{tfree}(x);$

$\{\mathsf{emp}\}$

$*x = x;$

The invariant gets broken.

Our solution

$$\mathsf{ResInv} \stackrel{def}{=} \exists a.\,(20 \mapsto a) * \mathsf{ls}(a, 0)$$

$$\frac{\begin{array}{l}\{\mathsf{emp} * \mathsf{ResInv}\} M_1 \{\mathsf{ret}{\mapsto}\_ * \mathsf{ResInv}\} \\ \{x{\mapsto}\_ * \mathsf{ResInv}\} M_2 \{\mathsf{emp} * \mathsf{ResInv}\} \\ \{\mathsf{emp}\}\mathtt{tmalloc}()\{\mathsf{ret}{\mapsto}\_\},\ \{x{\mapsto}\_\}\mathtt{tfree}(x)\{\mathsf{emp}\} \ \vdash\ \{A\}C\{B\}\end{array}}{\{A * \mathsf{ResInv}\}\ \mathsf{let}\ (\mathtt{tmalloc}(){=}M_1)\ \mathsf{and}\ (\mathtt{tfree}(x){=}M_2)\ \mathsf{in}\ C\ \{B * \mathsf{ResInv}\}}$$

Our solution

$$\mathsf{ResInv} \stackrel{def}{=} \exists a. (20 \mapsto a) * \mathsf{ls}(a, 0)$$

$$\{\mathsf{emp} * \mathsf{ResInv}\} M_1 \{\mathsf{ret} \mapsto \_ * \mathsf{ResInv}\}$$
$$\{x \mapsto \_ * \mathsf{ResInv}\} M_2 \{\mathsf{emp} * \mathsf{ResInv}\}$$

$$\frac{\{\mathsf{emp}\}\texttt{tmalloc}()\{\mathsf{ret} \mapsto \_\},\ \{x \mapsto \_\}\texttt{tfree}(x)\{\mathsf{emp}\} \ \vdash\ \{A\}C\{B\}}{\{A * \mathsf{ResInv}\}\ \mathsf{let}\ (\texttt{tmalloc}() = M_1)\ \mathsf{and}\ (\texttt{tfree}(x) = M_2)\ \mathsf{in}\ C\ \{B * \mathsf{ResInv}\}}$$

- Reasoning about module implementation show the preservation of ResInv.

Our solution

$$\mathsf{ResInv} \stackrel{def}{=} \exists a.\,(20 \mapsto a) * \mathsf{ls}(a, 0)$$

$$\{\mathsf{emp} * \mathsf{ResInv}\}M_1\{\mathsf{ret}{\mapsto}\_ * \mathsf{ResInv}\}$$

$$\{x{\mapsto}\_ * \mathsf{ResInv}\}M_2\{\mathsf{emp} * \mathsf{ResInv}\}$$

$$\frac{\{\mathsf{emp}\}\mathtt{tmalloc}()\{\mathsf{ret}{\mapsto}\_\},\ \{x{\mapsto}\_\}\mathtt{tfree}(x)\{\mathsf{emp}\} \ \vdash\ \{A\}C\{B\}}{\{A * \mathsf{ResInv}\}\ \mathsf{let}\ (\mathtt{tmalloc}()=M_1)\ \mathsf{and}\ (\mathtt{tfree}(x)=M_2)\ \mathsf{in}\ C\ \{B * \mathsf{ResInv}\}}$$

- Reasoning about module implementation show the preservation of ResInv.
- But, ResInv is hidden in the proof about the client C.

Our solution

$$\mathsf{ResInv} \stackrel{def}{=} \exists a.\, (20 \mapsto a) * \mathsf{ls}(a, 0)$$

$$\frac{\{\mathsf{emp} * \mathsf{ResInv}\} M_1 \{\mathsf{ret} {\mapsto} \_ * \mathsf{ResInv}\} \quad \{x {\mapsto} \_ * \mathsf{ResInv}\} M_2 \{\mathsf{emp} * \mathsf{ResInv}\} \quad \{\mathsf{emp}\}\mathtt{tmalloc}()\{\mathsf{ret}{\mapsto}\_\},\ \{x{\mapsto}\_\}\mathtt{tfree}(x)\{\mathsf{emp}\} \vdash \{A\}C\{B\}}{\{A * \mathsf{ResInv}\}\ \mathsf{let}\ (\mathtt{tmalloc}() {=} M_1)\ \mathsf{and}\ (\mathtt{tfree}(x) {=} M_2)\ \mathsf{in}\ C\ \{B * \mathsf{ResInv}\}}$$

- Reasoning about module implementation show the preservation of ResInv.
- But, ResInv is hidden in the proof about the client C.
- There is a side condition on modified variables by C.

Protection from outside interference

- Tie a cycle in the free list f:

$$x{=}\texttt{tmalloc}();\texttt{tfree}(x);*x{=}x$$

- Cannot fill in ???:

$$\{\mathsf{emp}\}$$
$$x{=}\texttt{tmalloc}();$$
$$\{x \mapsto \_\}$$
$$\texttt{tfree}(x);$$
$$\{\mathsf{emp}\}$$
$$*x{=}x$$
$$\{???\}$$

Modular proc. call rule

$$\frac{\begin{array}{c}\{A * R\}M\{B * R\} \\ \{A\}f()\{B\} \vdash \{P\}C\{Q\}\end{array}}{\vdash \{P * R\}\mathsf{let}\ f()=M\ \mathsf{in}\ C\{Q * R\}}\ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

Modular proc. call rule

$$\frac{\{A * R\}M\{B * R\} \quad \{A\}f()\{B\} \vdash \{P\}C\{Q\}}{\vdash \{P * R\}\text{let } f()=M \text{ in } C\{Q * R\}} \; \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

Equivalent to the hypo. frame rule in terms of derivability.

$$\frac{\{P\}f()\{Q\} \vdash \{A\}C\{B\}}{\{P * R\}f()\{Q * R\} \vdash \{A * R\}C\{B * R\}} \; \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

Modular proc. call rule

$$\frac{\{A * R\}M\{B * R\} \qquad \{A\}f()\{B\} \vdash \{P\}C\{Q\}}{\vdash \{P * R\}\mathsf{let}\ f()=M\ \mathsf{in}\ C\{Q * R\}}\ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

Equivalent to the hypo. frame rule in terms of derivability.

$$\frac{\{P\}f()\{Q\} \vdash \{A\}C\{B\}}{\{P * R\}f()\{Q * R\} \vdash \{A * R\}C\{B * R\}}\ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

Exercise 4: derive one from the other.

Homework

1. The hypo. frame rule is not always sound.

$$\frac{\Gamma, \{P\}f()\{Q\} \ \vdash \ \{A\}C\{B\}}{\Gamma, \{P * R\}f()\{Q * R\} \ \vdash \ \{A * R\}C\{B * R\}} \ \mathsf{FV}(R) \cap \mathsf{Mod}(C) = \emptyset$$

   Look at the "Separation and information hiding" paper, and understand the counterexample there.

2. Write a stack module that allocates a cell using our toy memory manage. Prove the correctness of this module using the hypo. frame rule.

References

- Local reasoning about programs that alter data structures [CSL01].
- Separation and information hiding [POPL04, TOPLAS09].