Separation Logic for Higher-order Programs

Day 1 :
Higher-order Frame Rules

Lars Birkedal (IT University of Copenhagen)
Hongseok Yang (Queen Mary, Univ. of London)

List Creation

$\{\mathsf{emp}\}$

$x{=}0; n{=}0;$

$\texttt{while}\ (\mathsf{NONDET})\ \{$

$\quad t{=}\texttt{malloc}(2);$

$\quad *(t+1){=}x; *t{=}n; n{=}n{+}1;$

$\quad x{=}t;$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

$\{\mathsf{emp}\}$

$x{=}0; n{=}0;$

INV : $\mathsf{ls}\,(x, 0)$

$\texttt{while}$ (NONDET) $\{$

$t{=}\texttt{malloc}(2);$

$*(t+1){=}x; *t{=}n; n{=}n{+}1;$

$x{=}t;$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

$\{\mathsf{emp}\}$

$x{=}0;\, n{=}0;$

INV : $\mathsf{ls}\,(x, 0)$

`while` (NONDET) $\{$

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\texttt{malloc}(2);$

$*(t+1){=}x;\, *t{=}n;\, n{=}n{+}1;$

$x{=}t;$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

$\{\mathsf{emp}\}$

$x{=}0; n{=}0;$

$\mathsf{INV}: \mathsf{ls}\,(x, 0)$

$\texttt{while}$ (NONDET) $\{$

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\texttt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t+1){=}x; *t{=}n; n{=}n{+}1;$

$x{=}t;$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

{emp}

$x{=}0; n{=}0;$

INV : $\mathsf{ls}\,(x, 0)$

$\texttt{while}$ (NONDET) {

{$\mathsf{ls}\,(x, 0)$}

$t{=}\texttt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t+1){=}x; *t{=}n; n{=}n{+}1;$

$\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$x{=}t;$

}

{$\mathsf{ls}\,(x, 0)$}

List Creation

$\{\mathsf{emp}\}$

$x{=}0; n{=}0;$

$\mathsf{INV} : \mathsf{ls}\,(x, 0)$

$\texttt{while}$ (NONDET) $\{$

$\quad\{\mathsf{ls}\,(x, 0)\}$

$\quad t{=}\texttt{malloc}(2);$

$\quad\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$\quad *(t+1){=}x; *t{=}n; n{=}n{+}1;$

$\quad\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$\quad x{=}t;$

$\quad\{\mathsf{ls}\,(x, 0)\}$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

$\{\mathsf{emp}\}$

$\{P\}t{=}\mathsf{malloc}(2)\{(t \mapsto \_, \_) * P\}$
$(\text{when } t \notin \mathsf{FV}(P))$

1. Can we ignore internal data structures of malloc?

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\mathtt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t+1){=}x; *t{=}n; n{=}n{+}1;$

$\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$x{=}t;$

$\{\mathsf{ls}\,(x, 0)\}$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

List Creation

$\{\mathsf{emp}\}$

$\{P\}t{=}\mathsf{malloc}(2)\{(t \mapsto \_, \_) * P\}$
$(\text{when } t \notin \mathsf{FV}(P))$

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\texttt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t+1){=}x; *t{=}n; n{=}n{+}1;$

$\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$x{=}t;$

$\{\mathsf{ls}\,(x, 0)\}$

$\}$

$\{\mathsf{ls}\,(x, 0)\}$

1. Can we ignore internal data structures of malloc?

struct kmem_cache {
    void $(*$ctor$)($**void** $*)$;   .... and about 19 fields ....
};

extern struct kmem_cache kmalloc_caches[PAGE_SHIFT $+$ 1];

List Creation

$\{\mathsf{emp}\}$

$x{=}0;\, n{=}0;$

$\mathsf{INV} : \mathsf{ls}\,(x, 0)$

$\mathtt{while}$ (NONDET) {

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\mathtt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t+1){=}x;\, *t{=}n;\, n{=}n{+}1;$

$\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$x{=}t;$

$\{\mathsf{ls}\,(x, 0)\}$

}

$\{\mathsf{ls}\,(x, 0)\}$

1. Can we ignore internal data structures of malloc?
2. What about code pointers?

struct kmem_cache {
  void $(*\,$ctor$)($**void** $*\,);$ .... and about 19 fields ....
};

extern struct kmem_cache kmalloc_caches[PAGE_SHIFT $+$ 1];

List Creation

$\{\mathsf{emp}\}$

$x{=}0;\, n{=}0;$

$\mathsf{INV} : \mathsf{ls}\,(x, 0)$

$\texttt{while}$ (NONDET) {

$\{\mathsf{ls}\,(x, 0)\}$

$t{=}\texttt{malloc}(2);$

$\{(t \mapsto \_, \_) * \mathsf{ls}\,(x, 0)\}$

$*(t + 1){=}x;\, *t{=}n;\, n{=}n{+}1;$

$\{(t \mapsto \_, x) * \mathsf{ls}\,(x, 0)\}$

$x{=}t;$

$\{\mathsf{ls}\,(x, 0)\}$

}

$\{\mathsf{ls}\,(x, 0)\}$

1. Can we ignore internal data structures of malloc?
2. What about code pointers?
3. What if malloc is updated?

```
struct kmem_c
    void (* ctor
                         ....
};

extern struct kmem_cache kmalloc_caches[PAGE_SHIFT + 1];
```

Doug Lee's Malloc

Topics of the Course

1. Can we ignore internal data structures of malloc?
2. What about code pointers?
3. What if malloc is updated?

Topics of the Course

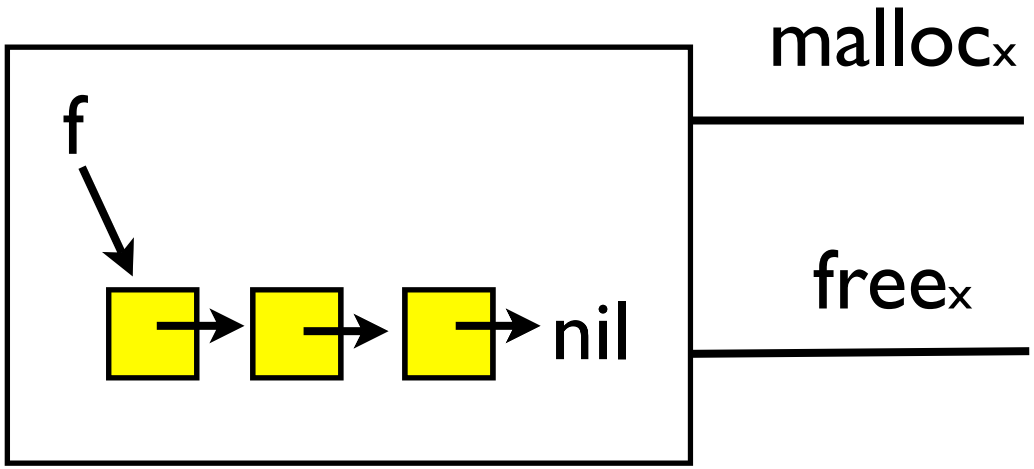
1. Higher-order Frame Rule.
2. General References.
3. Relational Reading of Sep. Logic.
4. Higher-order Sep. Logic.

1. Can we ignore internal data structures of malloc?
2. What about code pointers?
3. What if malloc is updated?

Outcome of Today's Lectures

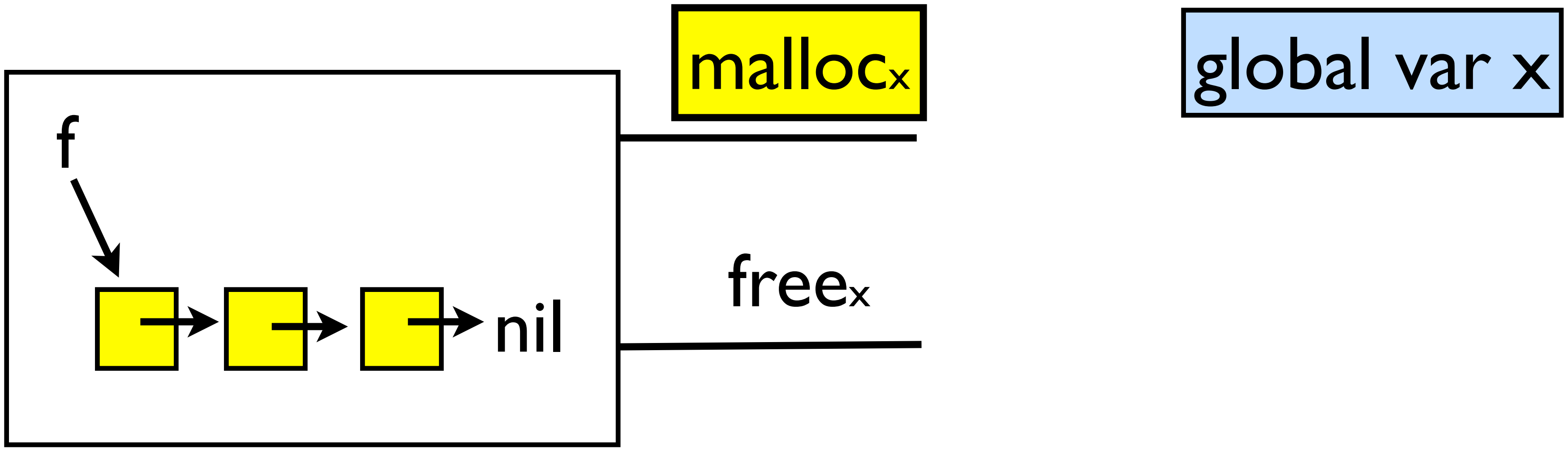
- Be able to use HO frame rules and describe how it is related to information hiding.
- Learn one semantic technique: resource Kripke semantics.

Toy Memory Manager

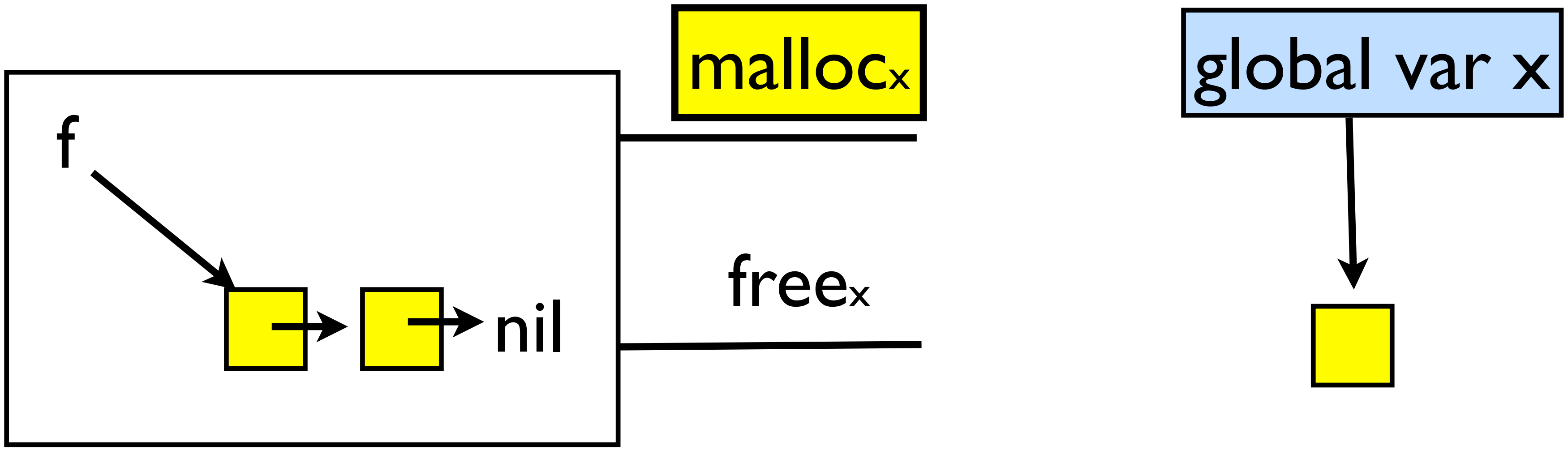


global var x

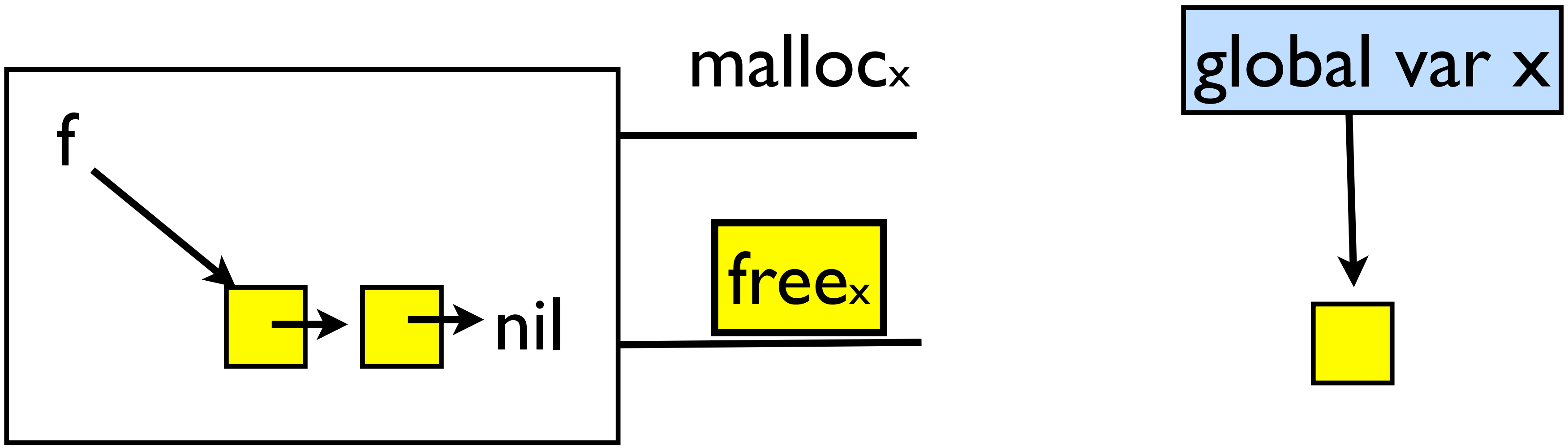
Toy Memory Manager



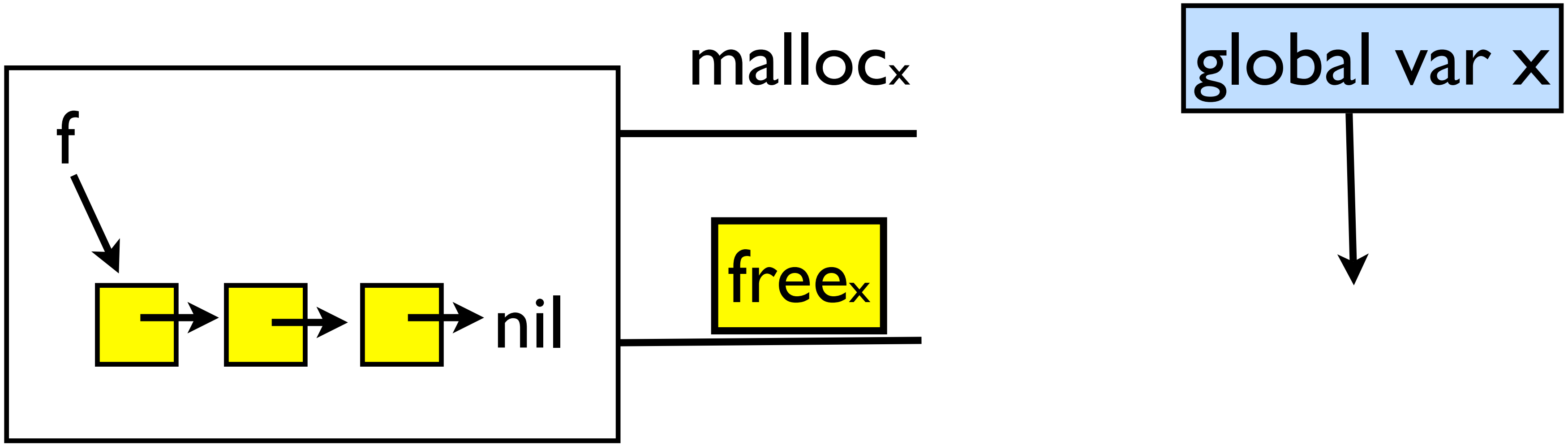
Toy Memory Manager



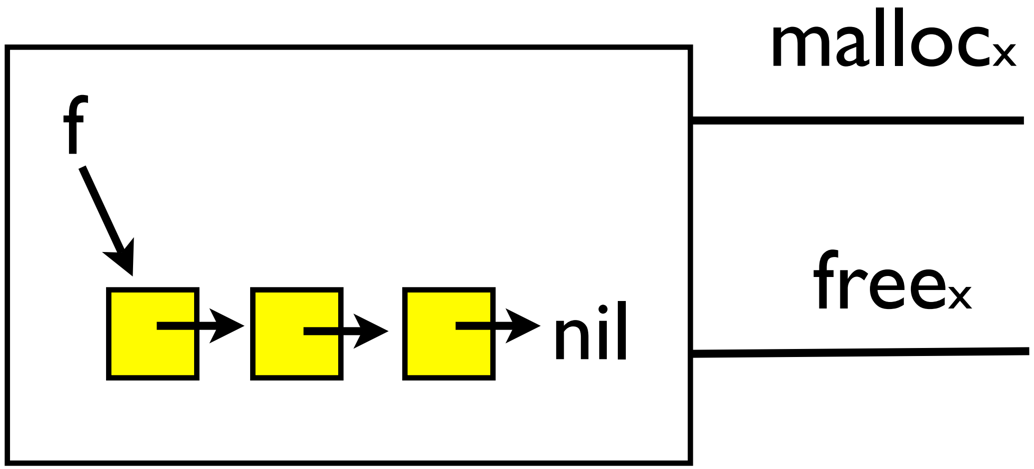
Toy Memory Manager



Toy Memory Manager



Toy Memory Manager



global var x

- Implementation

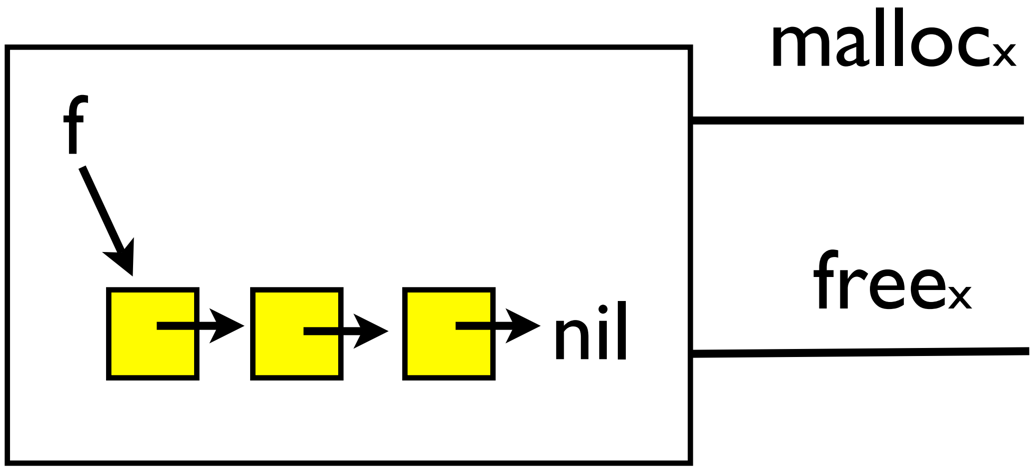
$\texttt{malloc}_x \;::=\; \texttt{if}\ (f{=}{=}\texttt{nil})\ \{\ x{=}\texttt{new}();\ \}\ \texttt{else}\ \{\ x{=}f; f{=}f{\rightarrow}nxt;\ \}$

$\texttt{free}_x \;::=\; x{\rightarrow}nxt{=}f; f{=}x;$

- Specification

$\{\mathsf{emp}{*}list(f)\}\texttt{malloc}_x\{(x{\mapsto}\_,\_){*}list(f)\}$ $\{(x{\mapsto}\_,\_){*}list(f)\}\texttt{free}_x\{\mathsf{emp}{*}list(f)\}$

Toy Memory Manager



global var x

- Implementation

$\texttt{malloc}_x$ ::= $\texttt{if}$ $(f{=}{=}\texttt{nil})$ $\{$ $x{=}\texttt{new}();$ $\}$ $\texttt{else}$ $\{$ $x{=}f; f{=}f{\to}nxt;$ $\}$

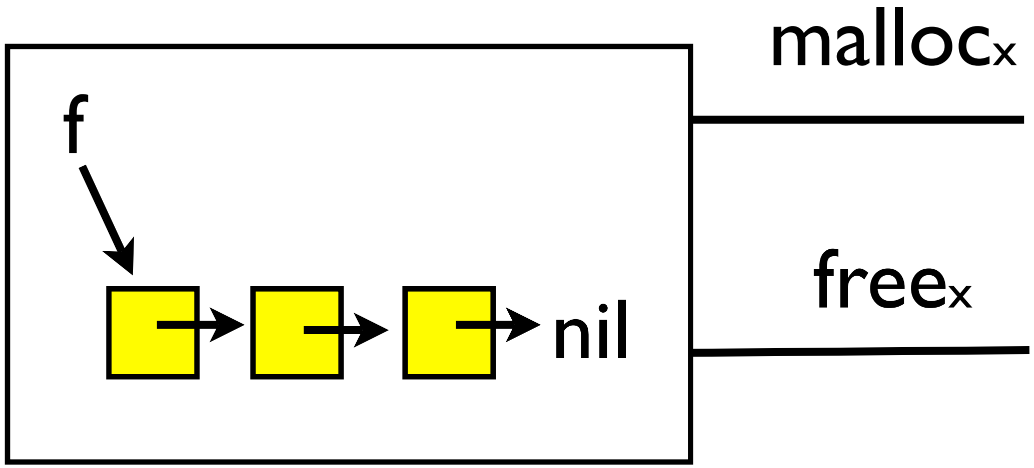
$\texttt{free}_x$ ::= $x{\to}nxt{=}f; f{=}x;$

- Specification

- Resource Inv: $list(f)$

$\{\texttt{emp}\}\texttt{malloc}_x\{(x{\mapsto}\_,\_)\}$ $\{(x{\mapsto}\_,\_)\}\texttt{free}_x\{\texttt{emp}\}$

Client Side Reasoning



global var x

$\{\mathsf{emp} * \mathit{list}(f)\}$

$\texttt{malloc}_x;$

$\{(x \mapsto \_, \_) * \mathit{list}(f)\}$

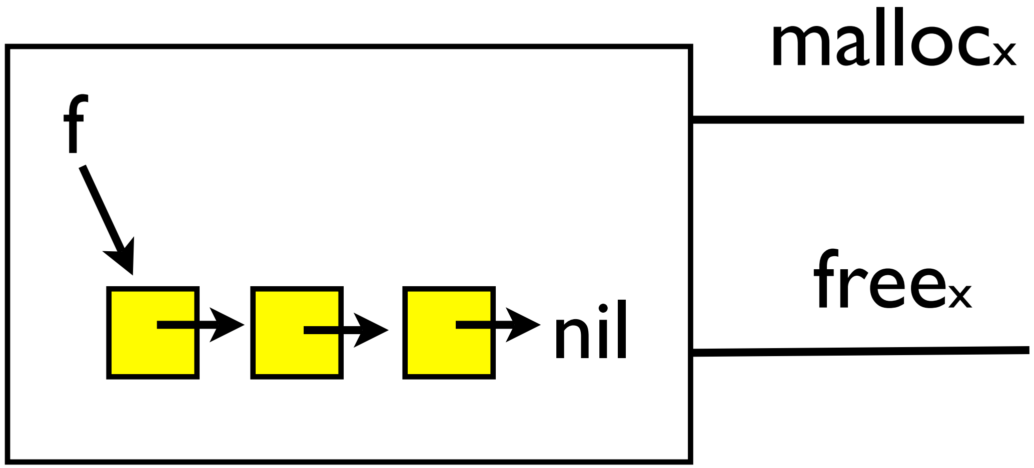
$*x\texttt{=nil};$

$\{(x \mapsto \texttt{nil}, \_) * \mathit{list}(f)\}$

$\texttt{free}_x;$

$\{\mathsf{emp} * \mathit{list}(f)\}$

Client Side Reasoning



global var x

$\{\mathsf{emp}\}$

$\texttt{malloc}_x;$

$\{(x \mapsto \_, \_)\}$

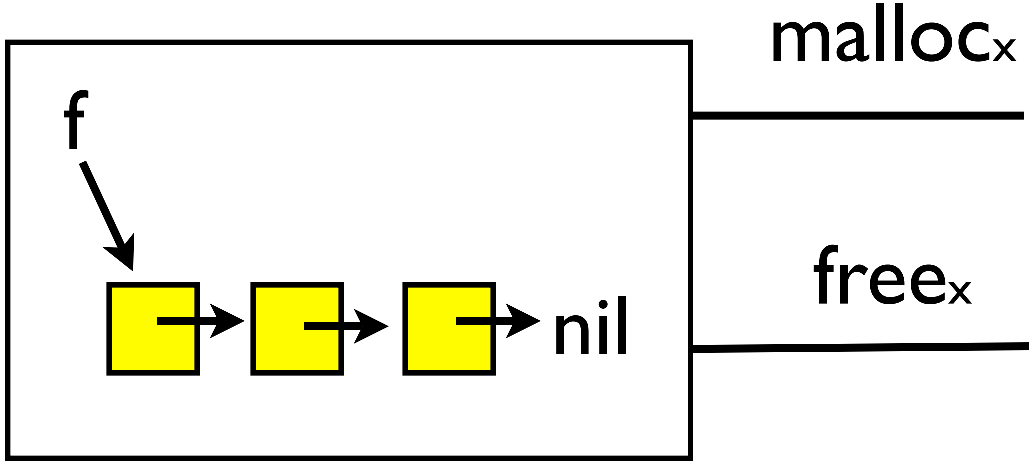
$*x = \texttt{nil};$

$\{(x \mapsto \texttt{nil}, \_)\}$

$\texttt{free}_x;$

$\{\mathsf{emp}\}$

Client Side Reasoning



global var x

$\{\mathsf{emp}\}$

$\texttt{malloc}_x;$

$\{(x \mapsto \_, \_)\}$

$*x = \texttt{nil};$

$\{(x \mapsto \texttt{nil}, \_)\}$

$\texttt{free}_x;$

$\{\mathsf{emp}\}$

$*x = x;$

Invariant list(f) broken

Our Solution

$$\dfrac{\begin{array}{l}\{\mathsf{emp} * \mathit{list}(f)\}M_1\{x \mapsto \_,\_ * \mathit{list}(f)\} \\ \{x \mapsto \_,\_ * \mathit{list}(f)\}M_2\{\mathsf{emp} * \mathit{list}(f)\} \\ \{\mathsf{emp}\}\mathtt{malloc}_x\{x \mapsto \_,\_\} \wedge \{x \mapsto \_,\_\}\mathtt{free}_x\{\mathsf{emp}\} \quad \Rightarrow \quad \{A\}C\{B\}\end{array}}{\{A * \mathit{list}(f)\}\ \mathsf{let}\ (\mathtt{malloc}_x{=}M_1)\ \mathsf{and}\ (\mathtt{free}_x{=}M_2)\ \mathsf{in}\ C\ \{B * \mathit{list}(f)\}}\ \text{Cond. on Vars}$$

Our Solution

$$\frac{\boxed{\begin{array}{l}\{\mathsf{emp} * \mathit{list}(f)\}M_1\{x{\mapsto}\_,\_ * \mathit{list}(f)\}\\ \{x{\mapsto}\_,\_ * \mathit{list}(f)\}M_2\{\mathsf{emp} * \mathit{list}(f)\}\end{array}} \\ \{\mathsf{emp}\}\mathtt{malloc}_x\{x{\mapsto}\_,\_\} \;\wedge\; \{x{\mapsto}\_,\_\}\mathtt{free}_x\{\mathsf{emp}\} \quad\Rightarrow\quad \{A\}C\{B\}}{\{A * \mathit{list}(f)\}\ \mathsf{let}\ (\mathtt{malloc}_x{=}M_1)\ \mathsf{and}\ (\mathtt{free}_x{=}M_2)\ \mathsf{in}\ C\ \{B * \mathit{list}(f)\}}\ \text{Cond. on Vars}$$

Our Solution

$$\frac{\begin{array}{l} \{\mathsf{emp} * \mathit{list}(f)\}M_1\{x\mapsto\_,\_ * \mathit{list}(f)\} \\ \{x\mapsto\_,\_ * \mathit{list}(f)\}M_2\{\mathsf{emp} * \mathit{list}(f)\} \\ \boxed{\{\mathsf{emp}\}\mathtt{malloc}_x\{x\mapsto\_,\_\} \wedge \{x\mapsto\_,\_\}\mathtt{free}_x\{\mathsf{emp}\} \;\Rightarrow\; \{A\}C\{B\}} \end{array}}{\{A * \mathit{list}(f)\}\ \mathsf{let}\ (\mathtt{malloc}_x{=}M_1)\ \mathsf{and}\ (\mathtt{free}_x{=}M_2)\ \mathsf{in}\ C\ \{B * \mathit{list}(f)\}}\ \text{Cond. on Vars}$$

The internal list is absent in reasoning.

Protection from Outside Interference

- Tie a cycle in the free list f.

  $\mathtt{malloc}_x; \mathtt{free}_x; *x = x$

- Cannot fill in ???:

  $\{\mathsf{emp}\}$

  $\mathtt{malloc}_x;$

  $\{x \mapsto \_, \_\}$

  $\mathtt{free}_x;$

  $\{\mathsf{emp}\}$

  $*x = x$

  $\{???\}$

Second-order Frame Rule

$$\frac{\{P\}k\{Q\} \;\; \Rightarrow \;\; \{A\}C\{B\}}{\{P * R\}k\{Q * R\} \;\; \Rightarrow \;\; \{A * R\}C\{B * R\}} \; \text{Cond. on Vars}$$

Second-order Frame Rule

$$\frac{\{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}}{\{P * R\}k\{Q * R\} \;\Rightarrow\; \{A * R\}C\{B * R\}}\ \text{Cond. on Vars}$$

- Can derive the modular proc. call rule.

$$\frac{\{A * R\}M\{B * R\} \qquad \{A\}k\{B\} \;\Rightarrow\; \{P\}C\{Q\}}{\{P * R\}\mathsf{let}\ k{=}M\ \mathsf{in}\ C\{Q * R\}}\ \text{Cond. on Vars}$$

Exercise: Derive it.

Issues

$$\frac{\{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}}{\{P * R\}k\{Q * R\} \;\Rightarrow\; \{A * R\}C\{B * R\}}\ \text{Cond. on Vars}$$

1. The rule leads to inconsistency. (Reynolds's example.)

2. The side condition is annoying.

3. What about higher-order k?

Issues

$$\frac{\{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}}{\{P * R\}k\{Q * R\} \;\Rightarrow\; \{A * R\}C\{B * R\}}\ \text{Cond. on Vars}$$

1. The rule leads to inconsistency. (Reynolds's example.)

   Drop the conjunction rule. Use only precise R.

2. The side condition is annoying.

   Put vars into the heap. Or, use read-only vars.

3. What about higher-order k?

   Resource Kripke models.

Issues

$$\frac{\{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}}{\{P * R\}k\{Q * R\} \;\Rightarrow\; \{A * R\}C\{B * R\}}\ \text{Cond. on Vars}$$

1. The rule leads to inconsistency. (Reynolds's example.)

   Drop the conjunction rule. Use only precise R.

2. The side condition is annoying.

   Put vars into the heap. Or, use read-only vars.

3. What about higher-order k?

   Resource Kripke models.

Storage Model

$$\begin{aligned} \mathsf{Val} &= \mathsf{Locs} \cup \{nil\} \\ \mathsf{Heaps} &= \mathsf{Locs} \rightharpoonup_{fin} \mathsf{Vals} \times \mathsf{Vals} \\ \mathsf{States} &= \mathsf{Heaps} \end{aligned}$$

Storage Model

No ints. Location values only.

$$\boxed{\mathsf{Val} \;=\; \mathsf{Locs} \cup \{nil\}}$$

$$\begin{aligned} \mathsf{Heaps} &= \mathsf{Locs} \rightharpoonup_{fin} \mathsf{Vals} \times \mathsf{Vals} \\ \mathsf{States} &= \mathsf{Heaps} \end{aligned}$$

Storage Model

$$
\begin{array}{rcl}
\mathsf{Val} & = & \mathsf{Locs} \cup \{nil\} \\
\mathsf{Heaps} & = & \mathsf{Locs} \rightharpoonup_{fin} \boxed{\mathsf{Vals} \times \mathsf{Vals}} \\
\mathsf{States} & = & \mathsf{Heaps}
\end{array}
$$

Binary cells.

Storage Model

$$\begin{array}{rcl} \mathsf{Val} & = & \mathsf{Locs} \cup \{nil\} \\ \mathsf{Heaps} & = & \mathsf{Locs} \rightharpoonup_{fin} \mathsf{Vals} \times \mathsf{Vals} \\ \mathsf{States} & = & \mathsf{Heaps} \end{array}$$

No stack. Heap only.

Storage Model

$$
\begin{aligned}
\mathsf{Val} &= \mathsf{Locs} \cup \{nil\} \\
\mathsf{Heaps} &= \mathsf{Locs} \rightharpoonup_{fin} \mathsf{Vals} \times \mathsf{Vals} \\
\mathsf{States} &= \mathsf{Heaps}
\end{aligned}
$$

Higher-order Programs with Read-only Vars

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \to \tau \\
E & ::= & x \mid \mathsf{nil} \\
M & ::= & \mathsf{let}\, x{=}\mathsf{new}\ \mathsf{in}\, M \mid \mathsf{free}(E) \mid \mathsf{let}\, x{=}E.f\ \mathsf{in}\, M \mid E.f{:=}E \quad (f \in \{0,1\}) \\
& \mid & M; M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
& \mid & k \mid \lambda k{:}\tau.\, M \mid M\, M \mid \mathsf{fix}\, M
\end{array}
$$

Higher-order Programs with Read-only Vars

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \to \tau \\
\boxed{E} & \boxed{::=} & \boxed{x \mid \mathsf{nil}} \\
M & ::= & \mathsf{let}\ x{=}\mathsf{new}\ \mathsf{in}\ M \mid \mathsf{free}(E) \mid \mathsf{let}\ x{=}E.f\ \mathsf{in}\ M \mid E.f{:=}E \quad (f \in \{0,1\}) \\
& \mid & M; M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
& \mid & k \mid \lambda k{:}\tau.\, M \mid M\ M \mid \mathsf{fix}\, M
\end{array}
$$

Higher-order Programs with Read-only Vars

$$\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \to \tau \\
E & ::= & x \mid \mathsf{nil} \\
M & ::= & \boxed{\mathsf{let}\ x{=}\mathsf{new}\ \mathsf{in}\ M \mid \mathsf{free}(E) \mid \mathsf{let}\ x{=}E.f\ \mathsf{in}\ M \mid E.f{:=}E \quad (f \in \{0,1\})} \\
& \mid & M;M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
& \mid & k \mid \lambda k{:}\tau.\, M \mid M\ M \mid \mathsf{fix}\ M
\end{array}$$

1. Binary cell access.
2. let-binding, instead of variable update.

Higher-order Programs with Read-only Vars

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \to \tau \\
E & ::= & x \mid \mathsf{nil} \\
M & ::= & \mathsf{let}\ x{=}\mathsf{new}\ \mathsf{in}\ M \mid \mathsf{free}(E) \mid \mathsf{let}\ x{=}E.f\ \mathsf{in}\ M \mid E.f{:=}E \quad (f \in \{0,1\}) \\
 & \mid & M ; M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
 & \mid & k \mid \lambda k{:}\tau.\,M \mid M\ M \mid \mathsf{fix}\ M
\end{array}
$$

Higher-order Programs with Read-only Vars

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \rightarrow \tau \\
E & ::= & x \mid \mathsf{nil} \\
M & ::= & \mathsf{let}\ x{=}\mathsf{new}\ \mathsf{in}\ M \mid \mathsf{free}(E) \mid \mathsf{let}\ x{=}E.f\ \mathsf{in}\ M \mid E.f{:=}E \quad (f \in \{0,1\}) \\
& \mid & M; M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
& \mid & \boxed{k \mid \lambda k{:}\tau.\, M \mid M\ M \mid \mathsf{fix}\ M}
\end{array}
$$

Higher-order functions with recursions.

Higher-order Programs with Read-only Vars

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{com} \mid \tau \to \tau \\
E & ::= & x \mid \mathsf{nil} \\
M & ::= & \mathsf{let}\ x{=}\mathsf{new}\ \mathsf{in}\ M \mid \mathsf{free}(E) \mid \mathsf{let}\ x{=}E.f\ \mathsf{in}\ M \mid E.f{:=}E \quad (f \in \{0,1\}) \\
& \mid & M; M \mid \mathsf{if}\ (E{=}E)\ M\ M \\
& \mid & k \mid \lambda k{:}\tau.\, M \mid M\ M \mid \mathsf{fix}\, M
\end{array}
$$

Assertions and Specifications

$$P, Q \ ::= \ E \mapsto F \mid \mathsf{emp} \mid P \Rightarrow Q \mid P \wedge Q \mid \forall x.\, P \mid P * Q \mid \ldots$$

$$\varphi, \psi \ ::= \ \{P\}M\{Q\} \mid \varphi \Rightarrow \psi \mid \varphi \wedge \psi \mid \forall x.\, \varphi \mid \forall k.\, \varphi \mid \varphi \otimes P \mid \ldots$$

Assertions and Specifications

Assertions:

1. Properties of states.
2. Contains x only.
3. Classical logic and separating connectives.

$$\boxed{P, Q \ ::= \ E \mapsto F \mid \mathsf{emp} \mid P \Rightarrow Q \mid P \wedge Q \mid \forall x.\, P \mid P * Q \mid \ldots}$$

$$\varphi, \psi \ ::= \ \{P\}M\{Q\} \mid \varphi \Rightarrow \psi \mid \varphi \wedge \psi \mid \forall x.\, \varphi \mid \forall k.\, \varphi \mid \varphi \otimes P \mid \ldots$$

Assertions and Specifications

Assertions:

1. Properties of states.
2. Contains x only.
3. Classical logic and separating connectives.

$$P, Q \ ::= \ E \mapsto F \mid \mathsf{emp} \mid P \Rightarrow Q \mid P \wedge Q \mid \forall x.\, P \mid P * Q \mid \ldots$$

$$\boxed{\varphi, \psi \ ::= \ \{P\}M\{Q\} \mid \varphi \Rightarrow \psi \mid \varphi \wedge \psi \mid \forall x.\, \varphi \mid \forall k.\, \varphi \mid \varphi \otimes P \mid \ldots}$$

Specifications:

1. Properties of programs.
2. Contains x and k.
3. Intuitionistic logic and invariant extension.

Invariant Extension

$$\varphi \otimes P$$

- *-conjoin P to the pre/posts of all the triples in φ.
- Distribution axioms for invariant extensions:

$$
\begin{array}{rcll}
\{A\}M\{B\} \otimes P & \Leftrightarrow & \{A * P\}M\{B * P\} & \\
(\varphi \oplus \psi) \otimes P & \Leftrightarrow & (\varphi \otimes P \oplus \psi \otimes P) & (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\}) \\
(\delta\, i.\, \varphi) \otimes P & \Leftrightarrow & (\delta\, i.\, \varphi \otimes P) & (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))
\end{array}
$$

$$\Big(\forall k.\, \{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}\Big) \otimes R$$

sion

- *-conjoin P to the pre/posts of all the triples in φ.
- Distribution axioms for invariant extensions:

$$
\begin{array}{rcll}
\{A\}M\{B\} \otimes P & \Leftrightarrow & \{A * P\}M\{B * P\} & \\
(\varphi \oplus \psi) \otimes P & \Leftrightarrow & (\varphi \otimes P \oplus \psi \otimes P) & (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\}) \\
(\delta\, i.\, \varphi) \otimes P & \Leftrightarrow & (\delta\, i.\, \varphi \otimes P) & (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))
\end{array}
$$

$$\Big(\forall k.\, \{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R$$

$$\Leftrightarrow \Big(\forall k.\, \Big(\{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R\Big)$$

sion

- *-conjoin P to the pre/posts of all the triples in φ.
- Distribution axioms for invariant extensions:

$$\begin{array}{rcll}
\{A\}M\{B\} \otimes P & \Leftrightarrow & \{A * P\}M\{B * P\} & \\
(\varphi \oplus \psi) \otimes P & \Leftrightarrow & (\varphi \otimes P \oplus \psi \otimes P) & (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\}) \\
\boxed{(\delta\, i.\, \varphi) \otimes P \quad \Leftrightarrow \quad (\delta\, i.\, \varphi \otimes P)} & & & (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))
\end{array}$$

$$\Big(\forall k.\, \{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}\Big) \otimes R$$
$$\Leftrightarrow\; \Big(\forall k.\, \Big(\{P\}k\{Q\} \;\Rightarrow\; \{A\}C\{B\}\Big) \otimes R\Big)$$
$$\Leftrightarrow\; \Big(\forall k.\, \{P\}k\{Q\} \otimes R \;\Rightarrow\; \{A\}C\{B\} \otimes R\Big)$$

sion

- *-conjoin P to the pre/posts of all the triples in φ.
- Distribution axioms for invariant extensions:

$$\begin{array}{rcll}
\{A\}M\{B\} \otimes P & \Leftrightarrow & \{A * P\}M\{B * P\} & \\
\boxed{(\varphi \oplus \psi) \otimes P \;\;\Leftrightarrow\;\; (\varphi \otimes P \oplus \psi \otimes P)} & & & (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\}) \\
(\delta\, i.\, \varphi) \otimes P & \Leftrightarrow & (\delta\, i.\, \varphi \otimes P) & (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))
\end{array}$$

$$
\begin{aligned}
&\Big(\forall k.\, \{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R \\
\Leftrightarrow\;& \Big(\forall k.\, \Big(\{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R\Big) \\
\Leftrightarrow\;& \Big(\forall k.\, \{P\}k\{Q\} \otimes R \Rightarrow \{A\}C\{B\} \otimes R\Big) \\
\Leftrightarrow\;& \Big(\forall k.\, \{P * R\}k\{Q * R\} \Rightarrow \{A * R\}C\{B * R\}\Big)
\end{aligned}
$$

sion

- *-conjoin P to the pre/posts of all the triples in φ.
- Distribution axioms for invariant extensions:

$$\boxed{\{A\}M\{B\} \otimes P \;\Leftrightarrow\; \{A * P\}M\{B * P\}}$$

$$(\varphi \oplus \psi) \otimes P \;\Leftrightarrow\; (\varphi \otimes P \oplus \psi \otimes P) \qquad (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\})$$

$$(\delta\, i.\, \varphi) \otimes P \;\Leftrightarrow\; (\delta\, i.\, \varphi \otimes P) \quad (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))$$

$$\Big(\forall k.\, \{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R$$
$$\Leftrightarrow \Big(\forall k.\, \Big(\{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\Big) \otimes R\Big)$$
$$\Leftrightarrow \Big(\forall k.\, \{P\}k\{Q\} \otimes R \Rightarrow \{A\}C\{B\} \otimes R\Big)$$
$$\Leftrightarrow \Big(\forall k.\, \{P * R\}k\{Q * R\} \Rightarrow \{A * R\}C\{B * R\}\Big)$$

sion

Question: Guess what the HO frame rule is.

- Distribution axioms for invariant extensions:

$$\{A\}M\{B\} \otimes P \quad \Leftrightarrow \quad \{A * P\}M\{B * P\}$$
$$(\varphi \oplus \psi) \otimes P \quad \Leftrightarrow \quad (\varphi \otimes P \oplus \psi \otimes P) \qquad (\text{where } \oplus \in \{\wedge, \vee, \Rightarrow\})$$
$$(\delta\, i.\, \varphi) \otimes P \quad \Leftrightarrow \quad (\delta\, i.\, \varphi \otimes P) \quad (\text{where } \delta \in \{\forall, \exists\},\ i \in \{x, k\},\ i \notin \mathsf{FV}(P))$$

HO Frame Rule

$$\varphi \quad \Rightarrow \quad \varphi \otimes P$$

HO Frame Rule

$$\varphi \quad \Rightarrow \quad \varphi \otimes P$$

First-order Frame Rule

$$\{P\}M\{Q\}$$

$$\Rightarrow \; \{P\}M\{Q\} \otimes R$$

$$\Rightarrow \; \{P * R\}M\{Q * R\}$$

HO Frame Rule

$$\varphi \;\Rightarrow\; \varphi \otimes P$$

Fir

Second-order Frame Rule

$$\Big(\{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\big)$$

$$\Rightarrow \Big(\{P\}k\{Q\} \Rightarrow \{A\}C\{B\}\big) \otimes R$$

$$\Rightarrow \Big(\{P\}k\{Q\} \otimes R \Rightarrow \{A\}C\{B\} \otimes R\Big)$$

$$\Rightarrow \Big(\{P * R\}k\{Q * R\} \Rightarrow \{A * R\}C\{B * R\}\Big)$$

HO Frame Rule

$$\varphi \;\Rightarrow\; \varphi \otimes P$$

Third-order Frame Rule

$$\Big(\Big(\forall k'.\,\{P\}k'\{Q\} \Rightarrow \{P'\}k(k')\{Q'\}\Big) \Rightarrow \{A\}M\{B\}\Big)$$

$$\Rightarrow\; \Big(\Big(\forall k'.\,\{P\}k'\{Q\} \Rightarrow \{P'\}k(k')\{Q'\}\Big) \Rightarrow \{A\}M\{B\}\Big) \otimes R$$

$$\Rightarrow\; \Big(\Big(\forall k'.\,\{P * R\}k'\{Q * R\} \Rightarrow \{P' * R\}k(k')\{Q' * R\}\Big) \Rightarrow \{A * R\}M\{B * R\}\Big)$$

HO Frame Rule

$$\varphi \;\Rightarrow\; \varphi \otimes P$$

Third-order Frame Rule

$$\Big(\big(\forall k'.\{P\}k'\{Q\} \Rightarrow \{P'\}k(k')\{Q'\}\big) \Rightarrow \{A\}M\{B\}\Big)$$

$$\Rightarrow\; \Big(\big(\forall k'.\{P\}k'\{Q\} \Rightarrow \{P'\}k(k')\{Q'\}\big) \Rightarrow \{A\}M\{B\}\Big) \otimes R$$

$$\Rightarrow\; \Big(\big(\forall k'.\{P * R\}k'\{Q * R\} \Rightarrow \{P' * R\}k(k')\{Q' * R\}\big) \Rightarrow \{A * R\}M\{B * R\}\Big)$$

$$\Rightarrow\; \Big(\big(\forall k'.\{P * R\}k'\{Q * R\} \Rightarrow \{P'\}k(k')\{Q'\}\big) \Rightarrow \{A * R\}M\{B * R\}\Big)$$

Soundness

- Not trivial. Reynolds's counter-example.
- Two positions for the soundness proof.
  - Standard semantics. Hard work.
  - Non-standard semantics. Less work.

Trick : Resource Kripke Semantics

$$\eta, \mu \qquad \models \varphi$$

- η gives the meaning of k and μ the meaning of x.

Trick : Resource Kripke Semantics

$$\eta, \mu, \boxed{w} \models \varphi$$

- η gives the meaning of k and μ the meaning of x.
- Resource Kripke world:

$$w \in \mathcal{P}(\mathsf{Heaps}), \qquad w \sqsubseteq w' \text{ iff } \exists w_0.\, w * w_0 = w'.$$

Trick : Resource Kripke Semantics

$$\eta, \mu, w \ \models \ \varphi$$

- η gives the meaning of k and μ the meaning of x.
- Resource Kripke world:

$$w \in \mathcal{P}(\text{Heaps}), \qquad w \sqsubseteq w' \ \text{ iff } \ \exists w_0.\, w * w_0 = w'.$$

- Use standard semantics of intuitionistic logic:

$$\begin{array}{rcl} \eta, \mu, w \ \models \varphi \Rightarrow \psi & \text{iff} & \forall w'.\, \text{if } w' \sqsupseteq w \text{ and } (\eta, \mu, w' \models \varphi), \text{ then } \eta, \mu, w' \models \psi \\ & \text{iff} & \forall w_0.\, \text{if } (\eta, \mu, w * w_0 \models \varphi), \text{ then } (\eta, \mu, w * w_0 \models \psi) \\ \eta, \mu, w \ \models \varphi \otimes P & \text{iff} & \eta, \mu, (w * [\![P]\!]_\mu) \models \varphi \end{array}$$

$$(\text{where } [\![P]\!]_\mu = \{h \mid \mu, h \models P\})$$

Trick : Resource Kripke Semantics

$$\eta, \mu, w \;\models\; \varphi$$

- η gives the meaning of k and μ the meaning of x.
- Resource Kripke world:

Standard Kripke Semantics

$$w \sqsubseteq w' \;\text{ iff }\; \exists w_0.\, w * w_0 = w'.$$

- antics of intuitionistic logic:

$$\begin{array}{lll}
\eta, \mu, w \models \varphi \Rightarrow \psi & \text{iff} & \forall w'.\text{ if } w' \sqsupseteq w \text{ and } (\eta, \mu, w' \models \varphi),\text{ then } \eta, \mu, w' \models \psi \\
 & \text{iff} & \forall w_0.\text{ if } (\eta, \mu, w * w_0 \models \varphi),\text{ then } (\eta, \mu, w * w_0 \models \psi) \\
\eta, \mu, w \models \varphi \otimes P & \text{iff} & \eta, \mu, (w * [\![P]\!]_\mu) \models \varphi
\end{array}$$

$$(\text{where } [\![P]\!]_\mu = \{h \mid \mu, h \models P\})$$

Trick : Resource Kripke Semantics

$$\eta, \mu, w \ \models \ \varphi$$

- η gives the meaning of k and μ the meaning of x.
- Resource Kripke world:

$$w \in \mathcal{P}(\mathsf{Heaps}), \qquad w \sqsubseteq w' \ \text{ iff } \ \exists w_0.\, w * w_0 = w'.$$

World keeps all the extensions.

emantics of intuitionistic logic:

w'. if $w' \sqsupseteq w$ and $(\eta, \mu, w' \models \varphi)$, then $\eta, \mu, w' \models \psi$

w_0. if $(\eta, \mu, w * w_0 \models \varphi)$, then $(\eta, \mu, w * w_0 \models \psi)$

$$\eta, \mu, w \ \models \varphi \otimes P \quad \text{iff} \quad \eta, \mu, (w * [\![P]\!]_\mu) \models \varphi$$

(where $[\![P]\!]_\mu = \{h \mid \mu, h \models P\}$)

Exercise 1: Prove the HO frame rule.
[Hint: Use the Kripke monotonicity.]
Exercise 2: Prove the distribution rule for $\Longrightarrow$.

T

$$\eta, \mu, w \;\models\; \varphi$$

- η gives the meaning of k and μ the meaning of x.
- Resource Kripke world:

$$w \in \mathcal{P}(\text{Heaps}), \qquad w \sqsubseteq w' \;\text{ iff }\; \exists w_0.\, w * w_0 = w'.$$

- Use standard semantics of intuitionistic logic:

$$\begin{array}{lll} \eta, \mu, w \;\models \varphi \Rightarrow \psi & \text{iff} & \forall w'.\text{ if } w' \sqsupseteq w \text{ and } (\eta, \mu, w' \models \varphi), \text{ then } \eta, \mu, w' \models \psi \\ & \text{iff} & \forall w_0.\text{ if } (\eta, \mu, w * w_0 \models \varphi), \text{ then } (\eta, \mu, w * w_0 \models \psi) \\ \eta, \mu, w \;\models \varphi \otimes P & \text{iff} & \eta, \mu, (w * [\![P]\!]_\mu) \models \varphi \end{array}$$

$$(\text{where } [\![P]\!]_\mu = \{h \mid \mu, h \models P\})$$

HW

- Define the semantics of triples.
- Prove the distribution rule holds for the triples.
- Prove basic rules, such as the rule for free(E).

Recipe for Adding HO Frame Rules

1. Pick resource Kripke worlds.
   1) Select a monoid $(W, 0, +)$
   2) Resource order on W: $w \sqsubseteq w'$ iff $\exists w''.\ w' = w + w''$.
2. Externalize $+$ by introducing a connective to your logic.
3. Use the standard intuitionistic semantics.
4. Hack the meaning of basic constructs, such as triples.

Tomorrow

1. Higher-order Frame Rule.
2. General References.
3. Relational Reading of Sep. Logic.
4. Higher-order Sep. Logic.

1. Can we ignore internal data structures of malloc?
2. What about code pointers?
3. What if malloc is updated?

Tomorrow

1. Higher-order Frame Rule.
2. General References.
3. Relational Reading of Sep. Logic.
4. Higher-order Sep. Logic.

1. Can we ignore internal data structures of malloc?
2. What about code pointers?
3. What if malloc is updated?

Reference:
POPL04/Journal (O'Hearn, Yang, Reynolds)
FOSSACS07/Journal (Birkedal, Yang)