7. Information Hiding and Global Memory

John C. Reynolds
Carnegie Mellon University
March 23, 2009

©2009 John C. Reynolds
Some Changes

We will regard assertions as sets of states. Instead of writing

\[ s, h \models p. \]

we will write

\[ (s, h) \in p. \]

Also, we will assume that \( h = h_0 \cdot h_1 \) implies \( h_0 \perp h_1 \).
Some Changes We Won’t Make

Note that O’Hearn writes

\[ h_0 \neq h_1 \] instead of \( h_0 \perp h_1 \)

and

\[ h_0 \ast h_1 \] instead of \( h_0 \cdot h_1 \).

Also, he often writes

\[ e.i := e' \] instead of \( [e + i - 1] := e' \)

and

\[ v := e.i \] instead of \( v := [e + i - 1] \).

Also note that O’Hearn calls stores “stacks”.

___
Parameterless Procedures (with Global Variables)

A (nonrecursive) *parameterless procedure definition* is a command of the form

\[
\text{let } k_1 = C_1, \ldots, k_n = C_n \text{ in } C,
\]

where

- \( k_1, \ldots, k_n \) are distinct binding occurrences of procedure names, whose scope is \( C \).
- \( C_1, \ldots, C_n \) and \( C \) are commands.

Then a *procedure call* is a command of the form

\[
k,
\]

where \( k \) is a procedure name.
Hypotheses for Parameterless Procedures

A judgement (often called a hypothetical specification or sequent) has the form

$$\Gamma \vdash \{p\} C \{q\},$$

where the context $\Gamma$ is a sequence of hypotheses of the form

$$\{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n],$$

the $k_i$ are distinct procedure identifiers, and the $X_i$ are finite sets of variables (denoted by lists).
Rules for Parameterless Procedures with Globals

- Parameterless Procedures (PLPROC)

\[ \Gamma \vdash \{p_1\} C_1 \{q_1\} \]
\[ \vdots \]
\[ \Gamma \vdash \{p_n\} C_n \{q_n\} \]
\[ \Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\} \]
\[ \Gamma \vdash \{p\} \text{let } k_1 = C_1, \ldots, k_n = C_n \text{ in } C \{q\}, \]
where
- \(k_1, \ldots, k_n\) do not occur free in \(\Gamma\),
- \(C_i\) only modifies variables in \(X_i\).

- Parameterless Procedure Call (PLCALL)

\[ \Gamma, \{p\} k \{q\}[X] \vdash \{p\} k \{q\}. \]
The Hypothetical Frame Rule

- Hypothetical (Second-Order) Frame Rule (HFR)

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1, Y], \ldots,
\{p_n \ast r\} k_n \{q_n \ast r\}[X_n, Y] \vdash \{p \ast r\} C \{q \ast r\},
\]

where

- \(C\) does not modify variables in \(r\), except through using \(k_1, \ldots, k_n\),
- \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
\]

Note that, to obtain soundness, it will be necessary to place restrictions on this rule.
A Derived Rule

From an instance of (PLPROC):

\[
\Gamma \vdash \{p_1 \ast r\} C_1 \{q_1 \ast r\} \\
\vdots \\
\Gamma \vdash \{p_n \ast r\} C_n \{q_n \ast r\}
\]

\[
\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1, Y], \cdots, \\
\{p_n \ast r\} k_n \{q_n \ast r\}[X_n, Y] \vdash \{p \ast r\} C \{q \ast r\}
\]

\[\Gamma \vdash \{p \ast r\} \text{ let } k_1 = C_1, \cdots, k_n = C_n \text{ in } C \{q \ast r\},\]

where

- \(k_1, \ldots, k_n\) do not occur free in \(\Gamma\),
- \(C_i\) only modifies variables in \(X_i, Y\),

the hypothetical frame rule:

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1, Y], \cdots, \\
\{p_n \ast r\} k_n \{q_n \ast r\}[X_n, Y] \vdash \{p \ast r\} C \{q \ast r\},
\]

where

- \(C\) doesn’t modify variables in \(r\), except through using \(k_1, \ldots, k_n\),
- \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
\]
gives

- Modular Parameterless Procedure Rule (MODPLPROC)

\[
\Gamma \vdash \{p_1 \ast r\} C_1 \{q_1 \ast r\}
\]

\[
\vdots
\]

\[
\Gamma \vdash \{p_n \ast r\} C_n \{q_n \ast r\}
\]

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma \vdash \{p \ast r\} \text{let } k_1 = C_1, \cdots, k_n = C_n \text{ in } C \{q \ast r\},
\]

where there is a set \(Y\) of variables such that:

- \(k_1, \ldots, k_n\) do not occur free in \(\Gamma\),
- \(C_i\) only modifies variables in \(X_i, Y\),
- \(C\) does not modify variables in \(r\), except through using \(k_1, \ldots, k_n\),
- \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
\]
Modular Specification Format

An application of the modular parameterless procedure rule:

\[ \Gamma \vdash \{ p_1 \cdot r \} \ C_1 \ \{ q_1 \cdot r \} \]

\[ \vdots \]

\[ \Gamma \vdash \{ p_n \cdot r \} \ C_n \ \{ q_n \cdot r \} \]

\[ \Gamma, \{ p_1 \} \ k_1 \ \{ q_1 \}[X_1], \cdots, \{ p_n \} \ k_n \ \{ q_n \}[X_n] \vdash \{ p \} \ C \ \{ q \} \]

\[ \Gamma \vdash \{ p \cdot r \} \ \text{let} \ k_1 = C_1, \cdots, k_n = C_n \ \text{in} \ C \ \{ q \cdot r \}, \]

where there is a set \( Y \) of variables such that . . . .

determines the *modular specification format*:

- **Interface Specification**:
  \[ \{ p_1 \} \ k_1 \ \{ q_1 \}[X_1], \cdots, \{ p_n \} \ k_n \ \{ q_n \}[X_n] \]

- **Resource Invariant**: \( r \)

- **Private Variables**: \( Y \)

- **Internal Implementations**: \( C_1, \ldots, C_n \). 

—
A Memory Manager

• Interface Specification:

\[
\{\text{emp}\} \text{ alloc } \{x \mapsto -, -\}[x]
\]
\[
\{x \mapsto -, -\} \text{ free } \{\text{emp}\}[\]
\]

• Resource Invariant:

\[
\text{anonlist } f \overset{\text{def}}{=} \exists \alpha. \text{ list } \alpha \ f
\]
\[
\iff (f = \text{nil} \land \text{emp}) \lor (\exists g. f \mapsto -, g \ast \text{anonlist } g)
\]

• Private Variables: f

• Internal Implementations:

\[
\text{if } f = \text{nil} \text{ then } x := \text{cons}(-, -)
\]
\[
\text{else } (x := f ; f := [x + 1]) \quad \text{(code for alloc)}
\]
\[
[x + 1] := f ; f := x \quad \text{(code for free)}
\]
A Proof of the Implementation of alloc

\{\texttt{emp} \; \ast \; \texttt{anonlist} \; f\}
\{\texttt{anonlist} \; f\}
if \; f = \texttt{nil} \; then
   \{\texttt{anonlist} \; f\}
   x := \texttt{cons}(\; - , \; - )
   \{x \mapsto - , - \; \ast \; \texttt{anonlist} \; f\}
else
   \left( \{\exists g. \; f \mapsto - , g \; \ast \; \texttt{anonlist} \; g\} \right)
   x := f ;
   \{\exists g. \; x \mapsto - , g \; \ast \; \texttt{anonlist} \; g\}
   f := [x + 1]
   \{x \mapsto - , f \; \ast \; \texttt{anonlist} \; f\}
\{x \mapsto - , - \; \ast \; \texttt{anonlist} \; f\}


A Proof of the Implementation of \texttt{free}

\{x \mapsto -, - * \texttt{anonlist f}\}
\[x + 1] := f ;
\{x \mapsto -, f * \texttt{anonlist f}\}
\{\exists g. x \mapsto -, g * \texttt{anonlist g}\}
f := x
\{\exists g. f \mapsto -, g * \texttt{anonlist g}\}
\{\texttt{anonlist f}\}
\{\texttt{emp} * \texttt{anonlist f}\}

—
Using alloc to insert an element in a list

\[
\begin{align*}
\{\text{emp}\} & \quad \text{alloc} \quad \{x \mapsto -, -\}[x], \{x \mapsto -, -\} \quad \text{free} \quad \{\text{emp}\}[] \vdash \\
\{y \mapsto a, z \mapsto c, w\} & \\
\{\text{emp}\} & \quad \text{alloc} ; \\
\{x \mapsto -, -\} & \\
[x] := b ; [x + 1] := z ; \\
\{x \mapsto b, z\} & \quad \{y \mapsto a, x \mapsto b, z \mapsto c, w\} \\
[y + 1] := x & \quad \{y \mapsto a, x \mapsto b, z \mapsto c, w\}
\end{align*}
\]
Using `free` to delete an element in a list

\[
\{\text{emp}\} \text{ alloc } \{x \mapsto -, -\}[x], \{x \mapsto -, -\} \text{ free } \{\text{emp}\}[] \vdash \\
\{y \mapsto a, x \mapsto b, z \mapsto c, w\} \\
\{x \mapsto -, -\} \quad \text{free ; } \quad \{\text{emp}\} \quad \{y \mapsto a, x \mapsto b, z \mapsto c, w\} \\
[y + 1] := z \\
\{y \mapsto a, z \mapsto c, w\}
\]
Some Erroneous Programs

```plaintext
{emp} alloc \{x \mapsto -, -\}[x], \{x \mapsto -, -\} free \{emp\}[] ⊢
{y \mapsto a, x \ast x \mapsto b, z \ast z \mapsto c, w}
{x \mapsto -, -}
free;
{emp}
{y \mapsto a, x \ast z \mapsto c, w}
y := [x + 1]
{???

{emp} alloc \{x \mapsto -, -\}[x], \{x \mapsto -, -\} free \{emp\}[] ⊢
{emp}
alloc;
{x \mapsto -, -}
free;
{emp}
[x + 1] := x
{???
```

---
A Generalization of List Segments

\[ glseg \varepsilon (i, j) \overset{\text{def}}{=} \text{emp} \land i = j \]
\[ glseg [a] \cdot \alpha (i, k) \overset{\text{def}}{=} \exists j. \ P(a) \ast i \mapsto a, j \ast glseg \alpha (j, k) \]

Properties:

\[ glseg [a] (i, j) \iff P(a) \ast i \mapsto a, j \]
\[ glseg \alpha \cdot \beta (i, k) \iff \exists j. \ glseg \alpha (i, j) \ast glseg \beta (j, k) \]
\[ glseg \alpha \cdot [b] (i, k) \iff \exists j. \ glseg \alpha (i, j) \ast P(b) \ast j \mapsto b, k. \]

Here \( P(a) \) is “parametric” predicate intended to describe the storage used by the list element \( a \). Possible instantiations include:

- \( P(a) = \text{emp} \): A list element is a plain value requiring no heap storage.
- \( P(a) = a \mapsto - , - \): A list element is a two-cell.
- \( P(a) = \text{anonlist} a \): A list element is itself a list.
A Queue Module

- Interface Specification:
  \[
  \{ \alpha = \alpha_0 \land a = a_0 \land P(a) \} \text{ enq } \{ \alpha = \alpha_0 \cdot [a] \land \text{emp} \}[\alpha] \\
  \{ \alpha = [a] \cdot \alpha_0 \land \text{emp} \} \text{ deq } \{ \alpha = \alpha_0 \land a = a_0 \land P(a) \}[\alpha, a] \\
  \{ \text{emp} \} \text{ isempty? } \{(w \leftrightarrow (\alpha = \epsilon)) \land \text{emp} \}[w]
  \]

- Resource Invariant:
  \[
  \text{glseg } \alpha \ (x, y) \ast y \mapsto \_, 
  \]
  (Note that \( \alpha \) appears in \( X_1, X_2 \) and the resource invariant.)

- Private Variables: \( x, y \)

- Internal Implementations:
  \[
  \alpha := \alpha \cdot [a] ;
  \]
  \[
  \text{newvar } t \text{ in}
  \]
  \[
  (t := \text{cons}(\_, \_); [y] := a ;
  \]
  \[
  [y + 1] := t ; y := t)
  \]
  \[
  \alpha := \text{rest } \alpha ;
  \]
  \[
  \text{newvar } t \text{ in}
  \]
  \[
  (t := x ; a := [t] ; x := [t + 1] ;
  \]
  \[
  \text{dispose } t ; \text{dispose } t + 1)
  \]
  \[
  w := (x = y) \quad (\text{code for isempty?})
  \]
A Proof of the Implementation of \texttt{enq}

\[
\{(\alpha = \alpha_0 \land a = a_0 \land P(a)) \ast (\text{glseg} \alpha (x, y) \ast y \mapsto -, -)\}
\]

\[
\alpha := \alpha \cdot [a] ;
\]

\[
\{\alpha = \alpha_0 \cdot [a_0] \land P(a)) \ast (\text{glseg} \alpha_0 (x, y) \ast y \mapsto -, -)\}
\]

newvar \ t in

\[
\left(\{\text{glseg} \alpha_0 (x, y) \ast P(a) \ast y \mapsto -, -\}
\right)
\]

\[
t := \text{cons}(-, -) ;
\]

\[
\{\text{glseg} \alpha_0 (x, y) \ast P(a) \ast y \mapsto -, - \ast t \mapsto -, -\}
\]

\[
[y] := a ; [y + 1] := t ;
\]

\[
\{\text{glseg} \alpha_0 (x, y) \ast P(a) \ast y \mapsto a, t \ast t \mapsto -, -\}
\]

\[
\{\text{glseg} \alpha_0 \cdot [a] (x, t) \ast t \mapsto -, -\}
\]

\[
y := t
\]

\[
\{\text{glseg} \alpha_0 \cdot [a] (x, y) \ast y \mapsto -, -\}
\]

\[
\{(\alpha = \alpha_0 \cdot [a_0] \land \text{emp}) \ast (\text{glseg} \alpha_0 \cdot [a_0] (x, y) \ast y \mapsto -, -)\}
\]

\[
\{(\alpha = \alpha_0 \cdot [a_0] \land \text{emp}) \ast (\text{glseg} \alpha (x, y) \ast y \mapsto -, -)\}
\]
A Proof of the Implementation of \texttt{deq}

\[
\{(\alpha = [a_0] \cdot \alpha_0 \land \text{emp}) \ast (\text{glseg} \ \alpha (x, y) * y \mapsto -, -)\} \\
\alpha := \text{rest} \ \alpha; \\
\{(\alpha = \alpha_0 \land \text{emp}) \ast (\text{glseg} \ [a_0] \cdot \alpha_0 (x, y) * y \mapsto -, -)\} \\
\{(\text{glseg} \ [a_0] \cdot \alpha_0 (x, y) * (\alpha = \alpha_0 \land y \mapsto -, -)\} \\
\text{newvar} \ t \ \text{in} \\
\left(\text{\{glseg} \ [a_0] \cdot \alpha_0 (x, y)\}\right) \\
\quad t := x; \\
\quad \{\text{glseg} \ [a_0] \cdot \alpha_0 (t, y)\} \\
\quad \exists x. \ P(a_0) \ast t \mapsto a_0, x \ast \text{glseg} \ \alpha_0 (x, y) \\
\quad a := [t]; \ x := [t + 1]; \\
\quad \{(a = a_0 \land P(a)) \ast t \mapsto a_0, x \ast \text{glseg} \ \alpha_0 (x, y)\} \\
\quad \text{dispose} \ t; \ \text{dispose} \ t + 1 \\
\left(\{(a = a_0 \land P(a)) \ast \text{glseg} \ \alpha_0 (x, y)\}\right) \\
\left(\{(a = a_0 \land a = a_0 \land P(a)) \ast (\text{glseg} \ \alpha (x, y) * y \mapsto -, -)\}\right)
\]
Ownership is in the Eye of the Asserter

Suppose \( P(a) \) is \( \text{emp} \). Then, in the calling program, one can write

\[
\begin{align*}
\{ \alpha = \alpha_0 \land \text{emp} \} \\
\text{a := cons}(-, -) ; \\
\alpha_0 := \text{a} ; \\
\{ \alpha = \alpha_0 \land \text{a} = \alpha_0 \land P(a) \} \\
\text{enq} ; \\
\{ \alpha = \alpha_0 \cdot \text{[a]} \land \text{emp} \} \\
[\text{a}] := 42 \\
\{ (\alpha = \alpha_0 \cdot \text{[a]} \land \text{emp}) \land \text{a} \rightarrow 42, - \}
\end{align*}
\]

But

\[
\begin{align*}
\{ \alpha = [\text{a}] \cdot \alpha_0 \land \text{emp} \} \\
\text{deq} ; \\
\{ \alpha = \alpha_0 \land \text{a} = \alpha_0 \land P(a) \} \\
[\text{a}] := 42 \\
\{ \ldots \}
\end{align*}
\]
Ownership is in the Eye of the Asserter (continued)

Suppose \( P(a) \) is a \( \mapsto -,- \). Then, in the calling program, one can write

\[
\begin{align*}
\{ \alpha &= \alpha_0 \land \text{emp} \} \\
& a := \text{cons}(-,-) ; \\
& a_0 := a ; \\
& \{ \alpha = \alpha_0 \land a = a_0 \land P(a) \} \\
& \text{enq} ; \\
& \{ \alpha = \alpha_0 \cdot [a_0] \land \text{emp} \} \\
& [a] := 42 \\
& \{ ?? ?? \}
\end{align*}
\]

But

\[
\begin{align*}
\{ \alpha &= [a_0] \cdot \alpha_0 \land \text{emp} \} \\
& \text{deq} ; \\
& \{ \alpha = \alpha_0 \land a = a_0 \land P(a) \} \\
& [a] := 42 \\
& \{ \alpha = \alpha_0 \land a = a_0 \mapsto 42,- \}
\end{align*}
\]

However, \texttt{enq} and \texttt{deq} themselves are independent of \( P \).
A Worry

\[ \{ \alpha = \alpha_0 \land a = a_0 \land P(a) \} \text{ enq } \{ \alpha = \alpha_0 \cdot [a_0] \land \text{emp} \}^{[\alpha]} \]

is satisfied by

\[
\alpha := \alpha \cdot [a] ;
\]

\[
\text{newvar } t \text{ in }
\]

\[
(t := \text{cons}(-, -) ; [y] := a ; [y + 1] := t ; y := t).
\]

But the hypothesis does not specify that \(\alpha_0\) and \(a_0\) are ghost variables. Thus it is also satisfied by

\[
\alpha := \alpha_0 \cdot [a_0] ;
\]

\[
\text{newvar } t \text{ in }
\]

\[
(t := \text{cons}(-, -) ; [y] := a_0 ; [y + 1] := t ; y := t).
\]

Because of this, we cannot infer

\[
\{ \exists \alpha_0, a_0. \alpha = \alpha_0 \land a = a_0 \land P(a) \} \text{ enq } \{ \exists \alpha_0, a_0. \alpha = \alpha_0 \cdot [a_0] \land \text{emp} \}
\]

nor

\[
\{ \alpha = 7 \cdot 8 \land a = 9 \land P(a) \} \text{ enq } \{ \alpha = 7 \cdot 8 \cdot 9 \land \text{emp} \}.
\]

(See Section 10.1 in paper.)
A Conundrum

Let \( \text{one} \overset{\text{def}}{=} \exists x. x \mapsto \bot \). Then

\[
\begin{align*}
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{emp} \lor \text{one}\} k \{\text{emp}\} \\
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{emp}\} k \{\text{emp}\} \\
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{emp} \ast \text{one}\} k \{\text{emp} \ast \text{one}\} \quad (1) \\
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{one}\} k \{\text{one}\} \\
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{one}\} k \{\text{emp} \land \text{one}\} \\
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] & \vdash \{\text{one}\} k \{\text{false}\} \\
\{(\text{emp} \lor \text{one}) \ast \text{true}\} k \{\text{emp} \ast \text{true}\}[] & \vdash \{\text{one} \ast \text{true}\} k \{\text{false} \ast \text{true}\} \quad (3) \\
\{(\text{emp} \lor \text{one}) \ast \text{true}\} k \{\text{emp} \ast \text{true}\}[] & \vdash \{\text{one} \ast \text{true}\} k \{\text{false}\}
\end{align*}
\]

1. by frame rule
2. by conjunction rule
3. by hypothetical frame rule

But, for example,

\[
\{(\text{emp} \lor \text{one}) \ast \text{true}\} \text{ skip } \{\text{emp} \ast \text{true}\},
\]

is valid, while

\[
\{\text{one} \ast \text{true}\} \text{ skip } \{\text{false}\}
\]

is not.
Where to Place the Blame

- The specification \( \{\text{emp} \lor \text{one}\} \rightarrow \{\text{emp}\} \).
  No command in our programming language satisfies this specification (nor any specification whose precondition is imprecise and whose postcondition is precise).
  We will see that, by requiring preconditions to be precise, we can insure soundness.

- The invariant \text{true}.
  Intuitively, a resource invariant should precisely identify an unambiguous area of storage, that owned by the module.
  We will see that, by requiring resource invariants to be precise, we can insure soundness.

- The rule of conjunction (or perhaps strengthening precedent, weakening consequent, or the ordinary frame rule).
  Birkedal has investigated banning the rule of conjunction.
A Denotational Semantics for Partial Correctness

Stores = Variables → Values
Heaps = \( \bigcup \limits_{A \subseteq \text{Addresses}}^{\text{fin}} (A \rightarrow \text{Values}) \)
States = Stores \( \times \) Heaps

Note that we do not index a store by a finite set of variables.

The meaning of a command (evaluated in an environment) is a binary relation in

\[ \text{States} \leftrightarrow (\text{States} \cup \{ \text{fault} \}), \]

i.e., between initial states and outcomes, which are either final states or fault (i.e., abort).

This does not distinguish indeterminate nontermination. For example,

\[
\text{newvar } a, b \ \text{in} \\
(a = \text{cons}(0) \ ; \ \text{dispose } a \ ; \ b = \text{cons}(0) \ ; \ \text{dispose } b \ ; \\
\text{if } a = b \text{ then while true do skip else } x := 0)
\]

has the same meaning as \( x := 0 \).
Safety Monotonicity and the Frame Property

We say that a relation \( c \in \text{States} \leftrightarrow (\text{States} \cup \{\text{fault}\}) \) is safe at a state \((s, h)\) when \((s, h)[c]\text{fault}\) is false.

Then our programming language satisfies two “locality” properties:

**Safety Monotonicity** For all states \((s, h)\) and heaps \(h_1\) such that \(h \sqsubseteq h_1\), if \(c\) is safe at \((s, h)\), it is also safe at \((s, h \cdot h_1)\).

**The Frame Property** For all states \((s, h)\) and heaps \(h_1\) such that \(h \sqsubseteq h_1\), if \(c\) is safe at \((s, h)\) and also \((s, h \cdot h_1)[c](s', h')\),

\[ s, h \sqsubseteq s, h \cdot h_1 \supseteq h_1 \]
\[ c \text{ safe} \]
\[ \downarrow \]
\[ s', h' \]

then there is a subheap \(h'_0 \subseteq h'\) such that

\( h'_0 \sqsubseteq h_1, h'_0 \cdot h_1 = h', \) and \((s, h)[c](s', h'_0)\).

\[ s, h \sqsubseteq s, h \cdot h_1 \supseteq h_1 \]
\[ c \]
\[ \downarrow \]
\[ c \]
\[ \sqcup \supseteq \]
\[ s', h'_0 \sqsubseteq s', h' = s', h'_0 \cdot h_1 \]
The Domain LRel

We define the poset $LRel$ of local relations to be the set

\[
LRel = \{ c : \text{States} \leftrightarrow (\text{States} \cup \{\text{fault}\}) \mid c \text{ satisfies safety monotonicity and the frame property} \},
\]

ordered by subset inclusion. Then:

**Proposition 16**  The poset $LRel$ is a chain-complete partial order with a least element. The least element is the empty relation, and the least upper bound of a chain is given by the union of all the relations in the chain.
The General Semantics

\[ \eta \in \text{Env} = \text{ProcIds} \rightarrow \text{LRel} \quad \text{(ordered pointwise)} \]

\[ \llbracket C \rrbracket \in \text{Env} \xrightarrow{\text{cont}} \text{LRel} \]

\[ (s, h)[\llbracket A \rrbracket \eta]a \Leftrightarrow (s, h)[\overline{A}]a \quad (\overline{A} \in \text{LRel}) \]

\[ (s, h)[\llbracket \text{skip} \rrbracket \eta]a \Leftrightarrow a = (s, h) \]

\[ (s, h)[\llbracket C_1 ; C_2 \rrbracket \eta]a \Leftrightarrow (s, h)[\text{seq}(\llbracket C_1 \rrbracket \eta, \llbracket C_2 \rrbracket \eta)]a \]

\[ (s, h)[\llbracket \text{if} \ B \ \text{then} \ C_1 \ \text{else} \ C_2 \rrbracket \eta]a \]
\[ \Leftrightarrow (s, h)[\llbracket B \rrbracket \leadsto \llbracket C_1 \rrbracket \eta \ ; \ \llbracket C_2 \rrbracket \eta]a \]

\[ (s, h)[\llbracket \text{while} \ B \ \text{do} \ C \rrbracket \eta]a \Leftrightarrow \]

\[ (s, h)[\text{fix}(\lambda c \in \text{LRel.} \ (\llbracket B \rrbracket \leadsto \text{seq}(\llbracket C \rrbracket \eta, c) ; \ \llbracket \text{skip} \rrbracket \eta))]a \]

\[ (s, h)[\llbracket k \rrbracket \eta]a \Leftrightarrow (s, h)[\eta(k)]a \]

\[ (s, h)[\llbracket \text{let} \ k_1 = C_1, \ldots, k_n = C_n \ \text{in} \ C \rrbracket \eta]a \Leftrightarrow \]

\[ (s, h)[\llbracket C \rrbracket [\eta \mid k_1 : \llbracket C_1 \rrbracket \eta \mid \cdots \mid k_n : \llbracket C_n \rrbracket \eta]]a \]

where

\[ (s, h)[\text{seq}(c_1, c_2)]a \Leftrightarrow \]

\[ \exists (s', h'). ((s, h)[c_1](s', h') \land (s', h')[c_2]a) \cup ((s, h)[c_1]\text{fault} \land a = \text{fault}) \]

\[ (s, h)[b \leadsto c_1 ; c_2]a \Leftrightarrow \]

\[ \text{if } b(s) \ \text{then} \ (s, h)[c_1]a \ \text{else} \ (s, h)[c_2]a \]

—
Properties of the Semantics

For the general semantics:

**Proposition 17**  For each command $C$, $[[C]]$ is well-defined: For all environments $\eta$, $[[C]]\eta$ is in $\text{LRel}$, and $[[C]]$ is a continuous function from $\text{Env}$ to $\text{LRel}$.

For the RAM semantics:

**Proposition 18**  For each basic command $A$, we have $\overline{A} \in \text{LRel}$. 

—
Semantics of Sequents

To interpret sequents we define semantic cousins of the modifies clauses and Hoare triples. If $c \in LRel$ then

- $\text{modifies}(c, X)$ holds iff, whenever $(s, h)[c](s', h')$ and $y \notin X$, we have that $s(y) = s'(y)$.

- $\{p\} c \{q\}$ holds iff, for all states $(s, h) \in p$:
  1. $(s, h)[c]\text{fault}$ is false, and
  2. for all states $(s', h')$, if $(s, h)[c](s', h')$, then $(s', h') \in q$.

Then a sequent

$\{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}$

holds iff, for all environments $\eta$, if both

$\{p_i\} \eta(k_i) \{q_i\}$ and $\text{modifies}(\eta(k_i), X_i)$

hold for all $1 \leq i \leq n$, then

$\{p\} (\llbracket C \rrbracket_\eta) \{q\}$

also holds.
The Conundrum Revisited

The standard semantics validates the conjunction rule (CONJ):

\[
\frac{\Gamma \vdash \{p_1\} C \{q_1\} \quad \Gamma \vdash \{p_2\} C \{q_2\}}{
\Gamma \vdash \{p_1 \land p_2\} C \{q_1 \land q_2\}
}\]

and so also the instance of it:

\[
\frac{\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] \vdash \{	ext{one}\} k \{	ext{one}\} \quad \{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] \vdash \{	ext{one}\} k \{\text{emp} \land \text{one}\}}{
\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] \vdash \{	ext{one}\} k \{\text{emp} \land \text{one}\},
}\]

(The conclusion here specifies that any \(c \in \text{LRel}\) satisfying \(\{\text{emp} \lor \text{one}\} c \{\text{emp}\}\) must diverge when started in a state satisfying \(\text{one}\).)

On the other hand, the instance

\[
\frac{\{\text{emp} \lor \text{one}\} k \{\text{emp}\}[] \vdash \{	ext{one}\} k \{\text{false}\} \quad \{(\text{emp} \lor \text{one}) \ast \text{true}\} k \{\text{emp} \ast \text{true}\}[] \vdash \{	ext{one} \ast \text{true}\} k \{\text{false} \ast \text{true}\}}{
\}
\]

of the hypothetical frame rule is invalidated, since its conclusion is false when \(\eta(k)\) is the identity relation (the meaning of \text{skip}).

—
Precise Predicates

A predicate \( p \) is precise iff, for all states \((s, h)\), there is at most one subheap \( h_p \subseteq h \) for which \((s, h_p) \in p\).

**Proposition 19** A predicate \( p \) is precise iff, for all predicates \( q \) and \( r \),

\[
p \ast (q \land r) \iff (p \ast q) \land (p \ast r).
\] (23)

**Proof** (if-direction only) Suppose \( p \) is not precise. Then there are distinct states \((s, h_p) \in p\) and \((s, h_p') \in p\), such that \( h_p \) and \( h_p' \) are subheaps of \( h \).

Let \( q = \{(s, h - h_p)\} \) and \( r = \{(s, h - h_p')\} \). Then \((s, h) \in (p \ast q) \land (p \ast r)\). But \((s, h) \notin p \ast (q \land r)\) since \( q \land r \) is empty. Thus (23) is false.

**Proposition 20** If \( p \) and \( q \) are precise and \( B \) is pure, then \( p \land r \), \( p \ast q \), and \((B \land p) \lor ((\neg B) \land q)\) are precise.
Disjointness in the Hypothetical Frame Rule

\[
\Gamma, \{p_1\} \ k_1 \ \{q_1\}[X_1], \ldots, \{p_n\} \ k_n \ \{q_n\}[X_n] \vdash \{p\} \ C \ \{q\}
\]

\[
\Gamma, \{p_1 \ast r\} \ k_1 \ \{q_1 \ast r\}[X_1, Y], \ldots,
\]

\[
\{p_n \ast r\} \ k_n \ \{q_n \ast r\}[X_n, Y] \vdash \{p \ast r\} \ C \ \{q \ast r\},
\]

where

- \(C\) does not modify variables in \(r\), except through using \(k_1, \ldots, k_n\),
- \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} \ k_1 \ \{q_1\}[X_1], \ldots, \{p_n\} \ k_n \ \{q_n\}[X_n] \vdash \{p\} \ C \ \{q\}.
\]

Here “disjoint” is defined by:

- \(Y\) is disjoint from \(X\) iff \(Y \cap X\) is empty.
- \(Y\) is disjoint from \(C\) iff \(Y \cap \text{FV}(C)\) is empty.
- \(Y\) is disjoint from \(p\) iff, for all \(s, s', h\),

  if \(s(x) = s'(x)\) for all \(x \not\in Y\) then \((s, h) \in p \iff (s', h) \in p\).
- \(Y\) is disjoint from \(\Gamma' \vdash \{p\} \ C \ \{q\}\) iff it is disjoint from \(p\), \(q\), and \(C\), and from \(p_i, q_i\), and \(X_i\) in each hypothesis in \(\Gamma'\).
The “Modifies” Clause

- $C$ does not modify variables in $r$, except through using $k_1, \ldots, k_n$, means $\text{Modifies}(C)(\Gamma)$ is disjoint from $r$, where

\[
\begin{align*}
\text{Modifies}(k)(\Gamma) &= X & \text{if } \{p\} k \{q\}[X] \in \Gamma \\
\text{Modifies}(k)(\Gamma) &= \{} & \text{otherwise} \\
\text{Modifies}(x := E)(\Gamma) &= \{x\} \\
\text{Modifies}([E] := E')(\Gamma) &= \{} \\
\text{Modifies}(F(C_1, \ldots, C_n))(\Gamma) &= \text{Modifies}(C_1)(\Gamma) \cup \cdots \cup \text{Modifies}(C_n)(\Gamma)
\end{align*}
\]
Decomposing the Hypothetical Frame Rule

The hypothetical frame rule

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma, \{p_1 * r\} k_1 \{q_1 * r\}[X_1, Y], \ldots,
\]

\[
\{p_n * r\} k_n \{q_n * r\}[X_n, Y] \vdash \{p * r\} C \{q * r\},
\]

where

- \text{Modifies}(C)(\Gamma) is disjoint from \(r\),
- \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
\]

has two special cases:

- **Modifies Weakening**

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1, Y], \ldots,
\]

\[
\{p_n \ast r\} k_n \{q_n \ast r\}[X_n, Y] \vdash \{p \ast r\} C \{q \ast r\},
\]

where \(Y\) is disjoint from the judgement

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
\]

- **Simple Hypothetical Frame Rule**

\[
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}
\]

\[
\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1], \ldots,
\]

\[
\{p_n \ast r\} k_n \{q_n \ast r\}[X_n] \vdash \{p \ast r\} C \{q \ast r\},
\]

where

- \text{Modifies}(C')(\Gamma) is disjoint from \(r\),
- For all \(\{p'\} k \{q'\}[X]\) in \(\Gamma\), the set \(X\) is disjoint from \(r\).
Regaining the Hypothetical Frame Rule from the Special Cases

Suppose

\[ \Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} \ C \ \{q\} \]

where

- \( \text{Modifies}(C)(\Gamma) \) is disjoint from \( r \),
- \( Y \) is disjoint from the judgement

\[ \Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} \ C \ \{q\} \].
Regaining the Hypothetical Frame Rule from the Special Cases (continued)

Let $\Gamma_0$ be obtained from $\Gamma$ by retaining only the hypotheses $\{p'_k \} k \{q'_k\}[X]$ for which $k$ occurs in $C$. Then

$$
\frac{\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}{
\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}
$$

where we can replace $\Gamma$ by $\Gamma_0$ in the side conditions:

- $\text{Modifies}(C')(\Gamma_0)$ is disjoint from $r$,
- $Y$ is disjoint from the judgement

$$
\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}.
$$

The second side condition allows us to use Modifies Weakening.

—
Regaining the Hypothetical Frame Rule from the Special Cases (continued)

The second side condition allows us to use Modifies Weakening:

\[
\begin{align*}
\Gamma, \{p_1\} & \quad k_1 \quad \{q_1\}[X_1], \ldots, \{p_n\} & \quad k_n \quad \{q_n\}[X_n] \vdash \{p\} & \quad C \quad \{q\} \\
\Gamma_0, \{p_1\} & \quad k_1 \quad \{q_1\}[X_1], \ldots, \{p_n\} & \quad k_n \quad \{q_n\}[X_n] \vdash \{p\} & \quad C \quad \{q\} \\
\Gamma_0, \{p_1\} & \quad k_1 \quad \{q_1\}[X_1, Y], \ldots, \{p_n\} & \quad k_n \quad \{q_n\}[X_n, Y] \vdash \{p\} & \quad C \quad \{q\}
\end{align*}
\]

where we still have:

- \(\text{Modifies}(C')(\Gamma_0)\) is disjoint from \(r\).

But, since every hypothesis in \(\Gamma_0\) occurs in \(C\), this condition implies

- For all \(\{p'\} \quad k \quad \{q'\}[X]\) in \(\Gamma_0\), the set \(X\) is disjoint from \(r\).

So we can use the Simple Hypothetical Frame Rule.
Regaining the Hypothetical Frame Rule from the Special Cases (continued)

So we can use the Simple Hypothetical Frame Rule:

\[
\frac{\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}{\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}
\]

\[
\frac{\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1, Y], \cdots, \{p_n\} k_n \{q_n\}[X_n, Y] \vdash \{p\} C \{q\}}{\Gamma_0, \{p_1 * r\} k_1 \{q_1 * r\}[X_1, Y], \cdots, \{p_n * r\} k_n \{q_n * r\}[X_n, Y] \vdash \{p * r\} C \{q * r\}}
\]
Regaining the Hypothetical Frame Rule from the Special Cases (continued)

Finally, since $\Gamma_0$ is a subcontext of $\Gamma$:

$$
\begin{array}{c}
\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\} \\
\hline
\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1], \cdots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\} \\
\hline
\Gamma_0, \{p_1\} k_1 \{q_1\}[X_1, Y], \cdots, \{p_n\} k_n \{q_n\}[X_n, Y] \vdash \{p\} C \{q\} \\
\Gamma_0, \{p_1 r\} k_1 \{q_1 r\}[X_1, Y], \cdots, \{p_n r\} k_n \{q_n r\}[X_n, Y] \vdash \{p r\} C \{q r\} \\
\hline
\Gamma, \{p_1 r\} k_1 \{q_1 r\}[X_1, Y], \cdots, \{p_n r\} k_n \{q_n r\}[X_n, Y] \vdash \{p r\} C \{q r\}.
\end{array}
$$
The Greatest Relation

For a specification \( \{ p \} - \{ q \} [X] \), we consider the greatest local relation \( \text{great}(p, q, X) \in \text{LRel} \) satisfying both

\[
\{ p \} \text{ great}(p, q, X) \{ q \} \quad \text{and} \quad \text{modifies( great}(p, q, X), X).
\]

This relation can be defined explicitly as follows:

- \((s, h)[\text{great}(p, q, X)]\) \(\text{fault} \iff (s, h) \notin p \ast \text{true}\)
- \((s, h)[\text{great}(p, q, X)](s', h') \iff\)
  1. \(s(y) = s'(y)\) for all variables \(y \notin X\); and
  2. For all \(h_P, h_1\),

\[
(h_P \cdot h_1 = h \land (s, h_P) \in p) \Rightarrow 
(\exists h_q. h_q' \cdot h_1 = h' \land (s', h_q') \in q).
\]

**Proposition 21** \( \text{great}(p, q, X) \in \text{LRel} \).

**Proposition 22**

- \( \{ p \} \text{ great}(p, q, X) \{ q \} \),
- \( \text{modifies( great}(p, q, X), X) \),
- \( \text{For all } c \in \text{LRel}, \)

\[
\{ p \} c \{ q \} \text{ and modifies}(c, X) \Rightarrow c \subseteq \text{great}(p, q, X).
\]
The Greatest Relation of All

We define the relation
\[ c_{\top} \overset{\text{def}}{=} \text{States} \times (\text{States} \cup \{\text{fault}\}), \]
which relates every state to every outcome. It is vacuously local, since it is not safe at any state. Thus it is the greatest relation in LRel.

The Greatest Environment

The greatest environment \( \eta \) satisfying the context
\[ \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \]
- maps each \( k_i \) into \( \text{great}(p_i, q_i, X_i) \),
- maps all other procedure identifiers into \( c_{\top} \).

**Proposition 23** A sequent \( \Gamma \vdash \{p\} C \{q\} \) holds iff the triple \( \{p\} ([C]\eta) \{q\} \) holds for the greatest environment \( \eta \) satisfying \( \Gamma \).

**Proposition 24** For all \( p, q, p', q', C, \Gamma, \) and \( \Gamma' \), the proof rule
\[
\Gamma \vdash \{p\} C \{q\} \\
\Gamma' \vdash \{p'\} C \{q'\}
\]
is sound iff
\[
\{p\} ([C]\eta) \{q\} \Rightarrow \{p'\} ([C]\eta') \{q'\} \]
holds for the greatest environments \( \eta \) and \( \eta' \) that satisfy \( \Gamma \) and \( \Gamma' \).
Simulation Between Command Meanings

Let $R : \text{States} \leftrightarrow \text{States}$ and $c, c' \in \text{LRel}$. We say that

$$c \text{ simulates } c' \text{ upto } R,$$

written

$$c[\text{sim}(R)]c',$$

iff:

- **Generalized Safety Monotonicity** If $c$ is safe at $(s, h)$, and $(s, h)[R](s', h')$, then $c'$ is safe at $(s'h')$.

- **Generalized Frame Property** If $c$ is safe at $(s, h)$, and we have $(s, h)[R](s', h')$ and $(s', h')[c'](s'_1, h'_1)$,

  $$s, h \xrightarrow{R} s', h'$$
  $$| \quad c \text{ safe} \quad | \quad c'$$
  $$\downarrow \quad s'_1, h'_1$$

  then there is a state $(s_1, h_1)$ such that $(s, h)[c](s_1, h_1)$ and $(s_1, h_1)[R](s'_1, h'_1)$.

$$s, h \xrightarrow{R} s', h'$$
$$\downarrow \quad c \quad | \quad c'$$
$$\downarrow \quad s_1, h_1 \xrightarrow{R} s'_1, h'_1$$
Simulation between Command Meanings (continued)

Let $R(p)$ denote the image of $p$ under $R$:

$$R(p) = \{ (s', h') \mid \exists (s, h) \in p. (s, h)[R](s', h') \}.$$ 

Then an alternative characterization of simulation is given by

**Proposition 25** $c[\text{sim}(R)]c'$ iff, for all predicates $p$ and $q$,

$$\{p\} c \{q\} \Rightarrow \{R(p)\} c' \{R(q)\}.$$ 

Note that $c\top[\text{sim}(R)]c'$ holds vacuously for all $R$ and $c'$, since $c\top$ is not safe at any state.

Simulation between Environments

Let $R : \text{States} \leftrightarrow \text{States}$ and $\eta$ and $\eta'$ be environments. We say that

$$\eta \text{ simulates } \eta' \text{ upto } R,$$

written

$$\eta[\text{sim}(R)]\eta',$$

iff $\eta(k)[\text{sim}(R)]\eta'(k)$ for all procedure names $k$.
Independence

We say that a command $C$ is independent of $R$ iff:

- The meaning of every basic command $A$ in $C$ simulates itself upto $R$:
  \[
  A \sim (R) A,
  \]
- The meaning of every boolean expression $B$ in $C$ maps $R$-related states to the same value:
  \[
  \forall (s, h), (s', h'). (s, h)[R](s', h') \Rightarrow [B]s = [B]s'.
  \]

**Proposition 26** If $C$ is independent of $R$, then

\[
\forall \eta, \eta'. \eta \sim (R) \eta' \Rightarrow ([C]\eta)[R]([C]\eta').
\]
Proposition 27  Let $\eta$ and $\eta'$ be the greatest environments satisfying $\Gamma$ and $\Gamma'$, and let $\mathcal{P}$ be a set of commands that are independent of $R$. Then $\eta \left[ \text{sim}(R) \right] \eta'$ implies that the proof rule

$$
\begin{align*}
\Gamma \vdash \{p\} & \quad C \quad \{q\} \\
\hline
\Gamma' \vdash \{R(p)\} & \quad C \quad \{R(q)\}
\end{align*}
$$

is sound for all $C \in \mathcal{P}$.

(Moreover, the converse holds if $\mathcal{P}$ contains the procedure call $k$ for every procedure name $k$.)
A Special Case (for Modifies Weakening)

For a set $Y$ of variables, the relation $R_Y$ relates states that differ only for variables in $Y$:

$$(s, h)[R_Y](s', h') \iff (h = h' \land \forall x \in \text{Variables. } x \notin Y \Rightarrow s(x) = s'(x)).$$

Note that

1. If $Y$ is disjoint from a predicate $p$, then $R_Y(p)$ is just $p$.
2. If $Y$ is disjoint from a command $C$, then $C$ is independent of $R_Y$. 

—
A Simulation Result for Greatest Relations

**Proposition 28** If $Y$ is disjoint from $p$, $q$, and $X$, and $Y_0 \subseteq Y$, then

$$\text{great}(p, q, X)[\text{sim}(R_Y)]\text{great}(p, q, X \cup Y_0).$$

**Proof** Suppose $(s, h)[R_Y](s_1, h_1)$ where $\text{great}(p, q, X)$ is safe at $(s, h)$. Then $h = h_1$ and $s$ and $s_1$ differ only at variables in $Y$. Also, by the definition of $\text{great}(p, q, X)$, $h$ can be split into $h_p \cdot h_0$ such that $(s, h_p) \in p$. Then $h_1$ is also $h_p \cdot h_0$ and, since $Y$ is disjoint from $p$, $(s_1, h_p) \in p$.

Thus generalized safety monotonicity holds, since the definition of $\text{great}(p, q, X \cup Y_0)$ shows that this local relation is safe at $(s_1, h_1)$.

To show the generalized frame property, we suppose that $(s_1, h_1)[\text{great}(p, q, X \cup Y_0)](s_1', h_1')$. Then $s_1$ and $s_1'$ differ only at variables in $X \cup Y_0$, and $h_1' = h_q' \cdot h_0$ for some $h_q'$ such that $(s_1', h_q') \in q$.

Now let $s'(z) = \text{if } z \in Y \text{ then } s(z) \text{ else } s_1'(z)$, and $h' = h_1'$. Then $(s', h')[R_Y](s_1', h_1')$, and since $Y$ is disjoint from $q$, $(s', h') \in q$. Moreover, if $z \notin X$, then $s'(z) = s(z)$, either by the definition of $s'(z)$ if $z \in Y$, or by $s'(z) = s_1'(z) = s_1(z) = s(z)$ if $z \notin Y$. From this it is easy to see $(s, h)[\text{great}(p, q, X)](s', h')$.

---

END OF PROOF
Soundness of Modifies Weakening

Consider

\[
\overline{\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}
\]

where \( Y \) is disjoint from the judgement \( \Gamma_1 \vdash \{p\} C \{q\} \).

Let \( \eta_1 \) and \( \eta_2 \) be the greatest environments satisfying \( \Gamma_1 \) and \( \Gamma_2 \). Each \( k \) that appears in \( \Gamma_1 \) (or equally well, \( \Gamma_2 \)) will be described by an hypothesis of the form \( \{p'\} k \{q'\}[X] \) in \( \Gamma_1 \), and by an hypothesis of either form \( \{p'\} k \{q'\}[X, Y] \) in \( \Gamma_2 \). Thus, \( \eta_1(k) \) will be \( \text{great}(p', q', X) \), and \( \eta_2(k) \) will be either \( \text{great}(p', q', X) \) or \( \text{great}(p', q', X \cup Y) \), both of which are \( \text{great}(p', q', X \cup Y_0) \) for some \( Y_0 \subseteq Y \). Then, since \( Y \) is disjoint from \( p', q', \) and \( X \), Proposition 28 gives \( \eta_1(k)[\text{sim}(R_Y)]\eta_2(k) \).

For those \( k \) that do not appear in \( \Gamma_1 \), \( \eta_1(k) = c_\top \) vacuously gives \( \eta_1(k)[\text{sim}(R_Y)]\eta_2(k) \). Thus \( \eta_1[\text{sim}(R_Y)]\eta_2 \).

Let \( \mathcal{P}_Y = \{ C' \mid Y \text{ is disjoint from } C' \} \), which contains \( C \) by the side-condition above. By Note 2, the commands in \( \mathcal{P}_Y \) are independent of \( R_Y \). Thus, by Proposition 27:

\[
\overline{\Gamma_1 \vdash \{p\} C \{q\}}
\]

Finally, since \( Y \) is disjoint from \( p \) and \( q \), Note 1 gives \( R_Y(p) = p \) and \( R_Y(q) = q \), so

\[
\overline{\Gamma_1 \vdash \{p\} C \{q\}}
\]

\[
\overline{\Gamma_2 \vdash \{p\} C \{q\}}
\]
Another Special Case (for the Simple Hypothetical Frame Rule)

For a predicate \( r \), the relation \( R_r \) relates \((s, h)\) and \((s', h')\) when we can obtain \((s', h')\) from \((s, h)\) by adding a new heap in \( r \):

\[
(s, h)[R_r](s', h') \Leftrightarrow 
(s = s' \land \exists h_1. h \perp h_1 \land h \cdot h_1 = h' \land (s, h_1) \in r).
\]

Note that:

3. The image \( R_r(p) \) of a predicate \( p \) is \( p \ast r \).

**Proposition 29** If \( c \in \text{LRel} \) satisfies \text{modifies}(c, X) and \( X \) is disjoint from \( r \), then \( c[\text{sim}(R_r)]c \).

Now suppose that \( \text{Modifies}(C)(\Gamma) \) is disjoint from \( r \). Then every basic command \( A \) in \( C \) modifies only variables that are disjoint from \( r \), so by Proposition 29, \( \overline{A}[\text{sim}(R_r)]\overline{A} \). Moreover, boolean expressions depend only upon the store, and \( R_r \)-related states contain the same store. Thus we note:

4. If \( \text{Modifies}(C)(\Gamma) \) is disjoint from \( r \), then \( C \) is independent of \( R_r \).
The Role of Precision

**Proposition 30**

(a) A predicate $p$ is precise (if and) only if

\[
great(p, q, X)[\text{sim}(R_r)]great(p \ast r, q \ast r, X)
\]

holds for all predicates $r$ and $q$, and sets $X$ of variables.

(b) A predicate $r$ is precise (if and) only if

\[
great(p, q, X)[\text{sim}(R_r)]great(p \ast r, q \ast r, X)
\]

holds for all predicates $p$ and $q$, and sets $X$ of variables.

Thus, if either $p$ or $r$ is precise then

\[
great(p, q, X)[\text{sim}(R_r)]great(p \ast r, q \ast r, X).
\]
Suppose either \( r \) or each of \( p_1, \ldots, p_n \) are precise, and consider

\[
\frac{\Gamma, \{p_1\} k_1 \{q_1\}[X_1], \ldots, \{p_n\} k_n \{q_n\}[X_n] \vdash \{p\} C \{q\}}{\Gamma, \{p_1 \ast r\} k_1 \{q_1 \ast r\}[X_1], \ldots, \{p_n \ast r\} k_n \{q_n \ast r\}[X_n] \vdash \{p \ast r\} C \{q \ast r\},}
\]

where

1. \( \text{Modifies}(C)(\Gamma) \) is disjoint from \( r \),
2. For all \( \{p'\} k \{q'\}[X] \) in \( \Gamma \), the set \( X \) is disjoint from \( r \).

Let \( \eta_1 \) and \( \eta_2 \) be the greatest environments satisfying \( \Gamma_1 \) and \( \Gamma_2 \). Each \( k \) that appears in \( \Gamma_1 \) (or equally well, \( \Gamma_2 \)) will be described by an hypothesis of the form \( \{p'\} k \{q'\}[X] \) in \( \Gamma_1 \), and by an hypothesis of either (1) the form \( \{p' \ast r\} k \{q' \ast r\}[X] \) or (2) the form \( \{p'\} k \{q'\}[X] \) in \( \Gamma_2 \).

Thus, \( \eta_1(k) \) will be \( \text{great}(p', q', X) \), and \( \eta_2(k) \) will be either (1) \( \text{great}(p' \ast r, q' \ast r, X) \) or (2) \( \text{great}(p', q', X) \). In the first case, Proposition 30 gives \( \eta_1(k)[\text{sim}(R_r)]\eta_2(k) \). In the second case, Proposition 29 gives the same consequence, since \( \text{great}(p', q', X) \) satisfies \( \text{modifies}(\text{great}(p', q', X), X) \), and \( X \) is disjoint from \( r \) by the second side-condition.

For those \( k \) that do not appear in \( \Gamma_1 \), \( \eta_1(k) = c_{\top} \) vacuously gives \( \eta_1(k)[\text{sim}(R_r)]\eta_2(k) \). Thus \( \eta_1[\text{sim}(R_r)]\eta_2 \).
Soundness of the Simple Hypothetical Frame Rule (continued)

Let \( \mathcal{P}_r = \{ C' \mid \text{Modifies}(C')(\Gamma) \text{ is disjoint from } r \} \), which contains \( C \) by the first side-condition. By Note 4, the commands in \( \mathcal{P}_r \) are independent of \( R_r \). Thus, by Proposition 27:

\[
\Gamma_1 \vdash \{p\} \ C \ \{q\} \\
\Gamma_2 \vdash \{R_Y(p)\} \ C \ \{R_Y(q)\}.
\]

Finally, Note 3 gives \( R_Y(p) = p * r \) and \( R_Y(q) = q * r \), so that

\[
\Gamma_1 \vdash \{p\} \ C \ \{q\} \\
\Gamma_2 \vdash \{p * r\} \ C \ \{q * r\}.
\]