Chapter 2

Assertions

In this chapter, we give a more detailed exposition of the assertions of separation logic: their meaning, illustrations of their usage, inference rules, and several classes of assertions with special properties.

First, we discuss some general properties that are shared among assertions and other phrases occurring in both our programming language and its logic. In all cases, variable binding behaves in the standard manner. In expressions there are no binding constructions, in assertions quantifiers are binding constructions, and in commands declarations are binding constructions. In each case, the scope of the variable being bound is the immediately following subphrase, except that in declarations of the form **newvar** $v = e$ **in** c, or iterated separating conjunctions of the form $\bigodot_{v=e_0}^{e_1} p$ (to be introduced later), the initialization e and the bounds e_0 and e_1 are not in the scope of the binder v.

We write $\mathrm{FV}(p)$ for the set of variables occurring free in p, which is defined in the standard way. The meaning of any phrase is independent of the value of those variables that do not occur free in the phrase.

Substitution is also defined in the standard way. We begin by considering *total* substitutions that act upon all the free variables of a phrase: For any phrase p such that $\mathrm{FV}(p) \subseteq \{v_1, \ldots, v_n\}$, we write

$$p/v_1 \to e_1, \ldots, v_n \to e_n$$

to denote the phrase obtained from p by simultaneously substituting each expression e_i for the variable v_i, (When there are bound variables in p, they will be renamed to avoid capture.)

When expressions are substituted for variables in an expression or assertion p, the effect mimics a change of the store. In the specific case where p is an expression:

Proposition 1 *(Total Substitution Law for Expressions) Let δ abbreviate the substitution*

$$v_1 \rightarrow e_1, \ldots, v_n \rightarrow e_n,$$

let s be a store such that $\mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \subseteq \mathrm{dom}\, s$, and let

$$\widehat{s} = [\, v_1 \colon [\![e_1]\!]_{\mathrm{exp}} s \mid \ldots \mid v_n \colon [\![e_n]\!]_{\mathrm{exp}} s \,].$$

If e is an expression (or boolean expression) such that $\mathrm{FV}(e) \subseteq \{v_1, \ldots, v_n\}$, then

$$[\![e/\delta]\!]_{\mathrm{exp}} s = [\![e]\!]_{\mathrm{exp}} \widehat{s}.$$

Here we have introduced a notation for describing stores (and more generally, functions with finite domains) by enumeration: We write $[\, x_1 \colon y_1 \mid \ldots \mid x_n \colon y_n \,]$ for the function with domain $\{x_1, \ldots, x_n\}$ that maps each x_i into y_i.

Next, we generalize this result to *partial* substitutions that need not act upon all the free variables of a phrase: When $\mathrm{FV}(p)$ is not a subset of $\{v_1, \ldots, v_n\}$,

$$p/v_1 \rightarrow e_1, \ldots, v_n \rightarrow e_n$$

abbreviates

$$p/v_1 \rightarrow e_1, \ldots, v_n \rightarrow e_n, v'_1 \rightarrow v'_1, \ldots, v'_k \rightarrow v'_k,$$

where $\{v'_1, \ldots, v'_k\} = \mathrm{FV}(p) - \{v_1, \ldots, v_n\}$. Then the above proposition can be generalized to

Proposition 2 *(Partial Substitution Law for Expressions) Suppose e is an expression (or boolean expression), and let δ abbreviate the substitution*

$$v_1 \rightarrow e_1, \ldots, v_n \rightarrow e_n,$$

Then let s be a store such that $(\mathrm{FV}(e) - \{v_1, \ldots, v_n\}) \cup \mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \subseteq \mathrm{dom}\, s$, and let

$$\widehat{s} = [\, s \mid v_1 \colon [\![e_1]\!]_{\mathrm{exp}} s \mid \ldots \mid v_n \colon [\![e_n]\!]_{\mathrm{exp}} s \,].$$

Then

$$[\![e/\delta]\!]_{\mathrm{exp}} s = [\![e]\!]_{\mathrm{exp}} \widehat{s}.$$

Here we have introduced a notation for describing the extension or variation of a function. We write $[\, f \mid x_1 \colon y_1 \mid \ldots \mid x_n \colon y_n \,]$ for the function whose domain is the union of the domain of f with $\{x_1, \ldots, x_n\}$, that maps each x_i into y_i and all other members x of the domain of f into $f\,x$.

A similar result for assertions will be given in the next section. As we will see in the next chapter, however, the situation for commands is more subtle.

2.1 The Meaning of Assertions

When s is a store, h is a heap, and p is an assertion whose free variables all belong to the domain of s, we write

$$s, h \models p$$

to indicate that the state s, h *satisfies* p, or p *is true in* s, h, or p *holds in* s, h. Then the following formulas define this relation by induction on the structure of p. (Here we write $h_0 \perp h_1$ when h_0 and h_1 are heaps with disjoint domains, and $h_0 \cdot h_1$ to denote the union of heaps with disjoint domains.)

$$\begin{aligned}
s, h \models b \ &\text{ iff } \ [\![b]\!]_{\text{bexp}} s = \textbf{true}, \\
s, h \models \neg\, p \ &\text{ iff } \ s, h \models p \text{ is false}, \\
s, h \models p_0 \wedge p_1 \ &\text{ iff } \ s, h \models p_0 \text{ and } s, h \models p_1 \\
&\qquad\qquad\qquad\qquad \text{(and similarly for } \vee,\ \Rightarrow,\ \Leftrightarrow\text{)}, \\
s, h \models \forall v.\ p \ &\text{ iff } \ \forall x \in \mathbf{Z}.\ [\, s \mid v \colon x \,], h \models p, \\
s, h \models \exists v.\ p \ &\text{ iff } \ \exists x \in \mathbf{Z}.\ [\, s \mid v \colon x \,], h \models p, \\
s, h \models \mathbf{emp} \ &\text{ iff } \ \operatorname{dom} h = \{\}, \\
s, h \models e \mapsto e' \ &\text{ iff } \ \operatorname{dom} h = \{[\![e]\!]_{\text{exp}} s\} \text{ and } h([\![e]\!]_{\text{exp}} s) = [\![e']\!]_{\text{exp}} s, \\
s, h \models p_0 \ast p_1 \ &\text{ iff } \ \exists h_0, h_1.\ h_0 \perp h_1 \text{ and } h_0 \cdot h_1 = h \text{ and} \\
&\qquad\qquad\qquad\qquad s, h_0 \models p_0 \text{ and } s, h_1 \models p_1, \\
s, h \models p_0 \mathrel{-\!\!\ast} p_1 \ &\text{ iff } \ \forall h'.\ (h' \perp h \text{ and } s, h' \models p_0) \text{ implies} \\
&\qquad\qquad\qquad\qquad s, h \cdot h' \models p_1.
\end{aligned}$$

All but the last four formulas coincide with the standard interpretation of predicate logic, with the heap h being carried along without change.

When $s, h \models p$ holds for all states s, h (such that the domain of s contains the free variables of p), we say that p is *valid*. When $s, h \models p$ holds for some state s, h, we say that p is *satisfiable*.

The following illustrates the use of these formulas to determine the meaning of an assertion:

$$\begin{aligned}
s, h \models \mathsf{x} \mapsto 0 \ast \mathsf{y} \mapsto 1 \text{ iff } & \exists h_0, h_1.\ h_0 \perp h_1 \text{ and } h_0 \cdot h_1 = h \\
& \quad \text{and } s, h_0 \models \mathsf{x} \mapsto 0 \\
& \quad \text{and } s, h_1 \models \mathsf{y} \mapsto 1 \\
\text{iff } & \exists h_0, h_1.\ h_0 \perp h_1 \text{ and } h_0 \cdot h_1 = h \\
& \quad \text{and } \operatorname{dom} h_0 = \{s\,\mathsf{x}\} \text{ and } h_0(s\,\mathsf{x}) = 0 \\
& \quad \text{and } \operatorname{dom} h_1 = \{s\,\mathsf{y}\} \text{ and } h_1(s\,\mathsf{y}) = 1 \\
\text{iff } & s\,\mathsf{x} \neq s\,\mathsf{y} \\
& \quad \text{and } \operatorname{dom} h = \{s\,\mathsf{x}, s\,\mathsf{y}\} \\
& \quad \text{and } h(s\,\mathsf{x}) = 0 \text{ and } h(s\,\mathsf{y}) = 1 \\
\text{iff } & s\,\mathsf{x} \neq s\,\mathsf{y} \text{ and } h = [\, s\,\mathsf{x}\colon 0 \mid s\,\mathsf{y}\colon 1\,].
\end{aligned}$$

The following illustrate the meaning of $\mapsto$ and $\hookrightarrow$:

$$\begin{aligned}
s, h \models \mathsf{x} \mapsto \mathsf{y} &\text{ iff } \operatorname{dom} h = \{s\,\mathsf{x}\} \text{ and } h(s\,\mathsf{x}) = s\,\mathsf{y} \\
s, h \models \mathsf{x} \mapsto - &\text{ iff } \operatorname{dom} h = \{s\,\mathsf{x}\} \\
s, h \models \mathsf{x} \hookrightarrow \mathsf{y} &\text{ iff } s\,\mathsf{x} \in \operatorname{dom} h \text{ and } h(s\,\mathsf{x}) = s\,\mathsf{y} \\
s, h \models \mathsf{x} \hookrightarrow - &\text{ iff } s\,\mathsf{x} \in \operatorname{dom} h \\
s, h \models \mathsf{x} \mapsto \mathsf{y}, \mathsf{z} &\text{ iff } h = [\, s\,\mathsf{x}\colon s\,\mathsf{y} \mid s\,\mathsf{x} + 1\colon s\,\mathsf{z}\,] \\
s, h \models \mathsf{x} \mapsto -, - &\text{ iff } \operatorname{dom} h = \{s\,\mathsf{x}, s\,\mathsf{x} + 1\} \\
s, h \models \mathsf{x} \hookrightarrow \mathsf{y}, \mathsf{z} &\text{ iff } h \supseteq [\, s\,\mathsf{x}\colon s\,\mathsf{y} \mid s\,\mathsf{x} + 1\colon s\,\mathsf{z}\,] \\
s, h \models \mathsf{x} \hookrightarrow -, - &\text{ iff } \operatorname{dom} h \supseteq \{s\,\mathsf{x}, s\,\mathsf{x} + 1\}.
\end{aligned}$$

To illustrate the meaning of the separating conjunction, suppose $s\,\mathsf{x}$ and

$s\,\mathsf{y}$ are distinct addresses, so that

$$h_0 = [\, s\,\mathsf{x}\colon 0\,] \qquad \text{and} \qquad h_1 = [\, s\,\mathsf{y}\colon 1\,]$$

are heaps with disjoint domains. Then

If p is:	then $s, h \models p$ iff:
$\mathsf{x} \mapsto 0$	$h = h_0$
$\mathsf{y} \mapsto 1$	$h = h_1$
$\mathsf{x} \mapsto 0 \ast \mathsf{y} \mapsto 1$	$h = h_0 \cdot h_1$
$\mathsf{x} \mapsto 0 \ast \mathsf{x} \mapsto 0$	**false**
$\mathsf{x} \mapsto 0 \vee \mathsf{y} \mapsto 1$	$h = h_0$ or $h = h_1$
$\mathsf{x} \mapsto 0 \ast (\mathsf{x} \mapsto 0 \vee \mathsf{y} \mapsto 1)$	$h = h_0 \cdot h_1$
$(\mathsf{x} \mapsto 0 \vee \mathsf{y} \mapsto 1) \ast (\mathsf{x} \mapsto 0 \vee \mathsf{y} \mapsto 1)$	$h = h_0 \cdot h_1$
$\mathsf{x} \mapsto 0 \ast \mathsf{y} \mapsto 1 \ast (\mathsf{x} \mapsto 0 \vee \mathsf{y} \mapsto 1)$	**false**
$\mathsf{x} \mapsto 0 \ast \mathbf{true}$	$h_0 \subseteq h$
$\mathsf{x} \mapsto 0 \ast \neg\, \mathsf{x} \mapsto 0$	$h_0 \subseteq h$.

Here the behavior of disjunction is slightly surprising.

The effect of substitution on the meaning of assertions is similar to that on expressions. We consider only the more general partial case:

Proposition 3 *(Partial Substitution Law for Assertions) Suppose p is an assertion, and let δ abbreviate the substitution*

$$v_1 \to e_1, \ldots, v_n \to e_n,$$

Then let s be a store such that $(\mathrm{FV}(p) - \{v_1, \ldots, v_n\}) \cup \mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \subseteq \mathrm{dom}\, s$*, and let*

$$\widehat{s} = [\, s \mid v_1\colon [\![e_1]\!]_{\mathrm{exp}} s \mid \ldots \mid v_n\colon [\![e_n]\!]_{\mathrm{exp}} s\,].$$

Then

$$s, h \models (p/\delta) \ \textit{iff}\ \ \widehat{s}, h \models p.$$

2.2 Inference

We will reason about assertions using inference rules of the form

$$\frac{\mathcal{P}_1 \qquad \cdots \qquad \mathcal{P}_n}{\mathcal{C},}$$

where the zero or more $\mathcal{P}_i$ are called the *premisses* and the $\mathcal{C}$ is called the *conclusion*.

The premisses and conclusion are schemata, i.e., they may contain *metavariables*, each of which ranges over some set of phrases, such as expressions, variables, or assertions. To avoid confusion, we will use italic (or occasionally Greek) characters for metavariables, but sans serif characters for the *object* variables of the logic and programming lanugage.

An instance of an inference rule is obtained by replacing each metavariable by a phrase in its range. These replacements must satisfy the *side conditions* (if any) of the rule. (Since this is replacement of metavariables rather than substitution for variables, there is never any renaming.) For instance,

Inference Rules	Instances
$\dfrac{p_0 \qquad p_0 \Rightarrow p_1}{p_1}$	$\dfrac{\mathsf{x} + 0 = \mathsf{x} \qquad \mathsf{x} + 0 = \mathsf{x} \Rightarrow \mathsf{x} = \mathsf{x} + 0}{\mathsf{x} = \mathsf{x} + 0}$
$\dfrac{}{e_2 = e_1 \Rightarrow e_1 = e_2}$	$\dfrac{}{\mathsf{x} + 0 = \mathsf{x} \Rightarrow \mathsf{x} = \mathsf{x} + 0}$
$\mathsf{x} + 0 = \mathsf{x}$	$\mathsf{x} + 0 = \mathsf{x}$

An inference rule is *sound* iff, for all instances, if the premisses of the instance are all valid, then the conclusion is valid.

A *formal proof* (for assertions) is a sequence of assertions, each of which is the conclusion of some instance of a sound inference rule whose premisses occur earlier in the sequence. For example,

$$\mathsf{x} + 0 = \mathsf{x}$$
$$\mathsf{x} + 0 = \mathsf{x} \Rightarrow \mathsf{x} = \mathsf{x} + 0$$
$$\mathsf{x} = \mathsf{x} + 0.$$

Since we require the inference rules used in the proof to be sound, it follows that the assertions in a formal proof must all be valid.

An inference rule with zero premisses is called an *axiom schema*. The overbar is often omitted. (Notice that the first assertion in a proof must be an instance of an axiom schema.) An axiom schema containing no metavariables (so that it is its own unique instance) is called an *axiom*. The overbar is usually omitted.

The following are inference rules that are sound for predicate calculus, and remain sound for the extension to assertions in separation logic:

$$
\begin{array}{ll}
\dfrac{p \qquad p \Rightarrow q}{q} & \text{(modus ponens)} \\[2ex]
\dfrac{p \Rightarrow q}{p \Rightarrow (\forall v.\ q)} & \text{when } v \notin \mathrm{FV}(p) \\[2ex]
\dfrac{p \Rightarrow q}{(\exists v.\ p) \Rightarrow q} & \text{when } v \notin \mathrm{FV}(q).
\end{array}
\tag{2.1}
$$

In addition, the following axiom schemas are sound for predicate calculus and assertions in separation logic:

$$
\begin{aligned}
p &\Rightarrow (q \Rightarrow p) \\
(p \Rightarrow (q \Rightarrow r)) &\Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r)) \\
(p \wedge q) &\Rightarrow p \\
(p \wedge q) &\Rightarrow q \\
p &\Rightarrow (q \Rightarrow (p \wedge q)) \\
p &\Rightarrow (p \vee q) \\
q &\Rightarrow (p \vee q) \\
(p \Rightarrow r) &\Rightarrow ((q \Rightarrow r) \Rightarrow ((p \vee q) \Rightarrow r)) \\
(p \Rightarrow q) &\Rightarrow ((p \Rightarrow \neg\, q) \Rightarrow \neg\, p) \\
\neg(\neg\, p) &\Rightarrow p \\
(p \Leftrightarrow q) &\Rightarrow ((p \Rightarrow q) \wedge (q \Rightarrow p)) \\
((p \Rightarrow q) \wedge (q \Rightarrow p)) &\Rightarrow (p \Leftrightarrow q) \\
(\forall v.\ p) &\Rightarrow (p/v \rightarrow e) \\
(p/v \rightarrow e) &\Rightarrow (\exists v.\ p).
\end{aligned}
\tag{2.2}
$$

The rules in (2.1) and (2.2) (with the exception of the rules involving $\Leftrightarrow$) are taken from Kleene [36, page 82]. They are (one of many possible) complete sets of *logical* inference rules (i.e., rules that are sound for any interpretation of the domain of discourse or the function and predicate symbols).

It is important to notice the difference between

$$\frac{p}{q} \qquad \text{and} \qquad \frac{}{p \Rightarrow q}.$$

A rule of the first form will be sound when the validity of p implies the validity of q, but a rule of the second form will be sound when $p \Rightarrow q$ is valid, i.e., when, for every state, if p holds in that state then q holds in the same state. For example, the rule of generalization (which can be derived from the rules above),

$$\frac{p}{\forall v.\ p}$$

is sound, but the corresponding implication is not:

$$p \Rightarrow \forall v.\ p \qquad \text{(unsound)}$$

(since, for instance, $\mathsf{x} = 0 \Rightarrow \forall \mathsf{x}.\ \mathsf{x} = 0$ is not valid).

We have already seen the following general inference rules for assertions:

$$\begin{aligned}
p_0 * p_1 &\Leftrightarrow p_1 * p_0 \\
(p_0 * p_1) * p_2 &\Leftrightarrow p_0 * (p_1 * p_2) \\
p * \mathbf{emp} &\Leftrightarrow p \\
(p_0 \vee p_1) * q &\Leftrightarrow (p_0 * q) \vee (p_1 * q) \\
(p_0 \wedge p_1) * q &\Rightarrow (p_0 * q) \wedge (p_1 * q) \\
(\exists x.\ p_0) * p_1 &\Leftrightarrow \exists x.\ (p_0 * p_1) \quad \text{when } x \text{ not free in } p_1 \\
(\forall x.\ p_0) * p_1 &\Rightarrow \forall x.\ (p_0 * p_1) \quad \text{when } x \text{ not free in } p_1
\end{aligned} \tag{2.3}$$

$$\frac{p_0 \Rightarrow p_1 \qquad q_0 \Rightarrow q_1}{p_0 * q_0 \Rightarrow p_1 * q_1} \quad \text{(monotonicity)}$$

$$\frac{p_0 * p_1 \Rightarrow p_2}{p_0 \Rightarrow (p_1 -\!\!* p_2)} \quad \text{(currying)} \qquad \frac{p_0 \Rightarrow (p_1 -\!\!* p_2)}{p_0 * p_1 \Rightarrow p_2.} \quad \text{(decurrying)}$$

as well as specific rules for $\mapsto$ and $\hookrightarrow$:

$$\begin{gathered}
e_0 \mapsto e_0' \wedge e_1 \mapsto e_1' \Leftrightarrow e_0 \mapsto e_0' \wedge e_0 = e_1 \wedge e_0' = e_1' \\
e_0 \hookrightarrow e_0' * e_1 \hookrightarrow e_1' \Rightarrow e_0 \neq e_1 \\
\mathbf{emp} \Leftrightarrow \forall x.\ \neg(x \hookrightarrow -) \\
(e \hookrightarrow e') \wedge p \Rightarrow (e \mapsto e') * ((e \mapsto e') -\!\!* p).
\end{gathered} \tag{2.4}$$

2.3 Special Classes of Assertions

There are several classes of assertions that play an important role in separation logic. In each case, the class has a semantic definition that implies the soundness of additional axiom schemata; often there are also useful syntactic criteria that imply membership in the class.

2.3.1 Pure Assertions

The simplest such class is that of pure assertions, which are independent of the heap. More precisely, an assertion p is *pure* iff, for all stores s and all heaps h and h',

$$s, h \models p \text{ iff } s, h' \models p.$$

A sufficient syntactic criteria is that an assertion is pure if it does not contain **emp**, $\mapsto$, or $\hookrightarrow$.

When all of the subphrases of an assertion are pure, the distinction between separating and ordinary operations collapses, so that $*$ and $\wedge$ are interchangeable, as are $-\!\!*$ and $\Rightarrow$. The following axiom schemata delineate the consequences when some subassertions are pure:

$$\begin{array}{rl}
p_0 \wedge p_1 \Rightarrow p_0 * p_1 & \text{when } p_0 \text{ or } p_1 \text{ is pure} \\
p_0 * p_1 \Rightarrow p_0 \wedge p_1 & \text{when } p_0 \text{ and } p_1 \text{ are pure} \\
(p \wedge q) * r \Leftrightarrow (p * r) \wedge q & \text{when } q \text{ is pure} \\
(p_0 -\!\!* p_1) \Rightarrow (p_0 \Rightarrow p_1) & \text{when } p_0 \text{ is pure} \\
(p_0 \Rightarrow p_1) \Rightarrow (p_0 -\!\!* p_1) & \text{when } p_0 \text{ and } p_1 \text{ are pure.}
\end{array}$$

(The third of these schemata is ubiquitous in proofs of programs.)

2.3.2 Strictly Exact Assertions

At the opposite extreme from pure assertions are strictly exact assertions, which uniquely determine the heap. An assertion is *strictly exact* iff, for all stores s and all heaps h and h',

$$s, h \models p \text{ and } s, h' \models p \text{ implies } h = h'.$$

(This classification of assertions was introduced by Yang [10].)

Examples of strictly exact assertions include:

- **emp**.
- $e \mapsto e'$.
- $p * q$, when p and q are strictly exact.
- $p \wedge q$, when p or q is strictly exact.

- p, when $p \Rightarrow q$ is valid and q is strictly exact.

Proposition 4 *When q is strictly exact,*

$$((q \ * \ \mathbf{true}) \wedge p) \Rightarrow (q \ * \ (q \mathrel{-\!\!*} p))$$

is valid.

PROOF Suppose $s, h \models (q \ * \ \mathbf{true}) \wedge p$, so that $s, h \models q \ * \ \mathbf{true}$ and $s, h \models p$. Then there are heaps h_0 and h_1 such that $h_0 \perp h_1$, $h_0 \cdot h_1 = h$, and $s, h_0 \models q$.

To see that $s, h_1 \models q \mathrel{-\!\!*} p$, let h' be any heap such that $h' \perp h_1$ and $s, h' \models q$. Since q is strictly exact, $h' = h_0$, so that $h' \cdot h_1 = h_0 \cdot h_1 = h$, and thus $s, h' \cdot h_1 \models p$.

Then $s, h_0 \cdot h_1 \models q \ * \ (q \mathrel{-\!\!*} p)$, so that $s, h \models q \ * \ ((q \mathrel{-\!\!*} p)$.

END OF PROOF

For example, taking q to be the strictly exact assertion $e \mapsto e'$ gives the final axiom schema in (2.4).

2.3.3 Precise Assertions

Given a heap, if a precise assertion holds for any subheap, then it holds for a unique subheap. In other words, an assertion q is *precise* iff, for all s and h, there is at most one $h' \subseteq h$ such that

$$s, h' \models q.$$

Examples of precise assertions include:

- Strictly exact assertions.
- $e \mapsto -$.
- $p \ * \ q$, when p and q are precise.
- $p \wedge q$, when p or q is precise.
- p, when $p \Rightarrow q$ is valid and q is precise.
- $\mathsf{list}\ \alpha\ e$ and $\exists \alpha.\ \mathsf{list}\ \alpha\ e$.
- $\mathsf{tree}\ \tau\ (e)$ and $\exists \tau.\ \mathsf{tree}\ \tau\ (e)$.

On the other hand, the following are instances of imprecise assertions:

$$\mathbf{true} \qquad \mathbf{emp} \lor \mathsf{x} \mapsto 10 \qquad \mathsf{x} \mapsto 10 \lor \mathsf{y} \mapsto 10 \qquad \exists \mathsf{x}.\ \mathsf{x} \mapsto 10$$

$$\mathsf{dag}\ \tau\ (\mathsf{i}) \qquad \exists \tau.\ \mathsf{dag}\ \tau\ (\mathsf{i})$$

There is a close connection between preciseness and distributivity. The semi-distributive laws

$$(p_0 \land p_1) \ast q \Rightarrow (p_0 \ast q) \land (p_1 \ast q)$$

$$(\forall x.\ p) \ast q \Rightarrow \forall x.\ (p \ast q) \quad \text{when } x \text{ not free in } q$$

are valid for all assertions. But their converses

$$(p_0 \ast q) \land (p_1 \ast q) \Rightarrow (p_0 \land p_1) \ast q$$

$$\forall x.\ (p \ast q) \Rightarrow (\forall x.\ p) \ast q \quad \text{when } x \text{ not free in } q$$

are not. For example, when

$$s(\mathsf{x}) = 1 \quad s(\mathsf{y}) = 2 \quad h = [\,1\colon 10 \mid 2\colon 20\,],$$

the assertion

$$(\mathsf{x} \mapsto 10 \ast (\mathsf{x} \mapsto 10 \lor \mathsf{y} \mapsto 20)) \land (\mathsf{y} \mapsto 20 \ast (\mathsf{x} \mapsto 10 \lor \mathsf{y} \mapsto 20))$$

is true, but

$$((\mathsf{x} \mapsto 10 \land \mathsf{y} \mapsto 20) \ast (\mathsf{x} \mapsto 10 \lor \mathsf{y} \mapsto 20))$$

is false.

However, the converses are valid when q is precise:

Proposition 5 *When q is precise,*

$$(p_0 \ast q) \land (p_1 \ast q) \Rightarrow (p_0 \land p_1) \ast q$$

is valid. When q is precise and x is not free in q,

$$\forall x.\ (p \ast q) \Rightarrow (\forall x.\ p) \ast q$$

is valid.

Proof (of the first law) Suppose $s, h \models (p_0 \ast q) \land (p_1 \ast q)$. Then there are:

- An $h_0 \subseteq h$ such that $s, h - h_0 \models p_0$ and $s, h_0 \models q$, and
- An $h_1 \subseteq h$ such that $s, h - h_1 \models p_1$ and $s, h_1 \models q$.

Thus, since q is precise, $h_0 = h_1$, $h - h_0 = h - h_1$, $s, h - h_0 \models p_0 \land p_1$, and $s, h \models (p_0 \land p_1) \ast q$. END OF PROOF

2.3.4 Intuitionistic Assertions

Intuitionistic assertions are monotone with respect to the extension of heaps. An assertion i is *intuitionistic* iff, for all stores s and heaps h and h':

$$(h \subseteq h' \text{ and } s, h \models i) \text{ implies } s, h' \models i.$$

Assume i and i' are intuitionistic assertions, p is any assertion, e and e' are expressions, and τ denotes an S-expression. Then the following assertions are intuitionistic:

$$\begin{array}{cc}
\text{Any pure assertion} & p \ast i \\
p \mathrel{-\!\!*} i & i \mathrel{-\!\!*} p \\
i \wedge i' & i \vee i' \\
\forall v.\ i & \exists v.\ i \\
\mathsf{dag}\ \tau\ (e) & \exists \tau.\ \mathsf{dag}\ \tau\ (e),
\end{array}$$

and as special cases:

$$p \ast \mathbf{true} \qquad \mathbf{true} \mathrel{-\!\!*} p \qquad e \hookrightarrow e'.$$

The following inference rules are sound when i and i' are intuitionistic:

$$(i \ast i') \Rightarrow (i \wedge i')$$

$$(i \ast p) \Rightarrow i \qquad i \Rightarrow (p \mathrel{-\!\!*} i)$$

$$\frac{p \Rightarrow i}{(p \ast \mathbf{true}) \Rightarrow i} \qquad \frac{i \Rightarrow p}{i \Rightarrow (\mathbf{true} \mathrel{-\!\!*} p).}$$

The last two of these rules, in conjunction with the rules

$$p \Rightarrow (p \ast \mathbf{true}) \qquad (\mathbf{true} \mathrel{-\!\!*} p) \Rightarrow p,$$

which hold for all assertions, implies that $p \ast \mathbf{true}$ is the strongest intuitionistic assertion weaker than p, and $\mathbf{true} \mathrel{-\!\!*} p$ is the weakest intuitionistic assertion that is stronger than p. They also imply that, when i is intuitionistic,

$$i \Leftrightarrow (i \ast \mathbf{true}) \qquad (\mathbf{true} \mathrel{-\!\!*} i) \Leftrightarrow i.$$

If we define the operations

$$\overset{\text{i}}{\neg}\, p \overset{\text{def}}{=} \textbf{true} \mathrel{-\!\!*} (\neg\, p)$$
$$p \overset{\text{i}}{\Rightarrow} q \overset{\text{def}}{=} \textbf{true} \mathrel{-\!\!*} (p \Rightarrow q)$$
$$p \overset{\text{i}}{\Leftrightarrow} q \overset{\text{def}}{=} \textbf{true} \mathrel{-\!\!*} (p \Leftrightarrow q),$$

then the assertions built from pure assertions and $e \hookrightarrow e'$, using these operations and $\wedge$, $\vee$, $\forall$, $\exists$, $*$, and $\mathrel{-\!\!*}$ form the image of Ishtiaq and O'Hearn's modal translation from intuitionistic separation logic to the classical version [5].

2.3.5 Supported Assertions

It is easily seen that no assertion that is true in any state can be both precise and intuitionistic.

Thus satisfiable precise assertions do not inhabit the intuitionistic world. This raises the question of whether there is a class of assertions that bears the same relationship to intuitionistic assertions that precise assertions bear to arbitrary (i.e., classical) assertions. In this section, we will see that this role is filled by supported assertions.

An assertion q is *supported* iff, for all s, h_0, and h_1, if $h_0 \cup h_1$ is a function, and $s, h_0 \models q$ and $s, h_1 \models q$ are true, then there is an h' such that $h' \subseteq h_0$, $h' \subseteq h_1$, and $s, h' \models q$ is true. Equivalently,

Proposition 6 *An assertion q is* supported *iff, for all s and h, if the set*

$$H = \{\, h' \mid h' \subseteq h \text{ and } s, h' \models q \,\}$$

is nonempty, then it has a least element.

PROOF Suppose that q is supported, fix s and h, and let h_0 be a member of H with minimum domain size, and h_1 be any member of H. Since h_0 and h_1 are both subsets of h, $h_0 \cup h_1$ must be a function. Then the first definition guarantees that there is an $h' \in H$ that is a subset of both h_0 and h_1. But h' must be equal to h_0, since otherwise it would have a smaller domain size. Thus $h_0 \subseteq h_1$ for every $h_1 \in H$.

On the other hand, suppose that q meets the conditions of the proposition, and $h_0 \cup h_1$ is a function, $s, h_0 \models q$ and $s, h_1 \models q$ are true. Take h to be

$h_0 \cup h_1$, so that $h_0, h_1 \in H$. Then take h' to be the least element of H.

END OF PROOF

For example, the following assertions are imprecise, intuitionistic, and supported:

$$\mathbf{true} \qquad \mathsf{x} \hookrightarrow 10 \qquad \mathsf{x} \hookrightarrow 10 \wedge \mathsf{y} \hookrightarrow 10 \qquad \mathsf{dag}\,\tau\,(\mathsf{i}) \qquad \exists \tau.\ \mathsf{dag}\,\tau\,(\mathsf{i}),$$

imprecise, intuitionistic, and unsupported:

$$\mathsf{x} \hookrightarrow 10 \vee \mathsf{y} \hookrightarrow 10 \qquad \exists \mathsf{x}.\ \mathsf{x} \hookrightarrow 10 \qquad \neg\,\mathbf{emp},$$

imprecise, nonintuitionistic, and supported:

$$\mathbf{emp} \vee \mathsf{x} \mapsto 10,$$

and imprecise, nonintuitionistic, and unsupported:

$$\mathsf{x} \mapsto 10 \vee \mathsf{y} \mapsto 10 \qquad \exists \mathsf{x}.\ \mathsf{x} \mapsto 10.$$

When q is supported and the remaining assertions are intuitionistic, the semidistributive laws given earlier become full distributive laws:

Proposition 7 *When p_0 and p_1 are intuitionistic and q is supported,*

$$(p_0 \ast q) \wedge (p_1 \ast q) \Rightarrow (p_0 \wedge p_1) \ast q$$

is valid. When p is intuitionistic, q is supported, and x is not free in q,

$$\forall x.\ (p \ast q) \Rightarrow (\forall x.\ p) \ast q$$

is valid.

PROOF (of the first law): Suppose $s, h \models (p_0 \ast q) \wedge (p_1 \ast q)$. Then there are:

An $h_0 \subseteq h$ such that $s, h - h_0 \models p_0$ and $s, h_0 \models q$,

An $h_1 \subseteq h$ such that $s, h - h_1 \models p_1$ and $s, h_1 \models q$.

Then, since q is supported and $h_0 \cup h_1$ is a function, there is an $h' \subseteq h_0, h_1$ such that $s, h' \models q$. Moreover, since $h - h_0, h - h_1 \subseteq h - h'$, and p_0 and p_1 are intuitionistic, $s, h - h' \models p_0 \wedge p_1$, and therefore $s, h \models (p_0 \wedge p_1) \ast q$.

END OF PROOF

We have already seen that, if p is any assertion, then $p \ast \mathbf{true}$ is intuitionistic, and if i is intuitionistic, then $i \Leftrightarrow (i \ast \mathbf{true})$. Thus, $- \ast \mathbf{true}$ maps arbitrary assertions into intuitionistic assertions, and acts as an identity (up to equivalence of assertions) on the latter.

In addition,

Proposition 8 *(1) If p is precise, then p is supported. (2) q is supported iff $q \ast \mathbf{true}$ is supported.*

PROOF (1) If p is precise, then, for any s and h, the set $H = \{ h' \mid h' \subseteq h \text{ and } s, h' \models q \}$ in Proposition 6 contains at most one element.

(2) Suppose q is supported, $h_0 \cup h_1$ is a function, $s, h_0 \models q \ast \mathbf{true}$ and $s, h_1 \models q \ast \mathbf{true}$. Then there are $h'_0 \subseteq h_0$ and $h'_1 \subseteq h_1$ such that $s, h'_0 \models q$ and $s, h'_1 \models q$, and since q is supported, there is an h' that is a subset of h'_0 and h'_1, and therefore h_0 and h_1, such that $s, h' \models q$, and therefore $s, h' \models q \ast \mathbf{true}$.

Suppose $q \ast \mathbf{true}$ is supported, $h_0 \cup h_1$ is a function, $s, h_0 \models q$ and $s, h_1 \models q$. Then $s, h_0 \models q \ast \mathbf{true}$ and $s, h_1 \models q \ast \mathbf{true}$, and since $q \ast \mathbf{true}$ is supported, there is a common subset h' of h_0 and h_1 such that $s, h' \models q \ast \mathbf{true}$. But then there is a subset h'' of h', and therefore of h_0 and h_1, such that $s, h'' \models q$. END OF PROOF

(The analogous conjecture, that q is supported iff $\mathbf{true} -\!\!\ast\, q$ is supported, fails in both directions. Suppose q is $1 \hookrightarrow 1 \vee 2 \hookrightarrow 2 \vee \mathbf{emp}$. Then q is supported, since the empty heap is the least heap in which it holds, but $\mathbf{true} -\!\!\ast\, q$ is not supported, since it holds for the heaps $[\,1\colon 1\,]$ and $[\,2\colon 2\,]$, but not for their only common subheap, which is the empty heap. On the other hand, suppose q is $1 \hookrightarrow 1 \vee 2 \mapsto 2$. Then $\mathbf{true} -\!\!\ast\, q$ is supported, since $[\,1\colon 1\,]$ is the least heap for which it holds, but q is not supported, since it holds for $[\,1\colon 1\,]$ and $[\,2\colon 2\,]$, but not for the empty heap.)

Thus $- \ast \mathbf{true}$ maps precise assertions into supported intuitionistic assertions and acts as an identity (up to equivalence of assertions) on the latter.

2.3.6 The Precising Operation

Having found an operation that maps precise assertions into supported intuitionistic assertions, we now introduce an operation, due to Yang, that moves in the opposite direction, which we call the *precising* operation:

$$\mathrm{Pr}\, p \stackrel{\text{def}}{=} p \wedge \neg(p \ast \neg\, \mathbf{emp}).$$

For example,

$$\mathrm{Pr}\ \mathbf{true} \text{ iff } \mathbf{emp}$$

$$\mathrm{Pr}\ (\mathsf{x} \hookrightarrow 10) \text{ iff } \mathsf{x} \mapsto 10$$

$$\mathrm{Pr}\ (\mathbf{emp} \vee \mathsf{x} \mapsto 10) \text{ iff } \mathbf{emp}$$

$$\begin{array}{l}\mathrm{Pr}\ (\mathsf{x} \hookrightarrow 10 \wedge \mathsf{y} \hookrightarrow 10) \text{ iff} \\ \qquad \mathbf{if}\ \mathsf{x} = \mathsf{y}\ \mathbf{then}\ \mathsf{x} \mapsto 10\ \mathbf{else}\ (\mathsf{x} \mapsto 10 \ast \mathsf{y} \mapsto 10)\end{array}$$

Then

Proposition 9 *(1) If p is supported, then* Pr p *is precise. (2) If p is precise, then* Pr $p \Leftrightarrow p$.

Proof (1) Suppose p is supported, and $h_0, h_1 \subseteq h$ are such that $s, h_0 \models \mathrm{Pr}\ p$ and $s, h_1 \models \mathrm{Pr}\ p$.

We must show $h_0 = h_1$. Assume the contrary. Since $s, h_0 \models p$, $s, h_1 \models p$, and p is supported, there is a common subset h' of h_0 and h_1 such that $s, h' \models p$. Since $h_0 \neq h_1$, however, h' must be a proper subset of h_i for $i = 0$ or $i = 1$. Thus $h_i = h' \cdot (h_i - h')$, where $s, h_i - h' \models \neg\,\mathbf{emp}$. Then $s, h_i \models p \ast \neg\,\mathbf{emp}$, which contradicts $s, h_i \models \mathrm{Pr}\ p$.

(2) Obviously, $\mathrm{Pr}\ p \Rightarrow p$. To show the opposite implication when p is precise, assume $s, h \models p$. If $s, h \models p \ast \neg\,\mathbf{emp}$ held, then there would be a proper subset h' of h such that $s, h' \models p$, which would contradict the preciseness of p. Thus $s, h \models \neg(p \ast \neg\,\mathbf{emp})$. end of proof

Thus Pr maps supported operations into precise operations, and acts as an identity on the latter.

(The conjecture that, when p is supported, Pr p is the weakest precise assertion stronger than p, is false. Suppose p is the supported assertion $1 \mapsto 1 \vee \mathbf{emp}$. Then $1 \mapsto 1$ is a precise assertion stronger than p, but Pr p, which is equivalent to **emp**, is not weaker than $1 \mapsto 1$.)

Additional relationships between $- \ast\ \mathbf{true}$ and Pr are provided by

Proposition 10

(1) $\mathrm{Pr}\ (p \ast \mathbf{true}) \Rightarrow p$.

(2) $p \Rightarrow \mathrm{Pr}\ (p \ast \mathbf{true})$ *when p is precise.*

(3) $(\mathrm{Pr}\ q) * \mathbf{true} \Rightarrow q$ *when* q *is intuitionistic.*

(4) $q \Rightarrow (\mathrm{Pr}\ q) * \mathbf{true}$ *when* q *is supported.*

Therefore, $\mathrm{Pr}\ (p * \mathbf{true}) \Leftrightarrow p$ *when* p *is precise, and* $(\mathrm{Pr}\ q) * \mathbf{true} \Rightarrow q$ *when* q *is supported and intuitionistic.*

PROOF (1) Suppose $s, h \models \mathrm{Pr}\ (p * \mathbf{true})$. Then $s, h \models p * \mathbf{true}$ and $s, h \models \neg(p * \mathbf{true} * \neg\,\mathbf{emp})$, and there is an $h' \subseteq h$ such that $s, h' \models p$. If $h' = h$, we are done; otherwise, h' is a proper subset of h, so that $s, h \models p * \mathbf{true} * \neg\,\mathbf{emp}$, which contradicts $s, h \models \neg(p * \mathbf{true} * \neg\,\mathbf{emp})$.

(2) Suppose $s, h \models p$. Then $s, h \models p * \mathbf{true}$. Moreover, $s, h \models \neg(p * \mathbf{true} * \neg\,\mathbf{emp})$, for otherwise $s, h \models p * \mathbf{true} * \neg\,\mathbf{emp}$ would imply that there is a proper subset h' of h such that $s, h' \models p$, which would contradict the preciseness of p.

(3) Suppose $s, h \models (\mathrm{Pr}\ q) * \mathbf{true}$. Then there is an $h' \subseteq h$ such that $s, h' \models \mathrm{Pr}\ q$. Then $s, h' \models q$, and since q is intuitionistic, $s, h \models q$.

(4) Suppose $s, h \models q$. Since q is supported, there is a least $h' \subseteq h$ such that $s, h' \models q$. Then $s, h' \models \mathrm{Pr}\ q$, since otherwise $s, h' \models q * \neg\,\mathbf{emp}$, which would imply that a proper subset h'' of h' would satisfy $s, h'' \models q$, contradicting the leastness of h'. Thus $s, h \models (\mathrm{Pr}\ q) * \mathbf{true}$.

END OF PROOF

Thus $- * \mathbf{true}$ and Pr are isomorphisms between the set of precise assertions and the set of supported intuitionistic assertions, and act as identities on these sets:

$$\begin{tikzcd} \boxed{\text{Precise}} \arrow[loop left, "\mathrm{Pr}"] \arrow[r, shift left, "- * \mathbf{true}"] & \boxed{\text{Supported Intuitionistic}} \arrow[l, shift left, "\mathrm{Pr}"] \arrow[loop right, "- * \mathbf{true}"] \end{tikzcd}$$

2.4 Some Derived Inference Rules

We conclude this chapter with the deriviations of five inference rules, which are interesting in their own right and will be useful later.

An inference-rule derivation is a proof schema, such that appropriate replacements of the metavariables will yield a formal proof from assumptions (steps that do not follow from previous steps). At the schematic level the assumptions are the premisses of the rule to be proved, and the final step is the conclusion. Thus the derivation shows how any instance of the derived rule can be replaced by a sound proof fragment.

$$\text{To derive:} \qquad \overline{q \ast (q -\!\!\ast\, p) \Rightarrow p} \tag{2.5}$$

1. $q \ast (q -\!\!\ast\, p) \Rightarrow (q -\!\!\ast\, p) \ast q$ $\qquad (p_0 \ast p_1 \Rightarrow p_1 \ast p_0)$
2. $(q -\!\!\ast\, p) \Rightarrow (q -\!\!\ast\, p)$ $\qquad (p \Rightarrow p)$
3. $(q -\!\!\ast\, p) \ast q \Rightarrow p$ $\qquad$ (decurrying, 2)
4. $q \ast (q -\!\!\ast\, p) \Rightarrow p$ $\qquad$ (trans impl, 1, 3)

where *transitive implication* is the inference rule

$$\frac{p \Rightarrow q \qquad q \Rightarrow r}{p \Rightarrow r}$$

(which can be derived from the rules in (2.1)).

Note that, from the rule derived above, it is easy to obtain

$$(q \ast (q -\!\!\ast\, p)) \Rightarrow ((q \ast \mathbf{true}) \wedge p),$$

which is the converse of Proposition 4 (without the restriction that q must be strictly exact).

$$\text{To derive:} \qquad \overline{r \Rightarrow (q -\!\!\ast\, (q \ast r))} \tag{2.6}$$

1. $(r \ast q) \Rightarrow (q \ast r)$ $\qquad (p_0 \ast p_1 \Rightarrow p_1 \ast p_0)$
2. $r \Rightarrow (q -\!\!\ast\, (q \ast r))$ $\qquad$ (currying, 1)

$$\text{To derive:} \qquad \overline{(p \ast r) \Rightarrow (p \ast (q -\!\!\ast\, (q \ast r)))} \tag{2.7}$$

1. $p \Rightarrow p$ $\qquad (p \Rightarrow p)$
2. $r \Rightarrow (q -\!\!\ast\, (q \ast r))$ $\qquad$ (derived above)
3. $(p \ast r) \Rightarrow (p \ast (q -\!\!\ast\, (q \ast r)))$ $\qquad$ (monotonicity, 1, 2)

$$\text{To derive:} \qquad \frac{p_0 \Rightarrow (q \mathrel{-\!\!*} r) \qquad p_1 \Rightarrow (r \mathrel{-\!\!*} s)}{p_1 * p_0 \Rightarrow (q \mathrel{-\!\!*} s)} \tag{2.8}$$

1. $p_1 \Rightarrow p_1$ $(p \Rightarrow p)$
2. $p_0 \Rightarrow (q \mathrel{-\!\!*} r)$ (assumption)
3. $p_0 * q \Rightarrow r$ (decurrying, 2)
4. $p_1 * p_0 * q \Rightarrow p_1 * r$ (monotonicity, 1, 3)
5. $p_1 \Rightarrow (r \mathrel{-\!\!*} s)$ (assumption)
6. $p_1 * r \Rightarrow s$ (decurrying, 5)
7. $p_1 * p_0 * q \Rightarrow s$ (trans impl, 4, 6)
8. $p_1 * p_0 \Rightarrow (q \mathrel{-\!\!*} s)$ (currying, 7)

$$\text{To derive:} \qquad \frac{p' \Rightarrow p \qquad q \Rightarrow q'}{(p \mathrel{-\!\!*} q) \Rightarrow (p' \mathrel{-\!\!*} q').} \tag{2.9}$$

1. $(p \mathrel{-\!\!*} q) \Rightarrow (p \mathrel{-\!\!*} q)$ $(p \Rightarrow p)$
2. $p' \Rightarrow p$ (assumption)
3. $(p \mathrel{-\!\!*} q) * p' \Rightarrow (p \mathrel{-\!\!*} q) * p$ (monotonicity, 1, 2)
4. $(p \mathrel{-\!\!*} q) * p \Rightarrow q$ (decurrying, 1)
5. $(p \mathrel{-\!\!*} q) * p' \Rightarrow q$ (trans impl, 3, 4)
6. $q \Rightarrow q'$ (assumption)
7. $(p \mathrel{-\!\!*} q) * p' \Rightarrow q'$ (trans impl, 5, 6)
8. $(p \mathrel{-\!\!*} q) \Rightarrow (p' \mathrel{-\!\!*} q')$ (currying, 7)

Exercise 1

Give a formal proof of the valid assertion

$$\big((\mathsf{x} \mapsto \mathsf{y} \,*\, \mathsf{x}' \mapsto \mathsf{y}') \,*\, \mathbf{true}\big) \Rightarrow \\ \big(((\mathsf{x} \mapsto \mathsf{y} \,*\, \mathbf{true}) \wedge (\mathsf{x}' \mapsto \mathsf{y}' \,*\, \mathbf{true})) \wedge \mathsf{x} \neq \mathsf{x}'\big)$$

from the rules in (2.3) and (2.4), and (some of) the following inference rules for predicate calculus (which can be derived from the rules in (2.1) and (2.2)):

$$p \Rightarrow \mathbf{true} \qquad p \Rightarrow p \qquad p \wedge \mathbf{true} \Rightarrow p$$

$$\frac{p \Rightarrow q \qquad q \Rightarrow r}{p \Rightarrow r} \quad \text{(trans impl)}$$

$$\frac{p \Rightarrow q \qquad p \Rightarrow r}{p \Rightarrow q \wedge r} \quad (\wedge\text{-introduction})$$

Your proof will be easier to read if you write it as a sequence of steps rather than a tree. In the inference rules, you should regard $*$ as left associative, e.g.,

$$e_0 \mapsto e_0' \,*\, e_1 \mapsto e_1' \,*\, \mathbf{true} \Rightarrow e_0 \neq e_1$$

stands for

$$(e_0 \mapsto e_0' \,*\, e_1 \mapsto e_1') \,*\, \mathbf{true} \Rightarrow e_0 \neq e_1.$$

For brevity, you may weaken $\Leftrightarrow$ to $\Rightarrow$ when it is the main operator of an axiom. You may also omit instances of the axiom schema $p \Rightarrow p$ when it is used as a premiss of the monotonicity rule.

Exercise 2

None of the following axiom schemata are sound. For each, given an instance which is not valid, along with a description of a state in which the instance is false.

$$p_0 * p_1 \Rightarrow p_0 \wedge p_1 \qquad \text{(unsound)}$$

$$p_0 \wedge p_1 \Rightarrow p_0 * p_1 \qquad \text{(unsound)}$$

$$(p_0 * p_1) \vee q \Rightarrow (p_0 \vee q) * (p_1 \vee q) \qquad \text{(unsound)}$$

$$(p_0 \vee q) * (p_1 \vee q) \Rightarrow (p_0 * p_1) \vee q \qquad \text{(unsound)}$$

$$(p_0 * q) \wedge (p_1 * q) \Rightarrow (p_0 \wedge p_1) * q \qquad \text{(unsound)}$$

$$(p_0 * p_1) \wedge q \Rightarrow (p_0 \wedge q) * (p_1 \wedge q) \qquad \text{(unsound)}$$

$$(p_0 \wedge q) * (p_1 \wedge q) \Rightarrow (p_0 * p_1) \wedge q \qquad \text{(unsound)}$$

$$\big(\forall x.\ (p_0 * p_1)\big) \Rightarrow (\forall x.\ p_0) * p_1 \quad \text{when } x \text{ not free in } p_1 \qquad \text{(unsound)}$$

$$(p_0 \Rightarrow p_1) \Rightarrow \big((p_0 * q) \Rightarrow (p_1 * q)\big) \qquad \text{(unsound)}$$

$$(p_0 \Rightarrow p_1) \Rightarrow (p_0 \mathrel{-\!\!*} p_1) \qquad \text{(unsound)}$$

$$(p_0 \mathrel{-\!\!*} p_1) \Rightarrow (p_0 \Rightarrow p_1) \qquad \text{(unsound)}$$