Semantical Analysis of Specification Logic, Part 2

P. W. O'Hearn* R. D. Tennent†

Department of Computing and Information Science
Queen's University
Kingston, Ontario, Canada K7L 3N6.

Information and Computation 107(1), pp25-57, 1993.

*Current address: School of Computer and Information Science, Syracuse University, Syracuse, New York. Supported by a Post-Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada and by the Information Technology Research Centre of Ontario.

†Supported by research grants from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario.

Abstract

The "specification logic" of J. C. Reynolds (1982) is a formal system for proving partial-correctness properties of programs in an Algol-like language with higher-order procedures. In a previous publication (Tennent, 1990), a model was presented that validates all axioms of the logic *except* those that involve non-interference formulas for *procedural* phrases. Following Reynolds, non-interference for procedural phrases was there defined *syntactically*, by induction on types.

Here, we present a new *semantic* interpretation of non-interference for *arbitrary* phrases. It is equivalent to the interpretation given earlier for basic phrases, and it provides the first model for *all* of Reynolds's axioms (or minor corrections thereof). A slightly more refined model is used to validate also a new axiom which formalizes a method used by Reynolds (1981a) to reason about programs with multiple levels of abstraction.

1 Introduction

Specification logic (Reynolds, 1981a, 1982) is a formal system for proving partial-correctness properties of programs in Algol-like languages (Reynolds, 1981b) with higher-order procedures. It is essentially a many-sorted first-order theory with programming-language types such as **comm** (command), **exp**[**int**] (integer expressions), **var**[**int**] (integer variables), and **assert** (assertions) as the sorts, and Hoare triples $\{P\}\,C\,\{Q\}$ as atomic formulas. (The "top-level" formulas, such as Hoare triples, are termed *specifications*, whereas the pre and post-condition formulas P and Q in the Hoare triple specification $\{P\}\,C\,\{Q\}$ are termed *assertions*.)

The use of conjunction (&) and implication ($\Rightarrow$) as specification connectives and quantifiers ($\forall$) at the level of specifications greatly clarifies the treatment of procedures, especially higher-order procedures and phrases with free procedure identifiers, because it is possible to write specifications about free procedure identifiers. Some related approaches are described in (Hoare, 1971; Clarke, German and Halpern, 1983; Damm and Josko, 1983; Olderog, 1984a, 1984b; Sieber, 1985; German, Clarke and Halpern, 1989).

The most distinctive feature of the system is the use of *non-interference* specifications. Informally, $C \# E$ asserts that every way of using C preserves any value obtained by using E. Notice that this notion of non-interference is *asymmetric*: C is a possibly interfering phrase, and E is a phrase possibly interfered with. For example, if C is a command and E is an expression, $C \# E$ means that the value of E is invariant throughout any (terminating) execution of C; if C is a variable, $C \# E$ means that no assignment to C can interfere with E; and if C is a procedure, $C \# E$ means that no call of C can interfere with E other than by using an argument that interferes with E. The main contribution of this paper is to provide a rigorous semantic interpretation of $C \# E$ for *arbitrary* phrases C and E.

Before addressing the semantics of non-interference in detail, we discuss the role of non-interference formulas in the formal system of specification logic. The most fundamental application is to permit Hoare-like reasoning about assignments in languages with aliasing, non-local variables in procedures, and variable phrases containing expressions. As is well-known, the familiar substitution-based axiom proposed in (Hoare, 1969) is not sound in

general in such languages. For example,

$$\{x = 1\}\, y := y + 1\, \{x = 1\}$$

fails if variable identifiers x and y are *aliased*, that is to say, denote the *same* storage variable. Less obviously,

$$\{f(2) = 1\}\, y := y + 1\, \{f(2) = 1\}$$

can fail if y is a non-local variable in the definition of expression procedure f. Finally, if a is an array variable,

$$\{\mathbf{true}\}\, a[a[i]] := 0\, \{a[a[i]] = 0\}$$

may fail when $a[i] = i \neq 0$ because the assignment changes the value of $a[i]$, and then the value of $a[a[i]]$ need not be the value (0) just assigned to it; such a variable phrase is termed a *bad* variable.

Reynolds proposed that, in the absence of *syntactic* restrictions that preclude such anomalies (Reynolds, 1978), assignments can be soundly reasoned about by requiring that appropriate *hypotheses* about the variable and the pre and post-condition assertions be proved in the logic. The law for assignment commands in specification logic is as follows:

Assignment:

$$\mathbf{gv}(V)\ \ \&\ \ V \,\#\, P\ \ \Rightarrow\ \ \{P(E)\}\, V := E\, \{P(V)\}$$

where $\mathbf{gv}(V)$ is a specification asserting that V is a *g*ood *v*ariable, $E\colon \mathbf{exp}[\tau]$ (i.e., an expression for values of data type τ), and $P\colon \mathbf{exp}[\tau] \longrightarrow \mathbf{assert}$ (i.e., a procedure whose arguments are expressions for τ values and whose calls are assertions). The consequent here is essentially like Hoare's axiom, but the good-variable and non-interference hypotheses prevent unsound inferences when V is a bad variable or when assignments to V can *indirectly* affect the truth value of the pre and post-conditions. For example, the invalid specifications discussed above are not derivable because, respectively, $y \,\#\, x$ fails when x and y are aliased, $y \,\#\, f(2)$ fails when y is a non-local variable in the definition of f, and $\mathbf{gv}(a[a[i]])$ fails when $a[i] = i$.

In fact, Reynolds *defined* the specification $\mathbf{gv}(V)$ as follows:

$$\forall p\colon \mathbf{exp}[\tau] \longrightarrow \mathbf{assert}.\ \forall e\colon \mathbf{exp}[\tau].$$
$$V \# p \Rightarrow \{p(e)\}\, V := e\, \{p(V)\}$$

(where identifiers p and e are not free in V). In effect, this defines a good variable to be one for which Hoare-style reasoning is possible whenever there is no inappropriate interference.

A second role for non-interference formulas is to allow reasoning about *invariance* properties of phrases, as in the following axiom:

Constancy:

$$C \# R \ \&\ \Big(\{R\} \Rightarrow \{P\}\, C\, \{Q\}\Big) \ \Rightarrow\ \{P \ \mathbf{and}\ R\}\, C\, \{Q \ \mathbf{and}\ R\}$$

where C is a command, P, Q, and R are assertions and $\{R\}$ is an abbreviation for the Hoare triple $\{\mathbf{true}\}\,\mathbf{skip}\,\{R\}$; i.e., R is *true* in *all* states. This axiom asserts that, if no execution of C interferes with assertion R and if R holds before an execution of C, then R will continue to hold *throughout* the execution, and so it is possible to treat R as a "local" mathematical fact in partial-correctness reasoning about C.

For example, suppose that C is a binary-search algorithm and R asserts that the array being searched is sorted; it is obvious by inspection that $C \# R$, and so the axiom justifies "factoring" R out of the pre and post-conditions for the algorithm and *also* assuming R whenever necessary in the verification of C.

Notice that this axiom requires a "strong" interpretation of $C \# R$: the value of R must remain constant *throughout* any execution of C. With a weaker definition of non-interference, such as

$$C \# R \iff \{R\}\, C\, \{R\} \ \&\ \{\mathbf{not}\ R\}\, C\, \{\mathbf{not}\ R\},$$

only the following weaker form of Constancy would be valid:

$$C \# R \ \&\ \{P\}\, C\, \{Q\} \ \Rightarrow\ \{P \ \mathbf{and}\ R\}\, C\, \{Q \ \mathbf{and}\ R\},$$

and R could not be used as an assumption in reasoning about C.

Similarly, the following axiom requires a strong interpretation of non-interference:

Non-Interference Composition:

$$C \# E \ \& \ \Big(X \# E \Rightarrow \{P\}\, C\, \{Q\}\Big) \ \Rightarrow \ \{P\}\, C\, \{Q\}$$

where C is a command, and P and Q are assertions. It asserts that, if C does not interfere with E, then it is possible to assume that *no* phrase interferes with E in partial-correctness reasoning about C. An example that requires use of this axiom is discussed by Reynolds (1982).

In Section 6, we will discuss a new axiom which we call *Non-Interference Abstraction* that formalizes a method of reasoning with "general invariants" used informally in (Reynolds, 1981a). In this axiom also, a non-interference assumption is essential.

Of course, the logic must provide ways of *discharging* non-interference assumptions. The following axiom is a basic tool that allows "decomposing" non-interference assumptions about arbitrary phrases into non-interference assumptions about their free identifiers:

Non-Interference Decomposition:

$$\cdots \&\ c_i \# e_j \ \& \cdots \ \Rightarrow \ C \# E$$

where $\{\ldots, c_i, \ldots\}$ includes all identifiers with free occurrences in C, and $\{\ldots, e_j, \ldots\}$ includes all identifiers with free occurrences in E. Essentially, this axiom asserts that all channels of interference are named by identifiers; in particular, $C \# E$ when either C or E is *closed*.

This decomposition can be refined by taking into account that all phrases of certain types are *passive*, in that they cannot interfere with anything, and that all phrases of some types are *chaste*, in that they cannot be interfered with. Then, $C \# E$ whenever C is a phrase of passive type or E is a phrase of chaste type.

Finally, there is a way to discharge good-variable and non-interference assumptions about *locally declared* variables; for clarity, we present it as a natural-deduction inference rule:

Local-Variable Declaration:

$$\frac{\begin{bmatrix} \mathbf{gv}(v) \\ v \# E_i \\ C_j \# v \end{bmatrix} \\ \vdots \\ \{P\}\, C\, \{Q\}}{\{P\}\, \mathbf{new}[\tau]\, v.\, C\, \{Q\}}$$

where the bound variable identifier v is not free in assertions P or Q, in phrases E_i or C_j, or in uncancelled assumptions. Essentially, this states that the identifier declared by a local-variable declaration denotes a variable that

- is a *good* variable;
- does not interfere with non-local entities (the E_i); and
- is not interfered with by non-local entities (the C_j).

We may now turn to an informal consideration of the semantics of non-interference. A satisfactory interpretation of $C \# E$ was presented in (Tennent, 1990) for the cases when C is a variable or command and E is an expression or assertion; however, *procedural* phrases were problematical. Reynolds treated procedural phrases by interpreting $C \# E$ *syntactically*, using an induction on types, by strengthening the following laws derivable from Non-Interference Decomposition: if $C{:}\theta \longrightarrow \theta'$ and identifier c is not free in C or E,

$$C \# E \;\Rightarrow\; \big(\forall c{:}\theta.\, c \# E \Rightarrow C(c) \# E\big),$$

and, if $E{:}\theta \longrightarrow \theta'$ and identifier e is not free in C or E,

$$C \# E \;\Rightarrow\; \big(\forall e{:}\theta.\, C \# e \Rightarrow C \# E(e)\big).$$

Essentially, these assert that if a procedure does not interfere with (or, respectively, is not interfered with by) a phrase, no call of the procedure will do so unless the argument does. Reynolds replaced the main $\Rightarrow$ connective by $\Longleftrightarrow$ and adopted the resulting equivalences as *definitions* of procedural non-interference.

However, as shown in (Tennent, 1990), this approach to procedural non-interference is *not* satisfactory, essentially because the procedure is unrestricted when its argument does interfere with (or, respectively, is interfered with by) the other phrase. For example, consider the procedure of type **comm** $\rightarrow$ **comm** defined by

$$P \;=\; \lambda c\colon \mathbf{comm}.\, \mathbf{if}\; i = 1\; \mathbf{then}\; (c\,;\, \mathbf{if}\; i > 1\; \mathbf{then}\; i := 0)$$

where i is a non-local integer-variable identifier. According to Reynolds's definition, $P \,\#\, i$ because the assignment to i in the procedure body is only executed if c has already interfered with i. Intuitively, however, this procedure *does* interfere with i; in particular, $P \,\#\, (i > 0)$ is false because

$$(\mathbf{if}\; i > 0\; \mathbf{then}\; i := i + 1) \;\#\; (i > 0)$$

is true and

$$P(\mathbf{if}\; i > 0\; \mathbf{then}\; i := i + 1) \;\#\; (i > 0)$$

is false. In short, non-interference is not a *logical* relation (Plotkin, 1980; Statman, 1985); nevertheless, we shall show that a very satisfactory interpretation *is* possible.

The interpretation of $C \,\#\, E$ in (Tennent, 1990) (for the case that $C\colon \mathbf{comm}$ and $E\colon \mathbf{exp}[\tau]$) is based on comparing

- executions of C that are *restricted to subsets* of the set of states for which the value of E is invariant, with
- *unrestricted* executions of C.

If the restricted executions are less defined than the corresponding restricted executions, this difference is attributable to C interfering with E. The new approach to be presented in detail in this paper generalizes this to *arbitrary* phrases C by comparing

- uses of C constrained to remaining within the equivalence classes of states for which the value of E is invariant, with
- unconstrained uses of C.

This is equivalent to the approach used previously if C is a *command* or a *variable*, but if C is a *procedural* phrase, its use can be constrained to remaining within an equivalence class without the parameter being so constrained, and so examples such as P above are correctly treated.

The further generalization to allow for procedural E is then straightforward using the fact, proved in (Tennent, 1990), that, if a phrase produces a value at some state, the value depends *only* on the values that can be produced by the meanings of the free identifiers of the phrase at the *same* state.

2 Categories and Functors

To realize the intuitive concept of "execution locally constrained to stay within an equivalence class of states," we adopt a category-theoretic form of possible-world semantics, as in (Reynolds, 1981b; Oles, 1982, 1985; Tennent, 1986, 1990). Each phrase type θ is interpreted as a *functor* $[\![\theta]\!]$ from a (small) category $\mathbf{X}$ of "possible worlds" to either $\mathbf{D}$, the category of directed-complete partial orders and continuous functions (for "computational" phrase types), or $\mathbf{S}$, the category of sets and arbitrary functions (for "logical" phrase types), and similarly for every phrase-type assignment π. Then, for any phrase X that is well-formed and of type θ in the context of phrase-type assignment π (notation: $\pi \vdash X{:}\theta$), we interpret X as a *natural transformation* $[\![X]\!]_{\pi\theta}{:}[\![\pi]\!] \stackrel{\cdot}{\longrightarrow} [\![\theta]\!]$; i.e., the right-hand side of the following diagram commutes for all $f{:}x \stackrel{\mathbf{X}}{\longrightarrow} y$:

$$\begin{array}{ccc} x & & [\![\pi]\!]x \xrightarrow{[\![X]\!]_{\pi\theta}(x)} [\![\theta]\!]x \\ f\downarrow & & [\![\pi]\!]f\downarrow \qquad\qquad \downarrow[\![\theta]\!]f \\ y & & [\![\pi]\!]y \xrightarrow{[\![X]\!]_{\pi\theta}(y)} [\![\theta]\!]y \end{array}$$

The category $\mathbf{X}$ of possible worlds used in (Tennent, 1990) is appropriate here as well. It is defined as follows.

- The objects are sets, thought of as sets of states. We assume that the set of all worlds (i.e., $\mathbf{X}$-objects) is closed under the following:
  - if V_τ is the set of values appropriate to a data type τ, V_τ is a world;
  - if X and Y are worlds, so is $X \times Y$; and
  - if X is a world, so is any $Y \subset X$.
- A morphism from X to Y is a pair (f, Q), where Q is an equivalence relation on Y and f is a function from Y to X that is injective on Q-equivalence classes.

The composition (in diagrammatic order) $(f, Q)\,;(g, R)\colon X \xrightarrow{\mathbf{X}} Z$ of morphisms $(f, Q)\colon X \xrightarrow{\mathbf{X}} Y$ and $(g, R)\colon Y \xrightarrow{\mathbf{X}} Z$ is the unique morphism (h, P) where $h = g\,;f$ and $z\,P\,z'$ iff $z\,R\,z'$ and $g(z)\,Q\,g(z')$.

Here are descriptions of some **X**-morphisms.

- The *identity* morphism on world X, $\mathrm{id}_X\colon X \xrightarrow{\mathbf{X}} X$, is (I_X, T_X) where I_X is the identity function on X and T_X is the everywhere-true binary relation on X.
- If R is an equivalence relation on a world X, the *state-change constraint* endomorphism $\downarrow R\colon X \xrightarrow{\mathbf{X}} X$ is (I_X, R).
- If X' is a subset of a world X, the *state-set restriction* morphism $\lceil X'\colon X \xrightarrow{\mathbf{X}} X'$ is $(f, T_{X'})$, where f is the inclusion function from X' to X and $T_{X'}$ is the everywhere-true relation on X'.
- If $\{s_0\}$ and $\{s_1\}$ are singleton worlds, $s_0 \mapsto s_1$ denotes the unique morphism from $\{s_0\}$ to $\{s_1\}$; i.e., (f, Q) where f is the only function from $\{s_1\}$ to $\{s_0\}$ and Q is the identity relation on $\{s_1\}$. This morphism is in fact an isomorphism in **X**.
- If X and Y are worlds, the *expansion* morphism $\times Y\colon X \xrightarrow{\mathbf{X}} X \times Y$ is (f, Q), where f is the projection function from $X \times Y$ to X, and $(x, y)\,Q\,(x', y')$ iff $y = y'$.
- If X and Y are worlds, $X \leftrightarrow Y\colon X \times Y \xrightarrow{\mathbf{X}} Y \times X$ is the isomorphism $(f, T_{Y\times X})$, where $f(y, x) = (x, y)$.

We refer the reader to (Tennent, 1990) for definitions of the functors $[\![\theta]\!]$ for all of the various phrase types θ, and $[\![\pi]\!]$ for any phrase-type assignment π. We mention only that

- **1**, $\times$, and $\longrightarrow$ refer, respectively, to the specified initial object, product, and exponentiation in the cartesian-closed functor category $\mathbf{D}^{\mathbf{X}}$;
- S is the *contravariant* functor from **X** to **D** such that $S(X) = X$, discretely-ordered, and $S(f, Q) = f$; and

- an element of $[\![\mathbf{comm}]\!]X$ is a *family* $c(\cdot)$, indexed by $\mathbf{X}$-morphisms $(f, Q)\colon X \longrightarrow Y$, such that $c(f, Q)$ is a partial function on Y and the following diagram commutes for every $y \in Y$:

$$\begin{array}{ccc} Y & \xrightarrow{\quad c(f,Q) \quad} & Y \\ {\scriptstyle S(\lceil Y' \rceil)}\uparrow & & \uparrow{\scriptstyle S(\lceil Y' \rceil)} \\ Y' & \xrightarrow{\quad c\big((f,Q)\,;\lceil Y' \rceil\big) \quad} & Y' \end{array}$$

where $Y' = \{y' \in Y' \mid y' \mathrel{Q} y\}$. The commutativity ensures that every terminating execution of $c(f, Q)$ is constrained to remain within a Q-equivalence class of states, so that any attempt to change state across an equivalence-class boundary fails to terminate. Of course, this is also the case when $Y = X$ and (f, Q) is an *endo*morphism, such as $\downarrow R$ for any equivalence relation R on X.

3 Semantics of Non-Interference

In this section, we define relations $\# \subseteq C(X) \times E(X)$ for arbitrary functors $C, E: \mathbf{X} \longrightarrow \mathbf{D}$ and world X, and then use these to interpret non-interference specifications in specification logic.

Two key ideas underlie our treatment of non-interference. The first is that the use of state-change constraint endomorphisms in $\mathbf{X}$ allows us to constrain semantical entities to executing in equivalence classes of states, without *changing* worlds. If $c \in C(X)$ and R is an equivalence relation on X, $C(\downarrow R)(c)$ is a restriction of c such that, when c causes a state change from s_0 to s_1 for which $s_0 \, R \, s_1$ fails, $C(\downarrow R)(c)$ diverges. Notice that because $C(\downarrow R)(c)$ and c are both elements of $C(X)$, they can be compared for equality.

For example, if R_i is the equivalence relation on X that relates states for which variable i has the same value, and u is an appropriate (in that i is good) environment,

$$\begin{aligned} [\![\mathbf{comm}]\!]\big(\downarrow R_i)([\![i := 0]\!]Xu\big) &= [\![\mathbf{if}\ i \neq 0\ \mathbf{then\ diverge}]\!]Xu \\ &\neq [\![i := 0]\!]Xu. \end{aligned}$$

An attempt to *change* the value of i in an execution constrained by R_i leads to divergence.

As an example of the effect obtained with procedural phrases, consider again

$$P \;=\; \lambda c{:}\,\mathbf{comm.\,if}\ i = 1\ \mathbf{then}\ (c\,;\mathbf{if}\ i > 1\ \mathbf{then}\ i := 0).$$

Let $\theta = \mathbf{comm} \longrightarrow \mathbf{comm}$; by naturality of $[\![P]\!]$, we have the commutativity of

$$\begin{array}{ccc} X & & [\![\pi]\!]X \xrightarrow{[\![P]\!]X} [\![\theta]\!]X \\ {\scriptstyle \downarrow R_i}\big\downarrow & & [\![\pi]\!](\downarrow R_i)\big\downarrow \qquad\qquad \big\downarrow [\![\theta]\!](\downarrow R_i) \\ X & & [\![\pi]\!]X \xrightarrow{[\![P]\!]X} [\![\theta]\!]X \end{array}$$

for any phrase-type assignment π such that $\pi(i) = \mathbf{var}[\mathbf{int}]$, and so, for appropriate $u \in [\![\pi]\!]X$, it follows that $[\![\theta]\!](\downarrow R_i)\Big([\![P]\!]Xu\Big)$ is equal to

$$[\![\lambda c\colon \mathbf{comm}.\ \mathbf{if}\ i = 1\ \mathbf{then}\ (c\,;\ \mathbf{if}\ i > 1\ \mathbf{then}\ \mathbf{diverge})]\!]Xu.$$

The state-change constraint allows us to (semantically) replace the interference with i in the assignment $i := 0$ by divergence; however, interference with i that comes from an argument c is unaffected because the identifier c is locally bound in P.

The second key idea is that, for any functor $E\colon \mathbf{X} \to \mathbf{D}$, world X, and $e \in E(X)$, we can define an equivalence relation $[e] \subseteq X \times X$ such that, intuitively, $s[e]s'$ if and only if the effect of using e in an expression evaluated at state s is equivalent to the effect of using e in that expression evaluated at s'. This is possible because of the result in (Tennent, 1990) that shows that expressions do not cause side-effects, not even "temporary" ones: the value of $e \in [\![\mathbf{exp}[\tau]]\!]X$ at a state s is equal to the value of $[\![\mathbf{exp}[\tau]]\!]\Big(\lceil\{s\}\Big)(e)$ at s.[1] This means that whenever an arbitrary $e \in E(X)$ is used in an expression evaluated at s, only the restricted part $E\Big(\lceil\{s\}\Big)(e) \in E\Big(\{s\}\Big)$ is relevant.

To obtain the desired equivalence relation, we should compare transformed meanings of the form $E\Big(\lceil\{s\}\Big)(e)$ and $E\Big(\lceil\{s'\}\Big)(e)$. In the category $\mathbf{X}$ of possible worlds, there is exactly one morphism $s' \mapsto s$ from the world $\{s'\}$ to $\{s\}$, and it is an isomorphism. Therefore, the map

$$E\Big(\{s'\}\Big) \xrightarrow{\quad E(s' \mapsto s)\quad} E\Big(\{s\}\Big)$$

is an isomorphism. $E\Big(\lceil\{s'\}\,;\, s' \mapsto s\Big)(e)$ is a meaning at the world $\{s\}$, but the values of (non-local) variables are evaluated as if the state were s'.

We therefore define the relation $[e]$ as follows:

$$s[e]s' \iff E\Big(\lceil\{s\}\Big)(e) = E\Big(\lceil\{s'\}\,;\, s' \mapsto s\Big)(e)$$

In the case that $X = \emptyset$, $[e]$ is the empty binary relation. It is straightforward to verify that $[e]$ is an equivalence relation, and the following can be proved by a routine calculation.

[1] The morphism $\lceil\{s\}$ precludes side-effects (to non-local variables) because there are no other states in $\{s\}$ to change to!

Proposition 1 *For any functor* $E\colon \mathbf{X} \to \mathbf{D}$, *morphism* $f\colon X \xrightarrow{\mathbf{X}} Y$, *and* $e_X \in E(X)$,

$$f\,;\downarrow[e_Y] \;=\; \downarrow[e_X]\,;f \;=\; \downarrow[e_X]\,;f\,;\downarrow[e_Y],$$

where $e_Y = E(f)(e_X)$.

The definition of $[e]$ is fairly abstract. To interpret specification logic, it can be presented more concretely by induction on types; we illustrate this with two sample cases.

- If $E = [\![\mathbf{exp}[\tau]]\!]$, $s[e]s'$ if and only if $e(\mathrm{id}_X)(s) \;=\; e(\mathrm{id}_X)(s')$.
- If $E = [\![\mathbf{exp}[\tau] \to \mathbf{exp}[\tau]]\!]$, $s[e]s'$ if and only if, for all $a \in [\![\mathbf{exp}[\tau]]\!]X$, $s[e(\mathrm{id}_X)(a)]s'$ whenever $s[a]s'$.

We will work with the abstract definition because it leads to simpler and more general proofs without induction on types.

We may finally define $\# \subseteq C(X) \times E(X)$ for any functors $C, E\colon \mathbf{X} \to \mathbf{D}$ and world X as follows:

$$c \,\#\, e \iff C\Big(\downarrow[e]\Big)(c) = c$$

The notations $c \,\#\, e$ and $[e]$ do not explicitly indicate the relevant functors (C, E) or object (X), but these will always be clear from context.

Proposition 2 *For any* $C, E\colon \mathbf{X} \to \mathbf{D}$, $f\colon X \xrightarrow{\mathbf{X}} Y$, $c \in C(X)$, *and* $e \in E(X)$,

$$\textit{if } c \,\#\, e \textit{ then } C(f)(c) \,\#\, E(f)(e).$$

Proof.

$$\begin{array}{lll}
C\Big(\downarrow\big[E(f)(e)\big]\Big)\Big(C(f)(c)\Big) & & \\
\quad = \quad C\Big(f\,;\downarrow\big[E(f)(e)\big]\Big)(c) & & \text{(functoriality of } C) \\
\quad = \quad C\Big(\downarrow[e]\,;f\Big)(c) & & \text{(Proposition 1)} \\
\quad = \quad C(f)\Big(C\Big(\downarrow[e]\Big)(c)\Big) & & \text{(functoriality of } C) \\
\quad = \quad C(f)(c) & & (c \,\#\, e)
\end{array}$$

□

We can now apply our definition of non-interference to the interpretation of specification logic. We first introduce the following syntax rule:

Non-Interference Specification:

$$\frac{\pi \vdash C{:}\theta \qquad \pi \vdash E{:}\theta'}{\pi \vdash C \mathbin{\#} E{:}\mathbf{spec}}$$

where π is any phrase-type assignment and θ and θ' are arbitrary phrase types (other than **spec**). Then, for any worlds X and Y, environment $u \in [\![\pi]\!]X$, and morphism $f{:}\, X \xrightarrow{\mathbf{X}} Y$,

$$[\![C \mathbin{\#} E]\!]Xuf \;=\; \Big([\![\theta]\!](f)\big([\![C]\!]Xu\big) \;\#\; [\![\theta']\!](f)\big([\![E]\!]Xu\big)\Big)$$

Proposition 2 may be used to prove that $[\![C \mathbin{\#} E]\!]Xu \in [\![\mathbf{spec}]\!]X$, and it is then routine to show that $[\![C \mathbin{\#} E]\!]$ is a natural transformation from $[\![\pi]\!]$ to $[\![\mathbf{spec}]\!]$.

In the following sections, we use this interpretation to show the validity of the axioms discussed in Section 1.

4 Non-Interference Decomposition

We first consider separately the left and right operands of the # operator, and then account for phrases that, because of their types, either cannot interfere with, or cannot be interfered with by, any phrase.

4.1 Leftside Decomposition

Our aim here is to prove the validity of

$$\cdots \& \ c_i \# E \ \& \cdots \ \Rightarrow \ C \# E$$

where $\{\ldots, c_i, \ldots\}$ includes all identifiers with free occurrences in C.

Proposition 3

(a) For any $E{:}\,\mathbf{X} \to \mathbf{D}$, *world* X, *and* $e \in E(X)$,

$$\textit{if } c \in \mathbf{1}(X) \textit{ then } c \# e.$$

(b) For any $C_1, C_2, E{:}\,\mathbf{X} \to \mathbf{D}$, *world* X, $(c_1, c_2) \in (C_1 \times C_2)(X)$, *and* $e \in E(X)$,

$$\textit{if } c_1 \# e \textit{ and } c_2 \# e \textit{ then } (c_1, c_2) \# e.$$

(c) For any $C, C', E{:}\,\mathbf{X} \to \mathbf{D}$, $\eta{:}\,C \stackrel{\cdot}{\to} C'$, *world* X, $c \in C(X)$, *and* $e \in E(X)$,

$$\textit{if } c \# e \textit{ then } \Big(\eta(X)(c)\Big) \# e.$$

Proof.

(a) For *any* constant functor $C{:}\,\mathbf{X} \to \mathbf{D}$ and $f{:}\,X \xrightarrow{\mathbf{X}} X$, $C(f) = I_{F(X)}$, and so

$$\mathbf{1}\Big(\downarrow[e]\Big)(c) \ = \ I_{\mathbf{1}(X)}(c) \ = \ c.$$

(b)
$$\begin{array}{lll} (C_1 \times C_2)\Big(\downarrow[e]\Big)(c_1, c_2) & & \\ = \Big(C_1\Big(\downarrow[e]\Big)(c_1), C_2\Big(\downarrow[e]\Big)(c_2)\Big) & \text{(definition of } \times) \\ = (c_1, c_2) & (c_1 \# e \text{ and } c_2 \# e) \end{array}$$

(c)
$$\begin{array}{rcll} C'\Big(\downarrow[e]\Big)\Big(\eta(X)(c)\Big) & = & \eta(X)\Big(C\Big(\downarrow[e]\Big)(c)\Big) & \text{(naturality of } \eta) \\ & = & \eta(X)(c) & (c \# e) \end{array}$$

□

We can show the validity of the leftside decomposition axiom above as follows. Assume that, for some environment $u \in [\![\pi]\!]X$, $u(c_i) \# e$ for all c_i free in C, where $e = [\![E]\!]Xu$. Let $u' \in [\![\pi]\!]X$ be the restriction of u to the c_i. If C is closed, $u' \in \mathbf{1}(X)$ and so $u' \# e$ by part (a) of the proposition; otherwise, $u' \# e$ by repeated use of part (b). By part (c) and naturality of $[\![C]\!]$, it follows that $([\![C]\!]Xu') \# e$. By the usual "coherence" property of valuations, we have that $[\![C]\!]Xu' = [\![C]\!]Xu$, and we are done.

4.2 Rightside Decomposition

The development here parallels that in the preceding sub-section. We show the validity of the axiom

$$\cdots \& \ C \# e_i \ \& \cdots \ \Rightarrow \ C \# E$$

where $\{\ldots, e_i, \ldots\}$ includes all identifiers with free occurrences in E.

Proposition 4

(a) For any $C\colon \mathbf{X} \to \mathbf{D}$, *world* X, *and* $c \in C(X)$.

$$\textit{if } e \in \mathbf{1}(X) \textit{ then } c \# e.$$

(b) For any $C, E_1, E_2\colon \mathbf{X} \to \mathbf{D}$, *world* X, $(e_1, e_2) \in (E_1 \times E_2)(X)$, *and* $c \in C(X)$,

$$\textit{if } c \# e_1 \textit{ and } c \# e_2 \textit{ then } c \# (e_1, e_2).$$

(c) *For any* $C, E, E' : \mathbf{X} \to \mathbf{D}$, $\eta : E \dot{\to} E'$, *world* X, $c \in C(X)$, *and* $e \in E(X)$,

$$\text{if } c \# e \text{ then } c \# \Big(\eta(X)(e)\Big).$$

Proof.

(a) For *any* constant functor $E : \mathbf{X} \to \mathbf{D}$,

$$\begin{aligned} s[e]s' &\iff E\Big(\downarrow\{s\}\Big)(e) = E\Big(\downarrow\{s'\} \,;\, s' \mapsto s\Big)(e) \\ &\iff I_{E(X)}(e) = I_{E(X)}(e), \end{aligned}$$

so that $\downarrow[e] = \mathrm{id}_X$, and so $C\Big(\downarrow[e]\Big)(c) = I_{C(X)}(c) = c$.

(b) $s[(e_1, e_2)]s' \iff s[e_1]s'$ and $s[e_2]s'$, and so $\downarrow[e_1, e_2] = \downarrow[e_1]; \downarrow[e_2]$. Hence,

$$\begin{aligned} C\Big(\downarrow[e_1, e_2]\Big)(c) &= C\Big(\downarrow[e_1] \,;\, \downarrow[e_2]\Big)(c) \\ &= C\Big(\downarrow[e_2]\Big(C\Big(\downarrow[e_1]\Big)(c)\Big) && \text{(functoriality of } C) \\ &= C\Big(\downarrow[e_2]\Big)(c) && (c \# e_1) \\ &= c && (c \# e_2) \end{aligned}$$

(c) We first prove the following lemma: if $s[e]s'$ then $s[\eta(X)(e)]s'$.

$$\begin{aligned} E\Big(\downarrow\{s\}\Big)\Big(\eta(X)(e)\Big) &= \eta(\{s\})\Big(E\Big(\downarrow\{s\}\Big)(e)\Big) && \text{(naturality of } \eta) \\ &= \eta(\{s\})\Big(E\Big(\downarrow\{s'\} \,;\, s' \mapsto s\Big)(e)\Big) && (s[e]s') \\ &= E\Big(\downarrow\{s'\} \,;\, s' \mapsto s\Big)\Big(\eta(X)(e)\Big) && \text{(naturality of } \eta) \end{aligned}$$

Then

$$\begin{aligned} C\Big(\downarrow[\eta(X)(e)]\Big)(c) &= C\Big(\downarrow[\eta(X)(e)]\Big)\Big(C\Big(\downarrow[e]\Big)(c)\Big) && (c \# e) \\ &= C\Big(\downarrow[\eta(X)(e)] \,;\, \downarrow[e]\Big)(c) && \text{(functoriality of } C) \\ &= C\Big(\downarrow\Big([\eta(X)(e)] \cap [e]\Big)\Big)(c) && \text{(composition in } \mathbf{X}) \\ &= C\Big(\downarrow[e]\Big)(c) && \text{(lemma)} \\ &= c && (c \# e) \end{aligned}$$

□

Validation of the rightside decomposition axiom follows the same pattern as that for leftside decomposition in the preceding sub-section, and is omitted.

4.3 Passive and Chaste Types

The decomposition axioms in (Reynolds, 1982) take into account the fact that certain phrases are not "command-like" and cannot cause interference, or are not "expression-like" and cannot be interfered with. Our treatment of these issues here is closer to the approach used in (Reynolds, 1978).

A phrase type θ is termed *passive* just if $c\#e$ for every functor $E\colon \mathbf{X} \longrightarrow \mathbf{D}$, world X, $c \in [\![\theta]\!]X$, and $e \in E(X)$. Intuitively, a phrase having a passive type cannot interfere with anything.

Proposition 5

(a) $\mathbf{val}[\tau]$ (Tennent, 1990), $\mathbf{exp}[\tau]$, *and* $\mathbf{assert}$ *are passive.*

(b) If θ' *is passive, so is* $\theta \longrightarrow \theta'$ *for every phrase type* θ.

Proof.

(a) $\mathbf{val}[\tau]$ is passive because $[\![\mathbf{val}[\tau]]\!]$ is a constant functor. We prove the result for $\mathbf{assert}$; the proof for $\mathbf{exp}[\tau]$ is similar. For any $f\colon X \xrightarrow{\mathbf{X}} Y$,

$$\begin{array}{rcll} [\![\mathbf{assert}]\!]\big(\downarrow[e]\big)(c)(f) & = & c\big(\downarrow[e]\,;f\big) & \text{(morphism part of } [\![\mathbf{assert}]\!]) \\ & = & c(f) & \text{(uniformity for } S \longrightarrow [\![\mathbf{bool}]\!]) \end{array}$$

(b) Consider any worlds X and Y, $p \in [\![\theta \longrightarrow \theta']\!]X$, $f\colon X \xrightarrow{\mathbf{X}} Y$, $a \in [\![\theta]\!]Y$, and $e_X \in E(X)$. Let $e_Y = E(f)(e_X)$; then

$$\begin{array}{rcll} \Big([\![\theta \longrightarrow \theta']\!]\big(\downarrow[e_X]\big)(p)\Big)(f)(a) & & & \\ = & p\big(\downarrow[e_X]\,;f\big)(a) & & \text{(morphism part of } [\![\theta \longrightarrow \theta']\!]) \\ = & p\big(f\,;\downarrow[e_Y]\big)(a) & & \text{(Proposition 1)} \\ = & [\![\theta']\!]\big(\downarrow[e_Y]\big)\Big(p\big(f\,;\downarrow[e_Y]\big)(a)\Big) & & (\theta' \text{ is passive}) \\ = & p\big(f\,;\downarrow[e_Y]\,;\downarrow[e_Y]\big)\Big([\![\theta]\!]\big(\downarrow[e_Y]\big)(a)\Big) & & \text{(uniformity for } [\![\theta \longrightarrow \theta']\!]) \\ = & p\big(f\,;\downarrow[e_Y]\big)\Big([\![\theta]\!]\big(\downarrow[e_Y]\big)(a)\Big) & & \text{(Proposition 1)} \\ = & [\![\theta']\!]\big(\downarrow[e_Y]\big)\big(p(f)(a)\big) & & \text{(uniformity for } [\![\theta \longrightarrow \theta']\!]) \\ = & p(f)(a) & & (\theta' \text{ is passive}) \end{array}$$

□

Similarly, a phrase type θ is termed *chaste* just if $c \# e$ for every functor $C\colon \mathbf{X} \to \mathbf{D}$, world X, $c \in C(X)$, and $e \in [\![\theta]\!]X$. Intuitively, a phrase having a chaste type cannot be interfered with by anything.

Proposition 6

(a) $\mathbf{val}[\tau]$ *is chaste.*

(b) If θ' *is chaste, so is* $\theta \to \theta'$ *for every phrase type* θ*.*

Proof.

(a)

(b)

The definitions of passive and chaste types justify the following axiom: if $\pi \vdash C\colon\theta$ and $\pi \vdash E\colon\theta'$, $C \# E$ whenever θ is passive or θ' is chaste. We can now use the rule of free substitution to derive the following "practical" form of Non-Interference Decomposition:

$$\cdots \,\&\; c_i \# e_j \;\&\, \cdots \;\Rightarrow\; C \# E$$

where $\{\ldots, c_i, \ldots\}$ is the set of identifiers having at least one free occurrence in C that is outside of any sub-phrase of passive type, and $\{\ldots, e_j, \ldots\}$ is the set of identifiers having at least one free occurrence in E that is outside of any sub-phrase of chaste type. Identifier occurrences are themselves sub-phrases and so no c_i has passive type and no e_j has chaste type. Also, C and E are sub-phrases of themselves, and so it follows that $C \# E$ when C has a passive type or E has a chaste type.

In (Reynolds, 1982) commands are regarded as phrases that cannot be interfered with, whereas **comm** is not, in our terminology, chaste. But, to prove that phrases of type **comm** cannot be interfered with, a tedious structural induction over the entire language is necessary; if, for example, block expressions, as in (Tennent, 1986), were added to the language, it would no longer hold. Our definitions of passive and chaste types are based on fundamental semantic properties, rather than details of a specific language design.

5 Local-Variable Declarations

If an identifier of phrase type θ is free in the block

$$\mathbf{new}[\tau]\, x.\, C$$

then its meaning $c' \in [\![\theta]\!](X \times V)$ in the body of the block is $[\![\theta]\!](\times V)(c)$ for some $c \in [\![\theta]\!]X$. That is, c' comes from a non-local world X, so it should be *independent* of the V-valued component of $X \times V$ used to hold the value of the local variable.

On the other hand, the meaning v' of identifier x in C is concerned entirely with the V-valued component, and should be independent of X. One way to view this is to consider v' as coming from the world V, just as c' comes from X, and so v' should not depend on X at all. That is, $v' \in [\![\mathbf{var}[\tau]]\!](X \times V)$ arises from some meaning $v \in [\![\mathbf{var}[\tau]]\!]V$ as

$$v' \;=\; [\![\mathbf{var}[\tau]]\!](\times X\,;\, V \leftrightarrow X)(v)$$

where $\times X : V \xrightarrow{\mathbf{X}} V \times X$; note the exchange resulting from use of

$$V \leftrightarrow X :\; V \times X \xrightarrow{\mathbf{X}} X \times V.$$

The local variables described in (Tennent, 1990) can in fact be obtained in this way (although they are not presented as such there). Therefore, to show that local variables don't interfere with non-local entities, and *vice versa*, it is sufficient to show that $c' \# v'$ and $v' \# c'$ whenever c' and v' arise in this manner.

Proposition 7 *For any non-empty worlds X, Y, functors $C, V : \mathbf{X} \to \mathbf{D}$, $c \in C(X)$, and $v \in V(Y)$, let $c' = C(\times Y)(c)$, and $v' = V(\times X\,;\, Y \leftrightarrow X)(v)$, where $\times Y : X \xrightarrow{\mathbf{X}} X \times Y$ and $\times X : Y \xrightarrow{\mathbf{X}} Y \times X$; then $c' \# v'$ and $v' \# c'$.*

Proof. We will show $c' \# v'$; the proof of $v' \# c'$ is similar. Let the equivalence relation $R \subseteq (X \times Y) \times (X \times Y)$ be defined by $(x, y)\, R\, (x', y')$ if and only if $y = y'$, and let $g : Y \xrightarrow{\mathbf{X}} X \times Y$ be defined by

$$g = \times X\,;\, Y \leftrightarrow X.$$

Firstly, we show that

$$C(\downarrow R)(c') \;=\; c' \tag{1}$$

as follows:

$$\begin{array}{lcll} C(\downarrow R)(c') & = & C(\downarrow R)\Big(C(\times Y)(c)\Big) & \text{(definition of } c'\text{)} \\ & = & C(\times Y\,;\downarrow R)(c) & \text{(functoriality of } C\text{)} \\ & = & C(\times Y)(c) & (\times Y = \times Y\,;\downarrow R) \\ & = & c' & \text{(definition of } c'\text{)} \end{array}$$

Secondly, we show that

$$R \subseteq [v'] \quad ; \tag{2}$$

suppose that $(x,y)\,R\,(x',y)$; then

$$\begin{array}{ll} & V\Big(\lceil\{(x,y)\}\Big)(v') \\ = & V\Big(g\,;\lceil\{(x,y)\}\Big)(v) \qquad\qquad \text{(definition of } v'\text{)} \\ = & V\Big(g\,;\lceil\{(x',y)\}\,;(x',y)\mapsto(x,y)\Big)(v) \\ = & V\Big(\lceil\{(x',y)\}\,;(x',y)\mapsto(x,y)\Big)(v') \qquad \text{(definition of } v'\text{)} \end{array}$$

where the intermediate step follows from the fact that

$$g\,;\lceil\{(x,y)\} \;=\; g\,;\lceil\{(x',y)\}\,;(x',y)\mapsto(x,y).$$

Finally, to show that $c' \,\#\, v'$:

$$\begin{array}{lcll} C\Big(\downarrow[v']\Big)(c') & = & C\Big(\downarrow[v']\Big)\Big(C(\downarrow R)(c')\Big) & \text{(by 1)} \\ & = & C\Big(\downarrow R\,;\downarrow[v']\Big)(c') & \text{(functoriality of } C\text{)} \\ & = & C(\downarrow R)(c') & \text{(by 2)} \\ & = & c' & \text{(by 1)} \end{array}$$

□

In the empty world, $c \,\#\, v$ trivially, because the only endomorphism on the empty set is the identity.

We have shown that local variables (however these might be defined, so long as they are truly *local*) do not interfere with non-local entities, and *vice versa*. To validate the local-variable declaration axiom, it remains only to show that a local variable is a *good* variable; a proof of this may be found

in (Tennent, 1990), and the proof is still valid for our new interpretation of non-interference.

Similarly, the remaining axioms of the system described in (Reynolds, 1982) were validated in (Tennent, 1990) and remain valid with our new interpretation of non-interference because of the following result.

Proposition 8 *If* $\pi \vdash C\colon \mathbf{comm}$ *and* $\pi \vdash E\colon\theta$, *then, for all worlds* X *and* Y, $u \in [\![\pi]\!]X$ *and* $f\colon X \xrightarrow{\mathbf{X}} Y$, $[\![C \# E]\!]Xuf$ *if and only if, for all* $g\colon Y \xrightarrow{\mathbf{X}} Z$ *and* $z \in Z$,

$$\textit{if } [\![C]\!]Xu(f\,;g)z = z' \textit{ then } [\![C]\!]Xu(f\,;g\,;\lceil Z_e)z = z'$$

where $Z_e = \{z_e \in Z \mid z_e[e]z\}$ *for* $e = [\![\theta]\!](f\,;g)\Big([\![E]\!]Xu\Big)$.

This shows that, at base types, our new interpretation of # is equivalent to that in (Tennent, 1990). The proof is immediate from the definitions of $[\![\mathbf{comm}]\!]$ and $[\![C \# E]\!]$.

6 Non-Interference Abstraction

When writing a program $P(C)$ that uses an operation C that manipulates an abstract type, it is often convenient to view the program from the level of abstraction where C is considered as an indivisible action. If R is an invariant of C and P doesn't affect R except through using C then, *at this level of abstraction*, R is true throughout execution of $P(C)$ if it is true initially. In (Reynolds, 1981a), this was used as intuitive justification for a method of reasoning where invariant relationships, like R, were used as "local" mathematical facts when reasoning about an abstract program.

This kind of reasoning is similar to that provided by the constancy axiom, in that an assertion is factored out by treating it as a local mathematical fact. However, constancy would require a hypothesis of the form $P(C) \# R$, which would not be justified because

- on the level of the *implementation* of C, R might be false at an intermediate state because we only assume $\{R\}\, C\, \{R\}$; and
- even if C is indivisible we need not have $P(C) \# R$, because C is only required to preserve the truth, and not the falsity, of R.

In this section we present a new axiom that also uses a local assumption, but in a way that does not require it to be true on the level of all concrete operations. The axiom is as follows:

Non-Interference Abstraction:

$$\begin{array}{l} P \# R \;\;\&\;\; \{R\}C\{R\} \;\&\; \{Q\}C\{D\} \;\& \\ (\forall \iota : \mathbf{comm} \,.\, \{R\} \;\&\; \{Q\}\iota\{D\} \;\Rightarrow\; \{A\}P(\iota)\{B\}\,) \\ \qquad\qquad\qquad\qquad \Rightarrow\; \{R \wedge A\}P(C)\{R \wedge B\} \end{array}$$

where $P : \mathbf{comm} \to \mathbf{comm}$, $C : \mathbf{comm}$, and ι is not free in P,C, or the assertions A, B, R, D or Q. The idea is that if $P \# R$, $\{R\}C\{R\}$, and R is true before executing $P(C)$ then, throughout execution of $P(C)$, R should hold continuously *at the level of abstraction* where C is considered as an

indecomposable action. When treating R as a mathematical fact, it is essential to confine reasoning about $P(C)$ to be at this same level of abstraction, because R might actually be false at an internal state of execution of C. Non-interference abstraction ensures that reasoning is carried out on this level by using an identifier ι which, unlike C, doesn't have any "parts" to give us information about intermediate states. Furthermore, the meaning of ι is arbitrary and, for all we know, could be that of an indivisible action because it is universally quantified. The specification $\{R\}$ is essentially a local "atomicity" assumption.

A number of examples of reasoning that can be formalized by this axiom, or minor variations thereof, can be found in chapter 5 of (Reynolds, 1980a). Typically, the assertion R describes an invariant relationship between concrete values of a representation and the abstract values being represented and C is an operation that manipulates the values in a way that preserves the invariant.

The axiom requires a slight refinement to our model. We modify the category $\mathbf{X}$ so that morphisms $(f, R) : X \xrightarrow{\mathbf{X}} Y$ are such that

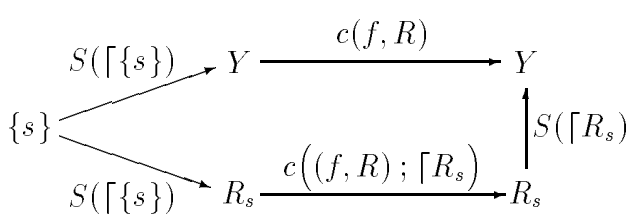
- R is a reflexive and transitive relation on Y, and
- $f : Y \to X$ is a function that, for each $s \in Y$, is injective on the set $R_s = \{s' \mid sRs'\}$ of states reachable from s via R.

Command meanings are also modified:

- $c(\cdot) \in [\![\mathbf{comm}]\!]X$ is again family of partial functions, but with the requirement that, for all $(f, R) : X \xrightarrow{\mathbf{X}} Y$ and $s \in Y$,

  $$c\,(f, R)\,s \;=\; c\Big((f; R)\,;\,\lceil R_s\Big)\,s.$$

  As a diagram,



Elements of $[\![\mathbf{comm}]\!]X$ are ordered pointwise, so that $c_1 \sqsubseteq c_2$ iff for all $f : X \xrightarrow{\mathbf{X}} Y$ and $s \in Y$, $c_1\, f\, s_1 = c_2\, f\, s_2$ whenever $c_1\, f\, s$ is defined.

The injectivity requirement on morphisms is needed for the definition of local variables, as in (Tennent, 1990). The definition of composition of **X**-morphisms works in this modified category, and all of our previous definitions and results go through.

This refined category allows the following weak variation on non-interference: if $C : \mathbf{X} \longrightarrow \mathbf{D}$, X is a world, $c \in C(X)$, and $r \in [\![\mathbf{assert}]\!]X$, then

$$c \dashv r \iff C\Big(\downarrow\langle r\rangle\Big)(c) = c$$

where, for all $s_1, s_2 \in X$,

$$s_1\langle r\rangle s_2 \iff r\,\mathrm{id}_X\, s_1 = \mathit{false} \text{ or } r\,\mathrm{id}_X\, s_2 = \mathit{true}$$

Intuitively, $c \dashv r$ means that all ways of using c preserve the truth, but not necessarily the falsity, of r. Notice that the relation $\langle r\rangle$ is reflexive and transitive, but not symmetric.

Let $\{q\}c\{r\}$ be the semantical counterpart of the Hoare triple; that is, for $q, r \in [\![\mathbf{assert}]\!]X$ and $c \in [\![\mathbf{comm}]\!]X$,

$$\{q\}c\{r\} \iff \forall\, s_1, s_2 \in X,\ q\,\mathrm{id}_X\, s_1 = \mathit{true} \ \wedge\ c\,\mathrm{id}_X\, s_1 = s_2 \Rightarrow r\,\mathrm{id}_X\, s_2 = \mathit{true}$$

Proposition 9

(a) For all worlds X, $c \in [\![\mathbf{comm}]\!]X$, and $q, s, r \in [\![\mathbf{assert}]\!]X$,

$$\textit{if } \{r\}c\{r\} \textit{ and } \{q\}c\{s\} \textit{ then } \exists c' \in [\![\mathbf{comm}]\!]X \textit{ such that}$$
$$c \sqsubseteq c',\ \{q\}c'\{s\} \textit{ and } c' \dashv r.$$

(b) For all worlds X, functors $C : \mathbf{X} \longrightarrow \mathbf{D}$, $c \in C(X)$, and $r \in [\![\mathbf{assert}]\!]X$,

$$\textit{if } c \# r \textit{ then } c \dashv r.$$

(c) For all worlds X, $r \in [\![\mathbf{assert}]\!]X$, $p \in [\![\theta \to \theta']\!]X$ and $c \in [\![\theta]\!]X$,

$$p \dashv r \textit{ and } x \dashv r \textit{ then } (p\,\mathrm{id}_X\, c) \dashv r.$$

Proof.

(a) Define $c' \in [\![\mathbf{comm}]\!]X$ by

$$c'\,(f,Q)\,s = \begin{cases} s' & \textit{if } sQs' \textit{ and } c\,\mathrm{id}_X\,f(s) = f(s') \\ \textit{undefined} & \textit{if no such } s' \textit{ exists} \end{cases}$$

for $(f,Q) : X \xrightarrow{\mathbf{X}} Y$ and $s \in Y$. The injectivity requirement on $\mathbf{X}$-morphisms guarantees that s' in the definition is unique, if it exists. One can easily verify that this c' satisfies the required properties.

(b) Note that $[r] \subseteq \langle r \rangle$, so $\downarrow[r] = \downarrow[r]\,;\downarrow\langle r\rangle$.

$$\begin{array}{rcll} c & = & C\Big(\downarrow[r]\Big)\,(c) & (c \,\#\, e) \\ & = & C\Big(\downarrow[r]\,;\downarrow\langle r\rangle\Big)\,(c) & (\downarrow[r] = \downarrow[r]\,;\downarrow\langle r\rangle) \\ & = & C\Big(\downarrow\langle r\rangle\Big)\,\Big(C(\downarrow[r]\,c\Big) & (\text{functoriality of } C) \\ & = & C\Big(\downarrow\langle r\rangle\Big)\,(c) & (c \,\#\, e) \end{array}$$

In fact, it is possible to prove that $c \,\#\, r$ iff $c \dashv r$ *and* $c \dashv \neg r$.

(c)

$$\begin{array}{rcll} p\,\mathrm{id}_X\,c & = & \Big([\![\theta \to \theta']\!]\,\Big(\downarrow\langle r\rangle\Big)\,p\Big)\,\mathrm{id}_X\,c & (p \dashv r) \\ & = & p\,\Big(\downarrow\langle r\rangle\Big)\,c & (\text{morphism part of } [\![\theta \to \theta']\!]) \\ & = & p\,\Big(\downarrow\langle r\rangle\Big)\,\Big([\![\theta]\!]\,\Big(\downarrow\langle r\rangle\Big)\,c\Big) & (c \dashv r) \\ & = & [\![\theta']\!]\,\Big(\downarrow\langle r\rangle\Big)\,\Big(p\,\mathrm{id}_X\,c\Big) & (\text{uniformity for } [\![\theta \to \theta']\!]) \end{array}$$

□

We can now validate non-interference abstraction. Let $c \in [\![\mathbf{comm}]\!]X$, $p \in [\![\mathbf{comm} \to \mathbf{comm}]\!]X$, and $q, r, a, b, d \in [\![\mathbf{assert}]\!]X$ be the meanings of P, C, Q, R, A, B, and D in an environment $u \in [\![\pi]\!]X$. Suppose

$$p\,\mathrm{id}_X\,c\,\mathrm{id}_X\,s_1 \;=\; s_2.$$

If the conditions in the antecedent of non-interference abstraction hold, then by Proposition 9(a) we get that there exists $c' \in [\![\mathbf{comm}]\!]X$ such that

(1) $p\,\mathrm{id}_X\,c'\,\mathrm{id}_X\,s_1 \;=\; s_2$, (by monotonicity of $p\,\mathrm{id}_X$),

(2) $c' \dashv r$, and

(3) $\{q\}\,c'\,\{d\}$.

Assuming that $r\,\mathrm{id}_X\,s_1 = a\,\mathrm{id}_X\,s_1 = \mathit{true}$, we need to show that $r\,\mathrm{id}_X\,s_2 = b\,\mathrm{id}_X\,s_2 = \mathit{true}$. (2) and Proposition 9(b) and (c) yield $p\,\mathrm{id}_X\,c' \dashv r$. By the definition of $\dashv$ and the uniformity condition on $[\![\mathbf{comm}]\!]X$ we get

(4) $p\,\mathrm{id}_X\,c'\,(\lceil Z)\,s_1 \;=\; s_2$

where $Z = \{s \in X \mid r\,\mathrm{id}_X\,s \;=\; \mathit{true}\}$. The specification $\{R\}$ is true at the world Z, i.e. $\forall s \in Z,\ r\,(\lceil Z)\,s = \mathit{true}$. Therefore, if we bind ι to c' then $\{R\}\ \&\ \{Q\}\iota\{D\}$ is true at Z; more precisely

$$[\![\{R\}\ \&\ \{Q\}\iota\{D\}]\!]Z\ ([\![\pi]\!]\,(\lceil Z)\,u \mid \iota \mapsto c')\,\mathrm{id}_Z.$$

Thus, by the assumption

$$\forall \iota : \mathbf{comm}\,.\,\{R\}\ \&\ \{Q\}\iota\{D\} \Rightarrow\ \{A\}P(i)\{B\},$$

(4) and (3) imply that $b\,\mathrm{id}_X\,s_2 = \mathit{true}$. $r\,\mathrm{id}_X\,s_2 = \mathit{true}$ follows from (4) and the definition of Z, and we are done.

7 Semantic Equivalences

We briefly consider some semantic equivalences. These will serve to illustrate the treatment of local variables given by our model, but will also point to some apparent weaknesses of the logic. The first two equivalences are from (Meyer and Sieber, 1988), and are valid also in the models presented there and in (Oles, 1982).

Example 1

Consider the block

$$\mathbf{new}[\tau]\ x\,.\,C$$

This block *is* equivalent to C in the model, if C is a command identifier (contrary to the remark in Tennent, 1990). To show the equivalence, consider a map $f = (g, T_X) : X \times V_\tau \xrightarrow{\mathbf{W}} X$, where $g(x) = (x, v_\tau)$ for each $x \in X$ and v_τ is the initial value for τ-variables. Let $c \in [\![\mathbf{comm}]\!]X$ and $s \in X$. If $c(\mathrm{id}_X)\ s$ is defined and equal to s', then $c(\times V_\tau ; f)\, s = s'$ since $\times V_\tau ; f = \mathrm{id}_X$. The uniformity condition on $\rightsquigarrow$ (Tennent, 1990) then implies that $c(\times V_\tau)\ (s, v_\tau) = (s', v_\tau)$. That is, $c(\times V_\tau)\ (s, v_\tau)$ has the same effect on the X-valued component of the stack as does $c(\mathrm{id}_X)$. In the case that $c(\mathrm{id}_X)\ s$ is undefined, the uniformity condition guarantees that $c(\times V_\tau)\ (s, v_\tau)$ is also undefined. The desired equivalence follows. □

Example 2

The following blocks are equivalent:

$$\mathbf{new}[\mathbf{int}]\ x\,.\,\Big(\mathbf{new}_{\mathbf{int}}\ y\,.\,x := 0\ ;\ y := 0\ ;\ Q(x)(y)\Big), \text{ and}$$

$$\mathbf{new}[\mathbf{int}]\ x\,.\,\Big(\mathbf{new}_{\mathbf{int}}\ y\,.\,x := 0\ ;\ y := 0\ ;\ Q(y)(x)\Big)$$

where $Q : \mathbf{var}[\mathbf{int}] \to (\mathbf{var}[\mathbf{int}] \to \mathbf{comm})$ is an identifier. This equivalence can be shown by a straightforward calculation using an endomorphism that

exchanges the two V components in a world of the form $X \times V \times V$, thus exchanging the declared variables. □

Our final equivalence is subtler than the previous two, and the model here is the only one we know of for which it is valid.

Example 3

The block

$$\mathbf{new}[\mathbf{int}]\, x \,.\, \Big(x := 1 \;;\; P(x)\Big)$$

is equivalent to $P(1)$, where $P : \mathbf{exp}[\mathbf{int}] \to \mathbf{comm}$ is an identifier. The idea is that P doesn't have write access to x, so whenever x is evaluated in the call $P(x)$ it will produce 1.

We sketch the proof of the equivalence. Suppose that $e \in [\![\mathbf{exp}[\mathbf{int}]]\!] X \times V$ is the denotation of the r-value of x and $p \in [\![\mathbf{exp}[\mathbf{int}] \to \mathbf{comm}]\!] X \times V$ is the denotation of P in the body of the block. Since P is non-local, we know that $p \# e$. Using the passivity of $\mathbf{exp}[\mathbf{int}]$ and procedure-call laws derived from Non-Interference Decomposition (section 1), we obtain $(p\,\mathrm{id}_{X\times V}\, e) \# e$. By the definitions of $[\![\mathbf{comm}]\!]$ and $\#$, any call $p\,\mathrm{id}_{X\times V}\; e\,\mathrm{id}_{X\times V}\, s$ is defined and equal to s' iff $p\,\mathrm{id}_{X\times V}\; e\; \lceil Y\, s = s'$, where $Y = \{s' \in X \times V \mid s[e]s'\}$. In particular, if s is a state where the value of e is 1, then Y is the set of states where e is 1. In the restricted world Y, e and 1 are semantically equal - both are (families of) functions that return 1 in any state - so e can be replaced by 1 in the call to p. The equivalence follows. □

We have not been able to prove any of the equivalences in Examples 1-3 in specification logic. The logic does quite well on specific Hoare-triples of the Meyer/Sieber variety, such as the ones for divergence in section 1 and 6, but not on general equivalences between *open* terms. These examples suggest that stronger axioms for various control structures (most importantly, **new**) need to be stated in which $\Leftrightarrow$, rather than $\Rightarrow$, is the principle connective.

8 Concluding Remarks

We have shown here the soundness of axioms for reasoning about *procedural* non-interference in specification logic. Because their proofs do not depend on any details of the interpretations of types, the laws for non-interference decomposition and variable declarations remain valid in continuation semantics (Tennent and Tobin, 1991). The procedural instances of these axioms are even more indispensable in the presence of jumps because, semantically, *commands* themselves are procedures that map continuations to continuations. It should be noted, however, that our treatment of non-interference abstraction does not carry over into the continuation framework because its proof relies on details of $[\![\mathbf{comm}]\!]$.