Hoare's Logic in the LF

Ian A. Mason

May 19, 1993

1 Introduction

In this paper we describe several attempts at defining a simple version of Hoare's logic in the LF. This is used as a framework for discussing certain issues that are raised. We give, in detail, three different attempts at formulating Hoare's logic in the LF, only two of these are in any sense successful. Both successful versions suggest certain desiderata concerning the nature of tacticals in the LF and how they can be used in faithfully presenting logics. We also mention some alternative solutions in passing. The structure of this paper is as follows. In the next section we give a brief description of the particular version of Hoare's logic that we shall deal with. The next section will then deal with the simple minded attempt to represent it in the LF. The most interesting feature of this attempt is that it is not successful. The fourth section rectifies this, and the fifth section refines this solution.

2 The Subject Matter

We begin by describing the logic, in the traditional fashion, that we will be studying. Our description here is based closely on that in [Apt, 1981]. Let τ denote a first order language with equality, the meta-variables x, y, z denote or range over the variables of τ, the meta-variables s, t denote or range over the terms or *expressions* of τ, the meta-variable e is used to denote a quantifier-free formula or *boolean expression* of τ, and, finally, p, q, r denote or range over the formulas or *assertions* of τ.

Let W denote the least class of programs such that

- **1.** for every variable x and expression t, $x := t \in W$; and
- **2.** if $S_1, S_2 \in W$ then $S_1; S_2 \in W$, and for every boolean expression e of τ, we have that $\texttt{if}(e, S_1, S_2) \in W$ and $\texttt{while}(e, S_1) \in W$.

The elements of W are called *while programs,* and we take their interpretation to be understood. The basic formulas of Hoare's logic are objects of the form $\{p\}S\{q\}$ where p, q are assertions and S is a while program. The intuitive meaning of an *asserted program,*

$$\{p\}S\{q\},$$

is as follows: whenever p holds before execution of S and S terminates, then q holds after execution of S. Hoare's logic is a system of formal reasoning about these asserted programs. Its axioms and proof rules are the following.

Axiom 1: Assignment Axiom

$$\{p[t/x]\}x := t\{p\}.$$

Rule 2: Composition Rule

$$\frac{\{p\}S_1\{r\}, \quad \{r\}S_2\{q\}}{\{p\}S_1; S_2\{q\}}$$

Rule 3: If Rule

$$\frac{\{p \wedge e\}S_1\{q\}, \quad \{p \wedge \neg e\}S_2\{q\}}{\{p\}\texttt{if}(e, S_1, S_2)\{q\}}$$

Rule 4: While Rule

$$\frac{\{p \wedge e\} S \{p\}}{\{p\}\texttt{while}(e, S)\{p \wedge \neg e\}}$$

The final rule involves some notion of a consequence relation, [Avron, 1986], for the assertion language. The usual approach is to assume a background first order theory, T, for example Peano arithmetic, and a proof system for the assertion language, for example the usual natural deduction system. The rule in question is the following.

Rule 5: Consequence Rule

$$\frac{p \Rightarrow p_1, \quad \{p_1\} S \{q_1\}, \quad q_1 \Rightarrow q}{\{p\} S \{q\}}$$

Here $p \Rightarrow p_1$ and $q_1 \Rightarrow q$ are assumed to follow from the background first order theory using the proof system for the assertion language. As usual, $p[t/x]$ stands for the result of substituting t for the free occurences of x in p.

3 Version One

3.1 The Basic Types

We begin the first version by fixing a finite first order language, τ. Relative to τ we have the following basic types, l_τ is the type of identifiers or memory locations of the while language, i_τ is the type of first order terms in the language τ, b_τ is the type of boolean expression or equivalently quantifier–free first order formulas in the language τ, o_τ is the type of first order formulas in the language τ, w_τ is the type of while programs over τ and finally h_τ is the type of Hoare triples over τ. We omit explicitly mentioning the dependence on τ whenever possible, thus we write l rather than l_τ.

$$\text{Syntactic Categories} = \begin{cases} l : \texttt{TYPE}, & \text{locations.} \\ i : \texttt{TYPE}, & \text{terms.} \\ o : \texttt{TYPE}, & \text{formulas.} \\ b : \texttt{TYPE}, & \text{boolean expressions.} \\ w : \texttt{TYPE}, & \text{while programs.} \\ h : \texttt{TYPE}, & \text{Hoare triples.} \end{cases}$$

It is important to notice that we are, by necessity, distinguishing between the variables of the first order logic, which are simply the variables of the LF, and the variables of the while language. This is because the latter are not substitutive. In particular we could not model $:=$ as an object of type $i \to i \to w$ since this would allow the formation of expressions like $0 := 1$, which are obviously syntactically and semantically illformed. The variables of i and o are the variables of the LF. Also note that since the LF does not have subtypes we must distinguish between the boolean expressions and the

first order formulas. The force of this distinction is somewhat diminished by an identification α,

- $\alpha : b \to o$.

It is supposed to be the obvious identification of boolean expressions and quantifier–free formulas. Later we shall state some axioms which attempt to force this identification to be the obvious one. The next operation is the bang which takes an identifier to its contents.

- $! : l \to i$

In other words if $x : l$ then $x!$ (which we will write instead of the awkward $!(x)$) is the contents of x. It can thus be thought of as an evaluation mechanism. The logical constants on terms are as follows:

- $c_i : i$,
  - for each constant symbol $c \in \tau$.
- $f_i : \underbrace{i \to i \to \ldots \to i}_{n+1 \ i's}$,
  - for each n-ary function symbol $f \in \tau$.

Now for the logical constants on formulas:

- $R_o : \underbrace{i \to i \to \ldots \to i \to o}_{n \ i's}$
  - for each n-ary relation symbol $R \in \tau$.
- $=_o : i \to (i \to o)$
- $\Rightarrow_o : o \to (o \to o)$
- $\neg_o : o \to o$
- $\forall : (i \to o) \to o$

However to make things more readable we include all the usual logical constants, either as new constants (as we do here) or as defined objects, which we do when it comes to proofs.

- $\wedge_o : o \to (o \to o)$
- $\vee_o : o \to (o \to o)$
- $\exists : (i \to o) \to o$

As our notation suggests we will use infix notation whenever appropriate. Thus rather than write $\vee(x_1)(x_2)$ we will use the more standard notation of $x_1 \vee x_2$. These operations, of course, have analogous versions in the case of boolean expressions. These tedious duplications are forced by our inability, on the face of it, to have subtypes in the LF. In the final version of Hoare's logic we will look at another way, using *syntactic judgements*, of representing subtypes.

- $R_b : \underbrace{i \to i \to \ldots \to i \to b}_{n \ i's}$
  - for each n-ary relation symbol of τ.
- $=_b$: $i \to (i \to b)$
- $\Rightarrow_b$: $b \to (b \to b)$
- $\neg_b : b \to b$

Again to make things more readable we include all the usual logical constants, either as new constants or as defined objects, whichever is most convenient at the time.

- $\wedge_b : b \to (b \to b)$
- $\vee_b : b \to (b \to b)$

The o and b versions of the same operation can be identified at the level of definitions in the current implementation of the LF. By this we mean that we can use the same symbol for both variations, commonly called *overloading*. Although in the current implementation it is the user that does the overloading not the editor itself. In what follows we will also do this, thus we will write $x \wedge y$ in either of the cases when the operands are of type o or b. Context will always prevent confusion.

3.2 The While Language

We now add constants that correspond to the constructs of our while language, they are simply the curried versions of the constructs described in the informal introduction.

- $\texttt{ass} : l \to (i \to w)$,
  - denoting $\texttt{ass}(x)(t)$ by $x := t$.
- $\texttt{seq} : w \to (w \to w)$,
  - denoting $\texttt{seq}(w_1)(w_2)$ by $w_1; w_2$.
- $\texttt{if} : b \to (w \to (w \to w))$,
  - denoting $\texttt{if}(b_1)(w_1)(w_2)$ by $\texttt{if}(b_1, w_1, w_2)$.
- $\texttt{while} : b \to (w \to w)$,
  - denoting $\texttt{while}(b_1)(w_1)$ by $\texttt{while}(b_1, w_1)$.

3.3 Hoare's Triples

The syntax of the Hoare triples is easily taken care of by a single constant.

- $\texttt{triple} : o \to (w \to (o \to h))$,
  - denoting $\texttt{triple}(o_1)(w_1)(o_2)$ by $\{o_1\}w_1\{o_2\}$.

3.4 The Judgements

The following are the judgements of our first version, they correspond to judgement TRUE or more appropriately provable sentences in the respective classes of expressions.

- $\vdash_h : h \to \texttt{TYPE}$
- $\vdash_b : b \to \texttt{TYPE}$
- $\vdash_o : o \to \texttt{TYPE}$

Again we can use the definition mechanism to hide the distinction between these three judgements. Thus we shall omit the subscript leaving context to do its job.

3.5 The Rules Concerning α

The rules concerning the identification α are straight forward and so we shall not make any undue fuss.

- $\alpha_1 : \Pi_{x:b} \quad (\vdash x) \to (\vdash \alpha(x))$
- $\alpha_2 : \Pi_{x:b} \quad (\vdash \alpha(x)) \to (\vdash x)$
- $\alpha_{3R} : \Pi_{x_1:i} \ldots \Pi_{x_n:i} \quad (\vdash \alpha(R_b(x_1, \ldots, x_n))) \to (\vdash R_o(x_1, \ldots, x_n))$
  - for $R \in \tau \cup \{=\}$.
- $\alpha_{4R} : \Pi_{x_1:i} \ldots \Pi_{x_n:i} \quad (\vdash R_o(x_1, \ldots, x_n)) \to (\vdash \alpha(R_b(x_1, \ldots, x_n)))$
  - for $R \in \tau \cup \{=\}$.

For ease of use it is probably also useful to include, as either new constants or derived terms, the following:

- $\alpha_5 : \Pi_{x_1:i}\Pi_{x_2:i} \quad \vdash (\alpha(x_1 = x_2) \Leftrightarrow x_1 = x_2)$
- $\alpha_6 : \Pi_{x:b} \quad \vdash (\alpha(\neg x) \Leftrightarrow \neg\alpha(x))$
- $\alpha_{7\chi} : \Pi_{b_1:b}\Pi_{b_2:b} \quad \vdash (\alpha(b_1 \ \chi \ b_2) \Leftrightarrow (\alpha(b_1) \ \chi \ \alpha(b_2)))$
  - for $\chi \in \{\wedge, \vee, \Rightarrow\}$.

Clearly, given the usual interpretation of the logical constants, α_1 and α_2 imply $\alpha_5, \ldots, \alpha_7$ but the converse, even with the addition of α_3 and α_4 is false due to our inability to force b to be inductively generated by the relevant constants. In this sense it could hardly be said that we are formalizing the syntax.

3.6 The Hoare Rules and Axioms

We omit the rules of first order logic, the interested reader is refered to [Harper, Honsell and Plotkin, 1986]. We continue with the rules of Hoare, they are the obvious analogs of the informal rules given the preceeding setting.

- $\texttt{Ass} : \Pi_{x:l}\Pi_{t:i}\Pi_{p:i\to o}$
$$\vdash \{p(t)\}x := t\{p(x!)\}$$

- $\texttt{Comp} : \Pi_{a:o}\Pi_{b:o}\Pi_{c:o}\Pi_{w_1:w}\Pi_{w_2:w}$
$$\vdash \{a\}w_1\{b\} \to \quad \vdash \{b\}w_2\{c\} \to \quad \vdash \{a\}w_1; w_2\{c\}$$

- $\texttt{If} : \Pi_{e:b}\Pi_{e_1:o}\Pi_{e_2:o}\Pi_{w_1:w}\Pi_{w_2:w}$
$$\vdash \{e_1 \wedge \alpha(e)\}w_1\{e_2\} \to \quad \vdash \{e_1 \wedge \neg\alpha(e)\}w_2\{e_2\} \to \quad \vdash \{e_1\}\texttt{if}(e, w_1, w_2)\{e_2\}$$

- $\texttt{While} : \Pi_{e:b}\Pi_{e_1:o}\Pi_{w_1:w}$
$$\vdash \{e_1 \wedge \alpha(e)\}w_1\{e_1\} \to \quad \vdash \{e_1\}\texttt{while}(e, w_1)\{e_1 \wedge \neg\alpha(e)\}$$

- $\texttt{Con} : \Pi_{a:o}\Pi_{a_1:o}\Pi_{b:o}\Pi_{b_1:o}\Pi_{w_1:w}$
$$\vdash a_1 \Rightarrow a \to \quad \vdash b_1 \Rightarrow b \to \quad \vdash \{a_1\}w_1\{b_1\} \to \quad \vdash \{a\}w_1\{b\}$$

3.7 The Adequacy Theorems

We now state the adequacy theorems for syntax and semantics that should hold in any successful internalization of Hoare's logic in the LF. Beginning with the adequacy for syntax, which is stated for each separate syntactic category. We begin with some notation. Let Γ_n^m be the following context, for m, n integers:

$$\Gamma_n^m = \{y_0 : i, \ldots, y_m : i, x_0 : l, \ldots, x_n : l\}.$$

Theorem 1 (Adequacy for Syntax) *In the above LF signature and in the context Γ_n^m we have the following facts concerning syntax:*

- *l : All well formed long $\beta\eta$–normal forms of type l are LF variables of type l, and hence are among the $x_0, \ldots, x_n$.*

- *There are compositional bijections τ_K, for each syntactic category K, which map long $\beta\eta$–normal forms of type K to:*

  - *well formed terms of the assertion language built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$, in the case when $K = i$.*
  - *quantifier–free formulas of the assertion language built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$, in the case when $K = b$.*
  - *formulas of the assertion language built up from the set of identifiers $\bar{x}$ (which if they occur must occur free) and the logical variables $\bar{y}$, in the case when $K = o$.*
  - *while programs of τ built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$ (which do not occur in the left hand side of any assignment statement), in the case when $K = w$.*
  - *asserted programs (i.e. Hoare triples) built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$ (here again no variable from $\bar{y}$ can occur in the left hand side of any assignment statement), in the case of $K = h$.*

Remarks:

1. Actually there is a minor problem in the statement of the result for w and hence for h. This arises because in the informal description of the while language there is no distinction between $S_0; (S_1; S_2)$ and $(S_0; S_1); S_2$ whereas these are quite distinct from the point of view of the LF internalization. Perhaps the simplest solution to this minor technicality is to be somewhat more precise in the informal description, incorporating this distinction there. Thus the second clause in the informal description should be restated as

   - **2.** if $S_1, S_2 \in W$ then $(S_1; S_2) \in W$, and for every boolean expression e of τ, we have that $\texttt{if}(e, S_1, S_2) \in W$ and $\texttt{while}(e, S_1) \in W$.

2. Long $\beta\eta$–normal forms (see [Jensen and Pietrzykowski, 1976] for a definition) have been chosen, rather than $\beta\eta$–normal forms, so as to overcome the difficulty of deciding which bound variables may occur in the set of first order formulas built over $\bar{x}$ and $\bar{y}$. Since both λ-expressions and formulas are only considered up to α-equivalence and also since $\forall(M)$ occurs in long $\beta\eta$ normal form iff $M \equiv \lambda x.N$ where N is of type o we can define $\tau_o(\forall(\lambda x.N))$ to be $\forall x \tau_o(N)$.

Outline of Proof: We begin by defining, recursively, the collections of long $\beta\eta$–normal forms, in the context Γ_n^m, of the syntactic categories in question. In what follows N_x^m denotes these forms for the category x.

- $N_l^m = x_0 \mid \ldots \mid x_n$.
- $N_i^m = y_0 \mid \ldots \mid y_m \mid c_i \mid N_l^m! \mid f_i \underbrace{N_i^m \ldots N_i^m}_{s \text{ times}}$
  - for all constants c and s-ary functions symbols in τ.
- $N_b^m = R_b \underbrace{N_i^m \ldots N_i^m}_{s \text{ times}} \mid N_i^m =_b N_i^m \mid \neg_b N_b^m \mid N_b^m \Rightarrow_b N_b^n$
  - for any s-ary relation in τ.
- $N_o^m = R_o \underbrace{N_i^m \ldots N_i^m}_{s \text{ times}} \mid N_i^m =_o N_i^m \mid \neg_o N_o^m \mid N_o^m \Rightarrow_o N_o^n \mid \forall(\lambda y_{m+1} : i.N_o^{m+1})$
  - for any s-ary relation in τ.
- $N_w^m = N_l^m := N_i^m \mid (N_w^m; N_w^m) \mid \texttt{if}(N_b^m, N_w^m, N_w^m) \mid \texttt{while}(N_b^m, N_w^m)$.
- $N_h^m = \{N_o^m\} N_w^m \{N_o^m\}$.

The compositional bijections, for each category, are then defined inductively on the above categories in the obvious fashion. Q.E.D.

Now we proceed to the adequacy for semantics. In the case for b and o there is very little difference between the results here and those stated in [Harper, Honsell and Plotkin, 1986]. The only remark needed to be made is that the identifiers are taken to behave like constants. Thus we shall concentrate on the novel case of h.

Theorem 2 (Adequacy for Semantics) *There is a compositional bijection between proofs of a Hoare triple* $\{p\}S\{q\}$ *from assumptions* $r_0, \ldots, r_s$ *(in the assertion language) and assumptions* $\{p_0\}S_0\{q_0\}, \ldots, \{p_t\}S_t\{q_t\}$ *(concerning asserted programs) and well formed* $\beta\eta$*–normal forms of type*

$$\vdash_h \{p\}S\{q\}$$

in the above signature and in the context Γ *where*

$$\Gamma = \Gamma_n^m \cup \{w_j : (\vdash_o r_j), v_i : (\vdash_h \{p_i\}S_i\{q_i\})\}_{0 \leq j \leq s, 0 \leq i \leq t},$$

and Γ_n^m *is adequate for the syntax of objects involved.*

3.8 The Paradoxes

Unfortunately the preceeding theorem is false under this internalization, the rule `Ass` is in fact erroneous. We give two examples of this failure, one at the level of locations and the other at the level of LF terms. They are essentially the same example, viewed at different levels of abstraction, and provide us with two different (though essentially equivalent) motivations for the solution we shall present. Take τ to be the language of arithmetic.

Example 1 *In the context*

$$\Gamma = \{x : l,\ y : l\}$$

we have the following instance of the assignment axiom `Ass`,

$$\{\neg(x! = 1)\}y := 1\{\neg(x! = y!)\}.$$

Now x *and* y *are simply LF variables that are declared to be of type* l*. No other assumption about them has been made. Consequently it is reasonable to assume that they both denote the same location or identifier* l_0*. In this case the axiom states that*

$$\{\neg(l_0! = 1)\}l_0 := 1\{\neg(l_0! = l_0!)\},$$

and since it is reasonable to assume that $\neg(l_0! = 1)$ *is possible we arrive at a somewhat unfortunate state of affairs.*

Example 2 *Suppose* $\Gamma = \{y : l\}$ *is the current context and take the following instantiation of the assignment axiom,*

$$\mathtt{ASS}(y)(1)(\lambda u.\neg(y! = u)).$$

This term inhabits the following type

$$\vdash \{\neg(y! = 1)\}y := 1\{\neg(y! = y!)\},$$

which one would have hoped was uninhabitable.

The problem in the first example, intuitively, is that

$$\{P(t)\}x := t\{P(x!)\}$$

can be false because the assignment $x := t$ can alter the meaning of the predicate $\lambda z P(z)$. Thus the simplest solution to this problem is to axiomatize the relevant notion of non-interference. This solution in some sense however hides the source of the problem, since a casual glance reveals no important differences between our interpretation here and the informal description we began with.

One point that can be made here is that the LF cannot handle meta-variables in the way they are commonly used in describing logics. The reason that the inconsistency does not arise in the informal version is that, in essence, the operator $p[t/x]$ is call by value; it replaces all occurences of the *value* of the meta-variable x (which is a variable of τ) by the *value* of the meta-variable t (which is a term of τ). In contrast to this, substitution in the LF,

$$(\lambda z P(z))(t)$$

takes place at the level of the meta-variables (i.e. the variables of the LF) and not at the level of their values. In this sense LF substitution could be called call by name.

Perhaps a clearer explanation can be given by examining when the two notions of substitution, 1. $p[t/x]$ and 2. $(\lambda z.P(z))(t)$, coincide and when they differ. To make the following discussion more readable we omit the bang operator, ! , since neither its presence nor its absence has any bearing on the phenomenon we are considering. The first and most obvious difference between these two operators is that they apply to different sorts of objects.

The first form has as its arguments — a formula, a variable and a term. The second form has as arguments (in the notation of the LF) a function from terms to formulas and a term. To make this explicit we shall write $Sub1(p, x, t) = p[t/x]$, and $Sub2(P, t) = P(t)$.

One way of unifying the picture is to decompose the first form of substitution, $Sub1(p, x, t)$, into two separate operations. Given a formula p we first form the function, $\lambda x.p$, from terms to formulas, we then apply this function to the given term t. Thus we conclude that

$$Sub1(p, x, t) = p[t/x] = (\lambda x.p)(t) = Sub2(\lambda x.p, t),$$

hence the first form of substitution can be considered as a special case of the second.

Similarly we can express the second form of substitution, $Sub2(P, t)$, in terms of the first, but in this case we need a side condition. $Sub2(P, t)$ can be also be decomposed into two operations. Given P we first obtain a formula by applying P to a *new* variable x, we then replace all occurences of this new variable by the supplied term t. By a *new* variable we mean a variable that does not occur *free* in P. This is the same, in the LF, as saying that the variable does not occur free in $\forall P$. In this case we can conclude that

$$Sub2(P, t) = P(t) = (P(x))[t \setminus x] = Sub1(P(x), x, t) \quad \text{when} \quad x \notin FV(\forall P).$$

It is important to notice that in the Hoare triple

$$\{P(t)\}x := t\{P(x!)\}$$

free occurences of x in $P(x!)$ are bound by the assignment operator $x := t$, this is not true of those occurences in $P(t)$ nor of the occurences in t in the assignment. Thus in example 2. we have a clear case of a variable being captured, in the right hand side, during the process of substitution. Thus in one sense α-conversion should take place. This however would not be in the spirit of Hoare's logic, since we want to reason about the identifier x not some α-conversion of it. Consequently we have our first example of a binding operator that doesn't α-convert and the associated problems that arise. The assignment axiom can now be correctly stated, informally, as

- $\texttt{Ass} : \Pi_{x:l}\Pi_{t:i}\Pi_{p:i\to o}$

$$x \notin FV(\forall P) \to \quad \vdash \{p(t)\}x := t\{p(x!)\}.$$

We now put this knowledge into practice.

4 Version Two

The easiest solution to this problem of formalizing the corrected version of the assignment axiom is to incorporate syntactic notions explicitly into the theory. We do this by adding three new judgements concerning non-interference along the lines of [Reynolds, 1978].

- $\neq: l \to (l \to \texttt{TYPE})$
- $\sharp_i : l \to (i \to \texttt{TYPE})$
- $\sharp_o : l \to (o \to \texttt{TYPE})$

As usual we will identify a judgement with the type of its evidence, which in this case consists of its proofs. The intuitive meaning of the judgements can be explained, again using infix notation, as follows:

- $x \neq y$ is interpreted as meaning that x and y denote distinct identifiers or locations. As we have already mentioned, because we have no constructors or constants of type l, terms of type l must $\beta\eta$ reduce to LF variables.
- $x \sharp_i t$ is interpreted as meaning that no assignment to the location denoted by x effects the value of the term denoted by t. This of course is equivalent to saying that the location or identifier denoted by x does not occur in the term denoted by t.
- $x \sharp_o e$ is interpreted as meaning that no assignment to the location denoted by x effects the value or meaning of the formula denoted by e. Again this is equivalent to saying that the location or identifier denoted by x does not occur freely in the formula denoted by e (Note that it cannot occur bound).

Again we will omit the subscripts in favor of context.

4.1 Non-Interference Axioms and Rules

It is a simple task to axiomatize the above notions, and we present one such here.

- $\sharp_{0c} : \Pi_{x:l} \quad x\sharp c_i$,
  - for each constant c in τ.
- $\sharp_1 : \Pi_{x:l}\Pi_{y:l} \quad x \neq y \rightarrow x\sharp y!$
- $\sharp_{2f} : \Pi_{t_1:i}\Pi_{t_2:i}\ldots\Pi_{t_n:i}\Pi_{x:l} \quad x\sharp t_1 \rightarrow x\sharp t_2 \rightarrow \ldots \rightarrow x\sharp t_n \rightarrow x\sharp f(t_1, t_2, \ldots, t_n)$,
  - for each n-ary operation in τ.
- $\sharp_{3R} : \Pi_{t_1:i}\Pi_{t_2:i}\ldots\Pi_{t_n:i}\Pi_{x:l} \quad x\sharp t_1 \rightarrow x\sharp t_2 \rightarrow \ldots \rightarrow x\sharp t_n \rightarrow x\sharp R(t_1, t_2, \ldots, t_n)$,
  - for each n-ary relation in $\tau \cup \{=\}$.
- $\sharp_4 : \Pi_{e:o}\Pi_{x:l} \quad x\sharp e \rightarrow x\sharp\neg e$
- $\sharp_{5\chi} : \Pi_{e_1:o}\Pi_{e_2:o}\Pi_{x:l} \quad x\sharp e_1 \rightarrow x\sharp e_2 \rightarrow x\sharp e_1 \ \chi \ e_2$
  - for $\chi \in \{\wedge, \vee, \Rightarrow\}$.
- $\sharp_6 : \Pi_{f:i\rightarrow o}\Pi_{x:l}\Pi_{y:l} \quad (x \neq y \rightarrow x\sharp f(y!)) \rightarrow x\sharp\forall f$

That we have captured the correct notion is expressed in the following proposition. Its proof is an easy induction.

Proposition 1 *Suppose that* $\Gamma = \{x_0 : l, x_i : l, z_i : x_0 \neq x_i\}_{i\in I}$ *and* $\Gamma \vdash (\phi : o)$. *Then the following are equivalent*

1. $\Gamma \vdash x_0\sharp\phi$
2. $FV(\phi) \subseteq \{x_i\}_{i\in I}$.

Before we continue we should make several remarks.

- The first thing to notice about our solution is that we take the notion of a free variable as primitive. Another possible solution is to axiomatize the notion of *substituting a term for all free occurences of a banged location*. This would involve introducing two new operations (rather than the two judgements $\sharp_i$ and $\sharp_o$) sub_i and sub_o.

- $sub_t : i \to l \to i \to i$, where $sub_i(t_0, x, t_1)$ is to represent the result of substituting the term t_0 for all free occurences of $x!$ in the term t_1.
- $sub_o : i \to l \to o \to o$, where $sub_o(t, x, p)$ is to represent the result of substituting the term t for all free occurences of $x!$ in the formula p.

To axiomatize these operations, in particular the base case, one must still retain the judgement $\neq$, and so in some sense the two solutions are dual. There is little reason, on the face of it, to choose one over the other. We should point out, however, that to correctly formalize more complex versions of Hoare's logic, for example one in which recursive procedure calls were allowed, it would be necessary to incorporate the notion on non-interference anyway. Thus in the long run the solution we have adopted seems most suited to Hoare's logic. On the other hand in dynamic logic the substitution approach is more natural, since there is no clear notion of free and bound variables in that logic.

- The above proof system for non-interference has a very special property that we shall discuss in detail later. Put crudely if a non-interference judgement can be proved, then there is, in a strong sense, a unique such proof. It is for this reason that we omit symmetry from the axioms concerning $\neq$.

- The solution we have offered above is a syntactic one, symmetry in the $\neq$ judgement, if desired, must be inforced at the level of the context. More importantly, however, is the fact that at the semantic level we have done nothing to force $\neq$ to be interpreted as inequality. Thus there are models in which the type $l_0 \neq l_0$ is non–empty for some l_0. A more elaborate system could no doubt be developed that didn't have this defect, it would however be a further deviation from the simple logic at hand. We are content with a syntactic solution.

- Another solution that we shall not discuss, but can be found in [Avron, Honsell and Mason 1987], is to use λ to model the $:=$ binding operator. As we have already remarked, the former has α–conversion while the latter does not. The solution to this problem is solved at the level of

the signature and only works for a version of Hoare's logic with a fixed finite number of locations.

4.2 The Axioms and Rules of Hoare Revisited

Using the non interference judgement we can formulate the correct version of the assignment axiom as follows:

- $\mathtt{ASS} : \Pi_{x:l}\Pi_{t:i}\Pi_{p:i\to o} \quad x\sharp\forall p \to \quad \vdash \{p(t)\}x := t\{p(x!)\}$

The remaining rules are the same as in the inadequate version and so we do not waste space repeating them here.

4.3 The Adequacy Theorems Repaired

In this version the adequacy theorem for syntax remains the same, however we modify the definition of Γ_n^m so that it includes the assumption that all distinct LF variables of type l denote distinct locations or identifiers. Explicitly define Γ_n^m as follows

$$\Gamma_n^m = \{y_0 : i, \ldots, y_m : i, x_0 : l, \ldots, x_n : l, z_{i,j} : x_i \neq x_j, z^*_{i,j} : x_j \neq x_i\}_{0\leq i\leq j\leq n}.$$

The adequacy theorem for semantics is then identical to the false one in the preceeding section. The only difference now is that it is true.

Theorem 3 (Adequacy for Semantics) *There is a compositional bijection between proofs of a Hoare triple $\{p\}S\{q\}$ from assumptions $r_0, \ldots, r_s$ (in the assertion language) and assumptions $\{p_0\}S_0\{q_0\}, \ldots, \{p_t\}S_t\{q_t\}$ (concerning asserted programs) and well formed $\beta\eta$–normal forms of type*

$$\vdash_h \{p\}S\{q\}$$

in the above signature and in the context Γ where

$$\Gamma = \Gamma_n^m \cup \{w_j : (\vdash_o r_j), v_i : (\vdash_h \{p_i\}S_i\{q_i\})\}_{0\leq j\leq s, 0\leq i\leq t},$$

and

$$\Gamma_n^m = \{y_0 : i, \ldots, y_m : i, x_0 : l, \ldots, x_n : l, z_{i,j} : x_i \neq x_j, z^*_{i,j} : x_j \neq x_i\}_{0\leq i\leq j\leq n}.$$

is adequate for the syntax of objects involved.

The fact that there is a compositional bijection depends heavily on the following fact concerning the non-interference proof system.

Proposition 2 (Uniqueness) *Suppose there are well formed LF terms x, e, P_0 and P_1 such that*

1. $\Gamma_n^m \vdash x : l$,
2. $\Gamma_n^m \vdash e : o$,
3. $\Gamma_n^m \vdash P_0 : x \sharp e$,
4. $\Gamma_n^m \vdash P_1 : x \sharp e$.

Then P_0 and P_1 have the same $\beta\eta$–normal forms.

This is important since if there were distinct proofs of a non-interference judgement then the extra parameter to the `Ass` axiom would force the mapping to identify different $\beta\eta$–normal forms.

5 Version Three

The third method of representing Hoare's logic is to utilize the method of using judgements to interpret or implement subtypes. In the case of Hoare's logic there is only one problematic subtype, that of the boolean expressions b. In this example we introduce a new judgement `QF` over o, whose interpretation is that of a formula being *quantifier-free*.

- $\texttt{QF} : o \to \texttt{TYPE}$

The two program constructions `if` and `while` now take a extra argument, namely a proof, or equivalently an element of the `QF` judgement, that their o argument is quantifier–free. We shall discuss the advantages and disadvantages of this approach after we have described it fully.

- $\texttt{IF} : \Pi_{e:o}\texttt{QF}(e) \to w \to w \to w$,
  - denoting $\texttt{IF}(e)(p)(w_1)(w_2)$ by $\texttt{if}(e, w_1, w_2)_p$.
- $\texttt{WHILE} : \Pi_{e:o}\texttt{QF}(e) \to w \to w$,
  - denoting $\texttt{WHILE}(e)_p(w_1)$ by $\texttt{while}(e, w_1)_p$.

5.1 The Rules of the Judgement QF

Axiomatizing the syntactic notion of being quantifier–free, as in the case of non-interference, presents no problems.

- $\mathtt{QF}_{1R} : \Pi_{t_1:i}\Pi_{t_2:i}\ldots\Pi_{t_n:i} \quad \mathtt{QF}(R(t_1, t_2, \ldots, t_n))$,
  - for each n-ary relation in $\tau \cup \{=\}$.
- $\mathtt{QF}_2 : \Pi_{e_1:o} \quad \mathtt{QF}(e_1) \to \mathtt{QF}(\neg e_1)$
- $\mathtt{QF}_{3\chi} : \Pi_{e_1:o}\Pi_{e_2:o} \quad \mathtt{QF}(e_1) \to \mathtt{QF}(e_2) \to \mathtt{QF}(e_1\ \chi\ e_2)$
  - for $\chi \in \{\wedge, \vee, \Rightarrow\}$.

Just as in the case of the proof system for non-interference, this proof system has the property (which we will discuss in more detail when we come to the adequacy theorem for syntax) that proofs are in a sense unique.

Proposition 3 (Uniqueness) *Suppose there are well formed LF terms e, P_0 and P_1 such that*

1. $\Gamma_n^m \vdash e : o$,
2. $\Gamma_n^m \vdash P_0 : \mathtt{QF}(e)$,
3. $\Gamma_n^m \vdash P_1 : \mathtt{QF}(e)$.

Then P_0 and P_1 have the same $\beta\eta$–normal forms.

5.2 The Rules and Axioms of Third Version of Hoare's Logic

In this final version the rules need only be modified so as to take into account the extra argument to the `if` and `while` constructs. We state them here, in their entirety, for reason of emphasis.

- $\mathtt{H}_1 : \Pi_{x:l}\Pi_{t:i}\Pi_{p:i\to o}$

$$x \sharp \forall p \to \quad \vdash \{p(t)\}x := t\{p(x!)\}$$

- $\mathtt{H}_2 : \Pi_{e_1:o}\Pi_{e_2:o}\Pi_{e_3:o}\Pi_{w_1:w}\Pi_{w_2:w}$

$$\vdash \{e_1\}w_1\{e_2\} \rightarrow \quad \vdash \{e_2\}w_2\{e_3\} \rightarrow \quad \vdash \{e_1\}w_1; w_2\{e_3\}$$

- $\mathtt{H}_3 : \Pi_{e:o}\Pi_{e_1:o}\Pi_{e_2:o}\Pi_{w_1:w}\Pi_{w_2:w}\Pi_{p:\mathtt{QF}(e)}$

$$\vdash \{e_1 \wedge e\}w_1\{e_2\} \rightarrow \quad \vdash \{e_1 \wedge \neg e\}w_2\{e_2\} \rightarrow \quad \vdash \{e_1\}\mathtt{if}(e, w_1, w_2)_p\{e_2\}$$

- $\mathtt{H}_4 : \Pi_{e:o}\Pi_{e_1:o}\Pi_{w_1:w}\Pi_{p:\mathtt{QF}(e)}$

$$\vdash \{e_1 \wedge e\}w_1\{e_1\} \rightarrow \quad \vdash \{e_1\}\mathtt{while}(e, w_1)_p\{\neg e_1\}$$

- $\mathtt{H}_5 : \Pi_{e_1:o}\Pi_{e_1':o}\Pi_{e_2:o}\Pi_{e_2':o}\Pi_{w_1:w}$

$$\vdash e_1 \Rightarrow e_1' \rightarrow \quad \vdash e_2' \Rightarrow e_2 \rightarrow \quad \vdash \{e_1'\}w_1\{e_2'\} \rightarrow \quad \vdash \{e_1\}w_1\{e_2\}$$

5.3 The Adequacy Theorems Revisited

In this third version the only syntactic categories which have changed (other than the elimination of b) are w and h. Consequently we need only state the adequacy theorem for syntax for these two categories, since the previous adequacy theorem for syntax is still applicable to the remaining categories. As before define

$$\Gamma_n^m = \{y_0 : i, \ldots, y_m : i, x_0 : l, \ldots, x_n : l, z_{i,j} : x_i \neq x_j, z_{i,j}^* : x_j \neq x_i\}_{0 \leq i \leq j \leq n}.$$

Theorem 4 (Adequacy for Syntax) *In the above LF signature and in the context Γ_n^m we have the following facts concerning syntax:*

- *w : There is a compositional bijection between well formed $\beta\eta$–normal forms of type w and the while programs of τ built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$ (which do not occur in the left hand side of any assignment statement).*

- *h : There is a compositional bijection between well formed $\beta\eta$–normal forms of type h and asserted programs (i.e. Hoare triples) built up from the set of identifiers $\bar{x}$ and the logical variables $\bar{y}$. Where again no variable from $\bar{y}$ can occur in the left hand side of any assignment statement.*

That there is a compositional bijection relies heavily on the uniqueness theorem for the `QF` judgement, since if there were distinct proofs then the extra parameter to the `if` and `while` operations would force the mapping to identify different $\beta\eta$–normal forms.

Theorem 5 (Adequacy for Semantics) *There is a compositional bijection between proofs of a Hoare triple* $\{p\}S\{q\}$ *from assumptions* $r_0, \ldots, r_s$ *(in the assertion language) and assumptions* $\{p_0\}S_0\{q_0\}, \ldots, \{p_t\}S_t\{q_t\}$ *(concerning asserted programs) and well formed* $\beta\eta$*–normal forms of type*

$$\vdash_h \{p\}S\{q\}$$

in the above signature and in the context Γ *where*

$$\Gamma = \Gamma_n^m \cup \{w_j : (\vdash_o r_j), v_i : (\vdash_h \{p_i\}S_i\{q_i\})\}_{0 \leq j \leq s, 0 \leq i \leq t},$$

and

$$\Gamma_n^m = \{y_0 : i, \ldots, y_m : i, x_0 : l, \ldots, x_n : l, z_{i,j} : x_i \neq x_j, z_{i,j}^* : x_j \neq x_i\}_{0 \leq i \leq j \leq n}$$

is adequate for the syntax of objects involved.

6 Conclusions

Thus we have presented two distinct versions of Hoare's logic, both somewhat more complex than the informal description. The question then arises as to which one is better or more faithful. In this conclusion we shall try to argue that the third version has the potential to be the more faithful. If we translate the uniqueness properties for the non-interference and quantifier-free judgements into properties concerning the search space, we notice that they assert that these spaces are linear. Consequently it would be desirable for the LF to provide tacticals which automatically construct the proofs. Note that such a tactical is no more complex in nature that the already implemented **Pi-Intro***, [Griffin, 1987]. It would however be more complex than those found in [Schmidt, 1983]. If this were the case then both judgements, `QF` and $\sharp$, could be hidden from the user. In the case of the third version this would result in a system almost identical to the informal description. The only difference being the presence of the type l and the associated function !. The

third version also exemplifies a useful technique for implementing syntactic subtypes as judgements rather than distinct types. If the proof system for these judgements have the uniqueness property, then they have a distinct methodological advantage over the *proliferation of types* method.

Implementing syntactic subtypes as judgements may be seen as a way of solving the openess problem By this we mean that we could introduce a judgement of being a syntactic term, $I : \iota \rightarrow \texttt{TYPE}$, along the lines of `QF`. Work needs to be done here to understand the consequences of such a move.

One last remark concerning the interpretation of syntactic subtypes as judgements is that in logic most (all?) subtypes are syntactic. Thus since in the LF we are restricting our attention to logic(s) we could paraphrase this method as *subtypes as judgements*.

7 Acknowledgements

This paper, such as it is, could not have been written without the numerous helpful comments and suggestions of Furio Honsell. Arnon Avron also made many useful comments in the course of the papers evolution. Gordon Plotkin also helped "show the fly the way out of the fly-bottle", as did Tim Griffin and Bob Harper.

8 References

- Apt, K.R. Ten Years of Hoare's Logic: A Survey— Part 1. A.C.M. Transactions on Programming Languages and Systems. Vol. 3, No. 4, October 1981, pp 431-483.
- Avron, A. Simple Consequence Relations. (to appear in the L.F.C.S. series).
- Avron, A., Honsell, F., and Mason, I.A. Using Judgements to Implement Logics on a Machine (to appear in the L.F.C.S. series).
- Griffin, T. An Environment for Formal Systems. (to appear).

- Harper, R., Honsell, F., and Plotkin, G. A Framework for Defining Logics. Proceedings of the Second Annual Conference on Logic in Computer Science. Cornell, 1987.
- Jensen, D.C., and Pietrzykowski, T. Mechanizing ω-order Type Theory through Unification. Theoretical Computer Science. Vol 3. 1976. pp123 171.
- Reynolds, J.C. Syntactic Control of Interference. Conference Record of the Fifth Annual Symposium on Principles of Programming Languages, Tucson, 1978.
- Schmidt, D. A Programming Notation for Tactical Reasoning. Department of Computer Science Internal Report No. CSR-141-83, Edinburgh University, 1983.

Contents