Strict Semantics for Hoare Logic with Procedures

Ian Stark

The University of Edinburgh

Wednesday 18 June 2008

Mobius

Mobility, Ubiquity and Security

Enabling proof-carrying code for Java on mobile devices

Overview

A brief summary of the material to come:

- We observe a mild mismatch between the conventional semantics of Hoare triples and type systems; particularly in the case of recursive procedures, and exacerbated by refined types.
- We propose an alternative *strict* semantics that strengthens Hoare assertions to enforce requirements on procedure call.
- We formalize this semantics in Nipkow's Isabelle model of a while language with procedures.
- We show soundness of some modified Hoare rules, and speculate on completeness and further extensions of the system.

This is joint work with Randy Pollack and Alberto Momigliano.

Hoare Logic is Hypothetical

Assertions in classic Hoare logic are *hypothetical*: the meaning of $\{P\}\, C\, \{Q\}$ is that if proposition P holds before executing code C, then proposition Q will hold after.

In particular, there need be no expectation or requirement that P actually will hold on any particular run: the same code C may satisfy many triples, and any code satisfies $\{\mathbf{false}\}\, C\, \{\mathbf{true}\}$.

This is also exposed in entailments between triples, as required for reasoning about recursive procedures: the meaning of

$$\{P\}\, C\, \{Q\} \quad \vdash \quad \{P'\}\, C'\, \{Q'\}$$

is that $\{P'\}\, C'\, \{Q'\}$ can be derived knowing only that if C is called with P holding, then Q holds on return. There is no requirement that C' should ensure P before calling C: only that if it does not, then it cannot depend on Q when the call returns.

Procedure Call

Consider a while language with a single procedure, whose body satisfies the specification $\{P\}\ \mathrm{body}\ \{Q\}$. Then we may derive the triple

$$\{P'\}\,\mathrm{CALL}\,\{Q'\} \qquad \text{if } \forall s,t.\ (P(s) \Rightarrow Q(t)) \ \Rightarrow\ (P'(s) \Rightarrow Q'(t))\ .$$

This matches the invocation rule of the Mobius Base Logic in D3.1

[See Nipkow 2001; Hofmann 1997]

Similarly, the corresponding weakest precondition is given

$$wp_{\mathrm{CALL}}(Q') = \lambda s.\forall t.((P(s) \Rightarrow Q(t)) \Rightarrow Q'(t))\ .$$

Notably, $wp_{\mathrm{CALL}}(\mathbf{true}) = \mathbf{true}$ irrespective of P, Q. [See Olderog 1983]

Consequences

This also affects the rule of consequence in a language with recursive procedures.

$$\frac{\{P\}\ C\ \{Q\}}{\{P'\}\ C'\ \{Q'\}} \qquad \begin{array}{l} \forall s, t. ((\forall z. P(z, s) \Rightarrow Q(z, t)) \\ \qquad \Rightarrow (\forall z. P'(z, s) \Rightarrow Q'(z, t))) \end{array}$$

This version is taken from [Nipkow 2001], and includes a treatment of auxiliary variables following [Kleymann 1999].

Note that these all follow precisely from the underlying semantics of triples:

$$\vDash \{P\}\ C\ \{Q\} \quad \iff \quad \forall s, t, z.\ (s \mathrel{-C\!\!\to} t \land P(z, s)) \Rightarrow Q(z, t)\ .$$

Types

Compare these two:

$$\{P\}\,C\,\{Q\} \vdash \{P'\}\,C'\,\{Q'\} \qquad\qquad F : A \to B \vdash M : T\,.$$

The typing assertion tells us that in any execution of M, function F will only be applied to arguments of type A.

["Well-typed programs cannot go wrong" — Milner 1978]

This impedes the coding of type assertions into classic Hoare logic. The situation is more severe when refined types express tighter constraints on procedure call. Suppose, for example, we wished to interpret:

```
void setSoundVolume(int v where 0 <= v <= 10)
int getCoordinate(window w, int n in {X,Y,W,H})
                                    // Constants X=1, Y=2, W=4, H=8
int getVectorElement(vector<n> v, int i where 0 <= i < n)
```

Alternative Semantics

As the existing Hoare rules are determined by the underlying semantics, to encode type-like assertions we propose modifying that semantics.

For a language with a single procedure, we annotate execution traces with a record of the states from which the procedure is invoked:

$$s \xrightarrow{C} t, S \qquad \text{where } S = [s_0, s_1, \ldots].$$

Procedure definition must now include an additional *requires* predicate R, and Hoare triples assert that it is satisfied on each call:

$$\models_R \{P\}\, C\, \{Q\} \quad \iff \quad \forall s, t.\ (s \xrightarrow{C} t, S \wedge P(s)) \Rightarrow (Q(t) \wedge \forall u \in S . R(u))\ .$$

This revised semantics resembles Meyer's "contract" notion in Eiffel.

New Rules for Old

The existing derivation rules are not sound for the new semantics, so we propose stronger ones, for procedure call:

$$\frac{\{P\}\ \mathrm{body}\ \{Q\}}{\{P\}\ \mathrm{CALL}\ \{Q\}} \quad \forall s. P(s) \Rightarrow R(s)$$

and for consequence:

$$\frac{\{P\}\ C\ \{Q\}}{\{P'\}\ C\ \{Q'\}} \quad \begin{aligned} &\forall s, t, S. ((P(s) \Rightarrow (Q(t) \land \forall u \in S. R(u))) \\ &\qquad\qquad \Rightarrow (P'(s) \Rightarrow (Q'(t) \land \forall u \in S. R(u)))) \end{aligned}$$

New Rules for Old

The existing derivation rules are not sound for the new semantics, so we propose stronger ones, for procedure call:

$$\frac{\{P\}\ \mathrm{body}\ \{Q\}}{\{P\}\ \mathrm{CALL}\ \{Q\}} \quad \forall s. P(s) \Rightarrow R(s)$$

and for consequence:

$$\frac{\{P\}\ C\ \{Q\}}{\{P'\}\ C\ \{Q'\}} \quad \begin{array}{l} \forall s,t.((P(s) \Rightarrow Q(t) \Rightarrow (P'(s) \Rightarrow (Q'(t)) \\ \qquad \wedge ((P'(s) \Rightarrow P(s)) \vee (\forall u.R(u)))) \end{array}$$

New Rules for Old

The existing derivation rules are not sound for the new semantics, so we propose stronger ones, for procedure call:

$$\frac{\{P\}\ \mathrm{body}\ \{Q\}}{\{P\}\ \mathrm{CALL}\ \{Q\}} \quad \forall s. P(s) \Rightarrow R(s)$$

and for consequence:

$$\frac{\{P\}\ C\ \{Q\}}{\{P'\}\ C\ \{Q'\}} \quad \begin{array}{r} \forall s, t.((P(s) \Rightarrow Q(t) \Rightarrow (P'(s) \Rightarrow (Q'(t)) \\ \wedge\ ((P'(s) \Rightarrow P(s))\ \vee\ (\forall u. R(u)))) \end{array}$$

Interestingly, apart from R, this is exactly Nipkow's rule for *total* Hoare triples.

New Rules for Old

The existing derivation rules are not sound for the new semantics, so we propose stronger ones, for procedure call:

$$\frac{\{P\}\ \mathrm{body}\ \{Q\}}{\{P\}\ \mathrm{CALL}\ \{Q\}} \quad \forall s. P(s) \Rightarrow R(s)$$

and for consequence:

$$\frac{\{P\}\ C\ \{Q\}}{\{P'\}\ C\ \{Q'\}} \quad \begin{array}{r} \forall s, t.((P(s) \Rightarrow Q(t) \Rightarrow (P'(s) \Rightarrow (Q'(t)) \\ \wedge\ ((P'(s) \Rightarrow P(s))\ \vee\ (\forall u. R(u)))) \end{array}$$

Interestingly, apart from R, this is exactly Nipkow's rule for *total* Hoare triples.

These revised rules are sound for the strict semantics of triples:

$$\vdash_R \{P\}\ C\ \{Q\} \quad \Longrightarrow \quad \vDash_R \{P\}\ C\ \{Q\}$$

Formalization

We have implemented this alternate semantics in Isabelle, working with Nipkow's existing formalization.

- While language with a single recursive procedure.
- Shallow embedding in Isabelle/HOL.
- Treatment of auxiliary variables via additional $\forall z \ldots$
- Extend operational semantics with record of calling states
- New definition of Hoare validity checks *required* predicate.
- New derivation rules as above.
- Proof of soundness.

The record of calling states is similar to previous use of resource algebras in Mobius. In the revised proof of soundness, it conveniently replaces Nipkow's induction over call depth.

Completeness and Nontermination

Nipkow has a proof of completeness based on Most General Triples (MGT), using an auxiliary state variable. We have yet to attempt to replicate this with the strict semantics.

It is not clear that we should expect straightforward success, though: consider procedure declaration

$$\mathrm{body} = (x := x - 1; \mathrm{CALL}) \qquad \text{requires } x > 0.$$

Then the partial triple $\{x = 5\}\ \mathrm{CALL}\ \{\mathrm{true}\}$ is valid (as nonterminating) but seems not to be derivable under strict invocation rules.

One response could be to extend the semantics to require satisfaction of strict preconditions also for nonterminating runs.

Next

We plan to continue and work with Nipkow's MGT proof of completeness, but in anticipation that it will require reworking the semantics to include partial executions

Once we have a sound and complete set rules for strict Hoare logic, we can then use it to model rich type systems, as intended.

Specifically, we have a reduced language *J Minor* with mutable state and object heap, and a corresponding small bytecode VM. It has a bytecode logic following the Mobius base logic, but with strict method invocation.

We propose to introduce refined types in the source language — integer enumerations, value ranges — and demonstrate their encoding in the bytecode logic.

Acknowledgements

Mobius: Mobility, Ubiquity and Security

Mobius is funded from 2005–2009 as project IST-015905 under the Global Computing II proactive initiative of the Future and Emerging Technologies objective in the Information Society Technologies priority of the European Commission's 6th Framework programme.

Disclaimer: This document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein.