Hoare's logic revisited

TINY

Generalising

Rather than just working with **Int**, consider an arbitrary underlying data type given by:

- Σ: an algebraic signature with sort $Bool$ and boolean constants and connectives
- $\mathcal{A}$: a Σ-structure with the boolean part interpreted in the standard way

$\text{TINY}_{\mathcal{A}}$

Syntax: As in TINY, except that:

- Σ-terms used instead of integer expressions
- variables classified by the sorts of Σ, assignments allowed only when the sorts of the variable and the term coincide
- Σ-terms of sort $Bool$ used instead of boolean expressions

Semantic domains: As in TINY, except with a modified notion of state:

$$\boxed{\mathbf{State}_{\mathcal{A}} = \mathbf{Var} \to |\mathcal{A}|}$$

(with variables and their values classified by the sorts of Σ)

Semantic functions: As in TINY, except that referring to $\mathcal{A}$ for interpretation of the operations on $|\mathcal{A}|$.

Hoare's logic

$$\boxed{\{\varphi\}\ S\ \{\psi\}}$$

— — — *as before* — — —

For instance

- add the following to the original signature Σ for $\textsc{Tiny}$:

$$\begin{array}{ll} \textbf{sorts} & Array; \\ \textbf{opns} & newarr\colon Array; \\ & put\colon Array \times Int \times Int \to Array; \\ & get\colon Array \times Int \to Int; \end{array}$$

- and expand the original algebra $\mathcal{A}$ for $\textsc{Tiny}$ as follows:

$$\begin{array}{ll} \textbf{carriers} & \mathcal{A}_{Array} = \mathbf{Int} \to \mathbf{Int} \\ \textbf{operations} & newarr_{\mathcal{A}}(j) = 0 \\ & put_{\mathcal{A}}(a, i, n) = a[i \mapsto n] \\ & get_{\mathcal{A}}(a, i) = a(i) \end{array}$$

Example

$$\{a{:}Array \wedge 0 \leq n\}$$
$$m := 0;$$
$$\textbf{while } \{0 \leq m \leq n \wedge \textit{is-sorted}(a,0,m)\}\ m+1 \leq n \textbf{ do}$$
$$m := m+1; k := m;$$
$$\textbf{while } \{0 \leq k \leq m \leq n \wedge \textit{is-nearly-sorted}(a,0,k,m)\}\ 1 \leq k \textbf{ do}$$
$$k := k-1;$$
$$\textbf{if } get(a,k) \leq get(a,k+1) \textbf{ then } k := 0$$
$$\textbf{else } x := get(a,k+1); a := put(a,k+1,get(a,k)); a := put(a,k,x)$$
$$\{\textit{is-sorted}(a,0,n)\}$$

where:

$$\textit{is-sorted}(a,i,j) \equiv a{:}Array \wedge \forall i',j'{:}Int.i \leq i' \leq j' \leq j \Rightarrow get(a,i') \leq get(a,j')$$

$$\textit{is-nearly-sorted}(a,i,k,j) \equiv \textit{is-sorted}(a,i,k-1) \wedge \textit{is-sorted}(a,k,j) \wedge \\ \forall i',j'{:}Int.(i \leq i' \leq k-1 \wedge k+1 \leq j' \leq j) \Rightarrow get(a,i') \leq get(a,j')$$

Hoare's logic: proof rules

— — — as before — — —

$$\frac{}{\{\varphi[x \mapsto e]\}\; x := e\; \{\varphi\}}$$

$$\frac{}{\{\varphi\}\; \mathbf{skip}\; \{\varphi\}}$$

$$\frac{\{\varphi\}\; S_1\; \{\theta\} \qquad \{\theta\}\; S_2\; \{\psi\}}{\{\varphi\}\; S_1; S_2\; \{\psi\}}$$

$$\frac{\{\varphi \wedge b\}\; S_1\; \{\psi\} \qquad \{\varphi \wedge \neg b\}\; S_2\; \{\psi\}}{\{\varphi\}\; \mathbf{if}\; b\; \mathbf{then}\; S_1\; \mathbf{else}\; S_2\; \{\psi\}}$$

$$\frac{\{\varphi \wedge b\}\; S\; \{\varphi\}}{\{\varphi\}\; \mathbf{while}\; b\; \mathbf{do}\; S\; \{\varphi \wedge \neg b\}}$$

$$\frac{\varphi' \Rightarrow \varphi \qquad \{\varphi\}\; S\; \{\psi\} \qquad \psi \Rightarrow \psi'}{\{\varphi'\}\; S\; \{\psi'\}}$$

Soundness

Fact: *Hoare's proof calculus is sound, that is:*

$$\boxed{\textit{if } \boxed{\mathcal{TH}(\mathcal{A}) \vdash \{\varphi\}\, S\, \{\psi\}} \quad \textit{then} \quad \boxed{\models_{\mathcal{A}} \{\varphi\}\, S\, \{\psi\}}}$$

Proof

— — — as before — — —

Toward completeness

We have to ensure that all the assertions necessary in the proofs may be formulated in the assertion logic.

Given $S \in \mathbf{Stmt}_\Sigma$ and $\psi \in \mathbf{Form}_\Sigma$, define:

$$\boxed{wpre_{\mathcal{A}}(S,\psi) = \{s \in \mathbf{State}_{\mathcal{A}} \mid \text{ if } \mathcal{S}_{\mathcal{A}}[\![S]\!]\, s = s' \in \mathbf{State}_{\mathcal{A}} \text{ then } \mathcal{F}_{\mathcal{A}}[\![\psi]\!]\, s' = \mathbf{tt}\}}$$

Definition: *First-order logic is expressive over $\mathcal{A}$ for $\text{TINY}_{\mathcal{A}}$ ($\mathcal{A}$ is expressive) if for all $S \in \mathbf{Stmt}_\Sigma$ and $\psi \in \mathbf{Form}_\Sigma$, there exists the weakest liberal precondition for S and ψ, that is, a formula $\varphi_0 \in \mathbf{Form}_\Sigma$ such that*

$$\boxed{\{\varphi_0\}_{\mathcal{A}} = wpre_{\mathcal{A}}(S,\psi)}$$

Relative completeness of Hoare's logic

(completeness in the sense of Cook)

Fact: *If $\mathcal{A}$ is expressive then Hoare's proof calculus is sound and relatively complete, that is:*

$$\boxed{\boxed{\mathcal{TH}(\mathcal{A}) \vdash \{\varphi\}\, S\, \{\psi\}} \quad \textit{iff} \quad \boxed{\models_{\mathcal{A}} \{\varphi\}\, S\, \{\psi\}}}$$

Proof: By structural induction on S. In fact: given expressivity and arbitrary use of facts from $\mathcal{TH}(\mathcal{A})$, all the cases go through easily!

Fact: *$\mathcal{A}$ is expressive if and only if either the standard model of Peano arithmetic is definable in $\mathcal{A}$, or for each $S \in \mathbf{Stmt}_\Sigma$, there is a finite bound on the number of states reached in any computation of S.*

Beyond TINY

Procedures: Given **proc** p **is** (S_p):

$$\frac{\{\varphi\}\,\mathbf{call}\ p\,\{\psi\} \vdash \{\varphi\}\,S_p\,\{\psi\}}{\{\varphi\}\,\mathbf{call}\ p\,\{\psi\}}$$

Not quite good enough; requires additional rules to manipulate auxiliary variables to ensure relative completeness

Variables: Given a fresh variable y:

$$\frac{\{\varphi \wedge y = ??\}\,S[x \mapsto y]\,\{\psi\}}{\{\varphi\}\,\mathbf{begin\ var}\ x\ S\ \mathbf{end}\,\{\psi\}}$$

etc. . .

But there are limits. . .

Fact: *There exists no Hoare's proof system which is sound and relatively complete in the sense of Cook for a programming language which admits recursive procedures with procedure parameters, local procedures and global variables with static binding.*

Key to the proof:

Fact: *The halting problem is undecidable for programs of such a language even for finite data types $\mathcal{A}$ (with at least two elements).*

Total correctness revisited

What about $\textsc{Tiny}_{\mathcal{A}}$*?*

GOOD NEWS:

Proving termination using well-founded relations works as before!

Still, recall the basic rule:

$$\frac{(nat(l) \land \varphi(l+1)) \Rightarrow b \qquad [nat(l) \land \varphi(l+1)]\; S\; [\varphi(l)] \qquad \varphi(0) \Rightarrow \neg b}{[\exists l.nat(l) \land \varphi(l)]\; \textbf{while}\; b\; \textbf{do}\; S\; [\varphi(0)]}$$

Problem?

Given a signature Σ, let Σ^+ be its extension by the language of (Peano) arithmetic: predicates $nat(\_)$ and $\_ \leq \_$, constants 0, 1, operations $\_ + \_$, $\_ - \_$, $\_ * \_$.

Let $\mathcal{A}$ be a Σ^+-structure; assume that the interpretation of $nat(\_)$ in $\mathcal{A}$ is closed under the arithmetical constants and operations as expected.

Even then:

the loop rule need not be sound for $\textsc{Tiny}_{\mathcal{A}}$

Serious trouble?

For instance, we will typically get:

$$\mathcal{TH}(\mathcal{A}) \vdash [nat(x)]\ \mathbf{while}\ x > 0\ \mathbf{do}\ x := x - 1\ [\mathbf{true}]$$

BUT: This is not valid for instance if $\mathcal{A}$ is a non-standard model of arithmetic.

Soundness and completeness

A Σ^+-structure $\mathcal{A}$ is *arithmetical* if the interpretations in $\mathcal{A}$ of the arithmetical operations and predicates restricted to those elements $n \in |\mathcal{A}|$ for which $nat(n)$ holds in $\mathcal{A}$ form *the standard model of arithmetic*.

Fact: *If $\mathcal{A}$ is arithmetical then*

$$\text{if } \boxed{\mathcal{TH}(\mathcal{A}) \vdash [\varphi]\, S\, [\psi]} \text{ then } \boxed{\models_{\mathcal{A}} [\varphi]\, S\, [\psi]}$$

Soundness

If moreover, finite sequences of elements in $|\mathcal{A}|$ can be encoded using a formula as a single element in $|\mathcal{A}|$, then

$$\boxed{\mathcal{TH}(\mathcal{A}) \vdash [\varphi]\, S\, [\psi]} \text{ iff } \boxed{\models_{\mathcal{A}} [\varphi]\, S\, [\psi]}$$

Soundness & completeness