Hoare’s logic revisited

Generalising

Rather than just working with \texttt{Int}, consider an arbitrary underlying data type given by:

- $\Sigma$: an algebraic signature with sort $\texttt{Bool}$ and boolean constants and connectives
- $A$: a $\Sigma$-structure with the boolean part interpreted in the standard way
Syntax: As in \textsc{Tiny}, except that:

- \( \Sigma \)-terms used instead of integer expressions
- variables classified by the sorts of \( \Sigma \), assignments allowed only when the sorts of the variable and the term coincide
- \( \Sigma \)-terms of sort \textit{Bool} used instead of boolean expressions

Semantic domains: As in \textsc{Tiny}, except with a modified notion of state:

\[
\text{State}_\mathcal{A} = \text{Var} \rightarrow |\mathcal{A}|
\]

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

Semantic functions: As in \textsc{Tiny}, except that referring to \( \mathcal{A} \) for interpretation of the operations on \( |\mathcal{A}| \).
Hoare’s logic

\{ \phi \} S \{ \psi \}

— — — as before — — —
For instance

- add the following to the original signature $\Sigma$ for $\text{TINY}$:

\[
\begin{align*}
\text{sorts} & \quad Array; \\
\text{opns} & \quad \text{newarr}: Array; \\
& \quad \text{put}: \text{Array} \times \text{Int} \times \text{Int} \to \text{Array}; \\
& \quad \text{get}: \text{Array} \times \text{Int} \to \text{Int};
\end{align*}
\]

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

\[
\begin{align*}
\text{carriers} & \quad \mathcal{A}_{\text{Array}} = \text{Int} \to \text{Int} \\
\text{operations} & \quad \text{newarr}_\mathcal{A}(j) = 0 \\
& \quad \text{put}_\mathcal{A}(a, i, n) = a[i \mapsto n] \\
& \quad \text{get}_\mathcal{A}(a, i) = a(i)
\end{align*}
\]
Example

\{a: Array ∧ 0 ≤ n\}

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

where:

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

\( is-nearly-sorted(a, i, k, j) \equiv is-sorted(a, i, k - 1) \land is-sorted(a, k, j) \land \forall i', j': \text{Int}. i \leq i' \leq k - 1 \land k + 1 \leq j' \leq j \Rightarrow get(a, i') \leq get(a, j') \)
Hoare’s logic: proof rules

— — — as before — — —

\[
\begin{align*}
\{\varphi[x \mapsto e]\} & \ x := e \ \{\varphi\} \\
\{\varphi\} & \ S_1 \ \{\theta\} \quad \{\theta\} \ S_2 \ \{\psi\} \\
\{\varphi\} & \ S_1; S_2 \ \{\psi\} \\
\{\varphi \land b\} & \ S \ \{\varphi\} \\
\{\varphi\} & \ \text{while } b \ \text{do } S \ \{\varphi \land \neg b\} \\
\{\varphi \land b\} & \ S_1 \ \{\psi\} \quad \{\varphi \land \neg b\} \ S_2 \ \{\psi\} \\
\{\varphi\} & \ \text{if } b \ \text{then } S_1 \ \text{else } S_2 \ \{\psi\} \\
\varphi' & \Rightarrow \varphi \quad \{\varphi\} \ S \ \{\psi\} \quad \psi \Rightarrow \psi' \\
\{\varphi'\} & \ S \ \{\psi'\}
\end{align*}
\]
**Fact:** Hoare’s proof calculus is sound, that is:

\[
\text{if } \mathcal{T}H(A) \vdash \{\varphi\} S \{\psi\} \text{ then } \models_A \{\varphi\} S \{\psi\}
\]

**Proof**

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

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

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

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

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

\[ \mathcal{T}\mathcal{H}(\mathcal{A}) \vdash \{ \varphi \} S \{ \psi \} \text{ iff } \models_{\mathcal{A}} \{ \varphi \} S \{ \psi \} \]

Proof: By structural induction on $S$. In fact: given expressivity and arbitrary use of facts from $\mathcal{T}\mathcal{H}(\mathcal{A})$, 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 $S \in \text{Stmt}_\Sigma$, there is a finite bound on the number of states reached in any computation of $S$. 

**Procedures:** Given \( \text{proc } p \text{ is } (S_p): \)

\[
\{ \varphi \} \text{ call } p \{ \psi \} \vdash \{ \varphi \} S_p \{ \psi \} \\
\{ \varphi \} \text{ call } p \{ \psi \}
\]

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

**Variables:** Given a fresh variable \( y: \)

\[
\{ \varphi \land y = \text{??} \} S[x \mapsto y] \{ \psi \} \\
\{ \varphi \} \text{ begin var } x \text{ S end } \{ \psi \}
\]

e tc...
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 $\text{TINY}_A$?

**GOOD NEWS:**

> Proving termination using well-founded relations works as before!

Still, recall the basic rule:

\[
\begin{align*}
(nat(l) \land \varphi(l + 1)) & \Rightarrow b & [nat(l) \land \varphi(l + 1)] S [\varphi(l)] & \varphi(0) \Rightarrow \neg b \\
[\exists l. nat(l) \land \varphi(l)] & \text{while } b \text{ do } S [\varphi(0)]
\end{align*}
\]
Given a signature $\Sigma$, let $\Sigma^+$ be its extension by the language of (Peano) arithmetic: predicates $nat(\_)$ and $\_ \leq \_$, constants $0, 1$, operations $\_ + \_, \_ - \_, \_ \ast \_$. Let $\mathcal{A}$ be a $\Sigma^+$-structure; assume that the interpretation of $nat(\_)$ in $\mathcal{A}$ is closed under the arithmetical constants and operations as expected.

Even then:

\textbf{the loop rule need not be sound for $\mathsf{TINY}_\mathcal{A}$}

For instance, we will typically get:

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

\textbf{BUT:} This is not valid for instance if $\mathcal{A}$ is a non-standard model of arithmetic.
A \( \Sigma^+ \)-structure \( \mathcal{A} \) is \textit{arithmetical} if the interpretations in \( \mathcal{A} \) of the arithmetical operations and predicates restricted to those elements \( n \in |\mathcal{A}| \) for which \( \text{nat}(n) \) holds in \( \mathcal{A} \) form the standard model of arithmetic.

\textbf{Fact:} If \( \mathcal{A} \) is arithmetical then

\[
\text{if } \mathcal{T\mathcal{H}}(\mathcal{A}) \vdash [\varphi] S [\psi] \text{ then } \models_\mathcal{A} [\varphi] S [\psi]
\]

Soundness

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

\[
\mathcal{T\mathcal{H}}(\mathcal{A}) \vdash [\varphi] S [\psi] \text{ iff } \models_\mathcal{A} [\varphi] S [\psi]
\]

Soundness & completeness