Hoare’s logic revisited

Tiny

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
- $\mathcal{A}$: a $\Sigma$-structure with the boolean part interpreted in the standard way
**Syntax:** As in $\text{TINY}$, 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 $\text{Bool}$ used instead of boolean expressions

**Semantic domains:** As in $\text{TINY}$, 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 $\text{TINY}$, except that referring to $\mathcal{A}$ for interpretation of the operations on $|\mathcal{A}|$. 
Hoare’s logic

\{\varphi\} \text{ } S \text{ } \{\psi\}

— — — as before — — —
For instance

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

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

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

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

\begin{align*}
m & := 0; \\
\textbf{while} & \{ 0 \leq m \leq n \land \text{is-sorted}(a, 0, m) \} \quad m + 1 \leq n \ \textbf{do} \\
m & := m + 1; \quad k := m; \\
\textbf{while} & \{ 0 \leq k \leq m \leq n \land \text{is-nearly-sorted}(a, 0, k, m) \} \quad 1 \leq k \ \textbf{do} \\
k & := k - 1; \\
\textbf{if} & \ \text{get}(a, k) \leq \text{get}(a, k + 1) \ \textbf{then} \quad k := 0 \\
\textbf{else} & \quad x := \text{get}(a, k + 1); \quad a := \text{put}(a, k + 1, \text{get}(a, k)); \quad a := \text{put}(a, k, x) \\
\{ \text{is-sorted}(a, 0, n) \}\end{align*}

where:

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

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

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

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

**Soundness**

**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:

$$wpre_A(S, \psi) = \{ s \in \text{State}_A \mid \text{if } S_A[S] s = s' \in \text{State}_A \text{ then } F_A[\psi] s' = \text{tt} \}$$

**Definition:** First-order logic is expressive over $A$ for TINY$_A$ ($A$ 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

$$\{ \varphi_0 \}_A = wpre_A(S, \psi)$$
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:

$$
\mathcal{T\mathcal{H}}(\mathcal{A}) \vdash \{\varphi\} S \{\psi\} \iff \models_{\mathcal{A}} \{\varphi\} S \{\psi\}
$$

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 \( y \):

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

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:

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

Even then:

\begin{center}
\textit{the loop rule need not be sound for $\text{TINY}_\mathcal{A}$}
\end{center}

For instance, we will typically get:

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

\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 \textit{the standard model of arithmetic}.

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

\begin{align*}
\text{if } \mathcal{T}\mathcal{H}(\mathcal{A}) &\vdash [\varphi] S [\psi] \text{ then } \models_{\mathcal{A}} [\varphi] S [\psi] \\
\text{Soundness} &\\
\text{If moreover, finite sequences of elements in } |\mathcal{A}| \text{ can be encoded using a formula as a single element in } |\mathcal{A}|, \text{ then} \\
\mathcal{T}\mathcal{H}(\mathcal{A}) &\vdash [\varphi] S [\psi] \text{ iff } \models_{\mathcal{A}} [\varphi] S [\psi] \\
\text{Soundness} & \& \text{ completeness}
\end{align*}