Handling of Loops

by Christoph Gladisch

Induction rule:

\[
\Gamma \Rightarrow H(0) \quad \forall i, H(i) \Rightarrow H(i + 1) \quad \forall i. H(i) \Rightarrow \varphi
\]

In order to prove \( \varphi \) find a more general formula \( H(i) \) such that it can be instantiated and \( \forall i. H(i) \Rightarrow \varphi \) can be proven. \( \varphi \) is often an instance of \( H(i) \) so that for some term \( t \) it even holds that \( H(t) \equiv \varphi \).

Loop induction. Heuristics for finding an induction hypothesis:

For a formula \( \varphi \):

\[ \varphi = \{ \text{V} \}\{ \text{while(c){body}}\}\text{Post} \]

the induction hypothesis \( H(i) \) has often the form:

\[ H(i) = \{ \text{V} \}\{ \text{HPre(i)} \rightarrow \{ \text{while(c){body}}\}\text{Post} \} \]

Note the similarity to the original formula. The postfix \( \{ \text{while(c){body}}\}\text{Post} \) remains unchanged. Only the update \( \{ \text{V} \}\) and the formula \( \text{HPre(i)} \) have to be chosen appropriately. The prefix \( \{ \text{V} \}\) determines the state before the execution of the loop. \( \{ \text{V} \} \) assigns program variables to the right state and \( \text{HPre(i)} \) puts additional constrains on the state to filter out for instance invalid ranges etc.

\( \text{HPre(i)} \) has to be chosen such that:

1. \( \text{HPre(0)} \rightarrow (c \equiv \text{false}) \) the loop terminates and
2. \( \text{HPre(0)} \rightarrow \text{Post} \)

Because of (2) \( \text{HPre(0)} \) and \( \text{Post} \) are very similar. Sometimes it is that \( \text{HPre(0)} \equiv \text{Post} \).

3. \( \text{HPre(n)} \) describes the state before the execution of the loop.

Figure 5. illustrates the states of the prefix before the loop is entered and for the case when the loop iterates 0 times.

\[ \text{Figure 5. Visualisation of the semantics of the Prefix } \{ \text{V} \}\{ \text{HPre(i)} \} \text{ of the induction hypothesis.} \]

Figure 6. Illustrates what happens at the induction step.
A frequent pattern in the step case:

\[
\begin{align*}
C \Rightarrow A & \quad B \Rightarrow D \\
C \Rightarrow A, D & \quad B, C \Rightarrow D \\
\Rightarrow A \rightarrow B, C \Rightarrow D
\end{align*}
\]

With induction hypothesis pattern:

\[
\begin{align*}
\text{HPre}(i+1) \Rightarrow \text{HPre}(i) & \quad \{U(i)\}\{\text{while}(c)\}\text{Body}\text{Post}, \text{HPre}(i+1) \Rightarrow \{U(i+1)\}\{\text{while}(c)\}\text{Body}\text{Post} \\
\text{HPre}(i) \Rightarrow \text{HPre}(i) & \quad \{U(i)\}\{\text{while}(c)\}\text{Body}\text{Post}, \text{HPre}(i+1) \Rightarrow \{U(i+1)\}\{\text{while}(c)\}\text{Body}\text{Post} \\
\Rightarrow \text{HPre}(i) \rightarrow \{U(i)\}\{\text{while}(c)\}\text{Body}\text{Post}, \text{HPre}(i+1) \Rightarrow \{U(i+1)\}\{\text{while}(c)\}\text{Body}\text{Post} \\
\Rightarrow \text{HPre}(i) \rightarrow \{U(i)\}\{\text{while}(c)\}\text{Body}\text{Post} \Rightarrow \text{HPre}(i+1) \Rightarrow \{U(i+1)\}\{\text{while}(c)\}\text{Body}\text{Post}
\end{align*}
\]

Frequent problem after unwinding the loop of the right formula in the induction step:

\[
\{i := x||n := y\}\{\text{while}(c)\{..m=z..\}\} P \Rightarrow \{i := x||m := z||n := y\}\{\text{while}(c)\{..m=z..\}\} P
\]

Solution: Generalise the induction hypothesis in the form:

\[
\forall M. \{i := x||n := y\}\{m := M\}\text{HPre} \Rightarrow (\text{while}(c)\{..m=z..\}) P
\]

Classic Invariant rule:

\[
\begin{align*}
\Gamma \Rightarrow \{U\} & \Rightarrow \text{inv}, c \Rightarrow \text{inv}, \neg c \Rightarrow \text{Post} \\
\Gamma \Rightarrow \{U\}\{\text{while}(c)\}\text{Body}\text{Post}
\end{align*}
\]

In KeY without modifies clause:

\[
\begin{align*}
\Gamma \Rightarrow \{U\} & \Rightarrow \text{inv} \Rightarrow (\text{inv} \rightarrow (\text{body}\text{inv}) \Rightarrow \text{inv} \rightarrow \neg c \rightarrow \text{Post} \\
\Gamma \Rightarrow \{U\}\{\text{while}(c)\}\text{Body}\text{Post}
\end{align*}
\]

In KeY with modifies clause:

\[
\begin{align*}
\Gamma \Rightarrow \{U\} & \Rightarrow \{U\}\{M\} \Rightarrow (\text{inv} \rightarrow (\text{body}\text{inv}) \Rightarrow \text{inv} \rightarrow \neg c \rightarrow \text{Post} \\
\Gamma \Rightarrow \{U\}\{M\}\{\text{while}(c)\}\text{Body}\text{Post}
\end{align*}
\]

For every program variable in the modifies clause an update \{M\} is created that replaces the modifies program variable by a fresh program variable or in other words by a new skolem function.