Spezifikation und Implementierung von Zählerobjekten

Unsere Spezifikation für Zählerobjekte lautet

$$
\begin{array}{lcl}
\mathit{counter\_spec} & = & \lambda c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int} \rangle. \\
& & \quad \exists\, \mathit{cview} : \mathbf{int} \to \mathbf{bool}. \\
& & \qquad \mathit{cview}\ 0 \wedge \\
& & \qquad \forall\, i : \mathbf{int}. \{\mathit{cview}\ i\}\, c\#\mathit{inc}\, \{\mathit{cview}\ (i+1)\} \wedge \\
& & \qquad \forall\, i : \mathbf{int}. \{\mathit{cview}\ i\}\, c\#\mathit{get}\, \{\mathbf{returns}\ i\}
\end{array}
$$

Die naheliegende Implementierung für einen Zählergenerator ist

$$
\begin{array}{lcl}
\mathit{new\_counter\_1} & = & \lambda().\ \mathbf{object} \\
& & \qquad \mathbf{val}\ x = \mathit{ref}\ 0 \\
& & \qquad \mathbf{method}\ \mathit{inc} = x\ :=\ !\, x + 1 \\
& & \qquad \mathbf{method}\ \mathit{get} = !\, x \\
& & \quad \mathbf{end}
\end{array}
$$

Für diese Implementierung gilt folgende *konkrete Spezifikation*

$$
\begin{array}{l}
\{\mathit{true}\}\ \mathit{new\_counter\_1}\ ()\ \{\mathbf{returns}\ c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int} \rangle. \\
\qquad \exists\, x : \mathbf{int\ ref}. \\
\qquad\quad !\, x = 0 \wedge \\
\qquad\quad \forall\, i : \mathbf{int}. \{!\, x = i\}\, c\#\mathit{inc}\, \{!\, x = i + 1\} \wedge \\
\qquad\quad \forall\, i : \mathbf{int}. \{!\, x = i\}\, c\#\mathit{get}\, \{\mathbf{returns}\ i\}\}
\end{array}
$$

Um daraus die gewünschte *abstrakte Spezifikation*

$$\{\mathit{true}\}\ \mathit{new\_counter\_1}\ ()\ \{\mathbf{returns}\ c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int} \rangle.\ \mathit{counter\_spec}\ c\}$$

zu erhalten, verwenden wir die *consequence rule* zur Abschwächung der *post-condition*. Also bleibt folgende *verification condition* zu beweisen

$$
\begin{array}{l}
\forall\, c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int} \rangle. \\
\quad (\exists\, x : \mathbf{int\ ref}. \ldots x \ldots) \Rightarrow (\exists\, \mathit{cview} : \mathbf{int} \to \mathbf{bool}. \ldots \mathit{cview} \ldots)
\end{array}
$$

Sie lässt sich mit prädikatenlogischen Regeln zurückführen auf

$$\ldots x \ldots \Rightarrow (\exists\, \mathit{cview} : \mathbf{int} \to \mathbf{bool}. \ldots \mathit{cview} \ldots)$$

und letzteres beweist man, indem man $\mathit{cview} = \lambda i : \mathbf{int}.\ !\, x = i$ als Beispiel für die Existenzaussage wählt.

Eine alternative Implementierung für einen Zählergenerator ist

$$\begin{array}{lcl} \mathit{new\_counter\_2} & = & \lambda().\ \mathbf{object} \\ & & \quad \mathbf{val}\ x = \mathit{ref}\ 0 \\ & & \quad \mathbf{method}\ \mathit{inc} = x\ :=\ !\,x + 2 \\ & & \quad \mathbf{method}\ \mathit{get} = !\,x \bmod 2 \\ & & \mathbf{end} \end{array}$$

Für *new_counter_2* gilt eine andere *konkrete Spezifikation*, nämlich

$$\begin{array}{l} \{\mathit{true}\}\ \mathit{new\_counter\_2}\,()\,\{\mathbf{returns}\ c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int}\rangle. \\ \qquad \exists\, x : \mathbf{int\,ref}. \\ \qquad\quad !\,x = 0 \wedge \\ \qquad\quad \forall\, i : \mathbf{int}.\, \{!\,x = 2 * i\}\ c\#\mathit{inc}\ \{!\,x = 2 * (i+1)\} \wedge \\ \qquad\quad \forall\, i : \mathbf{int}.\, \{!\,x = 2 * i\}\ c\#\mathit{get}\ \{\mathbf{returns}\ i\}\} \end{array}$$

Aber nach wie vor erhält man die gleiche *abstrakte Spezifikation*

$$\{\mathit{true}\}\ \mathit{new\_counter\_2}\,()\,\{\mathbf{returns}\ c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int}\rangle.\ \mathit{counter\_spec}\ c\}$$

indem man die *postcondition* mit Hilfe der *consequence rule* abschwächt. Die verbleibende *verification condition*

$$\begin{array}{l} \forall\, c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int}\rangle. \\ \quad (\exists\, x : \mathbf{int\,ref}. \ldots x \ldots) \Rightarrow (\exists\, \mathit{cview} : \mathbf{int} \to \mathbf{bool}. \ldots \mathit{cview} \ldots) \end{array}$$

beweist man dieses Mal mit $\mathit{cview} = \lambda i : \mathbf{int}.\ !\,x = 2 * i$ als Beispiel für die Existenzaussage.

Basierend auf der Zählerspezifikation lässt sich nun die Korrektheit von *abstrakten Programmen* beweisen, die mit einem Zähler arbeiten, z.B.

$$\begin{array}{l} \forall\, c : \langle \mathit{inc} : \mathbf{unit}; \mathit{get} : \mathbf{int}\rangle. \\ \quad \mathit{counter\_spec}\ c \Rightarrow \{\mathit{true}\}\ c\#\mathit{inc};\ c\#\mathit{inc};\ c\#\mathit{get}\ \{\mathbf{returns}\ 2\} \end{array}$$

Daraus erhält man dann die Korrektheit von *konkreten Programmen*, die mit einer speziellen Implementierung eines Zählers arbeiten, z.B.

$$\{\mathit{true}\}\ \mathbf{let}\ c = \mathit{new\_counter\_1}\,()\ \mathbf{in}\ c\#\mathit{inc};\ c\#\mathit{inc};\ c\#\mathit{get}\ \{\mathbf{returns}\ 2\}$$

oder

$$\{\mathit{true}\}\ \mathbf{let}\ c = \mathit{new\_counter\_2}\,()\ \mathbf{in}\ c\#\mathit{inc};\ c\#\mathit{inc};\ c\#\mathit{get}\ \{\mathbf{returns}\ 2\}$$

Bemerkung:

Mit den bisherigen Überlegungen lässt sich nur die Korrektheit von abstrakten Programmen beweisen, die auf einem einzigen Objekt arbeiten. Sobald mehr als ein Objekt im Spiel ist, muss man wissen, dass jedes Objekt ausschließlich über seine eigenen Methoden "erreichbar" ist. Dazu bedarf es einer weiteren Spezifikation, die sich (vermutlich) unabhängig von der bisherigen formulieren lässt.