Technical report

Lasse Letager Hansen 201508114

March 10, 2019

1 Semantics

1.1 pWhile

Defined for some context {ident : eqType} {mem : memType ident}. We have:

$$\begin{aligned}
\texttt{cmd} := {} & \texttt{abort} \\
\mid {} & \texttt{skip} \\
\mid {} & \texttt{assign}\ \{\texttt{t}\}\ \texttt{name}\ \texttt{expr} \\
\mid {} & \texttt{random}\ \{\texttt{t}\}\ \texttt{distr} \\
\mid {} & \texttt{cond}\ \texttt{boolexp}\ \texttt{cmd}\ \texttt{cmd} \\
\mid {} & \texttt{while}\ \texttt{expr}\ \texttt{cmd} \\
\mid {} & \texttt{seqc}\ \texttt{cmd}\ \texttt{cmd}
\end{aligned}$$

$$\begin{aligned}
(\texttt{expr} : \texttt{T}) := {} & \texttt{var}\ \{\texttt{T}\}\ \texttt{vars} && \texttt{expr T} \\
\mid {} & \texttt{cst}\ \{\texttt{T}\}\ \texttt{value} && \texttt{expr T} \\
\mid {} & \texttt{prp}\ \texttt{m} && \texttt{expr bool} \\
\mid {} & \texttt{app}\ \{\texttt{T U}\}\ (\texttt{expr} : \texttt{T} \rightarrow \texttt{U})\ (\texttt{expr} : \texttt{T}) && \texttt{expr U}
\end{aligned}$$

$$\begin{aligned}
\texttt{vars} &\subseteq \Sigma^* \\
\texttt{value} &\in V
\end{aligned}$$

1.2 Rml

$$
\begin{aligned}
\texttt{Inductive Rml} : \texttt{Type} := {} & \texttt{Var} : \mathbb{N} \to \texttt{Rml} \\
& \mid \texttt{Const} : \forall\ (A : \texttt{Type}), A \to \texttt{Rml} \\
& \mid \texttt{Let\_stm} : \mathbb{N} \to \texttt{Rml} \to \texttt{Rml} \to \texttt{Rml} \\
& \mid \texttt{If\_stm} : \texttt{Rml} \to \texttt{Rml} \to \texttt{Rml} \to \texttt{Rml} \\
& \mid \texttt{App\_stm} : \texttt{Type} \to \texttt{Rml} \to \texttt{Rml} \to \texttt{Rml}.
\end{aligned}
$$

2 Rml in Coq

2.1 Well Formed

A Rml program is well formed when there are no variables not bound. So all consts are well formed, all variables are well formed if there is a binding for it in the environment. all let statements are well formed, if what is being assigned is well formed under the current environment, and what it is binded in is well formed under the current environment extended with the new bounded variable. All if and app statements are well formed, if each subexpression is well formed in the current environment.

2.2 Replace var with value

Given a Rml expression, we can substitute the value of a variable for the expression by induction. This may still leave `Var` n expressions

Fixpoint replace$_var_with_value(x : Rml)(index : nat)(value : Rml) : Rml := matchxwith|Varn =>$
$ifn == indexthenvalueelsex|ConstAc => x|Let_stmnab => letnew_value := replace_var_with_valueaindexvalueinifn =$
$indexthenreplace_var_with_valuebindexnew_valueelseLet_stmnnew_value(replace_var_with_valuebindexvalue)|If_stmbm1m2$
$letb' := replace_var_with_valuebindexvalueinletm1' := replace_var_with_valuem1indexvalueinletm2' := replace_var_with_va$
$lete1' := replace_var_with_valuee1indexvalueinlete2' := replace_var_with_valuee2indexvalueinApp_stmBe1'e2'end.$

2.3 sRml