RoboScene to CSP Translation Rules

October 2, 2024

$$[\![\mathsf{caps} : \texttt{Capabilities};\ \ \mathsf{seqdgrmgrps} : \texttt{SequenceDiagramGroup}]\!]_D \\ \widehat{=}\ [\![\mathsf{caps}]\!]_{CS}\ [\![\mathsf{seqdgrmgrps}]\!]_{SDS}$$

$$[\![\mathsf{def}]\!]_{SDS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{SD}$$
$$[\![\mathsf{def\ defS}]\!]_{SDS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{SD}\ [\![\mathsf{defS}]\!]_{SDS}$$

$$[\![\mathsf{constant} : \texttt{Constants};\ \ \mathsf{lifelines} : \texttt{LifelineBlock}]\!]_{CS}\ \widehat{=} \\ [\![\mathsf{constant}]\!]_{CO}\ [\![\mathsf{lifelines}]\!]_{LS}$$

$$[\![\mathsf{timeunit} : \texttt{TimeUnit},\ \mathsf{consts} : \texttt{VariableList}]\!]_{CO} \\ \widehat{=}\ [\![\mathsf{timeunit}]\!]_{TU}\ [\![\mathsf{consts}]\!]_{CONS}$$

$$[\![\mathsf{def}]\!]_{CONS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{CON}$$
$$[\![\mathsf{def\ defS}]\!]_{CONS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{CON}\ [\![\mathsf{defS}]\!]_{CONS}$$

$$[\![(\underline{\text{const } \mathsf{constID}\ =\ \mathsf{val}}) : \texttt{VariableList}]\!]_{CON}\ \widehat{=}\ \mathsf{constID} = \mathsf{val}$$

$[\![\mathsf{lifeline} : \texttt{Actor};\ \ \mathsf{ins}, \mathsf{outs} : \texttt{LifelineEvents};\ \ \mathsf{vars} : \texttt{VarsList}]\!]_{LS}$
$\quad \widehat{=}\ [\![\mathsf{outs}]\!]_{OS}\ [\![\mathsf{ins}]\!]_{IS}\ [\![\mathsf{vars}]\!]_{VS}$

$[\![\mathsf{def}]\!]_{OS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{O}$
$[\![\mathsf{def\ defS}]\!]_{OS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{O}\ [\![\mathsf{defS}]\!]_{OS}$
$[\![\mathsf{def}]\!]_{IS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{I}$
$[\![\mathsf{def\ defS}]\!]_{IS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{I}\ [\![\mathsf{defS}]\!]_{IS}$
$[\![\mathsf{def}]\!]_{VS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{V}$
$[\![\mathit{def},\ \mathsf{defS}]\!]_{VS}\ \widehat{=}\ [\![\mathsf{def}]\!]_{V}\ [\![\mathsf{defS}]\!]_{VS}$

$[\![(\underline{\mathrm{out}}\ \mathsf{event} - > \mathsf{actor}) : \texttt{LifelineEvents}]\!]_{O}\ \widehat{=}\ \boldsymbol{channel}\ \mathsf{event} : \underline{\mathrm{InOut}}$
$[\![(\underline{\mathrm{out}}\ \mathsf{event} : \mathsf{Type} - > \mathsf{actor}) : \texttt{LifelineEvents}]\!]_{O}$
$\quad \widehat{=}\ \boldsymbol{channel}\ \mathsf{event} : [\![\mathsf{Type}, \mathsf{event}]\!]_{TO}$
$[\![(\underline{\mathrm{in}}\ \mathsf{event} < - \mathsf{actor}) : \texttt{LifelineEvents}]\!]_{I}\ \widehat{=}\ \boldsymbol{channel}\ \mathsf{event} : \underline{\mathrm{InOut}}$
$[\![(\underline{\mathrm{in}}\ \mathsf{event} : \mathsf{Type} < - \mathsf{actor}) : \texttt{LifelineEvents}]\!]_{I}$
$\quad \widehat{=}\ \boldsymbol{channel}\ \mathsf{event} : [\![\mathsf{Type}, \mathsf{event}]\!]_{TI}$

$[\![(\mathsf{var} : \mathsf{Type}) : \texttt{VariableList}]\!]_{V}\ \widehat{=}$
$\quad \boldsymbol{channel}\ \mathit{set}\_\mathsf{var} :\ \mathsf{Type}$
$\quad \boldsymbol{channel}\ \mathit{get}\_\mathsf{var} :\ \mathsf{Type}$
$\quad \mathit{Memory}\_\mathsf{var}(\mathsf{var})\ =\ \mathit{get}\_\mathsf{var}!\mathsf{var}\ \rightarrow\ \mathit{Memory}\_\mathsf{var}(\mathsf{var})$
$\qquad \Box\ \mathit{set}\_\mathsf{var}?x\ \rightarrow \mathit{Memory}\_\mathsf{var}(x)$
$\qquad \Box\ \mathit{terminate}\ \rightarrow \boldsymbol{Skip}$
$\quad \mathit{Memory}\ =\ \underline{(\|\ \mathsf{var}\ \bullet\ \{\mathrm{terminate}\}\ \circ\ \mathrm{Memory\_var(var)})}$

$[\![\mathsf{name} : \texttt{EString};\ \mathsf{trait} : \texttt{EString};\ \mathsf{actors} : \texttt{Actor};\ \mathsf{sequences} : \texttt{Sequence}]\!]_{SD}$
$\quad \widehat{=}\ [\![\mathsf{actors}]\!]_A\ [\![\mathsf{sequences}]\!]_{SS}$

$[\![\mathsf{def}]\!]_{SS}\ \widehat{=}\ [\![\mathsf{def}]\!]_S$
$[\![\mathsf{def}\ \mathsf{defS}]\!]_{SS}\ \widehat{=}\ [\![\mathsf{def}]\!]_S\ [\![\mathsf{defS}]\!]_{SS}$

$[\![\mathsf{actors} : \texttt{Actor};\ \mathsf{vars} : \texttt{VarsList};\ \mathsf{frags} : \texttt{InteractionFragment}]\!]_S\ \widehat{=}\ [\![\mathsf{frags}]\!]_{FS}$
$\quad ((((\Big\|\ \mathrm{a} : \mathsf{actors}(\mathrm{a_1}, ..., \mathrm{a_n})\ \bullet\ \mathrm{alpha(a)}\ \circ\ \mathrm{lifeline(a)})$
$\quad |[\ \{str, par, terminate\}\ ]|\ \underline{Control(\mathrm{parFrags}, \mathrm{strFrags})})$
$\quad |[\ \{alt, opt, loop, guard, terminate\}\ ]|\ \underline{Guard(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})})$
$\quad |[\ sharedVars\ \cup\ \{terminate\}\ ]|\ Memory) \setminus \{alt, opt, loop, guard, str, par\}$

$\quad \underline{\mathrm{alpha(a)}}\ =\ \alpha(a_x, \mathsf{frags})\ \cap\ \alpha(a_{x+1}, \mathsf{frags})$
$\quad \underline{\mathrm{lifeline(a)}}\ =\ [\![\mathsf{frags}_1]\!]^F_{(a)}\ ;\ ...\ ;\ [\![\mathsf{frags}_n]\!]^F_{(a)}$
$\quad \underline{Control(\mathrm{parFrags}, \mathrm{strFrags})}\ =\ \underline{Parallel(\mathrm{parFrags})}\ |[\ \{terminate\}\ ]|\ \underline{Strict(\mathrm{strFrags})}$

$\quad \textbf{\textit{channel}}\ alt :\ IDs.Int$
$\quad \textbf{\textit{channel}}\ opt :\ IDs.Int$
$\quad \textbf{\textit{channel}}\ loop :\ IDs.Int$
$\quad \textbf{\textit{channel}}\ par :\ IDs.Int$
$\quad \textbf{\textit{channel}}\ str :\ IDs.Int$
$\quad \textbf{\textit{channel}}\ guard :\ IDs.Int.Int.Bool$
$\quad datatype\ IDs\ =\ ID\_ALT \mid ID\_OPT \mid ID\_LOOP \mid ID\_PAR \mid ID\_STR$

$\quad [\![\mathsf{frags}]\!]_{FS}\ \widehat{=}\ [\![\mathsf{frag}]\!]_F$
$\quad [\![\mathsf{frag}\ \mathsf{frags}]\!]_{FS}\ \widehat{=}\ [\![\mathsf{frag}]\!]_F\ [\![\mathsf{frags}]\!]_{FS}$

$[\![\underline{(\mathrm{wait(x)\ on\ a})} : \texttt{WaitOccurrence}]\!]_F\ =\ Wait(\mathsf{x})$
$[\![\underline{(\mathrm{wait([v,y])\ on\ a})} : \texttt{WaitOccurrence}]\!]_F\ =\ (\sqcap\, x : \{\mathsf{v}..\mathsf{y}\} \bullet wait(x))$

$[\![\underline{(\mathrm{deadline(x)\ b})} : \texttt{DeadlineFragment}]\!]_F\ =\ Deadline(\mathsf{b}, \mathsf{x})$

$[\![\underline{(\mathrm{destroy\ on\ a})} : \texttt{DestroyOccurrence}]\!]_F\ =\ terminate\ \rightarrow\ \textbf{\textit{Skip}}$

$$
\begin{array}{l}
Guard(\text{altFrags}, \text{loopFrags}, \text{optFrags}) \;=\; \\
\quad (Evaluation(\text{altFrags}, \text{loopFrags}, \text{optFrags}) \;|[\; alphaCounters \;\cup\; \{terminate\} \,]|\; Counters) \\
\quad \setminus\; alphaCounters \\
\\
\quad \textbf{\textit{channel}}\; getCount : \; ID.Int.Int \\
\quad \textbf{\textit{channel}}\; setCount : \; ID.Int.Int \\
\quad alphaCounters \;=\; \{getCount, setCount\} \\
\\
\quad Counter\_\mathsf{ID\_x}(count) \;=\; getCount.\mathsf{ID.x}!count \;\rightarrow\; Counter\_\mathsf{ID\_x}(count) \\
\qquad \Box\; setCount.\mathsf{ID.x}?y \;\rightarrow Counter\_\mathsf{ID\_x}(y) \\
\qquad \Box\; terminate \;\rightarrow \textbf{\textit{Skip}} \\
\quad Counters \;=\; (\Big\|\; \mathsf{ID} : IDs, \mathsf{x} : \{0..n\} \;\bullet\; \{terminate\} \;\circ\; Counter\_\mathsf{ID\_x}(0)) \\
\\
\quad Evaluation(\text{altFrags}, \text{loopFrags}, \text{optFrags}) \;= \\
\qquad reset\_counters \;\fatsemi\; guards\_response(\text{altFrags}, \text{loopFrags}, \text{optFrags}) \\
\quad reset\_counters \;=\; \fatsemi\, \mathsf{ID} : IDs, \mathsf{x} : \{0..n\} \;\bullet\; setCount.\mathsf{ID.x}!0 \;\rightarrow\; \textbf{\textit{Skip}} \\
\\
\quad guards\_response(\text{altFrags}, \text{loopFrags}, \text{optFrags}) \;= \\
\qquad \Box\; alt?ID\_ALT.\mathsf{id} \;\rightarrow\; (\llbracket \mathsf{altFrag} : \texttt{XAltFragment} \rrbracket_{\mathrm{AF}} : \mathsf{altFrags} \;\bullet \\
\qquad (\mathsf{id} = \text{id}(\text{altFrag}))\&(\text{alt\_eval}(\llbracket \mathsf{altFrag} \rrbracket_{\mathrm{AF}}))) \;\fatsemi\; guards\_response \\
\qquad \Box\; opt?ID\_OPT.\mathsf{id} \;\rightarrow\; (\llbracket \mathsf{optFrag} : \texttt{OptFragment} \rrbracket_{\mathrm{OF}} : \mathsf{optFrags} \;\bullet \\
\qquad (\mathsf{id} = \text{id}(\text{optFrag}))\&(\text{opt\_eval}(\llbracket \mathsf{optFrag} \rrbracket_{\mathrm{OF}}))) \;\fatsemi\; guards\_response \\
\qquad \Box\; loop?ID\_LOOP.\mathsf{id} \;\rightarrow\; (\llbracket \mathsf{loopFrag} : \texttt{LoopFragment} \rrbracket_{\mathrm{LF}} : \mathsf{loopFrags} \;\bullet \\
\qquad (\mathsf{id} = \text{id}(\text{loopFrag}))\&(\text{loop\_eval}(\llbracket \mathsf{loopFrag} \rrbracket_{\mathrm{LF}}))) \;\fatsemi\; guards\_response \\
\qquad \Box\; terminate \;\rightarrow \textbf{\textit{Skip}} \\
\\
\quad \text{alt\_eval}(\mathsf{altFrag}) \;\widehat{=} \\
\qquad (\text{variable} : \text{getFragmentGuardVariables}(\llbracket \mathsf{altFrag} \rrbracket_{\mathrm{AF}}) \;\bullet \\
\qquad (\text{get\_}\{\text{variable}\} \;\rightarrow))\; (\llbracket \mathsf{altFrag} \rrbracket_{\mathrm{G}})) \\
\\
\quad alt\_(\text{a}, \text{x}) \;=\; \llbracket \mathsf{altFrag} \rrbracket_{AF} \\
\\
\quad \text{opt\_eval}((\mathsf{optFrag}) \;\widehat{=} \\
\qquad (\text{variable} : \text{getFragmentGuardVariables}(\llbracket \mathsf{optFrag} \rrbracket_{\mathrm{OF}}) \;\bullet \\
\qquad (\text{get\_}\{\text{variable}\} \;\rightarrow))\; (\llbracket \mathsf{optFrag} \rrbracket_{\mathrm{G}})) \\
\\
\quad opt\_(\text{a}, \text{x}) \;=\; \llbracket \mathsf{optFrag} \rrbracket_{OF} \\
\\
\quad \text{loop\_eval}((\mathsf{loopFrag}) \;\widehat{=} \\
\qquad (\text{variable} : \text{getFragmentGuardVariables}(\llbracket \mathsf{loopFrag} \rrbracket_{\mathrm{LF}}) \;\bullet \\
\qquad (\text{get\_}\{\text{variable}\} \;\rightarrow))\; (\llbracket \mathsf{loopFrag} \rrbracket_{\mathrm{G}})) \\
\\
\quad loop\_(\text{a}, \text{x}) \;=\; \llbracket \mathsf{loopFrag} \rrbracket_{LF}
\end{array}
$$

$[\![\mathsf{altFrag} : \texttt{XAltFragment}]\!]^F_{(a)} = alt\_(\mathrm{a}, \underline{\mathrm{id}(\mathsf{altFrag})})$

$[\![\mathsf{optFrag} : \texttt{OptFragment}]\!]^F_{(a)} = opt\_(\mathrm{a}, \underline{\mathrm{id}(\mathsf{optFrag})})$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]^F_{(a)} = loop\_(\mathrm{a}, \underline{\mathrm{id}(\mathsf{loopFrag})})$

$[\![\mathsf{strFrag} : \texttt{StrFragment}]\!]_F = str.ID\_STR.\underline{\mathrm{id}(\mathsf{strFrag})} \rightarrow \boldsymbol{Skip}$

$[\![\mathsf{parFrag} : \texttt{ParFragment}]\!]_F = par.ID\_PAR.\underline{\mathrm{id}(\mathsf{parFrag})} \rightarrow \boldsymbol{Skip}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$\quad not(\mathsf{condition})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad ((count < \mathsf{min})and\ \mathsf{condition})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count == \mathsf{max})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (not(count < \mathsf{min})\ and\ not(count >= \mathsf{max})and\ \mathsf{condition})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad ((count >= \mathsf{min})\ and\ (count < \mathsf{max})\ and\ \mathsf{condition})\&$
$\quad ((setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip}) \sqcap$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}))),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop}(\mathsf{min}, \mathsf{max})[\mathsf{condition}])} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$\quad getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count\ \rightarrow ($
$\quad (count < \mathsf{min})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count == \underline{\mathrm{defaultMax}})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (not(count < \mathsf{min})\ and\ not(count >= \underline{\mathrm{defaultMax}}))\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count >= \mathsf{min}\ and\ count < \underline{\mathrm{defaultMax}})\&$
$\quad ((setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip}) \sqcap$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}))),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop}(\mathsf{min}))} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count \ \rightarrow ($
$not(\mathsf{condition})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\square$
$(count == \underline{\mathrm{defaultMax}})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\square$
$(not(count >= \underline{\mathrm{defaultMax}}) \ and \ \mathsf{condition} == true)\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop}[\mathsf{condition}])} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count \ \rightarrow ($
$not(\mathsf{condition})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\square$
$((count < \mathsf{min})and \ \mathsf{condition})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\square$
$(count == \underline{\mathrm{defaultMax}})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\square$
$(not(count < \mathsf{min}) \ and \ not(count >= \underline{\mathrm{defaultMax}}) \ and \ \mathsf{condition})\&$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\square$
$((count >= \mathsf{min}) \ and \ (count < \underline{\mathrm{defaultMax}}) \ and \ \mathsf{condition})\&$
$((setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip}) \sqcap$
$(setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}))),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop}(\mathsf{min})[\mathsf{condition}])} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$\quad getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count \ \rightarrow ($
$\quad (count < \mathsf{min})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count == \mathsf{max})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (not(count < \mathsf{min}) \ and \ not(count >= \mathsf{max}))\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count >= \mathsf{min} \ and \ count < \mathsf{max})\&$
$\quad ((setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip}) \sqcap$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}))),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop}(\mathsf{min}, \mathsf{max}))} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_G =$
$\quad getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count \ \rightarrow ($
$\quad (count < \underline{\mathrm{defaultMin}})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count == \underline{\mathrm{defaultMax}})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (not(count < \underline{\mathrm{defaultMin}}) and \ not(count >= \underline{\mathrm{defaultMax}}))\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})$
$\quad \Box$
$\quad (count >= \underline{\mathrm{defaultMin}} \ and \ count < \underline{\mathrm{defaultMax}})\&$
$\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow$
$\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})),$
$\underline{\text{if}}\ \mathsf{loopFrag} \in [\![\underline{(\mathrm{loop})} : \texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{optFrag} : \texttt{OptFragment}]\!]_G =$
$(\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_OPT.\underline{\mathrm{id}(\mathsf{optFrag})}.1!true)$
$\Box$
$not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}))\&$
$(guard.ID\_OPT.\underline{\mathrm{id}(\mathsf{optFrag})}.1!false)$
$\underline{\text{if}}\ \mathsf{optFrag} \in [\![\underline{(\text{opt}\ [\mathsf{g_0}\&\mathsf{g_1}\&...\&\mathsf{g_n}]\ \mathsf{x_1}\ \text{end})} : \texttt{OptFragment}]\!]_{OF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G =$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?true \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.2?false \rightarrow$
$... \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false)$
$\sqcap$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?false \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.2?true \rightarrow$
$... \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false)$
$\sqcap$
...
$\sqcap$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?false \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.2?false \rightarrow$
$... \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true)$
$\sqcap$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?false \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.2?false \rightarrow$
$... \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false),$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt}\ \mathrm{x_1}\ \text{else}\ \mathrm{x_2}\ \text{else}\ ...\ \mathrm{x_n}\ \text{end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G =$
$(\mathsf{g_0}\&\mathsf{g_1}\&...\&\mathsf{g_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true)$
$\Box$
$not((\mathsf{g_0}\&\mathsf{g_1}\&...\&\mathsf{g_n}))\&$
$(guard.ID\_ALT.id([\![\mathsf{altFrag}]\!]_{AF}).1!false),$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt}\ [\mathsf{g_0}\&\mathsf{g_1}\&...\&\mathsf{g_n}]\ \mathrm{x_1}\ \text{end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G =$
$(\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true)$
$\Box$
$not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}))\&$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false),$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt}\ [\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}]\ \mathrm{x_1}\ \text{else}\ \mathrm{x_2}\ \text{end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G$ =
$(\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true \rightarrow ...$
$\rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false)$
$\Box$
...
$\Box$
$(\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \rightarrow ...$
$\rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true)$
$\Box$
$not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\ or\ ...\ or\ (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}))\&$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \rightarrow ...$
$\rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true),$
if $\mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}] \text{ x}_1 \text{ else ... else } [\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}] \text{ x}_\text{n} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G$ =
$(\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true \rightarrow ...$
$\rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false)$
$\Box$
...
$\Box$
$(\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \rightarrow ...$
$\rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true)$
$\Box$
$not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\ or\ ...\ or\ (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}))\&$
$(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \rightarrow ... \rightarrow$
$guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true),$
if $\mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}] \text{ x}_1 \text{ else ... else } [\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}] \text{ x}_\text{n} \text{ else x}_\text{m} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{loopFrag} : \texttt{LoopFragment}]\!]_{LF}$ =
$loop.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})} \rightarrow guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1?id1 \rightarrow$
$(id1\&([\![x_1]\!]_P) \ \Box \ not(id1)\&(\boldsymbol{Skip}))$
if $\mathsf{loopFrag} \in [\![\texttt{LoopFragment}]\!]_{LF}$

$[\![\mathsf{optFrag} : \texttt{OptFragment}]\!]_{OF}$ =
$opt.ID\_OPT.\underline{\mathrm{id}(\mathsf{optFrag})} \rightarrow guard.ID\_OPT.\underline{\mathrm{id}(\mathsf{optFrag})}.1?id1 \rightarrow$
$(id1\&([\![x_1]\!]_P) \ \Box \ not(id1)\&(\boldsymbol{Skip}))$
if $\mathsf{optFrag} \in [\![\texttt{OptFragment}]\!]_{OF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} =$
$\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1 \rightarrow$
$\quad\quad guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.2?id2 \rightarrow \ldots guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n?idn$
$\quad\quad \rightarrow (id1\&([\![\mathsf{x_1}]\!]_P) \ \Box \ id2\&([\![\mathsf{x_2}]\!]_P) \ \Box \ \ldots \ idn\&([\![\mathsf{x_n}]\!]_P))$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } \mathsf{x_1} \text{ else } \mathsf{x_2} \text{ else } \ldots \mathsf{x_n} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{OptFragment}]\!]_{AF} =$
$\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1$
$\quad\quad \rightarrow (id1\&([\![\mathsf{x_1}]\!]_P) \ \Box \ not(id1)\&(\boldsymbol{Skip}))$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g}_0\&\mathrm{g}_1\&\ldots\&\mathrm{g_n}] \ \mathsf{x_1} \text{ end})} : \texttt{OptFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} =$
$\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1$
$\quad\quad \rightarrow (id1\&([\![\mathsf{x_1}]\!]_P) \ \Box \ not(id1)\&([\![\mathsf{x_2}]\!]_P))$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1}_0\&\mathrm{g\_1}_1\&\ldots\&\mathrm{g\_1_n}] \ \mathsf{x_1} \text{ else } \mathsf{x_2} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} =$
$\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1 \rightarrow \ldots$
$\quad\quad \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n?idn$
$\quad\quad \rightarrow (id1\&([\![\mathsf{x_1}]\!]_P) \ \Box \ \ldots \ idn\&([\![\mathsf{x_n}]\!]_P)$
$\quad\quad \Box \ not(id1 \ or \ \ldots \ or \ idn)\&(\boldsymbol{Skip}))$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1}_0\&\mathrm{g\_1}_1\&\ldots\&\mathrm{g\_1_n}] \ \mathsf{x_1} \text{ else } \ldots \text{ else}}$
$\underline{[\mathrm{g\_n}_0\&\mathrm{g\_n}_1\&\ldots\&\mathrm{g\_n_n}] \ \mathsf{x_n} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} =$
$\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1 \rightarrow \ldots$
$\quad\quad \rightarrow guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n?idn$
$\quad\quad \rightarrow (id1\&([\![\mathsf{x_1}]\!]_P) \ \Box \ \ldots \ idn\&([\![\mathsf{x_n}]\!]_P)$
$\quad\quad \Box \ not(id1 \ or \ \ldots \ or \ idn)\&([\![\mathsf{x_m}]\!]_P))$
$\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1}_0\&\mathrm{g\_1}_1\&\ldots\&\mathrm{g\_1_n}] \ \mathsf{x_1} \text{ else } \ldots \text{ else}}$
$\underline{[\mathrm{g\_n}_0\&\mathrm{g\_n}_1\&\ldots\&\mathrm{g\_n_n}] \ \mathsf{x_n} \text{ else } \mathsf{x_m} \text{ end})} : \texttt{AltFragment}]\!]_{AF}$

$$
\begin{array}{l}
\textit{Parallel}(\underline{\mathsf{parFrags}}) \;=\; \Box\ \mathsf{parFrag} : \mathsf{parFrags} \bullet \\
\quad (\textit{par}.\textit{ID\_PAR}.\underline{\mathrm{id}([\![\mathsf{parFrag} : \texttt{ParFragment}]\!]_{\mathrm{PF}})} \rightarrow \\
\quad \Big\|\ \mathsf{thread} : \mathsf{parFrag}.\underline{\mathrm{threads}} \bullet \textit{parallel\_}(\underline{\mathrm{id}([\![\mathsf{parFrag}]\!]_{\mathrm{PF}})}, \underline{\mathrm{id}(\mathrm{varsthread})}) \\
\\
\quad \textit{parallel\_}(\mathsf{id}, \mathsf{threadID}) = \Big\|\ \mathsf{actor} : [\![\mathsf{parFrag}]\!]_{PF}.\underline{\mathsf{threadID}.\mathrm{actors}} \bullet \\
\qquad \textit{parallel\_}(\mathsf{actor}, \mathsf{id}, \mathsf{threadID}) \\
\quad \textit{parallel\_}(\mathsf{a}, \textit{id}, \mathsf{threadID}) = {}^{\mathfrak{g}}_{\mathfrak{g}}\ \mathsf{frag} : [\![\mathsf{parFrag}]\!]_{PF}.\underline{\mathsf{threadID}.\mathrm{interactionFragments}} \bullet [\![\mathsf{frag}]\!]^F_{(\mathsf{a})}
\end{array}
$$

$$
\begin{array}{l}
\textit{Strict}(\underline{\mathsf{strFrags}}) \;=\; \Box\ \mathsf{strFrag} : \mathsf{strFrags} \bullet \\
\quad (\textit{str}.\textit{ID\_STR}.\underline{\mathrm{id}([\![\mathsf{strFrag} : \texttt{StrFragment}]\!]_{\mathrm{SF}})} \rightarrow \textit{strict\_}(\underline{\mathrm{id}([\![\mathsf{strFrag}]\!]_{\mathrm{SF}})}) \\
\\
\quad \textit{strict\_}(\textit{id}) = {}^{\mathfrak{g}}_{\mathfrak{g}}\ \textit{interactionFrags} : [\![\mathsf{strFrag}]\!]_{SF}.\underline{\mathrm{interactionFragments}}
\end{array}
$$

$$
[\![\underline{(\mathrm{ignore}(\mathsf{m_1}, \mathsf{m_2}, ..., \mathsf{m_n})\ \mathsf{b})} : \texttt{IgnoreFragment}]\!]^F_a \;=\; \textit{ignore\_}(\mathrm{a}, \underline{\mathrm{id}(\mathsf{ignoreFrag})})
$$

$$
\textit{ignore\_}(\underline{\mathrm{a}, \mathrm{x}}) \;=\; (\mathsf{b}) \setminus \{\mathsf{m_1}, \mathsf{m_2}, ..., \mathsf{m_n}\}
$$

$$
[\![\underline{(\mathrm{consider}(\mathsf{m_1}, \mathsf{m_2}, ..., \mathsf{m_n})\ \mathsf{b})} : \texttt{IgnoreFragment}]\!]^F_a \;=\; \textit{consider\_}(\mathrm{a}, \underline{\mathrm{id}(\mathsf{considerFrag})})
$$

$$
\textit{consider\_}(\underline{\mathrm{a}, \mathrm{x}}) \;=\; (\mathsf{b}) \setminus \{\underline{\mathrm{ALL}} \setminus \{\mathsf{m_1}, \mathsf{m_2}, ..., \mathsf{m_n}\}\}
$$

$$
\begin{array}{l}
[\![(\mathsf{a_1} -> \mathsf{a_2} : \mathsf{mID!var}) : \texttt{InteractionFragment}]\!]_F \;\widehat{=} \\
\quad \mathsf{a_1} = \mathsf{mID!var} \rightarrow \textbf{\textit{Skip}} \\
\quad \mathsf{a_2} = \mathsf{mID?var} \rightarrow \textit{set\_var!var} \rightarrow \textbf{\textit{Skip}} \\
[\![(\mathsf{a_1} -> \mathsf{a_2} : \mathsf{mID?var}) : \texttt{InteractionFragment}]\!]_F \;\widehat{=} \\
\quad \mathsf{a_1} = \mathsf{mID?var} \rightarrow \textit{set\_var!var} \rightarrow \textbf{\textit{Skip}} \\
\quad \mathsf{a_2} = \mathsf{mID!var} \rightarrow \textbf{\textit{Skip}} \\
[\![(\mathsf{a_1} -> \mathsf{a_1} : \mathsf{mID?var}) : \texttt{InteractionFragment}]\!]_F \;\widehat{=} \\
\quad \mathsf{a_1} = \mathsf{mID?var} \rightarrow \textit{set\_var!var} \rightarrow \textbf{\textit{Skip}}
\end{array}
$$

$$
\begin{array}{l}
[\![(\mathsf{a_1} -> \mathsf{a_2} : \mathsf{mID}) : \texttt{InteractionFragment}]\!]_F \;\widehat{=} \\
\quad \mathsf{a_1} = \mathsf{mID} \rightarrow \textbf{\textit{Skip}} \\
\quad \mathsf{a_2} = \mathsf{mID} \rightarrow \textbf{\textit{Skip}}
\end{array}
$$

Key

$[\![]\!]_{D} ->$ `RoboHumansDocument`

$[\![]\!]_{CS} ->$ `Capabilities`

$[\![]\!]_{SDS} ->$ `SequenceDiagramGroup`

$[\![]\!]_{SD} ->$ *An instance of* `SequenceDiagramGroup`

$[\![]\!]_{CO} ->$ `Constants`

$[\![]\!]_{LS} ->$ `LifelineBlock`

$[\![]\!]_{TU} ->$ `TimeUnit`

$[\![]\!]_{CONS} ->$ `ConstAssignment`

$[\![]\!]_{CON} ->$ *An instance of* `ConstAssignment`

$[\![]\!]_{OS} ->$ `Out`

$[\![]\!]_{O} ->$ *An instance of* `Out`

$[\![]\!]_{IS} ->$ `In`

$[\![]\!]_{I} ->$ *An instance of* `In`

$[\![]\!]_{VS} ->$ `VarsList`

$[\![]\!]_{V} ->$ *An instance of* `VarsList`

$[\![]\!]_{TO} ->$ *An* `Out LifelineEvent` *instance with a type*

$[\![]\!]_{TI} ->$ *An* `In LifelineEvent` *instance with a type*

$[\![]\!]_{SS} ->$ `Sequence`

$[\![]\!]_{S} ->$ *An instance of* `Sequence`

$[\![]\!]_{A} ->$ `Actor`

$[\![]\!]_{FS} ->$ `InteractionFragment`

$[\![]\!]_{F} ->$ *An instance of* `InteractionFragment`

$[\![]\!]^{F}_{(a)} ->$ *An instance of* `InteractionFragment` *for* `Actor` *a*

$[\![]\!]_{G} ->$ *The body of the evaluation for an* `InteractionFragment` *with a* `Guard`

$[\![]\!]_{AF} ->$ *An* `AltFragment`

$[\![]\!]_{OF} ->$ *An* `OptFragment`

$[\![]\!]_{LF} ->$ *A* `LoopFragment`

$[\![]\!]_{PF} ->$ *A* `ParFragment`

$[\![]\!]_{SF} ->$ *A* `StrFragment`

$[\![]\!]_{P} ->$ *An operand inside an* `InteractionFragment` *with a* `Guard`