RoboScene to CSP Translation Rules

Holly Hendry

January 22, 2025

$$\begin{array}{l}
[\![\mathsf{rsm} : \texttt{RoboSceneModel}]\!]_M \\
\quad \widehat{=}\ \ [\![\mathsf{rsm.constant}]\!]_{CO}\ [\![\mathsf{rsm.caps}]\!]_C\ [\![\mathsf{rsm.seqdgrmgrps}]\!]_{SDG}\ [\![\mathsf{rsm.types}]\!]_{TY}
\end{array}$$

$$[\![\mathsf{const} : \texttt{Constants}]\!]_{CO}\ \ \widehat{=}\ \ [\![\mathsf{const.timeunit}]\!]_{TU}\ [\![\mathsf{const.constants}]\!]_{CON}$$

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

$$[\![\mathsf{caps} : \texttt{Capabilities}]\!]_C\ \ \widehat{=}\ \ [\![\mathsf{caps.actorblock}]\!]_{AB}$$

$$\begin{array}{l}
[\![\mathsf{outs}, \mathsf{ins} : \texttt{LifelineEvents};\ \mathsf{vars} : \texttt{VariableList};\ \mathsf{ab} : \texttt{ActorBlock}]\!]_{AB} \\
\quad \widehat{=}\ \ [\![\mathsf{outs}]\!]_O\ [\![\mathsf{ins}]\!]_I\ [\![\mathsf{vars}]\!]_V \underline{(\text{if}}\ \mathsf{ab} \underline{\neq\ \ \text{null then}}\ [\![\mathsf{ab}]\!]_{AB} \underline{)}
\end{array}$$

$$\begin{array}{l}
[\![(\underline{\text{out}}\ \mathsf{event}- > \mathsf{actor}) :\ \texttt{LifelineEvents}]\!]_O\ \ \widehat{=}\ \ \boldsymbol{channel}\ \mathsf{event} : \underline{\text{InOut}} \\
[\![(\underline{\text{out}}\ \mathsf{event} : \mathsf{Type}- > \mathsf{actor}) :\ \texttt{LifelineEvents}]\!]_O \\
\quad \widehat{=}\ \ \boldsymbol{channel}\ \mathsf{event} : [\![\mathsf{Type}, \mathsf{event}]\!]_{TO} \\
[\![(\underline{\text{in}}\ \mathsf{event} < -\mathsf{actor}) :\ \texttt{LifelineEvents}]\!]_I\ \ \widehat{=}\ \ \boldsymbol{channel}\ \mathsf{event} : \underline{\text{InOut}} \\
[\![(\underline{\text{in}}\ \mathsf{event} : \mathsf{Type} < -\mathsf{actor}) :\ \texttt{LifelineEvents}]\!]_I \\
\quad \widehat{=}\ \ \boldsymbol{channel}\ \mathsf{event} : [\![\mathsf{Type}, \mathsf{event}]\!]_{TI}
\end{array}$$

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

$$[\![\mathsf{s} : \mathtt{Sequence};\ \mathsf{sdg} : \mathtt{SequenceDiagramGroup}]\!]_{SDG} = [\![s]\!]_S\ (\mathrm{if}\ \mathsf{sdg} \neq\ \mathrm{null\ then}\ [\![\mathsf{sdq}]\!]_{SDG})$$

$$\begin{array}{l}
[\![\mathsf{s} : \mathtt{Sequence}]\!]_S \ \widehat{=} \\
\quad ((((\Big\|\ \mathrm{a}\ :\ \mathsf{actors}\ \bullet\ \mathrm{alpha(a)}\ \circ\ \mathrm{lifeline(a)}) \\
\quad |[\ \{str, par, terminate\}\ ]|\ \ Control(\mathrm{parFrags}, \mathrm{strFrags})) \\
\quad |[\ \{alt, opt, loop, guard, terminate\}\ ]|\ \ Guard(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})) \\
\quad |[\ sharedVars\ \cup\ \{terminate\}\,]|\ \ Memory)\ \setminus\ \{alt, opt, loop, guard, str, par\}
\end{array}$$

$$\begin{array}{l}
\mathrm{alpha(a)}\ =\ \alpha(a_x, \mathsf{frags})\ \cap\ \alpha(a_{x+1}, \mathsf{frags}) \\
\mathrm{lifeline(a)}\ =\ [\![\mathsf{frags}_1]\!]^F_{(a)}\ \fatsemi\ \ldots\ \fatsemi\ [\![\mathsf{frags}_n]\!]^F_{(a)} \\
Control(\mathrm{parFrags}, \mathrm{strFrags})\ =\ Parallel(\mathrm{parFrags})\,|[\ \{terminate\}\ ]|\ \ Strict(\mathrm{strFrags})
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{channel}}\ alt:\ IDs.Int \\
\textbf{\textit{channel}}\ opt:\ IDs.Int \\
\textbf{\textit{channel}}\ loop:\ IDs.Int \\
\textbf{\textit{channel}}\ par:\ IDs.Int \\
\textbf{\textit{channel}}\ str:\ IDs.Int \\
\textbf{\textit{channel}}\ guard:\ IDs.Int.Int.Bool \\
datatype\ IDs\ =\ ID\_ALT\ |\ ID\_OPT\ |\ ID\_LOOP\ |\ ID\_PAR\ |\ ID\_STR
\end{array}$$

$$\begin{array}{l}
Guard(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})\ = \\
\quad (Evaluation(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})\ |[\ alphaCounters\ \cup\ \{terminate\}\,]|\ \ Counters) \\
\quad \setminus\ alphaCounters
\end{array}$$

$$\begin{array}{l}
\textbf{\textit{channel}}\ getCount:\ ID.Int.Int \\
\textbf{\textit{channel}}\ setCount:\ ID.Int.Int \\
alphaCounters\ =\ \{getCount, setCount\}
\end{array}$$

$$\begin{array}{l}
Counter\_\mathsf{ID}\_\mathsf{x}(count)\ =\ getCount.\mathsf{ID}.\mathsf{x}!count\ \to\ Counter\_\mathsf{ID}\_\mathsf{x}(count) \\
\quad \Box\ setCount.\mathsf{ID}.\mathsf{x}?y\ \to Counter\_\mathsf{ID}\_\mathsf{x}(y) \\
\quad \Box\ terminate\ \to \textbf{\textit{Skip}} \\
Counters\ =\ (\Big\|\ \mathsf{ID} : IDs, \mathsf{x} : \{0..n\}\ \bullet\ \{terminate\}\ \circ\ Counter\_\mathsf{ID}\_\mathsf{x}(0))
\end{array}$$

$$\begin{array}{l}
Evaluation(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})\ = \\
\quad reset\_counters\ \fatsemi\ guards\_response(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags}) \\
reset\_counters\ =\ \fatsemi\ \mathsf{ID} : IDs, \mathsf{x} : \{0..n\}\ \bullet\ setCount.\mathsf{ID}.\mathsf{x}!0\ \to\ \textbf{\textit{Skip}}
\end{array}$$

$$\begin{array}{l}
guards\_response(\mathrm{altFrags}, \mathrm{loopFrags}, \mathrm{optFrags})\ = \\
\quad \Box\ alt?ID\_ALT.\mathsf{id}\ \to\ ([\![\mathsf{altFrag} : \mathtt{XAltFragment}]\!]_{\mathrm{AF}} : \mathsf{altFrags}\ \bullet \\
\quad\ \ (\mathsf{id} = \mathrm{id}(\mathrm{altFrag}))\&(\mathrm{alt\_eval}([\![\mathsf{altFrag}]\!]_{\mathrm{AF}})))\ \fatsemi\ guards\_response \\
\quad \Box\ opt?ID\_OPT.\mathsf{id}\ \to\ ([\![\mathsf{optFrag} : \mathtt{OptFragment}]\!]_{\mathrm{OF}} : \mathsf{optFrags}\ \bullet \\
\quad\ \ (\mathsf{id} = \mathrm{id}(\mathrm{optFrag}))\&(\mathrm{opt\_eval}([\![\mathsf{optFrag}]\!]_{\mathrm{OF}})))\ \fatsemi\ guards\_response \\
\quad \Box\ loop?ID\_LOOP.\mathsf{id}\ \to\ ([\![\mathsf{loopFrag} : \mathtt{LoopFragment}]\!]_{\mathrm{LF}} : \mathsf{loopFrags}\ \bullet \\
\quad\ \ (\mathsf{id} = \mathrm{id}(\mathsf{loopFrag}))\&(\mathrm{loop\_eval}([\![\mathsf{loopFrag}]\!]_{\mathrm{LF}})))\ \fatsemi\ guards\_response \\
\quad \Box\ terminate\ \to\ \textbf{\textit{Skip}}
\end{array}$$

$$\begin{aligned}
&\underline{\mathrm{alt\_eval}(\mathsf{altFrag})}\ \widehat{=}\\
&\quad \underline{(\mathrm{variable} : \mathrm{getFragmentGuardVariables}(\llbracket\mathsf{altFrag}\rrbracket_{\mathrm{AF}})\ \bullet}\\
&\quad \underline{(\mathrm{get\_}\{\mathrm{variable}\}\ \ \rightarrow))\ (\llbracket\mathsf{altFrag}\rrbracket_{\mathrm{G}}))}
\end{aligned}$$

$$\begin{aligned}
&\underline{\mathrm{opt\_eval}((\mathsf{optFrag})}\ \widehat{=}\\
&\quad \underline{(\mathrm{variable} : \mathrm{getFragmentGuardVariables}(\llbracket\mathsf{optFrag}\rrbracket_{\mathrm{OF}})\ \bullet}\\
&\quad \underline{(\mathrm{get\_}\{\mathrm{variable}\}\ \ \rightarrow))\ (\llbracket\mathsf{optFrag}\rrbracket_{\mathrm{G}}))}
\end{aligned}$$

$$\begin{aligned}
&\underline{\mathrm{loop\_eval}((\mathsf{loopFrag})}\ \widehat{=}\\
&\quad \underline{(\mathrm{variable} : \mathrm{getFragmentGuardVariables}(\llbracket\mathsf{loopFrag}\rrbracket_{\mathrm{LF}})\ \bullet}\\
&\quad \underline{(\mathrm{get\_}\{\mathrm{variable}\}\ \ \rightarrow))\ (\llbracket\mathsf{loopFrag}\rrbracket_{\mathrm{G}}))}
\end{aligned}$$

$$\begin{aligned}
&\mathit{Parallel}\underline{(\mathsf{parFrags})}\ = \Box\ \mathsf{parFrag} : \mathsf{parFrags} \bullet\\
&\quad (\mathit{par}.\overline{\mathit{ID\_PAR}}.\underline{\mathrm{id}(\llbracket\mathsf{parFrag} : \mathtt{ParFragment}\rrbracket_{\mathrm{PF}})} \rightarrow\\
&\quad \big\Vert\ \mathsf{thread} : \mathsf{parFrag}.\underline{\mathrm{threads}} \bullet \mathit{parallel\_}(\underline{\mathrm{id}(\llbracket\mathsf{parFrag}\rrbracket_{\mathrm{PF}})}, \underline{\mathrm{id}(\mathrm{varsthread})}))\\
\\
&\quad \mathit{parallel\_}(\mathsf{id}, \mathsf{threadID}) = \big\Vert\ \mathsf{actor} : \llbracket\mathsf{parFrag}\rrbracket_{PF}\underline{.\mathsf{threadID}.\mathrm{actors}} \bullet\\
&\qquad \mathit{parallel\_}(\mathsf{actor}, \mathsf{id}, \mathsf{threadID})\\
&\quad \mathit{parallel\_}(\mathsf{a}, \mathit{id}, \mathsf{threadID}) = \fatsemi\ \mathsf{frag} : \llbracket\mathsf{parFrag}\rrbracket_{PF}\underline{.\mathsf{threadID}.\mathrm{interactionFragments}} \bullet \llbracket\mathsf{frag}\rrbracket^{F}_{(\mathsf{a})}
\end{aligned}$$

$$\begin{aligned}
&\mathit{Strict}\underline{(\mathsf{strFrags})}\ = \Box\ \mathsf{strFrag} : \mathsf{strFrags} \bullet\\
&\quad (\mathit{str}.\overline{\mathit{ID\_STR}}.\underline{\mathrm{id}(\llbracket\mathsf{strFrag} : \mathtt{StrFragment}\rrbracket_{\mathrm{SF}})} \rightarrow \mathit{strict\_}(\underline{\mathrm{id}(\llbracket\mathsf{strFrag}\rrbracket_{\mathrm{SF}})}))\\
\\
&\quad \mathit{strict\_}(\mathit{id}) = \fatsemi\ \mathit{interactionFrags} : \llbracket\mathsf{strFrag}\rrbracket_{SF}\underline{.\mathrm{interactionFragments}}
\end{aligned}$$

$$\begin{aligned}
&\underline{\mathrm{where}}\ \mathsf{actors} = \mathsf{s.actor}\ \underline{\mathrm{and}}\ \mathsf{frags} = \mathsf{s.fragment}\ \underline{\mathrm{and}}\ \mathsf{parFrags} = \underline{\mathrm{par}}(\mathsf{s.fragment})\ \underline{\mathrm{and}}\\
&\quad \mathsf{strFrags} = \underline{\mathrm{str}}(\mathsf{s.fragment})\ \underline{\mathrm{and}}\ \mathsf{altFrags} = \underline{\mathrm{alt}}(\mathsf{s.fragment})\ \underline{\mathrm{and}}\\
&\quad \mathsf{optFrags} = \underline{\mathrm{opt}}(\mathsf{s.fragment})\ \underline{\mathrm{and}}\ \mathsf{loopFrags} = \underline{\mathrm{loop}}(\mathsf{s.fragment})
\end{aligned}$$

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

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

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

$$[\![\mathsf{parFrag} : \texttt{ParFragment}]\!]_{F} = par.ID\_PAR.\underline{\mathrm{id}(\mathsf{parFrag})} \rightarrow \boldsymbol{Skip}$$
$$[\![\mathsf{strFrag} : \texttt{StrFragment}]\!]_{F} = str.ID\_STR.\underline{\mathrm{id}(\mathsf{strFrag})} \rightarrow \boldsymbol{Skip}$$

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

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

$$
\begin{array}{l}
\llbracket \mathsf{loopFrag} : \texttt{LoopFragment} \rrbracket_{G} = \\
\quad getCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}?count \ \ \rightarrow (\\
\quad not(\mathsf{condition})\& \\
\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow \\
\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}) \\
\quad \Box \\
\quad (count == \underline{\mathrm{defaultMax}})\& \\
\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!0 \rightarrow \\
\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!false \rightarrow \boldsymbol{Skip}) \\
\quad \Box \\
\quad (not(count >= \underline{\mathrm{defaultMax}}) \ and \ \mathsf{condition} == true)\& \\
\quad (setCount.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}!(count + 1) \rightarrow \\
\quad guard.ID\_LOOP.\underline{\mathrm{id}(\mathsf{loopFrag})}.1!true \rightarrow \boldsymbol{Skip})), \\
\underline{\mathrm{if}} \ \mathsf{loopFrag} \in \llbracket \underline{(\mathrm{loop}[\mathsf{condition}])} : \texttt{LoopFragment} \rrbracket_{LF}
\end{array}
$$

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

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

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

$$
\begin{array}{l}
[\![\mathsf{optFrag} : \texttt{OptFragment}]\!]_G \ = \\
\quad (\mathsf{g\_1}_0 \& \mathsf{g\_1}_1 \& ... \& \mathsf{g\_1}_\mathsf{n}) \& (\mathit{guard}.\mathit{ID\_OPT}.\underline{\mathrm{id}(\mathsf{optFrag})}.1!\mathit{true}) \\
\quad \Box \\
\quad \mathit{not}((\mathsf{g\_1}_0 \& \mathsf{g\_1}_1 \& ... \& \mathsf{g\_1}_\mathsf{n})) \& \\
\quad (\mathit{guard}.\mathit{ID\_OPT}.\underline{\mathrm{id}(\mathsf{optFrag})}.1!\mathit{false}) \\
\underline{\mathrm{if}}\ \mathsf{optFrag} \in [\![\underline{(\mathrm{opt}\ [\mathsf{g}_0 \& \mathsf{g}_1 \& ... \& \mathsf{g}_\mathsf{n}]\ \mathsf{x}_1\ \mathsf{end})} : \texttt{OptFragment}]\!]_{OF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G \ = \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1?\mathit{true} \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.2?\mathit{false} \ \rightarrow \\
\quad ... \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.n!\mathit{false}) \\
\quad \sqcap \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1?\mathit{false} \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.2?\mathit{true} \ \rightarrow \\
\quad ... \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.n!\mathit{false}) \\
\quad \sqcap \\
\quad ... \\
\quad \sqcap \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1?\mathit{false} \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.2?\mathit{false} \ \rightarrow \\
\quad ... \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.n!\mathit{true}) \\
\quad \sqcap \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1?\mathit{false} \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.2?\mathit{false} \ \rightarrow \\
\quad ... \ \rightarrow \ \mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.n!\mathit{false}), \\
\underline{\mathrm{if}}\ \mathsf{altFrag} \in [\![\underline{(\mathrm{alt}\ \mathrm{x}_1\ \mathrm{else}\ \mathrm{x}_2\ \mathrm{else}\ ...\ \mathrm{x}_\mathrm{n}\ \mathrm{end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G \ = \\
\quad (\mathsf{g}_0 \& \mathsf{g}_1 \& ... \& \mathsf{g}_\mathsf{n}) \& (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1!\mathit{true}) \\
\quad \Box \\
\quad \mathit{not}((\mathsf{g}_0 \& \mathsf{g}_1 \& ... \& \mathsf{g}_\mathsf{n})) \& \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\mathit{id}([\![\mathsf{altFrag}]\!]_{AF}).1!\mathit{false}), \\
\underline{\mathrm{if}}\ \mathsf{altFrag} \in [\![\underline{(\mathrm{alt}\ [\mathsf{g}_0 \& \mathsf{g}_1 \& ... \& \mathsf{g}_\mathsf{n}]\ \mathrm{x}_1\ \mathrm{end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_G \ = \\
\quad (\mathsf{g\_1}_0 \& \mathsf{g\_1}_1 \& ... \& \mathsf{g\_1}_\mathsf{n}) \& (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1!\mathit{true}) \\
\quad \Box \\
\quad \mathit{not}((\mathsf{g\_1}_0 \& \mathsf{g\_1}_1 \& ... \& \mathsf{g\_1}_\mathsf{n})) \& \\
\quad (\mathit{guard}.\mathit{ID\_ALT}.\underline{\mathrm{id}(\mathsf{altFrag})}.1!\mathit{false}), \\
\underline{\mathrm{if}}\ \mathsf{altFrag} \in [\![\underline{(\mathrm{alt}\ [\mathsf{g\_1}_0 \& \mathsf{g\_1}_1 \& ... \& \mathsf{g\_1}_\mathsf{n}]\ \mathrm{x}_1\ \mathrm{else}\ \mathrm{x}_2\ \mathrm{end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \mathtt{AltFragment}]\!]_{G} \ = \\
\quad (\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true \ \rightarrow \ ... \\
\quad \ \rightarrow \ guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false) \\
\quad \ \Box \\
\quad ... \\
\quad \ \Box \\
\quad (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \ \rightarrow \ ... \\
\quad \ \rightarrow \ guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true) \\
\quad \ \Box \\
\quad not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}) \ or \ ... \ or \ (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}))\& \\
\quad (guard.ID\_ALT.\underline{\underline{\mathrm{id}(\mathsf{altFrag})}}.1!false \ \rightarrow \ ... \\
\quad \ \rightarrow \ guard.ID\_A\overline{LT.\mathrm{id}(\mathsf{altFrag}}).n!true), \\
\underline{\mathrm{if}} \ \mathsf{altFrag} \in [\![\underline{(\mathrm{alt} \ [\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}] \ \mathrm{x_1} \ \mathrm{else} \ ... \ \mathrm{else}} \\
\quad \underline{[\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}] \ \mathrm{x_n} \ \mathrm{end})} : \mathtt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \mathtt{AltFragment}]\!]_{G} \ = \\
\quad (\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!true \ \rightarrow \ ... \\
\quad \ \rightarrow \ guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!false) \\
\quad \ \Box \\
\quad ... \\
\quad \ \Box \\
\quad (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n})\&(guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \ \rightarrow \ ... \\
\quad \ \rightarrow \ guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true) \\
\quad \ \Box \\
\quad not((\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}) \ or \ ... \ or \ (\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}))\& \\
\quad (guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1!false \ \rightarrow \ ... \ \rightarrow \\
\quad guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n!true), \\
\underline{\mathrm{if}} \ \mathsf{altFrag} \in [\![\underline{(\mathrm{alt} \ [\mathsf{g\_1_0}\&\mathsf{g\_1_1}\&...\&\mathsf{g\_1_n}] \ \mathrm{x_1} \ \mathrm{else} \ ... \ \mathrm{else}} \\
\quad \underline{[\mathsf{g\_n_0}\&\mathsf{g\_n_1}\&...\&\mathsf{g\_n_n}] \ \mathrm{x_n} \ \mathrm{else} \ \mathrm{x_m} \ \mathrm{end})} : \mathtt{AltFragment}]\!]_{AF}
\end{array}
$$

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

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

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

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{OptFragment}]\!]_{AF} \;=\\
\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \;\rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1\\
\qquad \rightarrow\; (id1\&([\![\mathsf{x_1}]\!]_P) \;\Box\; not(id1)\&(\boldsymbol{Skip}))\\
\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g_0}\&\mathrm{g_1}\&...\&\mathrm{g_n}]\ \mathsf{x_1} \text{ end})} : \texttt{OptFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} \;=\\
\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \;\rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1\\
\qquad \rightarrow\; (id1\&([\![\mathsf{x_1}]\!]_P) \;\Box\; not(id1)\&([\![\mathsf{x_2}]\!]_P))\\
\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1_0}\&\mathrm{g\_1_1}\&...\&\mathrm{g\_1_n}]\ \mathsf{x_1} \text{ else } \mathsf{x_2} \text{ end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} \;=\\
\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \;\rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1 \;\rightarrow\; ...\\
\qquad \rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n?idn\\
\qquad \rightarrow\; (id1\&([\![\mathsf{x_1}]\!]_P) \;\Box\; ...\; idn\&([\![\mathsf{x_n}]\!]_P)\\
\qquad \Box\; not(id1 \ or\ ...\ or\ idn)\&(\boldsymbol{Skip}))\\
\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1_0}\&\mathrm{g\_1_1}\&...\&\mathrm{g\_1_n}]\ \mathsf{x_1} \text{ else ... else}}\\
\quad \underline{[\mathrm{g\_n_0}\&\mathrm{g\_n_1}\&...\&\mathrm{g\_n_n}]\ \mathsf{x_n} \text{ end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

$$
\begin{array}{l}
[\![\mathsf{altFrag} : \texttt{AltFragment}]\!]_{AF} \;=\\
\quad alt.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})} \;\rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.1?id1 \;\rightarrow\; ...\\
\qquad \rightarrow\; guard.ID\_ALT.\underline{\mathrm{id}(\mathsf{altFrag})}.n?idn\\
\qquad \rightarrow\; (id1\&([\![\mathsf{x_1}]\!]_P) \;\Box\; ...\; idn\&([\![\mathsf{x_n}]\!]_P)\\
\qquad \Box\; not(id1 \ or\ ...\ or\ idn)\&([\![\mathsf{x_m}]\!]_P))\\
\underline{\text{if}}\ \mathsf{altFrag} \in [\![\underline{(\text{alt } [\mathrm{g\_1_0}\&\mathrm{g\_1_1}\&...\&\mathrm{g\_1_n}]\ \mathsf{x_1} \text{ else ... else}}\\
\quad \underline{[\mathrm{g\_n_0}\&\mathrm{g\_n_1}\&...\&\mathrm{g\_n_n}]\ \mathsf{x_n} \text{ else } \mathsf{x_m} \text{ end})} : \texttt{AltFragment}]\!]_{AF}
\end{array}
$$

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

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

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

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

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

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

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

$$\begin{aligned}
&[\![(\mathsf{a_1} -> \ \mathsf{a_2} : \ \mathsf{mID!var}) : \texttt{InteractionFragment}]\!]_F \ \widehat{=} \\
&\quad \mathsf{a_1} \ = \ \mathsf{mID!var} \ \rightarrow \ \boldsymbol{\mathit{Skip}} \\
&\quad \mathsf{a_2} \ = \ \mathsf{mID?var} \ \rightarrow \ \mathit{set\_}\mathsf{var!var} \ \rightarrow \ \boldsymbol{\mathit{Skip}} \\
&[\![(\mathsf{a_1} -> \ \mathsf{a_2} : \ \mathsf{mID?var}) : \texttt{InteractionFragment}]\!]_F \ \widehat{=} \\
&\quad \mathsf{a_1} \ = \ \mathsf{mID?var} \ \rightarrow \ \mathit{set\_}\mathsf{var!var} \ \rightarrow \ \boldsymbol{\mathit{Skip}} \\
&\quad \mathsf{a_2} \ = \ \mathsf{mID!var} \ \rightarrow \ \boldsymbol{\mathit{Skip}} \\
&[\![(\mathsf{a_1} -> \ \mathsf{a_1} : \ \mathsf{mID?var}) : \texttt{InteractionFragment}]\!]_F \ \widehat{=} \\
&\quad \mathsf{a_1} \ = \ \mathsf{mID?var} \ \rightarrow \ \mathit{set\_}\mathsf{var!var} \ \rightarrow \ \boldsymbol{\mathit{Skip}}
\end{aligned}$$

$$\begin{aligned}
&[\![(\mathsf{a_1} -> \ \mathsf{a_2} : \ \mathsf{mID}) : \texttt{InteractionFragment}]\!]_F \ \widehat{=} \\
&\quad \mathsf{a_1} \ = \ \mathsf{mID} \ \rightarrow \ \boldsymbol{\mathit{Skip}} \\
&\quad \mathsf{a_2} \ = \ \mathsf{mID} \ \rightarrow \ \boldsymbol{\mathit{Skip}}
\end{aligned}$$

Key

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

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

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

$[\![]\!]_{TY} - >$ `TypeSpecification`

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

$[\![]\!]_{AB} - >$ `ActorBlock`

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

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

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

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

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

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

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

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

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

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

$[\![]\!]_{(a)}^{F} - >$ *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`